Beyond Read-Only: Crafting a Comprehensive Chinese Text-to-SQL Dataset for Database Manipulation and Query

Xi Chen[1], Jinguo You[1,2]*, Kun Li[3], Xiang Li[1]

[1]Kunming University of Science and Technology, Kunming, China
[2]Yunnan Key Laboratory of Artificial Intelligence, Kunming, China
[3]Huawei, China
{chenxi328, li_xiang}@stu.kust.edu.cn, jgyou@126.com, likun75@huawei.com

Abstract

Text-to-SQL aims to convert natural language into structured query language, which is a challenging task. Current research focuses mainly on read operations and ignores other aspects of database operations such as create, update, and delete operations. The benchmark datasets as well as models that have been proposed also fail to cover these operations, limiting the development and practical applications in the field. To bridge this gap, we propose CRUDSQL, a large-scale cross-domain single-table CRUD operations Chinese Text-to-SQL dataset. The dataset contains 10,000 question/SQL pairs involving 625 tables from different domains. To support further research on this dataset, we also propose a baseline method, CRUDParser, which employs a two-phase approach based on BERT and T5 for SQL generation and incorporates two strategies, value matching, and value prompting, for interacting with databases to further improve the performance. The experimental results show that the new operation types bring different challenges for future research, and our approach achieves 67.08% and 83.8% exact set matching accuracy under both read and delete operations in the test set, but only 49.6% and 61.8% under create and update operations. We believe that the proposal of CRUDSQL as well as CRUDParser can provide new directions and possibilities for research and practical applications in the field of Text-to-SQL. The dataset is published at https://github.com/bizard-lab/CRUDSQL.

1 Introduction

The Text-to-SQL task aims to transform natural language questions into corresponding SQL query statements, and researchers have applied this technique to relational databases as natural language interface to database (NLIDB) to help users interact with databases more easily. However, in the early days the main methods used were based on keywords, pattern matching, syntax tree parsing and grammar rules. This is limited by the rules and patterns defined, cannot handle ambiguous natural language questions, is difficult to scale and is not suitable for complex queries.

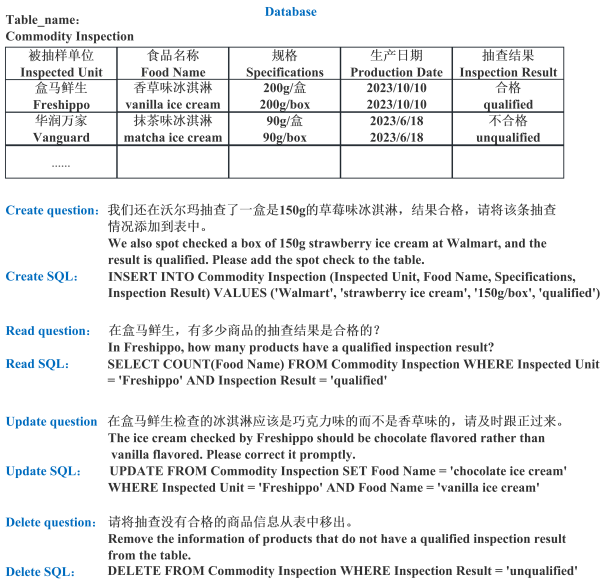

Figure 1: An example that introduces Text-to-SQL for CRUD operations.

With the release of WiKiSQL(Zhong et al., 2017) and Spider(Yu et al., 2018c), two large-scale Text-to-SQL datasets, the field has gained more and more attention and development. WiKiSQL is constructed based on Wikipedia and contains 24,241 tables, 80,654 natural language questions, and corresponding SQL queries. Each of its databases has only one table, does not need to consider primary and foreign keys, and the form of SQL to be predicted is relatively simple, and SeaD(Xu et al., 2022) currently achieves the highest accuracy on this dataset, with 93%. Spider contains 10,181 natural language question-SQL pairs covering 138 different domains, which is a greater

*Corresponding author

Findings of the Association for Computational Linguistics: NAACL 2024, pages 3383–3393

difficulty enhancement than WiKiSQL which involves the correlation between multiple tables and the generation of more complex SQL. Currently, DAIL-SQL(Gao et al., 2023) achieves 86.6% accuracy on this dataset.

