Learning Metadata-Agnostic Representations for Text-to-SQL In-Context Example Selection

Chuhong Mai*
Amazon Web Services
maichuh@amazon.com

Ro-ee Tal*
Amazon Web Services
rttal@amazon.com

Thahir Mohamed
Amazon Web Services
thahirm@amazon.com

Abstract

In-context learning (ICL) is a powerful paradigm where large language models (LLMs) benefit from task demonstrations added to the prompt. Yet, selecting optimal demonstrations is not trivial, especially for complex or multi-modal tasks where input and output distributions differ. We hypothesize that forming task-specific representations of the input is key. In this paper, we propose a method to align representations of natural language questions and those of SQL queries in a shared embedding space. Our technique, dubbed MARLO—**M**etadata-**A**gnostic **R**epresentation **L**earning for Text-t**O**-SQL— uses query structure to model querying intent without over-indexing on underlying database **metadata** (i.e. tables, columns, or domain-specific entities of a database referenced in the question or query). This allows MARLO to select examples that are structurally and semantically relevant for the task rather than examples that are spuriously related to a certain domain or question phrasing. When used to retrieve examples based on question similarity, MARLO shows superior performance compared to generic embedding models (on average +2.9%pt. in execution accuracy) on the Spider benchmark. It also outperforms the next best method that masks metadata information by +0.8%pt. in execution accuracy on average, while imposing a significantly lower inference latency.

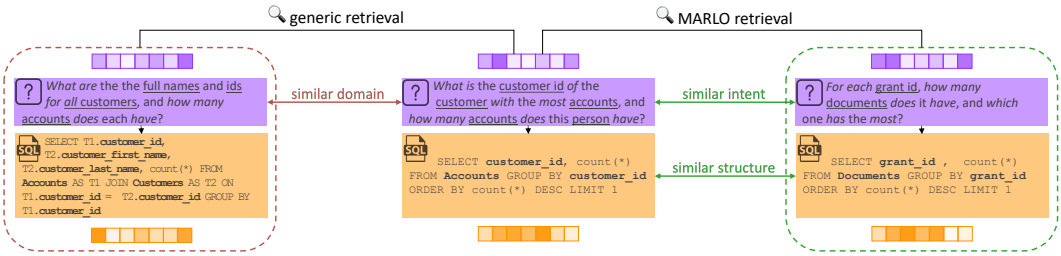


Figure 1: **Motivation of this work**. From the perspective of generic sentence embeddings, the left question is similar to the middle one but dissimilar from the one on the right. MARLO focuses on query structure (rather than metadata specifics) to represent the intent of each question more accurately. This allows it to retrieve the more instructive demonstration (rightmost). For emphasis, noun chunks, *parts-of-speech*, and **domain information specific to the database metadata** are annotated accordingly.

*These authors contributed equally to this work

Table Representation Learning Workshop at NeurIPS 2024.

1 Introduction

Large Language Models (LLMs) have demonstrated significant few-shot performance gains on a variety of NLP tasks simply by conditioning on example demonstrations during inference—an approach commonly referred to as in-context learning (ICL) (Brown et al., 2020; Dong et al., 2023). Not only is this little-understood emergent ability (Wei et al., 2022) an active area of research (Akyürek et al., 2023; Garg et al., 2022; Shin et al., 2022; Min et al., 2022; Xie et al., 2022), further effort has been made to understand why LLM performance remains sensitive to demonstration selection (Liu et al., 2022; Gao et al., 2023a; An et al., 2023; Wang et al., 2023b), ordering (Lu et al., 2022), and formatting (Chen et al., 2023a; Gao et al., 2023a). Inspired by the success of retrieval-augmented generation on knowledge intensive tasks (Hashimoto et al., 2018; Lewis et al., 2020; Karpukhin et al., 2020; Gao et al., 2023b), recent works demonstrate the effectiveness of selecting in-context examples that are semantically similar to the test exemplar (Rubin et al., 2022; Liu et al., 2022; Duan et al., 2023). However, semantic textual similarity has its challenges in the context of demonstration retrieval, particularity when the task is multi-modal (e.g. text $\mapsto$ image) or its input and output distributions differ (e.g. English $\mapsto$ French or SQL).

In this paper, we specifically focus on demonstration retrieval for Text-to-SQL as it is common practice in SOTA systems, and because the semantic and syntactic representations of natural language questions and structured queries are inherently difficult to compare. Generic retrievers tend to select examples by matching the domain or nouns in questions that relate to tables, columns, or entities of a database (e.g. 'accounts' or 'customers' in Figure 1), which we refer to broadly as database **metadata**. As a result, selected demonstrations describe semantically similar entities but have unrelated intent or query structure. To overcome this limitation, we fine-tune a pre-trained language model to align representations of natural language questions and those of SQL queries in a shared embedding space based on the question intent and corresponding query structure. Using a novel **metadata**-agnostic metric we propose (Equation 1), we learn representations that pay closer attention to structural information and retrieve lexically varied yet semantically meaningful demonstrations.

Our work, dubbed **M**etadata **A**gnostic **R**epresentation **L**earning for Text-t**O**-SQL (MARLO), differs from prior works that i) compare embeddings derived from heuristic feature extraction (Makiyama et al., 2015; Aligon et al., 2014; Kul et al., 2018), ii) use general-purpose retrievers to select from suitably formatted prompts (Sun et al., 2023), iii) use custom retrievers that add special code tokens to their vocabulary (Tipirneni et al., 2022) or are fine-tuned on masked task-specific data (Gao et al., 2023a), or iv) annotate demonstrations by hand, which is prohibitively expensive and inflexible (Pourreza and Rafiei, 2023).

