Re-appraising the Schema Linking for Text-to-SQL

Yujian Gan[1] **Xinyun Chen**[2] **Matthew Purver**[1,3]
[1]Queen Mary University of London [2]Google DeepMind [3]Jožef Stefan Institute
{y.gan,m.purver}@qmul.ac.uk xinyunchen@google.com

Abstract

Most text-to-SQL models, even though based on the same grammar decoder [1], generate the SQL structure first and then fill in the SQL slots with the correct schema items. This second step depends on *schema linking*: aligning the entity references in the question with the schema columns or tables. This is generally approached via **E**xact **M**atch based **S**chema **L**inking (**EMSL**) within a neural network-based schema linking module. EMSL has become standard in text-to-SQL: many state-of-the-art models employ EMSL, with performance dropping significantly when the EMSL component is removed. In this work, however, we show that EMSL reduces robustness, rendering models vulnerable to synonym substitution and typos. Instead of relying on EMSL to make up for deficiencies in question-schema encoding, we show that using a pre-trained language model as an encoder can improve performance without using EMSL, giving a more robust model. We also study the design choice of the schema linking module, finding that a suitable design benefits performance and interpretability. Finally, based on the above study of schema linking, we introduce the grammar linking to help model align grammar references in the question with the SQL keywords.[2]

1 Introduction

Recent years have seen great progress on the text-to-SQL problem, i.e. translating a natural language (NL) question into a SQL query (Dong and Lapata, 2018; Yu et al., 2018b; Zhong et al., 2017; Gan et al., 2021a; Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020), with neural networks the *de facto* approach. To achieve good performance on text-to-SQL tasks, a neural model needs to correlate natural language queries with the given database schema, a process called *schema linking*. Previous work often explicitly designs a module to perform the schema linking, which we term Exact Match based Schema Linking (*EMSL*) (Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020). Specifically:

- **Schema linking** is the alignment between the entity references in the question and the schema columns or tables.
- A **schema linking module** is a trainable component that learns to perform schema linking, based on features that relate word tokens in the question to schema items.
- A **schema linking feature** encodes this relational information; e.g., it can represent the similarity between words in the question and schema items.
- **Exact match based schema linking (EMSL)** is a type of schema linking feature obtained by the exact lexical match between the words in the question and words in schema items.

Figure 1 presents an example of schema linking and the EMSL feature matrix. Most previous work relies on this exact lexical matching to obtain schema linking features. Following the work of (Krishnamurthy et al., 2017; Guo et al., 2019; Bogin et al., 2019), EMSL is used in many subsequent works (Wang et al., 2020; Cai et al., 2021; Xu et al., 2021; Lei et al., 2020; Yu et al., 2021; Shi et al., 2021) and has been shown to be effective. For example, the ablation study in Guo et al. (2019) shows that removing the schema linking module incurs the most significant performance decrease.

Although EMSL has been widely used and helps models obtain the state-of-the-art performance on some text-to-SQL benchmarks (Yu et al., 2018b; Zhong et al., 2017), in this work, we show that EMSL renders models vulnerable to noise in the input, particularly synonym substitution and typos. We then investigate whether text-to-SQL models can preserve good prediction performance without EMSL. Previous ablation studies (Guo et al., 2019;

[1]The decoder will repeat the following two steps until a complete SQL is generated: (1) generates a SQL clause keyword; (2) fills in corresponding schema items.

[2]Our code and data are available at Github.

Wang et al., 2020) claiming the necessity of the schema linking module were conducted without pretrained language models (PLMs) such as BERT. In fact, we find that when a pretrained language model is used as a model encoder, removing EMSL has very little impact on the performance of the model. This observation is consistent for different model architectures and training schemes, such as RATSQL (Wang et al., 2020), GNN (Bogin et al., 2019), and GAP (Shi et al., 2021). Based on this finding, we introduce a more reasonable text-to-SQL encoder design.

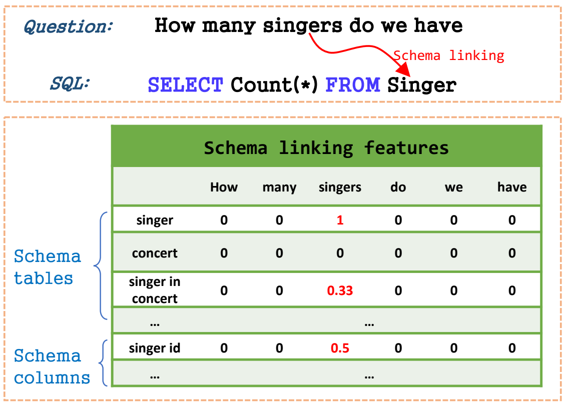
We evaluate the models in three settings: the original Spider benchmark without input noise (Yu et al., 2018b), the Spider-Syn version with synonym substitution (Gan et al., 2021a), and a new typo injection setting. Results show that the use of a PLM can provide the same performance benefit as EMSL, while achieving better robustness against synonym substitution and typos. Removing EMSL also allows the model to obtain better results when training with synonym substitution samples. We also show that MAS (Multi-Annotation Selection, Gan et al., 2021a), a method designed to improve model robustness with EMSL, can also improve models without EMSL. In conclusion, we demonstrate EMSL is no longer a necessary building block of text-to-SQL models.

The EMSL and pretrained language models are part of the schema linking module that learns a score to decide which schema item to select. There are two design choices to compute this score: the first sees it as a direct relation between the question and schema items (Bogin et al., 2019); while the second considers the question and schema items together (Wang et al., 2020). We show that our best model employing the first design choice achieves the state-of-the-art performance, while its schema linking scores are also more interpretable than models with the second design.

Despite the shortcomings of EMSL, it still significantly improves previous models: without a clear idea of how to align the entity references in the question and the schema columns or tables, EMSL plays an important role in alignment. Inspired by EMSL, we propose instead Exact Match Grammar Linking (GL), which improves the model's ability to generate correct SQL keywords.

In short, this work investigates the role of the schema linking module for text-to-SQL models, regarding its effect on model performance and robustness. We summarize our key findings below:

- EMSL is not a necessary building block of text-to-SQL models. A reasonable text-to-SQL encoder design can replace EMSL and make the model more robust.
- Compared to the cross relation score between the question and schema items together, a direct relation score between the question and schema items is more interpretable and can improve the performance of a state-of-the-art model.
- When it is unclear how to design the optimal encoder, EMSL improves the model performance significantly. Inspired by EMSL, we introduce Exact Match Grammar Linking (GL) to generate better SQL keywords.



Schema linking features

	How	many	singers	do	we	have
singer	0	0	1	0	0	0
concert	0	0	0	0	0	0
singer in concert	0	0	0.33	0	0	0
...			...			
singer id	0	0	0.5	0	0	0
...			...			

Figure 1: An example of schema linking and exact match based schema linking (EMSL) feature matrix.

2 Proposal I: Construct the Schema Linking without the EMSL

Our first proposal is that the schema linking should not include EMSL. Schema linking itself is essential for text-to-SQL models, but EMSL can be replaced with a better mechanism.

2.1 Schema Linking Feature

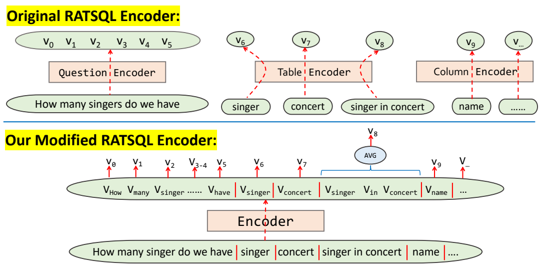
Figure 1 presents an example of schema linking features. The word *'singers'* in the question exactly matches (modulo stemming) the schema table name *'singer'*, giving feature value 1. It does not match the table *'concert'*, giving value 0; and matches one of the three words in *'singer in concert'*, giving value 0.33. Such type of schema linking mechanism based on exact lexical matching (EMSL) is the most common used in existing text-to-SQL models (Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020; Cai et al., 2021; Xu et al., 2021; Lei et al., 2020; Yu et al., 2021; Shi et al., 2021).