However, current datasets and research efforts focus on read operations and ignore create, update, and delete operations. To fill this gap, this paper proposes CRUDSQL, a large-scale cross-domain single-table CRUD operations Chinese Text-to-SQL dataset. The dataset contains 10,000 questions and their corresponding SQL queries involving 625 tables from different domains, in which the distribution of create, read, update, and delete is uniform. Figure 1 shows an example of data from CRUDSQL consisting of a table and questions oriented to the table about the CRUD operations and the corresponding SQL statements. To the best of our knowledge, CRUDSQL is the first Text-to-SQL dataset that contains all four operation types of create, read, update, and delete at the same time.

In addition, we propose CRUDParser, a model that incorporates BERT(Kenton and Toutanova, 2019) and T5(Raffel et al., 2020) for two-stage SQL generation as a baseline approach for this dataset. The first stage is sketch-based slot filling, where BERT predicts the SQL type corresponding to the question, as well as the SQL column names, keywords, operators, and aggregation functions, and fills them into a predefined SQL sketch. The SQL sketch is then used as the basis for interaction with the database to improve the accuracy of the final SQL generated. There are two types of interactions, one is value matching, which is used to improve the conditional value of the WHERE clause in the SQL statement, and the other is value prompting, which is used to optimize the values that are inserted into the database or the values that need to be updated. The second stage is the sequence-based generation, mainly used in update and create operations, where T5 can generate the final SQL based on SQL sketches as well as prompt words.

The experimental results show that the create and update operations bring new challenges to the Text-to-SQL task, and the accuracy of these two operation types on the test set is only 49.6% and 65.6% even with the use of pre-trained language models as well as methods that have performed well in the past. Although our value matching and value prompting strategies are effective, there is still a lot of room for improvement offered for further research.

The key contributions of this paper are summarized as follows:

(1) We propose CRUDSQL, a large-scale cross-domain single-table CRUD operations Chinese Text-to-SQL dataset. The dataset contains 10,000 question/SQL pairs involving 625 tables from different domains.

(2) We present a baseline model CRUDParser and its experimental benchmark results for this dataset, containing Bert and T5 for two-stage SQL generation.

(3) Further, two refinement strategies: value matching and value prompting are introduced to enhance conditional value matching and generate normalized values for update and create operations respectively.

2 Related Work

2.1 Existing Datasets

Over the past few decades, numerous datasets have been instrumental in propelling the development of Text-to-SQL systems. Early datasets such as ATIS (Price, 1990; Dahl et al., 1994), centered around flight reservations, GeoQuery(Zelle and Mooney, 1996), focused on U.S. geographical information, and Restaurants(Tang and Mooney, 2000), detailing restaurant information in Northern California, were all confined to singular domains.

In recent years, datasets in this domain have primarily been categorized into three classes: single-table queries, cross-table joint queries, and multi-turn dialogue queries. Single-table query datasets involve SQL operations predominantly focused on aggregation, comparison, and selection, exhibiting lower complexity. Examples include WiKiSQL(Zhong et al., 2017), WiKiTableQuestions(Pasupat and Liang, 2015), and TableQA(Sun et al., 2020).Cross-table joint query datasets introduce clauses such as JOIN, GROUP BY, ORDER BY, HAVING, INTERSECT, UNION, and LIKE, significantly elevating the complexity of SQL and aligning it more closely with real-world scenarios. Examples encompass Spider(Yu et al., 2018c), Spider-DK(Gan et al., 2021b), Spider-Syn(Gan et al., 2021a) KaggleDBQA(Lee et al., 2021), CSpider(Min et al., 2019), DUSQL(Wang et al., 2020), and BIRD(Li et al., 2023b). Multi-turn dialogue query datasets necessitate models capable of understanding and processing successive user queries while considering context and historical dialogue

information. Instances include SParC(Yu et al., 2020), CoSQL(Yu et al., 2019), CHASE(Guo et al., 2021), and SeSQL(Huang et al., 2022).

It's worth noting that current datasets solely focus on read operations. Hence, we've introduced CRUDSQL, a dataset encompassing four fundamental types of operations: create, read, update, and delete. Table 1 provides an overview comparison between CRUDSQL and other Text-to-SQL benchmark datasets.

