Pretrained Language Models Are All You Need For Text-to-SQL Schema Linking

Anonymous ACL submission

Abstract

The use of **E**xact **M**atch based **S**chema **L**inking (**EMSL**) has become standard in text-to-SQL: many state-of-the-art text-to-SQL models employ EMSL, and their performance drops significantly when the EMSL component is removed. In this work, however, we demonstrate 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 by utilizing the pre-trained language model as the encoder, we can improve the performance without using EMSL, and thus the model is more robust. Our experiments suggest that EMSL is not the icing on the cake, but it is the one that introduces the vulnerability, and it can be replaced by better input encoding.[1]

1 Introduction

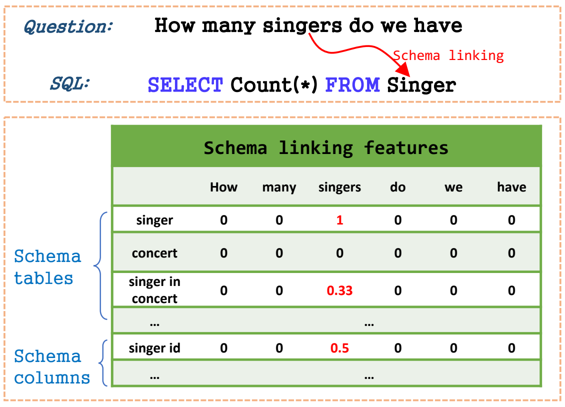
Recent years have seen great process on the text-to-SQL problem, i.e. translating a natural language question into a SQL query (Dong and Lapata, 2018; Yu et al., 2018b; Zhong et al., 2017; Gan et al., 2021; Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020), with neural networks having become the *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, and we call this process as *schema linking*. Previous work often explicitly designs a module to perform the schema linking, and we name it as Exact Match based Schema Linking (*E*MSL) (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 exact match based schema linking (EMSL) feature matrix. The method of obtaining schema linking features in previous work mainly relies on this exact lexical matching. 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; 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, 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 RAT-

[1]We will release code upon publication.


Question: How many singers do we have
Schema linking
SQL: SELECT Count(*) FROM Singer
Schema tables
Schema columns

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.

SQL (Wang et al., 2020), GNN (Bogin et al., 2019), and GAP (Shi et al., 2021).

We evaluate the models in three settings: the original Spider benchmark without input noise (Yu et al., 2018b), synonym substitution (Gan et al., 2021), and a new typo injection setting introduced in this work. Results show that the use of a pre-trained language model 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., 2021), a method designed to improve model robustness with EMSL, can also improve models without EMSL. In conclusion, we demonstrate that with pretrained language models, EMSL is no longer a necessary building block of text-to-SQL models.

2 Schema Linking

Following SQLNet (Xu et al., 2017), most text-to-SQL models generate the SQL structure first, and then fill in the schema items (Gan et al., 2020). Schema linking is needed in this workflow to locate the schema items from the question. Prior works show that models without schema linking perform poorly on text-to-SQL tasks, such as the sequence-to-sequence model (Yu et al., 2018b).[2]

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. This type of schema linking feature (EMSL) based on exact lexical matching is the most common (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). Some papers may not mention this exact matching explicitly, but it can be found in their published code. Implementation details vary; for example, some works add ConceptNet (Speer and Havasi, 2012) to get more linking features (Guo et al., 2019; Tan et al., 2021).

EMSL is often taken to be essential: ablation studies show that removing EMSL causes the biggest performance decline compared to removing other removable modules (Guo et al., 2019; Wang et al., 2020). 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., the EMSL is needed for textual matches. 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, which means a proper embedding can replace EMSL. We discuss details in Section 3.3.

2.2 Schema Linking Module

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. At present, the common method of this module is to calculate the similarity score between each question word and schema item. Although the implementation details of models are different, they all want the correct schema items to obtain higher similarity scores.

One difficulty in calculating the similarity scores is how to use a vector to represent a schema item that contains multiple words. For example, we need a proper vector to represent the *singer in concert* table in Figure 1, so that it has a higher score when calculating similarity with the words *singer* or *concert* in a question. If we cannot find such a vector, we need EMSL as the similarity score, e.g., use the 0.33 in Figure 1 to represent the similarity between *singer in concert* table and word *singer* in

[2]Note that prior works often use the phrase *schema linking* in different ways; it may refer to the schema linking feature or module or both, as discussed in Section 1.

