Improving Demonstration Diversity by Human-Free Fusing for Text-to-SQL

Anonymous ACL submission

Abstract

In-context learning with large language models (LLMs) is the current mainstream method for text-to-SQL. Previous studies have explored selecting relevant demonstrations from a human-labeled demonstration pool, but these methods lack diversity and incur high labeling costs. In this work, we address measuring and enhancing the diversity of the text-to-SQL demonstration pool. First, we introduce a diversity metric and present that the diversity of the existing labeling data can be further enhanced. Motivated by these findings, we propose FUSED that iteratively fuses demonstrations to create a diverse demonstration pool based on human labeling or even from scratch with LLMs, reducing labeling costs. FUSED achieves an average improvement of 3.2% based on existing labeling and 5.0% from scratch on several mainstream datasets, demonstrating its effectiveness.[1]

1 Introduction

Text-to-SQL is a critical task that has garnered widespread attention for its ability to reduce the overhead of accessing databases by automatically generating SQL queries in response to user questions (Qin et al., 2022). Recently, in-context learning based on large language models (LLMs) has become the predominant method for this task, significantly improving performance while minimizing the need for fine-tuning (Chen et al., 2024; Qu et al., 2024a; Talaei et al., 2024). For the in-context learning paradigm, besides the user question and the database, the LLM is also provided with several demonstrations, guiding the model to generate the corresponding SQL queries accurately.

Currently, numerous works (Su et al., 2023; Ren et al., 2024; Pourreza et al., 2024) explore how to select question-relevant demonstrations from a human-labeled demonstration pool. However, relying entirely on human labeling limits the performance of text-to-SQL based on in-context learning due to two main issues: *(i) Low Diversity*: Human-labeled data could lack diversity since the data labeled by the same annotator could be somewhat similar (Ramalingam et al., 2021; Guo, 2023); *(ii) High Cost*: Human labeling requires significant labor overhead. To address these issues, thereby improving text-to-SQL performance, we discuss: ***(i) Theoretical metric for measuring the diversity of the demonstration pool*** (§2); ***(ii) Practical method that builds a diverse demonstration pool with existing labeling or even from scratch*** (§3).

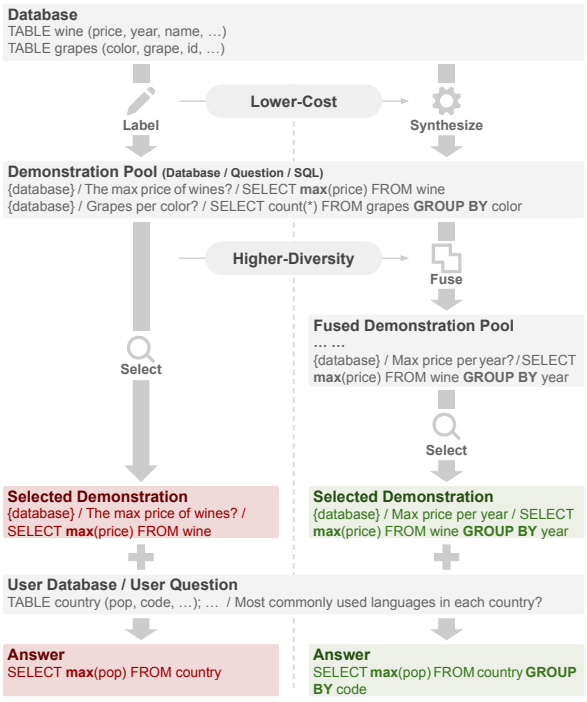

Figure 1: The comparison between the baseline (left) and FUSED (right) of obtaining the demonstration pool for text-to-SQL. FUSED can synthesize the demonstration pool from scratch or enhance the diversity of the existing labeling without additional human involvement.

First, we analyze that the diversity of the existing labeling can be further enhanced. We begin by discussing the necessity of demonstration pool diversity and present a diversity metric called **Di-**

[1] Our data and code will be released after review.

versity Measurement (DM). Using the metric, we prove that the existing labeling diversity can be further enhanced by showing that there exist demonstration pools with significantly higher DM.

Based on this analysis, we present our method called **FUS**ing it**E**ratively for **D**emonstrations (FUSED), which iteratively synthesizes the demonstrations using LLMs with existing labeling or from scratch, as shown in Figure 1. To tackle the *Low Diversity*, FUSED fuses demonstrations from previous iterations, ensuring that the new demonstrations are distinct from the previous, thus enhancing diversity. To address the *High Cost* of labeling, our method employs LLMs to generate demonstrations, thereby reducing the need for human labeling.

To validate the effectiveness of our method, we apply FUSED to several mainstream text-to-SQL datasets, including Spider (Yu et al., 2018) and KaggleDBQA (Lee et al., 2021). We synthesize demonstrations and compare performance with existing labeling and from scratch, where FUSED achieves an average performance improvement of 3.2% and 5.0%, respectively, confirming its effectiveness. Further analysis shows that FUSED significantly enhances DM of the existing labeling, demonstrating its capability to enhance the diversity of the existing demonstration pool.

Our contributions are as follows:

- We present **DM**, a metric to measure the diversity of a given demonstration pool for text-to-SQL, revealing that the diversity of the existing human-labeling data can be further enhanced.
- We propose **FUSED**, a method to build a high-diversity demonstration pool iteratively through human-free synthesis based on existing labeling data or even from scratch.
- We validate FUSED on multiple mainstream text-to-SQL datasets, achieving performance improvements of **3.2%** with existing labeling and **5.0%** from scratch, demonstrating its effectiveness.

2 Analysis

In this section, we present that **the diversity of the existing labeled demonstration pool can be further enhanced**. First, we discuss the necessity of high diversity for a demonstration pool. Then, we introduce a metric to quantify the demonstration pool diversity. Based on this metric, we discuss that the diversity of the existing labeling data can be further enhanced. We compare the metric present with the other existing metric in Appendix A.