2.2 Existing Text-to-SQL Approaches

Sketch-based slot-filling approaches Using sketch-based slot-filling approaches, complex query logic is streamlined into a series of basic operations, simplifying neural network predictions and ensuring correct syntax when generating SQL queries. For instance, SQLNet(Xu et al., 2017) employs a predefined sketch of the SQL query, breaking down the required SQL query into the SELECT and WHERE clause segments, predicting slots within these segments (such as column, operator, aggregation, and value). Similarly, approaches like Coarse2Fine(Dong and Lapata, 2018), TypeSQL(Yu et al., 2018a), SyntaxSQLNet(Yu et al., 2018b), SQLOVA(Hwang et al., 2019), X-SQL(He et al., 2019), HydraNet(Lyu et al., 2020), RYANSQL(Choi et al., 2021), and CatSQL(Fu et al., 2023) adopt similar strategies.

Sequence-based approaches Another common approach involves sequence generation, relying on encoder-decoder pre-trained language models such as T5(Raffel et al., 2020), BART(Lewis et al., 2020), and similar architectures. These methods directly transform natural language queries into SQL queries through end-to-end sequence generation. For instance, a novel encoder-decoder framework is proposed by Cai et al. (2018). During the encoding phase, the neural network identifies and maintains semantic information of natural language questions. In the decoding phase, based on the neural network's hidden states, it generates a new sequence in another language. Similarly, approaches like PICARD(Scholak et al., 2021), RASAT(Qi et al., 2022), RESDSQL(Li et al., 2023a), SCPROMPT(Gu et al., 2023a).

LLM-based approaches In recent years, with the emergence of large-scale language models such as GPT-3(Brown et al., 2020) and PaLM(Chowdhery et al., 2023), more and more researchers have turned their attention to the impressive reasoning capabilities and domain-generalization abilities of these models. Liu and Tan (2023) proposed the use of Chain of Thought (CoT)(Wei et al., 2022) to activate the reasoning capabilities of Large Language Models (LLMs) in text-to-SQL tasks. This method guides models to decompose complex tasks into subtasks and designs prompts based on task characteristics to induce LLMs to generate useful chains of reasoning. C3(Dong et al., 2023) introduced three prompt paradigms—clear prompts, prompt correction, and consistent outputs—to enhance zero-shot performance. Furthermore, approaches like ZERONL2SQL(Gu et al., 2023b) combine pre-trained language models with large-scale language models. They first use pre-trained language models to generate SQL sketches and then utilize large-scale language models to fill in the complete SQL queries, achieving optimal zero-shot NL2SQL performance.

3 Dataset Construction

3.1 Data Collection

We constructed the CRUDSQL benchmark by sampling 625 tables from the TableQA[1], encompassing various domains such as stocks, real estate, commodities, and schools. Subsequently, we conducted data cleaning on the table data. This involved manual modifications addressing several issues, including 1) Ambiguous time data that wasn't explicitly transformed into date-time formats. 2) Database values mixed with database column names. 3) Mismatches between database values and column types. Additionally, we retained a subset of original questions-SQL query pairs that originally corresponded to these 625 tables.

3.2 Data Annotation

We assembled an annotation team consisting of three master's degree students and three undergraduate students in the computer science program who were asked to pose four questions about creation, reading, updating, and deletion to each table and manually annotate the corresponding SQL.

The SQL template for annotation adopts the same style as TableQA and WiKiSQL, which divides the SQL into several important parts to be stored in JSON form, including sel parts, aggregation functions, logical operators connecting multiple conditions, and conditional expressions for WHERE clauses. In addition, in order to extend the single

[1] https://github.com/ZhuiyiTechnology/TableQA

Dataset	Language	#DB	#Table/DB	#Pairs	Create	Read	Update	Delete
CRUDSQL	Chinese	625	1	10000	✔	✔	✔	✔
WiKiSQL	English	24241	1	80654	✘	✔	✘	✘
TableQA	Chinese	5291	1	49974	✘	✔	✘	✘
Spider	English	200	5.1	10181	✘	✔	✘	✘

Table 1: Comparisons of existing Text-to-SQL benchmark datasets.

read statement to create, update, and delete statements, we also added SET expressions for update operations as well as categorized identifiers for the four types of operational SQL in the annotation templates. It is worth noting that the "conds" field, which was originally used to represent WHERE clauses, represents not only the WHERE part of the read, update, and delete operations, but also the columns and their corresponding values inserted in the create operation. This approach can greatly simplify the annotation task, as well as ensure the consistency of the annotated SQL templates.

