Multi-Hop Table Retrieval for Open-Domain Text-to-SQL

Anonymous ACL submission

Abstract

Open-domain text-to-SQL is an important task that retrieves question-relevant tables from massive databases and then generates SQL. However, existing retrieval methods that retrieve in a single hop do not pay attention to the text-to-SQL challenge of schema linking, which is aligning the entities in the question with table entities, reflected in two aspects: similar irrelevant entity and domain mismatch entity. Therefore, we propose our method, the multi-hop table retrieval with rewrite and beam search (MURRE). To reduce the effect of the similar irrelevant entity, our method focuses on unretrieved entities at each hop and considers the low-ranked tables by beam search. To alleviate the limitation of domain mismatch entity, MURRE rewrites the question based on retrieved tables in multiple hops, decreasing the domain gap with relevant tables. We conduct experiments on SpiderUnion and BirdUnion+, reaching new state-of-the-art results with an average improvement of 6.38%.[1]

1 Introduction

Text-to-SQL is a vital natural language processing task that lowers the difficulty of accessing databases, helping people query data efficiently, which is widely used in finance, education, and business (Qin et al., 2022). Different from the previous text-to-SQL task which is close-domain, a setting that is close to real-world scenarios is the open-domain text-to-SQL, which requires converting user questions to SQL in the face of countless tables (Kothyari et al., 2023). Specifically, open-domain text-to-SQL requires retrieving the question-relevant tables from open-domain databases which include multiple tables, i.e., the retrieval module, and then generating the SQL based on the question and retrieved tables, i.e., the text-to-SQL module. We refer to the table relevant to the question as the relevant table in the paper.

[1]Our code and data will be public upon acceptance.

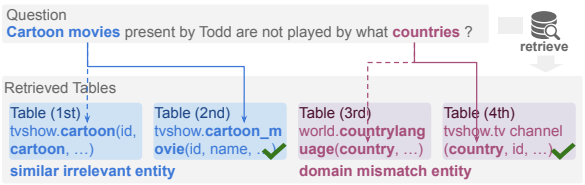


Figure 1: The two limitations of existing retrieval methods on schema linking. The sequence in brackets of each table denotes the retrieval rank, ✓ represents the question-relevant table. Solid arrows denote the correct schema linking, dotted arrows denote the incorrect.

The retrieval module in the open-domain text-to-SQL task should have a high recall to retrieve all relevant tables because generating correct SQL requires all relevant tables. However, most existing retrieval methodologies do not focus on **schema linking** that inherited from the text-to-SQL (Guo et al., 2019; Yin et al., 2020; Wang et al., 2020), which refers to aligning the entities between the question and the table (e.g., table name, column name), limiting the recall improvement.

Considering schema linking, CRUSH (Kothyari et al., 2023) rewrites the user question by using LLM to fit possible relevant tables, which is called the tabularized question, and retrieves in a single hop. Nevertheless, like most previous work, CRUSH still has two limitations on schema linking as shown in Figure 1. **1. Similar irrelevant entity** refers to that one entity in the question is similar to entities of irrelevant tables, causing relevant tables linked to other entities in the question inability to be retrieved at high ranks in a single hop. As presented in Figure 1, the relevant table *"tv channel"* is retrieved at a lower rank because the similar entity *"cartoon"* is linked mistakenly. **2. Domain mismatch entity** refers to that the entity in question could mismatch the relevant domain, causing the retrieved table to be further away from the relevant domain, creating the domain gap. As shown in Figure 1, the retrieved table in the *"world"* domain is far from the relevant table in the *"tv show"* domain because of the *"countries"* in the question.

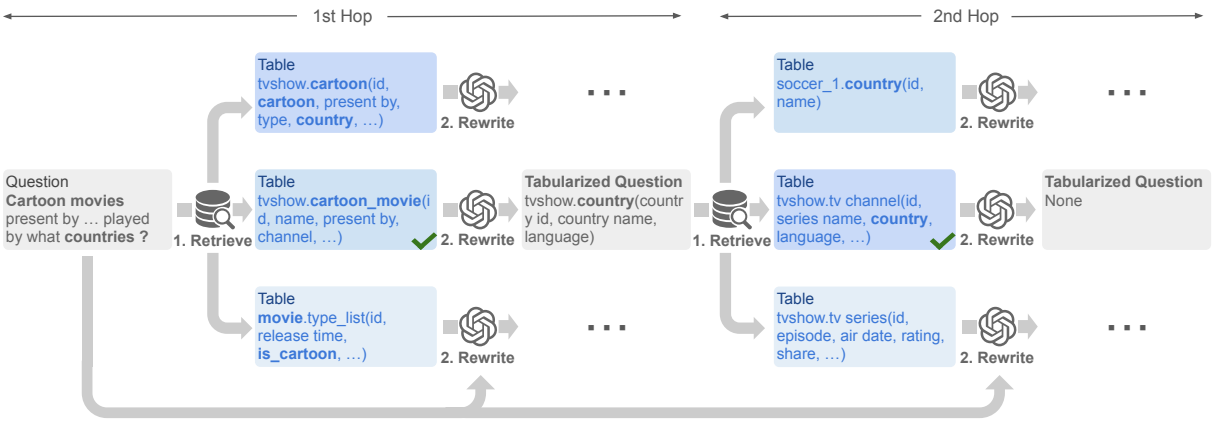


Figure 2: An overview of MURRE with multiple hops. Each hop consists of: *1.* **Retrieve** the tables similar to the question; *2.* **Rewrite** the question to a tabularized question by fitting unretrieved tables with LLM based on the original question and retrieved tables. We employ the beam search paradigm to maintain multiple retrievals at each hop. The table color depth represents the similarity with the question in the hop, and ✓ represents the relevant table. We demonstrate an example of MURRE with 2 hops for brevity.