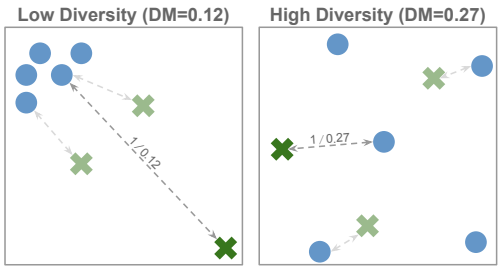

Figure 2: Two demonstration pools with different DM. ● represents the encoded demonstration, and ✖ represents the encoded user questions, in which the darkest denotes the user question with the least similarity to the most similar demonstration. The Euclidean distance between the user question and the most similar demonstration is indicated next to each line.

Necessity of the Diversity Regarding in-context learning, LLMs imitate the demonstration provided to generate the answer (Brown et al., 2020). Therefore, given a user question, previous works select the most similar demonstrations from a demonstration pool to guide the LLMs in generating answers (Luo et al., 2024). However, since user questions are unpredictable, the demonstration pool should be as diverse as possible to cover various user questions. The higher the diversity, the higher the similarity between any user questions and the demonstrations, thereby better guiding the answer generation; the lower the diversity, the more and more user questions are less similar to the demonstrations, decreasing the model performance.

Diversity Measurement Based on the preceding discussion, we employ the user question with the lowest similarity to the demonstration pool to measure the diversity of the demonstration pool. Formally, let $D = \{d_i\}$ represent the demonstration pool, $U = \{u\}$ denote the user questions, and $\texttt{sim}(u, d)$ as the similarity between u and d, calculated as the reciprocal of the Euclidean distance between their encoded vectors in this paper. We utilize Equation 1 to measure the diversity of the demonstration pool D, which is called **Diversity Measurement (DM)**. This metric corresponds to the similarity of the user question with the least similarity to the most similar demonstration in the demonstration pool, compared with any other user question. An illustration of the DM definition is shown in Figure 2. The detailed definitions of U, $\texttt{sim}$, and the calculation process of DM are discussed in Appendix B.

$$\text{DM} = \min_{u \in U} \max_{d_i \in D} \texttt{sim}(u, d_i) \quad (1)$$

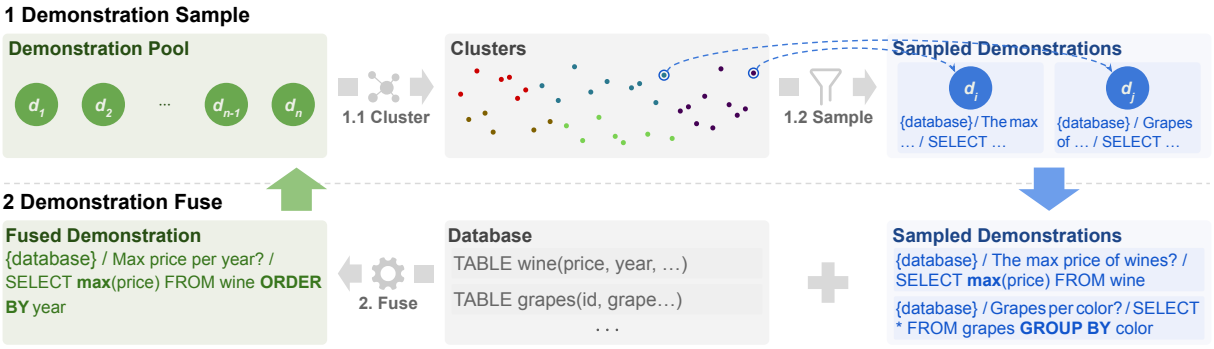

Figure 3: The pipeline of FUSED, which consists of two steps: ***(i)* Demonstration Sample**: Sample demonstrations to be fused from the demonstration pool; ***(ii)* Demonstration Fuse**: Fuse the sampled demonstrations with the randomly sampled database. The representation of `{database}` is discussed in Appendix C.

Diversity of the Existing Labeling Can be Further Enhanced With the metric present above, we then measure the diversity of the existing text-to-SQL labeling demonstration pool. The DM and performance of the existing labeling demonstration pool are depicted in Figure 4 and Figure 5. These figures reveal other demonstration pools where DM and performance are significantly higher than the existing labeling data. Thus, although the existing labeling exhibits relatively high diversity, it can be further improved, thereby enhancing the performance. Consequently, we next discuss the method for synthesizing demonstrations to enhance the diversity of the demonstration pool.

3 Method

Our method focuses on how to synthesize new demonstrations given databases with LLMs. Considering the poor diversity of directly generating demonstrations only relying on the sampling generation (Cegin et al., 2024), we present to synthesize by fusing different demonstrations iteratively, as shown in Figure 3. In each iteration, we guide the model to generate demonstrations that are not similar to the previous iterations, thereby enhancing the diversity. We theoretically prove that our method can enhance DM in Appendix D.

A simplified explanation of our method is that: we first cluster the demonstrations based on the SQL keywords (e.g., `WHERE`, `ORDER BY`). Then, we sample and fuse demonstrations from each cluster. The fused demonstration contains both `WHERE` and `ORDER BY` that are different from the sampled demonstrations, thereby enhancing the demonstration diversity. In practice, we use the encoded user questions rather than SQL keywords for synthesis since the user question has more semantic information than the SQL (Qin et al., 2022).

3.1 Overview

The fusion process of FUSED starts with an initial demonstration pool, which can be human-labeled or synthesized from scratch (see Appendix E). FUSED includes multiple iterations of fusion, where the synthesis of each iteration is based on the demonstration pool of the previous iteration. Each iteration consists of *demonstration sampling* (§3.2) and *demonstration fusing* (§3.3) two steps, which sample and fuse the demonstrations of the demonstration pool separately. The fused demonstrations of each iteration are then added to the demonstration pool, preparing for the next iteration.

After all iterations of fusion, we use the final demonstration pool for the text-to-SQL based on the in-context learning. We generate the SQL of each user question with LLMs directly following Chang and Fosler-Lussier (2023) since this is not the main topic of this paper.

3.2 Demonstration Sampling

This step is designed to sample the demonstrations to be fused, which consists of: *(i) Clustering* the demonstrations into multiple clusters; *(ii) Sampling* demonstrations from clusters to be fused.

3.2.1 Clustering

Before the fusion to get new demonstrations, it is required that the demonstrations sampled for fusing are not similar to ensure that the fused demonstration is not similar to the sampled demonstrations, thereby enhancing the diversity. The previous work (Zhang et al., 2023b) has shown that similar demonstrations are in the same cluster after encoding and then clustering. That is because the encoded vectors can reflect the semantics of the demonstrations, where the closer the vector distance, the more similar the semantics.