3.3 Data Review

In order to ensure the quality of the annotated data, two forms of review were adopted. The first form of review was primarily done by first checking with each other by the students to make sure that the questions posed by each other were clear and that the questions matched the annotated SQL. The annotated question-SQL pairs were then given to SQL experts for evaluation. The second form of review is mainly through the review script written, which is responsible for monitoring the proportion of different types of SQL in the annotated samples, the distribution of aggregation functions, the distribution of the number of conditions, the proportion of values that need to be linked, etc., and checking whether there are any problems in the SQL templates of the annotated questions, e.g., size comparisons and addition/subtraction operations are performed on the values which are not of numeric types, and the subscripts of the selected columns are out of the ranges of the columns in the table.

3.4 Data Statistics and Analysis

Overall Statistics CRUDSQL includes 625 tables and 10,000 question-SQL pairs, and the SQL types include create, read, update, and delete operations, Table 2 shows the statistics of the data division.

We separately counted the length of the question in CRUDSQL, the length of the SQL (table name, column name, keywords, operators, aggregation functions, and number of values), and the percentage of SQL that requires value joins, as shown in Figure 2.

Split	DB	Pair	Create	Read	Update	Delete
Train	440	7040	1760	1760	1760	1760
Dev	60	960	240	240	240	240
Test	125	2000	500	500	500	500

Table 2: Dataset split statistics.

In CRUDSQL, each question corresponds to only one table, where the SQL for read operations is consistent with the TableQA difficulty, with support for aggregation functions including MIN, MAX, AVG, SUM, MIN, and COUNT, and conditional operators including >, <, ==, and ! =. The SQL for update operations also includes the arithmetic operators +, and -.

Value Linking Value linking refers to the process of linking or mapping specific parts of a natural language query to actual values in the database. For example, if the query mentions "December 12, 2012", the actual value stored in the database may be "2012-12-12". So when we construct the SQL, we need to replace the corresponding conditional value with the actual value in the database. Situations that require value linking operations often include abbreviations, conversion of numeric text to numeric values, conversion of date formats, multiple or ambiguous words, conversion of units or measures, specific identifiers, and so on.

The WHERE clauses for read, delete, and update operations include almost all of the above cases where value linking is required, often with the help of values in the database, which is referred to as value matching. In addition, value linking is also required in the SET clause for create and update operations. In a create operation, the values to be inserted into the table need to match the type of columns and the style of data stored. Update operations, there may be a need to update the content that can not be obtained from the question in full, you need to use the database that has been

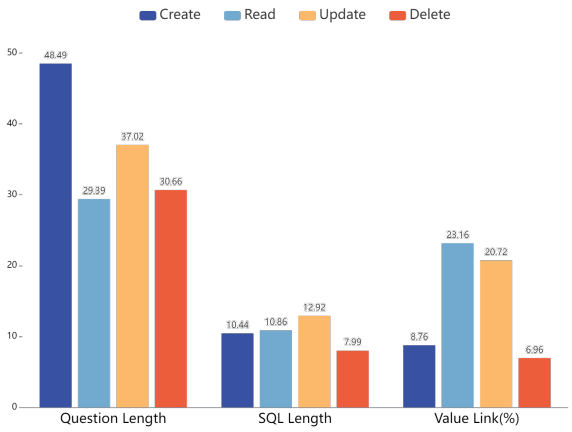

Figure 2: A statistical analysis of CRUDSQL, including question lengths, SQL lengths, and the percentage of value links.

stored in the content. For example, the question "Will the end of the training from the 1st change the time of the 2nd", at this time in the database stored at the end of the training time attribute of the value of " December 1", then the value after the modification should be "December 2", although our question does not indicate that the end time is in December, we need to rely on the content already stored as a prompt to modify, we will be this type of value linking is called a value prompting.

4 Benchmark Approach

In this paper, we propose a unified model called CRUDParser that aims to adapt create, read, update, and delete operations. The model combines two pre-trained language models, BERT and T5, and employs both value matching and value prompting strategies. We divide the task as a whole into two phases: first, the prediction results of the BERT model are utilized to populate the sketches; second, the T5 model is used to generate values involving update and create operations.

