Exploring Schema Generalizability of Text-to-SQL

Jieyu Li[1], **Lu Chen**[1*], **Ruisheng Cao**[1], **Su Zhu**[2], **Hongshen Xu**[1], **Zhi Chen**[1]
Hanchong Zhang[1] **and Kai Yu**[1*]
[1]X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, AI Institute
Shanghai Jiao Tong University, Shanghai, China
[2]AISpeech Co., Ltd., Suzhou, China
{oracion,chenlusz,211314,xuhongshen,zhenchi713}@sjtu.edu.cn
{zhanghanchong,kai.yu}@sjtu.edu.cn

Abstract

Exploring the generalizability of a text-to-SQL parser is essential for a system to automatically adapt the real-world databases. Previous investigation works mostly focus on lexical diversity, including the influence of the synonym and perturbations in both natural language questions and databases. However, the structural variability of database schema (DS), as a widely seen real-world scenario, is yet underexplored. Specifically, confronted with the same input question, the target SQL may be represented in different ways when the DS comes to a different structure. In this work, we provide in-depth discussions about the schema generalizability challenge of text-to-SQL tasks. We observe that current datasets are too templated to study schema generalization. To collect suitable test data, we propose a framework to generate novel text-to-SQL data via automatic and synchronous (DS, SQL) pair altering. When evaluating state-of-the-art text-to-SQL models on the synthetic samples, performance is significantly degraded, which demonstrates the limitation of current research regarding schema generalization.

1 Introduction

Given the corresponding database, text-to-SQL (Yu et al., 2018) aims to convert a natural language (NL) utterance into a structured SQL program. Recently, many advanced text-to-SQL models, such as RAT-SQL (Wang et al., 2019) and LGESQL (Cao et al., 2021), have been proposed to tackle this task.

Although significant progress has been achieved considering the ultimate accuracy, many researchers point out that actual performances of current text-to-SQL systems are over-estimated. Suhr et al. (2020) observed a dramatic performance decline when evaluating a state-of-the-art model on unseen datasets. Gan et al. (2021a) discovered that current parsers are vulnerable to the adversarial attack from synonyms of words in user questions. To explore the generalizablity, previous literature mainly focused on the semantically diversity of natural language. However, the topological feature of database schema is also important but is less investigated while studying the generalizablity in text-to-SQL tasks.

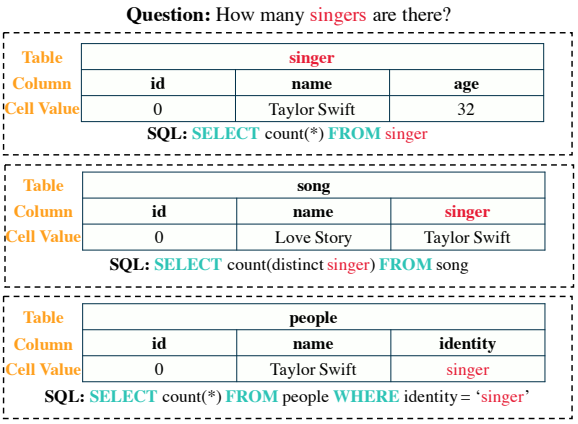

Figure 1: Given the same question, the target SQL responds in different ways when the database schema is different.

We named the ability to automatically adapt different schema the **schema generalizablity**. Different databases will lead to completely divergent SQL queries even given the same user question. For example, in Figure 1, the SQL queries become different when the query entity "*singer*" functions as a column, a table, or a specific cell value, depending on the ontology of the corresponding DS. Furthermore, although the current cross-domain text-to-SQL datasets use different databases during training and evaluation, they are insufficient for evaluating the schema generalizablity of text-to-SQL systems. In Section 3.1, we observe that models can predict the structure of SQL queries even without the corresponding database, which may result from the limited database structures of current datasets.

[*]The corresponding authors are Lu Chen and Kai Yu.

Findings of the Association for Computational Linguistics: ACL 2023, pages 1344–1360
July 9-14, 2023

In this work, we focus on studying the schema generalizablity of current STOA text-to-SQL systems and provide in-deep analysis. To avoid the aforementioned problems in existing datasets, we propose a data- and structure-driven framework to automatically synthesize new (DS, SQL) pairs given the same input question. The framework modifies the DS with a modest annotation cost and updates the SQL synchronously by altering the abstract syntax tree (AST). Inspired by the entity-relationships diagram (E-R Diagram) (Ling, 1985; Li and Chen, 2009), all the transformations follow the entity relationships of the database to guarantee that the modifications are reasonable. We also compared the execution results between new and original (DS, SQL) pairs to ensure the correctness of SQL updating.

Our experiments demonstrate that all four strong text-to-SQL models (RATSQL (Wang et al., 2019), LGESQL (Cao et al., 2021), T5 (Raffel et al., 2020), and T5-PICARD (Scholak et al., 2021)) suffer from poor schema generalizability. After generating the adversarial set from the Spider (Yu et al., 2018) dev set, adding perturbations to the database schema reduces the EM accuracy from an average of 67% to 35%. Even the performance on the adversarial set from the Spider training set drops dramatically (-46 points on EM). Furthermore, we observe that the adversarial examples that both DS and SQL changed are much more challenging for text-to-SQL models than those examples that only DS changed. Finally, we discuss the efficiency of additional training on adversarial examples (Jia and Liang, 2017). Experiment results show that the performance improvement mostly stems from the additional question-DS patterns by more training examples.

The main contributions are as follows:

- We propose a data- and structure-driven framework which can automatically synthesize samples containing unseen (DS, SQL) patterns with minimal human labor. This framework and corresponding synthesis data will be publicly available at https://github.com/Auracion/schema_generation_framework.
- By utilizing the plug-and-play framework, we synthesize a test suite and demonstrate the poor performance of existing text-to-SQL models regarding schema generalization.
- We analyze the reasons leading to modest generalization towards perturbations of synchronous changes in (DS, SQL) pairs and demonstrate that adversarial training is a possible way to inhibit the overfitting problem.

