DeTriever: Decoder-representation-based Retriever for Improving NL2SQL In-Context Learning

Raymond Li[1]* , **Yuxi Feng**[1]* , **Zhenan Fan**[2]† , **Giuseppe Carenini**[1],
Weiwei Zhang[2], **Mohammadreza Pourreza**[3], **Yong Zhang**[2],

[1]The University of British Columbia, Vancouver, Canada
[2]Huawei Technologies Canada Co., Ltd., Burnaby, Canada
[3]University of Alberta, Edmonton, Canada

Correspondence: zhenan.fan1@huawei.com

Abstract

While in-context Learning (ICL) has proven to be an effective technique to improve the performance of Large Language Models (LLMs) in a variety of complex tasks, notably in translating natural language questions into Structured Query Language (NL2SQL), the question of how to select the most beneficial demonstration examples remains an open research problem. While prior works often adapted off-the-shelf encoders to retrieve examples dynamically, an inherent discrepancy exists in the representational capacities between the external retrievers and the LLMs. Further, optimizing the selection of examples is a non-trivial task, since there are no straightforward methods to assess the relative benefits of examples without performing pairwise inference. To address these shortcomings, we propose *DeTriever*, a novel demonstration retrieval framework that learns a weighted combination of LLM hidden states, where rich semantic information is encoded. To train the model, we propose a proxy score that estimates the relative benefits of examples based on the similarities between output queries. Experiments on two popular NL2SQL benchmarks demonstrate that our method significantly outperforms the state-of-the-art baselines for the NL2SQL tasks.

1 Introduction

As large language models (LLMs) have become increasingly capable of solving a wide variety of complex tasks (Brown et al., 2020; Touvron et al., 2023a,b; Rozière et al., 2024), they have become an integral component in many user-facing applications. In particular, the task of translating natural language questions into structured query language (NL2SQL) in the context of existing databases, has received a lot of attention due to its relevance in enhancing business intelligence tools and offering significant potential to make data analytics and exploration more accessible to non-technical users. While the recent development of LLMs pre-trained in code completion tasks (Rozière et al., 2024) has offered a significant boost in NL2SQL capabilities, the current performance of these models still falls short of the standards required for deployment in production environments. This is mainly due to the many challenges including dealing with domain-specific jargon along with constructing complex queries, which requires an advanced understanding of relationships between database schemas and the underlying intentions of natural language questions (Deng et al., 2022; Swamidorai et al., 2023).

Research on LLMs has unveiled the emergent abilities (Wei et al., 2022a) of models such as GPT3 (Brown et al., 2020) and Llama (Touvron et al., 2023a,b) to improve task performance through the use of in-context learning (ICL), where demonstration examples are integrated into the input prompt during inference to allow the model to make use of aspects from the demonstrations (Min et al., 2022). As proven in a variety of complex tasks such as mathematical reasoning (Wei et al., 2022b), ICL has become a cost-efficient alternative to supervised fine-tuning which requires significant computational resources to update the model parameters. In the domain of NL2SQL, recent studies (Pourreza and Rafiei, 2023; Gao et al., 2023) have adopted ICL to achieve state-of-the-art performance on popular NL2SQL benchmarks including Spider (Yu et al., 2018) and BIRD (Li et al., 2024). This is done by prepending labeled examples to the task prompt, allowing the model to draw an analogy from the mapping between the problem description and the corresponding query. Figure 1 illustrates the overview of model inference with ICL demonstration examples on the NL2SQL task. In our setup, demonstration triples consisting of a database (DB) schema, question, answer are prepended to the NL2SQL question.

* Equal Contributions.
† Corresponding Author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 8173–8183
January 19–24, 2025.

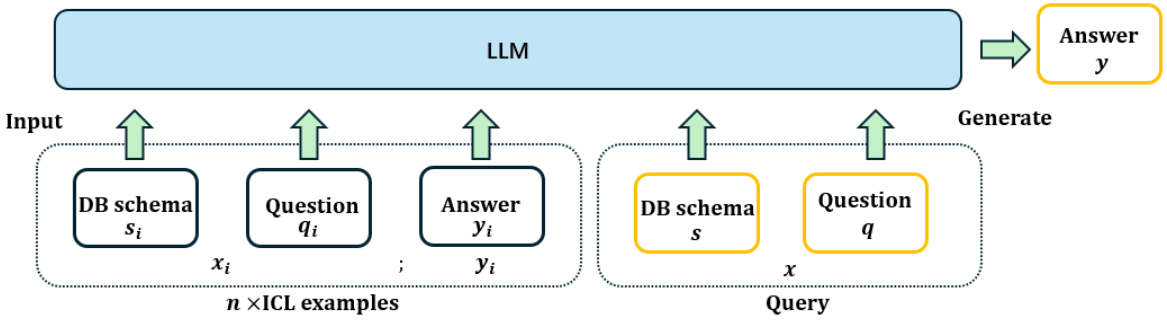

Figure 1: Examples of in-context learning (ICL) for NL2SQL.