From our ablation analysis and experiments, we observe example selection with MARLO outperforms alternative methods discussed in literature, including the state-of-the-art for the Text-to-SQL task on benchmark dataset. Compared to multi-stage masking approaches that require multiple LLMs calls, MARLO performs better or on par with lower inference latency. In summary our major contributions are:

1. A novel method to jointly learn aligned embeddings for natural language questions and SQL queries using a query edit distance heuristic. The customized embedding model is able to better comprehend the anticipated SQL structure associated with natural language questions.

2. An in-depth analysis of ICL example selection methods for Text-to-SQL, highlighting how the choice of selected examples is particularly useful for question understanding in this task and why MARLO works.

3. First comprehensive showcase of competitive instruction-tuned foundation models on this task.

2 Related Work

2.1 ICL Demonstrations for Text-to-SQL

Several lines of work have explored what aspects of a demonstration contribute to ICL performance gains – the input distribution (e.g. formatting, overall perplexity) (Min et al., 2022; Gonen et al., 2023); the semantic similarity between demonstrations and the inference exemplar (Liu et al., 2022; Duan et al., 2023; Qin et al., 2023); and diversity (Qin et al., 2023). For a skill-intensive task like Text-to-SQL, similarity is a more important dimension than diversity. Demonstrations that possess similar questions (Liu et al., 2022) or SQL queries (Nan et al., 2023) to those of the inference exemplar are more helpful than random selections. However, solely relying on embeddings of raw input questions or queries for similarity retrieval often suffers from the bias of surface language features (e.g. nouns that contain domain information). Guo et al. 2023 and DAIL-SQL (Gao et al., 2023a) address this by masking noun chunks in the question or query referenced in the metadata before encoding. Skill-KNN (An et al., 2023) take an alternative approach by rewriting the questions as skill descriptions which, rather than the questions, are encoded and retrieved using generic embedding models. To the best of our knowledge, we are the first to tackle this problem by learning embeddings for natural language questions and SQL queries that are aligned in a shared space and represent the user's intent agnostic to the domain or any referenced database metadata.

2.2 Similarity-Based Retrieval

Similarity-based retrieval is typically framed as a metric learning problem that uses dot-product or cosine similarity as a ranking function (Kulis, 2012). In cases where input and output distributions are distinct, it is a common practice to fine-tune or align pre-trained transformers using language modeling or constrastive objectives, often in a Siamese configuration (Cer et al., 2018; Reimers and Gurevych, 2019; Yang et al., 2020; Gao et al., 2021; Ni et al., 2022a; Neelakantan et al., 2022). For example, CLIP (Radford et al., 2021) uses internet-scale language supervision to learn broader semantic concepts in the vision domain via contrastive pre-training over image-caption pairs. In the Text-to-SQL task, where we perform similarity-based retrieval for ICL demonstrations, we face the same problem of representing asymmetric entities, namely the natural language question and the SQL query. Some previous works (Ni et al., 2022b) look to supervised fine-tuning where sufficient in-domain data is available, and others propose new objective functions that are optimized for the task or domain (Li and Li, 2023). In this work, we devise a task-specific metric and perform weak supervised learning to align natural language with SQL using standard objective functions. Our method effectively learns metadata-agnostic embeddings without masking in-domain data, customizing the base model vocabulary, rewriting questions, or using custom objective functions.

3 Learning Metadata-Agnostic Text-to-SQL Embeddings

We seek to learn aligned embeddings of natural language questions and SQL queries that are semantically meaningful for retrieving ICL demonstrations in the Text-to-SQL task. We define a novel similarity metric in § 3.1 based on a query edit distance heuristic.

3.1 Query Edit Distance (QED)

Consider a dataset D comprised of n (database d, question q, query s) triplets $\{(q_i \mapsto s_i|d_i)\}^n$. To measure the alignment between two exemplar pairs (i.e. $(q_i, s_i) \leftrightarrow (q_j, s_j)$), we take a query editing perspective and use a keyword matching heuristic that matches the structural similarity between the queries s_i and s_j as a proxy. Since the same expression can be expressed in many different ways using different SQL keywords, we group keywords based on their semantic nature and assign weights to them according to their impact on query structure (see Table 5). For each group k, we obtain SQL

keywords of the minimal insertions (I_k) and removals (R_k) to change s_i to s_j.

$$\text{QED} = \sum_k \Big(w_k \cdot |I_k - R_k| + w_r \cdot \min(I_k, R_k) \Big) \quad (1)$$

Intuitively, any keyword in group k that can be replaced with another keyword from the same group to make s_i closer to s_j comprises a small difference between the queries. For example, replacing `MIN` with `MAX`. Each replacement operation contributes a low score of $w_r = 0.2$ to QED. Additional keywords that need to be added, for example if an `ORDER BY` clause is missing, contribute a score of w_k based on the significance of the group to the structure of the query. Operations pertaining to table and column names are ignored (e.g. aliases) and will result in the same score. This ensures the representations are agnostic to the underlying metadata of d_i and focus more on the query structure.

3.2 Dataset

We construct an augmented training dataset for our encoder using the training set of **Spider**[1] (a large-scale cross-domain Text-to-SQL benchmark that covers over 200 databases) comprised of 7000 examples (Yu et al., 2018). Following § 3.1 each question is paired with all queries in the dataset to generate a training dataset $\bar{D} = \{(q_i, s_j, l_{ij})\}^{49M}$, where l_{ij} represents a measure of alignment between q_i and s_j based on $\text{QED}(s_i, s_j)$. However, we find $\bar{D}$ is unbalanced, with only $\sim$1% of the queries considered similar (i.e. QED score lower than 1). We undersample $\bar{D}$ by comparing the euclidean distance (τ) of generic embeddings[7] of q_i and q_j and l_{ij}. To enhance the learning process, we sample more from the discordant group of exemplars, which we define as those with a large $\tau = \|q_i - q_j\|^2$ and low l_{ij}, or vice versa. The total size of the filtered training set is brought down $\sim$2.4M with majority of the exemplars representing some degree of discordance, see Table 4 for a complete breakdown. Since we are interested in learning fine-grained representations, we cap $QED \leq 5$, and min-max normalize the scores $l_{ij} \in [0, 1]$ such that a score of 0 represents the most distant exemplars and is compatible with the chosen loss function. See Table 6 for a list of labeled examples. For a complete set of hyperparameters used to train the encoder, refer to Table 3.

