MAVIDSQL: A Model-Agnostic Visualization for Interpretation and Diagnosis of Text-to-SQL Tasks

Jingwei Tang, Guodao Sun, Jiahui Chen, Gefei Zhang, Baofeng Chang, Haixia Wang, Ronghua Liang

***Abstract*—Significant advancements in semantic parsing for text-to-SQL tasks have been achieved through the employment of neural network models, such as LSTM, BERT, and T5. The exceptional performance of large language models, like ChatGPT, has been demonstrated in recent research, even in zero-shot scenarios. However, the inherent transparency of text-to-SQL models presents them as black boxes, concealing their inner workings from both developers and users, which complicates the diagnosis of potential error patterns. Despite the fact that numerous visual analysis studies have been conducted in natural language processing communities, scant attention has been paid to addressing the challenges of semantic parsing, specifically in text-to-SQL tasks. This limitation hinders the development of effective tools for model optimization and evaluation. This paper presents an interactive visual analysis tool, MAVIDSQL, to assist model developers and users in understanding and diagnosing text-to-SQL tasks. The system comprises three modules: the Model Manager, the Feature Extractor, and the Visualization Interface, which adopt a model-agnostic approach to diagnose potential errors and infer model decisions by analyzing input-output data, facilitating interactive visual analysis to identify error patterns and assess model performance. Two case studies and interviews with domain experts demonstrate the effectiveness of MAVIDSQL in facilitating the understanding of text-to-SQL Tasks and identifying potential errors.**

***Index Terms*—Text-to-SQL, Error Diagnosis, Visual Analytics, Information Visualization**

I. Introduction

TEXT-TO-SQL tasks is a crucial subtask in the semantic parsing of natural language processing (NLP), as it involves mapping natural language utterances to structured query language (SQL) that can be executed on a relational database. These techniques bridge the semantic gap between a natural language and database [1]. It can benefit various applications, such as enabling non-experts to access data query and facilitating the development of human-computer intelligent interaction. In recent years, text-to-SQL (T2S) tasks have embraced a new breakthrough with the Transformer architecture [2] and large language models [3]. As a result of pre-training and fine-tuning techniques, an increasing number of *T2S* models have demonstrated remarkable performance.

During the development of *T2S* models, NLP scientists are confronted with a series of challenges. Firstly, deep learning models have complex internal mechanisms, making it difficult for developers to explain why and how the model derives a particular decision or prediction. Secondly, the evaluation metrics of the model do not possess the capacity to offer a comprehensive diagnostic assessment for the model. The evaluation metrics for the *T2S* task provide an overall accuracy score for a model, making it challenging to pinpoint specific areas where errors are likely to occur. Model developers are required to examine SQL statements individually to identify error patterns of the model. Extensive textual data presents challenges in the analysis. Data scientists must possess an in-depth understanding of the context surrounding each question and SQL statement to make informed judgments regarding specific instances. Both of these challenges require more effective approaches that enable developers to interactively explore insights from model results and iterate novel models.

Visualization techniques have been employed to assist model developers in comprehending and enhancing deep learning models [4]. In the realm of NLP tasks, model developers confront challenges such as the unstructured nature of natural language and its semantic diversity. Various visualization techniques have been proposed to facilitate the development of various deep learning based language models, such as LSTMVis [5], Seq2Seq-Vis [6] RNNVis [7]. The incorporation of visual analysis can aid in the better understanding of the decision-making process and outcomes of reasoning models, assisting users in analyzing and comprehending the application scenarios and limitations of such models. However, applying existing techniques to text-to-SQL tasks poses challenges because of their exclusive design for specific neural network models, which might not be appropriate for diverse *T2S* parsers that adopt varying neural network frameworks.

In this paper, we propose MAVIDSQL, a novel model-agnostic visual analysis tool that aids developers and users in comprehending and diagnosing for text-to-SQL tasks. The system does not require access to the internal logic of the model and only relies on input instances and output results. This design enables it to support a wide range of model types, as long as they target the same machine learning task and have a consistent input-output format. We conducted an extensive review of the literature in the field to identify the design requirements of our system.

Inspired by the NLP community, we employ a sentence similarity comparison method based on semantic role labeling and syntactic dependency parsing to analyze the impact of input sentence semantics on prediction results. The projected view is generated using similarity measures, enabling users to filter input questions of interest interactively and explore the relationship between input questions and model error patterns at a global-class level in conjunction with model performance statics view. Through the control panel, users have the capability to select datasets and inspect databases for analysis. They may also filter specific attributes to assess the complexity of the model. To enhance the efficiency of analyzing large-scale SQL predictions by users, we employ a modeling technique to identify the differences between predicted and ground-truth SQL queries. These disparities are presented in the SQL comparison view, which users can interact with alongside the raw data view to explore instance-level details. We conducted two case studies and an expert interview to demonstrate the effectiveness and usability of MAVIDSQL in helping model developers understand and diagnose *T2S* tasks.

In summary, the major contributions of our work are as follows:

1) An effective method for extracting semantic attributes from natural language texts and differentiating structural features among SQL statements, enhancing text comparison and interactive exploration in text-to-SQL tasks.
2) MAVIDSQL, a visual analysis tool that generates model-agnostic visualizations for text-to-SQL tasks designed to assist both model developers and users in understanding and diagnosing potential errors.
3) Case studies and expert interviews that demonstrate the effectiveness of our approach in assisting users with the identification and comprehension of prediction errors in *T2S* models through an interactive exploration of model input and prediction results.

II. Related Work

This section discusses the relevant research of our approach. The related work of this paper can be categorized into two groups: Text-to-SQL tasks and Visual Analytics for Deep Learning Models

A. Text-to-SQL Tasks

Text-to-SQL tasks have been developed to bridge the gap between users and data. It enables non-expert users to access and perform intelligent exploratory analysis on tabular data [8]–[10]. Recently, novel text-to-SQL parsers using deep learning methods have gained promising results.

Numerous techniques have been developed for *T2S* tasks, which can be categorized into four modules: *input-encoding*, *schema-linking*, *output-decoding* and *output-refinement*. Seq2SQL [11] and SQLNet [12] were early attempts to apply deep neural networks to *T2S* tasks, using bi-LSTM to encode both natural language sentences and database column names. Pretrained models based on Transformer architecture, such as BERT [13], RoBERTa [14], GraPPa [15], and TaBERT [16], are commonly used.

To better clarify the concept of *schema-linking*, humans typically try to link key words in a natural language question to corresponding elements in a given database when constructing SQL queries. Similarly, a text-to-SQL parser may benefit from employing a similar approach. For example, Schema-GNN [17] and Global-GNN [18] use graph-based approaches to represent database schemas and compute relevance probabilities for schema elements based on the question. RAT-SQL [19] uses a relation-aware transformer to explicitly encode relations between question words and schema elements, improving its ability to handle complex queries. The *output-decoding* in *T2S* models can be categorized into Sequence-based, Sketch-based, and Grammar-based approaches. Sequence-based methods consider SQL queries as a sequential set of sequences [11], while Sketch-based approaches [12] simplify the SQL generation task by decomposing it into multiple simple multi-category sub-tasks [20]. Grammar-based decoders [21] generate a sequence of grammar rules [22] instead of basic components and slots, allowing them to generate grammatically correct complex nested queries. IRNet [23] goes a step further by defining SemQL, an intermediate expression based on an abstract syntax tree to bridge natural language and SQL generation. The *output-refinement*, which can be implemented during the decoding phase in order to reduce the possibility of errors [18] and attain better outcomes [24].

In addition, while large language models (LLMs) such as GPT-4, LLaMA and Alpaca which are trained with Reinforcement Learning for Human Feedback (RLHF), have exhibited remarkable zero-shot capabilities [25]. There are still scenarios where these models may not perform as expected. In some cases, these models may encounter out-of-distribution data, which can lead to inaccurate or nonsensical predictions. Additionally, the performance of these models can be influenced by the quality and diversity of the training data. As a result, there is still a need for researchers and practitioners to carefully evaluate and fine-tune these models for specific applications and domains.

B. Visual Analytics for Deep learning Models

There is a line of related research focusing on visualization to understand, interpret, and diagnose deep neural network models [26] [27]. Techniques such as direct inference and user interactivity in LSTMVis [5], and *what-if* explorations in Seq2Seq-Vis [6] have been commonly used for model interpretation. The mainstream research idea for model interpretation is to combine explainable models with visual analysis techniques. VBridge [28] and NLIZE [29] utilize popular explainable models such as SHAP and LIME to generate contribution-based explanations for a large number of features, which are then organized hierarchically to aid users in interactive refinement of models. For model diagnose, various works conduct performance analysis and provide support for common performance metrics like accuracy, recall, true or false prediction rates, including Squares [30], Manifold [31], which are especially effective in exploring multi-classification tasks. ConfusionFlow [32] proposes three levels of machine learning model exploration: Global level, Class level, and Instance level. For the diagnosis of prediction errors in specific model task scenarios, GNNLens [33] facilitates the exploration and comprehension of prediction error patterns in graph neural network (GNN) models. This tool benefits both model developers and users in their endeavor to gain insights into the inner workings of GNNs.

In summary, while various approaches have been proposed to enhance the interpretability of deep learning models, limited efforts have been devoted to incorporating interaction and

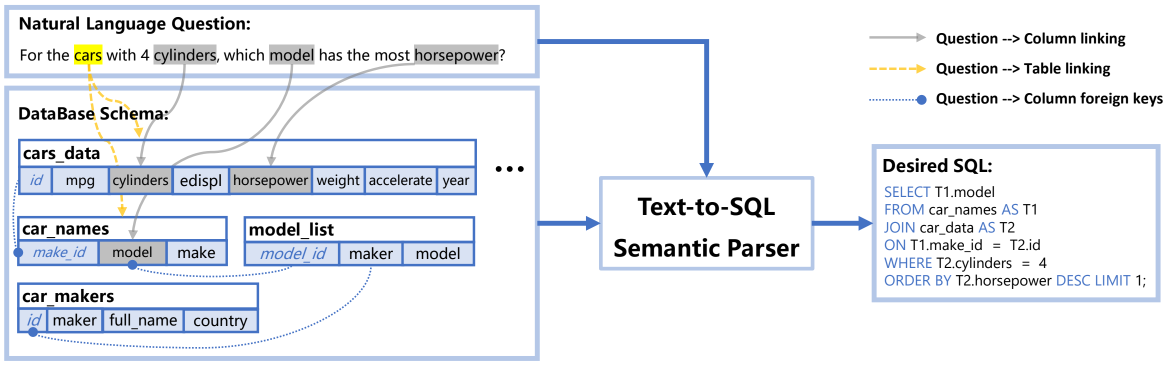

Natural Language Question:
For the cars with 4 cylinders, which model has the most horsepower?
DataBase Schema:
cars_data
id | mpg | cylinders | edispl | horsepower | weight | accelerate | year
car_names
make_id | model | make
model_list
model_id | maker | model
car_makers
id | maker | full_name | country
Text-to-SQL Semantic Parser
Question --> Column linking
Question --> Table linking
Question --> Column foreign keys
Desired SQL:
SELECT T1.model
FROM car_names AS T1
JOIN car_data AS T2
ON T1.make_id = T2.id
WHERE T2.cylinders = 4
ORDER BY T2.horsepower DESC LIMIT 1;

Fig. 1. The goal of text-to-SQL tasks is to convert the input natural language question and database schema into its corresponding SQL queries. This process involves encoding the input question and database table/column names, and also linking the semantics of the question with the SQL schema, to improve the model's prediction robustness. The resulting output is the SQL statement that corresponds to the input question and schema.

visualizations in text-to-SQL tasks to improve their explainability. To address this research gap, this paper proposes a visualization tool that leverages text-linking comparison to aid developers and users in diagnosing and comprehending text-to-SQL tasks.