Since model performance can vary widely depending on the choice of in-context examples (Liu et al., 2022), a key challenge is determining the most effective strategy to select demonstration examples for any given task. While prior studies have examined the usefulness of demonstrations based on characteristics of the examples (Wei et al., 2022a), a more effective strategy is to dynamically retrieve demonstrations for a given example during inference. Following recent studies on informational retrieval (IR), many existing proposals (Das et al., 2021; Khattab et al., 2022) have utilized off-the-shelf encoder models such as Sentence-BERT (Reimers and Gurevych, 2019) and Contriver (Izacard et al., 2021) to retrieve ICL examples based on the embeddings similarity between the task input prompts, while more recent studies (Rubin et al., 2022; Shi et al., 2023) have opted to fine-tune the retriever based on the LLM perplexity to predict the target output. However, this fine-tuning strategy cannot be trivially adapted for code completion tasks such as NL2SQL since there is an arbitrary number of correct predictions (i.e., multiple different queries retrieving the same records) for each input question. In addition, while it might be tempting to use query execution results as a target for fine-tuning the retriever, the binary discrete signal of inference accuracy cannot effectively differentiate the relative benefits between individual ICL examples. Furthermore, there exists a fundamental gap between the retrieval encoders and the LLM due to the inherently more sophisticated representational capabilities of the LLM to differentiate the intricate nuances embedded within the task prompt. Lastly, as evident by prior studies on probing in measuring the differences between information captured at different locations of the model (Chuang et al., 2023), it remains a non-trivial task to effectively leverage LLM hidden states as representations for a given prompt.

To overcome the above-mentioned challenges, we propose *DeTriever*, a novel ICL retrieval framework that learns a weighted combination of LLM layer transformations to dynamically retrieve examples based on similarities. To train this model, we first perform an in-depth study to determine the proxy metric that best approximates the relative benefits of demonstration examples. Based on the findings from our analysis, we convert the proxy metric as the target for the contrastive loss used to fine-tune *DeTriever*. Our experiments on two popular benchmarks demonstrated the effectiveness of our approach where we achieved significant improvements over state-of-the-art baselines. The contributions of our work can be summarized as the following:

- We propose *DeTriever*, a lightweight retriever that learns a weighted combination of LLM's layer transformations as representation for retrieval demonstration examples.
- We train *DeTriever* by selecting a proxy metric for measuring the relative benefit between examples for ICL and using it as the target for supervised contrastive loss.
- We perform comprehensive experiments on two popular benchmarks to demonstrate the effectiveness of our approach, where *DeTriever* significantly outperforms state-of-the-art baselines for NL2SQL.
- We conduct a detailed analysis of our approach and reveal important insights for developing future methods to retrieve demonstration examples for practical applications.

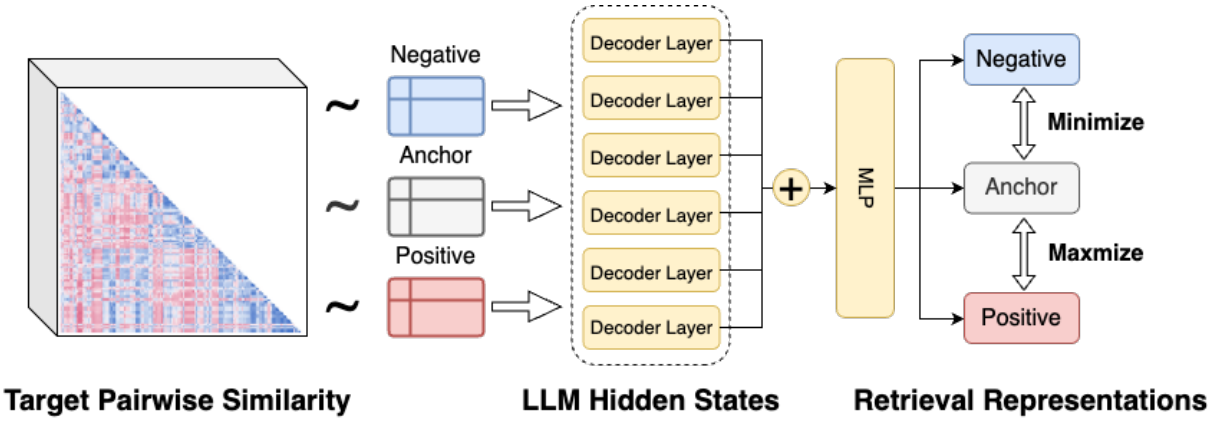

Figure 2: Overview of the training process of our proposed *DeTriever* method.