Inspired by this, we empirically employ an encoder model to encode the question of all demonstrations in the pool into vectors, and then use K-means to cluster encoded results into multiple clusters. Compared with not using the cluster, FUSED can ensure that the corresponding encoding vectors of the sampled demonstrations from different clusters are far away, leading to the demonstration used for fusion is not similar, enhancing diversity.

3.2.2 Sampling

After obtaining different clusters of the demonstration pool, we then sample demonstrations from different clusters for fusing. Considering that even in a single cluster, there also exist differences between the demonstrations, since the encoded vector can not accurately reflect the complete information of the demonstration (Morris et al., 2023). To enhance diversity, during the demonstration sampling, we randomly choose several distinct clusters, and then randomly sample demonstrations from each cluster separately, making the fused demonstration reflect the difference between different demonstrations.

3.3 Demonstration Fusing

We employ LLM to fuse demonstrations as the discussion in Appendix E, where we add the sampled demonstrations to guide the synthesis of the new demonstration as in-context learning, comparing with the randomly sampled database. Adding the sampled demonstrations comes up because LLMs imitate the demonstrations to generate results with the few-shot, whereas we let the LLM imitate both sampled demonstrations at the same time to get the fused demonstration. Thus, the fused demonstrations can reflect the attributes of and be different from all sampled demonstrations, thereby enhancing the diversity of the demonstration pool.

4 Experiments

4.1 Experiment Setup

Dataset We evaluate FUSED on two text-to-SQL datasets: Spider (Yu et al., 2018) and KaggleDBQA (Lee et al., 2021). Spider, a multi-domain text-to-SQL dataset, is one of the most widely used datasets currently. KaggleDBQA[2] is smaller in scale but involves more complex database and SQL structures, presenting higher hardness.

Metric Following previous works (Yu et al., 2018; Pourreza and Rafiei, 2023; Li et al., 2023), we employ execution match (EX) as our evaluation metric. EX measures the accuracy by comparing the execution results of the generated SQL on the database. There are two ways to evaluate EX: *(i)* directly using the predicted SQL conditional value (w. value); *(ii)* replacing the conditional value with that in the correct SQL (w/o. value).

Model In our experiments, we use SGPT-125m (Muennighoff, 2022) to encode demonstrations for clustering and use CodeLlama (Rozière et al., 2023) and GPT3.5[3] to synthesize demonstrations and convert user questions into SQLs. We apply FUSED to the Vanilla method, ACT-SQL (Zhang et al., 2023a) and ODIS (Chang and Fosler-Lussier, 2023), where the detail of these models and methods can be seen in Appendix F.

Implementation Details We study FUSED on two types of synthesis: from scratch (***w/o. Human***) and based on human labeling (***w. Human***). We synthesize 8 SQLs for each given database, set the generation temperature to 0.3, and synthesize in turns of 3 (*w/o. Human*) and 1 (*w. Human*) based on the analysis in § 4.4. About KaggleDBQA, we synthesize the demonstrations with both Spider and KaggleDBQA databases following the previous work (Chang and Fosler-Lussier, 2023). The size of demonstration pools of different settings is shown in Appendix G. We employ the 5-shot for text-to-SQL selected with BM-25 similarity, where the prompts for text-to-SQL are shown in Appendix C.

4.2 Main Result

The text-to-SQL performance is shown in Table 1, where FUSED brings 3.2% and 5.0% performance improvement on average with and without human-labeling across different settings, showing the effectiveness of our method. We further discuss the performance under different SQL hardness in Appendix H. From Table 1, we can also see that:

Model Scale Our method brings significant performance improvements on models of different scales. However, our method brings performance degradation with CodeLlama-7b, because of the low quality of the synthesized demonstrations due to the relatively poor performance of the 7b model, while ACT-SQL and ODIS are more sensitive to the demonstration quality since they employ the

[2]We call KaggleDBQA as Kaggle for simplicity.

[3]Document for GPT3.5.

Dataset	Method	Label	CodeLlama 7b w.	CodeLlama 7b w/o.	CodeLlama 13b w.	CodeLlama 13b w/o.	CodeLlama 34b w.	CodeLlama 34b w/o.	GPT3.5 - w.	GPT3.5 - w/o.	Δ - w.	Δ - w/o.
Spider	Vanilla	*w/o. Human*	48.5	59.8	54.9	67.6	56.9	72.2	57.9	74.9	+3.4	+3.4
		+ FUSED	54.4	66.4	58.8	70.9	59.7	75.1	58.7	75.8		
		w. Human	55.3	67.5	58.8	72.1	61.6	76.7	61.6	80.3	+1.6	+1.4
		+ FUSED	56.8	69.0	60.4	74.2	63.2	78.4	63.2	80.7		
	ACT-SQL†	*w. Human*	62.1	63.2	67.5	69.1	71.0	72.8	75.8	77.6	+0.7	+0.9
		+ FUSED	60.3	61.7	68.4	69.8	74.6	76.7	76.0	78.0		
	ODIS†	*w. Human*	58.2	71.8	61.9	76.6	64.3	80.9	63.9	81.1	+0.8	+0.9
		+ FUSED	58.0	71.0	62.9	78.0	65.6	82.1	64.8	83.0		
Kaggle	Vanilla	*w/o. Human*	9.9	18.0	13.2	23.5	13.2	23.2	14.0	25.4	+6.1	+7.2
		+ FUSED	22.8	32.0	19.1	29.0	18.0	30.1	14.7	27.6		
		w. Human	27.9	39.7	32.4	44.1	26.5	38.6	26.5	40.4	+5.4	+4.2
		+ FUSED	35.3	47.1	34.6	46.0	32.4	45.6	32.4	40.8		
	ACT-SQL†	*w. Human*	27.6	30.5	30.5	33.8	33.8	38.2	29.4	31.6	+0.4	+0.5
		+ FUSED	27.6	30.9	30.5	33.8	33.8	38.6	30.9	32.7		
	ODIS†	*w. Human*	33.8	43.4	34.6	47.1	31.6	46.3	34.6	48.9	+2.3	+3.0
		+ FUSED	35.7	47.1	36.0	48.5	35.3	50.4	36.8	51.5		