Schema linking has been shown to be essential for achieving good performance (Guo et al., 2019; Wang et al., 2020). For example, Wang et al. (2020) consider that the representations produced by vanilla self-attention were insensitive to textual matches even though their initial representations were identical, i.e., EMSL is needed for textual matches. Some works add ConceptNet (Speer and Havasi, 2012) to get more linking features (Guo et al., 2019; Tan et al., 2021) and thus improve the model performance. However, we argue that a well-designed encoder can solve this problem, and note that the feature values in Figure 1 are equal to the average dot product results when using lemma one-hot embeddings, suggesting that a proper embedding can replace EMSL. Appendix E provides previous ablation study on EMSL. We test our new proposal experimentally in Section 5.3.

2.2 Text-to-SQL Encoder Design

For a text-to-SQL encoder, we expect that the correct schema item vectors obtained from the encoder are as close to the question vector as possible. The SQL cares about which schema item to use instead of the words in the schema item. Therefore, unlike keeping every question word vector, only one vector is used to present the schema item even if it contains multiple words. Since both the encoder mechanics and content style are different between question and schema, many models (Yu et al., 2018a; Guo et al., 2019; Wang et al., 2020) uses different encoders to encode the question and schema separately. The upper part of Figure 2 shows a design case, which is the structure of the RATSQL model (Wang et al., 2020) containing three encoders with similar structure and size.

We believe that the shortcoming of the original RATSQL design is the use of three encoders. For example, in the initial state, the parameters of the three encoders are different. Therefore, even though the word *'singers'* appears in the question, the vector v_6 initially generated by the table encoder is probably irrelevant to all vectors output by the sentence encoder. If using EMSL, this does not matter: in both training and evaluation we can link v_6 to v_2 through EMSL. However, without EMSL, we need to relate v_6 from the table encoder to the vectors from the question encoder, which is more challenging to train than using only one encoder, as shown in the lower part of Figure 2. Since the



Original RATSQL Encoder:
v0 v1 v2 v3 v4 v5
Question Encoder
How many singers do we have
v6
v7
v8
Table Encoder
singer
concert
singer in concert
v9
v...
Column Encoder
name
......
Our Modified RATSQL Encoder:
v8
AVG
v0 v1 v2 v3-4 v5 v6 v7 v9 v...
VHow Vmany Vsinger ...... Vhave | Vsinger | Vconcert | Vsinger Vin Vconcert | Vname | ...
Encoder
How many singer do we have | singer | concert | singer in concert | name | ....

Figure 2: The original RATSQL encoder structure (Wang et al., 2020) and our modified version.

output of our modification is the same as the original, it can be easily replaced and connected to the subsequent modules.

In the lower part of Figure 2, our modification is inspired by several text-to-SQL models with BERT, including RATSQL+BERT (Wang et al., 2020; Guo et al., 2019; Zhang et al., 2019). In our modification, RATSQL uses only one encoder instead of three. We believe using three encoders is one of the main reasons why the base RATSQL performance significantly drops when removing EMSL. For the convenience of discussion, we named our modified RATSQL as RATSQL$_O$, where O means one encoder.

RATSQL$_O$ uses only one encoder whose structure and size are the same as the original question encoder. For the schema item representation, RATSQL takes the hidden state after all the words of the entire schema item are encoded, while RATSQL$_O$ takes the average of all word encodings. The advantage of our RATSQL$_O$ is that v_6, v_8, and v_2 initially have a certain similarity, which benefits the schema linking in both single and multi words. RATSQL$_O$ also deals with words outside the embedding vocabulary better than RATSQL. Suppose the word *concert* and *name* are outside the vocabulary: v_7 and v_9 from the RATSQL table encoder will be the same since their inputs are the same UNK vector. However, the RATSQL$_O$ encoder will output different vectors for v_7 and v_9, as the contexts before and after the word *concert* and *name* are different. In this way, even if there are multiple UNK words, the RATSQL$_O$ encoding vector will be different.

3 Proposal II: Schema Linking Module Design Choice

We believe that a text-to-SQL model with good performance can ignore the schema linking feature, but it must include a schema linking module. While implementation details of such models differ,

Labels	SQL Keywords	Matching Rules
DB	WHERE	The words exact match to the database value.
BCOL	WHERE	Words to successfully build the EMSL to Boolean-like columns
COL, TABLE	NONE	Words to successfully build the EMSL.
AGG	Aggregation Function	Matching words: average, maximum, minimum, etc.
PDB	WHERE	Matching words in quotes.
GRSM	>, <	Matching words: before, after, greater, smaller, etc.
JJS	max(), min(), limit	Extracted from POS Tags, matching the superlative adjective
SDB	WHERE	Match person name, place name, organization name, etc.
UDB	WHERE	Match capitalized words.
DATE,NUM,YEAR	WHERE	Matching: words in time format, numbers, years
NOT	!=, except, not in	Matching words: not, no, n't, never, without, etc.
IN, at, as	WHERE	Matching: preposition, subordinating conjunction, and words at and as.
each	GROUP BY	Exact matching the word each.
#	NONE	Words that cannot match the above labels.

Table 1: Exact Match Grammar Linking labels. SQL keywords indicate which keyword the label is extracted from, but it does not mean that the corresponding keyword must appear in the target SQL when the label appears.

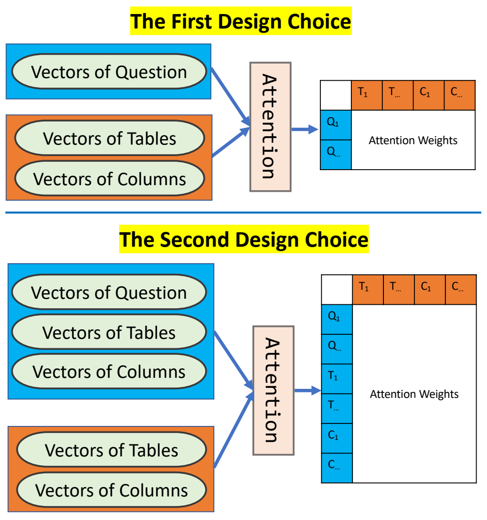
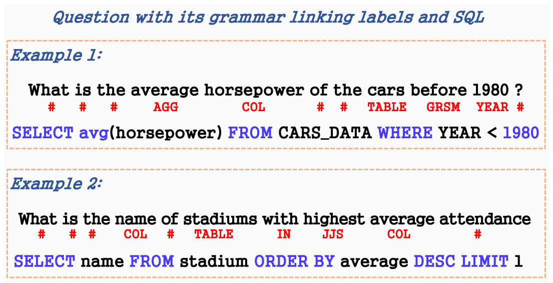


Figure 3: Two text-to-SQL schema linking design choices.



Question with its grammar linking labels and SQL

Example 1:

```
What is the average horsepower of the cars before 1980 ?
 #   #   #    AGG         COL       #  #  TABLE  GRSM  YEAR #
SELECT avg(horsepower) FROM CARS_DATA WHERE YEAR < 1980
```

Example 2:

```
What is the name of stadiums with highest average attendance
 #   #  #   COL  #  TABLE    IN   JJS     COL          #
SELECT name FROM stadium ORDER BY average DESC LIMIT 1
```

Figure 4: Two text-to-SQL examples with generated grammar linking labels.

the common factor is the calculation of a similarity score between each question word and schema item: correct schema items should obtain higher similarity scores.