2 Related Work

NL2SQL Traditionally, NL2SQL tasks are solved by sequence-to-sequence encoder-decoder models (Choi et al., 2021). Recently, large language models (LLMs) have impressively enhanced sample efficiency and performance across numerous natural language processing (NLP) tasks. A notable area of improvement is in translating natural language questions into SQL (NL2SQL) queries. In this domain, models leveraging in-context learning, which involves providing a few example questions and SQL pairs in the prompt, have achieved state-of-the-art performance (Pourreza et al., 2024; Pourreza and Rafiei, 2023; Gao et al., 2023) on the largest existing comprehensive and cross-domain NL2SQL benchmarks: Spider(Yu et al., 2018) and BIRD(Li et al., 2024).

LLMs and Retrieval-augmented ICL Recently large language models (LLMs) like GPT4 (OpenAI, 2023), Llama-2 (Touvron et al., 2023b), and Code Llama(Rozière et al., 2024) have raised people's attention with their emergent ability (Wei et al., 2022a) to learn from a few examples in the context, which is so-called in-context learning (ICL). The key idea of in-context learning is to learn from analogy to examples in a given prompt. Different from supervised learning requiring a training stage that uses backward gradients to update model parameters, ICL does not conduct parameter updates and directly performs predictions on the pre-trained language models. Few-shot ICL of LLMs have been proven to achieve comparable performance with supervised fine-tuning on smaller models on numerous tasks like mathematical reasoning (Wei et al., 2022b). One key question is how to retrieve the best examples to input as a prompt to LLMs for a given problem.

Related work of in-context retrieval can be organized into two categories. (1) Use an external encoder to retrieve examples, e.g., Sentence-BERT (S-BERT) (Reimers and Gurevych, 2019) or Dense Passage Retriever (DPR) (Karpukhin et al., 2020). (2) Retrieval-augmented language models (Izacard et al., 2021, 2023; Borgeaud et al., 2022; Ma et al., 2023) These works are based on the encoder-decoder transformer architectures, where the model representation is trained to retrieve examples based on its hidden representations. As a result, applying such an approach requires further fine-tuning to recover from the modified representation, and cannot be directly applied for decoder-only architectures.

Recently people have started to leverage LLM's own hidden information to compute text embedding. Shi et al. (2023) treat LLM as a black box to retrieve relevant documents from an external corpus using an off-the-shelf retrieval model, where retrieved documents are prepended to the input context and fed into the black-box LLM. Wang et al. (2024) proposes to use the last hidden layer of the end-of-sentence (EOS) token as the sentence embedding for retrieving ICL examples.

Unlike all previously mentioned methods which either use an external encoder-based retriever or simply use the last hidden layer of the EOS token, we propose to use the internal hidden states of LLMs to retrieve in-context examples.

3 Approach

3.1 Problem Formalization

Given a database schema s and the natural language question q, the goal of the NL2SQL task is to generate the SQL query y that correctly retrieves

database records to answer the question according to the given schema. We leverage a large language model M for the query generation task, where the database schema and question are formatted into an input prompt $x = [s; q]$ such that query prediction $\hat{y} = M(x)$. For ICL inference, we prepend demonstration examples to the input prompt such as $\hat{y} = M([x_1, y_1; \ldots; x_n, y_n; x])$, where question-answer pairs (x_i, y_i) are sampled from a labeled training set $\mathcal{D}_{\text{train}}$. For demonstration retrieval, we first apply a retrieval model $\mathcal{R}$ to compute the retrieval representation for the input of all examples in $\mathcal{D}_{\text{train}}$. During inference for problem description x, we select an ICL example $I(x)$ based on the highest dot-product similarity between the retrieval model representations.

$$I(x) = (x_i, y_i) = \underset{(x_i,y_i)\in\mathcal{D}_{\text{train}}}{\text{argmax}} \; \mathcal{R}(x) \cdot \mathcal{R}(x_i) \quad (1)$$

Since the probability of predicting the correct query can be measured through execution accuracy by comparing the predicted query $\hat{y} = M([I(x); x])$ against the ground-truth query y over an evaluation corpus $\mathcal{D}_{\text{test}}$, where $\text{Acc}(x, \hat{y}, y) \in \{0, 1\}$, the objective of optimizing the retriever $\mathcal{R}$ can be expressed in Equation 2.

