

REFORMER: A ChatGPT-Driven Data Synthesis Framework Elevating Text-to-SQL Models

Shenyang Liu
Department of Computer Science
University of Central Florida
Orlando, USA
shenyang.liu@ucf.edu

Saleh Almohaimeed
Department of Computer Science
University of Central Florida
Orlando, USA
sa247216@ucf.edu

Liqiang Wang
Department of Computer Science
University of Central Florida
Orlando, USA
liqiang.wang@ucf.edu

Abstract—The existing Text-to-SQL models suffer from a shortage of training data, inhibiting their ability to fully facilitate the applications of SQL queries in new domains. To address this challenge, various data synthesis techniques have been employed to generate more diverse and higher quality data. In this paper, we propose REFORMER, a framework that leverages ChatGPT’s prowess without the need for additional training, to facilitate the synthesis of (question, SQL query) pairs tailored to new domains. Our data augmentation approach is based on a “retrieve-and-edit” method, where we generate new questions by filling masked question using explanation of SQL queries with the help of ChatGPT. Furthermore, we demonstrate that cycle consistency remains a valuable method of validation when applied appropriately. Our experimental results show that REFORMER consistently outperforms previous data augmentation methods. To further investigate the power of ChatGPT and create a general data augmentation method, we also generate the new data by paraphrasing the question in the dataset and by paraphrasing the description of a new SQL query that is generated by ChatGPT as well. Our results affirm that paraphrasing questions generated by ChatGPT help augment the original data.

Index Terms—Data Augmentation, Text-to-SQL, GPT

I. INTRODUCTION

For Text-to-SQL tasks, popular training datasets like WikisQL [1] and Spider [2] are often too small, limiting model generalization [3] [4]. This makes models struggle in new domains. To address this, data synthesis is crucial. Unlike past SQL-to-Text research that relied on zero-shot methods [5] [6], we believe using one-shot learning with examples can enhance performance. Previous data synthesis methods also lacked query diversity, reducing model effectiveness [7] [8] [9] [5].

We introduce REFORMER, a framework leveraging ChatGPT for data synthesis. By retrieving similar (question, SQL query) pairs and masking the questions, we use ChatGPT to generate new queries without fine-tuning. A novel cycle-consistency validation approach compares the similarity between ChatGPT-generated questions and SQL query explanations, improving stability over past methods [10]. Additionally, we explore two data augmentation techniques: paraphrasing existing questions and creating new SQL queries using schema templates.

Our contributions include: (1) to the best of our knowledge, we are the first one to create a SQL-to-Text framework REFORMER using ChatGPT without fine-tuning, (2) we introduce question-query-question cycle consistency validation, (3) we show REFORMER’s improvement over other data augmentation methods, and (4) demonstrate effective data generation with two paraphrasing techniques.

II. RELATED WORK

A. SQL-to-Text Task

The models for SQL-to-Text task include those built upon the Gated Recurrent Unit (GRU) [11], LSTM networks [7] [8], and advanced models like BART [12] and T5 [5] [6]. In specific cases, some models, as demonstrated by [7], leverage BERT [13] as an encoder to enhance overall performance. The SQL queries used in training these models can come from a variety of sources. Some models employ hand-crafted templates, while others leverage grammars extracted from existing SQL queries [8] or draw from a corpus of SQL queries obtained through web scraping or crawling open-source code repositories [12]. The authors of [14] and us independently came up with the same question-query-question cycle consistency as in our paper concurrently, but they train a GPT-3 model and do not use the method for data augmentation purpose but for validation in text-to-SQL task.

B. Large Language Models and Prompt Engineering For SQL-to-Text

It is noteworthy that, to date, there has not been many research [5] [6] [14] considering the utilization of state-of-the-art Large Language Models (LLMs) for SQL-to-text tasks. Our work, therefore, can be viewed as pioneering in this regard. By harnessing the capabilities of LLMs, we aim to inspire and open up new avenues for further investigations into data augmentations using these new models.

Prompt-based learning directly uses the model without training or fine-tuning to get the answer for the questions. Prompt engineering usually uses manually designed templates [15] [16] or optimization techniques [17] to construct more effective prompts for the model.