III. Domain Characterization

Text-to-SQL tasks are typically considered to be end-to-end semantic parsing tasks, that generate SQL statements based on natural language questions, database schemas and schema-linking features constructed by hidden layers, graph structures, or other techniques. This section introduces the related background about the formal problem definition of text-to-SQL parsing, evaluation metrics for verifying text-to-SQL parsing, and datasets used in this study.

A. Text-to-SQL Task Formulation

Text-to-SQL tasks are classic examples of end-to-end semantic parsing. These tasks take a natural language (NL) question and a database schema as input and output their corresponding Structured Query Language (SQL) without the need for additional manual intervention or processing. The Fig. 1 showcases the parsing process of text-to-SQL tasks from the Spider dataset, which employs similar schema annotation as RAT-SQL [19].

Given a natural language question Q and its corresponding database schema $S = \langle T, C\rangle$, the schema consists of table names $T = \{t_1, t_2, \cdots, t_{|T|}\}$, and all column names of each table t_i are expressed as a set $C = \{c_1^{t_1}, c_2^{t_1}, \cdots, c_1^{t_2}, c_2^{t_2}, \cdots, c_{|C|}^{t_{|T|}}\}$. Each table name t_i is described by its name and is further composed of several words $[t_{i1}, t_{i2}, \cdots, t_{i|t_i|}]$, and each column c_j^{ti} in table t_i is represented by words or a phrase $[c_1^{ti}, c_2^{ti}, \cdots, c_{|c_j^{ti}|}^{ti}]$. The whole input can be represented as $X = \langle Q, S\rangle$, the goal of the text-to-SQL task is to generate the SQL query Y.

Then text-to-SQL models typically learn to represent the input using LSTM or Transformer-based encoding models, along with schema-linking techniques. After extensive training and fine-tuning, the models generate SQL queries through a decoder, as introduced in Section II-A.

Our work combines statistical evaluation metrics with the semantic information of the input sentences to create an interactive visual analytics system. This system enables users to explore the impact of semantics on model performance in a dynamic and interactive manner. When users identify error patterns of interest, they can delve into the SQL source data to view and analyze the underlying data structures and relationships. We will provide a detailed discussion of the visualization design and interaction logic in Section V.

B. Motivation

Previous studies have summarized the overview of the challenges encountered by researchers and developers during the development of novel deep learning models. These challenges have encompassed tasks such as debugging coding errors, model comparison and comprehending the inherent characteristics of these models [31]. Drawing inspiration from these studies, we expand this investigation to *T2S* tasks and delineate potential challenges as follows.

Lack of Interpretability in Model Comprehension: The internal mechanisms of many *T2S* models act like a "black box", making their predictive logic difficult to interpret. Model developers apply advanced network architectures and pre-training techniques such as Seq2Seq, Transformer, BERT, GPT-3, to enhance the performance of *T2S* tasks. However, different mechanisms lead to a lack of clear rationale or evidence for developers, which is essential in guiding the development and debugging of these models. The high-dimensional representations within the internal layers of the model pose a significant challenge in comprehending and inferring the predictive behavior exhibited by the model. For instance, when the inquiry takes the form as, "*What is the model of the car with the smallest amount of horsepower?*". The SQL query predicted by the model is as follows: "`SELECT cars_data.Horsepower FROM cars_data ORDER BY cars_data.Horsepower LIMIT 1`". The model's misinterpretation of the question's semantic context resulted in inaccurate predictions [1].

Lack of Diagnosability in Evaluation Metrics: Existing evaluation metrics for *T2S* tasks include accuracy, recall, precision, and F1-score, which serve as valuable and commonly used measures to assess the performance of various models.

Nevertheless, these metrics may not provide a comprehensive understanding of the model's performance. They also do not provide insights into the model's errors and weaknesses, nor do they facilitate focused exploration of the model's predictions [34]. In *T2S* tasks, execution accuracy serves as a metric to assess whether the execution result of the model-predicted SQL in the database matches the ground truth. Additionally, exact set match accuracy evaluates the precision of the output SQL structure in relation to the ground truth SQL clauses. The result can be considered a correct model prediction only if all SQL subcomponents match. However, these metrics provide only numerical accuracy values and may not offer a systematic diagnosis of the errors underlying the model. For example, relying on evaluation metrics makes developers challenging to discern where the model is prone to prediction failures and in which query patterns the model is more likely to generate accurate SQL queries.

Lack of Interactivity in Model Improvement: Model developers encounter challenges during the analysis of model behavior due to handling a substantial volume of unstructured natural language input and structured SQL output. On one hand, aligning and correlating a substantial volume of textual data in the model's input and output. On the other hand, the dataset includes hundreds of databases and thousands of column names. Understanding the diverse contexts of these databases is essential and adds complexity to the data exploration and analysis process. Furthermore, a large volume of textual data also presents challenges in analyzing and uncovering the primary data patterns within the constraints of limited space, thereby limiting the utilization of domain knowledge by developers. Designing an interactive system that enables developers to integrate domain knowledge and visualize the distribution of model input semantics and model output comparison could aid in gaining a comprehensive understanding of the strengths and weaknesses of models. This approach has the potential to provide insights to developers, improving their efficiency when iterating on new models, and it reduces the burden on developers while mitigating the risk of errors.

While the challenges faced by the model have been outlined, addressing these issues may not be straightforward. It requires a comprehensive understanding of the task's workflow, input-output data, and domain knowledge. Visual analytics techniques can assist in both of these aspects. Model developers can apply appropriate visualizations to explore the large-scale and complex input and output data, thereby gaining additional insights into the data and uncovering relationships between the data and the model output. The next section presents a design requirement analysis specifically tailored to visual analysis for T2S tasks.

IV. Design Requirement Analysis

MAVIDSQL is a tool designed to interpret and diagnose the error pattern patterns between model inputs and corresponding prediction results of text-to-SQL models. This tool aims to enhance the efficiency of model fine-tuning. The general goal of MAVIDSQL is to understand the factors that cause these models predict failure, and to interactive discover common pat-

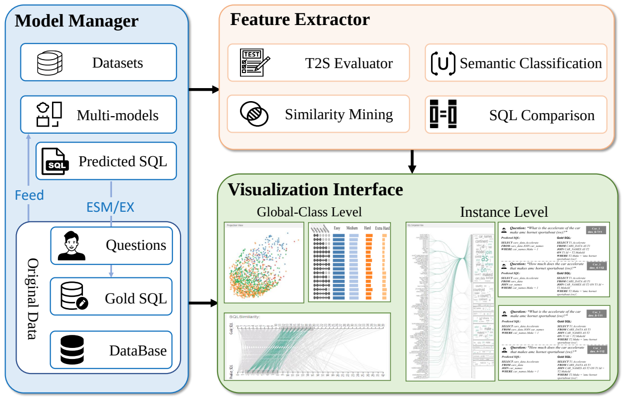
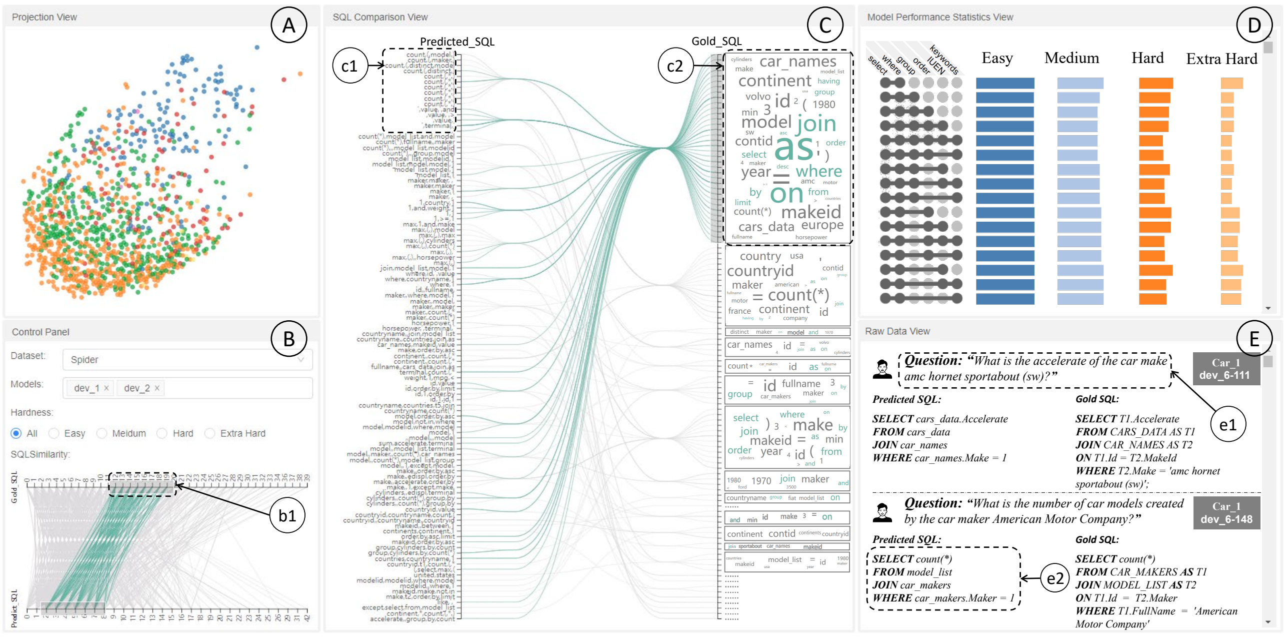


Fig. 2. Overview of the system architecture. Model Manager: the model manager that executes text-to-SQL models, generates predictions, and manages datasets. Feature Extractor: evaluates the model provided by the model manager, extracts relevant semantic features, and passes them into the visualization interface. Visualization Interface: the visualization interface allows for interactive exploration from global-class level to instance level.

terns of model prediction errors for improving the efficiency of fine-tuning. This could involve identifying patterns in the types of errors that the models tend to make, or identifying areas of the input text where the models have difficulty interpreting the meaning. By gaining insights from these factors, it may be possible to improve the performance of text-to-SQL models and make them more effective at generating precise queries. Based on the discussions with domain experts, a review of current literature, and our own attempts to reproduce the SOTA models one Spider leaderboard, the specific requirements are formulated as follows:

R1: Associate and align Text-to-SQL data. Neural network models are characterized by their black-box nature, which refers to the embedding of input and output features within the hidden state of the model. Consequently, the interpretation and application of such models are hindered by a lack of intuitive understanding of the underlying data. In the context of model input data, the test dataset comprises a substantial number of text messages that exhibit similar structural characteristics but differ in terms of their semantic content. Identifying which question patterns are beneficial for model prediction is important for enhancing model performance. For model output data, SQL statements have a complex and flexible structure, as well as the absence of precise evaluation metrics to evaluate the accuracy of model output data, a collaborative approach involving aligned model predictions, ground truth data, human feedback can be instrumental in identifying and interpreting patterns of model error.