2 Background and Related Work

Structural Features in Text-to-SQL Tasks Modeling the structural information in a database and designing an efficient algorithm to decode structured output sequences are crucial in text-to-SQL. Several studies achieved remarkable progress using GNN (Scarselli et al., 2008) to encode the schema linking, which enhanced the graph structure of DS and the relationships between DS and question tokens (Bogin et al., 2019; Lin et al., 2020; Chen et al., 2021; Hui et al., 2022; Wang et al., 2019; Cao et al., 2021). Another line of research focuses on the grammar structure of SQL. Corresponding works proposed novel algorithms to precisely decode according to the syntax (Guo et al., 2019; Rubin and Berant, 2021; Gan et al., 2021b). Recent works attempted to utilize the developed generative pre-trained language models (Raffel et al., 2020; Lewis et al., 2020) to generate SQL. Based on T5 (Raffel et al., 2020), Scholak et al. (2021) proposed a rule-based post-processor to prune syntax-illegal SQL subsequence in beam search, and they achieved stable improvement in the end-to-end text-to-SQL system.

Synthetic Data	Lexical		Structure	
	Question	Schema	Schema	SQL
Spider-Syn(Gan et al., 2021a)	✓	✗	✗	✗
MR-UT(Ma and Wang, 2021)	✓	✗	✗	✗
MR-ST(Ma and Wang, 2021)	✗	✗	✓	✗
ADVETA-RPL (Pi et al., 2022)	✗	✓	✗	✗
ADVETA-ADD (Pi et al., 2022)	✗	✓	✓	✗
Unaffected	✗	✗	✓	✗
Affected	✗	✗	✓	✓

Table 1: Setups of previous evaluation datasets and our synthetic samples. The synthetic evaluation data was modified from Spider. Unaffected and Affected are two types of data we synthesize in this work. We introduce them in Section 6 The mark ✓represents that the corresponding attribute is different from that in Spider. Oppositely, we use ✗to note in this table.

Robustness of text-to-SQL models Early datasets (Dahl et al., 1994; Hemphill et al., 1990; Zelle and Mooney, 1996; Tang and Mooney, 2000; Li and Jagadish, 2014; Yaghmazadeh et al., 2017; Iyer et al., 2017; Finegan-Dollak et al., 2018) only considered the text-to-SQL tasks on

a single database. To build a robust text-to-SQL model that can automatically adapt unseen domain data, Recent works (Yu et al., 2018; Zhong et al., 2017) collected cross-domain text-to-SQL datasets. Based on the cross-domain setup, researchers further considered some different real-world scenes and proposed corresponding datasets (Yu et al., 2019b,a; Wang et al., 2020). However, Suhr et al. (2020) observed that the execution (EX.) accuracy of a well-trained model on Spider (Yu et al., 2018) always decreases remarkably on the unseen domain data from other datasets[1]. Although Suhr et al. (2020) depicted the reasons leading to performance decline, in-deep discussions are necessary. To this end, recent studies generated synthetic data under different setups to further assess the practical model generalization in different environments. We summarize the characteristic of the synthetic evaluation set in Table 1. In respect of text, Gan et al. (2021a) generated evaluation samples by replacing the schema-related words in NL questions with synonyms. Ma and Wang (2021) substituted the aggregation-related words and prefix phrases with synonym representations. Pi et al. (2022) modified the column names in DS. In respect of structure, Ma and Wang (2021) created different DS structures by imposing perturbations. Pi et al. (2022) added adversarial columns in DS. However, the golden SQLs in both of their synthetic datasets remain unchanged when applying perturbations. In this work, we consider both changed and unchanged golden SQLs to provide a comprehensive appraisal regarding schema generalization.

Dataset	RATSQL	LGESQL
Spider (Yu et al., 2018)	69.57	70.11
SParC (Yu et al., 2019b)	42.20	43.59
Spider-Syn (Gan et al., 2021a)	49.81	50.93
Academic (Li and Jagadish, 2014)	6.26	7.36
GeoQuery (Zelle and Mooney, 1996)	7.51	7.86
IMDB (Yaghmazadeh et al., 2017)	18.96	18.74
Restaurant (Tang and Mooney, 2000)	0.00	0.53
Scholar (Iyer et al., 2017)	0.18	0.24
Yelp (Yaghmazadeh et al., 2017)	6.01	7.51

Table 2: Models are trained on Spider, while evaluated on other datasets. The databases of SParC and Spider-Syn are similar to Spider.

[1]Considered most related studies report the EM. accuracy as the results, we additionally reproduce the experiments and use the EM. accuracy as the metric and illustrate the results in Table 2.

3 Suitable Evaluation Data

To evaluate the schema generalizability of text-to-SQL models, a test dataset with novel databases is crucial. However, current text-to-SQL datasets are not suitable because of the over-templated features (Section 3.1). Therefore, we propose a data- and structure-driven generation framework to synthesize relevant data to assess the generalization capability (Section 3.2).

Question	How many dogs have not gone through any treatment?
SQL	`SELECT count(*) FROM Dogs WHERE Dogs.dog_id NOT IN (SELECT Treatments.dog_id FROM Treatment)`
Syntax Roles	Select Aggregation `WHERE` Condition Nested SQL in Condition

Table 3: The structure of SQL can be represented with the syntax roles.

3.1 Current Datasets are Undesirable

To verify that current text-to-SQL datasets are over-templated, we conduct a syntax role prediction experiment. As the example shown in Table 3, the structure feature of SQL can be represented using the syntax roles. We show the details of all used syntax role labels in Appendix D

Syntax Role Prediction aims to predict which SQL syntax roles are mentioned in the query, including the SQL keywords, nested structure, and aggregation clause. For the user question $\mathbf{Q} = (q_0, q_1, ..., q_{|\mathbf{Q}|})$, the given database schema $\mathcal{D}$, and the corresponding SQL $\mathbf{S} = (s_0, s_1, ..., s_{|\mathbf{S}|})$. Set $\mathcal{R} = \{r_0, r_1, ..., r_{|\mathcal{R}|}\}$ contains all the predefined syntax roles involved in $\mathbf{S}$. We formulate the syntax role prediction task as

$$\mathcal{R} = \mathcal{F}(\mathbf{X}) \tag{1}$$

, where $\mathbf{X} = \mathbf{Q}$ if using database schema information otherwise $\mathbf{X} = (\mathbf{Q}, \mathcal{D})$. The metric used in this experiment is joint accuracy, which means the case is treated as correct if and only if all the syntax roles are correctly predicted.