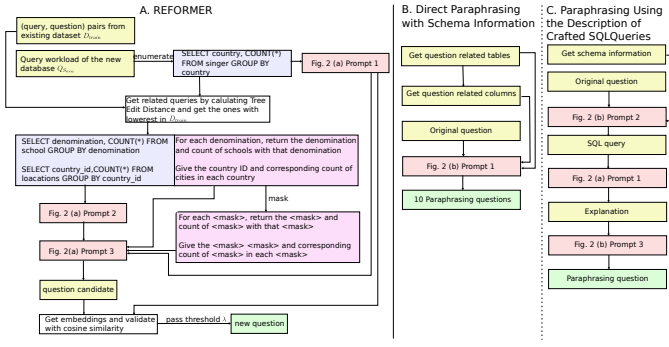


Fig. 1: Framework for REFORMER and Paraphrasing-based Data Synthesis methods

III. DATA SYNTHESIS WITH REFORMER

In this study, we focus on data synthesis for Text-to-SQL parsers by leveraging the power of GPT-based models. Directly training a SQL-to-Text model on existing data and using manually crafted templates for generating new SQL queries has limitations that the results may fail to accurately capture the underlying data distribution present in the original dataset due to the lack of guided question generation. To address this issue, we adopt the “retrieve-and-edit” approach introduced by [10] and improve the approach to avoid training models. Our approach is summarized in Algorithm 1 and the whole framework is shown in Figure 1A. More detailed explanations for the algorithm are provided from Section 3.1 to Section 3.3.

Algorithm 1 Data Augmentation with REFORMER

Input $Q_{S_{new}}, M, D_{train}$

$D_{new} \leftarrow \emptyset$

for $q \leftarrow \text{enumerate}(Q_{S_{new}})$ **do**

$q_{sim}, u_{sim} \leftarrow \text{GetRelatedQueries}(q, D_{train})$

$u_{temp} \leftarrow \text{MaskSchemaTokens}(q_{sim}, u_{sim})$

$expl1 \leftarrow \text{GetExplanation}(q)$

$u_{result} \leftarrow \text{FillTemplate}(u_{temp}, expl1)$

$expl2 \leftarrow \text{GetExplanation}(q)$

$D_{new} \leftarrow D_{new} \cup \text{Validate}(expl2, u_{result})$

end for

$M_{new} \leftarrow \text{Fine-tune}(M, D_{new})$

A. Related SQL queries and question templates preparation

The SQL queries we use within a new schema S_{new} can be achieved by manually creating or using a grammar-based SQL generator [7] [9] using random parts of the schema information, denoted as $Q_{S_{new}}$. We consider a Text-to-SQL parser model M trained by D_{train} to be enhanced by our proposed method.

GetRelatedQueries: Given a query q , we aim to identify a SQL query with high structural similarity (denoted by q_{sim}) to q and its corresponding question (or utterance denoted by u_{sim}). Our goal is to derive question templates that ensure fidelity to the training data distribution. To retrieve SQL

queries exhibiting high structural similarity to a given query q , we adopt the method in [10] that utilizes the tree-edit-distance [18]. This tree-edit-distance measures the similarity between two relational algebra trees constructed from SQL queries. The smaller the distance, the higher the structural similarity between the queries. This approach does not consider schema-specific information to assess structural similarity, allowing us to focus on the query structure itself. We adopt the hyperparameters in [10] so that we only retain queries with a distance of less than 0.1 from the given query q , ensuring that they exhibit substantial structural resemblance.

MaskSchemaTokens: The question templates play an important role in guiding our data synthesis process. These templates are derived from existing questions but with schema-specific information masked out. The purpose of these templates is to guide the generation of questions while ensuring that the resulting questions conform to the data distribution observed in the training set. We adopt the frequency-based method in [10] to determine which words should be included in the question templates and which should be masked. Specifically, we retain all words that appear in more than 50% of the schemas, as these words are indicative of common patterns and characteristics in the training data. Any words that fall below this threshold are masked and replaced with a special token, denoted as “MASK”. Consecutive masked words are represented by a single “MASK” token to maintain conciseness and readability. For a question u , we get template u_{temp} after masking based on the explanation above. This preparation of related SQL queries and question templates serves as the foundation for our data synthesis methodology, enabling the generation of high-quality synthetic data to enhance the performance of Text-to-SQL parsers.