4.1 BERT Encoder

The input to BERT contains a natural language question Q and a database schema DS, where $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$, $|Q|$ denotes the length of the question, and $DS = \{table\_name, C_1, t_1, C_2, t_2, \ldots, C_{|N|}, t_{|N|}\}$, including table name, column name and column type, $|N|$ denotes how many columns there are in a table and each column $C_i = \{c_1, \ldots, c_{|N_i|}\}$, $|N_i|$ denotes the length of the field in the ith column. It is worth noting that there are two main types of columns, text type and real type.

As can be seen in Figure 3, we have divided multiple categorized sub-tasks, including the SELECT part, aggregation function part, WHERE clause part, SET expression part, logical operators, and question types. Among them, the aggregation function, operator and question type belong to the multi-classification task, which is denoted as p_1, while the columns and values belong to the bi-classification task, which is denoted as p_2, and its calculation formula is shown below:

$$p_1 = Softmax(FC(BERT(Q, DS))) \quad (1)$$

$$p_2 = Sigmoid(FC(BERT(Q, DS))) \quad (2)$$

First, we need to determine the type of question to choose a predetermined sketch and need to choose which parts to fill, such as the example in figure3, when the prediction of the question is an update operation, then the sketch is "UPDATE FROM _ SET _ WHERE _", we call it *SK*, and then need to focus on the SET expression part, the logical operators and the WHERE clause part. The *SK′* is obtained by filling the *SK* with the results of the corresponding subtask classifier, where the value is predicted using a pointer network [41], by intercepting it from the question. At this point, we get a preliminary SQL.

4.2 Value Matching

We found that many times the value intercepted from the question often does not correspond to the actual value in the database, so we fuzzy match the value in the WHERE clause in *SK′* with the actual value stored in the corresponding column in the database with the following expression:

$$score = fuzzy\_match\,(value, D\,[col]) \quad (3)$$

Depending on the score, decide whether to replace it with the database value or keep the intercepted value. Write the result after value matching as *SK″*.

4.3 Value Prompting

After the completion of the value matching can help us to improve the accuracy of the conditional matching, but in the create and update operations, it is no longer a match, both for the insertion of new values or modified new values, the database value can only be used as a kind of prompt existence. This prompt can help the value inserted into the table be in line with the style of the existing values