Consequently, to enhance the table retrieval performance, a good retrieval module in the open-domain text-to-SQL system should fulfill the following two requirements to solve the limitations of schema linking presented above: *1. can link the entities in the low-ranked tables to relevant tables*; *2. can effectively reduce the gap between the question and relevant tables*.

Therefore, we propose our method, **MU**lti-hop table **R**etrieval with **R**ewrite and b**E**am search (MURRE), to enhance the recall of open-domain text-to-SQL. Our method retrieves the relevant tables with multi-hop (Xiong et al., 2021; Lee et al., 2022), rewrites the question at each hop by fitting unretrieved tables based on retrieved tables with LLM, and employs the beam search paradigm to select multiple tables and maintain multiple retrievals at each hop inspired by Zhang et al. (2023). For the first requirement, MURRE removes retrieved information at each hop to guide the module to get unretrieved tables and then employs beam search to set the low-ranked tables as the candidate results, thereby better retrieving the relevant tables at low ranks in a single hop, mitigating the effect of similar irrelevant entity. For the second requirement, our method rewrites the question referenced to the retrieved tables in multiple hops, decreasing the domain gap, and thereby enhancing the performance of the schema linking.

To validate the effectiveness of MURRE, we conduct experiments on two datasets, SpiderUnion (Yu et al., 2018; Kothyari et al., 2023) and BirdUnion+, which we propose based on Bird (Li et al., 2023b). Compared to previous methods, our method gets an average of 6.38% retrieval performance improvement across all experimental datasets, achieving new state-of-the-art (SOTA) results, proving the effectiveness of MURRE.

Our contributions are as follows:

- To alleviate the limitation of the similar irrelevant entity, we propose employing the multi-hop retrieval with the beam search paradigm to focus on the low-ranked tables, detecting the unretrieved relevant tables.
- To eliminate the limitation of domain mismatch entity, we propose to use LLM to fit unretrieved tables according to retrieved tables in the multi-hop retrieval, which effectively reduces the domain gap with relevant tables.
- To validate the effectiveness of MURRE, we validate it on the SpiderUnion and BIRDUnion+ and achieve new SOTA with an average of 6.38% improvement, proving that our method is effective.

2 Methodology

2.1 Overview

MURRE aims to retrieve the relevant tables from massive tables according to the user question. The retrieved tables would be fed into the text-to-SQL module to translate the question to the corresponding SQL. To alleviate the limitations of schema linking in a single hop, we present MURRE which employs multi-hop retrieval, rewriting the question to unretrieved tables and maintaining multiple retrievals at each hop. The overview of MURRE is shown in Figure 2.

In MURRE, each hop consists of two phases: **Retrieve** (§2.2) and **Rewrite** (§2.3). In the first hop retrieval, we use the original user question to retrieve tables, while in subsequent hops, we use the tabularized question which is rewritten by LLM to retrieve. Also, we use the beam search paradigm which maintains multiple retrieval lists at each hop by updating and selecting top retrieval lists at the end of Retrieve, and input each selected retrieval list to the Rewrite phase separately. MURRE repeats the above process until reaches the maximum hop number, or meets the early stop condition, which is discussed in §2.3. After the multi-hop retrieval, we **Rank** (§2.4) each table and input the top multiple tables and the user question to the text-to-SQL module

2.2 Retrieve

In the Retrieve phase, we retrieve tables based on the question in the hop, update, and select the retrieval lists. First, we linearise the tables, which include the database name, table name, and column names. We then embed the question and linearized tables as vectors using the embedding and compute the cosine similarity between the question vector and the table vector as the raw score of the table. We update the retrieval list by adding the table retrieved at the hop to its corresponding retrieval list which includes the tables retrieved at previous hops. We maintain multiple retrieval lists for each original user question, by selecting the retrieval lists with top scores at the end of Retrieve in each hop, where the score of the retrieval list is the product of all the raw table scores in the list.

2.3 Rewrite

In the Rewrite phase, we rewrite the question with LLM referenced to retrieved tables and determine if to early stop at the hop. To reduce the domain gap and retrieve the low-rank tables, we prompt LLM to fit the unretrieved table according to retrieved tables and use the generation of LLM as the question in the next hop. Since each user question requires a different number of tables, to avoid extra hops introducing errors, MURRE can automatically determine whether the retrieved tables are sufficient to answer the question, i.e., early stop. We prompt LLMs to generate a special mark to indicate that the retrieved tables are sufficient to answer the question, where we stop the retrieval if this special mark is generated. The prompts we use are shown in Appendix A.

Algorithm 1 The table scoring algorithm in MURRE

Input: The similarity corresponding to each table t in each hop h: $all\_lists = [[(table_{11}, score_{11}), \ldots, (table_{1H}, score_{1H})], \ldots, [(table_{T1}, score_{T1}), \ldots, (table_{LH}, score_{LH})]]$, the number of max hops H, the number of all lists L.