B. New Question Generation

GetExplanation and **FillTemplate:** We generate new questions for Text-to-SQL parsers by leveraging ChatGPT to fill masked templates with one-sentence query explanations. Using Prompt 1 in Figure 2a, ChatGPT extracts a concise explanation $expl1$ from a given SQL query $q \in Q_{S_{new}}$, resulting in question u_{result} . Since the Spider dataset typically has succinct, one-sentence questions, we match this format.

We improve upon [10] by avoiding intermediate representations, which often include unnecessary terms (e.g., “belongs to” for tables, “the number of” for COUNT(*)) that negatively affect ChatGPT’s performance. Instead, we instruct ChatGPT to explain the SQL in a single sentence without using table names, yielding more accurate question generation. We employ in-context learning [19], providing original (question, SQL query) pairs to guide ChatGPT, allowing it to generate questions aligned with the desired structure. This method helps create a new dataset D_{new} for fine-tuning existing Text-to-SQL parsers.

C. Result validation

GetExplanation and **Validate:** To ensure the consistency of the generated SQL queries and the associated questions,

we came up with a validation approach that leverages the embeddings produced by ChatGPT. Rather than training a dedicated model to validate the alignment between SQL queries and newly generated questions, we employ embeddings generated from ChatGPT. Specifically, we compute the cosine similarity between the embeddings derived from the one-sentence explanation of the SQL query and the generated question. To increase the diversity of the questions, we generate a new one-sentence query explanation *expl2* based on the corresponding Prompt 2 in Figure 2a for the cosine similarity rather than use the one to generate the result question. This method unveils the relationship between SQL queries and natural language questions, effectively establishing what can be termed as a “question-query-question cycle” consistency. Notably, our approach differs from previous methods [7] that directly train binary classifiers with (question, SQL query) pairs. The drawback of such direct training is the absence of specific structures tailored for SQL queries. Unlike models designed specifically for SQL queries [20] [21], where the classifiers might struggle to elucidate the nuanced relationship between queries and questions, our method addresses this limitation. We agree with the concerns raised by [10] about the usage of “query-question-query” cycle-consistency, which get a query to generate a question, and utilize a Text-to-SQL model to generate a query and then compare the similarity between the two queries. However, we demonstrate that the concept of cycle consistency can be valuable when applied appropriately, as evident in our methodology. To ensure the coherence and consistency of the generated questions, we employ a threshold for cosine similarity, similar to previous work [10]. This threshold serves as a filter, allowing us to identify and retain only the questions that exhibit a satisfactory degree of consistency with their corresponding SQL queries. The questions that pass this filtering process, along with their associated queries, are then utilized for **fine-tuning** existing Text-to-SQL model M with a new dataset D_{new} to get a better model M_{new} .

IV. PARAPHRASING-BASED DATA SYNTHESIS METHOD USING CHATGPT

To fully leverage the natural language processing capabilities of ChatGPT, we explore two general data augmentation methods that utilize the paraphrase with schema information to generate new data without considering a specific domain. The method is shown in Fig. 1B.

A. Direct Paraphrasing with Schema Information

Traditionally, paraphrasing methods focused solely on the sentences themselves. [22] [23] However, for datasets like Spider, where questions are associated with specific schema information, we propose an enhanced approach. Beyond the question text, we utilize ChatGPT in conjunction with the LangChain library to extract related tables from the schema for a given question. Subsequently, we retrieve all relevant column names for these tables. The paraphrasing process involves incorporating the question, related table names, and column

names, ensuring that the generated data is closely aligned with the underlying database structure. Figure 2b Prompt 1 illustrates the prompt used in this approach. To maintain the result quality, the same cycle-consistency validation method is employed here.