In this experiment, we compare the performances of whether it contains database schema in the inputs. For the model only uses user questions, we encode the inputs as

$$\mathbf{h} = \text{Bert}(\mathbf{Q}). \tag{2}$$

Train.	Test.	Test. Setup	w/o. DB Schema	w. DB Schema
Spider Train.	Spider Dev.	Spider-like Cross-Domain	86.08	87.34 $\uparrow_{1.26}$
Spider Train.	Spider-Syn Dev.	Spider-like Cross-Domain	85.59	84.72 $\downarrow_{0.87}$
Spider-Syn Train.	Spider-Syn Dev.		85.69	85.40 $\downarrow_{0.29}$
Spider Train.	SParC Dev.	Spider-like Cross-Domain	74.31	74.48 $\uparrow_{0.17}$
SParC Train.	SParC Dev.		66.92	66.50 $\downarrow_{0.42}$
Spider Train.	Academic	Single-Domain	92.27	89.50 $\downarrow_{2.77}$
	GeoQuery		51.42	45.57 $\downarrow_{5.85}$
	IMDB		90.83	93.58 $\uparrow_{2.75}$
	Restaurants		75.20	89.60 $\uparrow_{12.40}$
	Scholar		67.66	71.00 $\uparrow_{3.34}$
	Yelp		96.40	93.69 $\downarrow_{2.71}$
Average Joint Accuracy			79.31	80.13 $\uparrow_{0.82}$

Table 4: Experiment results of syntax role prediction. **w/o. DB Schema** represents a vanila model using BERT-base to encode user questions. **W. DB Schema** represents the model using RAT encoder to process the user questions and database schema.

For the model uses both user questions and database schema information, we encode the input as

$$\mathbf{h} = \text{RAT-Encoder}(\mathbf{Q}, \mathcal{D}), \quad (3)$$

where RAT-Encoder is the encoder of RAT-SQL (Wang et al., 2019). We calculate the probability of using the role r_i as

$$P(\hat{y}_i|\mathbf{X}) = \text{Sigmoid}(\mathbf{v}_i^\top \mathbf{h}), \quad (4)$$

where $\mathbf{v}_i$ is learnable parameters corresponding to syntax role r_i.

As the results in Table 4 shown, the performances of the models without using DS information (column 4) achieve 79.31 on average. The model can directly predict the approximate structure of the target SQL only with the user question most of the time, even though the databases for training and for testing are not overlapping. Meanwhile, the performances of the models using database schema information (column 5) achieve 80.13 on average. The experiment results illustrate that using DS information can only improve 0.82 on average. The performance differences between using and without using DS demonstrate that the DS information is helpless for predicting the SQL structure. Additionally, we find that the phenomena not only happen when evaluating on Spider-like datasets but also exist in other text-to-SQL datasets. Therefore, we suspect that current datasets are too templated to evaluate the generalizability using them. To this end, we need to synthesize suitable evaluation data.

3.2 Evaluation Data Generation

To assess the structural generation capability, we propose a data- and structure-driven generation framework to synthesize relevant data. The synthetic data in this paper are modified from Spider (Yu et al., 2018) [2] which is the most popular cross-domain text-to-SQL dataset. It contains 8659 training examples and 1034 validation examples across 146 databases. The test dataset is unseen and contains 2147 samples with 40 databases.

For a given sample, we synthesize a new sample by altering the DS while keeping the question constant. In order to obtain a reasonable DS, we construct the entity-relationship graph of the given DS and apply graph-based transformations. Moreover, we synchronously update the SQL by modifying the abstract syntax tree. We show more details in Appendix A

In this work, we use four different transformations in DS. Figure 2 illustrates the examples of each transformation, and we show a brief introduction below:

- **Entity to Attribute (E2A)** merges two tables into one.
- **Concept to Attribute (C2A)** converts the concept[3] of an entity, which represents via table name in DS, to its attribute.
- **Named to Unnamed (N2U)** replaces the table corresponding to a relationship with foreign

[2] https://yale-lily.github.io//spider.

[3] It refers to the definition of concept node in the knowledge graph.

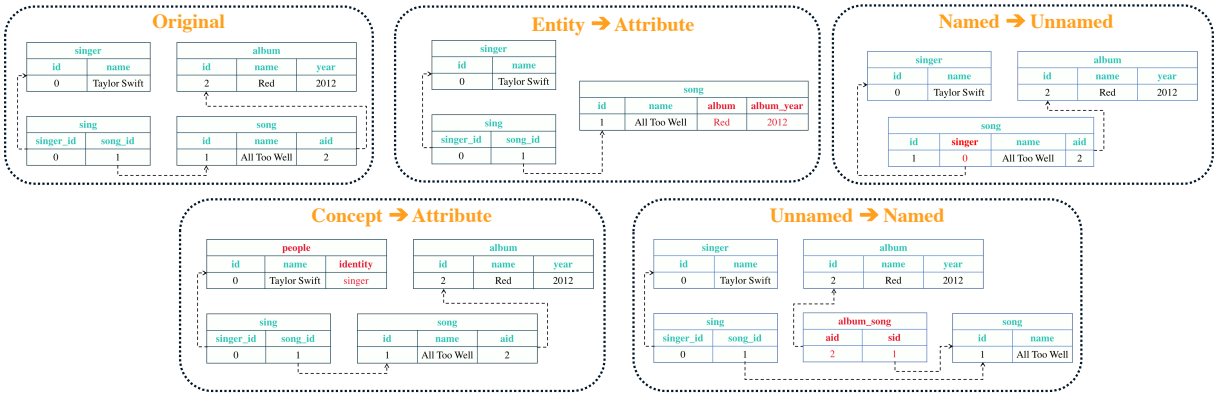

Figure 2: Examples of the DS synthesized via different transformations. The dotted lines denote foreign keys (from foreign key to primary key)

keys.

- **Unnamed to Named (U2N)** replaces a foreign key with a relationship table.

Table 5 shows the total number of each kind of synthetic data synthesized via different E-R transformations. We evaluate the synthetic quality by comparing the execution results of the original and synthetic (DS, SQL) pairs. Over 90.43% generated samples kept consistent execution results on average. In this work, we only consider 1-step transformation regard of the problem of textual noise accumulation in automatic multi-step transformation.

Trans.	Train.		Dev.	
	Affected	Unaffected	Affected	Unaffected
E2A	3035	9466	493	1477
C2A	2659	4271	379	445
U2N	2969	12910	114	376
N2U	2605	48507	303	4484

Table 5: Statistics of generated data for four transformations. **Affected** represents the samples containing different SQL. **Unaffected** represents the samples that the SQL query does not change when applying DS transformations.

4 Generalization Evaluation

In this section, we conduct experiments to evaluate the practical generalization of current text-to-SQL models:

4.1 Experiment Setup

In this work, we experiment with two grammar-based SOTA text-to-SQL parsers, RATSQL (Wang et al., 2019) and LGESQL (Cao et al., 2021). Besides, we also experiment with the T5-based end-to-end text-to-SQL parser, including the methods of decoding with and without PICARD(Scholak et al., 2021). The evaluation metric we use to report the results is exact set match accuracy (EM). Results are averaged over three trials to reduce variance.