Table 1: The main experimental results on the Spider and KaggleDBQA dev sets. About the label setting, *w/o. Human* denotes synthesis from scratch using zero-shot and *w. Human* denotes synthesis based on human labeling with few-shot. About the metric, w. denotes with values and w/o. denotes without values. † denotes the reproduced results since the performance differences brought by the API version of GPT3.5. The improved results led by FUSED are marked green, the degradation is marked in red, and unchanged results are marked in black. The best results of different models and datasets are annotated in underline. **Δ** denotes the average improvement of different prompt methods leading by FUSED. We only adapt *w/o. Human* to the Vanilla method since ACT-SQL and ODIS cannot be adapted to the zero-shot inference without labeling data.

demonstrations to guide the intermediate generation rather than only for the few-shot. However, on KaggleDBQA, the performance does not increase as the model scale increases, because the demonstration pool used is synthesized or labeled with Spider databases (as described in § 4.1), which could mislead the generation for the KaggleDBQA.

Method Our method continues to improve performance based on all experiment methods under most settings, even improving performance based on ODIS and ACT-SQL such two well-performed baselines, proving the generalization and effectiveness of FUSED. Compared to the Vanilla method, our method shows relatively minor improvements with ACT-SQL and ODIS. This is because ACT-SQL and ODIS are more effective in helping the model understand the reasoning process within demonstrations, rather than merely imitating. This reduces the dependency on the similarity between demonstrations and user questions, making performance improvements less sensitive to the diversity of the demonstration pool compared to Vanilla.

Dataset FUSED brings significant performance improvements on all experimental datasets and even achieves results close to *w. Human* on Spider under the *w/o. Human* setting, demonstrating the effectiveness of our method under different domains. Besides, our method significantly improves KaggleDBQA more than Spider, showing that the demonstrations synthesized by FUSED are more effective for complex text-to-SQL questions.

4.3 Ablation Studies

To verify the effectiveness of the iteration and the cluster designed by FUSED, we perform ablation experiments on each part separately. The experimental results are shown in Table 2. Based on such results, we discuss the impact of different parts on the performance of our method.

4.3.1 Ablation of Iteration

To demonstrate that iterations work by improving the quality rather than quantity of the demonstrations, we conduct experiments that generate the same number of data as our method without iterations. From Table 2, we can see that: *(i)* There is a significant performance degradation after removing iteration, proving that FUSED enhances the performance by improving the demonstration quality rather than quantity; *(ii)* For larger-scale

Label	Spider			KaggleDBQA		
	7b	13b	34b	7b	13b	34b
FUSED *w/o. Human*	66.4	70.9	75.1	32.0	29.0	30.1
- Iteration	66.2(−0.2)	69.9(−1.0)	73.9(−1.2)	30.1(−1.9)	28.8(−0.2)	28.7(−1.4)
- Cluster	65.3(−1.1)	69.9(−1.0)	74.6(−0.5)	26.5(−5.5)	26.5(−2.5)	30.0(−0.1)
FUSED *w. Human*	69.0	74.2	78.4	47.1	46.0	45.6
- Iteration	67.6(−1.4)	71.9(−2.3)	76.6(−1.8)	38.6(−8.5)	44.1(−1.9)	38.6(−7.0)
- Cluster	67.7(−1.3)	70.5(−3.7)	75.4(−3.0)	41.2(−5.9)	40.4(−5.6)	35.7(−9.9)

Table 2: EX without values on CodeLlama ablating: *(i) Iteration*: synthesizing the same demonstration number of FUSED in one single turn; *(ii) Cluster*: randomly sampling demonstration to be fused without clustering.

models, iteration has a more significant impact on performance, indicating that larger-scale models can more effectively synthesize diverse demonstrations through multiple iterations; *(iii)* Compared with *w/o. Human*, FUSED under the *w. Human* setting has a more obvious decrease after removing iteration, because the quality of the synthesis without labeling data is lower than the labeling data, mixing which leads to a quality degradation compared with the original labeling data.

4.3.2 Ablation of Cluster

To demonstrate the effectiveness of the cluster, we perform ablation experiments on it. We compare our method with randomly selecting demonstrations during the demonstration sampling. From Table 2, we can find: *(i)* synthesis without clustering brings performance degradation in all settings, proving the effectiveness of the cluster; *(ii)* The performance degradation of KaggleDBQA is more obvious compared to Spider, indicating that the more complex text-to-SQL questions are more sensitive to the demonstration diversity.

4.4 Analysis

In this part, we discuss the impact of different parameters on the model performance. The analysis experimental settings are shown in Appendix I.

Can Diversity Measurement Reflect the Diversity of the Demonstration Pool? To prove that the metric DM we proposed can reflect the diversity of the demonstration pool, we randomly sample 20 demonstration pools, where each pool has 100 demonstrations from the Spider train set with different diversities. Then we use the Vanilla method to evaluate the performance of each pool on the Spider dev set. The experiment results are shown in Figure 4, from which we can see that: *(i)* With the same demonstration pool size, as DM enhances, the overall performance of the model is on the rise, indicating that the higher DM, the higher quality of the demonstration pool, denoting higher diversity; *(ii)* Most of the results are concentrated around 73.3, because such randomly sampled pools could not contain any demonstrations similar to the user questions, resulting in consistent performance.

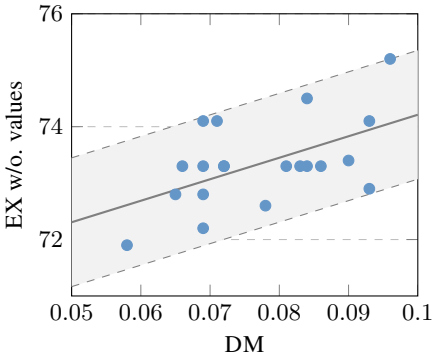

Figure 4: EX of 20 different demonstration pools with different DM on the Spider dev set. Different points denote different pools containing 100 demonstrations randomly sampled from the Spider train set.