R2: Provide an overview of Text-to-SQL prediction. Experts and literature research show that an overview of the model performance is crucial for *T2S* task results analysis. To gain an overview of the dataset and predict results, the system needs to summarize various types of information, such as performance statistics and ground truth label distribution. This information, covering various aspects of a *T2S* model, needs to be organized and presented in a clear manner. Meanwhile, users should be provided with the interrelation between this information to aid in forming preliminary hypotheses regarding



Question: "What is the accelerate of the car make amc hornet sportabout (sw)?"

Car_1 dev_6-111

Predicted SQL:

```
SELECT cars_data.Accelerate
FROM cars_data
JOIN car_names
WHERE car_names.Make = 1
```

Gold SQL:

```
SELECT T1.Accelerate
FROM CARS_DATA AS T1
JOIN CAR_NAMES AS T2
ON T1.Id = T2.MakeId
WHERE T2.Make = 'amc hornet
sportabout (sw)';
```

Question: "What is the number of car models created by the car maker American Motor Company?"

Car_1 dev_6-148

Predicted SQL:

```
SELECT count(*)
FROM model_list
JOIN car_makers
WHERE car_makers.Maker = 1
```

Gold SQL:

```
SELECT count(*)
FROM CAR_MAKERS AS T1
JOIN MODEL_LIST AS T2
ON T1.Id = T2.Maker
WHERE T1.FullName = 'American
Motor Company'
```

Fig. 3. The explanatory interface of MAVIDSQL consists of five views. (a) The Projection View provides the similarity among input natural language questions, and supports lasso-selection interactions for exploring global-level model performance and instance-level SQL prediction results. (b) The Control Panel enables users to interactively configure basic parameters (e.g., the dataset). Users can also swipe a particular set of SQL predictions of interest to obtain further details. (c) The SQL Comparison View further visualizes the specific category of user-selected SQL comparison results. (d) The Model Performance Statistics View provides users with an overall evaluation of SQL prediction performance. (e) The Raw Data View presents detailed information on the original data used in the model.

potential error patterns in *T2S* results. This entails identifying a group of incorrect predictions that exhibit comparable question patterns or SQL structures.

R3: Identify Text-to-SQL model failure patterns. After developing initial hypotheses about the error patterns, users need more detailed information to verify them. Specifically, users need to examine the question pattern or SQL structure shared by a set of wrong predictions and verify whether error patterns formed by these patterns make sense in analyzing *T2S* based on their domain knowledge. During the literature review and expert interview, domain experts agreed that there are some relatively complex SQL structures that often make the model wrong [1]. For example, the model could encounter an issue of robustness, resulting in its inability to identify the SQL table name that corresponds to a given statement. Therefore, the system should support users in examining the influences of these association characteristics and identifying error patterns.

R4: Support multi-level exploration of Text-to-SQL model prediction. For a comprehensive understanding and analysis of the *T2S* model's predictions, visualization should empower users to explore the model input and output. The exploration should be conducted on multiple levels, comprising an overview of the model's performance at the global level, and a detailed analysis of the raw data at the instance level. Specifically, users infer the causes of error patterns by interacting with a filtering mechanism for global performance, which allows them to pinpoint the raw question and SQL structures. Aggregating sentences of the same category together makes it convenient for users to quickly search and infer them.

V. MAVIDSQL

As shown in Fig. 2, MAVIDSQL consists of three major modules: the Model Manager, Feature Extractor, and Visualization Interface. The Model Manager model is responsible for storing and managing text-to-SQL datasets and model predictions. The Feature Extractor model carries out the necessary feature construct procedures for analyzing the *T2S* model predictions. The processed data feature is then passed to the visualization module, which supports the interactive visual analysis of the *T2S*. The Model Manager and Feature Extractor modules are developed using the Python, while MongoDB is employed as the data storage engine. The system is integrated into a backend web server built using Flask. We implement the visualization module as a front-end application using Vue, JavaScript, and D3.

A. Methods

In this section, we introduce the primary data mining and layout optimization techniques employed in this paper for extracting features from input and output data. For model input natural language question data, we perform semantic classification and sentence similarity mining to assess the similarity characteristics. For model output SQL queries, we calculate the text-level differences between the model Predicted-SQL and the ground truth Gold-SQL to facilitate large-scale comparisons of p-g SQL pairs in visualization.

*1) **Input natural language question similarity mining**:* In the context of text-to-SQL models, the performance of prediction results is influenced by the variability in natural language descriptions and syntax structures. In order to address the challenge of measuring semantic similarity in interrogative and imperative sentences, our approach was informed by the NLP field and has undergone multiple experimental iterations to refine and optimize the methodology. The approach integrates multi-granularity semantic information, including lexical semantics, dependency syntax, and sentence structure patterns, to accurately capture the nuances of sentence similarity. Text-to-SQL models usually trains an input encoding model to map natural language query tokens into a high-dimensional embedding vector space. We initially considered whether it could directly use the results of the model input encoding to calculate the similarity of the questions. However, it is experimentally proven that mapping input sentences into high-dimensional vector space after encoding by various models can introduce uncertainty, make it difficult to reflect the desired syntactic structure and semantic information.

Researchers commonly consider both lexical and semantic factors when evaluating the semantic similarity between sentences in the natural language processing (NLP) community [35]. Various corpus-based methods have been proposed for this purpose, including N-gram, WordNet, Jaccard similarity, and Word Embedding. These approaches primarily concentrate on the semantic representation of words, exhibiting limited engagement with the contextual associations within the text. For instance, two syntactically similar natural language questions, *"How much water is available?"* and *"How many countries are listed?"* may exhibit substantial semantic differences in their vector representations, but they correspond to very similar SQL query structures.

Additionally, dependency parsing is a another widely used technique in NLP that involves identifying the syntactic structure of a sentence by analyzing the relationships between words in a sentence. The output of a dependency parser is a tree-like structure that represents the syntactic relationships between words in a sentence. On the other hand, Role labeling is a technique that involves identifying the semantic relationships between words in a sentence, such as who or what is the subject, object, or indirect object of a sentence. However, relying solely on dependency syntax trees for evaluation may prove inadequate in capturing the nuances of sentence structure. For example, consider the following inquiries: *"In which years cars were produced weighing no less than 3000 and no more than 4000?"* and *"What are the different years in which there were cars produced that weighed less than 4000 and also cars that weighted more than 3000?"* Although these two sentences exhibit semantic proximity, they manifest significant differences in their syntactic structures.

We comprehensively integrate both the semantic information and syntactic structure of the model input questions. Furthermore, to correlate questions from similar databases, we also incorporate additional meta features such as sentence length and the count of shared words. The process involves generating dependency parsing trees for two sentences, identifying corresponding roles through specific formulas. We utilize Hanlp [36] for the extraction of relevant features, which comprise categorically labeled identifiers based on the NLP task. The dependency parsing trees of two sentences are obtained using Hanlp, followed by the computation of identical dependency parsing roles' positions in the sentences.

Firstly, we calculate sentence semantic similarity between two natural language question sentences Q_a and Q_b, denoted as $SenSemSim(Q_a, Q_b)$. We tokenize sentence Q to obtain tokens $T_1...T_n$, where each token represents an individual word in the sentence.

Then, we utilize the Word2Vec module from the gensim library to train a Skip-gram model and determine the correlation tokens denoted as $SemSim(T_{ai}, T_{bj})$. The calculation for $SenSemSim(Q_a, Q_b)$ is as follows.

$$SenSemSim(Q_a, Q_b) = (\frac{\sum_{i=1}^{m} S_{ai}}{m} + \frac{\sum_{j=1}^{n} S_{bj}}{n})/2 \tag{1}$$

$$S_{ai} = max(Sem(T_{ai}, T_{b1}), ..., Sem(T_{ai}, T_{bn})) \tag{2}$$

$$S_{bj} = max(Sem(T_{a1}, T_{bj}), ..., Sem(T_{am}, T_{bj})) \tag{3}$$

$$SemSim(T_{ai}, T_{bj}) = \frac{V(T_{ai}) \cdot V(T_{bj})}{\|V(T_{ai})\| \cdot \|V(T_{bj})\|} \tag{4}$$

Where S_{ai} and S_{bi} are intermediate variables that measure the semantic similarity between words in sentences Q_a and Q_b, respectively. $V(T_{ai})$ and $V(T_{bj})$ are the vector representation of tokens T_{ai} and T_{bj} obtained from the word2Vec model. $\|V(T_{ai})\|$ and $\|V(T_{bj})\|$ are the euclidean norms (or magnitudes) of the vector representations for T_{ai} and T_{bj}, respectively.

To enhance the assessment of sentence syntactic similarity between questions Q_a and Q_b, denoted as $SenSynSim(Q_a, Q_b)$, we focus on the hierarchy distance, $HierDis(r_{ai}, r_{bi})$, between tokens. In this context, r_{ai} and r_{bi} are defined as the i^{th} dependency parsing roles corresponding to the tokens in questions Q_a and Q_b, respectively. The measurement of $HierDis(r_{ai}, r_{bi})$ is pivotal as it quantifies the hierarchical distance between each node and the root in the dependency syntactic tree of the sentences. The formula for calculating the semantic similarity between sentences is provided as follows.

$$SenSynSim(Q_a, Q_b) = \frac{\sum_{i=1}^{n} HierDis(r_{ai}, r_{bi})}{n} \tag{5}$$

$$HierDis(r_{ai}, r_{bi}) = 1 - \left|\frac{Deep(r_{ai}) - Deep(r_{bi})}{Deep(r_{ai}) + Deep(r_{bi})}\right| \tag{6}$$

Here, $Deep(r_{ai})$ and $Deep(r_{bi})$ denote the depth of the respective roles in the semantic dependency parsing tree, using the root node as the reference point. The proximity of these depths to one another indicates the degree of similarity between the two questions.

In addition, to distinguish queries aimed at different databases, we have incorporated an analysis of sentence metadata, which includes sentence length and the frequency of common words. The similarity of common words between sentences denoted by $ComWordSim(Q_a, Q_b)$. Moreover, the sentence length similarity denoted as $SenLenSim(Q_a, Q_b)$.

$$ComWordSim(Q_a, Q_b) = \frac{2 * SameWord}{Len(Q_a) + Len(Q_b)} \tag{7}$$

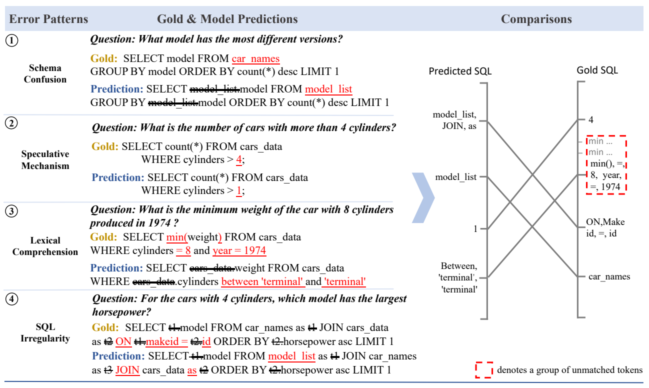
$$SenLenSim(Q_a, Q_b) = 1 - \left|\frac{Len(Q_a) - Len(Q_b)}{Len(Q_a) + Len(Q_b)}\right| \tag{8}$$

Where $SameWord$ represents the number of identical words present in questions Q_a and Q_b, while $Len(Q_a)$ and $Len(Q_b)$ denote the number of tokens in questions. To attain optimal results, it is imperative to adopt a comprehensive approach rather than focusing exclusively on a single aspect. Therefore, in the processing of sentence similarity, we treat the aforementioned factors as different feature items of the sentences. By integrating these feature items and assigning different weights according to their importance, we obtain the final similarity score between the sentences.