B. Paraphrasing Using the Description of Crafted SQL Queries

Different from the direct paraphrasing method, we introduce a strategy based on crafting SQL query templates. Starting with basic queries containing select, from, where, group by, order by, and limit clauses, we progress to generating complex queries with in, union, except, and intersect clauses. Following the creation of these templates, ChatGPT is employed to fill in the blanks with tables, columns and values, guided by the related schema. The synthesized SQL queries are then executed to validate their correctness, and only those without errors are retained. For each SQL query, we use ChatGPT to generate a concise one-sentence description. New questions are subsequently generated by paraphrasing these descriptions. The prompt used for this method is detailed in Figure 2b Prompt 2 and Prompt 3.

Prompt 1:
Explain sql query: {query} in one sentence with as many details to a person does not know SQL and do not directly use table name.
ChatGPT return: {explanation1}

Prompt 2:
Explain sql query: {query_similar} in one sentence with as many details to a person does not know SQL and do not directly use table name.
ChatGPT return: {explanation2}

Prompt 3:
Use this sentence {explanation1} to fill in {question_template}, so that they have the same meaning and concise. Consider the example, when the sentence is: {explanation2} and the result is: {question for query_similar}. Give the result based on the style of the example.
ChatGPT return: {question_result}

(a) Prompts for REFORMER

Prompt 1:
Given the question and the related table names and the column names for each of the table inside the parentheses after the table, get 10 new paraphrase for the question. The paraphrase should not contain word 'table' or 'tables' or 'column' or 'columns' explicitly.
Question: {question}
Table and Column Names: {table_column_names}
ChatGPT return: {10_questions}

Prompt 2:
The database schema defined as below: {schema}, give me only one sql query as answer without description, the template is: {SQL template}. Use on keyword to join tables and use 'T1', 'T2', 'T3' and so on as table alias. Please replace [column], [table] and [val] in the template with schema information. If the column is a number type, then you may randomly use these aggregation function: min, max, avg, sum to the final answer. Do not give more than one sql queries.
ChatGPT return: {SQL_query}

Prompt 3:
Based on explanation: {explanation}, please generate a imperative sentence with the explanation said by a person does not know SQL without mentioning the word 'query' or 'column' or 'table' or 'joining' or 'data' or 'database' or 'dataset'. display only the generated question.
ChatGPT return: {question_result}

(b) Prompts for Paraphrasing-based Method

Fig. 2: Prompts for data generation, where “query”, “query_similar”, “explanation 1/2”, “question_result”, “question_template” etc. denote the corresponding placeholders, which will be replaced by real queries or explanations when handling SQL query and question pairs.

V. EXPERIMENT SETTING

A. Datasets

For our experiments, we use the Spider dataset [2], which includes 7,000 (question, SQL query) pairs in the training set and 1,034 pairs in the dev set. Following prior work [10], we categorize database schemas by domain, such as music, pets, university records, and country records. We also replace constants in 70% of the SQL queries with other values from the same columns, as in [10]. For paraphrasing-based methods,

we use the Spider dataset directly without modifications, as these methods are not domain-specific.

B. Text-to-SQL Model

In keeping with previous studies [10], we employ the SmBop model [24] as our baseline Text-to-SQL model. The SmBop model utilizes the RoBERTA [25] model as its backbone and connects it with four Relation-Aware Attention (RAT) [20] layers. The dev set in the Spider dataset is modified to ensure that it does not include any data associated with the evaluation categories.

C. SQL-to-Text Model

Our approach capitalizes on the capabilities of ChatGPT [26] for the generation of text-based outputs. As detailed in Section 3.2, we utilize ChatGPT to fill the masked templates, and a sample prompt is illustrated in Fig. 2a. For paraphrasing-based method, we use ChatGPT to generate related tables, SQL queries and paraphrases, and a sample prompt is shown in Fig. 2b.

D. Validation Module

The validation module (outlined in Section 3.3) computes the cosine similarity between the GPT embeddings (text-embedding-ada-002) derived from the one-sentence explanation of the SQL query and the generated question. A threshold value, denoted as λ , is set to 0.85. This threshold aids in filtering out inconsistent results, enhancing the overall quality of our synthesized data. Beyond the threshold for similarity, we keep only the best 5 results to maintain the high quality of the generated data used for fine-tuning. The method of “Direct Paraphrasing with Schema Information” uses a larger threshold λ for the similarity to decrease the size of the dataset. We tried different values of λ (such as 0.9, 0.93, 0.95) in our experiments.