Output: The scores of each table t

```
Initialization : table_score ← {}
for each_list in all_lists do
    score ← 1
    for example in each_list do
        score = score × example[1]
    end for
    for example in each_list do
        table_score[example[0]] ←
        max(score,
        table_score[example[0]])
    end for
end for
return table_score
```

For example, in Figure 2, we prompt the LLM to fit the unretrieved table given the question *"What countries that not playing cartoons written by Todd Casey?"* and retrieved table *"tvshow.cartoon(...)"*, obtaining *"tvshow.country(country id, country name, language)"* as the question for the second hop retrieval. At the second hop, we prompt the LLM given the original question and retrieved tables *"tvshow.cartoon(...)"* and *"tvshow.tv channel(...)"*, and then the LLM generates *"None"* which is the special mark of early stop, showing that the retrieved tables are sufficient and the retrieval stops.

2.4 Rank

After completing all hops of retrieval, because each table could have multiple scores obtained during multiple hops and beam search, we propose a table scoring strategy to integrate the similarity and obtain the final retrieval results of tables, as shown in Algorithm 1. We multiply the similarity scores in the retrieval list as the score of each table, and in the face of the same table being retrieved multiple times, we select its highest multiplied score as the final score of this table. We follow Algorithm 1 to get the final score for each table, select the top multiple tables according to the score, and then input them to the text-to-SQL module.

Dataset	#table				
	1	2	3	4	All
SpiderUnion	395	214	43	6	658
BirdUnion+	364	943	207	20	1534

Table 1: Statistics on the number of the relevant table for each question in the SpiderUnion and BirdUnion+. **#table** denotes the number of the relevant table. **All** refers to the total number of questions in the dataset.

3 Experiments

3.1 Experiment Setup

Dataset To verify the effectiveness of MURRE, we validate MURRE on two open-domain text-to-SQL datasets: SpiderUnion (Kothyari et al., 2023) and BirdUnion+. SpiderUnion is sourced from the Spider (Yu et al., 2018) dev-set. Also, we propose BirdUnion+, which is created by combining tables in Bird (Li et al., 2023b) train-set and dev-set. We count the number of questions requiring different numbers of tables, as shown in Table 1. We introduce Spider and Bird in Appendix B.

Metric We use recall and complete recall as evaluation metrics for retrieval, and Execution Accuracy (EX) (Yu et al., 2018) for text-to-SQL. Recall, as an important indicator in information retrieval, is the proportion of relevant tables retrieved to all relevant tables, which we use following the previous work (Kothyari et al., 2023). However, in the open domain text-to-SQL, we are more interested in whether all relevant tables are retrieved, deciding the error cascading to text-to-SQL, thereby we propose complete recall, which measures whether all relevant tables are retrieved. For the text-to-SQL task, following the previous work (Gao et al., 2023a), we use execution match (EX) to measure the correctness of the execution results of predicted SQL compared to those of gold SQL.

Model We use SGPT (Muennighoff, 2022), the widely-recognized Dense Passage Retrieval (DPR) baseline, with two different scales of models SGPT-125M and SGPT-5.8B, as the embedding in the retrieval experiments following the previous work (Kothyari et al., 2023) and limited by API. For the Rewrite phase, we use the `gpt-3.5-turbo` with the few-shot prompt. For text-to-SQL, we use the `gpt-3.5-turbo` to generate SQL under the zero-shot setting. We present the SGPT and `gpt-3.5-turbo` in detail in Appendix C.

Comparing System In our experiments, we compare MURRE with the following methods: baseline, which retrieves tables based on the user question in a single hop, and CRUSH (Kothyari et al., 2023).

Implement Details We set the beam size to 5 since the performance of MURRE with the beam size is the best with the smallest beam size (see § 3.4.2). We set the max hop to 3 because the proportion of questions requiring less than or equal to 3 tables in the SpiderUnion and BirdUnion+ datasets is both more than 98% (see Table 1).

3.2 Main Result

The main results of our experiments are shown in Table 2. Compared with the baseline and CRUSH, MURRE has obvious improvements in different datasets, models of different scales, with an average improvement of 6.38%, reaching a new SOTA, proving the effectiveness of our method. We analyze the performance of MURRE from three perspectives: dataset, model, and metric, and obtain the following conclusions.

The improvement of MURRE on BirdUnion+ is more significant than on SpiderUnion. Because the questions in BirdUnion+ demand more tables on average (see Table 1), requiring multi-hop retrieval of MURRE more to retrieve multiple relevant tables, thus improving retrieval performance.

MURRE improves the performance more with SGPT-125M compared with SGPT-5.8B. SGPT-5.8B, as an embedding with a larger parameter scale, has a stronger capability to embed questions and relevant tables into similar vectors, so the Rewrite benefits SGPT-5.8B less than SGPT-125M. The performance of CRUSH with SGPT-5.8B on both datasets is behind the baseline also because of the strong embedding capabilities of SGPT-5.8B.

MURRE improves performance more with small top numbers than with large top numbers. Because improving metrics with large top numbers requires retrieving relevant tables that are extremely dissimilar to the user question, our method improves the metrics with more difficulty. Especially, the $recall@20$ and complete recall $k = 20$ of MURRE and CRUSH on SpiderUnion with SGPT-5.8B declines compared with the baseline because even with LLM, it is still difficult to fit the extremely dissimilar tables, introducing errors and leading to retrieval far away from the relevant tables compared with the baseline.