Does the Diversity and Performance of Synthesized Data Continue to Rise with the Iteration Turn Increasing? To analyze the effectiveness of the iteration, we adapt experiments with different iterative turns, which are summarized in Figure 5. From the table, we can see that: *(i)* When the turn is ≤ 3 (*w/o. Human*) or ≤ 1 (*w. Human*), as the turn increases, DM and the performance of our method improves steadily, indicating that multiple iterations can enhance the diversity, thereby enhancing performance; *(ii)* When the turn is > 3 (*w/o. Human*) or > 1 (*w. Human*), with the number of turns increasing, diversity and performance improvement brought by FUSED becomes less and less, indicating the diversity can not be infinitely enhanced. Based on the above discussion, we use 3 and 1 as the synthesized turns.

How Does the Synthesized Scale Effect the Performance To verify the impact of different synthesized scales on performance, especially the performance under the small synthesized scale, we

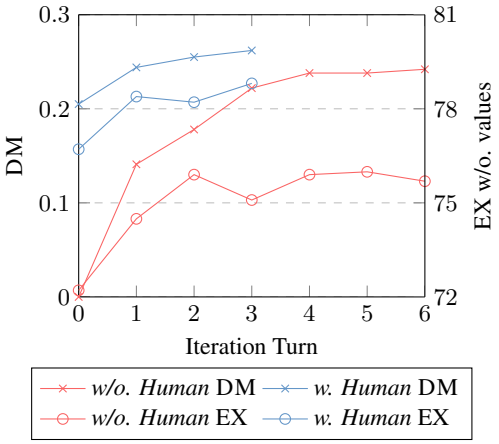

Figure 5: DM and EX without values on the Spider dev set of CodeLlama-34b across different iterations with FUSED. Turn 0 denotes the origin demonstration pool without FUSED. The sizes of the demonstration pools can be seen in Appendix G.

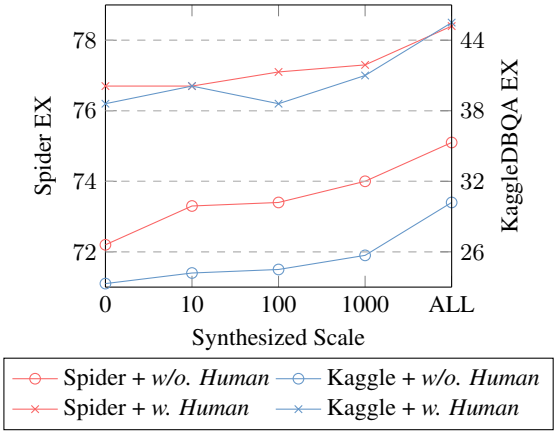

Figure 6: The EX without values of CodeLlama-34b with different synthesized scales. The X-axis denotes the number of demonstrations randomly sampled from the synthesized data, where ALL denotes 1947 and 10653 demonstrations under the *w/o. Human* and *w. Human* respectively. The Y-axis on the left and right are the results of Spider and KaggleDBQA respectively.

adapt experiments on synthesizing different demonstration numbers. The experiment results are shown in Figure 6, from which we can see that: *(i)* With the small synthesized scale ($\leq$ 100), FUSED can also improve the performance, proving the effectiveness under low synthesis overhead; *(ii)* With the synthesized scale increasing, the performance is continuously enhancing, indicating that the synthesized scale has a significant impact on performance.

How Does the Initial Labeling Scale Effect Our Synthesized Performance Although the main experiments of Table 1 demonstrate the effectiveness of our method on labeled data, the practical applications could lack labeled data with the same

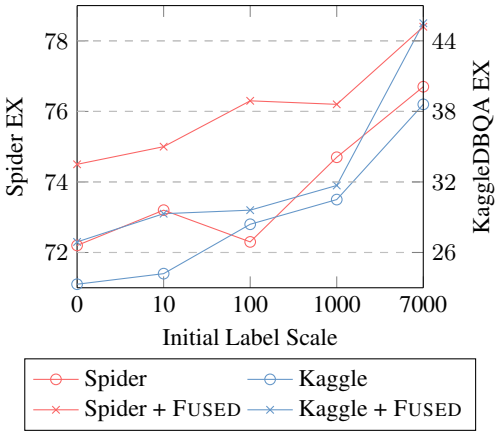

Figure 7: The EX without values of CodeLlama-34b under different initial human labeling scales sampled from the Spider train set. The X-axis represents the number of labeled demonstrations used for synthesis. The Y-axis on the left and right represent the results of Spider and KaggleDBQA respectively.

Database	7b	13b	34b
None	18.0	23.5	23.2
Kaggle	29.0	24.3	27.6
Kaggle + Spider	32.0	29.0	30.1

Table 3: EX without values of FUSED using CodeLlama evaluated on the KaggleDBQA dev set with the data synthesized based on the databases of different datasets under the *w/o. Human* setting. None denotes no synthesis data, Kaggle denotes synthesis only with the KaggleDBQA databases, and Kaggle + Spider denotes synthesis by mixing Spider databases.

scale as the Spider training data. Therefore, to validate the effectiveness of FUSED across varying scales of labeling, we randomly sample and conduct experiments on initial labeling demonstrations of different numbers from Spider training data.

The experiment results are shown in Figure 7, from which we can see that: *(i)* Under most settings, our method brings performance improvement, indicating its widespread effectiveness under different initial label scales; *(ii)* With the increase of the initial label scale, the performance demonstrates a consistent increase, suggesting that expanding the labeling scale can reliably enhance performance.

Can FUSED Effectively Help LLMs Migrate to the Domain without Labeling? In this part, we evaluate that FUSED can improve the text-to-SQL performance across different domains without human labeling. The experimental results are shown in Table 3. From the table, we can see that: *(i)* Compared with not synthesizing demonstrations, FUSED can bring performance improvements when

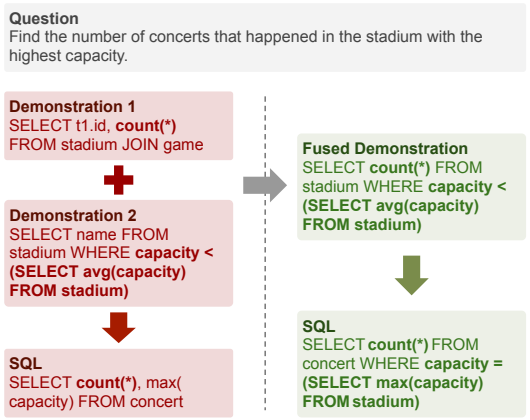

Figure 8: The case study of demonstrations by human-labeling (left) and FUSED (right) from Spider. The corresponding SQL keywords between demonstrations and the answer are annotated in **bold**.

only using KaggleDBQA databases, proving the effectiveness of our method adapted to a new domain without labeling; *(ii)* Compared to using only KaggleDBQA databases, the demonstrations obtained by mixing Spider databases can bring greater performance improvements, indicating that increasing the diversity of databases can also enhance the diversity of synthesized demonstrations.