Finally, the final formula $Similarity(Q_a, Q_b)$ is derived by weighting the semantic and syntactic structures and the metadata. When querying different databases with the intention of expressing similar SQL query, the common vocabulary in the two queries tends to decrease, leading to a lower sensitivity to sentence metadata in the final metric, and consequently, a lesser weight is assigned to it. Conversely, the dependency parse tree and semantic analysis exhibit a higher sensitivity, and thus, are assigned a greater weight. The final calculation formula for $Similarity(Q_a, Q_b)$ is as follows.

$$\begin{aligned} Similarity(Q_a, Q_b) = {} & \gamma_1 SenSemSim(Q_a, Q_b) + \\ & \gamma_2 SenSynSim(Q_a, Q_b) + \\ & \gamma_3 ComWordSim(Q_a, Q_b) + \\ & \gamma_4 SenLenSim(Q_a, Q_b) \\ s.t. \quad \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 = 1 & \end{aligned} \tag{9}$$

Employing our sentence similarity algorithms, we calculated the pairwise similarity among input sentences. This similarity metric was then applied to quantify sentence distances within the projection view of dimensionality reduction processes.



① Schema Confusion

Question: What model has the most different versions?

Gold: SELECT model FROM car_names GROUP BY model ORDER BY count(*) desc LIMIT 1

Prediction: SELECT ~~model_list.~~model FROM model_list GROUP BY ~~model_list.~~model ORDER BY count(*) desc LIMIT 1

② Speculative Mechanism

Question: What is the number of cars with more than 4 cylinders?

Gold: SELECT count(*) FROM cars_data WHERE cylinders > 4;

Prediction: SELECT count(*) FROM cars_data WHERE cylinders > 1;

③ Lexical Comprehension

Question: What is the minimum weight of the car with 8 cylinders produced in 1974 ?

Gold: SELECT min(weight) FROM cars_data WHERE cylinders = 8 and year = 1974

Prediction: SELECT ~~cars_data.~~weight FROM cars_data WHERE ~~cars_data~~.cylinders between 'terminal' and 'terminal'

④ SQL Irregularity

Question: For the cars with 4 cylinders, which model has the largest horsepower?

Gold: SELECT ~~t1.~~model FROM car_names as ~~t1~~ JOIN cars_data as ~~t2~~ ON ~~t1.~~makeid = ~~t2.~~id ORDER BY ~~t2.~~horsepower asc LIMIT 1

Prediction: SELECT ~~t1.~~model FROM model_list as ~~t1~~ JOIN car_names as ~~t3~~ JOIN cars_data as ~~t2~~ ORDER BY ~~t2.~~horsepower asc LIMIT 1

Fig. 4. Common error patterns in text-to-SQL models and principal methods of SQL alignment.

2) ***Output SQL result side-by-side alignment:*** The evaluation of T2S models typically involves a set of test dataset, comprising a series of query inputs, against which the model's predictions are compared to assess its performance against the ground truth. Developers are required to conduct thorough analyses of extensive comparisons between predicted and actual SQL outputs when evaluating model performance. The complexity of SQL statement structures necessitates domain knowledge and an in-depth comprehension of the database context. Additionally, it is challenging to display an adequate number of SQL comparison pairs within the constraints of limited screen space.

Extensive research in visual analytics focuses on the alignment and comparison of textual data [37], such as sequence-aligned [38], aligned barcodes [39], and side-by-side [40]. We aim for developers to preserve the essential information in SQL statements during the analysis process, while minimizing redundant words, such as SQL keywords, that appear in both predictions and the ground truth. We used Algorithm. 1, to preprocess the model's output of predict SQL and gold SQL. As shown in Fig. 4, considering the two SQL statements provided as input, this approach involved parsing each SQL statement into a set of tokens, while removing extraneous information such as table aliases and punctuation. We conducted a pairwise comparison between the generated tokens, eliminating any identical tokens and identifying discrepant ones to accentuate dissimilarities between the two token sets.

We visualize the data of SQL-pairs using a side-by-side approach based on parallel coordinates shown in Fig. 5(A). Furthermore, we group SQL statements with common prefixes, allowing for class-level comparison of differences among SQL queries of this category. This approach is illustrated in Fig. 4.

Algorithm 1: SQL Statement comparison

```
Result: Two sets of the same token elements after
        elimination, $p_{diff}$ and $g_{diff}$
1  Input: The string of model predict SQL $p_{sql}$ and gold
     SQL $g_{sql}$ provided by dataset;
2  $p_{sql}$ and $g_{sql}$ are parsed into sets of tokens $p_{tokens}$ and
     $g_{tokens}$ respectively, based on SQL syntax. During
     the parsing process, table aliases are removed and
     SQL keywords are standardized to a uniform format;
3  /* Eliminate the same tokens between $p_{tokens}$ and
     $g_{tokens}$ */;
4  Set $p_{diff} \leftarrow p_{tokens}$;
5  Set $g_{diff} \leftarrow g_{tokens}$;
6  foreach $p_{tokens}$ $p_t$ do
7      if $p_t$ in $g_{tokens}$ then
8          $p_{diff} \leftarrow$ remove $p_t$ from $p_{diff}$;
9          $g_{diff} \leftarrow$ remove $p_t$ from $g_{diff}$;
10     end
11 end
12 return $(p_{diff}, g_{diff})$;
```

3) ***SQL group comparison layout optimization:*** We aspire to enable users to interactively analyze a large number of SQL pairs within the constraints of limited screen space. To this end, we design views based on parallel coordinate visualization, which involve the comparison of SQL tokens in a side-by-side alignment, as shown in Fig. 5. However, with the increase in data volume, the resultant crossing of lines leads to visual clutter, posing challenges in pattern recognition. To address the issue, we employ a hierarchical density aggregation technique to group similar results and present them as word clouds organized by category. These approaches draw inspiration from previous work on aggregation techniques [7] and visual designs [41]. We employ edge-bundling and categorization techniques to reduce visual clutter and cognitive load for users by sorting and grouping data on the axes.

In conventional parallel coordinate plots, lines are employed to connect the values between two axes. Nevertheless, this approach often engenders issues of visual clutter and excessive plotting, particularly as the quantity of data points and connections increases. Initially, following the previously discussed group method, we arrange SQL tokens belonging to the same group together for presentation. Items within a group will appear higher on the axis based on their quantity. Consequently, as the volume of data on the axis escalates, the primary error patterns predicted by the model become increasingly concentrated towards the upper of the axis.

To provide smooth curves that are easy to follow with the eyes, we bundle cluster around each axis, after categorizing and sorting the data. The classic parallel coordinate plot employs lines to link the values between the two axes. However, this often leads to issues with visual clutter and overplotting, especially when the number of data points and links increases. Drawing inspection from previous work [41], we add virtual bundling axes adjacent to each data axis in the parallel coordinate plot, as illustrated in Fig. 5. All data points belonging to the same category are bound to the corresponding position on the virtual axis through cubic Bézier curves. The y-coordinate of the points is calculated based on the median of all points in the same category The points on the two aggregated virtual axes are also connected using Bézier curves, which helps to improve the visual grouping and reduce visual clutter. From Fig. 5, we can observe the differences between the classical parallel coordinate plot and the aggregated version.

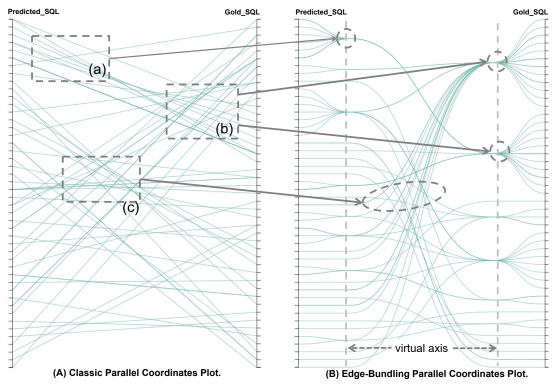


Fig. 5. Comparison of the different layouts for parallel coordinates plots depicting two dimensions from the comparison of predicted and gold SQL: (a) Classic Parallel Coordinates Plot; (b) Edge-Bundling Parallel Coordinates Plot. The edge-bundling technique effectively reduces visual clutter caused by line crossings and alleviates the visual cognitive load.

B. Visualization

As shown in Fig. 3, MAVIDSQL consists of five visualization modules. The Projection View (A) and the Model Performance Statistics View (D) provide an overview of the input data and the model's prediction results. The Control Panel(B) provides users with options to select a dataset, choose specific models and hardness levels for analysis. The SQL Comparison View (C) and the Raw Data View (E) offer interactive analysis at different granularities, assisting users in identifying error patterns of interest. In this section, we provide a comprehensive overview of the visual representations and interactive capabilities of each module, along with a detailed explanation of our design considerations.

*1) **Projection view:*** In order to better illustrate the association between input natural language question (R1) and help users analyze the impact of semantic patterns on prediction results (R2), we design the Projection View to visualize the similarity of input sentence. In the Projection View, we use the multidimensional scaling projection algorithm to map all the input natural language questions. Each sentence is described by a series similarity distance, which is introduced in Section V-A1. The distance between these sentences is further mapped to 2-dimensional space to obtain a quick overview of semantic association of input interrogatives. The Projection View allows users to explore the similarity of combine question categories and semantic information and explore the impact of different categories of sentences on the input result. The visual encoding is used to differentiate between different question categories. (e.g., blue for "*how*" questions, Orange for "*what*" questions, red for "*for which*" questions, and green for "*others*" questions). It can be helpful for investigating whether the sentence with similar semantic structure share similar error patterns. In our implementation, we categorize the input question into seven groups according to wh-word in conjunction with expert opinion. When users lasso-select a set of sentences in a projection plane, the corresponding prediction errors and accuracy will be displayed. This enables users to interactively explore the impact of nodes on model predictions.

Since we have extracted the semantic features of input question and model predict score, we are considering displaying more data information when projecting dimensionality reduction while avoiding the visual clutter caused by data superposition.

*2) **SQL Comparison View:*** The *SQL Comparison View* is the key component of MAVIDSQL, which allowing users to compare and identify discrepancies between model-generated SQL and ground truth (R2, R3, R4). The system should reveal the difference between both predict and gold SQL queries. We work closely with the experts to incorporate their feedback and the core design consideration is to show scenarios of SQL prediction failures while retaining enough of the original SQL information. During the design process, we faced the challenge of presenting a large number of predicted and gold SQL pairs without aggravating the cognitive burden. Therefore, we followed both paper and code-based prototyping approaches to refine this system according to user feedback. We summarize our design considerations and describe the details of each component as follows:

SQL Similarity Component: As introduced in Section V-A2, the SQL Similarity Component eliminates the same tokens from the predicted and ground truth SQL statements and defines the number of remaining words as similarity between them. As shown in Fig. 3(b1), this component provides an initial overview of SQL queries where the model has failed to accurately predict the expected output. It features a 2-axis parallel coordinate component that showcases the similarity between the predicted and ground truth SQL. In this component's design, each polyline represents a specific number of remaining words after removing the common tokens. Users can explore the distribution of these errors by brushing on each coordinate to select a specific sequence within a certain range. The SQL-Pairs Comparison Component then displays the corresponding detail SQL words for further class-level analysis.