3.3 Loss Function

The objective function used to train embedding models is usually selected based on the scale and nature of available training data. For instance a contrastive loss (Ni et al., 2022a; Neelakantan et al., 2022) such as Multiple Negatives Ranking Loss (MNRL) is used when pairwise data (e.g. images and text captions) is available, while a Triplet Loss is used when a neutral anchor can be compared to both positive and negative examples explicitly. Normally Text-to-SQL datasets are comprised of (question, query) pairs, however a contrastive loss will likely not work well here as different questions might correspond to structurally similar queries in the same batch. Hence, we create an augmented dataset with pseudo-labels based on § 3.1 and train our encoder by minimizing the Cosine Similarity Loss ($\mathfrak{L}$), where $\vec{q}_t$ and $\vec{s}_t$ are the question and query embeddings of the t^{th} training example, respectively, l_t is the alignment score derived using Equation 1, and $\text{sim}(\vec{q}_t, \vec{s}_t)$ is the cosine similarity of the embeddings, defined in Equation 3. We choose the Cosine Similarity Loss rather than a Triple Loss as the latter does not distinguish distances between similar and dissimilar exemplars whereas the derived label l_t does.

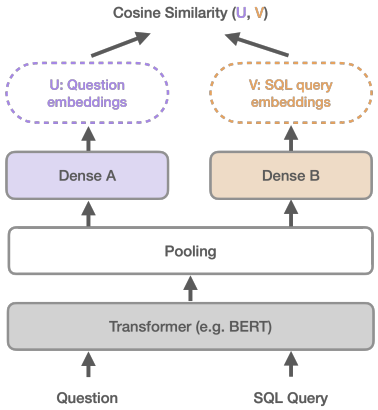


Figure 2: **Semi-asymmetric bi-encoder architecture.**. Parameters in the base transformer and the pooling layer are shared while two separate dense layers are trained to align question and SQL query embeddings, respectively.

$$\mathfrak{L}(\vec{q}_t, \vec{s}_t, l_t) = \| \, l_t - \text{sim}(\vec{q}_t, \vec{s}_t) \, \|_2 \quad (2)$$

$$\text{sim}(\vec{q}_t, \vec{s}_t) = \frac{\vec{q}_t \cdot \vec{s}_t}{\| \, \vec{q}_t \, \| \| \, \vec{s}_t \, \|} \quad (3)$$

[1]We exclude the Yelp and additional training sets

3.4 Architecture

Our embedding model is based on a semi-asymmetric bi-encoder architecture[2] (Figure 2). A pre-trained transformer[3] serves as a common backbone to produce intermediate representations of either a question or a query using the same vocabulary. A pooling layer connects the backbone to two separate dense layer heads, one for each embedding type. Following common practice, we use mean pooling and `tanh` activation in the dense layers. This architecture diverges from symmetric (Reimers and Gurevych, 2019) or asymmetric (Gillick et al., 2018; Lee et al., 2020) sentence embedding models, where either all or no parameters are shared in the Siamese configuration, respectively (Figure 4). Given the declarative nature of SQL queries, we hypothesize that parameter sharing in the backbone benefits from the transfer learning abilities of pre-trained transformers and the separate embedding heads acts as a form of regularization during alignment. In § 5.3, we validate this hypothesis.

4 Experimental Setup

Given an input question q', an LLM with frozen parameters θ and sampling parameters ϕ and a pool of n demonstrations $D = \{(q_i \mapsto s_i | d_i)\}^n$, the LLM will generate a SQL query s' by sampling from the distribution: $s' \sim \text{LLM}_{\theta,\phi}\left(S\left(q', D\right) \oplus q'\right)$ where S selects k demonstrations from the pool D based on q', and $\oplus$ formats the prompt with the inference exemplar alongside the demonstrations. We provide sampling hyperparameters in Table 7.

4.1 Evaluation Dataset and LLM

Given the complexity of this task, we use both open and closed large, instruction-tuned models that exhibit competitive natural language understanding, reasoning, and coding abilities (Claude[4], Mistral[5], and Llama 3[6]) to assess the generalizability of our approach, and evaluate their Text-to-SQL ICL capabilities on the **Spider** development (1034 instances) and test (2147 instances) sets. Note, these two datasets and the training split used to train the embedding model do not have any database in common, which allows us to validate the generalizability and domain-agnostic property of our learned representations. Using prompts like Listing 1 formatted suitably for each model, we insert the question and table metadata from each instance at inference time.

4.2 Demonstration Selection Methods

We run a comprehensive list of experiments inspired by related work (many of which we are unable to reproduce exactly). We include a zero-shot setting (no in-context demonstrations) as a baseline. In total, we implement 5 different example selection methods to pick $k = 8$ demonstrations (the number of examples is chosen based on an ablation, see Figure 3), namely:

- **Random Sampling** Another common baseline which uniformly samples examples.
- **Question Similarity (QS)** The top-k examples are selected based on the euclidean distance between embeddings[7] of the test question and those of questions from the example pool. Liu et al. 2022 adopted this approach using SBERT.
- **Masked Question Similarity (MQS)** Same as QS, except information from the database metadata is masked in each question using a `[MASK]` token. Compared to our implementation, Guo et al. 2023 use GPT-3.5 (`text-davinci-003`).
- **Skills Similarity (SS)** An LLM first produces a summary description of the *skills* required to solve any given exemplar, and then examples are retrieved based on the similarity of encoded[7] skill descriptions. We use the same prompt and 16 hand-crafted examples as An et al. 2023.