Dataset	Model	Method	$k=3$	$k=5$	$k=10$	$k=20$	r@3	r@5	r@10	r@20
SpiderUnion	SGPT-125M	baseline	54.3	66.0	75.4	82.2	63.0	73.1	80.7	86.3
		CRUSH†	60.2	71.3	80.7	**86.8**	68.9	76.3	**83.4**	**88.9**
		MURRE	**65.0**	**74.2**	**81.0**	85.3	**70.2**	**77.5**	82.3	86.9
	SGPT-5.8B	baseline	76.3	86.8	94.1	**97.6**	84.0	91.5	96.2	**98.7**
		CRUSH†	68.2	80.1	88.4	92.2	75.5	85.1	91.2	94.5
		MURRE	**86.0**	**93.5**	**96.7**	97.3	**89.3**	**94.3**	**96.8**	97.5
BirdUnion+	SGPT-125M	baseline	39.0	50.3	62.1	70.9	54.0	63.2	73.3	80.9
		CRUSH†	42.1	56.1	70.2	77.7	60.2	70.0	79.5	**86.1**
		MURRE	**51.4**	**62.7**	**72.9**	**78.3**	**64.8**	**72.7**	**79.6**	84.2
	SGPT-5.8B	baseline	55.3	67.3	79.4	86.4	72.9	80.8	88.6	92.8
		CRUSH†	52.2	63.5	78.4	88.1	70.0	77.9	87.5	93.0
		MURRE	**69.1**	**80.1**	**88.7**	**92.7**	**81.0**	**87.6**	**92.6**	**95.4**

Table 2: The main results on complete recall and recall of MURRE, compared with baseline and CRUSH on SpiderUnion and BirdUnion+, using SGPT-125M and SGPT-5.8B. k refers to the complete recall, and r refers to the recall. † denotes our run since the performance difference led by the API change. The best results of different datasets and models are annotated in **bold**.

Model	Method	3	5	10	20
SGPT-125M	baseline	43.2	48.2	50.8	52.7
	MURRE	**50.8**	**52.9**	**54.6**	**56.5**
SGPT-5.8B	baseline	55.3	57.4	60.3	57.8
	MURRE	**59.9**	**62.5**	**63.5**	**62.3**

Table 3: EX for predicted SQL with the input, which includes the user question and different numbers of retrieved top tables on the SpiderUnion. The best results with different models are annotated in **bold**.

Text-to-SQL Experiments We perform the text-to-SQL experiments on the SpiderUnion with the user question and retrieved tables, as presented in Table 3. Since Spider is the mainstream dataset for text-to-SQL, we select SpiderUnion corresponding to Spider to perform subsequent experiments, and the text-to-SQL results on BirdUnion+ are presented in Appendix D. MURRE, achieving higher recall in the retrieval method, outperforms the baseline consistently in the text-to-SQL experiment. As the number of input tables increases, the EX improvement slows down and even declines from the top number is 10, because too many irrelevant tables make it difficult for the method to focus on the tables relevant to the question. This also proves the necessity of MURRE improving retrieval performance with small top numbers under the open domain text-to-SQL setting.

3.3 Ablation Studies

To prove the effectiveness of our method, we conduct ablation experiments on SpiderUnion. The results of the experiments are shown in Table 4. Since SGPT-125M and SGPT-5.8B show the same trend with different datasets and methods in Table 2 and 3, we use SGPT-125M for subsequent experiments to trade off the embedding speed and retrieval recall (Muennighoff et al., 2023).

The Effectiveness of Rewrite To demonstrate the effectiveness of the Rewrite with LLM in our method, we compare the performance of directly splicing the user question and retrieved tables of each hop without rewrite. Compared with MURRE, the performance of splicing methods drops significantly and consistently, demonstrating the effectiveness of rewriting with LLM in our method and the necessity of mitigating similar irrelevant entities by removing already retrieved information.

The Effectiveness of Rewrite to Table To prove the effectiveness of rewriting questions into the form of the table in MURRE, we rewrite the questions at each hop into natural language questions that query about unretrieved information to conduct experiments. It can be found that compared to rewriting to natural language, rewriting to table significantly improves performance, proving the effectiveness of rewriting to table in MURRE.

The Effectiveness of Early Stop To verify the effectiveness of early stop in MURRE, we compare the results without employing the mechanism of early stop, which does not prompt the model to generate the special early stop mark. The performance without early stop is significantly degraded, which proves that the introduction of early stop in MURRE effectively guarantees the performance.

Method	$k=3$	$k=5$	$k=10$	r@3	r@5	r@10
MURRE	**65.0**	**74.2**	**81.0**	**70.2**	**77.5**	**82.3**
w/o rewrite	46.2 (−18.8)	56.7 (−17.5)	67.2 (−13.8)	50.6 (−19.6)	60.7 (−16.8)	70.0 (−11.6)
w/o tabulation	54.6 (−10.4)	64.9 (−9.3)	75.5 (−5.5)	63.4 (−6.8)	72.5 (−5.0)	80.9 (−1.4)
w/o early stop	52.6 (−12.4)	64.9 (−9.3)	71.0 (−10.0)	57.1 (−13.1)	67.0 (−10.5)	72.2 (−10.1)

Table 4: The ablation results on evaluating the MURRE of rewriting user question to tables, compared with splicing the question and previously retrieved tables (denoted as *w/o rewrite*), rewriting to natural language question (denoted as *w/o tabulation*), and without employing the mechanism of early stop (denoted as *w/o early stop*) on SpiderUnion with SGPT-125M. k refers to the complete recall, and r refers to the recall. The best results are annotated in **bold**.