SQL-Pairs Comparison Component: In order to mitigate potential uncertainties associated with existing model evaluation strategies (e.g. false negative, worst case [42]), and to prevent visual clutter caused by displaying large amount of SQL statements in text form. We design a comprehensive SQL comparison component that allows users to easily compare the predicted and ground truth SQL statements at a class level. This view presents the data in a concise and informative manner, making it easy for users to interpret and analyze the results. Previous work has explored a class-level analysis of machine learning models to understand their behavior and identify common error patterns [31], [32]. Inspired by previous research, we use this strategy to investigate error patterns in text-to-SQL tasks. We employ an enhanced edge-bundling layout for interactive parallel coordinates to support a comparative analysis of predicted and gold SQL queries, present in Section V-A3. The SQL groups that have been aggregated will be presented in this view, supporting the exploration at the class level. This layout allows users to visually identify patterns and discrepancies in the data side-by-side. Users can select which group of detail information they are interested in to be displayed in the Raw Data View.

As shown in Fig. 3(c), when user selects a group of predicted and ground truth SQL results from the SQL similarity component, the SQL comparison view will display more detailed SQL comparison information categorized by specific word feature metrics. The two axes in the SQL similarity component represent the remaining words in the predicted and ground truth SQL statements after eliminating the common tokens. The left axis represents the predicted SQL tokens and the right axis represents the ground truth SQL tokens. Both axes are categorized based on the common initial tokens and sorted by their quantity. Each tick on axis represents a specific state after eliminating the common tokens. The tokens are mapped on both sides of the axis, as shown in Fig. 3(c1, c2). Given the wide variety of forms in which predicted SQL statements can occur, the associated tokens may be scattered. To avoid the potential loss of important cases, we display the original text directly on the axis. However, as gold SQL statements are typically structured uniformly, we present them in the form of word clouds to facilitate class level pattern recognition and improve user comprehension. The predict and gold SQL pairs between the two axes are connected using an improved version of the cubic Bézier curves.

In general, the SQL Comparison View aims to present the distribution of predicted results at the class-level. To address the visual clutter and reduce cognitive load during parallel coordinate plotting, we aggregate word clouds using a combination of classification and edge bundling based on axis number ordering. This technique effectively reduces visual clutter and enhances the user experience.

*3) **Model Performance Statistics View:*** To facilitate evaluation and calculation, each SQL statement within a structure is assigned to an individual object instance, as depicted in Fig. 6.

We first compare the ground truth and predicted components on a module-by-module basis. This allows users to obtain the accuracy of each module of the predicted SQL (Fig. 6a). Furthermore, since all databases have executable SQLite files, we can also measure the execution accuracy of the *T2S* model. To do so, we execute the generated SQL statements in the SQLite environment and compare the results of the Gold and Predicted SQL runs (Fig. 6b).

We design the Model Performance Statistics View to help users intuitively perceive the distribution of the model performance, as shown in Fig. 3(D). The Model Performance Statistics View includes two components: a multi-set combination matrix plot and a stack bar chart, inspired by Upset [43], LineUp [44] and VSumVis [45]. For evaluation of the model performance, Exact Set Match Accuracy (ESM) is the primary metric, which is a multidimensional metric that predicts various components of the SQL statements such as select, where, group, order, IUEN, and keywords. The goal is to classify and analyze the model performance based on these metrics to identify potential error patterns in the model. Therefore, we employ the combination matrix from UpSet [43] to display different categories in the Model Performance Statistics View.

Moreover, queries in the dataset are often categorized by their level of difficulty. Examining the model predictions for different levels of difficulty can greatly aid in identifying and improving the performance of the model. We display the model's performance on different difficulty levels through four bar charts.

*4) **Raw Data View:*** As shown in Fig. 3(e), the Raw Data View is linked interactively with the SQL Comparison View, enabling users to access detailed information by conducting an instance level exploration. In the process of designing the system, it becomes apparent that users would benefit from intuitive guidance at the instance-level when exploring raw data, without being overwhelmed by cognitive load. In Raw Data View, we provide a comprehensive set of raw data on the model's input and output, including the natural language questions, the SQL statements generated by the model and dataset ground truth. All the raw data is represented in original text format, annotated with database numbers and corresponding data index. Researchers can combine SQL Comparison View and Model Performance Statistics View to analyze the results of the model. Additional offline data can be exported, allowing users to debug and improve their models.

C. User Interactions

The MAVIDSQL provides a set of interactions, which facilitate the interplay of different views and enable multi-level exploration with details on demand.

Lassoing for Input Exploration. Firstly, users can view the overall performance of the model, or obtain a preliminary overview, through the Projection View and Model Performance Statistics View. To enhance scalable exploration in the Projection View, users may employ the lasso tool to select concentrate on particular instances of interest. Then, the detailed information will be displayed in the SQL Similarity Component and the SQL Comparison View.

Filtering for Dataset Selection. In order to filter datasets and text complexity according to user interests, users have the capability to select and view these filters within the Control Panel.

Brushing for Feature Analysis. Users can employ the brushing feature on SQL similarity and SQL comparison components to assess the unmatched SQL tokens. This process facilitates the discovery of distinct model prediction patterns. Furthermore, the SQL predictions brushed in the SQL comparison view will be displayed in the Raw Data View, assisting users in detailed analysis.

Clicking for Detailed Access. Our system offers click-responsive features for viewing detailed information. For instance, users can click on the SQL comparison component to generate a word cloud visualization, highlighting a group of SQL keywords present in the gold SQL after performing a comparison.

VI. Evaluation

In this section, we demonstrate the effectiveness of MAVIDSQL in facilitating the understanding and diagnosis of *T2S* tasks at both the global-class and instance levels. Two case studies and expert interviews are conducted with three domain experts (*E1*, *E2*, *E3*), within the Spider dataset [46] and test suite data [42]. *E1* is an NLP researcher with a decade of experience in NLP. Furthermore, *E1* possesses expertise in applying and designing deep learning models for the purpose of converting unstructured text into structured formats. Both of *E2* and *E3* are senior Ph.D. students in computer science. *E2* is engaged in the visualization of NLP tasks, while *E3* primary focus on data visual analytics. Notably, none of these experts are co-authors in our research. The two cases are discovered by *E1* and *E2* during the system exploration. Furthermore, comprehensive feedback from all the experts is also collected and summarized.

A. Datasets and Benchmark

In our case study, we employ Spider [46] dataset to establish analytical tasks. It is recognized as one of the most widely used single-turn datasets in the text-to-SQL community. The datasets is a large-scale, cross-domain semantic parsing dataset containing 10,181 natural language questions and 5,693 unique SQL queries across 200 databases, spanning 138 different domains.

The SQL queries are divided into four levels based on their difficulty: easy, medium, hard, and extra hard. This categorization allows for a more comprehensive evaluation of model performance across various types of queries. This categorization allows for a more comprehensive evaluation of model performance across various types of queries. As illustrated in Fig. 6, each Spider instance consists of: *database id, question, query* and *gold sql*. The model's input-output performance is assessed using the 21 leaderboard submissions for the Spider dataset as proposed by Zhong et al. [42] to ensure the validity of our results.

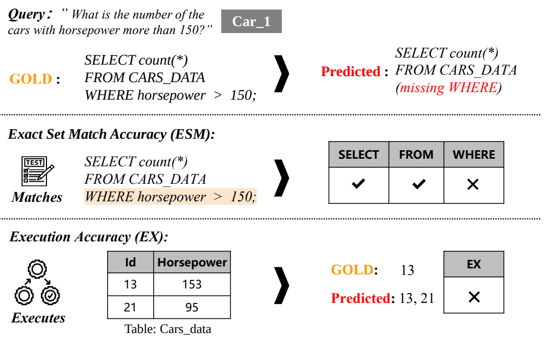

Query : " What is the number of the cars with horsepower more than 150?"
Car_1
GOLD :
SELECT count(*) FROM CARS_DATA WHERE horsepower > 150;
Predicted :
SELECT count(*) FROM CARS_DATA (missing WHERE)
Exact Set Match Accuracy (ESM):
TEST
Matches
SELECT count(*) FROM CARS_DATA WHERE horsepower > 150;
SELECT FROM WHERE
✓ ✓ ✗
Execution Accuracy (EX):
Executes
Id Horsepower
13 153
21 95
Table: Cars_data
GOLD: 13
Predicted: 13, 21
EX
✗

Fig. 6. Examples of evaluation metrics for text-to-SQL tasks, illustrating how model predictions are evaluated through Exact Set Match Accuracy and Execution Accuracy metrics. Evaluating errors in different modules can assist users in analyzing the semantic understanding issues of the model across different SQL slots. In addition, execution accuracy can prevent misclassification caused by SQL variations in different forms.

B. Model Metrics and Error Patterns

1) Evaluation metrics of T2S tasks: Typically, text-to-SQL parsers are evaluated by comparing the generated SQL queries to the ground truth. Specifically, there are two main types of evaluation metrics used for evaluating the *T2S* tasks: Exact Set Match Accuracy (ESM) and Execution Accuracy (EX) [46]. Additionally, the official evaluation metric for the Spider dataset is the Test Suites Accuracy [42], which is an expansion of the ESM.

Exact Set Match Accuracy is determined by comparing whether the sets of SQL clause match exactly between the ground-truth SQL query and the prediction. Both the prediction and the ground truth are parsed into bags of several subcomponents, including SELECT, WHERE, GROUP BY, ORDER BY, KEYWORDS, all SQL keywords without column names and operators. The evaluation script compares the sets of subcomponents in each SQL component side by side to determine if they match between the ground-truth and predicted SQL queries. *Execution Accuracy* has been introduced to consider the accurate execution of a predicted SQL query on a specific instance of a database. Considering the possible variance in SQL structure predictions across different models, the developers introduce that these queries must be executed directly within the SQL database.

Moreover, existing *T2S* datasets are often divided into classes based on the difficulty level of the queries, but most evaluation metrics are only used for global-level analysis, making it difficult to conduct a more detailed exploration of the performance of *T2S* tasks. In order to overcome these limitations, we propose a global-class level exploration that combines our designed projection view, model performance statistics view, and SQL similarity view in MAVIDSQL. These views allow for a more fine-grained analysis of the performance of *T2S* models and facilitate the discovery of patterns and errors in the model's predictions.

Global-class level evaluation metrics provide a rapid and high-level overview of model performance. In contrast, instance-level analysis enables users to gain a more comprehensive understanding of the model's performance by identifying specific errors and patterns. In our system, we present global-class level model metrics as an overview in the statistical view, allowing users to understand the model's overall performance distribution. Subsequently, through interaction, users can examine more detailed, instance-level analysis. By leveraging instance-level analysis, users can explore individual instances in detail, compare the predicted and gold SQL statements, and evaluate the model's performance on specific subtasks, ultimately leading to a more comprehensive understanding of the model's strengths and limitations. Additionally, users can also uncover additional instances based on existing error patterns.

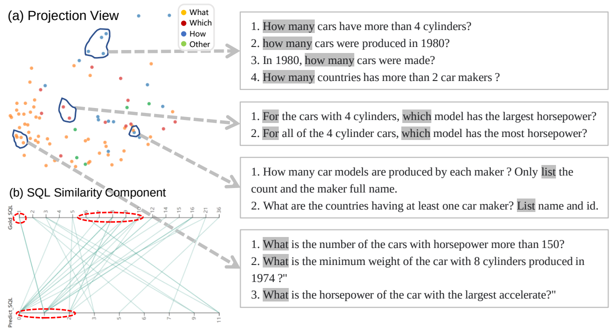
2) Error patterns of T2S tasks: Through collaborative exploration within our experts, we have characterized the common error patterns that MAVIDSQL can identify.