Schema linking modules output attention scores from computing the schema linking feature and word embeddings. There are currently two attention mechanism designs. The first calculates a score that relates the question on one side, to the schema items on the other. The second approach also considers the attention scores between different schema items (Guo et al., 2019), thus it takes the question and schema items together as input to produce the attention scores. In Figure 3, we elucidate the inputs and outputs associated with the two types of attention computations.

Both designs have their own rationale, but the first design is more interpretable: it requires the model to infer the correct schema item from the question, so you can see which part of the question is related to each selected schema item. With the second design, it is sometimes difficult to explain why the specific schema item is chosen due to the presence of some other schema items. For ease of understanding, we show and discuss the attention and interpretability of an example under different design choices in Appendix B; we test the effects on performance in Section 5.4.

4 Proposal III: Exact Match Grammar Linking (GL)

The mechanisms of existing models limit their performance in some specific situations discussed in Section 4.2 and Appendix C. Our third proposal is GL which can provide ideas for addressing these limitations.

4.1 Overview

We propose GL, inspired by the EMSL and the NL question label generation method (Ma et al., 2020). EMSL attempts to match the words in the question with the schema item words, with the matching result then used as a feature to help the model generate the correct schema items. Similarly, GL matches the words in the question with those related to the SQL keywords. Considering that the number of SQL keywords is limited, GL can be implemented by matching words to specific categories/labels.

In general, GL has two main steps in the text-to-SQL process: (1) identify the label of each word, as shown in Figure 4; (2) input both label and word embeddings into the encoder. Formally, given a natural language question $Q = q_1..q_Q$, we generate a label l_i to each q_i, as shown in Figure 4. There are different embeddings for l_i and q_i, where the l_i embeddings Emb_l is randomly initialized while the q_i embeddings Emb_q can obtain from GLOVE (Pennington et al., 2014) or PLMs. The encoder input X is the sum of embeddings of q_i and l_i:

$$X = (Emb_l(l_0) + Emb_q(q_0), ..., Emb_l(l_Q) + Emb_q(q_Q))$$

Specifically, $Emb_l(l_i)$ does not enforce binding to specific SQL keywords. We expect the model to learn the meaning of GL label embeddings.

Before we can use GL for text-to-SQL, we need to define the labels and match rules. We use the method of information extraction combined with manual annotation to collect the words related to SQL keywords. We remove the words that match the schema items and then use the tf-idf (term frequency-inverse document frequency) score to extract the words that are highly relevant to the target SQL keyword. After manual correction and adding words of the same type, such as synonyms, we obtain the word categories/labels corresponding to the SQL keywords. Table 1 presents all labels with their matching rules used by this work.

In Table 1, uppercase labels match a class of words, while lowercase labels only match words with the same name as the label, such as 'at' and 'each' labels. The order of the labels in Table 1 is sorted by their priority. We start matching from the highest priority label and end once the match is successful. Taking Figure 4 as an example, the label for the word 'average' in example 1 is 'AGG', but that in example 2 is 'COL'. Because the two examples belong to different domains where the database of the second example contains a column named 'average' that can be built an EMSL with the question word 'average'. Therefore, although the 'average' in example 2 satisfies both the matching rules of the 'AGG' and 'COL' labels, its generated label is 'COL' instead of 'AGG' since the priority in the 'COL' label is higher.

4.2 Benefits from GL

Existing models (Bogin et al., 2019; Guo et al., 2019; Wang et al., 2020; Cao et al., 2021) tend to output the incorrect SQL clause 'ORDER BY avg(average)' instead of 'ORDER BY average' for the second example in Figure 4, even though their generation process is based on the same decoder. This error states that the 'average' word is used twice in generating the 'average' column and the 'avg' function. Utilizing GL can help existing models solve this problem. When training with GL, the model will learn that most of the examples generating 'avg' function require the 'AGG' label of GL. Therefore, since there is no 'AGG' label in example 2, the model will not tend to generate the 'avg' function. For other benefits from GL, please refer to Appendix C.

5 Experiments

5.1 Generating Typos

To evaluate robustness against typos, we randomly insert a letter into the correct schema annotation word. (This is enough to break EMSL, so we do not also modify the question words). We generated three typo development sets, named Spider-T1 to Spider-T3. The typos in Spider-T1 are generated by randomly inserting a letter at any position except the end. In contrast, Spider-T2 appends a random letter at the end of the schema annotation words. We examine these separately: the BERT tokenizer may be able to split Spider-T2 typos into a correct word and a suffix, but is less likely to split the Spider-T1 typos well. We convert every schema annotation word in Spider-T1 and T2 to typos when word length is greater than five letters; typos are generally more likely to occur in longer words, and words with more than five letters account for about 40% of the dataset. Spider-T3 is then the same as Spider-T1, but only converts the most frequent schema item words to typos. While Spider-T1 and T2 simulate the impact of large numbers of typos in extreme cases, Spider-T3 evaluates the impact of a more realistic, smaller number of typos. Other typos are possible, e.g. by deleting and swapping letters; we discuss these in Appendix D.

5.2 Experimental Setup

We evaluate the previous state-of-the-art models on Spider (Yu et al., 2018b), Spider-T, and Spider-Syn (Gan et al., 2021a) datasets. All results report their maximum value. All experiments were performed on a machine with an Intel i5 9600 3.1GHz processor and a 24GB RTX3090 GPU. Since the Spider test set is not publicly accessible and Spider-Syn and Spider-T do not contain test sets, our evaluation is based on the development sets. The Spider-Syn benchmark contains three development sets: Spider-Syn, ADV_{BERT}, and $\text{ADV}_{\text{GLOVE}}$, for evaluating model robustness against synonym substitution. Therefore, we have the following evaluation sets:

- **Spider**: The original Spider development set with 1,034 examples.
- **Spider-T1, T2 and T3**: Three development sets that replace the correct word with typos, introduced in Section 5.1.
- **Spider-Syn**: The human-curated development set built upon Spider, for evaluating synonym substitution in real-world question paraphrases.
- **ADV_{BERT}:** The set of adversarial examples generated by BERT-Attack (Li et al., 2020).
- **$\text{ADV}_{\text{GLOVE}}$:** The set of adversarial examples generated using the nearest GLOVE word vector (Pennington et al., 2014; Mrkšić et al., 2016).

Our evaluation is based on the exact match metric defined in the original Spider benchmark. This metric measures whether the syntax tree of the predicted query without condition values is the same as that of the gold query. Our experiment setting is consistent with the ablation study in Appendix E. Following the encoder design in Section 2.2, we evaluate different variants of the RATSQL model:

- **RATSQL**: The base RATSQL+GLOVE model trained on Spider using EMSL in training and evaluation (Wang et al., 2020).
- **RATSQL_O**: Our modified RATSQL+GLOVE model trained on Spider using EMSL in training and evaluation, discussed in Section 2.2.
- **RATSQL_B**: The RATSQL+BERT model trained on Spider using EMSL in training and evaluation. (Note that RATSQL_O+BERT is just RATSQL+BERT: using BERT means that the BERT encoder will replace all encoders in Figure 2).
- **RATSQL_BS**: RATSQL+BERT trained on **Spider-Syn** using EMSL (Gan et al., 2021a).
- **RATSQL_G**: RATSQL+GAP trained on Spider using EMSL (Shi et al., 2021).

Model	Spider
RATSQL	62.7%
RATSQL **w/o** EMSL	51.9%
RATSQL_O	62.2%
RATSQL_O **w/o** EMSL	58.4%

Table 2: Accuracy of two RATSQL ablations on the development set.

- **w/o EMSL:** Models without EMSL in training and evaluation, consistent with Tables 9 and 10.
- **ManualMAS** (Gan et al., 2021a): Schema annotations include synonyms used in Spider-Syn.
- **AutoMAS** (Gan et al., 2021a): Schema annotations include synonyms generated according to the nearest GLOVE word vector.