[2] Implemented using `https://www.sbert.net/`
[3] We use `bert-based-uncased`
[4] Created by Anthropic
[5] Created by Mistral AI
[6] Created by Meta
[7] We use `amazon.titan-embed-g1-text-02`

- **MARLO (*this work*)** Same as QS, except we use embeddings from our encoder described in § 3.4. In § 5.3 we also conduct an ablation using demonstrations retrieved with query embeddings instead of question embeddings.

4.3 Metrics

We evaluate all experiments using execution accuracy (EX), which measures the match between the execution outputs of the predicted and ground-truth SQL queries. While the official Spider test suite (Seq-Eval) is used ubiquitously, it does not allow for the presence of redundant columns in the predicted query, which may be included depending on the LLMs query writing style. To relax this requirement, we propose a modified execution accuracy (EX†) which is based on Seq-Eval but considers all columnar permutations of the predicted query execution with the same number of columns as the ground-truth execution (Algorithm 1).

5 Results & Discussions

We present the main results from our experiments in Table 1. Overall, example selection with MARLO outperforms generic methods by +2.9 percentage points on average. It also outperforms the next best method (SS) with significantly less inference latency as no additional LLM call is needed.

5.1 Example Selection with MARLO

It is not surprising that MQS and SS outperform QS, as the former methods also exclude domain-specific information contained in the database metadata using either question masking or rephrasing, respectively. MARLO, on the other hand, achieves the same objective without risking information loss or increasing inference latency and complexity (discussed further in § A.6). It does so by leveraging fine-grained embeddings of natural language questions that are aligned to their corresponding SQL structure but still retain all relevant linguistic information.

Table 1: **Execution accuracy (%) of example selection methods**. All experiments select $k = 8$ examples. Results of the top performing method are in **bold**. MARLO mostly outperforms other methods, and is competitive otherwise.

LLM	Selection Method	Spider-dev		Spider-test	
		EX	EX†	EX	EX†
Claude 2.1	0-shot	67.5	78.5	69.0	80.0
	Random	72.2	79.4	75.1	82.2
	QS	73.5	80.5	75.1	80.5
	MQS	75.3	80.3	78.0	82.8
	SS	77.9	81.2	80.7	83.1
	MARLO	**80.8**	**83.6**	**81.2**	**84.0**
Mistral Large	0-shot	74.4	79.5	76.8	81.8
	Random	76.3	79.9	78.2	82.3
	QS	77.7	79.8	79.4	82.9
	MQS	77.3	79.4	80.2	83.1
	SS	78.3	79.7	81.6	**83.4**
	MARLO	**80.0**	**81.7**	**81.7**	83.2
Llama 3 70B Instruct	0-shot	77.0	81.5	75.6	81.3
	Random	79.0	81.8	78.6	81.8
	QS	81.5	**83.1**	78.6	80.7
	MQS	81.8	82.9	80.5	82.3
	SS	82.1	82.9	**83.4**	**84.4**
	MARLO	**82.2**	83.0	83.2	83.8

Table 9 further shows examples of different demonstrations retrieved using MARLO versus the other methods. MARLO stands out by selecting demonstrations with structurally-alike queries and question intents across domains and a variety of question phrasings. Its success in this task reveals that fine-grained task-specific embeddings can be used to select relevant but linguistically diverse demonstrations and, therefore, have the potential to help LLMs better comprehend natural language inputs of complex or multi-modal tasks during inference via ICL. This ability is exemplified on difficult exemplars, where MARLO with both Claude 2.1 and Mistral Large significantly outperform other methods by +4-7%pt. (see Figure 5).

5.2 LLM Text-to-SQL Capability

EX vs EX† As shown in Table 1, the accuracy scores are systematically higher yet more stable when measured with EX† regardless the LLM chosen. This effect is more pronounced in Claude

2.1 than other LLMs. Upon investigation, we find that SQL queries produced by Claude 2.1 tend to order numerical columns before categorical columns and include additional redundant columns. This explains why its EX is significantly lower, while EX† is on par with other LLMs. We hypothesize an LLM's query writing style, and therefore sensitivity under EX, is attributed to differences in training data and instruction tuning.

ICL & Llama 3 Instruct We observe that Llama 3 Instruct behaves differently to the other LLMs. It exhibits competitive zero-shot performance but does not benefit as much from ICL, regardless of the selection method used. It is possible that its instruction-tuning procedure trades off Text-to-SQL ICL ability as an "alignment-tax" (Ouyang et al., 2022), or is unable to benefit from proprietary data.

Comparison with GPT 4 Despite the inconsistent use of sampling hyperparameters, prompts, and number of examples reported in literature, we list the execution accuracy reported by other studies using GPT 4 for comparison in Table 10. EX† is computed for comparison when possible (i.e. works are reproducible or their predictions are shared publicly). We observe a similar trend of performance improvement across the selection methods with GPT 4 as with Claude 2.1 and Mistral Large. Notably, MARLO using Claude 2.1 is competitive with DAIL-SQL[8] using GPT 4 on Spider-dev (EX†: 83.6 vs. 83.3, respectively).

5.3 Ablation Study

We perform four ablations to study our proposed retrieval process, illustrated in Table 2 and Figure 3.