P1: Schema Confusion. In the *T2S* tasks, it is imperative for the model to effectively comprehend the intent of the questions. This involves discerning the primary element of the query and creating a schema that links it to the corresponding database [17]. Subsequently, it is crucial to precisely interpret the semantic query's intentions, such as discerning whether it is seeking a maximum or minimum value or determining which column to group by. As illustrated in Fig. 4(1), while the model accurately predicted subsequent filtering and aggregation operations, confusion regarding the table structure led to the selection of an incorrect database table, resulting in a prediction failure.

P2: Lexical Comprehension. When the input query contains vocabulary or expressions not encountered during the model's training, there is a potential for misunderstanding the meaning of these words, which might lead to incorrect interpretation or selection of the query's intent [1]. As demonstrated in Fig. 4(2), the model misinterpreted the intent of the question, failing to predict the "`MIN`" operation.

P3: Speculative Mechanism. Due to the characteristics of the Spider dataset's evaluation mechanism, test suite execution accuracy focuses on assessing the SQL execution and the alignment of various SQL components. During the evaluation process, it does not examine the values [42]. During the prediction process, several models resort to utilizing default value inputs to populate the SQL queries. As shown in Fig. 4(3), although the SQL structure and column names are correctly forecasted, there is a mismatch with the actual raw data in the SQL.

P4: SQL Irregularity. Several models produce prediction results that do not conform to SQL syntax rules. For instance, as shown in Fig. 4(4), although the models accurately identify the "`JOIN`" module, they fail to specify the conditions for the "`JOIN`". This discrepancy leads to a divergence in execution accuracy between the predicted SQL and the gold SQL.



1. How many cars have more than 4 cylinders?
2. how many cars were produced in 1980?
3. In 1980, how many cars were made?
4. How many countries has more than 2 car makers ?

1. For the cars with 4 cylinders, which model has the largest horsepower?
2. For all of the 4 cylinder cars, which model has the most horsepower?

1. How many car models are produced by each maker ? Only list the count and the maker full name.
2. What are the countries having at least one car maker? List name and id.

1. What is the number of the cars with horsepower more than 150?
2. What is the minimum weight of the car with 8 cylinders produced in 1974 ?"
3. What is the horsepower of the car with the largest accelerate?"

Fig. 7. Exploring at the global-class level combing Projection View and SQL Similarity Component: (a) Examples of a user exploring input sentences freely, it is common for questions in the same category to exhibit correlations in their similarity metrics; (b) Potential error patterns in initial user inference can help users compare the differences between the predicted and ground truth SQL statements.

C. Case 1: Analyzing Error Patterns from Model Prediction

The first case conducted by *E1* involved an exploration of the database "car_1" within the Spider dataset. This was carried out in conjunction with our system, and a comparative analysis was made against the workflow of the original analytical model. When *E1* load in a *T2S* model and accesses MAVIDSQL, the system extracts the relevant model prediction data and add it to the existing dataset. Then, the system recalculates the feature extraction results and update the visualization views. As shown in Fig. 3(D), *E1* acquire a comprehensive understanding of the model's performance while also obtaining information on varying degrees of hardness. *E1* analyzed the projection views and model performance statics views to gain an overview from a global level to the class level. During the exploration, *E1* observed that in the "car_1" database, the predictions yielded satisfactory results for easy and medium tasks but underperformed in handling hard-level predictions.

E1 further investigated the projection view, focusing on exploring specific questions of interest. The projection view clusters semantically similar sentences together based on the SQL similarity measure, facilitating easy navigation and exploration of the input questions. This combination of interactive visualization techniques enables *E1* to gain insight into how well the model performs for different question types. As depicted in Fig. 7(1), *E1* employ the lasso tool to select several areas of query sentences, where sentences of the same category tend to cluster together. The corresponding model prediction performance is displayed in the SQL Similarity Component, Fig. 7(2). *E1* found that the predicted axis values predominantly cluster at or below five, whereas the actual gold axis values are mainly distributed at one, eight, and above. *E1* speculated the model's reduced similarity score, when compared to the gold SQL statement, may stem from inadequate prediction of certain words or the omission of key components in the SQL statement. *E1* further filtered the results in the SQL similarity view and performed instance-level comparisons by combining specific SQL comparison components and raw data views to verify false evidence.

As shown in Fig. 3(c2), *E1* observed in the SQL comparison view that the gold SQL contains a larger number of SQL keywords. The word cloud visualization demonstrates that SQL keywords , such as "`AS`", "`JOIN`", "`ON`", and "`WHERE`" constitute a substantial proportion of the visualization. *E1* integrating experience with analysis of raw data, observed that while the model accurately predicted certain database values, some predictions were erroneous. This inaccuracy was primarily due to a disregard for SQL syntax rules, specifically the failure to properly match the 'ON' condition during SQL joins (***P4***). Additionally, table names, such as "`car_names`" and "`cars_data`" and column names, such as "`model`" and "`makeid`" also occupy a considerable proportion in the word cloud. *E1* further explored this category of SQL statements by utilizing the Raw data view. The specific details related to the problem, including the predicted and ground truth SQL statements, along with the corresponding natural language questions, are displayed. As depicted in Fig. 3(e1), *E1* also noted that the model evaluation does not examine the values, leading to the replacement of the WHERE clause with "`1`" instead of leaving it unstandardized (***P3***). This is exemplified in the query "`AMC Hornet Sportabout (sw)`".

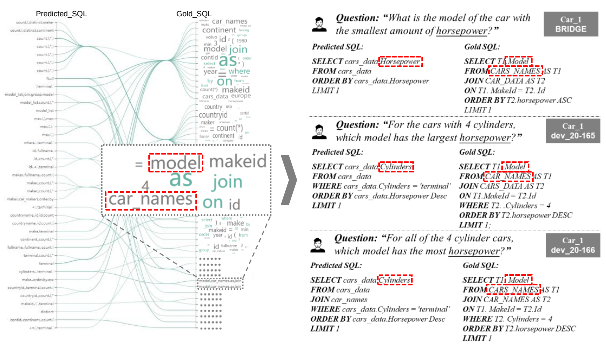
Furthermore, while the predicted SQL statement correctly identifies the table and column names, the lack of uniformity in the form of the predicted and gold SQL statements, including the absence of keywords such as "`AS`" and "`ON`", results in many SQL keywords remaining after comparison. Upon inspecting the original evaluation results for this example, *E1* found that the match accuracy was *True* while the execution accuracy was *False*. *E1* indicated that this inactive instance-level analysis can help distinguish and validate the reasons for inconsistencies between match and execution accuracy. In Fig. 3(e2), *E1* observed that it is demonstrated that the model has misunderstood the correspondence between "`car maker`" and the corresponding column name. "`American Motor Company`" should correspond to the "`full_name`" column in the "`car_makers`" table, but the model mistakenly predicted the "`maker`" column (***P2***). Further analysis revealed that cases with fewer unmatched tokens in the Gold SQL generally involve errors in value predictions or incorrect table and column names. *E1* noted that since the model metrics calculation does not check the values in SQL, some models often overlook this aspect, typically substituting with default values. This issue is a subsequent challenge in the *T2S* task.

E1 observed that our system significantly enhances the efficiency of analyzing model predictions through preliminary interactive exploration of extensive model prediction data. Traditional methods of analyzing model predictions involve directly comparing extensive textual information, including each token of the model's erroneously predicted SQL statements with the original data, which depends heavily on the developers' proficiency in SQL statements for accurate problem analysis. Our interactive tool greatly improves the efficiency of data analysis.

D. Case 2: Discovering Additional Cases from Existing Patterns

In this case, *E2* combined visualization and interactive techniques to explore instance level analysis. Guided by an overview from the global level exploration, the approach involves comparing groups of predicted and gold SQL statements to identify differences and similarities, thereby quickly discovering error patterns in SQL predictions.

E2 noted that instance-level analysis facilitates a more efficient and targeted approach to locating known errors and incorporating more cases into their model, thereby allowing users to conduct a fine-grained diagnosis of their models. In Fig. 8, an example of robustness issue mentioned in BRIDGE [1] is illustrated. Specifically, the model's inability to incorporate significant information from the utterance, even when the underlying logic is relatively straightforward, suggests that the model may have learned spurious correlations during training. In the example from BRIDGE, the model includes the "`Horsepower`" field in the "`SELECT`" clause, while the question specifically asks for "`the model of the car`". *E2* hopes to uncover additional cases through instance-level analysis using the MAVIDSQL system Firstly, *E2* locate the "`model`", "`car_names`", "`as`", "`join`" cluster in the SQL comparison view, and then conduct a detailed analysis by utilizing the raw data view. For instance, in the second and third case where the model predicts "`SELECT Cylinders`" while the question asks for "which model". *E2* suggest that improved modeling of compositionality in natural language may help reduce such errors. This could involve modeling the span structure of the language, as well as constructing interpretable grounding with the database schema (***P1***).



Question: *"What is the model of the car with the smallest amount of horsepower?"*

Predicted SQL:

```
SELECT cars_data.Horsepower
FROM cars_data
ORDER BY cars_data.Horsepower
LIMIT 1
```

Gold SQL:

```
SELECT T1.Model
FROM CARS_NAMES AS T1
JOIN CAR_DATA AS T2
ON T1. MakeId = T2. Id
ORDER BY T2.horsepower ASC
LIMIT 1
```

Question: *"For the cars with 4 cylinders, which model has the largest horsepower?"*

Predicted SQL:

```
SELECT cars_data.Cylinders
FROM cars_data
WHERE cars_data.Cylinders = 'terminal'
ORDER BY cars_data.Horsepower Desc
LIMIT 1
```

Gold SQL:

```
SELECT T1. Model
FROM CAR_NAMES AS T1
JOIN CARS_DATA AS T2
ON T1.MakeId = T2.Id
WHERE T2. .Cylinders = 4
ORDER BY T2.horsepower DESC
LIMIT 1;
```

Question: *"For all of the 4 cylinder cars, which model has the most horsepower?"*

Predicted SQL:

```
SELECT cars_data.Cylinders
FROM cars_data
JOIN car_names
WHERE cars_data.Cylinders = 'terminal'
ORDER BY cars_data.Horsepower Desc
LIMIT 1
```

Gold SQL:

```
SELECT T1. Model
FROM CARS_NAMES AS T1
JOIN CAR_NAMES AS T2
ON T1. MakeId = T2.Id
WHERE T2. Cylinders = 4
ORDER BY T2.horsepower DESC
LIMIT 1
```

Fig. 8. Building upon the established robustness error analysis, instance-level exploration can be used to identify additional relevant cases and diagnosis models accordingly.

E2 summarized that the SQL comparison view facilitates rapid validation of the hypotheses presented in the global-class level through the projection and SQL similarity views. This approach provides users with an in-depth understanding of the model's strengths and limitations and can inform further refinements to the model.

Furthermore, *E2* hopes to identify issues such as lexical misunderstandings of the model and SQL irregular through this system. Specifically, he pinpoints problems within the SQL Comparison View, such as finding operators like "`MIN`", or instances containing numerous JOINs and aliases. This is to be followed by conducting a fine-grained analysis in combination with the Raw Data View.