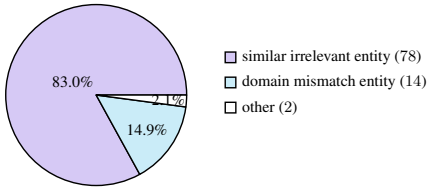


Figure 3: The proportion of performance improvements due to solving different limitations by MURRE on SpiderUnion compared with CRUSH. The number in parentheses in the legend represents the number of examples with the corresponding limitation type.

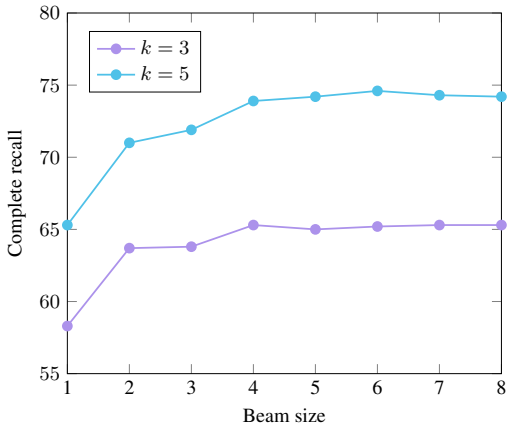


Figure 4: The complete recall with different beam sizes on SpiderUnion with SGPT-125M.

3.4 Analysis

The increasing trend of the performance in the text-to-SQL becomes slow or even drops when inputting retrieved tables more than 5 as shown in Table 3, and considering that the two SpiderUnion and BirdUnion+ datasets require up to 4 tables for each question, so in the following analysis experiments, we are mainly concerned with the performance of the top 5 retrieval results. Furthermore, complete recall $k=5$ is a more strict indicator than $recall@5$, so we mainly utilize complete recall $k=5$ as the evaluation metric in the analysis.

3.4.1 Limitations of Recall Improvement

To explore why our method can improve the retrieval performance significantly, we analyze the reasons for performance improvement of our method compared with CRUSH and count their proportion in Figure 3. It can be found that our method improves the retrieval performance mainly because MURRE can alleviate the limitations of similar irrelevant entity and domain mismatch entity. Our statistical criteria is presented in Appendix E.

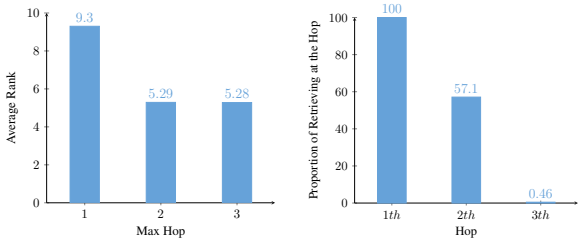
3.4.2 Beam Size

To observe the impact of different beam sizes on the retrieval performance, we compare the performance of our method using SGPT-125M as the embedding on the SpiderUnion dataset under the setting of different beam sizes, as shown in Figure 4.

Max Hop	#table 1	2	3	≥ 4	All
1	73.7	59.8	25.6	**50.0**	66.0
2	73.2	77.6	**58.1**	**50.0**	73.4
3	**74.2**	**78.0**	**58.1**	**50.0**	**74.2**
4	**74.2**	**78.0**	**58.1**	**50.0**	**74.2**

Table 5: Complete recall $k=5$ of MURRE with different numbers of the max hop. We divide the SpiderUnion according to the number of the relevant tables (denoted as **#table**) for each question. **All** refers to the whole SpiderUnion that is not divided. The best results with different tables are annotated in **bold**.

It can be found that as the beam size increases, complete recall presents an obvious upward trend until the beam size is 5, and then the performance increases slightly or even declines, which proves that within a certain range of less than 5, the increase in beam size promotes performance improvement. However, too large beam size which is more than 5 introduces too many irrelevant tables, which not only costs more computing overheads but is also no longer helpful for improving performance.



(a) (b)

Figure 5: (a) The average rank of relevant tables with different numbers of max hop on the SpiderUnion using SGPT-125M with MURRE. (b) The proportion that still retrieves at each hop when the max hop is 3 on the SpiderUnion using SGPT-125M with MURRE.

3.4.3 Max Hop

To verify the effectiveness of the multi-hop in our method, we conduct experiments on the SpiderUnion divided based on the number of relevant tables for each question using the SGPT-125M and compare the complete recall $k = 5$ with different numbers of max hop, as shown in Table 5. From Table 5, we can see that: *1*. The overall trend of MURRE is to achieve the best performance when the number of max hops is greater than or equal to the required number of tables. *2*. For questions requiring 1 and 2 tables, the best performance is achieved at the max hop of 3, which shows that our method can not only retrieve more relevant tables but also reduce the gap between the question and relevant tables during multiple hops. *3*. The performance of questions requiring 1 table drops slightly with 2 hops because the rewriting of questions with 1 required table could introduce errors, but this error is reduced and eliminated with the max hop of 3. *4*. For questions requiring ≥ 4 tables, complete recall $k = 5$ requires the retrieved top 5 tables to contain all relevant tables, which is difficult to improve, leading to their performance remaining unchanged in multiple hops.