Encoder Architecture As discussed in § 3.4, the architecture of our encoder deviates from that of common symmetric (e.g. SBERT) and asymmetric (e.g. CLIP) bi-encoders. In Table 2b we explore the effect of parameter sharing by training either a single encoder to embed both questions and queries, a separate encoder for each, or separate output layers that process representations from a shared backbone. See Figure 4 for the three different architectures explored. We find the semi-asymmetric architecture is more capable of learning expressive and aligned representations than the symmetric or asymmetric alternatives. We expect parameter sharing in the backbone benefits from the transfer learning abilities of large pre-trained language models and helps build alignment between latent representations of question and queries from the same parameter space. This idea has been explored and validated by Dong et al. 2022 in the context of question-answer retrieval. Moreover, separation of parameters in the later layers helps enforce alignment in the backbone independent of the way the questions or queries are encoded, which explains why the asymmetric architecture performs slightly better than a symmetric one.

Table 2: **Execution accuracy (%) of MARLO ablations**. Ablation (a) explores the effect of parameter sharing in the encoder architecture, (b) compares the chosen objective function to Multiple Negatives Ranking Loss (MNRL), and (c) compares end-task performance for various retrieval recipes. In (c) a relative number of unique demonstration selected during each evaluation ($\mathfrak{R}$) is also reported. All studies use Claude 2.1.

(a) retriever dual-encoder architectures

Architecture	Spider-dev		Spider-test	
	EX	EX†	EX	EX†
Symmetric	78.6	81.4	80.9	83.6
Semi-asymmetric	**80.8**	**83.6**	81.2	**84.0**
Asymmetric	77.6	81.6	**81.7**	**84.0**

(b) retriever training objective functions

Loss function	Spider-dev		Spider-test	
	EX	EX†	EX	EX†
MNRL	75.8	81.6	79.1	82.9
Cosine Similarity	**80.8**	**83.6**	**81.2**	**84.0**

(c) linearly combined question/query embeddings

Embed. Weight	Spider-dev			Spider-test		
	EX	EX†	$\mathfrak{R}$	EX	EX†	$\mathfrak{R}$
Query	76.3	82.3	.28	79.2	**84.3**	.38
50 / 50	79.0	82.2	.45	**81.3**	84.1	.62
70 / 30	79.0	82.9	.55	81.0	83.4	.77
Question	**80.8**	**83.6**	.71	81.2	84.0	1

[8]We consider DAIL-SQL state-of-the-art (SOTA)

Encoder Loss Function Retrievers are often trained with contrastive loss functions when pairwise data (such as questions and queries) is available. Here, we compare the effectiveness of using more fine-grained pseudo-labels derived from the metric we propose in § 3.1 with the Multiple Negatives Ranking Loss (MNRL, Henderson et al. 2017). MNRL computes the cross entropy of all possible question-query combinations in a batch, where pairs of questions and their ground-truth queries are assigned a positive label and all other pairs a negative label. Due to the impact of the batch size on the loss functions, we adjust the training hyperparameters for MNRL to ensure sufficient coverage of both positive and negative pairs (see Table 3). As Table 2b shows, cosine similarity using pseudo-labels derived using QED leads to better end-task performance than MNRL. This is because QED scores provide better signal for supervision than the binary scores used in MNRL and are less noisy than the latter. It is probable that false negatives exist in batch for questions that are dissimilar but have similar corresponding queries, and vice versa. Nevertheless, the results using MNRL are either on par or better than those achieved with other example selection methods, highlighting the utility of aligned embeddings.

Embedding Alignment & Retrieval Recipe Unlike previous work (e.g. Gao et al., 2023a), our question and query embeddings can be used interchangeably without the additional overhead of predicting a preliminary query because they are closely aligned. Table 2c shows that selecting demonstrations for a given test question based on its semantic similarity to candidate questions or queries results in comparable performance on $EX^{\dagger}$. Nevertheless, question embeddings are more expressive than their query counterparts as the former leads to a greater number of unique demonstrations selected during the evaluation ($\Re$). Therefore, it appears to be more beneficial when including the exact number of output fields in the predicted query is required (i.e. EX). Intuitively, a natural language question can be written in many more ways than its corresponding SQL query, which explains why their respective embeddings are aligned but not equally expressive. Although linear combinations of question and query embeddings does not result in obvious gains, we believe further exploration of the compositionality and factorizaiton of these embeddings (Trager et al., 2024) can help boost performance and support their broader application.

Number of Selected Demonstrations Overall, we observe Text-to-SQL performance initially increases then plateaus as we increase the number of in-context demonstrations (Figure 3). It is likely the case that selecting demonstrations based solely on semantic similarity has its limits, as the information gain of each new demonstration added to the context saturates. Perhaps an ensemble of example selection methods that optimize for other aspects might be beneficial in large-context settings. In addition, EX is systematically lower than $EX^{\dagger}$, and this effect is more pronounced when fewer demonstrations are used in context. Based on the asymptotic difference between evaluation algorithms in Figure 3, one of the first patterns LLMs learn from the in-context demonstrations is an understanding of what (and the exact number of) fields to include in the predicted query. Since this pattern solely relies on question understanding as apposed to query writing abilities, our interpretation is that the LLM uses its inherent (zero-shot) SQL understanding to inform question understanding. This corroborates recent work by Wang et al., 2023a who argue demonstration outputs serve as anchors through which information flows from the demonstration inputs in the shallow layers of the model (Min et al., 2022; Dziri et al., 2023). We expect these anchors represent latent interpretations of the questions in the context of their corresponding queries,

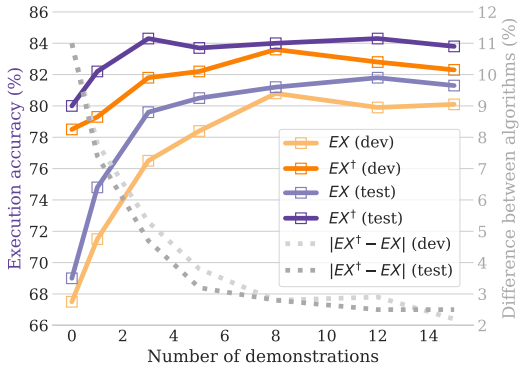


Figure 3: **Execution accuracy (%) of MARLO for various numbers of selected demonstrations.** Performance initially increases and then plateaus as more demonstrations are included in the context, implying in-context learning scaling limitations for this task.

and in the deeper layers of the model, the relevant representations inform the query generation process for the test exemplar.