$$\underset{\mathcal{R}}{\text{argmax}} \sum_{(x_j,y_j)\in\mathcal{D}_{\text{test}}} \text{Acc}(x_j, M([I(x_j); x_j]), y_j) \quad \text{s.t. } I(x_j) \in \mathcal{D}_{\text{train}} \quad (2)$$

3.2 LLM Hidden States

Although prior studies have often employed external encoders, such as Sentence-BERT (Reimers and Gurevych, 2019) and DPR (Karpukhin et al., 2020), to retrieve in-context exemplars for the LLM, there is a fundamental gap between these models and the LLM. This is because the LLM has more advanced abilities to capture semantic meanings and encode the subtle nuances of the input. Therefore, we directly use the hidden states of the LLM as input embeddings to the retriever. However, unlike encoder models such as BERT (Devlin et al., 2019), where the hidden states of the last layer can be directly used as representations of the sequence, there are no straightforward methods of adapting an autoregressive LLM for encoding the input problem for retrieval. While it may be tempting to simply use the last layer as the problem embedding Wang et al. (2024), hidden states of lower layers can have better representation in the semantic meaning of the tokens (Chuang et al., 2023) which may lead to improvements in retrieval qualities.

To test this hypothesis, we perform an analysis on the performance of each layer for retrieving in-context examples. Specifically, for layer ℓ of the LLM, we use the mean-pooled or end-of-sequence (EOS) token hidden state of the problem prompt $x = [s; q]$ to retrieve the most similar examples in the training set $\mathcal{D}_{\text{train}}$ based on cosine similarity of the ℓ-th layer hidden-states M_ℓ of the large language model. (Equation 3).

$$\mathcal{I}(x) = \underset{(x_i,y_i)\in\mathcal{D}_{\text{train}}}{\text{argmax}} \; M_\ell(x) \cdot M_\ell(x_i) \quad (3)$$

As illustrated in Figure 3, we see that the hidden representations from lower-mid layers (10-20) have a significantly better execution accuracy compared to the last layer (40). Motivated by this finding, we propose to learn a weighted combination over the layers (Bahdanau et al., 2015) to effectively leverage the different granularity of information encoded in each layer of the LLM, since we do not know which layer performs the best beforehand. Specifically, given the hidden states representation for question x in each LLM layer $\{h_\ell(x)\}_{\ell=0}^{L}$ where L is the number of layers of LLM, we first use a 3-layer multi-layer-perception (MLP) (Murtagh, 1991) to transform the hidden states of each layer before computing the final retrieval representation $\mathcal{R}(x)$ as the weighted sum of layer transformations.

$$\mathcal{R}(x) = \sum_{\ell=1}^{L} w_\ell \cdot \text{MLP}_\ell(h_\ell(x)) \quad (4)$$

3.3 Answer similarity-based training objective

To optimize Equation 2, we need a labeled training set that indicates whether each in-context example $(x_i, y_i) \in \mathcal{D}_{\text{train}}$ is beneficial for the question x. However, since execution accuracy only provides binary signals, there are no trivial methods for directly computing the relative improvement of using an in-context example for any question x without performing pairwise inference. Further, there are often cases when a question can be correctly predicted for nearly all demonstration examples in $\mathcal{D}_{\text{train}}$ (and vice versa), which reduces the number