3.4.4 Average Rank

To verify that our method can improve the average rank of relevant tables, we compute the average rank of relevant tables at different numbers of max hop in the results of MURRE, as shown in Figure 5a. MURRE can indeed improve the average rank of relevant tables. The improvement at the max hop of 2 is the most significant because most questions in the SpiderUnion require 1 or 2 tables (see Table 1) and need 2 hops to link to different tables.

Method	Level				
	Easy	Medium	Hard	Extra	All
baseline	70.5	71.1	55.8	51.3	66.0
MURRE	**71.8**	**76.0**	**73.3**	**73.1**	**74.2**

Table 6: Complete recall $k = 5$ of MURRE compared with the baseline in different SQL hardness levels on SpiderUnion. **Extra** denotes extra hard. **All** refers to the performance of the whole SpiderUnion dataset. The best results of different hardness are annotated in **bold**.

Also, the improvement at the max hop of 3 is weak, not only because the questions that need 3 or 4 tables are too few, but also because we take the mechanism of early stop causing most questions in SpiderUnion to stop retrieving before the $3th$ hop which is presented in Figure 5b.

3.4.5 SQL Hardness

To observe the retrieval performance of our method adapting for SQL of different hardness levels, we categorize the SQL and its corresponding question according to the SQL hardness criteria (Yu et al., 2018) and calculate the retrieval performance of different hardness levels, as shown in Table 6. MURRE improves performance more significantly for more difficult SQL questions. Because more difficult SQL often requires more tables to operate and query, the baseline is challenging to retrieve all relevant tables merely in a single hop, while our method can retrieve more relevant tables at low ranks and reduce the domain gap with relevant tables with multi-hop retrieval.

3.4.6 Case Study

We demonstrate an example with MURRE compared with baseline and CRUSH, as shown in Figure 6. Baseline and CRUSH fail to retrieve the *"world_1.countrylanguage"* table at top 3 since there are many similar entities to *"city"* and *"population"* causing the irrelevant tables to occupy high ranks in a single hop. Meanwhile, the tabularized question enlarges the domain gap further with CRUSH. In comparison, MURRE adopts multi-hop retrieval to separately link the entities in the question and successfully retrieve *"world_1.city"* and *"world_1.countrylanguage"*. Also, our method rewrites the question according to the top 3 tables separately, including one relevant table *"world_1.city"* causing the rewritten question to keep the same domain as the relevant table and easily link to *"world_1.countrylanguage"*. The detailed case can be seen in Appendix F.

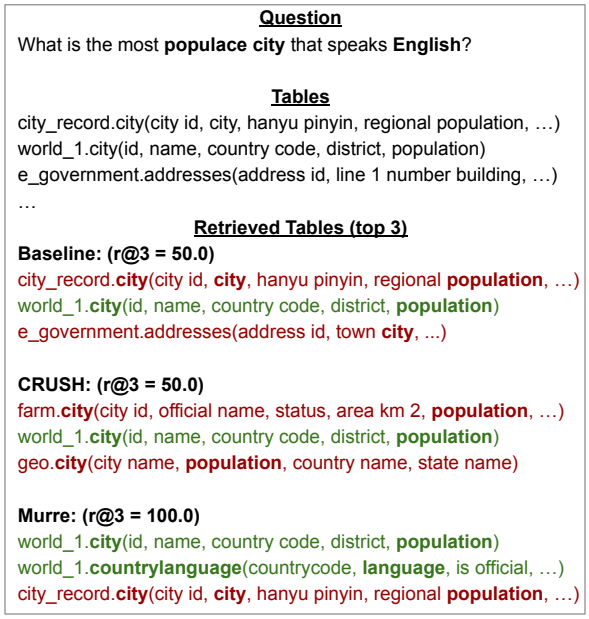


Figure 6: Case study comparing MURRE with baseline and CRUSH. The green means the relevant table, while the red means irrelevant. Each table is expressed in the form of "database name.tabel name(column name, column name, ...)". r denotes recall.

4 Related Work

4.1 Text-to-SQL

Text-to-SQL is an important task because it can convert the user question into SQL, helping people access databases efficiently (Qin et al., 2022). Currently, LLM-based methods become the mainstream method in text-to-SQL, because they surpass the performance of pre-trained language models with only a small amount of annotated data (Li et al., 2023a; Gao et al., 2023a). For example, to solve the example selection in text-to-SQL, DAIL-SQL (Gao et al., 2023a) proposes to use masked question similarity selection. However, these methods do not focus on the open-domain text-to-SQL task and exist a gap with real-world applications. To solve the problem, CRUSH (Kothyari et al., 2023) proposes to retrieve relevant tables using the LLM hallucination before text-to-SQL.

However, existing methods can only rely on single-hop retrieval where similar entities cause irrelevant tables to have high ranks, and the tabularized question is only generated based on the question causing the domain gap. To solve this problem, we propose a multi-hop retrieval method for open-domain text-to-SQL, which rewrites the question referenced to the retrieved tables with LLM.