5.3 Experiment on EMSL (Proposal I)

5.3.1 Evaluation on Spider

Table 2 presents the exact matching accuracy of models trained on the Spider training set. Without EMSL, our RATSQL_O model significantly improves over RATSQL. Models with PLMs also obtained similar results with RATSQL_O, which also supported our proposal to remove the EMSL. Detailed experimental results and discussion are shown in Appendix E. Furthermore, we conduct an error analysis in Appendix F.

5.3.2 Robustness Evaluation

Typo Results Table 3 presents the robustness evaluation results on several datasets. For typos, GLOVE will treat them as UNK words, so the RATSQL and RATSQL_O cannot obtain good performance on Spider-T1 and T2 due to too many UNK words. The RATSQL_O without EMSL significantly outperforms the RATSQL without EMSL in Spider-T3, which is another evidence that the RATSQL_O is better in handling UNK words. After using PLMs, the performance on typos has been significantly improved, especially on Spider-T2. Spider-T3 contains only a few typos, i.e., it is close to the Spider to some extent. Thus, the T3 result characteristics are close to Spider, i.e., their performance gap between with and without EMSL is close. With the increase of typos, the performance gap will be expanded, where the model+PLM without EMSL will be better.

Synonym Substitution Results Gan et al. (2021a) propose three development sets for evaluating the robustness of text-to-SQL models

Approach	**Spider**	**Spider-T1**	**Spider-T2**	**Spider-T3**	**Spider-Syn**	**ADV$_{\text{GLOVE}}$**	**ADV$_{\text{BERT}}$**
RATSQL	**62.7%**	**23.9%**	**26.4%**	**51.2%**	33.9%	30.9%	37.1%
RATSQL **w/o** EMSL	51.9%	20.8%	21.7%	44.1%	**39.1%**	**38.1%**	**40.9%**
RATSQL$_O$	**62.2%**	**22.8%**	**25.7%**	**51.6%**	32.1%	32.7%	36.3%
RATSQL$_O$ **w/o** EMSL	58.4%	20.8%	23.3%	51.5%	**42.6%**	**38.6%**	**43.8%**
RATSQL$_B$	**69.7%**	30.9%	54.8%	**63.2%**	48.2%	38.0%	48.8%
RATSQL$_B$ **w/o** EMSL	69.3%	**32.3%**	**66.2%**	63.0%	**52.7%**	**45.4%**	**54.3%**
RATSQL$_{BS}$	68.1%	33.6%	58.1%	62.7%	58.0%	47.7%	55.7%
RATSQL$_{BS}$ **w/o** EMSL	**69.7%**	**38.1%**	**66.4%**	**65.0%**	**60.4%**	**51.0%**	**58.8%**
RATSQL$_G$	**71.8%**	48.1%	64.6%	68.0%	54.6%	46.6%	54.8%
RATSQL$_G$ **w/o** EMSL	71.7%	**53.4%**	**67.6%**	**68.6%**	**58.7%**	**49.4%**	**57.3%**
S2SQL (Hui et al., 2022)	76.4%	-	-	-	51.4%	-	-

Table 3: Exact match accuracy on original (Spider), typos (Spider-T1 to T3), and synonym substitution (Spider-Syn, ADV$_{\text{GLOVE}}$, and ADV$_{\text{BERT}}$) development sets. S2SQL results are quoted from Hui et al. (2022).

Approach	**Spider**	**Spider-Syn**	**ADV$_{\text{GLOVE}}$**	**ADV$_{\text{BERT}}$**
RATSQL$_B$ + ManualMAS	67.4%	**62.6%**	34.2%	44.5%
RATSQL$_B$ + ManualMAS **w/o** EMSL	**68.6%**	58.9%	**43.6%**	**53.1%**
RATSQL$_B$ + AutoMAS	68.7%	**56.0%**	61.2%	52.5%
RATSQL$_B$ + AutoMAS **w/o** EMSL	**68.9%**	55.3%	**62.1%**	**54.7%**
RATSQL$_{BS}$ + ManualMAS	65.6%	59.5%	46.9%	51.7%
RATSQL$_{BS}$ + ManualMAS **w/o** EMSL	**68.7%**	**61.7%**	**50.3%**	**58.8%**
RATSQL$_{BS}$ + AutoMAS	66.8%	57.5%	61.0%	55.7%
RATSQL$_{BS}$ + AutoMAS **w/o** EMSL	**69.2%**	**59.4%**	**63.2%**	**59.0%**

Table 4: Evaluation on the combination of MAS with RATSQL$_B$ and RATSQL$_{BS}$ respectively.

against synonym substitution, including: Spider-Syn, ADV$_{\text{BERT}}$, and ADV$_{\text{GLOVE}}$. Table 3 shows that models without EMSL consistently outperform those with EMSL when evaluated against Spider-Syn, ADV$_{\text{GLOVE}}$ and ADV$_{\text{BERT}}$. When using PLMs, RATSQL$_B$ and RATSQL$_G$ without EMSL show a huge performance improvement on these three development sets with only a tiny performance loss on Spider. RATSQL$_O$ without EMSL consistently outperforms RATSQL without EMSL, which means a reasonable design can reduce reliance on EMSL. Unlike other models, the RATSQL$_{BS}$ without EMSL outperforms that with EMSL in all evaluation sets. We discuss this in Appendix A. Although the S2SQL (Hui et al., 2022) model achieves pretty good performance in the Spider, its EMSL module causes its performance on Spider-Syn to be much worse than that of the RATSQL$_G$ without EMSL.

MAS Results Gan et al. (2021a) also propose a MAS method to improve the robustness of text-to-SQL models. MAS provides multiple annotations to repair the breaking of EMSL due to synonym substitutions. Although we advocate not relying on EMSL, MAS can still improve the performance of models without EMSL, as shown in Table 4. Comparing the data in Table 3 and Table 4, ManualMAS improves the performance of RATSQL$_B$ and RATSQL$_{BS}$ with and without EMSL on Spider-

Model	Spider
RATSQL$_G^1$	70.2%
RATSQL$_G^2$	71.8%
RATSQL$_G^2$ **with** NatSQL (Gan et al., 2021b)	73.7%
LGESQL + ELECTRA (Cao et al., 2021)	75.1%
RATSQL$_G^1$ **with** NatSQL	75.5%
RATSQL$_G^1$ **with** NatSQL and GL	**76.4%**
T5 Rerankers + PICARD (Zeng et al., 2022)	**76.4%**
S2SQL (Hui et al., 2022)	**76.4%**

Table 5: Exact match accuracy on Spider development set. The superscript number of RATSQL$_G$ indicates the design choice introduced in Section 3 (Proposal II). The results are compared with the top two published models on the Spider leaderboard.

Syn development set since the ManualMAS provide synonym annotations appearing in the Spider-Syn. In the same way, AutoMAS has also improved their performance on ADV$_{\text{GLOVE}}$. Experimental results show that although MAS is designed to repair EMSL, it is still effective for models without EMSL. Besides, based on MAS, the overall performance of the model without EMSL is still better than that with EMSL. In general, even though EMSL is not used, a reasonable annotation is still essential to the text-to-SQL problem.