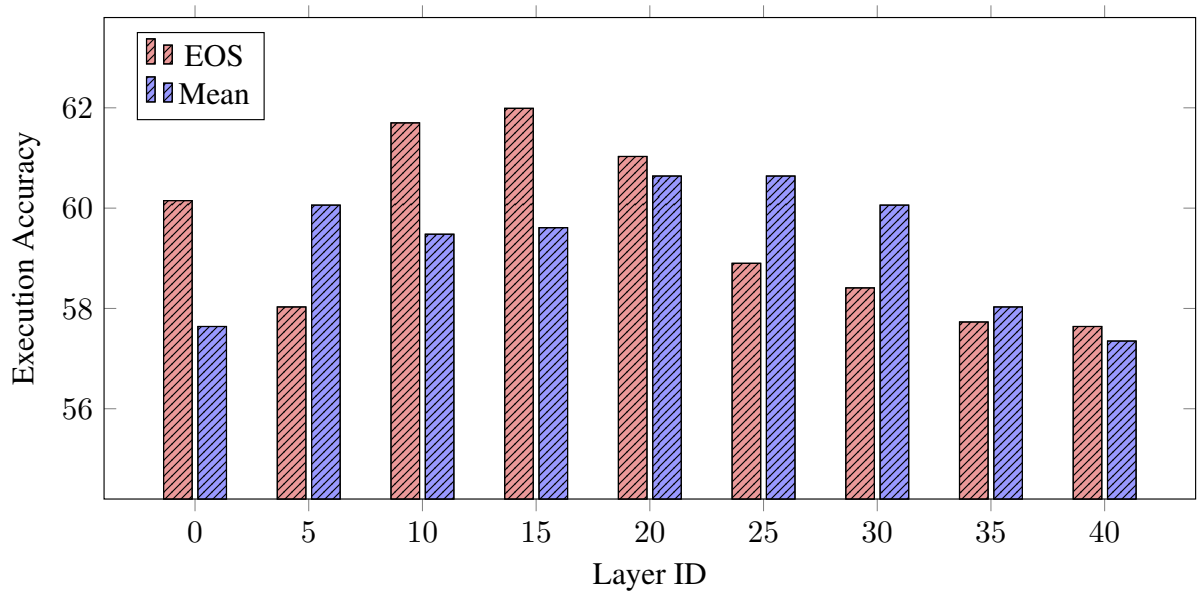

Figure 3: One-shot ICL results (execution accuracy on development set of Spider dataset) of CodeLlama-13b where prompt is retrieved by representations computed in each decoder layer. Here *Mean* represents average hidden state of a specific layer, where *EOS* represents the hidden state of EOS token in a specific layer. The one-shot Prompt is retrieved by the one with biggest cosine similarity.

of usable training examples. Therefore, we propose to use an approximate metric $\text{Sim}([x_i; y_i], [x; y])$ to approximate the benefits of using $[x_i; y_i]$ as examples to predict $\hat{y}$.

However, during model inference, since the target query is unknown, $\text{Sim}([x_i; y_i], [x; y])$ is not directly computed. Instead, we propose an indirect approach to predict $\text{Sim}([x_i; y_i], [x; y])$ by training the retrieval model $\mathcal{R}$ using supervised contrastive loss. The main intuition is to align the retriever embedding space (problem description as inputs) with the similarity between the concatenated representation of problem description and ground-truth SQL query. This contrasts the prior work by Nan et al. (2023a), where a proxy query is first generated through zero-shot inference, resulting in significant added latency. In practice, we select n_{pos} positive and n_{neg} negative examples for a given anchor x based on the dot-product of LLM hidden state representations. We formally define the contrastive loss in Equation 5, where τ is a constant temperature.

$$L = \sum_{x \in \mathcal{D}} \sum_{x_i \in \mathcal{P}_{\text{pos}}} \log \frac{\exp(\mathcal{R}(x) \cdot \mathcal{R}(x_i)/\tau)}{\sum_{x_j \in \mathcal{P}} \exp(\mathcal{R}(x) \cdot \mathcal{R}(x_j)/\tau)} \quad (5)$$

4 Experiments

4.1 Datasets

Our evaluation involved two comprehensive cross-domain datasets, Spider (Yu et al., 2018) and BIRD (Li et al., 2024). Spider contains 10,181 questions linked to 5,693 SQL queries across 138 domains and 200 databases, divided into 8,659 training, 1,034 development, and 2,147 test examples from unique databases. BIRD comprises 12,751 question-SQL pairs from 95 databases across 37+ domains, including blockchain and healthcare, totaling 33.4 GB. The dataset breakdown for BIRD includes 1,534 development, 9,428 training, and 1,789 test queries. Following prior works (Pourreza and Rafiei, 2023), the development sets from both datasets are used for evaluating the models without performing hyperparameter tuning.

In contrast to prior works using in-domain (ID) examples (Pourreza et al., 2024), where they assumed the demonstrations have the same DB schema as the final problem, we consider a more practical out-of-domain (OOD) scenario, where the DB schema of the problem is different from all demonstration. This OOD scenario reflects the practical challenges in real-world applications, where it is very uncommon to have accessible labeled examples under the same DB schema.[1]

[1]Note that OOD for NL2SQL is different from the tra-

	Spider		BIRD	
	Exe Acc. ↑	Error Rate ↓	Exe Acc. ↑	Error Rate ↓
Zero-shot	56.74	16.3	26.66	42.75
Random 1-shot	56.80	16.0	20.79	45.43
External embedding models				
Contriever	56.83	16.1	20.51	46.83
OpenAI Embeddings	57.06	14.3	20.99	45.38
LLM representations				
Mean (best)	57.90	15.5	28.55	41.92
EOS (last)	57.64	14.4	27.77	42.05
EOS (best)	61.99	11.2	29.07	42.96
DeTriever	**68.38**	**9.77**	**29.73**	**38.59**

Table 1: Results on Spider dataset and BIRD dataset.