4.5 Case Study

Although the above analysis proves the effectiveness of FUSED, how our method improves the performance of the text-to-SQL using in-context learning remains to be discovered. To analyze how our method improves the model performance more specifically, in this part, we conduct a case study. A comparison between results based on labeled data and the demonstrations obtained using FUSED is shown in Figure 8. From the figure, we can see that the results using only labeled data do not combine the SQL keywords of the two demonstrations well. The demonstration obtained with our method, on the other hand, has already combined the SQL keywords of the two demonstrations, which guides the model to successfully generate the correct SQL.

5 Related Works

5.1 Text-to-SQL

Text-to-SQL is a vital task that generates SQL based on the user question and the provided databases. Recent research shows that text-to-SQL based on LLMs can approach or exceed the performance of fine-tuned models without fine-tuning, which greatly advances research on this task while reducing labeling overhead (Chang and Fosler-Lussier, 2023; Zhang et al., 2023a; Li and Xie, 2024). For example, DIN-SQL (Pourreza and Rafiei, 2023) decomposes the text-to-SQL task into multiple sub-tasks. DAIL-SQL (Gao et al., 2023) evaluates different prompt formats to find the best combination. MCS-SQL (Lee et al., 2024) consistency the results generated with multiple prompts.

However, existing LLM-based methods entirely rely on human-labeled demonstrations, demanding high labeling costs be adapted to a new domain. Therefore, we propose FUSED to synthesize text-to-SQL demonstrations based on LLMs using provided domain databases without human labeling, effectively reducing the labor cost.

5.2 In-Context Learning

In-context learning is an effective method to enhance the reasoning ability of LLMs by providing several demonstrations to guide reasoning (Xun et al., 2017; Wei et al., 2022). Some works propose to automatically select relevant demonstrations for each user question to improve the performance of LLMs (Zhang et al., 2023b; Shum et al., 2023; Qu et al., 2024b). Another kind of work enhances in-context learning by synthesizing relevant data by supervised fine-tuning (Wang et al., 2023; Yang et al., 2024; Sun et al., 2023).

However, existing methods only demonstrate that increasing the diversity of the demonstrations can enhance performance but do not discuss if the diversity of the existing labeling data is sufficient, and how to increase the diversity of the demonstrations (Su et al., 2023; Levy et al., 2023). Therefore, we present DM to show that the existing labeling data of the text-to-SQL is not diverse enough and propose FUSED to enhance the diversity.

6 Conclusion

In this paper, we improve the performance of the text-to-SQL task using in-context learning from the perspectives of measuring and enhancing the demonstration pool diversity. We first present DM to measure the diversity of the demonstration pool, based on which we present that the diversity of the existing labeling data can be further enhanced. Based on the above analysis, we present FUSED, which synthesizes demonstrations using LLMs, lowering the labeling cost. Experiments show that FUSED brings an average improvement of 3.2% and 5.0% with and without labeling data on Spider and KaggleDBQA, proving the effectiveness.

Limitations

FUSED has two limitations, including: *(i)* About the encoding of the demonstration sample step, directly splice the user question and the SQL could not fully reflect the attributes of them. In future work, we will try to encode the question and SQL according to the attributes separately; *(ii)* For the synthesized demonstration pool, we only enhance the diversity, while ignoring the effect of the scale on the demonstration selection. Our future work will filter the synthesis, reducing the scale of synthesis under the premise of ensuring diversity.

Ethics Statement

All datasets and models used in this paper are publicly available, and our usage follows their licenses and terms.

References

Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ján Cegin, Branislav Pecher, Jakub Simko, Ivan Srba, Mária Bieliková, and Peter Brusilovsky. 2024. Effects of diversity incentives on sample diversity and downstream model performance in llm-based text augmentation. *ArXiv*, abs/2401.06643.

Shuaichen Chang and Eric Fosler-Lussier. 2023. Selective demonstrations for cross-domain text-to-SQL. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14174–14189, Singapore. Association for Computational Linguistics.

Xiaojun Chen, Tianle Wang, Tianhao Qiu, Jianbin Qin, and Min Yang. 2024. Open-sql framework: Enhancing text-to-sql on open-source large language models. *Preprint*, arXiv:2405.06674.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

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

Tonglei Guo. 2023. The re-label method for data-centric machine learning. *ArXiv*, abs/2302.04391.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. 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 2261–2273, Online. Association for Computational Linguistics.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *Preprint*, arXiv:2405.07467.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422, Toronto, Canada. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Chenhao Ma, Kevin C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *ArXiv*, abs/2305.03111.

Zhenwen Li and Tao Xie. 2024. Using llm to select the right sql query from candidates. *ArXiv*, abs/2401.02115.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *ArXiv*, abs/2401.11624.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *Preprint*, arXiv:2202.08904.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir R. Radev. 2023. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *ArXiv*, abs/2305.12586.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohammadreza Pourreza, Davood Rafiei, Yuxi Feng, Raymond Li, Zhenan Fan, and Weiwei Zhang. 2024. Sql-encoder: Improving nl2sql in-context learning through a context-aware encoder. *Preprint*, arXiv:2403.16204.

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.

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024a. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-sql generation. *Preprint*, arXiv:2405.15307.

Xingwei Qu, Yiming Liang, Yucheng Wang, Tianyu Zheng, Tommy Yue, Lei Ma, Stephen W. Huang, Jiajun Zhang, Wenhu Chen, Chenghua Lin, Jie Fu, and Ge Zhang. 2024b. Deep-icl: Definition-enriched experts for language model in-context learning. *Preprint*, arXiv:2403.04233.