Equivalent Test Set (ETS) To precisely evaluate the model robustness, we construct an equivalent test set for the given dataset, which contains the same number of samples. We restrict that each sample in the original dataset matches exactly one synthetic variant in the ETS. If a sample can not generate a variant, we will add the duplication in the ETS. In this work, the percentages of these samples in ETS are 34.3% and 14.7% for affected and unaffected respectively. Furthermore, to reduce the influence of hardness[4], we utilize a heuristic algorithm to modulate the ETS so that its distribution is close to the original dataset. We show more details of the algorithm in Appendix E

4.2 Practical Schema Generalization

We construct the equivalent test set (ETS) for both of the training set and the development set of Spider. The training data in this experiment is the Spider training set. We compare the performances on the Spider training set, Spider development set, and their corresponding ETSs.

Experiment results (Spider Train. vs Spider Train. ETS) illustrated in Table 6 indicate that the perturbation applied to the database schema will disturb the parsing process. The models can not precisely infer the representation of the SQL

[4]The hardness rate is used to represent the complexity of a SQL. In this work, we follow the calculation method proposed in Spider(Yu et al., 2018)

Model	Spider Train.	Spider Train. ETS	Spider Dev.	Spider Dev. ETS
RATSQL	98.19	60.81$\downarrow_{37.82}$	69.83	44.68$\downarrow_{25.15}$
LGESQL	98.94	62.73$\downarrow_{36.21}$	70.57	45.10$\downarrow_{25.47}$
T5	81.20	26.62$\downarrow_{54.58}$	59.09	49.23$\downarrow_{34.33}$
T5-Picard	81.28	26.75$\downarrow_{54.53}$	67.60	26.98$\downarrow_{40.62}$

Table 6: Models are trained on the Spider training set. The backbones of RATSQL and LGESQL are Bert-base (Devlin et al., 2019). The size of T5 is also base.

query when confronting novel DS structures despite the questions and the other parts of the DS being the same as they appeared in the training phrase. When it comes to the development set, as well as the corresponding ETS, databases are completely novel because they do not overlap with the databases in the training set. However, experiment results (Spider Dev. vs Spider Dev. ETS) in Table 6 illustrate a dramatical performance decline. These phenomena demonstrate that the practical schema generalization capability is also modest, which is similar to the structural robustness. Therefore, we suspect that current text-to-SQL parsers can not automatically infer the SQL pattern according to the DS. We will discuss the true reason that caused this issue in the next section.

5 Discussion about Schema Generalizability

In this section, we discuss the schema generalizability of text-to-SQL by answering the following questions:

- Q1: What is the actual function of database schema input? (Section 5.1)
- Q2: What is the actual reason causing the modest generalizability? (Section 5.2)

5.1 Function of Database Schema Input

To answer Q1, we first verify that the database schema (DS) information is independent of the process of constructing SQL patterns. Reviewing the experiments in Section 4, models always make mistakes when facing out-of-dataset (OOD) DS. To estimate whether the OOD structure confuses the parsers, we consider the evaluation data containing OOD DS while keeping the SQL query unchanged.
Setup: Different from the evaluation data used in Section 4, we generate the data with different DS but the same SQL. For each piece of data, the DS transformations are applied to untapped parts so that the SQL will not be influenced. Similarly, we also construct the equivalent test set (ETS) for the Spider development set with this kind of synthetic data. The training data in this experiment is the training set of Spider.

Model	Test Data	EM. Acc.
RATSQL	Spider Dev.	69.83
	Spider Dev. ETS	67.67$\downarrow_{2.16}$
LGESQL	Spider Dev.	70.57
	Spider Dev. ETS	67.41$\downarrow_{3.16}$
T5	Spider Dev.	59.09
	Spider Dev. ETS	49.23 $\downarrow_{9.86}$
T5+Picard	Spider Dev.	67.60
	Spider Dev. ETS	56.38 $\downarrow_{11.22}$

Table 7: EM. accuracy on the evaluation data synthesized from the Spider training set. The SQL in synthetic data is consistent with the original.

Results of using a grammar-based decoder (line 1-4) shown in Table 7 demonstrate that the OOD structure does not influence the inference process. Reviewing the syntax role prediction experiments discussed in Section 3.1, we suggest that current text-to-SQL models construct SQL query via sentence pattern of the user question rather than the actual structure of DS. We suspect that the function of DS input is providing the correct presentation of the SQL non-keywords (table name, column name, and value). The efficiency of using schema linking provides a strong signal on the target database item. Once the explicit relationships between these SQL non-keywords and the presentations in question are destroyed, models will make mistakes in selecting the correct schema item. However, the SQL structure is always predicted in the correct ways(Gan et al., 2021a). Results of using a token-based decoder (line 5-8) in Table 7 illustrate the remarkable performance decline, which seems in contrast to the previous conclusion. We analyze the error cases and suggest that this issue is caused by the unnatural schema item names, which we report in Limitations.

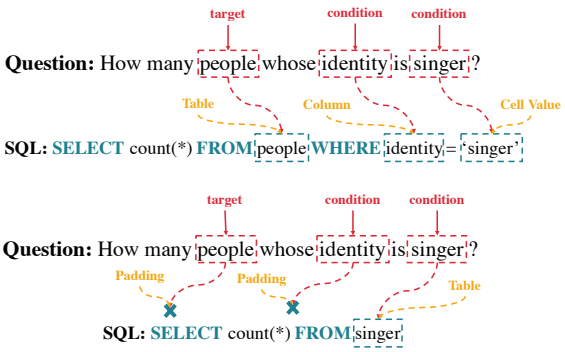

Figure 3: Different combinations of NL role and DS role determine different SQL sketches.

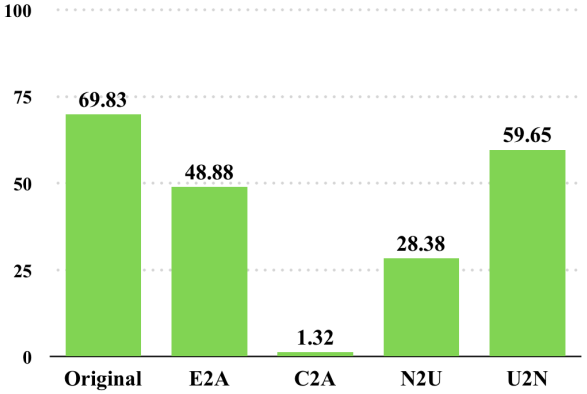

Figure 4: Training without extra data, evaluating on different synthetic samples.