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 1: 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 2: 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)

the question.

A well-designed model structure can reduce the need for EMSL. For example, even without EMSL, RATSQL and IRNet still outperform SyntaxSQLNet (Yu et al., 2018a) and SQLNet (Yu et al., 2018b; Xu et al., 2017). Appropriate auxiliary modules are also necessary for building schema linking. Graph neural networks make it easier to encode the schema structure and construct the correct schema linking (Wang et al., 2020; Bogin et al., 2019).

3 Case Study

In this section, we conduct an ablation study on EMSL using different models, including GNN (Bogin et al., 2019), IRNet (Guo et al., 2019), and RATSQL (Wang et al., 2020). We then conduct a more detailed examination using RATSQL, which is the most competitive model architecture.

3.1 Ablation Study on EMSL

Table 1 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), 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 2. Comparing Table 1 and Table 2, it can be found that PLMs compensate for the function of EMSL, i.e., the performance in Table 2 is less degraded than that in Table 1 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 [3].

3.2 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

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

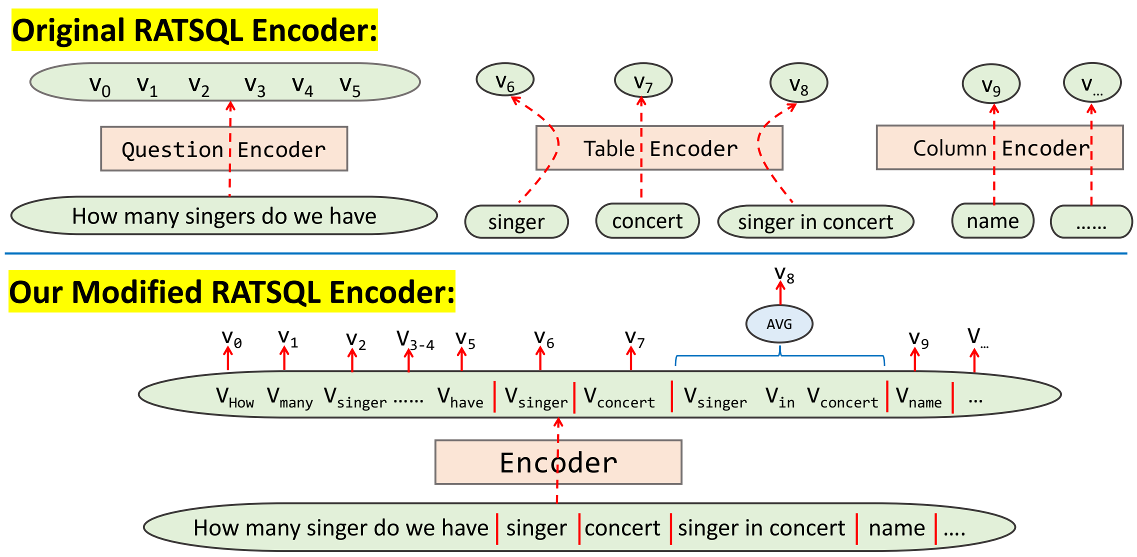

Original RATSQL Encoder:
v0 v1 v2 v3 v4 v5
Question Encoder
How many singers do we have
v6 v7 v8
Table Encoder
singer
concert
singer in concert
v9 v...
Column Encoder
name
......
Our Modified RATSQL Encoder:
v8
AVG
v0 v1 v2 v3-4 v5 v6 v7 v9 v...
vHow vmany vsinger ...... vhave vsinger vconcert vsinger vin vconcert vname ...
Encoder
How many singer do we have | singer | concert | singer in concert | name | ....

Figure 2: The original RATSQL encoder structure and our modified version.

context, so although adjacent words may be unrelated in word meaning, their word vectors will still be highly correlated. We provide more discussion in Appendix A.