4.2 Experimental Settings

We adopt Code-Llama-13b (Rozière et al., 2024) consisting of 40 attention layers as the base LLM. To reduce the computational complexity of the retriever, we only consider the hidden states of the LLM for every 5 layers (i.e., 0, 5, ..., 40). For the *DeTriever* model, we use 3-layer MLPs with an intermediate size of 1024 that project LLM hidden states to a dimension of 512 for retrieval representations. In total, the learnable parameters of *DeTriever* make up only 0.084% of the total size of the LLM.

We use the training set of BIRD and Spider, respectively, for training *DeTriever* to predict the similarity of the target SQL queries based on learned representation from the problem description. We train the retriever for a total 10k steps and store the checkpoints every 1k steps. Each training only takes less than 10 minutes in a single NVIDIA V100 GPU node. We tuned all hyperparameters on the Spider held-out development set and applied the best hyperparameter setting to the BIRD dataset. In the training phase, we set the number of negative examples $n_{\text{neg}} = 100$ and tuned the number of positive examples $n_{\text{pos}} \in \{5, 10, 20, 40, 60, 80\}$ and finally choose $n_{\text{pos}} = 40$. We also tuned batch size $\in \{16, 32, 64\}$ and finally chose the batch size to be 64. We use AdamW (Loshchilov and Hutter, 2019) as an optimizer. The learning rate is $1e-4$, weight decay is 0.01, beta1 is 0.9, and beta2 is 0.98, following the default setting. The temperature parameter τ for the supervised contrastive loss is set to 0.07 by default.

During inference, we use the demonstration template provided in the original code base for BIRD benchmark[2] in all our experiments. For retrieval, we select the example with the largest cosine similarity based on the representation from the trained retriever and use greedy decoding to generate the SQL query.

The evaluation metric is the execution accuracy (Exe Acc.) of predicted SQL queries. We also include the percentage of generated SQLs that cannot be successfully compiled (Error Rate) for additional information.

4.3 Baselines

We tested the following methods for one-shot in-context-learning on the NL2SQL task.

- **Random**: Randomly select example as a prompt without any retrieval.
- **External embedding models**: (1) **Contriever** (Izacard et al., 2021): an unsupervised encoder-based retriever using contrastive learning. (2) **OpenAI Embeddings**[3]: The method for retrieving NL2SQL demonstrations used by Nan et al. (2023b), where they used the general text embedding provided by OpenAI. We choose to use *text-embedding-3-large* to compute the embedding for retrieval.
- **LLM representations**: Use the mean-pooled or EOS hidden states of a specific layer of LLM to compute the similarity. (1) **Mean(best)**: the result using mean-pooled hidden representation from the best LLM layer. (2) **EOS(last)**: proposed by Wang

ditional definition where a model is trained in data in one domain and tested on another domain.

[2] https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/bird

[3] https://platform.openai.com/docs/guides/embeddings/what-are-embeddings

et al. (2024), the result using EOS token representation from the last hidden layer. (3) **EOS(best)**: the result using EOS token representation from the best LLM layer, note that this is an upper bound for the performance of few-shot learning based on problem description only, since there is no way of knowing the best layer ahead of time.

4.4 Results

From the results reported in Table 1, we see that our approach achieved the best performance across both benchmarks, with a 11.58 point increase in execution accuracy over the random one-shot baseline on Spider and 8.94 on Bird. Furthermore, we are able to validate our hypothesis that LLM hidden states offer a more effective means of encoding task descriptions for demonstration retrieval, where the representations derived from mean-pooling and end-of-sequence (EOS) hidden states surpassed the performance of both Contriever and the state-of-the-art OpenAI embeddings. It is also worth noting that the EOS token hidden states from the best layer demonstrated a 3.35 and 1.30 improvements in execution accuracy over the last layer. This observation challenges previous studies' reliance on using the final layer representations for retrieval, suggesting a reevaluation of the presumed equivalence to the hidden states of encoder-only models. Lastly, we are pleased to report that *DeTriever* outperforms both *EOS (best)* and *Mean (best)*, as they represent the upper-bound performance for retrieval using single-layer representations. This further confirms our hypothesis that using a combination of hidden layers can more effectively leverage the different granularity of information encoded in each layer of the LLM.