DS Role	Train.		C2A.	
	Target	Condition	Target	Condition
Table	9.09%	42.62%	5.00%	15.18%
Column	81.82%	24.04%	83.33%	21.43%
Cell Value	0.00%	25.68%	11.67%	58.04%
Padding	9.09%	7.65%	0.00%	5.36%

Table 8: Distribution of (NL, DS) pattern in the training set and C2A samples.

5.2 (NL, DS) Pattern

To answer Q2, we first introduce the concept of (NL, DS) pattern. The (NL, DS) pattern represents the combination of a natural language (NL) role and a database schema (DS) role. Then we will illustrate how the (NL, DS) pattern influences the generalizability.

NL Role: As the examples in Figure 3 shown, we assume that some words (except stop words) in the NL question describe the key information of the query. In this section, we simply split these keywords into two categories[5], *target* and *condition*, which we call them the NL role of these words. *target* represents the querying entity we focus on. For instance, in the first example, we attempt to obtain the number of "*people*", and "*people*" is a *target* in this case. *condition* represents the constraint of the *target*. For example, the specific "*people*" we querying is restricted with the condition "*whose identity is singer*". Therefore, the *condition* keywords are "*identity*" and "*people*". The NL roles are DS-independent, in other words, they only depend on the semantics of the NL question.

DS Role: For a DS, some elements link the keywords in the given question, such as the word "*singer*" in the first case, and all of them play a unique role in the given DS. We define the DS role as *table*, *column*, *cell value* and a padding role to link the non-schema-related keywords, for instance, the word "*people*" in the second case.

(NL, DS) Pattern: For each of these elements, we named the combination of an NL role and a DS role as an (NL, DS) pattern, which determines the syntax role in SQL. For example, the element "*singer*" in the first case functions the NL role *condition* and the DS role *cell value*, in this case, it locates in the `WHERE` clause. However, when the DS role comes to *table*, as shown in the second case, the element "*singer*" will locate in the `FROM` clause. For the given NL question and DS, the structure of the SQL query depends on the containing (NL, DS) patterns.

We assume that the modest generalization capability is because of the over-fitting of (NL, DS) patterns. Unseen (NL, DS) patterns in the evaluation stage lead to failed parsing. To verify it, we first evaluate the performance on the samples synthesized via different E-R transformations. The experiment results are illustrated in Table 4. We notice that models make mistakes on almost all the samples generated using C2A E-R transformation. Actually, C2A is a special transformation that must create an (NL, DS) pattern, (*target*, *cell value*). In general, this pattern represents a condition in the `WHERE` clause. On the other hand, we randomly sample 100 pieces of data from the training set and the synthetic C2A data to evaluate whether (*target*, *cell value*) is not in the original dataset but appears in the synthesis dataset. Table 8 shows the statistic results of manually calculating the distribution of (NL, SQL) patterns. The combination (*target*, *cell value*) is not contained in the training set but exists as unseen patterns when it comes to

[5]Notice that we only introduce a kind of simple splitting way in this section. It is more complicated in the real world.

Test Data	Training Data	RATSQL	LGESQL	T5	T5+PICARD
Spider Dev.	Spider Train.	69.83	70.57	59.09	67.60
	Spider Train. + Affected	70.12 $\uparrow_{0.29}$	69.89 $\downarrow_{0.68}$	58.22 $\downarrow_{0.87}$	67.70 $\uparrow_{0.81}$
	Spider Train. + Unaffected	70.38 $\uparrow_{0.55}$	70.05 $\downarrow_{0.52}$	58.99 $\downarrow_{0.16}$	66.73 $\downarrow_{0.04}$
Spider Dev. ETS	Spider Train.	44.68	45.10	24.76	26.98
	Spider Train. + Affected	67.21 $\uparrow_{22.53}$	67.57 $\uparrow_{22.47}$	51.64 $\uparrow_{26.01}$	58.68 $\uparrow_{31.70}$
	Spider Train. + Unaffected	45.23 $\uparrow_{0.55}$	45.17 $\uparrow_{0.07}$	23.98 $\downarrow_{0.78}$	26.40 $\downarrow_{0.58}$

Table 9: Results of using extra synthetic samples in the training stage.

C2A samples. We additionally enumerate some typical error cases in Appendix C. The examples demonstrate that models tend to parse according to experiences so that they make mistakes on novel patterns. In this case, we suggest that the actual reason causing the modest generalization capability is the (NL, DS)-pattern-wise over-fitting.

6 Pattern-Specific Adversarial Training

In this section, we study whether adversarial training can improve structural generalization by evaluating the efficiency of training with extra synthetic data.

Setup: We conduct experiments to on both original and synthetic evaluation data. As the adversarial training, we train models with the original training set of Spider and additional synthetic data with a 1:0.2 ratio. We consider two kinds of extra synthetic training data in these experiments. The one is the data containing novel database schema (DS) and different SQL queries (compared with the original data). They are similar to the evaluation data we used to build the ETS in Section 4. We named these data **Affected**. The other are the data containing novel DS while the same SQL queries, which is similar to the data from the Dev. ETS in Section 5.1. We named them **Unaffected**. The synthetic evaluation data we used in this section is **Affected**.

We report our results in Table 9. Experiment results in the upper block (line 1-3) illustrate that neither affected nor unaffected extra training data can improve the performance on the original development set. The reason is that extra training data do not provide the (NL, DS) patterns which are rare in the original training set but appear in the original development set. Actually, the problem of over-templated demonstrates that it is hard to find the aforementioned patterns. Unlikely, the transformations applied in this work either do not guarantee these patterns are created.

Experiment results in the lower block (line 4-6) of Table 9 show that the affected extra training data is helpful to improve the performance on the synthetic evaluation data. However, the usage of unaffected data can not. The reason is that the former provides the (NL, DS) patterns which are rare in the original training set while are contained in the ETS. On the other hand, the latter do not provide any of these patterns because the perturbations are applied on untapped parts of DS in unaffected data. Therefore, we suggest that specific adversarial training can enhance the model despite it can not be verified on current datasets. This experiment amplifies the improvement of adversarial training by increasing the overlap of (NL, DS) patterns between the extra training data and the synthetic evaluation data. Therefore, we suggest that adversarial training is a possible way to improve structure generalization capability, and it needs more investigation in future works.