Srikumar Ramalingam, Daniel Glasner, Kaushal Patel, Ravi Vemulapalli, Sadeep Jayasumana, and Sanjiv Kumar. 2021. Less is more: Selecting informative and diverse subsets with balancing constraints. *ArXiv*, abs/2104.12835.

Tonghui Ren, Yuankai Fan, Zhenying He, Ren Huang, Jiaqi Dai, Can Huang, Yinan Jing, Kai Zhang, Yifan Yang, and X. Sean Wang. 2024. Purple: Making a large language model a better sql writer. *Preprint*, arXiv:2403.20014.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950.

Kashun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12113–12139, Singapore. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Daniel Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *Preprint*, arXiv:2405.16755.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Guangxu Xun, Xiaowei Jia, Vishrawas Gopalakrishnan, and Aidong Zhang. 2017. A survey on context learning. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):38–56.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning. *Preprint*, arXiv:2402.13669.

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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023a. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Comparison with Other Diversity Metrics

To better explore the progress of the diversity metric we proposed in §2, we compare it with the past metric present by Nan et al. (2023): *(i)* Nan et al. (2023) mainly focus on selecting demonstrations, while the motivation of ours is to synthesize demonstrations; *(ii)* Nan et al. (2023) does not give a numerical measure of diversity, while our method gives a numerical measure of diversity; *(iii)* Nan et al. (2023) is based on clustering, and the granularity of judging diversity is relatively coarse, while our method is based on the entire demonstration pool, which can more accurately measure the diversity of demonstrations.

B How to Calculate Diversity Measurement

$$\begin{aligned} &\min_{u \in U} \max_{d_i \in D} \texttt{sim}(u, d_i) \\ &:= \min_{u \in \texttt{Convex}(D)} \max_{d_i \in D} |u - d_i|^{-1} \end{aligned} \tag{2}$$

As the discussion in §2, given a demonstration pool, the calculation process of DM can be formalized as Equation 2, where $\texttt{sim}(u, d) = |u - d|^{-1}$ is the reciprocal of the Euclidean distance between the encoded vectors of u and d, and $\texttt{Convex}(D)$ denotes the convex hull of the demonstrations.

We use the Euclidean distance to represent sim since the closer the distance between the embedding question and the embedding demonstration, the more similarity between the question and the demonstration. The user question u should be in the area surrounded by $\texttt{Convex}(D)$ corresponds to the question-related domain, and the user questions are highly related to the domain and have a high probability of locating in the convex.

We use SciPy (Virtanen et al., 2020) to solve Equation 2, and use SGPT-125m (Muennighoff, 2022) to encode demonstrations. We first generate the Voronoi diagram (Aurenhammer, 1991) and compute the convex hull for the encoded demonstration points. For each point, we then calculate the maximum distance to any vertex in its corresponding Voronoi region confined within the convex hull and use the greatest of these maximum distances as the result.

C Text-to-SQL Prompts

The prompts of the SQL generation and the question generation are shown in Table 4 and Table 5, where the formats of {database} and {demonstration} are same as Chang and Fosler-Lussier (2023).

D Why FUSED can Enhance the Diversity Measurement

In this section, we explain why the demonstration sampling (§3) in FUSED can enhance DM. To increase Equation 1, it is required to maximum $\min_{u \in U} \max_{d_i \in D} \texttt{sim}(u, d_i)$. Let $u^* = \text{argmin}_{u \in U} \max_{d_i \in D} \texttt{sim}(u, d_i)$, then we aim to update D to make $\max_{d_i \in D} \texttt{sim}(u^*, d_i)$ as large as possible.

We define that the cluster corresponding to d_i is c_{d_i}, and let $\texttt{sim}(u, c_i) = \max_{d \in c_i} \texttt{sim}(u, d)$. We denote d_i that maximum $\texttt{sim}(u^*, d_i)$ as d^*. Then we have $\max_{d_i \in D} \texttt{sim}(u^*, d_i) = \texttt{sim}(u^*, c_{d^\dagger}) = |u^* - c_{d^*}|^{-1} > (|u^* - c| + |c - c_{d^*}|)^{-1}$, where c is any cluster. The above inequality holds because $c_{d^\dagger}, c, u^*$ can be considered as the vertices of a triangle, and the sum of the lengths of two sides is greater than the length of the third side.

According to the discussion in Appendix B, it is hard to precisely find u^*, so we maximize the right-hand side of the inequality as much as possible to increase $\texttt{sim}(u^*, c_{d^*})$. Therefore, as described in §3.2, the demonstration sampling continuously combines demonstrations to generate new demonstrations between different clusters, thereby reducing the distance between different clusters. During the sampling, adding new results can also decrease the distance between u^* and c, so the right-hand side of the inequality is continuously decreasing. In summary, FUSED can continuously increase $\max_{d_i \in D} \texttt{sim}(u^*, d_i)$, thus increasing DM of the results.

E Synthesize Text-to-SQL Demonstrations with LLMs

In this section, we discuss how to employ LLMs to obtain the initial demonstration pool with the given database, lowering the labeling cost. The prompts we used are shown in Appendix C.

SQL Synthesize Following the previous work (Chang and Fosler-Lussier, 2023), we synthesize SQL based on the linearized schema of the given database with LLMs. During synthesis, we ask LLMs to generate multiple SQLs for each database to enhance the diversity of the results with the sampling generation. The prompt we used is shown in Table 4.

SQL Synthesize
Synthesize one SQL query for the given database.

{database}
– Synthesize a new single SQL for the above database imitating {SQL1} and {SQL2}.
SELECT

Table 4: The prompt for the SQL synthesis.

Question Synthesize
Using natural language, generate a question corresponding to the given SQL.
Different examples are separated with '\n\n'.

{demonstration1}

...

{demonstration5}

{database}
– Using natural language, generate a question corresponding to the given SQL: {SQL}.
Question:

Table 5: The prompt for the question synthesis.

Question Synthesize We synthesize the corresponding questions of the generated SQL with the linearized schema of the. We first synthesize SQL instead of questions because LLMs could generate questions that are hard to answer using SQL (Cheng et al., 2023), and it is harder to validate the semantic consistency between the SQL and the question for generating questions first. The prompt of this step is shown in Table 5.