4.5 Analysis

4.5.1 Training Target

In the first analysis, we study the effects of using different proxy scores as approximations for the relative benefits of demonstration examples. Specifically, we select the pairwise embedding similarity of different representations as the target for the contrastive loss presented in Equation 5. In the results presented in Table 2, we compare the performance of mean-pooled and EOS representation of using only the target SQL query (Query-only) with using both the problem description and the query (Problem+Query). We also include the results for retrieving examples based on the problem description (Problem-only) as a reference for comparison. Since using the problem description does not require knowledge of the ground-truth query, we simply use the best-performing layer from Table 1 without the need for training.

	Exe Acc. ↑	Error Rate ↓
Problem-only (no training)	62.0	11.2
Query-only Mean	67.8	8.9
Query-only EOS	67.4	10.0
Problem+Query Mean	62.4	12.7
Problem+Query EOS	68.4	9.8

Table 2: Results on Spider dataset with different targets.

From the experiments, we see that training the retriever with Query-only does not significantly reduce the task performance, causing a less than 1 point accuracy decline over the EOS of Problem+Query. This finding stands in contrast to the results reported by Min et al. (2022), where they find that using only the task description without the ground-truth label can achieve similar performance compared to using input-label pairs. Since they only performed experiments on classification and multi-choice tasks, we hypothesize that for code-completion tasks such as NL2SQL, where there can be high variations in potential predictions, the model will often resort to copying or augmenting the target from the demonstration examples rather than relying on the priors from pre-training. This tendency renders the model particularly sensitive to the choice of demonstration examples, highlighting the importance of an efficient ICL retriever for optimal performance. This hypothesis also explains the poor performance of using the mean-pooled Problem+Query representations as training target, since problem description dilutes the information from the target query due to the autoregressive nature of the decoder-only LLM.

4.5.2 Number of Positive Anchors

We also study the effects of classifying different numbers of the most similar examples as positives in the contrastive objective. From the results presented in Figure 4, we see that the performance increases along with the number of positives with peak performance achieved using 40 positives. Since during the sampling process of the contrastive loss, the number of negatives n_{neg} is fixed to 100, we believe maintaining a good ratio between positives and negatives is critical for aligning the model to our proxy target. Since the

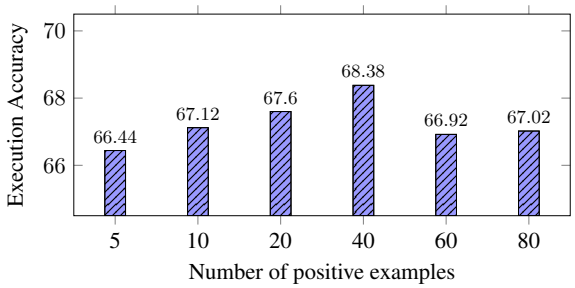

Figure 4: One-shot ICL results (execution accuracy on development set of Spider dataset) of CodeLlama-13b where the prompt is retrieved by our trained retriever with a different number of positive examples. The one-shot Prompt is retrieved by the one with the largest cosine similarity.

optimal ratio could also be dependent on the data distribution, we hope our findings can motivate further studies on modifying the objective based on the characteristics of the available ICL examples.

4.5.3 Batch Size

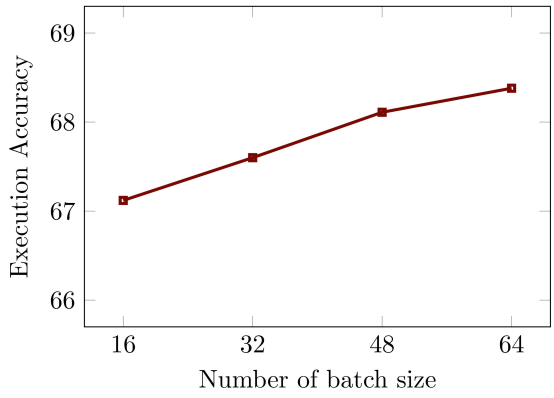

Figure 5: One-shot ICL execution accuracy on Spider where *DeTriever* is trained with different batch sizes.

Figure 5 illustrates the effects of different batch sizes when training *DeTriever* with supervised contrastive loss. Consistent with the results from prior studies (Chen et al., 2020; Khosla et al., 2020), we find that the task performance steadily improves as we increase the size of the mini-batch from 16 to 64. Since large batch sizes provide a more diverse set of negative examples, it enhances the model's ability to differentiate between subtle variations between examples, leading to superior task performance. We leave the experimentation of adapting more sophisticated metric learning techniques (Suárez-Díaz et al., 2020) for demonstration retrieval as an exciting venue for follow-up works. Table 3 shows the performance of our *DeTriever* on examples with different difficulty levels. We find out that for a larger training batch size, *DeTriever* tends to perform better on easier data.