3.3 RATSQL Encoder

The text-to-SQL encoder is part of the schema linking module. As discussed in Section 2.2, 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, RATSQL uses different encoders to encode the question and schema separately, as shown in the upper part of Figure 2. These three encoders are based on biLSTM and have 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. It does not matter when using EMSL for both training and evaluation because we can link the v_6 to v_2 through EMSL. However, when without EMSL, it requires the v_6 from the table encoder must close 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 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 the BERT encoder instead of the three encoders. 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 the 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. Besides, RATSQL$_O$ deal with words outside the GLOVE vocabulary better than RATSQL. Supposing that the word *concert* and *stadium* are outside the GLOVE vocabulary, the v_7 and v_9 output from RATSQL table encoder will be the same since their inputs are the same UNK vector. However, the RATSQL$_O$ encoder (BiLSTM) output different vectors for v_7 and v_9 because the contents before and after the word *concert* and *stadium* are different. In this way, even if there are multiple UNK words, the RATSQL$_O$ encoding vector will be different.

4 Experiment

4.1 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 B.

4.2 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., 2021) datasets. 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$_{\text{BERT}}$, and 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 typo development sets with 1,034 examples respectively, discussed in Section 4.1.
- **Spider-Syn**: The human-curated development set built upon Spider, for evaluating synonym substitution in real-world question paraphrases.
- **ADV$_{\textbf{BERT}}$:** The set of adversarial examples generated by BERT-Attack (Li et al., 2020).
- **ADV$_{\textbf{GLOVE}}$:** 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 Section 3.1. Following the case study in Section 3, 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$_{\textbf{O}}$**: Our modified RATSQL+GLOVE model trained on Spider using EMSL in training and evaluation, discussed in Section 3.3.
- **RATSQL$_{\textbf{B}}$**: 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$_{\textbf{BS}}$**: RATSQL+BERT trained on **Spider-Syn** using EMSL (Gan et al., 2021).
- **RATSQL$_{\textbf{G}}$**: RATSQL+GAP trained on Spider using EMSL (Shi et al., 2021).
- **w/o EMSL:** Models do not use EMSL in training and evaluation, consistent with Tables 1 and 2.
- **ManualMAS** (Gan et al., 2021): Schema annotations include synonyms used in Spider-Syn.
- **AutoMAS** (Gan et al., 2021): Schema annotations include synonyms generated according to the nearest GLOVE word vector.

4.3 Evaluation on Spider

Table 4 presents the exact matching accuracy of models trained on the Spider training set. It is clear that our RATSQL$_O$ significantly improves the without-EMSL performance. Tables 4 and 2 illustrate that the EMSL can be replaced by better encoding. The performance of RATSQL is slightly better than that of RATSQL$_O$, because Guo et al. (2019) conducted 100 time hyperparameter search to optimize the RATSQL while we did not do that. Therefore, when we modify the model structure, it may cause a slight performance degradation.

	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 3: Statistics of the types of error column predictions of different models evaluated on the Spider development set (the larger the number, the worse).

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

Table 4: Accuracy of two RATSQL models ablations on the development set.

Error Analysis Table 3 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 3.3. More importantly, the single-word performance of RATSQL$_O$ without EMSL is close to that of RATSQL and RATSQL$_O$. As discussed in Section 3.2, 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 3, it can also be found that the BERT brings around 5% absolute improvement on multi-word, while that on single-word is only 2%.

4.4 Robustness Evaluation

Typo Results Table 5 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. (2021) propose three development sets for evaluating the robustness of text-to-SQL models against synonym substitution, including: Spider-Syn, ADV$_{\text{BERT}}$, and ADV$_{\text{GLOVE}}$. Table 5 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 Section 4.5.

MAS Results Gan et al. (2021) also propose a MAS method to improve the robustness of text-to-SQL models. MAS provides multiple annotations

Approach	Spider	Spider-T1	Spider-T2	Spider-T3	Spider-Syn	ADV$_{\text{GLOVE}}$	ADV$_{\text{BERT}}$
RATSQL	**62.9%**	**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%**

Table 5: 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.

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

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 6. Comparing the data in Table 5 and Table 6, ManualMAS improves the performance of RATSQL$_B$ and RATSQL$_{BS}$ with and without EMSL on Spider-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.