5.3.3 Discussion

We introduce RATSQL$_O$, an enhanced text-to-SQL encoder design, challenging the need for Exact Match Schema Linking (EMSL) assumed by previous research. RATSQL$_O$ offers an alternative perspective, arguing that the benefit of consolidating the encoder actually promotes schema linking. As a result, relying solely on PLM can only address certain issues. For instance, PLMs have input limitations that may suffice for current text-to-SQL benchmarks with small-scale schemas. However, for large-scale schemas, modifications to the PLM

Question:	What is the name and capacity for the stadium with the highest average attendance ?	
RATSQL$_{GN}$ with GL:	SELECT name , capacity FROM stadium ORDER BY average DESC LIMIT 1	✓
S²SQL+ELECTRA:	SELECT name , capacity FROM stadium GROUP BY average ORDER BY avg(average) DESC LIMIT 1	X
LGESQL+ELECTRA:	SELECT name , capacity FROM stadium GROUP BY Highest ORDER BY avg(average) DESC LIMIT 1	X
RATSQL$_{GN}$ w/o GL:	SELECT name , capacity FROM stadium GROUP BY stadium_id ORDER BY avg(average) DESC LIMIT 1	X

Question:	Give the names of countries with English and French as official languages .	

RATSQL$_{GN}$ with GL:	SELECT Name FROM ... WHERE Language = "English" AND IsOfficial = "T" INTERSECT
	SELECT Name FROM ... WHERE Language = "French" AND IsOfficial = "T"
		✓

S²SQL+ELECTRA:	SELECT Name FROM ... WHERE Language = "value" INTERSECT
	SELECT Name FROM ... WHERE Language = "value"
		X

LGESQL+ELECTRA:	SELECT Name FROM ... WHERE Language = "value" INTERSECT
	SELECT Name FROM ... WHERE Language = "value"
		X

RATSQL$_{GN}$ w/o GL:	SELECT Name FROM ... WHERE Language = "English" INTERSECT
	SELECT Name FROM ... WHERE Language = "French"
		X

Table 6: Two examples related to the discussion in Section 4.2 and Appendix C. RATSQL$_{GN}$ indicate the RATSQL$_G$+NatSQL model. We selected the current top 2 opensource models in the exact match metric of Spider leaderboard to compare with RATSQL$_{GN}$.

encoding method are necessary. If the PLM were to encode the question and schema separately, the EMSL would still be required.

5.4 Experiment on Schema Linking Module Design Choices (Proposal II)

As discussed in Section 3, there are two design choices for the schema linking module. The first calculates a score that relates the question on one side, to the schema items on the other. The second approach assumes the attention between one schema item and others is needed, and it therefore takes question and schema items together as input to produce the score. The original RATSQL chose the 2nd design, named it RATSQL^2_G here. We modify the RATSQL according to the 1st design, named it RATSQL^1_G, and observe that its performance drops slightly, as shown in Table 5. Error analysis shows that RATSQL^1_G tends to use the schema items mentioned in the question and is not so good at dealing with implicit schema items.

Although the performance of the RATSQL^1_G is slightly worse, we found that its schema linking performance is not inferior. The accuracy of the *SELECT* clause is the best way to measure the schema linking performance because every SQL contains at least one *SELECT* clause that only contains schema items. The *SELECT* accuracy of the RATSQL^1_G is slightly (0.4%) better than the RATSQL^2_G, which inspired us that the RATSQL^1_G is likely to perform well if removing the implicit schema items. Fortunately, we found NatSQL, an SQL intermediate representation that removes many implicit schema items from the SQL (Gan et al., 2021b). Experiments show that the performance of RATSQL^1_G+NatSQL is better than the RATSQL^2_G+NatSQL. Table 5 gives a detailed performance comparison, from which it can be found that by replacing the design, the RATSQL$_G$+NatSQL is improved to the second place, evaluated on the development set, which is close to the current best published model. It should be noted that RATSQL$_G$+NatSQL does not use the complex graph neural network as S2SQL and LGESQL, nor does it use the ELECTRA, which is shown to be better than GAP (Clark et al., 2020; Cao et al., 2021; Hui et al., 2022).

5.5 Experiment on GL (Proposal III)

We assemble GL onto RATSQL$_G$+NatSQL, obtaining 0.9% absolute improvement, rising from 75.5% to 76.4%, which improves the performance of the RATSQL model to be consistent with the best-published model, as shown in Table 5. Although the performance improvement of GL is less apparent than that of the EMSL in previous ablation studies, GL is an indispensable module for solving specific problems. We compared the RATSQL$_G$+NatSQL+GL with S²SQL and LGESQL, where S²SQL and LGESQL are the current top 2 opensource models in the exact match metric of the Spider leaderboard. Table 6 presents the output of these models for two examples in the Spider development set. No models without GL, including RATSQL$_G$+NatSQL, generate the correct SQL. We discuss how GL can help the model generate correct SQL in Section 4.2 and Appendix C, respectively.

During our experiment, we faced a number of challenges with using the GL. Unlike schema linking, which creates a direct link from question tokens to a schema item, GL doesn't establish a similar connection. In GL, it just identifies question tokens related to SQL keywords based on rules defined in Table 1. GL requires the neural model to find out which SQL keywords it's connected to by training. However, we found this difficult to accomplish when in scenarios involving complex SQL queries, the model struggled to accurately connect GL with the appropriate SQL keywords. To address this, we opted for using NatSQL from Spider-SS (Gan et al., 2022) in our (RATSQL^1_G with NatSQL and GL) model, rather than the original version. The benefit of Spider-SS is that it breaks down SQL/NatSQL according to question clauses. This means the model rarely comes across complex SQL/NatSQL during training, simplifying the task of associating GL with the proper SQL keywords. To further increase the chances of success with GL, we also made slight adjustments to NatSQL, which you can find in Appendix C. We ran ablation studies to check if NatSQL modification or the use of Spider-SS data impacted the results. Experiments showed that without GL, the outcomes using Spider-SS data remained consistent with those using Spider. Similarly, the performance of the adjusted NatSQL, when used without GL, is also consistent with the original NatSQL.

6 Related Work

Schema Linking Schema linking has been an important design choice for existing text-to-SQL models (Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020; Chen et al., 2020; Cao et al., 2021). Besides designing new models, some works focus on investigating the effect of schema linking. Lei et al. (2020) demonstrate that more accurate schema linking conclusively leads to better text-to-SQL parsing performance. To support further schema linking studies, Lei et al. (2020) and Taniguchi et al. (2021) invest human resources to annotate schema linking corpus, respectively. Guo et al. (2019) and Wang et al. (2020) conducted an ablation study on EMSL, respectively, and the results show that removing the EMSL would lead to the greatest decrease in model performance. These studies have influenced many follow-up works to use EMSL (Cai et al., 2021; Xu et al., 2021; Lei et al., 2020; Yu et al., 2021; Shi et al., 2021). Moreover, in the ablation study of BRIDGE (Lin et al., 2020), its performance declines notably when the PLM is removed, as it does not utilize EMSL. Our work found that existing text-to-SQL models with EMSL tend to overly rely on this schema linking module, which degrade their robustness. Meanwhile, more advanced pretrained language models can replace EMSL and thus improve the model robustness, without notably degrading the performance.

Robustness of Text-to-SQL Existing works on improving the robustness of the text-to-SQL model are mainly through adversarial training, data augmentation, and repairing EMSL. Xiong and Sun (2019) and Radhakrishnan et al. (2020) propose data augmentation techniques for improving the generalization in cross-domain text-to-SQL and in search-style questions resepctivly. However, these approaches only support SQL queries executed on a single table, e.g., WikiSQL. Zeng et al. (2020) introduce a Spider$_{\text{UTran}}$ dataset that includes original Spider (Yu et al., 2018b) examples and some untranslatable questions examples. Spider$_{\text{UTran}}$ can be used to evaluate whether the text-to-SQL model can distinguish the untranslatable NL question. Gan et al. (2021a) investigate the robustness against synonym substitution for cross-domain text-to-SQL translation and found that synonym substitution would break EMSL, giving a significant drop in performance; to solve this problem, they proposed the MAS method to repair the broken EMSL. Following (Gan et al., 2021a), our work found that the EMSL can be replaced by better encoding, and models without EMSL has better generalization ability.