E. User Study

We conduct a user study to further evaluate the effectiveness and usability of MAVIDSQL in facilitating the interpretation and diagnosis of text-to-SQL tasks. Moreover, we also aim to further evaluate whether the visual design can effectively improve the identification and discovery of error patterns.

1) Participants: we invited 12 participants (P1-P12, seven females and five males, age 22-29, μ =25, σ =2.38) to perform the user evaluation. All 12 participants possess varied background knowledge and are currently pursuing further studies at the graduate level. P1-P5 are primarily focused on visual analysis, while P6-P8 have experience in developing text-to-SQL models. P9-P11 specialize in Natural Language Processing, and Participant P12 concentrates on computer vision. All participants have experience in designing and debugging deeplearning models. None of the 12 participants are co-authors of this paper.

2) Procedures: We first provided a brief introduction of the research background, including the research motivation and the exploration workflow. Subsequently, we collected demographic data from the participants and obtained their informed consent for recording their activities and outcomes for subsequent analysis. Then we introduced the system's features and demonstrated the case study mentioned above. We have designed five tasks for participants, guiding them in exploring the distribution patterns of Text-to-SQL predictions and explaining the reasons for model failure. After participants concluded their exploration, we invited them to complete a post-interview questionnaire featuring 5-point Likert scale questions, ranging from 1 (Strongly Disagree) to 5 (Strongly Agree), to collect their feedback on MAVIDSQL. As illustrated in Table. I, the questionnaire primarily evaluates the effectiveness, visual design, and usability of MAVIDSQL. The results and feedback have been thoughtfully summarized.

3) Results: Drawing upon the user ratings from the questionnaire and feedback obtained during the interviews, we evaluate the effectiveness, visual design and usability of MAVIDSQL.

Effectiveness: Most participants indicated that the system facilitated their understanding of the dataset's overview and the prediction outcomes for Text-to-SQL tasks. P6 commented, "I can easily identify

	Question	Score	Score Distribution
	Effectiveness		
Q1	The system can help me understand the overview of the dataset and the prediction results of Text-to-SQL tasks.	4.50 ± 0.49	5 7
Q2	The system can help me identify the error patterns in Text-to-SQL tasks.	4.00 ± 0.56	2 8 2
Q3	The system can help me explore model prediction at multiple levels.	3.92 ± 0.76	4 5 3
Q4	The system can help me analyze the predictions more efficiently compared to the traditional method of directly comparing extensive textual information.	4.17 ± 0.76	2 6 4
	Visual Design		
Q5	The Projection View facilitates my comprehension of the inherent distribution characteristics in the input queries.	3.58 ± 0.76	1 4 6 1
Q6	The Statistics View facilitates my understanding of the distribution characteristics within model prediction outcomes.	4.50 ± 0.65	1 4 7
Q7	The Control Panel and SQL Comparison View facilitate my exploration and discovery of potential predictive error patterns.	4.00 ± 0.71	3 6 3
Q8	The Raw Data View facilitates my validation of specific error instances.	4.41 ± 0.64	1 5 6
Q9	The overall system is intuitive and easy to understand.	4.33 ± 0.62	1 6 5
	Usability		
Q10	It is easy to learn and use the system.	3.91 ± 0.76	4 5 3
Q11	I think it is useful to use this system to explore the prediction of Text-to-SQL tasks.	4.00 ± 0.82	1 8 3
Q12	I would use this system to diagnose errors of Text-to-SQL tasks in the future.	4.00 ± 0.82	4 4 4
Q13	I would like to recommend this system to others who are working on explore Text-to-SQL tasks.	4.67 ± 0.47	4 8

🟥 1 (Strongly Disagree) 🟧 2 ▨ 3 🟦 4 ■ 5 (Strongly Agree)

TABLE I
USER STUDY ON EFFECTIVENESS (Q1-Q4), VISUAL DESIGN (Q5-Q9), AND USABILITY (Q10-Q13) OF MAVIDSQL. SCORE (MEAN ± STD) FOR EACH QUESTION ARE REPORTED. SCORE DISTRIBUTION MAPS OUT THE SCORING OUTCOMES FOR EACH PARTICIPANT.

the primary error patterns in MAVIDSQL." As shown in Table. I, participants agreed that MAVIDSQL can help users analyze the predictions more efficiently compared to the traditional method of directly comparing extensive textual information. P5 commented that the visualization is clear and insightful, allowing model developers to analyze model predictions systematically and hierarchically with MAVIDSQL, without the need to directly handle extensive textual data. This approach facilitates a more rapid analysis of model results. Moreover, several participants expressed a preference for detailed comparisons of raw data, finding that it aligned with their perception and enabled them to validate hypotheses based on practical experience.

Visual Design: Based on the results of the evaluation of visual design, it is observed that the most of the participants agreed that the overall visual design of MAVIDSQL is intuitive and easy to understand. Sometimes participants lacked familiarity with the database schema. For instance, P5 found the depiction of the projection view unclear, advocating for additional demonstration of model input queries to enhance user intuition. The participants agreed that the statistics view is the most intuitive and easy to understand, as shown in Table. I. Regarding the control panel and SQL comparison view, participants believe that aggregated comparisons could help users understand the distribution of model prediction outcomes, though the learning curve is a bit steep.

Usability: The participants thought that the workflow of our system is intuitive and the interface is user-friendly. They also appreciated the system's filtering and interaction features, which facilitated a more flexible exploration of prediction instances of interest. P6 suggested displaying more information within the Raw Data View to support users in diagnosing model errors more effectively. Lastly, all the participants expressed their willingness to recommend the system to others who are working on explore text-to-SQL tasks in the future. P11 stated, "Exploring the predict and gold SQL pairs in the system was immersive, and I enjoyed the process."

F. Expert Interviews

In order to further evaluate the effectiveness and practicality of MAVIDSQL, we obtained feedback through individual interviews with the previously mentioned three domain experts (*E1*, *E2*, *E3*). Prior to the interviews, none of the experts had experience with the system. We first presented an overview of the system's background and design. Subsequently, we requested the experts to engage with MAVIDSQL, exploring the model's prediction on two distinct databases. Following a 40-minute exploration, we gathered their feedback regarding the system's workflow, design, application scenarios, and suggestions for improvements.

System workflow. All the experts confirmed the effectiveness of the system workflow of MAVIDSQL in providing explanations for *T2S* tasks. They indicated that their typical approach to model evaluation involves relying on performance matrices and conducting instance-level analyses individually. However, this approach often lacks comprehensive details and does not facilitate in-depth support. Analyzing individual instances of model predictions is a tedious process, consuming substantial time and requiring manual summarization based on domain knowledge. Our system enhances this approach by providing both global-class level and instance-level explanations, thereby facilitating a comprehensive and systematic understanding of model predictions and identifying points where the model is prone to failure. *E1* and *E3* praised that the interactive exploration tools (i.e., lasso, brushing) are impressive and useful for analyzing *T2S* tasks and discovering error patterns. *E2* mentioned that if he finds the model predictions ineffective in identifying database table and column names, he might consider optimizing schema linking to enhance the model's association with the database structure. *E3* added that the aggregation in the SQL comparison group helps to generalize the model error patterns. *E1* summarized that the system assisted users in discovering interesting insights into the models. For example, she was surprised to find that certain models unexpectedly used default values to fill in the SQL value section.

Visual design and interactions. Overall, the experts concurred that the visualizations are both efficacious and comprehensible, with fluent interactions. The SQL Comparison View is highly favored by experts for providing a quick overview of large-scale SQL pairs. The design of the Model Performance Statistics View is also highly regarded by the experts. *E3* really liked the SQL Similarity component for providing an overview of the overall distribution of model predictions. *E1* thought the scatter plot was very intuitive, and the interactions such as lasso and brush are really helpful for the exploration of a large amount of data. Moreover, she valued how the raw data view restored the original data, enhancing the credibility of the exploration results. Nevertheless, *E1* and *E2* mentioned that in the Raw Data View, they still need to spend several time linking the content in the questions with the corresponding SQL. They suggested that color-coding entity fields would greatly facilitate their review of these contents.

Improvements The experts offered constructive suggestions for improvements. During exploration in the Raw Data View, *E2* pointed out: "*Currently, I still need to spend some time connecting the content in the Question to its corresponding SQL. If entities could be mapped out with color coding, this would greatly facilitate my review of these contents.*" *E3* requested that in the projection view, mapping more information about queries could be implemented, as this might aid in discovering additional patterns. For a more thorough review, *E1* suggested that the system could include multi-model comparisons to help uncover more error patterns. At the same time, during the exploration by *E1* and *E2*, it was observed that some significant model errors are caused by the SQL structure. They recommended that the system should support filtering high-frequency errors.

VII. Discussions and Future Work

In this work, we presented a technique for understanding and exploring *T2S* tasks using MAVIDSQL. We preliminarily validated the effectiveness of the visual interface, MAVIDSQL, through two use scenarios and an expert interview.

MAVIDSQL can be applied to analyze various kinds of text-to-SQL models and different datasets. As a model-agnostic visual analytics workflow, the system has been designed to facilitate the discovery and diagnosis of model errors both globally and at the instance level. By focusing on the model's inputs and outputs, users can gain insight into its decision-making process. Additionally, the system's adaptable framework can be applied to various machine learning models, regardless of their specific algorithms or techniques. Despite the remarkable performance achieved by large language models in various natural language tasks, such as semantic parsing and generation, further fine-tuning is required in specific vertical domains. Moreover, the high training cost of large models hinders their flexibility in deployment. Therefore, there is still research value in exploring the development of customizable deep learning language models for specific domain tasks, which can be quickly deployed and responsively updated. However, it may not be suitable for prior classification exploration of text-to-SQL prediction results as it is designed to facilitate free-form exploration tasks. Moreover, the system cannot directly support and apply datasets that involve multi-turn conversations.

Our scalability analysis on the Spider dataset revealed that MAVIDSQL is capable of analyzing up to thousands of questions and their corresponding SQL statements. We reduce visual clutter in our system by binding SQL comparison data with the same category using a method based on sorting and category aggregation sampling. In SQL Comparison View, we present a method for grouping SQL tokens with common prefixes. In the SQL Comparison View, the more item a SQL group contains, the higher it is positioned on the axes. As the volume of predicted SQL data escalates, error patterns are likely to become more concentrated, with principal inaccuracies manifesting more distinctly within the views. However, there may still be issues with excessive visual clutter and cognitive burden in the word cloud and SQL similarity parallel coordinate plot visualization, despite our efforts to bind data with the same category using sorting and category aggregation.

In our work, additional error patterns not fully delineated, such as logic errors, robustness issues, and missing commonsense, as mentioned in previous works [1], [47]. If the tokens in the similar parts of the SQL structure are incorrect, they can be explored through our system. Conversely, our system still exhibits evident deficiencies in cases of significant differences in SQL structure. As mentioned earlier, the current version of MAVIDSQL only supports analysis of single-turn dialogues. When facing multi-turn dialogues and complex conversations, our system may exhibit potential insufficiencies in adaptability to dynamic data and handling of complex queries. Multi-turn dialogues are prevalent in many real-world applications, making the development of methods for analyzing and visualizing such dialogues an important direction for future work.