6 Conclusions

In this work we explore the problem of selecting demonstration examples to improve ICL capabilities of LLMs on the Text-to-SQL task. Given the disjointed distributions of natural language questions and SQL queries, general-purpose internet-scale encoders are generally not well-suited to retrieve semantically similar demonstrations. Therefore, we propose a novel approach, MARLO, that trains a bi-encoder to align the representations of natural language questions and SQL queries according to their underlying intent. Via weak supervision from a metadata-agnostic similarity label, MARLO selects ICL demonstrations that enables LLMs to excel in generating queries. Not only are our results competitive with the state-of-the-art, MARLO is also more efficient and effective than previous domain masking techniques. Our ablations reveal that the selections it provides are semantically relevant for the task yet linguistically unconstrained. MARLO's success suggests that fine-grained task-specific embeddings have the potential to enhance LLMs in complex or multi-modal ICL settings.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models.

Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. 2014. Similarity measures for olap sessions. *Knowl. Inf. Syst.*, 39(2):463–489.

Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. 2023. Skill-based few-shot selection for in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13472–13492, Singapore. Association for Computational Linguistics.

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.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023a. Unleashing the potential of prompt engineering in large language models: a comprehensive review.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Zhe Dong, Jianmo Ni, Daniel M Bikel, Enrique Alfonseca, Yuan Wang, Chen Qu, and Imed Zitouni. 2022. Exploring dual encoder architectures for question answering. *arXiv preprint arXiv:2204.07120*.

Hanyu Duan, Yixuan Tang, Yi Yang, Ahmed Abbasi, and Kar Yan Tam. 2023. Exploring the relationship between in-context learning and instruction tuning.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. 2023. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598. Curran Associates, Inc.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.

Hila Gonen, Srini Iyer, Terra Blevins, Noah Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10136–10148, Singapore. Association for Computational Linguistics.

Chunxi Guo, Zhiliang Tian, Jintao Tang, Pancheng Wang, Zhihua Wen, Kang Yang, and Ting Wang. 2023. A case-based reasoning framework for adaptive prompting in cross-domain text-to-sql. *arXiv preprint arXiv:2304.13301*.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

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.

Gokhan Kul, Duc Thanh Anh Luong, Ting Xie, Varun Chandola, Oliver Kennedy, and Shambhu Upadhyaya. 2018. Similarity metrics for sql query clustering. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2408–2420.

Brian Kulis. 2012. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2020. Learning dense representations of phrases at scale. *arXiv preprint arXiv:2012.12624*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings.

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.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

Vitor Hirota Makiyama, Jordan Raddick, and Rafael D. C. Santos. 2015. Text mining applied to sql queries: A case study for the sdss skyserver. In *Symposium on Information Management and Big Data*.

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?

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

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022b. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

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

Chengwei Qin, Aston Zhang, Anirudh Dagar, and Wenming Ye. 2023. In-context learning with iterative demonstration selection.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

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.

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.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, and Nako Sung. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5168–5186, Seattle, United States. Association for Computational Linguistics.

Jiashuo Sun, Yi Luo, Yeyun Gong, Chen Lin, Yelong Shen, Jian Guo, and Nan Duan. 2023. Enhancing chain-of-thoughts prompting with iterative bootstrapping in large language models.

Sindhu Tipirneni, Ming Zhu, and Chandan K Reddy. 2022. Structcoder: Structure-aware transformer for code generation. *arXiv preprint arXiv:2206.05239*.

Matthew Trager, Pramuditha Perera, Luca Zancato, Alessandro Achille, Parminder Bhatia, and Stefano Soatto. 2024. Linear spaces of meanings: Compositional structures in vision-language models.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023b. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models.

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. 2022. Emergent abilities of large language models.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2020. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

A Appendix

A.1 Limitations

A.1.1 Open vs. Closed Models

Closed foundation models significantly outperform open models across a broad range of complex reasoning tasks and offer certain capabilities to which open models have not caught up. However, two major limitations stand out when using closed models for research. First, the cost and throughput can be prohibitively expensive and slow, respectively. Second, the lack of interoperability of these models and transparency concerning their training data and methods creates a restrictive research environment in which reproducing and benchmarking against prior works becomes challenging. Hence, we are not able to perform a direct and thorough comparison of Text-to-SQL capabilities between additional LLMs (e.g. GPT 4) by keeping hyperparameters, prompting technique and example selection method in a controlled manner. Moreover, it is difficult to reason about performance differences between open and closed models as we are unable to compare parameter counts, training data, or fine-tuning and alignment methods.

A.1.2 Automatic Query Generation

While SQL generation solutions built using probabilistic models might appeal to database management and information extraction business use cases, the risk of hallucination and catastrophic customer impact remains to be explored. By observing the upper-bound performance in Table 1, we see there is still ample room for improvement on the robustness of LLMs across benchmarks and domains.

A.2 MARLO Encoder Training

A.2.1 Hyperparamters

In Table 3 we provide the hyperparamters used for all MARLO encoder experiments. All experiments use a NVIDIA A10G Tensor Core GPU.

Table 3: **Hyperparameters of training our customized embedding model**.