7 Conclusion

In this work, we investigate the role of schema linking for text-to-SQL models regarding model performance and robustness. In particular, we demonstrate that by leveraging pretrained language models, EMSL is no longer a necessary building block to ensure a high performance on text-to-SQL benchmarks. We observe that when EMSL is used, models become overly reliant on it, making them vulnerable to attacks that break the exact-match assumptions of EMSL. Beyond this, by studying different schema linking module designs we find that a direct relation between the question and schema items is more interpretable and works well with intermediate representation SQL. Finally, inspired by EMSL, we introduce Exact Match Grammar Linking for dealing with some examples where existing models can easily make mistakes.

8 Limitation

We notice that there are some works based on pretrained large language models has obtained good performance (Scholak et al., 2021; Li et al., 2023). Due to limited computing resources, our evaluation mainly focuses on model architectures specially designed for text-to-SQL problems, and we did not conduct experiments on recent pretrained large language models, such as T5 (Raffel et al., 2020) and GPT3 (Brown et al., 2020), due to limited computing resources. However, note that most models with top text-to-SQL performance still employ specialized architecture design, e.g., with EMSL. We consider extending our study to recent pretrained large language models as future work.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Matthew Purver acknowledges financial support from the UK EPSRC under grant EP/S033564/1, and from the Slovenian Research Agency for research core funding (No. P2-0103 and No. P5-0161). Yujian Gan is supported by the ARCIDUCA project, funded by EPSRC (EP/W001632/1).

References

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

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

Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql.

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. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2541–2555, Online. Association for Computational Linguistics.

Sanxing Chen, Aidan San, Xiaodong Liu, and Yangfeng Ji. 2020. A tale of two linkings: Dynamically gating between schema linking and structural linking for text-to-SQL parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2900–2912, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

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.

Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, and Matthew Purver. 2022. Measuring and improving compositional generalization in text-to-sql via component alignment.

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

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.

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.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^{2}SQL: 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.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the Role of Schema Linking in Text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention extraction and linking for SQL query generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6936–6942, Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Karthik Radhakrishnan, Arvind Srikantan, and Xi Victoria Lin. 2020. ColloQL: Robust Cross-Domain Text-to-SQL Over Search Queries.

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. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13806–13814.

Robyn Speer and Catherine Havasi. 2012. Representing General Relational Knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).

Sinan Tan, Mengmeng Ge, Di Guo, Huaping Liu, and Fuchun Sun. 2021. Knowledge-based embodied question answering.

Yasufumi Taniguchi, Hiroki Nakayama, Kubo Takahiro, and Jun Suzuki. 2021. An investigation between schema linking and text-to-sql performance.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Hongvu Xiong and Ruixiao Sun. 2019. Transferable Natural Language Interface to Structured Queries Aided by Adversarial Generation. In *2019 IEEE 13th*

International Conference on Semantic Computing (ICSC), pages 255–262. IEEE.

Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J.D. Prince, and Yanshuai Cao. 2021. Optimizing deeper transformers on small datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2089–2102, Online. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing.

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

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

Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. 2020. Photon: A Robust Cross-Domain Text-to-SQL System. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 204–214, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lu Zeng, Sree Hari Krishnan Parthasarathi, and Dilek Hakkani-Tur. 2022. N-best hypotheses reranking for text-to-sql systems. *arXiv preprint arXiv:2210.10668*.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. pages 5338–5349.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*, abs/1709.0.

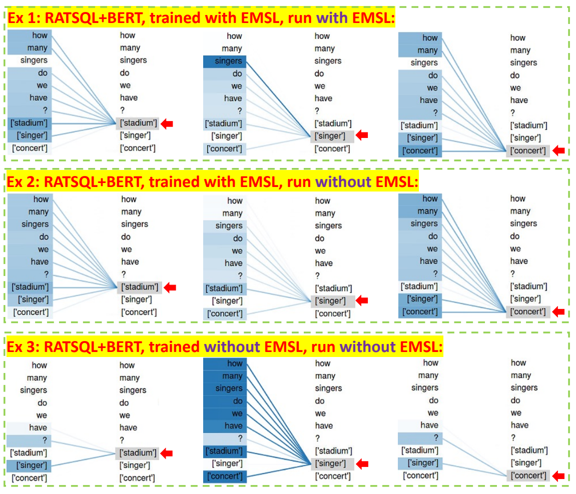

Ex 1: RATSQL+BERT, trained with EMSL, run with EMSL:
Ex 2: RATSQL+BERT, trained with EMSL, run without EMSL:
Ex 3: RATSQL+BERT, trained without EMSL, run without EMSL:
how
many
singers
do
we
have
?
['stadium']
['singer']
['concert']

Figure 5: Examples of the question-table attention. The darker the color, the greater the attention score. The first two examples are extracted from RATSQL$_B$, while the last one is from RATSQL$_B$ without EMSL. Each attention subgraph represents the attention between only one table schema and other words.

A Further Discussion on EMSL

The text-to-SQL model can quickly locate the correct schema items through EMSL, but this advantage will cause the models to not work properly when EMSL fails. To better understand the impact of EMSL on text-to-SQL models, we present the question-table attention [3] extracted from RATSQL$_B$ with and without EMSL in Figure 5. In the first example, we can see that the alignment score between table *singer* and question word *singer* is the biggest, while we can not observe a clear connection between other tables and question word *singer*. However, when removing the EMSL in the second example, the alignment score between table *singer* and question word *singer* drop clearly, and the connection between other tables and question word *singer* becomes clear. It can be seen that under other conditions unchanged, only removing EMSL has a considerable impact on the model trained with EMSL.

The third example is extracted from RATSQL$_B$ without EMSL. Different from the RATSQL$_B$ with EMSL, the *singer* table has a high alignment score not only with the word *singer* but also with the whole sentence. Since the loss function only calculates whether the output schema items are correct, the model does not care which question word

[3] It is named *m2t_align_mat* in the code: https://github.com/microsoft/rat-sql/blob/master/ratsql/models/spider/spider_enc_modules.py

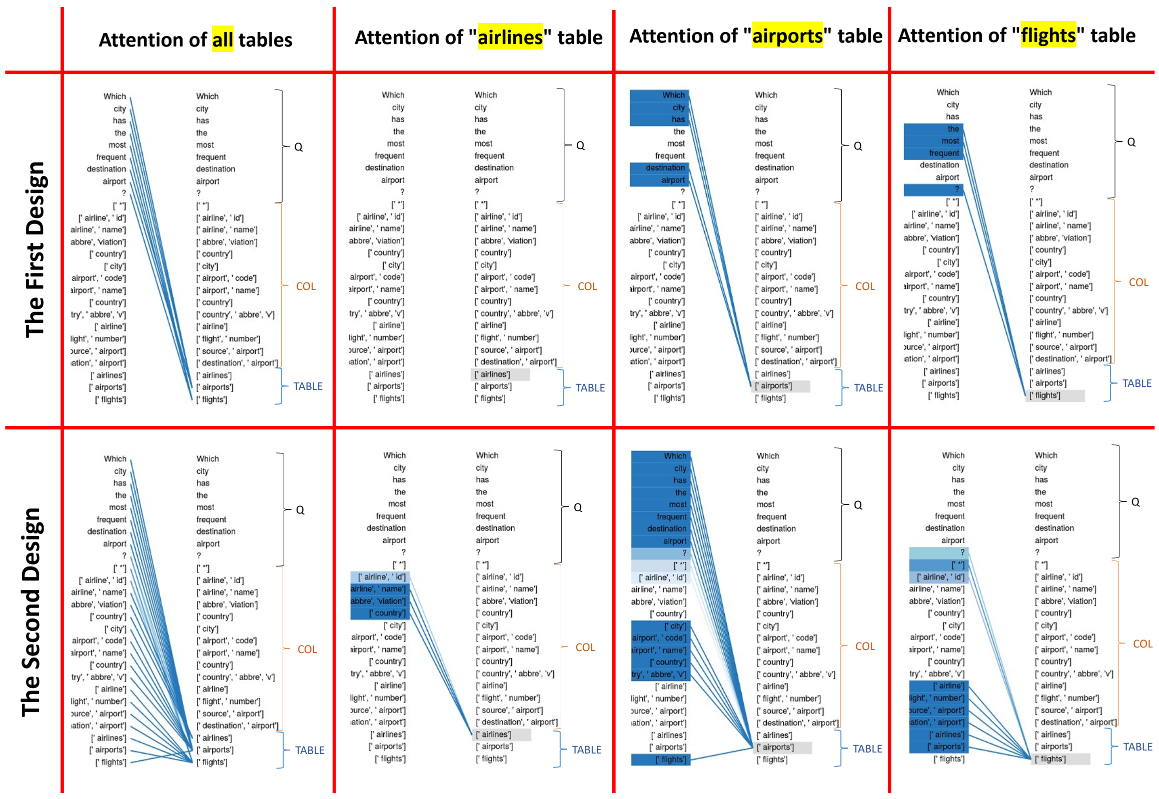
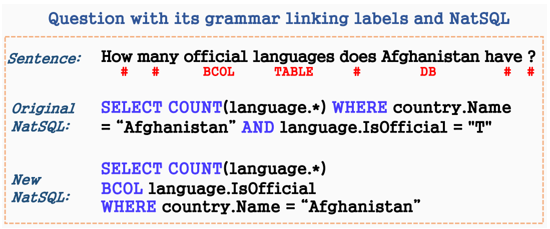