6.1 Not Only in Cross-Domain

Actually, the problem of (NL, DS)-pattern-wise overfitting is not the specific problem that only exists in cross-domain text-to-SQL. Modest structural generalization is just one of the phenomena under a cross-domain setup. Single-domain text-to-SQL also has the same problem.

From the view of (NL, DS) patterns, the deficiency of patterns in the training stage leads to the appearance of unseen patterns in the test stage and further causes performance decline. However, leaving out patterns is inevitable during the data collection process. Annotators can neither ensure to ask questions in all possible sentence patterns nor guarantee that all combinations of schema items are considered. For instance, as the example illustrated in Figure 1, confronting the third DS, annotators may not come up with the question about "*singer*", or they may ask in the way of *How many people whose identity is a singer?*. In this case, automatically addressing unseen patterns is also essential in single-domain text-to-SQL.

7 Conclusion

In this work, we first report that current text-to-SQL datasets are too templated to investigate generalization capability. To this end, we constructed a generation framework to synthesize text-to-SQL data for evaluation in this work. Experiment results illustrate that the model generalization is poor. Furthermore, the analysis illustrates that the problem is caused by the overfitting of (NL, DS) patterns. Finally, we demonstrate that when adding extra training data to bring more unseen patterns in the evaluation stage, the performance will improve. Adversarial training is a possible way to enhance the text-to-SQL parser.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable comments. This work has been supported by the China NSFC Project (No.62106142 and No.62120106006), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and Startup Fund for Youngman Research at SJTU (SFYR at SJTU).

Limitations

The main limitation is the lexical noise in the automatic synthesizing process. We rename the related tables and columns by a series of rules. Therefore, naturalness is not always sufficient. For example, we create and rename a table by combining two table names in some cases. It will lead to a long table name with too much redundant noise. Therefore, we only considered the one-step transformations in this work to inhibit the influence of lexical noise accumulation.

References

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. *arXiv preprint arXiv:1908.11214*.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. *arXiv preprint arXiv:2106.01093*.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. Shadowgnn: Graph projection neural network for text-to-sql parser. *arXiv preprint arXiv:2104.04689*.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2106.01065*.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021b. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

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, Online. Association for Computational Linguistics.

Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Qing Li and Yu-Liu Chen. 2009. *Entity-Relationship Diagram*, pages 125–139. Springer Berlin Heidelberg, Berlin, Heidelberg.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *arXiv preprint arXiv:2012.12627*.

Tok Wang Ling. 1985. A normal form for entity-relationship diagrams. In *Proceedings of the Fourth International Conference on Entity-Relationship Approach*, page 24–35, USA. IEEE Computer Society.

Pingchuan Ma and Shuai Wang. 2021. Mt-teql: Evaluating and augmenting neural nlidb on real-world linguistic and schema variations. *Proc. VLDB Endow.*, 15(3):569–582.

Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2007–2022, Dublin, Ireland. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*, pages 12–21, Online. Association for Computational Linguistics.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388. Association for Computational Linguistics.

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.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

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.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. 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, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

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

A Generation Framework

The overview of the generation framework is shown in Figure 5. For a given sample, we synthesize a new sample via altering the DS while keeping the question constant. In order to obtain a reasonable DS, we construct the entity-relationship graph of the given DS and apply graph-based transformations, which we introduce in Section A.1 and Section A.2 respectively. Moreover, we synchronously update the SQL by modifying the abstract syntax tree, and we show more details in Section A.3.

A.1 Entity-Relationship Graph

A relational database organizes data in predefined relationships, which are represented as the structural relationships among tables and columns. To clearly describe reasonable relationships, developers always utilize **E**ntity-**R**elationship Diagram (E-R Diagram) (Ling, 1985; Li and Chen, 2009) to define the relationships among the raw data, which is helpful to design the database structure. Inspired by ER Diagram, we attempt to modify the DS following the entity relationships so that the rationality of the altered DS can be ensured. To this end, we introduce the definition of Entity-Relationship (E-R) Graph in this paper, which evolves from E-R Diagram while leaving out the attributes vertexes to emphasize the topology feature [6]. The vertex in E-R Graph represents an entity, and the edge represents the relationship between the entities that its terminal vertexes correspond. Both the vertex and the edge function as a table in DS. For example, as shown in Figure 5, each of the table *people*, the table *author*, and the table *novel* corresponds to a vertex in E-R Graph, and the table *write* corresponds to an edge.

Thus, to construct the E-R Graph, we manually annotate a binary tag for each table in DS to distinguish between entity and relationship. We label *relationship* following two principles and label the others as *entity*:

Bridge Structure: The given table should contain exactly two foreign keys.

Semantic Convention: The table name should be the combination of two entities such as the relationship *Customer_Addresses* combining *Customer* and *Address*. Apart from that, the phrase obeys human language conventions is also considered. For instance, the relationship *visit* linking *visitor* and *museum*.

A.2 E-R Transformation

E-R transformation is the graph transformation in the E-R graph. There are ten kinds of E-R transformation, containing five operations applied on vertexes or edges. We assume the databases that store the same data in the different schema can transform between each other via a sequence of E-R transformations. We illustrate all kinds of E-R transformations and the corresponding transformations in DS in Appendix. However, some transformations are insecure. For example, the usage of *delete edge* transformation will lead to information loss. Besides, some transformations rely on strict annotation criteria and costly manual labeling. For instance, whether a table can be split into two need rigorous judgment according to the semantic environment. In this work, we use three E-R transformations with no need for additional annotations, and they totally correspond to four different transformations in DS. Figure 2 illustrates the examples of each transformation, and we show more details below.

Entity to Attribute (E2A) corresponds to a kind of *merge vertexes* E-R transformation. For a pair of vertexes in the E-R graph, we split them as a

[6] The attributes node in E-R Diagram refers to a column in the database. To emphasize the topology feature, we replace *Diagram* with *Graph*.