4.5 Discussion

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 [4] extracted from RATSQL$_B$ with and without EMSL in Figure 3. 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 the correct schema item is linked to. Therefore, the attention of the RATSQL$_B$ without EMSL is

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

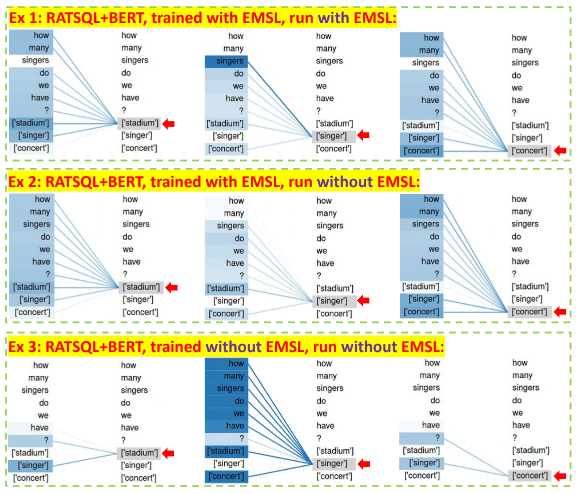

Ex 1: RATSQL+BERT, trained with EMSL, run with EMSL:
Ex 2: RATSQL+BERT, trained with EMSL, run without EMSL:
Ex 3: RATSQL+BERT, trained without EMSL, run without EMSL:
how
many
singers
do
we
have
?
['stadium']
['singer']
['concert']

Figure 3: 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.

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 3, which increases the difficulty of training.

5 Related Work

Schema Linking According to the review (Gan et al., 2020), schema linking is widely used in recent text-to-SQL models. In addition to discussing schema linking in the paper as part of the model (Guo et al., 2019; Bogin et al., 2019; Wang et al., 2020; Chen et al., 2020; Cao et al., 2021), some works focus on the 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). Our work found that once the model uses EMSL, it seems to become reliant on it and found that PLM can replace EMSL and make the model more robust.

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 supports 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. Huang et al. (2021) provide the empirical study on the robustness of semantic parsers in the presence of adversarial attacks, but this work focuses on single-domain. Gan et al. (2021) investigate the robustness against synonym substitution for cross-domain text-to-SQL translation and found that synonym substitution would break the EMSL leading a significant drop in performance. To solve this problem, Gan et al. (2021) proposed the MAS method to repair the broken EMSL. Following (Gan et al., 2021), our work found that the EMSL can be replaced by better encoding, and models without EMSL has better generalization ability.

6 Conclusion

In this work, we demonstrate that with the presence of pretrained language models, EMSL is no longer a necessary building block to ensure a high performance on text-to-SQL benchmarks. Specifically, we evaluate the robustness of text-to-SQL models with and without EMSL against synonym substitution and typos. 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. On the other hand, models without EMSL are more robust than those with EMSL, and the use of pretrained language models further show that they can replace the role of EMSL without compromising the performance on clean data. Therefore, we argue for the use of PLMs instead of EMSL for text-to-SQL tasks.

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.

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.

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, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. 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, Matthew Purver, and John R. Woodward. 2020. A review of cross-domain text-to-SQL models. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 108–115, Suzhou, China. 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.

Shuo Huang, Zhuang Li, Lizhen Qu, and Lei Pan. 2021. On robustness of neural semantic parsers.

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.

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.

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.

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.

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.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. *Mathematics of Computation*, 22(103):651.

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.

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.

A Visual Example of BERT Head

Unlike GLOVE, which gives static word vectors, BERT based on the attention mechanism makes adjacent word vectors in a sentence have a certain similarity. Figure 4, generated by the bertviz (Vig, 2019), presents the BERT head view of attention patterns in the one transformer layer where the word *bandmate* clearly links to the word *id*.

B 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.

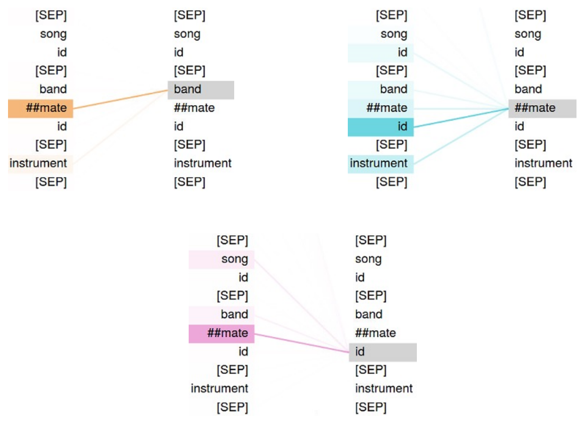


Figure 4: The BERT head view of attention patterns of word *bandmate* and *id* in the one transformer layer.

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.

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.