4.2 LLM-based Retrieval

The existing methods of LLM-based retrieval leverage the powerful in-context learning capabilities of LLM and knowledge stored in parameters to enhance retrieval and prove the effectiveness on multiple benchmarks (Gao et al., 2023b). Among them, some studies focus on iterating the process of retrieval with retriever and generation with LLM, which can improve retrieval performance. For example, Self-Ask (Press et al., 2023) uses LLM to decompose the user question into the next sub-question dynamically based on the original question and current intermediate answer and calls the search engine to retrieve the next intermediate answer. To reduce the overheads of retrieval and generation, ITER-RETGEN (Shao et al., 2023) proposes to splice the question and generation of LLM as the new retrieval query for the next iteration.

However, these LLM-based methods can not adapt to open-domain text-to-SQL directly, because most entities in the tables are abbreviations and simpler than expressions of natural language, leading to that there exist many similar entities in the different tables. If only selecting the most similar table as the intermediate result, it could be irrelevant and cause subsequent retrieval and text-to-SQL tend to the irrelevant tables. To solve the problem, we employ the beam search paradigm to select multiple possible tables at each hop and maintain multiple retrieval lists, effectively alleviating the limitation of similar irrelevant entities.

5 Conclusion

In the paper, we figure out that most previous retrieval methods do not pay attention to schema linking in the open-domain text-to-SQL, limiting the performance reflected in the similar irrelevant entity and domain mismatch entity. To solve the limitations, we propose MURRE which employs the multi-hop retrieval to focus on the unretrieved entities and rewrite the question based on the retrieved tables at each hop to reduce the domain gap between the tabularized question and relevant tables, alleviating the limitations of the similar irrelevant entity and domain mismatch entity separately. MURRE achieves an average of 6.38% performance improvement on SpiderUnion and BirdUnion+ datasets and reaches new SOTA results, verifying the effectiveness of our method. Our analysis experiments prove that MURRE indeed alleviates the two limitations above.

Limitations

We discuss the limitations of our work from the following two aspects. *1*. Considering the efficiency, our method significantly improves the retrieval recall, however, our method also reduces the efficiency of retrieval. We leave the trade-off between efficiency and recall as future work. *2*. From the perspective of recall, our method does not consider the recall improvement brought by the text-to-SQL feedback (Trivedi et al., 2023; Yu et al., 2023). We leave the retrieval recall improvement leveraging the results of text-to-SQL for future work. Although our method achieves significant improvements, future work can improve our method from the aspects of efficiency and recall further.

Ethics Statement

Every dataset and model used in the paper is accessible to the public, and our application of them adheres to their respective licenses and conditions.

References

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

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997.

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*. Association for Computational Linguistics.

Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy J. Lin. 2020. Which bm25 do you mean? a large-scale reproducibility study of scoring variants. *Advances in Information Retrieval*, 12036:28 – 34.

William Kent. 1991. Solving domain mismatch and schema mismatch problems with an object-oriented database programming language. In *Very Large Data Bases Conference*.

Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. Opentab: Advancing large language models as open-domain table reasoners. In *The Twelfth International Conference on Learning Representations*.

Mayank Kothyari, Dhruva Dhingra, Sunita Sarawagi, and Soumen Chakrabarti. 2023. CRUSH4SQL: Collective retrieval using schema hallucination for Text2SQL. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative multi-hop retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press.

Jinyang Li, Binyuan Hui, GE QU, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. 2023c. Llatrieval: Llm-verified retrieval for verifiable generation. *ArXiv*, abs/2311.07838.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *ArXiv*, abs/2202.08904.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *ArXiv*, abs/2208.13629.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 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*. Association for Computational Linguistics.

Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21. Association for Computing Machinery.

Tianshu Wang, Hongyu Lin, Xianpei Han, Le Sun, Xiaoyang Chen, Hao Wang, and Zhenyu Zeng. 2023. Dbcopilot: Scaling natural language querying to massive databases. *ArXiv*, abs/2312.03463.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 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. 2018. 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*. Association for Computational Linguistics.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Jiahao Zhang, H. Zhang, Dongmei Zhang, Yong Liu, and Sheng Huang. 2023. Beam retrieval: General end-to-end retrieval for multi-hop question answering. *ArXiv*, abs/2308.08973.

A Prompts for Rewrite

We show the prompts we use to rewrite the question on SpiderUnion (see Table 7) and BirdUnion+ (see Table 8). We only show the first two examples here limited by pages. The code and the whole prompt will be public in the future.

B Dataset Details

Spider (Yu et al., 2018) is a multi-domain mainstream text-to-SQL dataset that contains 658 questions, with an average of 1.48 tables per question in the dev-set. Bird (Li et al., 2023b), as a text-to-SQL dataset, is closer to the actual scenario featuring its larger scale and more difficult questions. Bird contains 1534 questions, with an average of 1.92 tables per question in the dev-set.

C Model Details

SGPT is the popular retrieval baseline, employing a decoder-only architecture and showing excellent performance on tasks such as sentence matching. The `gpt-3.5-turbo` model has undergone instruction fine-tuning and human alignment and has superior in-context learning and inference capability.

D Additional Results

We show the results of text-to-SQL with the input including the user question and retrieved top schema on BirdUnion+ in Table 9.

E Statistical Criteria of Limitations

To facilitate statistics on the number of results reflected in the two limitations of similar irrelevant entities and domain mismatch entity, we set the following rules.