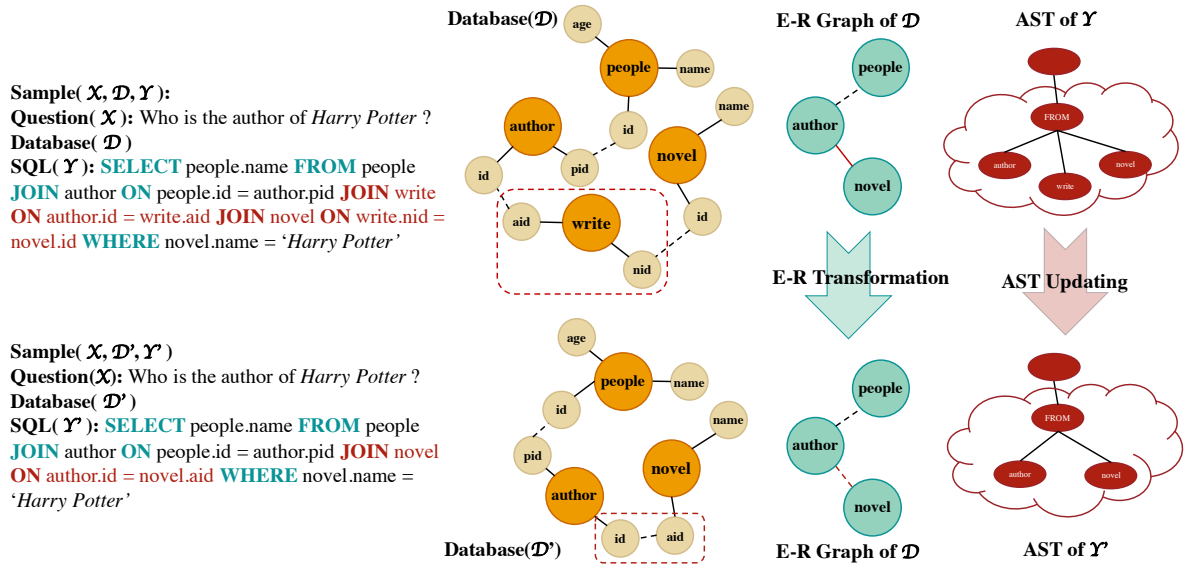

Figure 5: Overview of the generation framework we proposed. The upper part illustrates the original sample, and the lower part illustrates the synthetic sample. The red rectangles and red lines denote the modified parts. For the E-R graph, the dotted lines denote unnamed relationships, and solid lines denote named relationships.

source entity and a target entity. The target entity corresponded table is the only one that contains the foreign keys of the source entity corresponded table in the DS. Both of the vertexes can be treated as the source entity as long as the combination is suitable. As the example shown in Figure 2, for the attributes in the source entity, we convert them into the new attributes in the target entity. To avoid semantic loss, we rename the attributes following rules. Besides, we utilize a series of rules to recognize a special column as the agent of the entity, such as the column `name`, and it will be used to replace the foreign key.

Concept to Attribute (C2A) corresponds to a kind of *modify vertex* E-R transformation. Different from the column-wise modification, we focus on altering the role of the table. We attempt to convert the concept[7] of an entity, which represents via table name in DS, to its attribute. Firstly, we detect a high-level category of the entity using a pre-trained NER model (Qi et al., 2020). In the example shown in Figure 2, `people` is the high-level category of `singer`. Then, we create an additional attribute to store the concept by rules. In this case, we use the new column `identity` to record the concept `singer`.

Named to Unnamed (N2U) corresponds to a kind of *modify edge* E-R transformation. We name the relationship represented by a table as **Named**, and that by foreign keys as **Unnamed**. For instance, in the original DS illustrated in Figure 2, the table `sing` is a named relationship and the foreign key `aid` in the table `song` represents an unnamed relationship. We change the type of relationship by creating a foreign key of one table in the other table, as the example shows.

Unnamed to Named (U2N) also corresponds to a kind of *modify edge* E-R transformation, which is the reversed transformation of *Named to Unnamed*. We create a relationship table and name it with the combination of two target table names to store the relationship. Then, we build the connection by transferring the foreign key in the table and creating another foreign key in it, as the example in Figure 2 shows.

A.3 AST Updating

To update the SQL precisely, we construct the AST of the given SQL following grammar rules and alter the SQL by modifying the AST. For each E-R transformation, we detect related subtrees in the AST and apply the corresponding rule to update the subtrees. For instance, we add an additional condition subtree in the corresponding `WHERE` subtrees while applying *concept to attribute* transformation. Finally, we parse the altered SQL with the modified AST.

In this work, we consider two type of synthetic data, **affected** and **unaffected**. Affected samples

[7] It refers to the definition of concept node in knowledge graph.

contain different SQL compared with the original data, and the unaffected contain the same. We distinguish these two types according to whether the SQL involves a DS element that is influenced by the transformation. And the AST updating module is only used to synthesize affected data.

B All kinds of E-R Transformations

Table 10 illustrates all kind of E-R transformations and the corresponding transformation in DS.

	Operation	Transformation in DS
Vertexes	modify	inner modification of a table
	merge	merge two tables
	split	split an table in two
	add	add a table
	delete	delete a table
Edges	modify	conversion between table and foreign keys
	merge	merge two tables/foreign keys
	split	split a table in foreign keys
	add	add a table or a foreign key
	delete	delete a table or a foreign key

Table 10: All kinds of transformation and the necessity of labels. Considered the cost of manual annotation, we only choose three of them in this work.

C Errors Cases in Synthetic Evaluation Data

Examples in Figure 6, Figure 7, and Figure 8 illustrate that models tend to predict following familiar sketch.

D Syntax Roles

Table 11 illustrates all the syntax role labels in the syntax role prediction experiment.

E Hardness-Oriented Sampling Algorithm

Algorithm 1 Hardness-Oriented Sampling Algorithm

Require: original examples $x_1, x_2, ..., x_n$
Ensure: synthesis samples $y_1, y_2, ..., y_n$

$D_x \leftarrow$ *get_hardness_distribution*$(x_1, x_2, ..., x_n)$
initialize the hardness distribution of synthesis samples: $D_y \leftarrow \phi$
$R \leftarrow \phi$
for $i = 1$ to n **do**
  $\mathcal{X}_i \leftarrow$ *generate_all_variants*(x_i)
  if $|\mathcal{X}_i| = 0$ **then**
    $y_i \leftarrow x_i$
    update(D_y, y_i)
  else
    $\mathcal{C}_i \leftarrow$ *find_same_hardness*$(\mathcal{X}_i, x_i)$
    if $|\mathcal{C}_i| > 0$ **then**
      $y_i \leftarrow$ *random_sampling*$(\mathcal{X}_i)$
      update(D_y, y_i)
    else
      R.append$(\mathcal{X}_i)$
    end if
  end if
end for
while $|R| > 0$ **do**
  Find the hardness category with the largest difference between the current distribution and the original distribution: $h \leftarrow$ *find_hard_cat*(D_x, D_y)
  Sample an $\mathcal{X}_i$ from R which contain at least one variant with hardness h
  if Can not sample an $\mathcal{X}_i$ **then**
    break the loop
  end if
end while
return $y_1, y_2, ..., y_n$