Hyperparameter	**Cosine Similarity**	**MNRL**
maximum input sequence length	256	256
fixed output size	128	128
batch size	32	32
Optimizer	`AdamW`	`AdamW`
learning rate schedule	linear warm-up	linear warm-up
maximum learning rate	2×10^{-5}	2×10^{-5}
# of learning rate warm up steps	10000	100
weight decay for model parameters	0.01	0.01
maximum gradient normalization	1	1
# of epochs	2	15

A.2.2 Sampling Discordant Questions & Queries

Table 4 displays details about how we undersample our original 49M pairs of questions and queries to the final training set used in MARLO. We sample the given number of exemplars (from the total per category) according to QED score and euclidean distance between embedded questions q_i and q_j criteria outlined in the table. To facilitate better learning of the encoder, we intentially select more examples from discordant categories.

Table 4: **Under-sampling of augmented spider training set**. We reduce the training dataset from 49M to ∼2.4M.

$\tau = \|q_i - q_j\|^2$	$1 < l_{ij}$	$1 \leq l_{ij} < 3$	$3 \leq l_{ij}$
$\tau \geq 300$	600K (∼5.4M)	600K (∼10M)	100K (∼32.7M)
$\tau < 300$	170K (170K)	150K (150K)	800K (800K)

A.2.3 Similarity via Query Edit Distance

This section provides the details about how we compute Query Edit Distance (QED) score based on the differences in SQL keywords that appear in insert and remove operations when two queries are compared. Since SQL keywords are grouped based on their influence on the overall query structure

Table 5: **SQL keyword group weights.** For groups with multiple keywords, a universal weight 0.3 is used. Since these groups can have non-zero counts of both insertions and removals, they are considered to lead smaller within-group distances and are assigned a weight of 0.2. See Equation 1 for details. While this list is complete for our given dataset, it might not be complete for all possible queries and dialects. Additional keywords would have to be included appropriately.

Group (k)	**SQL Keywords**	w_k
Aggregation	`COUNT`, `AVG`, `SUM`, `MIN`, `MAX`	0.3
Comparison	`EQ`, `NEQ`, `LIKE`, `GT`, `GTE`, `LT`, `LTE`, `BETWEEN`, `IN`	
Composition	`AND`, `OR`	
Arithmetic	`ADD`, `SUB`	
–	`LIMIT`	0.1
	`DISTINCT`	0.2
	`WHERE`	0.5
	`HAVING`	0.7
	`GROUP`	0.6
	`ORDER`	0.6
	`JOIN`	3.0
	`SELECT`	3.0
	`SUBQUERY`	4.0
	`EXCEPT`	4.0
	`UNION`	3.0
	`INTERSECT`	3.5

(Table 5), we consider two different queries closer in edit distance when their differences are within the same compared. See Equation 1 for the formula of QED computation.

A.2.4 QED Example

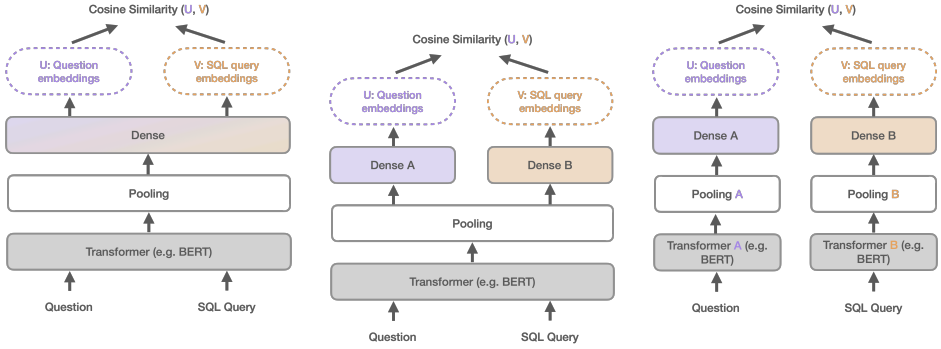
Table 6 shows an example of how QED scores and corresponding similarity labels look like for a single question paired with four different possible queries. The top row is the ground truth query of the question so the QED score is 0 and similarity score is 1. From top to bottom, we see an increase in QED, representing the queries become more and more irrelevant to the question asked.

A.2.5 Bi-encoder Architectures

Figure 4 shows three different architecture choices for bi-encoders. Symmetric architectures are commonly used in literature (e.g. SBERT) while asymmetric architectures are also found effective in use cases where the input and output have different lengths or distributions (e.g. document retrieval based on questions). Our encoder is trained from a semi-asymmetric architecture (Figure 4b) where the backbone transformer and pooling layers are shared but dense layers are independent.

Table 6: **Examples of QED score and similarity score labeling**. The scores are impacted by the dissimilar SQL structure but not by the domain information.

Question	Ground-Truth Query	Possible Query	QED Score	l_{ij}
How many heads of the departments are older than 56?	`SELECT count(*) FROM head WHERE age > 56`	`SELECT count(*) FROM head WHERE age > 56`	0	1
		`SELECT count(*) FROM professor WHERE prof_high_degree = 'Ph.D.'`	0.2	0.96
		`SELECT major, count(*) FROM Student GROUP BY major`	1.4	0.72
		`SELECT DISTINCT T1.age FROM management AS T2 JOIN head AS T1 ON T1.head_id = T2.head_id WHERE T2.temporary_acting = 'Yes'`	5	0



(a) Symmetric (b) Semi-asymmetric *(this work)* (c) Asymmetric

Figure 4: **Architectures choices for bi-encoders.** A symmetric architecture (a) have parameters shared in all three modules while an asymmetric architectures (c) does not share any layer between the two towers. Our work adopted a semi-asymmetric structure where a common backbone transformer and pooling layer are shared, but dense layers are separated.

A.3 Query Generation & Evaluation

A.3.1 Decoding Hyperparamters

Table 7 provides the configuration used to predict SQL queries. For experiments that require multiple predicted queries (i.e. self-consistency and upper bound) we use a non-zero temperature of 0.7.

Table 7: **Hyperparameters for SQL query generation**.