Batch size	Easy	Medium	Hard	Extra
16	87.5	66.14	62.07	43.37
32	88.71	67.49	61.49	42.77
48	89.03	68.76	62.57	42.16
64	89.52	69.28	63.79	39.16

Table 3: One-shot ICL execution accuracy for different classes on Spider development set of *DeTriever*.

4.5.4 In-Domain Performance

All prior experiments in this work are conducted in the out-of-domain (OOD) settings, where there are no schema overlaps between the query and reference set. Nevertheless, when in-domain (ID) demonstrations are available, ID examples are usually better prompts in in-context learning for NL2SQL (Pourreza et al., 2024). Pourreza et al. (2024) trained an encoder-based retriever for in-domain demonstrations. To show the effectiveness of our *DeTriever* in the ID setting, we now study the effects of in-domain (ID) demonstrations where we directly retrieve ICL examples from the same schema in the evaluation set. In the results reported in Table 4, we see that retrieving in-domain examples with *DeTriever* significantly outperforms the best ID baselines by 15.9 accuracy points. This finding demonstrates that our trained *DeTriever* can effectively capture the nuances between examples under the same schema while emphasizing the advantages of having access to in-domain demonstrations. Besides, we find that *DeTriever* (ID) achieves an astounding 13.4 point improvement over *DeTriever* (OOD) retrieval model, indicating that *DeTriever* can generalize well when in-domain demonstrations are available.

	Exe Acc. ↑
OpenAI Embeddings†	69.1
Cohere†	69.3
SQL-Encoder†	73.2
DeTriever (ID)	**86.6**

Table 4: Results on Spider dataset retrieved by development set data. † are obtained from Pourreza et al. (2024)

5 Conclusion

In this work, we propose *DeTriever*, which to the best of our knowledge, is the first proposed method for fine-tuning the demonstration retrieval model for the task of NL2SQL. Specifically, we learn a weighted combination of LLM layer transformations is used to dynamically retrieve demonstration examples from the training set. We first perform an in-depth study to determine the proxy metric that best approximates the relative benefits of demonstration examples and then convert the proxy metric as the target for the contrastive loss objective used to fine-tune *DeTriever*. Our experiments on two popular benchmarks demonstrated the effectiveness of our approach. While we focus on NL2SQL in this work, our method can be naturally applied to other open-ended generation tasks such as code completion and question answering. For future work, we plan to conduct further experiments to analyze the performance of other models and tasks.

6 Limitations

The publicly available datasets used in this work are limited to English. Datasets in other languages (e.g., Vietnamese, Swahili) require LLMs with sufficient capabilities to understand the natural language questions containing domain-specific jargon. Further, while the scope of our experiments is limited to NL2SQL, we recognize the generalizability of our proposed method to other downstream tasks (e.g., question answering, code generation, etc.). However, one important assumption is the correlation between the similarity of the ground-truth answer (estimated through LLM embeddings) and the relative benefits of the corresponding examples for in-context learning demonstrations. This assumption is influenced by several task-specific factors, such as the complexity of the problem and the nature of the output space, and requires analysis to verify for any given task. Lastly, we only evaluate the performance on the NL2SQL benchmarks based on execution accuracy. Execution accuracy is not a perfect metric since it does not directly measure logical equivalence and is dependent on the correctness of the underlying database and the ability of the SQL queries to retrieve the expected results. Lastly, there may be other criteria such as query efficiency and simplicity, that could be important for the end-user.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *arXiv*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for

contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-SQL: A survey of what we have and what we expect. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2166–2187, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

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. *Preprint*, arXiv:2308.15363.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. *arXiv preprint arXiv:2310.08319*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Fionn Murtagh. 1991. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197.

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

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023b. Enhancing text-to-SQL capabilities of large language models: A study on

prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956, Singapore. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015*.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Juan Luis Suárez-Díaz, Salvador García, and Francisco Herrera. 2020. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges (with appendices on mathematical background and detailed algorithms explanation). *Preprint*, arXiv:1812.05944.

Sindhuja Swamidorai, T Satyanarayana Murthy, and KV Sriharsha. 2023. Translating natural language questions to sql queries (nested queries). *Multimedia Tools and Applications*, pages 1–15.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem 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, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. *Preprint*, arXiv:2401.00368.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

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*, Brussels, Belgium. Association for Computational Linguistics.