E. Baselines

For REFORMER framework, we incorporate ReFill [10] as the baseline model in our experiments. ReFill has demonstrated superior performance compared to several previous methods, such as L2S [9], GAZP [7], and SNOWBALL [27]. For our paraphrasing-based data synthesis method, we compare with SmBop model [24], T5 model+Picard [28] and T5 model+Picard+synthesized data [6].

F. Evaluation Metrics

In line with the Spider dataset’s evaluation methodology [2], we evaluate the performance of our models using Exact Set Match (EM) and Execution Accuracy (EX). EM disregards database-related values and compares the result query with the gold query word by word. On the other hand, EX executes both the result query and the gold query via a SQL engine and measures their concordance.

TABLE I: Results for fine-tuning the SmBop parser on (question, SQL query) pairs generated using REFILL and REFORMER

Method	Category1 (Music)	Category2 (Pets)	Category3 (University)	Category4 (Country)	Average
	EM/EX	EM/EX	EM/EX	EM/EX	EM/EX
SmBop [24]	87.8/88.7	63.7/65.3	69.5/69.5	44.2/35.8	66.3/65.0
REFILL [10](In Paper)	88.7/87.0	69.7/73.8	73.2/70.1	55.8/45.0	71.8/68.9
REFILL [10](Experiment)	85.2/86.1	56.5/58.1	67.7/67.7	56.7/46.7	66.4/64.7
Ours (validation only)	87.8/88.7	56.5/57.3	72.6/73.2	54.2/46.7	67.9/66.8
Ours (data augmentation only)	87.8/88.7	59.7/62.9	72.6/73.8	56.7/47.5	69.2/68.5
Ours	89.6/89.6	58.9/62.1	73.8/74.4	57.5/50.8	70.0/69.4

VI. EXPERIMENTAL RESULTS AND ANALYSIS

A. Overall Evaluation

Our overall evaluation, detailed in Table I, demonstrates the superior performance of our method compared to ReFill across different categories. Notably, our approach outperforms ReFill in all categories in our own experiment (denoted by “Experiment”), while surpassing the claimed performance in most categories (except for category 2) in [10] (denoted by “In Paper”). Further scrutiny of the data in Category 2 highlights potential reasons for performance deterioration with data augmentation methods. On average, we achieve an increase in Exact Set Match (EM) by 3.6% and Execution Accuracy (EX) by 4.7% when compared to the REFILL framework (Experiment). We do not beat EM for REFILL (In Paper) but beat EX by 0.5%.

B. Generated Question Evaluation

To have a fair comparison, we only compare our method with REFILL(Experiment). Table I shows the performance of using the SQL-to-Text module of the REFORMER framework independently (data augmentation only) and then using the REFILL filtering module. The results reveal improvements in EM by 2.8% and EX by 3.8%. This analysis underscores the effectiveness of our data augmentation approach, which contributes to the overall performance enhancement.

C. Cycle Consistency Method Evaluation

In Table I, we examine the performance when utilizing the validation module of the REFORMER framework exclusively (validation only). The results illustrate improvements in EM by 1.5% and EX by 2.1%. Notably, comparing with the performance improvement by our approach with only data augmentation (denoted by “Ours (data augmentation only)”), our approach using only cycle consistency validation method (denoted by “Ours (validation only)”) is less effective.

D. Error Analysis for REFORMER

Our analysis reveals that these errors disproportionately impact the data augmentation in Category 2, ultimately yielding suboptimal results (see Table I). Notably, some of the generated templates retain information from the original questions should be excluded, which may result in inaccuracies. However, for brevity, we focus solely on errors attributable to other causes.

The REFILL method in our experiment introduces multiple challenges. Specifically, we observe the questions generated in Category 2 in our experiment are less complex than the

data synthesized in [10]. This discrepancy in data distribution adversely affects the performance of the model. Moreover, certain errors manifest more frequently in our generated data than in the data synthesized by [10].

Our method REFORMER also has its own limitations in terms of data diversity and the reliance on one-shot settings. While data diversity is generally a desirable aspect of data augmentation, it can inadvertently disrupt existing models due to unseen novel expressions during training. The one-shot setting, where substantial emphasis is placed on only one example, restricts the flexibility to incorporate additional information into the question template.