For the limitation of similar irrelevant entities, our statistical standard is that the entity in the irrelevant schema appears in the question. If the same entity appears in the schema as in the question, then intuitively we consider that the schema is similar to the question, and in practice, the cosine similarity between the schema and the question after embedding is also high (Kamphuis et al., 2020; Wang et al., 2021, 2023; Li et al., 2023c; Kong et al., 2024).

For the limitation of domain mismatch entity, if all entities in the rewritten question do not appear in the relevant schema, we consider that the rewritten question does not match the domain of the relevant schema. If the rewritten question does not overlap with any entity in the relevant schema, the retrieval similarity is also low, which also means that the rewritten question cannot be well matched with the relevant schema and reflected on the domain (Kent, 1991).

F Detailed Case Study

We present one example in detail with MURRE compared with the baseline and CRUSH in Table 10. We demonstrate the example, with setting the beam_size to 3 and max hop to 3, while it stops early at the second hop. The baseline and CRUSH are both single-hop retrieval and suffer from the limitation of similar irrelevant entity and fail to retrieve the *"world_1.countrylanguage"* table in top 3. Moreover, CRUSH rewrites the question that belongs to the *"population"* domain and still mismatches the domain of relevant tables *"world_1"*. However, MURRE retrieves the left table *"world_1.countrylanguage"* at the second hop by removing the retrieved information from the question and matching the *"world_1.countrylanguage"* more referenced to the retrieved table *"world_1.city"*.

Given the following SQL tables, your job is to complete the possible left SQL tables given a user's request.
Return None if no left SQL tables according to the user's request.

Question: Which models are lighter than 3500 but not built by the 'Ford Motor Company'?
Database: car_1.model list(model id, maker, model)
car_1.cars data(id, mpg, cylinders, edispl, horsepower, weight, accelerate, year)
car_1.car names(make id, model, make)
Completing Tables: car_1.car makers(id, maker, full name, country)

Question: Which employee received the biggest bonus? Give me the employee name.
Database: employee_hire_evaluation.evaluation(employee id, year awarded, bonus)
employee_hire_evaluation.employee(employee id, name, age, city)
Completing Tables: None
...

Table 7: The prompt we use for the SpiderUnion with `gpt-3.5-turbo`.

Given the following SQL tables, your job is to complete the possible left SQL tables given a user's request.
Return None if no left SQL tables according to the user's request.

Question: What was the growth rate of the total amount of loans across all accounts for a male client between 1996 and 1997?
Database: financial.client(client_id, gender, birth_date, location of branch)
financial.loan(loan_id, account_id, date, amount, duration, monthly payments, status)
Completing Tables: financial.account(account id, location of branch, frequency, date)
financial.disp(disposition id, client_id, account_id, type)

Question: How many members did attend the event 'Community Theater' in 2019?
Database: student_club.Attendance(link to event, link to member)
Completing Tables: student_club.Event(event id, event name, event date, type, notes, location, status)
...

Table 8: The prompt we use for the BirdUnion+ with `gpt-3.5-turbo`.

Model	**Method**	**3**	**5**	**10**	**20**
SGPT-125M	Baseline	11.0	12.5	15.4	16.2
	MURRE	**15.7**	**17.1**	**19.0**	**18.1**
SGPT-5.8B	Baseline	15.7	17.7	19.0	19.5
	MURRE	**20.3**	**21.5**	**22.2**	**21.9**

Table 9: EX for predicted SQL with the input, which includes the user question and different numbers of retrieved top schema on BirdUnion+.

Question
What is the most **populace city** that speaks **English**?

Tables
city_record.city(city id, city, hanzi, hanyu pinyin, regional population, gdp)
world_1.city(id, name, country code, district, population)
e_government.addresses(address id, line 1 number building, town city, zip postcode, state province county, country)
...

Baseline ($r@3 = 50.0$)
Tabularized Question
-
Retrieved Tables (top 3)
city_record.**city**(city id, **city**, hanzi, hanyu pinyin, regional population, gdp)
world_1.**city**(id, name, country code, district, **population**)
e_government.addresses(address id, line 1 number building, town city, zip postcode, state province county, country)

CRUSH ($r@3 = 50.0$)
Tabularized Question
population(city, language, population size)
Retrieved Tables (top 3)
farm.**city**(city id, official name, status, area km 2, **population**, census ranking)
world_1.**city**(id, name, country code, district, **population**)
geo.**city**(city name, **population**, country name, state name)

MURRE ($r@3 = 100.0$)
Retrieved Tables in 1st Hop (top 3)
city_record.city(city id, **city**, hanzi, hanyu pinyin, regional population, gdp)
world_1.**city**(id, name, country code, district, **population**)
e_government.addresses(address id, line 1 number building, town city, zip postcode, state province county, country)
Tabularized Questions
city_record.language(city id, language, percentage)
world_1.countrylanguage(countrycode, language, is official, percentage)
e_government.languages(language id, language name, language code, population)
Retrieved Tables in 2nd Hop (top 3)
world_1.**city**(id, name, country code, district, **population**))
world_1.**countrylanguage**(countrycode, **language**, is official, percentage)
city_record.city(city id, city, hanzi, hanyu pinyin, regional population, gdp)
Tabularized Questions
None
None
None

Table 10: Detailed case study comparing MURRE with baseline and CRUSH. The green means the relevant table, while the red means irrelevant. Each table is expressed in the form of "database name.tabel name(column name, column name, ...)". r denotes recall. Important entities in schema linking are **bold**.