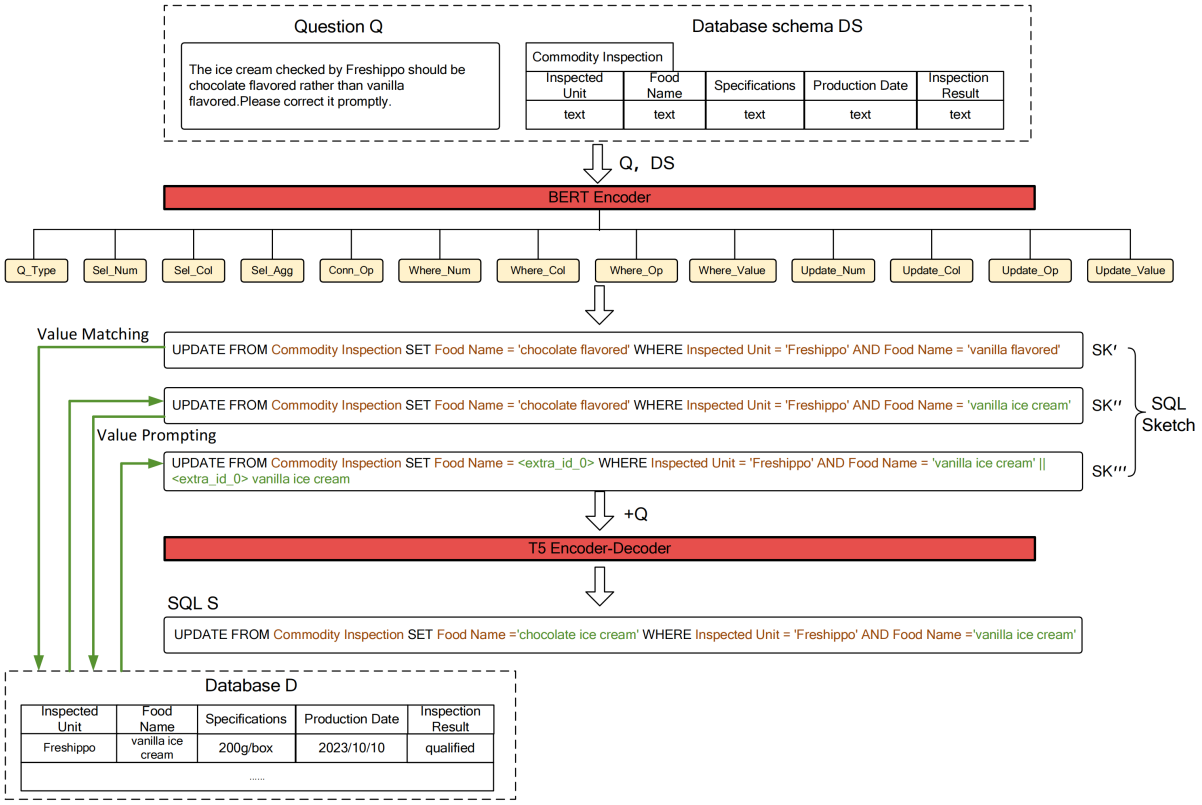

Figure 3: An overview of CRUDParser. Given a user's query Q and a database schema DS, the first stage fills the SQL sketch with the results predicted by BERT as well as multiple task classifiers, and the second stage generates the values that need to be created or updated by T5. There are also two strategies for interacting with the database, value matching and value prompting.

in the database, and can also help when updating can be based on the existing values in the database. For create operations, we will randomly select a database value as a prompt in the column where the value needs to be inserted. For update operations, we will first query the database for the value that was originally stored in the location that needs to be updated, and then use that value as the prompt. Then we would replace the part that needs to be inserted or modified with *<extra_id_x>*, $x \in$ [0, number of columns to be operated on], and then after SK'' we would splice "|| <extra_id_x> prompt value " as SK''', which is used for input into the T5 model that follows.

In Figure 3, both phases, value matching, and value prompting, obtain the required information by generating SQL queries that can interact with the database, i.e., we don't need to pass the actual values stored in the database into the model. When dealing with the actual database, it is not possible to write the database values into the input due to the invisibility of the data and the large amount of data. Therefore, the accuracy of the SQL can be improved by generating the required SQL queries to interact with the database during the process.

4.4 T5 Encoder-Decoder

We concatenate the question Q and SK''' with "||" as input to the T5 model, and use the model to generate the appropriate content to populate *<extra_id_x>* to get the final SQL S, which can be expressed as:

$$S = T5(Q, SK''') \tag{4}$$

5 Experiments

5.1 Experiment Setting

Data Settings As shown in Table 2, based on the cross-domain settings, we have divided CRUDSQL into train/dev/test sets, and there is no overlap of the databases contained in different datasets. Moreover, our SQL for the four types of operations is evenly divided, which helps the model to better learn the features in the dataset and better adapt to the SQL for different types of operations.

Evaluation Metrics We use three popular Text-to-SQL task evaluation metrics including Exact

Set Match Accuracy without values (EM without values), Exact Set Match Accuracy (EM), and Execution Accuracy (EX). EM without values measures whether the SQL generated by the model matches the standard SQL structurally, but does not consider specific values. EM requires not only structural matches but also exact matches of values. EX determines whether the generated SQL has the same result as the standard SQL after it is executed in the database.

Baseline Approach We use CRUDParser as a baseline approach for CRUDSQL. CRUDParser incorporates two pre-trained language models, BERT and T5, and combines a sketch-based slot-filling method with a sequence generation method to generate SQL in two phases, and in between, it uses value matching and value prompting strategies to further improve the accuracy of the task. It is worth noting that in order to better adapt to the Chinese context, the model versions we chose are BERT-WWM(Cui et al., 2019) and mT5-base(Xue et al., 2021), respectively. In addition to this, we also provide the results of the zero-shot demonstration of ChatGPT(gpt-3.5-turbo) in Table 5.

5.2 Experiment Results

Table 3 represents the EM without values of CRUDParser on CRUDSQL, and we can see that the worst performance is on the create operation, which is only 73.75% and 68.6% on the development and test sets, respectively, even though the create operation only corresponds to columns and values, which requires the fewest types of matches compared to other operations. The best performance is for the delete operation, which involves relatively fewer types of matches than the read and update operations.