Some information are missing.
SELECT T1.owner_id , T1.last_name FROM Owners AS T1 JOIN Dogs AS T2 ON T1.owner_id = T2.owner_id
JOIN Treatments AS T3 ON T2.dog_id = T3.dog_id GROUP BY T1.owner_id ORDER BY count(*) DESC LIMIT 1
Show the name of the owner with the highest count of treatments provided to their dogs.

Some question templates may not fit for the SQL query.
SELECT max(weight) , petType FROM pets GROUP BY petType
What are the different pet types and how many pets have the maximum weight?

Columns are not mentioned explicitly.
select T1.fname from student as T1 join has_pet as T2 on T1.stuid = T2.stuid join pets as T3 on T3.petid =
T2.petid where T3.pettype = 'cat' intersect select T1.fname from student as T1 join has_pet as T2 on
T1.stuid = T2.stuid join pets as T3 on T3.petid = T2.petid where T3.pettype = 'dog'
Find the names of students that have both a cat and a dog as pets.

Diversity may lead confusion.
SELECT count(*) FROM pets WHERE weight > 11
How many pets weigh over 11?

The column in SQL query does not have a proper name.
SELECT name , age , weight FROM Dogs WHERE abandoned_yn = 2
List the name, age, and weight for all abandoned dogs

Explain SQL directly.
SELECT pettype , weight FROM pets ORDER BY pet_age LIMIT 1
Show the type and weight of the pets, sorted by the pet's age in ascending order, and retrieve only the first row.

Fig. 3: Errors from REFORMER

In Figure 3, we comprehensively present the errors identified by our experiment on REFORMER. The specific errors examined are as follows:

“Some Information Are Missing” Error: This occurs when the SQL query specifies the last name, but the question only asks for “name.” Consistent rules for specifying “first name” or “last name” could help, though expecting perfect precision is challenging.

“Question Template Mismatch” Error: A mismatch arises when the SQL query retrieves the maximum weight for pets, but the question template introduces irrelevant phrases like “how many,” which doesn’t align with the intent of the SQL query.

“Columns Not Explicitly Mentioned” Error: This error happens when important columns, like student IDs, are needed in the SQL query but are not mentioned in the corresponding question.

“Diversity Leads to Confusion” Error: ChatGPT’s varied phrasing, like using “weigh” instead of “weight,” can confuse the model when it expects specific column names.

“Improper Column Names” Error: SQL queries may use abbreviations (e.g., “yn” for year number), and while ChatGPT can interpret common ones, unfamiliar abbreviations may cause failures.

“Explain SQL Directly” Error: Generated questions sometimes directly reflect SQL descriptions in an unnatural, non-conversational tone, highlighting the need for more human-like question generation.

E. Quality and Diversity of Generated Questions

To gauge the quality and diversity of the generated questions, we employ two metrics. The BLEU score [29] of the set $S(q)$ measures the quality of the synthesized data for a query q when compared to its corresponding gold query. Concurrently, the SelfBLEU metric [30] calculates the average BLEU score among $S(q)$, providing a measure of the diversity of the synthesized data. Higher BLEU scores denote higher quality, while lower SelfBLEU scores indicate greater diversity.

In the study of [10], the generation of hypotheses for each query involves producing ten candidates through beam sampling [31]. The selection of the final hypothesis is determined by the highest BLEU score, serving as our final result for evaluation. In our comparative analysis, we also consider L2S [9], GAZP [7], and SNOWBALL [27]. It’s essential to note that our approach lacks a dedicated training phase, which makes the direct comparisons with other methods unfair. Instead, we leverage all possible templates to generate hypotheses, selecting the one yielding the highest BLEU score.

The results in Table II indicate that our method does not surpass REFILL in terms of BLEU and SelfBLEU scores. However, it outperforms other methods in BLEU. Despite the lower score on BLEU and higher score on SelfBLEU, our method demonstrates superiority over REFILL in terms of Exact Match (EM) and Execution Accuracy (EX) across most categories.

F. Result Evaluation for Direct Paraphrasing with Schema Information

The findings (see Table III) show the performance of our method highly depends on the size of the augmented data.