Figure 6: The attention weight of the schema table under different schema linking module design choices. The darker the color, the greater the attention score.

the correct schema item is linked to. Therefore, the attention of the RATSQL$_B$ without EMSL is quite different from that with EMSL. The significant difference of the trained models may be one of the reasons why the overall performance of RATSQL$_{BS}$ without EMSL is better than that with EMSL. Because the training data in RATSQL$_{BS}$ contain many synonym substitution examples, and these examples do not have EMSL features, it requires the model to find a balance between states shown in examples 1 and 3 of Figure 5, which increases the difficulty of training.

B Attention Visualization of Different Schema linking Module Design Choices

Figure 6 presents the attention weight of schema tables and illustrates why the first design choice is more interpretable. The SQL for the question in Figure 6 is '*SELECT* T1.City *FROM* Airports *AS* T1 *JOIN* Flights *AS* T2 *ON* T1.AirportCode = T2.DestAirport *GROUP BY* T1.City *ORDER BY* count(*) *DESC LIMIT* 1'. So, the table 'airports' and 'flights' are needed. Although models under both design choices predict this example correctly, their attention scores are quite different. We observe that the attention under the first design can locate the proper question words. However, the attention of the table 'flights' can not locate any question words when using the second design choice, which is difficult to explain why the 'flights' table was selected instead of the 'airlines' table with similar attention. It should be noted that the 'flights' table is mentioned implicitly, but it does not prevent the first design choice from giving it the proper attention.



Question with its grammar linking labels and NatSQL

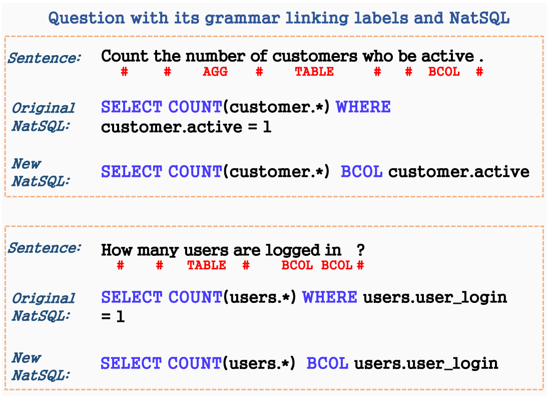
```
Sentence:    How many official languages does Afghanistan have ?
              #    #      BCOL      TABLE      #        DB       # #

Original     SELECT COUNT(language.*) WHERE country.Name
NatSQL:      = "Afghanistan" AND language.IsOfficial = "T"

             SELECT COUNT(language.*)
New          BCOL language.IsOfficial
NatSQL:      WHERE country.Name = "Afghanistan"
```

Figure 7: An text-to-SQL examples with grammar linking labels and newly designed NatSQL.

C More Benefits from GL

It is difficult for existing models (Bogin et al., 2019; Guo et al., 2019; Wang et al., 2020; Cao et al., 2021) to generate the 'WHERE isOfficial = 'T' ' clause in Figure 7 since there are no similar examples in the training data. This example requires the model to generate the WHERE condition based on a single NL word ‘official’. The word ‘official’ partly exactly matches the column ‘isOfficial’ and implicitly mentions the WHERE condition value. The implicit WHERE condition value challenges existing models since most WHERE condition values are explicitly mentioned in the training data. It is difficult for models to understand this implicit expression. Hence, models tend to regard the word ‘official’ as an occasionally mentioned column, and the WHERE condition will not be generated. GL can solve this special problem by giving a new label BCOL to the word ‘official’, different from the common exact match schema linking labels: COL and TABLE. Figure 8 presents two more examples with the BCOL label and newly designed NatSQL.

We found that the type of column in these special WHERE conditions is Boolean-like. For example, the ‘isOfficial’ column only contain the ‘T’ and ‘F’ values, and some others Boolean-like column only contain the ‘0’ and ‘1’ values. Therefore, we first analyze the database data and mark the columns with Boolean-like type. When generating the GL labels, if the word matches a schema column marked as Boolean-like type, give it a ‘BCOL’ label instead of ‘COL’. However, the addition of ‘BCOL’ only cannot solve this problem, i.e., the model still does not generate the ‘isOfficial’ condition. We found that training data make models tend to give fewer WHERE conditions to simpler questions. To avoid conflicts with other simple questions, we move the WHERE condition of Boolean-like type to a BCOL clause, as shown in the new NatSQL of Figure 7 The BCOL clause in this newly designed NatSQL is finally converted to a WHERE condition. At this point, with the BCOL label and clause, our method can generate correct SQL for questions similar to Figure 7.



```
Question with its grammar linking labels and NatSQL

Sentence:     Count the number of customers who be active .
                #     #    AGG    #    TABLE     #   #  BCOL  #

Original      SELECT COUNT(customer.*) WHERE
NatSQL:       customer.active = 1

New
NatSQL:       SELECT COUNT(customer.*)  BCOL customer.active

Sentence:     How many users are logged in   ?
               #    #   TABLE  #    BCOL BCOL #

Original      SELECT COUNT(users.*) WHERE users.user_login
NatSQL:       = 1

New
NatSQL:       SELECT COUNT(users.*)  BCOL users.user_login
```

Figure 8: Two text-to-SQL examples with BCOL labels and newly designed NatSQL.

Approach	Spider-T4	Spider-T5
RATSQL	29.0%	28.6%
RATSQL **w/o** EMSL	**32.8%**	**30.1%**
RATSQL$_O$	27.6%	26.5%
RATSQL$_O$ **w/o** EMSL	**34.5%**	**31.2%**
RATSQL$_B$	34.9%	32.6%
RATSQL$_B$ **w/o** EMSL	**38.8%**	**35.0%**
RATSQL$_{BS}$	35.6%	32.6%
RATSQL$_{BS}$ **w/o** EMSL	**40.3%**	**38.2%**
RATSQL$_G$	46.7%	46.8%
RATSQL$_G$ **w/o** EMSL	**50.6%**	**50.7%**

Table 7: Exact match accuracy on Spider-T4 and Spider-T5 development sets.