Table 4 represents CRUDParser's EM on CRUDSQL, with -m removing the value matching process, -p removing the value prompting process, and -m-p requiring neither. It is seen that CRUDParser with value matching and value prompting gives the best performance in all cases. Create operations are only affected by value prompting, read and delete operations are only affected by value matching, and update operations are affected by both. We notice that the update operation has a gap of 11.67% and 9.4% on the development and test sets, respectively, and we find that the impact of value matching is greater than that of value prompting, due to the fact that when the matching is unsuccessful, it tends to affect the ability to get the corresponding prompted value from the database, which in turn affects the subsequent process.

Finally, we compared the EX of the models on CRUDSQL, as shown in 5. For the create operation, the EX is unchanged if both the inserted value and the corresponding column are matched correctly. The EX of the other types of operations is higher than the EM, because when the predicted SQL statement is the same as the real annotated SQL, the result of the SQL execution must be the same, but when the predicted SQL is different from the real annotated SQL, it often achieves the same result as the real annotated SQL due to the difference in the condition matching. In addition, despite its strong language comprehension and reasoning capabilities, ChatGPT cannot show better performance without interacting with the database and without resorting to database values.

5.3 Error Analysis

Why is it that the create operation requires the fewest types to match, only columns and values, but has the lowest accuracy rate by all metrics? By analyzing the incorrect results, we found that 68.65% of the results had schema linking errors, mainly column linking. Often, the columns mentioned in the question are missing from the results or matched to the wrong columns, for example, "The Little Prince is a short story in children's literature written in 1942 by the French author Antoine de Saint-Exupéry" may contain attributes such as title, author, country, year, and genre, which often cannot be matched without using methods such as Named Entity Recognition (NER).

Schema linking errors have been a difficult problem in this field, and the influencing factors include diverse linguistic expressions, complex database schemas, and lexical and grammatical differences. By observing the four types of error results, we found that 32.58% of the results contain schema linking errors, especially for the Chinese language, how to match the Chinese language with the English abbreviation, for example, "每股收益" in English is "Earnings Per Share", and the attribute value in the database is the abbreviation "EPS". When this occurs, external knowledge may need to be brought in to help the model make the right choice. Another issue worth noting is schema ambiguity, where part of a problem may map to more than one database element. This problem is prone to occur not only in cross-table queries, i.e., columns with the same name in the middle of dif-

MODEL	DEV				TEST			
	create	read	update	delete	create	read	update	delete
CRUDParser	73.75	77.08	80.42	88.75	68.6	71	70.6	86.8

Table 3: The Exact Set Match Accuracy Without Values of CRUDParser in CRUDSQL.

MODEL	DEV				TEST			
	create	read	update	delete	create	read	update	delete
CRUDParser	**60**	**71.67**	**70**	**86.25**	**49.6**	**67.08**	**61.8**	**83.8**
CRUDParser -m	**60**	66.67	62.08	78.75	**49.6**	57.08	53.8	79.8
CRUDParser -p	55.42	**71.67**	65.83	**86.25**	43.8	**67.08**	59	**83.8**
CRUDParser -m -p	55.42	66.67	58.33	78.75	43.8	57.08	52.4	79.8

Table 4: The Exact Set Match Accuracy(EM) of CRUDParser in CRUDSQL.

MODEL	DEV				TEST			
	create	read	update	delete	create	read	update	delete
CRUDParser	**60**	**74.58**	**72.5**	**89.58**	**49.6**	**70.42**	**65.6**	**84.4**
CRUDParser -m	**60**	69.58	62.92	82.92	**49.6**	60	56.4	80.4
CRUDParser -p	55.42	**74.58**	67.92	**89.58**	43.8	**70.42**	62	**84.4**
CRUDParser -m -p	55.42	69.58	59.58	82.92	43.8	60	55	80.4
ChatGPT	55.83	62.92	57.5	67.08	49.2	57.6	57.2	63.8

Table 5: The Execution Accuracy(EX) of CRUDParser and ChatGPT in CRUDSQL.

ferent tables, but also in single tables. For example, in the Real Estate Sales Amount Forecast table, the attribute values are (Tier 1 cities, year-on-year growth rate, Tier 2 cities, year-on-year growth rate, Tier 3 cities, year-on-year growth rate), so how can we correspond to the correct columns when we refer to the year-on-year growth rate in the question. It turns out that although we matched the correct city type, the year-on-year growth rate still matches incorrectly. To solve this problem, we may need to add a correlation between year-on-year growth rate and city type when coding the database schema.