Validate Due to the limitation of the model performance, it is hard to guarantee that the semantics of all synthesized SQL-question pairs are completely consistent, resulting in a decrease in the quality of the synthesized demonstration. To improve the quality of the synthesized results, we verify the semantic consistency between the synthesized questions and SQL. We generate SQL based on the question and then evaluate if the generated SQL is the same as the synthesized SQL, for which we use LLMs to reduce the cost of fine-tuning. The prompts for text-to-SQL follow Chang and Fosler-Lussier (2023).

F Baselines

F.1 Baseline Models

CodeLlama CodeLlama is a model based on Llama2 (Touvron et al., 2023), which is fine-tuned on a large amount of code data and can better solve code-related problems (including SQL).

GPT3.5 GPT3.5 is an improved model based on GPT3 (Brown et al., 2020), which further enhances performance through additional task-specific fine-tuning. We use Azure OpenAI API of `gpt-3.5-turbo` of GPT3.5 for our experiments [4].

F.2 Baseline Methods

Vanilla Following the previous work (Chang and Fosler-Lussier, 2023), we design the Vanilla method that directly employs the few-shot to generate the answer, where the demonstrations are selected by the BM-25 similarity between the user question and the demonstration questions.

ACT-SQL ACT-SQL (Zhang et al., 2023a) is a method to construct the chain-of-thought rationales based on SQL automatically. This method synthesizes reasoning steps with table names, column names, and values used in the SQL.

ODIS ODIS (Chang and Fosler-Lussier, 2023) is an automatic demonstration selection method designed for the text-to-SQL task. This method selects out-domain demonstrations from the labeled data and synthesizes in-domain demonstrations based on the databases related to the user question.

G Number of Synthesized Data

The synthesized demonstrations under different settings are shown in Table 6. From the table, we can see that `gpt-3.5-turbo` has less data than that synthesized by CodeLlama, because the SQL synthesized by `gpt-3.5-turbo` is more complex, which makes it more difficult to pass the filter.

To find the best turn number of synthesis, we synthesize more turns on CodeLlama-34b, and the size of synthetic data is shown in Table 7.

H FUSED Performance under Different SQL Hardness

To analyze the effectiveness of FUSED on questions with different complexity, we evaluate our method on SQL categorized by different hardness. The category criteria follows Yu et al. (2018). The experimental results are shown in Table 8.

From the table, we can see that: *(i)* On most hardness, our method can bring significant performance improvements, which proves the effectiveness of

[4] https://azure.microsoft.com/en-us/products/cognitive-services/openai-service

Model	Label	Turn 0	1	2	3	Total
CodeLlama-7b	*w/o. Human*	0	584	561	608	1753
	w. Human	7000	1937	–	–	8937
CodeLlama-13b	*w/o. Human*	0	954	741	803	2489
	w. Human	7000	3001	–	–	10001
CodeLlama-34b	*w/o. Human*	0	457	668	822	1947
	w. Human	7000	3653	–	–	10653
gpt-3.5-turbo	*w/o. Human*	0	803	643	502	1948
	w. Human	7000	387	–	–	7387

Table 6: Synthesized size under different settings.

Model	Label	Turn 0	1	2	3	4	5	6
CodeLlama-34b	*w/o. Human*	0	457	668	822	854	981	897
	w. Human	7000	3653	4243	4344	–	–	–

Table 7: More synthesized size of CodeLlama-34b for Figure 5.

Dataset	Label	Easy	Medium	Hard	Extra
Spider	*w/o. Human*	**88.7**	80.7	57.5	40.4
	+ FUSED	87.5	**81.2**	**63.8**	**52.4**
Kaggle	*w/o. Human*	53.1	30.3	5.1	1.9
	+ FUSED	**59.4**	**32.9**	**11.4**	**1.9**

Table 8: EX without values of CodeLlama-34b under different SQL hardness with and without FUSED. The best result of each setting is annotated in **bold**.

Template (%)
SELECT * FROM * WHERE * <op> * (25.7)
SELECT * FROM * WHERE * <op> * AND * <op> * (13.9)
SELECT * FROM * JOIN * JOIN * WHERE * <op> * (5.2)
SELECT * FROM * JOIN * WHERE * <op> * (4.9)
SELECT * FROM * WHERE * IN (SELECT * FROM * WHERE * <op> *) (4.3)

Table 9: Top five SQL templates synthesized by FUSED using CodeLlama-34b. The numbers in the brackets denote the proportion of each template.

FUSED; *(ii)* On Spider, the more difficult SQL, the more significant the improvement, showing that synthesized demonstrations can more effectively guide complex SQL generation; *(iii)* For the easy questions of Spider, our method brings a slight performance degradation because the model already performs well under the *w/o. Human* setting for this hardness, and the additional demonstrations could mislead the model; *(iv)* On the extra questions of KaggleDBQA, our method does not bring performance improvement, which could be because it is too hard to synthesize too complex demonstrations (harder than Spider extra questions), resulting in the selected demonstrations being unable to effectively guide the generation of the extra hardness.

I Settings of Analysis Experiments

We adapt analysis experiments under the setting of:

CodeLlama-34b CodeLlama is one of the most mainstream code generation models at present, which achieves near the performance of the closed-source model (as shown in Table 1) in the open-source model with less inference cost (no need to call API), of which CodeLlama-34b is the best performance in this series of models.

Evaluating without values Regarding the text-to-SQL task, current research mainly focuses on how to generate SQL with the correct structure, while paying less attention to extracting the condition values exactly, since this requires the memorizing ability rather than the semantic parsing ability.

J Synthesized Template

To guide future works in generating more diverse demonstrations, in this part, we analyze the proportion of demonstrations with different SQL templates synthesized by our method. We replace table names, column names, and values with * and operators with <op> as the templates corresponding to each SQL. Our method synthesizes 175 different SQL templates, showing the diversity of the synthesized demonstrations. The five most frequent template types are shown in Table 9.

From the table, we can find: *(i)* The current model is most inclined to generate `SELECT` and `WHERE`, which is nearly 40%, indicating that such types of SQL occur more frequently in the pre-training data of LLMs we use and, thereby, are more frequently used in real-world scenarios; *(ii)* Existing models hardly generate complex SQL that contains nested SQL (less than 5% of synthetic data), indicating that future methods should specifically pay attention to guide the model to generate results that contain two or more sub-SQLs or even more complex structures.