When considering values of λ set at 0.93 and 0.95, corresponding to augmented data sizes of 21,781 and 8,939, respectively, the performance closely aligns with the baseline SmBop model, exhibiting marginal improvement of less than 1%. This observation suggests that, within certain data augmentation thresholds, the Direct Paraphrasing method does not significantly outperform the SmBop baseline.

However, a distinct shift is discerned when λ is set to 0.9, resulting in an augmented data size of 43,847. In this scenario, the results exhibit marked improvements, with Exact Match (EM) showing a commendable increase of 2.8% and the Exact F1 (EX) metric registering a corresponding rise of 2.7%. These enhancements bring the EM metric close to the performance of T5-3B with PICARD [28] and surpass the performance of T5-3B with Syn Data [6].

G. Issues for Paraphrasing using the Explanation of Crafted SQL Queries

The SmBop model excels due to its use of parsing trees for both input and output validation of SQL queries. However, its custom parser is limited to specific cases, leading to errors when parsing more complex SQL templates in our new dataset. To address this, we switched to RASAT(-small), a model based on T5-small, due to resource constraints preventing us from testing on T5-3B.

Our results showed that using the Explanation of Crafted SQL Queries for paraphrasing significantly decreased EM and EX scores. The errors mainly occurred with hard-level SQL queries, so we designed even more complex templates than those in the Spider dataset. However, the lack of ground truth for the generated data contributed to performance issues.

TABLE II: Results for BLEU and Self-BLEU score from different methods

Method	BLEU \uparrow (Quality)	100-SelfBLEU \uparrow (Diversity)
Gold-Ref	100	68.8
L2S	38.0	2.2
GAZP	38.8	2.0
SNOWBALL	40.2	2.8
REFILL	48.6	33.8
Ours	43.2	41.0

VII. CONCLUSION AND LIMITATION

In this study, we introduced REFORMER for generating (question, SQL query) pairs tailored to new domains using a "retrieve-and-edit" approach. By filling question templates with information from queries and leveraging ChatGPT with carefully crafted prompts, REFORMER outperforms previous methods without extra training. Our experiments highlight the effectiveness of cycle-consistency for validation. Additionally, we explored two paraphrasing-based data synthesis methods, demonstrating ChatGPT's ability to generate high-quality data. While we only focused on ChatGPT with spider dataset and manually crafted prompts, future research should explore automatic prompt generation and test other LLMs and Text-to-SQL datasets.

REFERENCES

[1] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv:1709.00103*, 2017.

[2] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv:1809.08887*, 2018.

[3] M. Hazoom, V. Malik, and B. Bogin, "Text-to-sql in the wild: a naturally-occurring dataset based on stock exchange data," *arXiv:2106.05006*, 2021.

Citation: S. Liu, S. Almohamed and L. Wang, "REFORMER: A ChatGPT-Driven Data Synthesis Framework Elevating Text-to-SQL Models," 2024 International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2024, pp. 828-833, doi: 10.1109/ICMLA61862.2024.00119. IEEE Xplore link: <https://ieeexplore.ieee.org/abstract/document/10903391>

TABLE III: Results for Paraphrasing-based method

	Training Set Size	EM/EX
Base(SmBop [24])	7000	72.1/72.3
With Schema Info ($\lambda=0.9$)	7000+43847	74.9/75.0
With Schema Info ($\lambda=0.93$)	7000+21871	72.6/72.3
With Schema Info ($\lambda=0.95$)	7000+8939	72.1/71.5
With Crafted SQL	7000+12173	25.7/33.7
T5-3B [28]	7000	71.5/74.4
T5-3B + PICARD [28]	7000	75.5/79.3
T5-3B + Syn Data [6]	7000+21851	74.5/78.6
T5-3B + PICARD + Syn Data [6]	7000+21851	76.1/81.4

[4] A. Suhr, M.-W. Chang, P. Shaw, and K. Lee, "Exploring unexplored generalization challenges for cross-database semantic parsing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8372-8388.

[5] W. Yang, P. Xu, and Y. Cao, "Hierarchical neural data synthesis for semantic parsing," *arXiv:2112.02212*, 2021.