Update operations require both condition matching and value modification, which poses multiple challenges. In particular, when modifying database values, one is that the updated value has no relationship with the original value, in which case the value prompts are used mainly to keep the value updated to the database in line with the database style, and the other is that the updated value is modified based on the original value, in which case the value prompts are used to fill in the missing information in the question. By analyzing the examples of errors in update operations, we found that 36.18% of them were errors in updating values, and in some cases, the model did not know what consistent style was, for example, "3A" was mentioned in the question, while the database stored the value as "AAA ". Moreover, we found that when using values as prompts to guide the model to generate results, if not controlled, the results may tend to deviate from our needs, especially when generating numeric content.

6 Conclusion

In this paper, we propose CRUDSQL, a large cross-domain, single-table Chinese Text-to-SQL benchmark dataset with a special focus on four types of SQL operations: create, read, update, and delete. Compared to existing benchmark datasets that only have read operations, this paper introduces new challenges, especially for value linking, and puts more emphasis on the accuracy of the updated and created values. In addition to this, we propose a baseline approach for this dataset, CRUDParser, which generates SQL in stages by pre-training the language model and incorporates value matching and value prompting strategies. We demonstrate the experimental benchmark results.

This paper has demonstrated initial promise, but there is still room for improvement. One potential future work is to extend single-table to multi-table query SQL generation although the focus of this

paper is unifying manipulation and query SQL producing. Another future work is to fine-tune large language models on this dataset and put it closer to more practical application scenarios.

Acknowledgements

This research work is supported by the National Natural Science Foundation of China (62062046) and CCF-Huawei Populus Grove Fund.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3977–3983.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Han Fu, Chang Liu, Bin Wu, Feifei Li, Jian Tan, and Jianling Sun. 2023. Catsql: Towards real world natural language to sql applications. *Proceedings of the VLDB Endowment*, 16(6):1534–1547.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-sql models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2505–2515.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-sql generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023a. Few-shot text-to-sql translation using structure and content prompt learning. *Proceedings of the ACM on Management of Data*, 1(2):1–28.

Zihui Gu, Ju Fan, Nan Tang, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Sam Madden, and Xiaoyong Du. 2023b. Interleaving pre-trained language models and large language models for zero-shot nl2sql generation. *arXiv preprint arXiv:2306.08891*.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A large-scale and pragmatic chinese dataset for cross-database context-dependent text-to-sql. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-sql: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

Saihao Huang, Lijie Wang, Zhenghua Li, Zeyang Liu, Chenhui Dou, Fukang Yan, Xinyan Xiao, Hua Wu, and Min Zhang. 2022. Sesql: Yet another large-scale session-level chinese text-to-sql dataset. *arXiv preprint arXiv:2208.12711*.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*.

Xiping Liu and Zhao Tan. 2023. Divide and prompt: Chain of thought prompting for text-to-sql. *arXiv preprint arXiv:2304.11556*.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. *arXiv preprint arXiv:2008.04759*.

Qingkai Min, Yuefeng Shi, and Yue Zhang. 2019. A pilot study for chinese sql semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3652–3658.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901.

Ningyuan Sun, Xuefeng Yang, and Yunfeng Liu. 2020. Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation. *arXiv preprint arXiv:2006.06434*.

Lappoon R Tang and Raymond Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020. Dusql: A large-scale and pragmatic chinese text-to-sql dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Kuan Xu, Yongbo Wang, Yongliang Wang, Zihao Wang, Zujie Wen, and Yang Dong. 2022. Sead: End-to-end text-to-sql generation with schema-aware denoising. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1845–1853.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. 2020. Sparc: Cross-domain semantic parsing in context. In *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*, pages 4511–4523. Association for Computational Linguistics (ACL).

JM Zelle and RJ Mooney. 1996. Learning to parse database queries using inductive logic programming. Technical report, American Association for Artificial Intelligence, Menlo Park, CA (United States).

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.