While our system has shown promising results in detecting and diagnosing errors, including patterns P1-P4 mentioned in Section VI-B2, it is imperative to recognize that there are still exist error patterns yet to be uncovered. Our system employs algorithms for SQL result alignment and SQL group comparison to identify similarities between predicted and Gold SQL structures despite differences in specific tokens. This approach aids in error pattern categorization at both class and instance levels, enhancing model refinement efforts. However, with complex query situations, such as nested queries and intricate multi-table queries, the potential structural divergence between model predictions and the ground truth in SQL may be uncontrollable. This issue complicates the process of identifying error patterns through SQL comparison. In future work, semantic parsing algorithms need to be introduced to further identify the syntactical structure of SQL, thereby enhancing its capability to analyze complex model predictions. Therefore, future work should explore techniques for identifying errors in the ground truth labels and incorporating these errors into the analysis process.

In addition, experts have offered numerous valuable insights regarding visual analysis and interactive exploration which subsequently assisted in the refinement of our work. To improve the efficiency of error detection during system interaction, the introduction of error model retrieval is recommended. To enhance user understanding and alleviate over-plotting issues in parallel coordinates plots (PCP), incorporating additional visual mappings in SQL comparison views is suggested. In exploring and analyzing diagnostic results of the model, it is recommended to include additional semantic hierarchical information. This can be achieved by introducing association analysis methods that are closely aligned with SQL semantics and strengthening the token link between input queries and output SQL.

Finally, we aim to improve the customization options for the Projection View and SQL Comparison View, allowing users to define their own metrics, including personalized similarity metrics. This will provide better insights for researchers and practitioners to comprehend and interpret these models.

VIII. Conclusion

In this paper, we present MAVIDSQL, a visual analytics system to help model developers and users understand and diagnose text-to-SQL model prediction results. MAVIDSQL comprises four visualization components: The Projection View allows for the display of multiple 2D projections of the input sentence according to similarity summarized from different perspectives enabling users to extract potential semantic relationships. The SQL Comparison View allows users to compare and analyze the differences between the model-generated SQL and the ground truth at a class-level. The Model Performance Statistics View displays the evaluation results of the models, categorized by the performance metric ESM. The Raw data View provides users with access to the raw data, allowing for detailed analysis and exploration. All four visualization components are linked together to support users in analyzing *T2S* tasks from multiple perspectives simultaneously and identifying common error patterns in *T2S* prediction results. Two case studies demonstrate the effectiveness and usability of our system *T2S*. While large language models like T5 [48] and GPT-4 [49] have shown promising performance in various tasks, the feasibility of using lightweight language models for specific natural language sub-domains, such as text-to-SQL, has not been fully explored. Future research in this direction can provide insights into cost-effective and rapid deployment options for such applications.

Acknowledgments

The work is partly supported by the Zhejiang Provincial Natural Science Foundation of China (LR23F020003, LTGG23F020005), National Natural Science Foundation of China (62372411, 62036009), and Fundamental Research Funds for the Provincial Universities of Zhejiang (RF-B2023006). Guodao Sun is the corresponding author.

References

[1] X. V. Lin, R. Socher, and C. Xiong, "Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 4870–4888.

[2] K.-M. George and G. Koutrika, "A Survey on Deep Learning Approaches for Text-to-SQL," *The VLDB Journal*, vol. 32, no. 4, pp. 905–936, 2023.

[3] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang, V. Zhong, B. Wang, C. Li, C. Boyle, A. Ni, Z. Yao, D. Radev, C. Xiong, L. Kong, R. Zhang, N. A. Smith, L. Zettlemoyer, and T. Yu, "UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 602–631.

[4] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.

[5] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks," *arXiv:1606.07461*, 2017.

[6] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush, "Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models," *arXiv:1804.09299*, 2018.

[7] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu, "Understanding Hidden Memories of Recurrent Neural Networks," in *Proceedings of IEEE Conference on Visual Analytics Science and Technology*, 2017, pp. 13–24.

[8] S. Zhu, G. Sun, Q. Jiang, M. Zha, and R. Liang, "A survey on automatic infographics and visualization recommendations," *Visual Informatics*, vol. 4, no. 3, pp. 24–40, 2020.

[9] Q. Jiang, G. Sun, Y. Dong, and R. Liang, "DT2VIS: A focus+context answer generation system to facilitate visual exploration of tabular data," *IEEE Computer Graphics and Applications*, vol. 41, no. 5, pp. 45–56, 2021.

[10] Q. Jiang, G. Sun, T. Li, J. Tang, W. Xia, S. Zhu, and R. Liang, "Qutaber: Task-based exploratory data analysis with enriched context awareness," *Journal of Visualization*, pp. 1–1, 2024.

[11] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning," *arXiv:1709.00103*, 2017.

[12] X. Xu, C. Liu, and D. Song, "SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning," *arXiv:1711.04436*, 2017.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv:1907.11692*, 2019.

[15] T. Yu, C.-S. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher, and C. Xiong, "GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing," *arXiv:2009.13845*, 2021.

[16] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8413–8426.

[17] B. Bogin, J. Berant, and M. Gardner, "Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4560–4565.

[18] B. Bogin, M. Gardner, and J. Berant, "Global Reasoning over Database Structures for Text-to-SQL Parsing," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, 2019, pp. 3659–3664.

[19] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7567–7578.

[20] W. Hwang, J. Yim, S. Park, and M. Seo, "A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization," *arXiv:1902.01069*, 2019.

[21] D. Choi, M. C. Shin, E. Kim, and D. R. Shin, "RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases," *Computational Linguistics*, vol. 47, no. 2, pp. 309–332, 2021.

[22] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev, "SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1653–1663.

[23] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang, "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation," *arXiv:1905.08205*, 2019.

[24] T. Scholak, N. Schucher, and D. Bahdanau, "PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 9895–9901.

[25] A. Liu, X. Hu, L. Wen, and P. S. Yu, "A Comprehensive Evaluation of ChatGPT's Zero-Shot Text-to-SQL Capability," *arXiv.2303.13547*, 2023.

[26] W. Liu, J.-L. Qiu, W.-L. Zheng, and B.-L. Lu, "Comparing Recognition Performance and Robustness of Multimodal Deep Learning Models for Multimodal Emotion Recognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 715–729, 2022.

[27] X. Li, Y. Hou, P. Wang, Z. Gao, M. Xu, and W. Li, "Trear: Transformer-Based RGB-D Egocentric Action Recognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 1, pp. 246–252, 2022.

[28] F. Cheng, D. Liu, F. Du, Y. Lin, A. Zytek, H. Li, H. Qu, and K. Veeramachaneni, "VBridge: Connecting the Dots Between Features and Data to Explain Healthcare Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 378–388, 2022.

[29] S. Liu, Z. Li, T. Li, V. Srikumar, V. Pascucci, and P.-T. Bremer, "NLIZE: A Perturbation-Driven Visual Interrogation Tool for Analyzing and Interpreting Natural Language Inference Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 651–660, 2019.

[30] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, 2017.

[31] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 364–373, 2019.

[32] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit, "ConfusionFlow: A Model-Agnostic Visualization for Temporal Analysis of Classifier Confusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1222–1236, 2022.

[33] Z. Jin, X. Wang, F. Cheng, C. Sun, Q. Liu, and H. Qu, "ShortcutLens: A Visual Analytics Approach for Exploring Shortcuts in Natural Language Understanding Dataset," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2023.

[34] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu, "OoD-Analyzer: Interactive Analysis of Out-of-Distribution Samples," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3335–3349, 2021.

[35] T. Nora Raju, P. A. Rahana, R. Moncy, S. Ajay, and S. K. Nambiar, "Sentence Similarity - A State of Art Approaches," in *Proceedings of International Conference on Computing, Communication, Security and Intelligent Systems*, 2022, pp. 1–6.

[36] H. He and J. D. Choi, "The Stem Cell Hypothesis: Dilemma behind Multi-Task Learning with Transformer Encoders," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5555–5577.

[37] T. Yousef and S. Janicke, "A Survey of Text Alignment Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1149–1159, 2021.

[38] S. Jänicke, A. Geßner, G. Franzini, M. Terras, S. Mahony, and G. Scheuermann, "TRAViz: A Visualization for Variant Graphs," *Digital Scholarship in the Humanities*, vol. 30, no. 1, pp. 83–99, 2015.

[39] B. Gipp, N. Meuschke, and J. Beel, "Comparative Evaluation of Text- and Citation-based Plagiarism Detection Approaches Using Guttenplag," in *Proceedings of the Annual International ACM/IEEE Joint Conference on Digital Libraries*, 2011, pp. 255–258.

[40] S. Jänicke and D. J. Wrisley, "Interactive Visual Alignment of Medieval Text Versions," in *Proceedings of IEEE Conference on Visual Analytics Science and Technology*, 2017, pp. 127–138.

[41] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf, "An Edge-Bundling Layout for Interactive Parallel Coordinates," in *Proceedings of IEEE Pacific Visualization Symposium*, 2014, pp. 57–64.

[42] R. Zhong, T. Yu, and D. Klein, "Semantic Evaluation for Text-to-SQL with Distilled Test Suites," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 396–411.

[43] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, "UpSet: Visualization of Intersecting Sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1983–1992, 2014.

[44] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit, "LineUp: Visual Analysis of Multi-Attribute Rankings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2277–2286, 2013.

[45] G. Sun, H. Wu, L. Zhu, C. Xu, H. Liang, B. Xu, and R. Liang, "VSumVis: Interactive Visual Understanding and Diagnosis of Video Summarization Model," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 4, pp. 1–28, 2021.

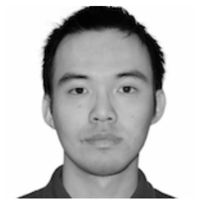
[46] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3911–3921.

[47] B. Hui, R. Geng, L. Wang, B. Qin, Y. Li, B. Li, J. Sun, and Y. Li, "S$ˆ2$SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers," in *Proceedings of the Association for Computational Linguistics*, 2022, pp. 1254–1262.

[48] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[49] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of Artificial General Intelligence: Early experiments with GPT-4," *arXiv:2303.12712*, 2023.

Jingwei Tang is currently a Ph.D. student in the College of Computer Science and Technology, Zhejiang University of Technology, HangZhou, China. He received his B.E. degrees in communication engineering from Zhejiang University of Technology. His main research interests are data mining, visual analytics of network and information visualization.



Guodao Sun is a professor at the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. He received the B.Sc. in computer science and technology, and Ph.D. degree in control science and engineering both from Zhejiang University of Technology. His main research interests are urban visualization, visual analytics of social media, and information visualization.

Jiahui Chen is currently working toward the MS degree with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. His main research interests are data mining and information visualization.

Gefei Zhang is currently a PhD student at the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. She received Bachelor of Education in education technology in College of Educational Science and Technology, Zhejiang University of Technology. Her main research interests include information visualization and educational data mining.

Baofeng Chang received Ph.D. degree in Computer Science and Technology at the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. He received B.Sc. degree in Automation in the College of Information Engineering, Zhejiang University of Technology. He is currently a researcher in Zhejiang Airport Innovation Institute, Hangzhou, China. His main research interests are dynamic network visualization, traffic information visualization and temporal sequence visualization.

Haixia Wang received her Ph.D. degree in 2012 from Nanyang Technological University, Singapore. She is currently a professor at Zhejiang University of Technology. Her research interests include image processing and pattern recognition.

Ronghua Liang received the Ph.D. in computer science from Zhejiang University. He worked as a research fellow at the University of Bedfordshire, UK, from April 2004 to July 2005 and as a visiting scholar at the University of California, Davis, US, from March 2010 to March 2011. He is currently a Professor of Zhejiang University of Technology, China. His research interests include Visual Analytics and Computer Vision.