[6] Y. Zhao, J. Jiang, Y. Hu, W. Lan, H. Zhu, A. Chauhan, A. Li, L. Pan, J. Wang, C.-W. Hang *et al.*, "Importance of synthesizing high-quality data for text-to-sql parsing," *arXiv:2212.08785*, 2022.

[7] V. Zhong, M. Lewis, S. I. Wang, and L. Zettlemoyer, "Grounded adaptation for zero-shot executable semantic parsing," *arXiv:2009.07396*, 2020.

[8] K. Wu, L. Wang, Z. Li, A. Zhang, X. Xiao, H. Wu, M. Zhang, and H. Wang, "Data augmentation with hierarchical sql-to-question generation for cross-domain text-to-sql parsing," *arXiv:2103.02227*, 2021.

[9] B. Wang, W. Yin, X. V. Lin, and C. Xiong, "Learning to synthesize data for semantic parsing," *arXiv:2104.05827*, 2021.

[10] A. Awasthi, A. Sathe, and S. Sarawagi, "Diverse parallel data synthesis for cross-database adaptation of text-to-sql parsers," *arXiv:2210.16613*, 2022.

[11] D. Guo, Y. Sun, D. Tang, N. Duan, J. Yin, H. Chi, J. Cao, P. Chen, and M. Zhou, "Question generation from sql queries improves neural semantic parsing," *arXiv:1808.06304*, 2018.

[12] P. Shi, P. Ng, Z. Wang, H. Zhu, A. H. Li, J. Wang, C. N. dos Santos, and B. Xiang, "Learning contextual representations for semantic parsing with generation-augmented pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 806-13 814.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] Y. Zhang, J. Deriu, G. Katsogiannis-Meimarakis, C. Kosten, G. Koutrika, and K. Stockinger, "Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems," *arXiv preprint arXiv:2306.04743*, 2023.

[15] L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang, "Template-based named entity recognition using bart," *arXiv:2106.01760*, 2021.

[16] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv:2104.08691*, 2021.

[17] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv:1909.01066*, 2019.

[18] M. Pawlik and N. Augsten, "Efficient computation of the tree edit distance," *ACM Transactions on Database Systems (TODS)*, vol. 40, no. 1, pp. 1-40, 2015.

[19] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877-1901, 2020.

[20] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," *arXiv:1911.04942*, 2019.

[21] Z. Chen, L. Chen, Y. Zhao, R. Cao, Z. Xu, S. Zhu, and K. Yu, "Shadowgnn: Graph projection neural network for text-to-sql parser," *arXiv:2104.04689*, 2021.

[22] T. Dopierre, C. Gravier, and W. Logerais, "Protaugment: Intent detection meta-learning through unsupervised diverse paraphrasing," in *ACL-IJCNLP*. ACL, 2021, pp. 2454-2466.

[23] H. Dai, Z. Liu, W. Liao, X. Huang, Z. Wu, L. Zhao, W. Liu, N. Liu, S. Li, D. Zhu *et al.*, "Chataug: Leveraging chatgpt for text data augmentation," *arXiv:2302.13007*, 2023.

[24] O. Rubin and J. Berant, "Smbop: Semi-autoregressive bottom-up semantic parsing," *arXiv:2010.12412*, 2020.

[25] 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 preprint arXiv:1907.11692*, 2019.

[26] OpenAI. (2022) Chatgpt (sep 25 version). [Online]. Available: <https://chat.openai.com/chat>

[27] C. Shu, Y. Zhang, X. Dong, P. Shi, T. Yu, and R. Zhang, "Logic-consistency text generation from semantic parses," *arXiv:2108.00577*, 2021.

[28] T. Scholak, N. Schucher, and D. Bahdanau, "Picard: Parsing incrementally for constrained auto-regressive decoding from language models," *arXiv:2109.05093*, 2021.

- [29] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *ACL*, 2002, pp. 311–318.
- [30] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, “Texygen: A benchmarking platform for text generation models,” in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 1097–1100.
- [31] A. Fan, M. Lewis, and Y. Dauphin, “Hierarchical neural story generation,” *arXiv:1805.04833*, 2018.