D More Typos

Besides generating typos by inserting a letter, we also generate typos by deleting a letter and swapping the letter position, named the generated development set Spider-T4 and Spider-T5, respectively. Like Spider-T1 and T2, here we only convert the words whose length is greater than five letters to typos. Table 7 presents the exact match accuracy on Spider-T4 and Spider-T5 development sets. Since PLM handles typos in Spider-T4 and T5 similar to Spider-T1, their evaluation results are also similar. Besides, we observe that the results of models using GLOVE in Spider-T4 are the best, followed by in T5, then in T2, and finally in T1. To understand this phenomenon, we found that although the number of generated typos is the same among these datasets, Spider-T1 has the most GLOVE UNK words, followed by T2, then T5, and T4 contains the least UNK words. It can be seen that in the case of fewer UNK words, the model+GLOVE can generate better encoding so that the model+GLOVE without EMSL surpasses that with EMSL in Spider-T4 and T5.

E Ablation Study on EMSL

Table 9 presents the ablation study results of three base models. The results of RATSQL here are different from that of (Wang et al., 2020) because Wang et al. (2020) remove the cell value linking first and then EMSL. According to the magnitude of the decline, our results are similar to theirs. According to (Wang et al., 2020; Guo et al., 2019),

	Number of errors			Number of example with errors		
Approach	**Multi words**	**Single word**	**UNK word**	**Multi words**	**Single word**	**UNK word**
RATSQL	118	57	13	112 (10.8%)	54 (5.2%)	12 (1.2%)
RATSQL **w/o** EMSL	178	107	33	170 (16.4%)	93 (9.0%)	30 (2.9%)
RATSQL$_O$	136	51	11	125 (12.1%)	50 (4.8%)	11 (1.1%)
RATSQL$_O$ **w/o** EMSL	152	63	15	141 (13.6%)	59 (5.7%)	14 (1.4%)
RATSQL$_B$	55	38	-	53 (5.1%)	37 (3.6%)	-
RATSQL$_B$ **w/o** EMSL	65	34	-	65 (6.3%)	34 (3.3%)	-

Table 8: Statistics of the types of error column predictions of different models evaluated on the Spider development set. The larger the number, the worse the performance.

Model	Exact Match Acc
GNN	47.6%
GNN **w/o** EMSL	24.9%
IRNet	48.5%
IRNet **w/o** EMSL	40.5%
RATSQL	62.7%
RATSQL **w/o** EMSL	51.9%

Table 9: Accuracy of three based models ablations on the development set. EMSL means schema linking feature based on the exact lexical match. The IRNet results are copied from the original paper (Guo et al., 2019), while others are conducted by ourselves.

Model	Exact Match Acc
GNN+BERT	49.3%
GNN+BERT **w/o** EMSL	47.1%
RATSQL+BERT	69.7%
RATSQL+BERT **w/o** EMSL	69.3%
RATSQL+GAP	71.8%
RATSQL+GAP **w/o** EMSL	71.7%

Table 10: Accuracy of three models with PLM ablations on the development set. The GAP (Shi et al., 2021) is a pretrained model based on RoBERTa (Liu et al., 2019)

they observe the biggest performance degradation by removing EMSL. Since then, EMSL has become a necessary module for most researchers to build text-to-SQL models.

We want to challenge this view and carry out the comparative experiment in Table 10. Comparing Table 9 and Table 10, it can be found that PLMs compensate for the function of EMSL, i.e., the performance in Table 10 is less degraded than that in Table 9 after removing EMSL.

From another perspective, BERT and its subsequent pretrained language model significantly improve the performance of models that do not use EMSL, which explains why some models can achieve higher performance improvements through BERT. For example, EditSQL (Zhang et al., 2019) does not use EMSL, while it obtains the highest performance improvement by extending BERT, as shown on the Spider leaderboard [4].

F Further Discussion on Section 5.3.1

F.1 BERT vs GLOVE

The base RATSQL uses GLOVE (Pennington et al., 2014) for word embedding. There are two main reasons why BERT (Devlin et al., 2019) is better than GLOVE at schema linking. The first reason is that BERT can better deal with out-of-vocabulary words. BERT converts these words into subwords, so BERT makes sure different word is represented by a unique vector. However, GLOVE cannot handle out of vocabulary words. Researchers generally replace them with a custom unknown (UNK) word vector. Suppose there are multiple words outside the GLOVE vocabulary in one schema. In that case, it is equivalent to multiple schema items being annotated as UNK, which will cause the model without EMSL to be unable to distinguish different schema items due to the same word vector.

The second reason is that GLOVE is not as good as BERT in the face of schema items containing multi-words. As opposed to static embeddings provided by GLOVE, BERT provides dynamic lexical representations generated by analyzing the context. Take the *bandmate id* column in the Spider dataset as an example. The cosine of the vectors for the two words *bandmate* and *id* in GLOVE is negative, which means if we sum these two vectors together to represent the *bandmate id* column, the sum vector will inevitably lose some information. The word vector output by BERT is calculated based on the context, so although adjacent words may be unrelated in word meaning, their word vectors will still be highly correlated. Figure 9, generated by the bertviz (Vig, 2019), presents the BERT head view

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

of attention patterns in the one transformer layer where the word *bandmate* clearly links to the word *id*.

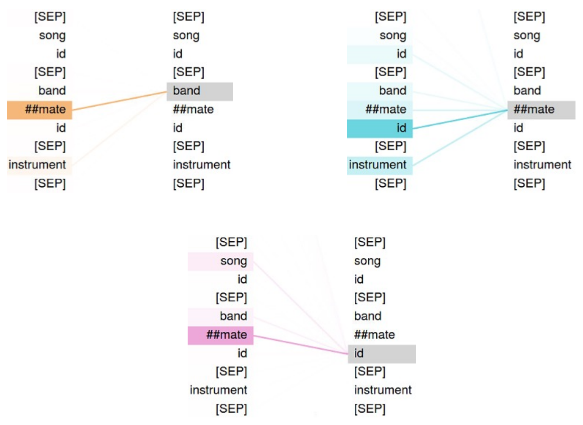


Figure 9: The BERT head view of attention patterns of word *bandmate* and *id* in the one transformer layer.

F.2 Error Analysis

Table 8 presents the error type statistics in the error column prediction. We count the prediction errors of single words, multiple words, and words outside the GLOVE vocabulary (UNK word) when the predicted SQL structure is correct. As BERT does not share GLOVE's vocabulary limitations, the UNK entry for RATSQL$_B$ is empty. Random initialization means that model results after each training may vary slightly, so we only focus on the more salient features.

Although the results of RATSQL and RATSQL$_O$ are similar, RATSQL$_O$ consistently outperforms RATSQL in three error types when EMSL is removed; this supports the view we discuss in Section 2.2. More importantly, the single-word performance of RATSQL$_O$ without EMSL is close to that of RATSQL and RATSQL$_O$. As discussed in Appendix F.1, the representation ability on multi-word of GLOVE is worse than that of BERT. The results support this view where the performance of RATSQL$_O$ and RATSQL on multi-word is worse than that on single-word. When replacing the GLOVE with BERT, due to the improvement of its multi-word representation ability, the performance of RATSQL$_B$ with and without EMSL are close in single and multiple words. From the right side of Table 8, it can also be found that the BERT brings around 5% absolute improvement on multi-word, while that on single-word is only 2%.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8

☑ A2. Did you discuss any potential risks of your work?
Section 8

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

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

B ☑ Did you use or create scientific artifacts?

Section 5

☑ B1. Did you cite the creators of artifacts you used?
Section 5

☐ 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)?
Section 5

☐ 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.?
Section 5

☑ 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.
Section 5

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5

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?
Section 5

☑ 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?
Section 5

☐ 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.)?
Not applicable. Left blank.

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

☐ 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.?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.