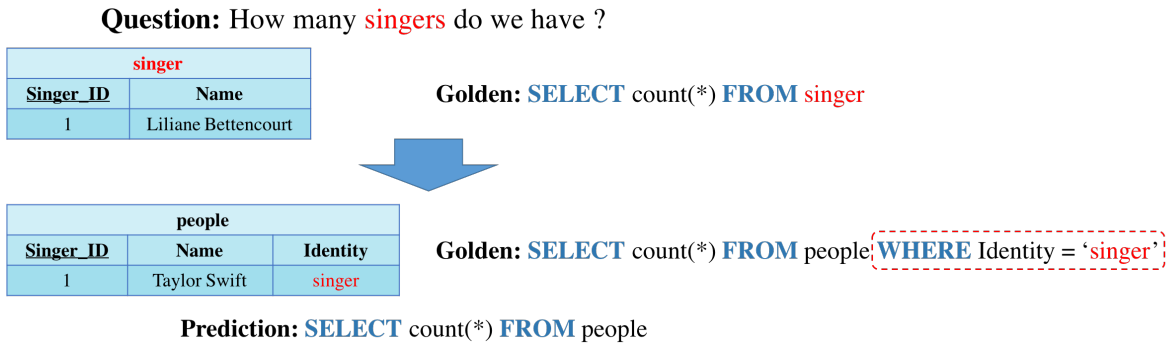

Figure 6: An example of failure prediction.

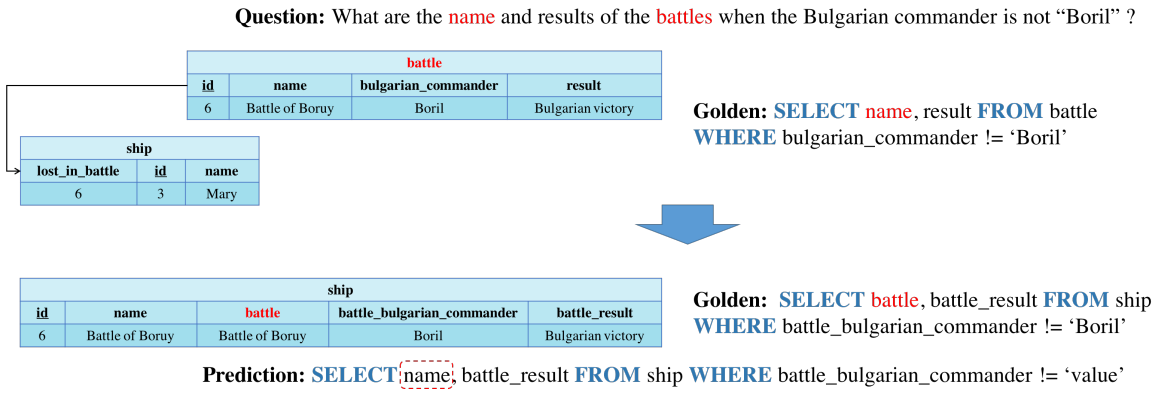

Figure 7: An example of failure prediction.

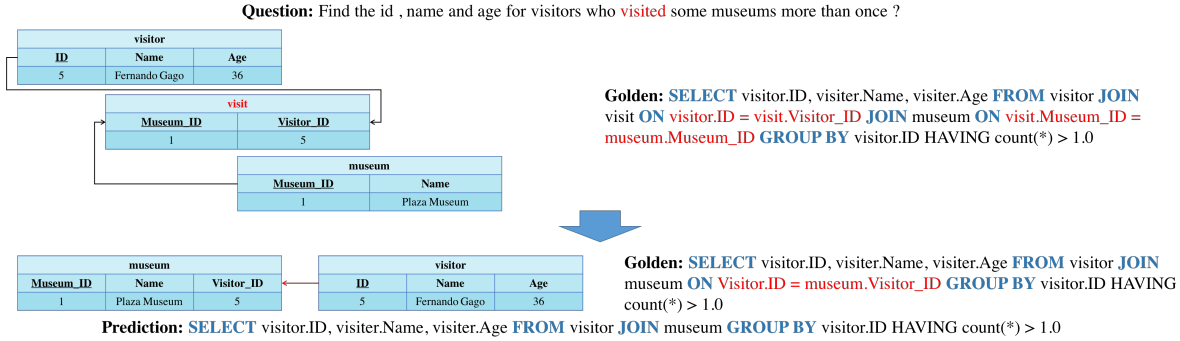

Figure 8: Another example of failure prediction.

Category	Syntax Role	Example
SQL Keyword	WHERE Clause	SELECT count(*) FROM head WHERE age > 56
	GROUP BY Clause	SELECT visitor.ID FROM visiter GROUP BY visitor.ID HAVING count(*) > 1.0
	ORDER BY Clause	SELECT Theme FROM farm_competition ORDER BY YEAR ASC
	LIMIT Clause	SELECT Official_Name , Status FROM city ORDER BY Population DESC LIMIT 1
Aggregation	Select Aggregation	SELECT count(DISTINCT bike_id) FROM trip
	Condition Aggregation	SELECT city , COUNT(*) FROM station GROUP BY city HAVING COUNT(*) >= 15
Nested SQL	UNION	SELECT course_id FROM SECTION WHERE semester = 'Fall' AND YEAR = 2009 UNION SELECT course_id FROM SECTION WHERE semester = 'Spring' AND YEAR = 2010
	INTERSECT	SELECT country FROM people WHERE age < 25 INTERSECT SELECT country FROM people WHERE age > 30
	EXCEPT	SELECT donator_name FROM endowment EXCEPT SELECT donator_name FROM endowment WHERE amount < 9
	Nested SQL in Condition	SELECT count(*) FROM Dogs WHERE Dogs.dog_id NOT IN (SELECT Treatments.dog_id FROM Treatment)
	Nested SQL in FROM Clause	SELECT count(*) FROM (SELECT * FROM endowment WHERE amount > 8.5 GROUP BY school_id HAVING count(*) > 1)

Table 11: All syntax roles.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
9

☑ A2. Did you discuss any potential risks of your work?
9

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

3

☑ B1. Did you cite the creators of artifacts you used?
3

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
3

C ☑ Did you run computational experiments?

3,4,5,6

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
It's not important

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☒ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Our work is not a methodology work.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
3,4,5,6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix A

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

3, Appendix A

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Appendix A

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Appendix A

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
9

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Appendix A