Hyperparameter	Value
temperature	0.7 (when sampling multiple queries) 0 (otherwise)
top k	400
top p	1
maximum # of sampling tokens	1000
stop sequence	"`</sql>`"

A.3.2 Prompts for Claude in SQL Query Generation

Claude is trained to generate text as an AI assistant in a dialogue with a human user, so the prompt includes `Human:` and `Assistant:` prefixes to indicate the context. Claude also prefers to work with XML tags which segment different parts of the prompt and help us parse the outputs reliably. Listing 1 is the prompt format we use for Claude to perform the Text-to-SQL task throughout all experiments. In Listing 2 we provide a complete, formatted prompt with a single in-context demonstration and real

Listing 1: **Prompt for Claude**. For zero-shot experiments, the example block is removed. Entities enclosed in {} are input elements inserted at inference time. `{demonstration}` are constructed with the corresponding metadata, question, and query of the selected examples following the same formatting below. `{table_schemas}` are the `CREATE` statements for each table in the database and `{row_inspections}` provide a single sample row for each table in YAML format.

```
Human: Paying careful attention to the table and column names in the given metadata,
↪  provide a correct {dialect} query to answer the given question. Enclose your query
↪  in '<sql></sql>' XML tags.
# example block
Here are some examples:
<example>{demonstration}</example>
.
.
.
# metadata block
Metadata:
<metadata>
{table_schemas}
{row_inspections}
</metadata>
# input question
Question: {question}
Assistant: SQL Query: <sql>
```

sample data.

A.3.3 Modified Execution Accuracy Algorithm

In Algorithm 1, we presents our modified execution accuracy evaluation method which allows for redundant columns to be included in predicted queries.

Algorithm 1 Modified Execution Accuracy (EX†) compares ground-truth query results to columnar permutations (of the same length) of the predicted query results. $P(a, b)$ denotes all permutations of list a with length b, and $[\cdot]$ columnar indexing.

Input: execution engine E, database D, gold query g, predicted query p
Compute gold exec. results $r_g := E(D, g)$
Compute pred. exec. results $r_p := E(D, p)$
$accurate = False$
for c_{p_i} **in** $P\left(cols\left(r_p\right), |cols\left(r_g\right)|\right)$ **do**
  if r_g **equals** $r_p\left[c_{p_i}\right]$ **then**
    $accurate = True$
    break
  end if
end for

Table 8: **Median Query Edit Distance ($M(\text{QED})$) between selected and ground truth examples for selection methods on Spider**. $M(\text{QED})$ generalizes to new domains as it is minimized considerably using MARLO compared to other domain masking methods.

Selection Method	$M(\text{QED})$	
	Spider-dev	**Spider-test**
Zero-shot Baseline	–	–
Random	53.4	62.8
Question Similarity (QS)	50.7	51.0
Masked Question Similarity (MQS)	46.0	50.3
Skills Similarity (SS)	38.2	38.1
MARLO (*ours*)	0.2	0.2

A.4 Additional Data for Result Analysis

A.4.1 Median QED Achieved by Different Selection Methods

Table 8 shows the median QED scores between gold queries and those of the selected demonstrations from different selection methods we experiment. Our encoder minimizes QED and helps MARLO select examples that leads to better end task performance.

From the median QED scores between gold queries and those of the selected demonstrations (Table 8), we observe how QS, MQS and SS are able to retrieve examples that have lower QED scores than random selection, which is correlated with their better end task performance. And that our encoder indeed minimizes QED while selecting examples for MARLO which further enhances the performance.

A.4.2 Comparison of Retrieved Examples

In Table 9 we compare retrieved natural language question and SQL query pairs for a single inference exemplar across selection methods. The metadata-agnostic property of MARLO is clearly demonstrated: structurally similar SQL queries and semantically similar questions are selected, under question phrasing and domains shifts. We expect this helps the LLM understand the given question conditioned on the anticipated query structure. In contrast across the other methods, the question phrasing may be diverse but the query structural less similar than the target, or the question and queries are too diverse but perhaps semantically "close" to the word **"conference"**.

Selection Method	Retrieved Question	Retrieved Query	QED
MARLO	Return the different countries for artists.	`SELECT DISTINCT country FROM artist`	0.0
	Show all distinct building descriptions.	`SELECT DISTINCT building_description FROM Apartment_Buildings`	0.0
	What are the different film Directors?	`SELECT DISTINCT Director FROM film`	0.0
	Show all distinct lot details.	`SELECT DISTINCT lot_details FROM LOTS`	0.0
	Give the distinct headquarters of manufacturers.	`SELECT DISTINCT headquarter FROM manufacturers`	0.0
	What are all the different book publishers?	`SELECT DISTINCT publisher FROM book_club`	0.0
	List all different genre types.	`SELECT DISTINCT name FROM genres;`	0.0
	List the distinct director of all films.	`SELECT DISTINCT Director FROM film`	0.0
Skill Similarity	Show all transaction types.	`SELECT DISTINCT transaction_type FROM Financial_Transactions`	0.0
	Show all video game types.	`SELECT DISTINCT gtype FROM Video_games`	0.0
	What are the different card type codes?	`SELECT DISTINCT card_type_code FROM Customers_Cards`	0.0
	What are the different types of player positions?	`SELECT count(DISTINCT pPos) FROM tryout`	0.3
	What are the different cities where people live?	`SELECT DISTINCT T1.city FROM addresses AS T1 JOIN people_addresses AS T2 ON T1.address_id = T2.address_id`	100.0
	Show all product sizes.	`SELECT DISTINCT product_size FROM Products`	0.0
	What document status codes do we have?	`SELECT document_status_code FROM Ref_Document_Status;`	0.2
	Find all the vocal types.	`SELECT DISTINCT TYPE FROM vocals`	0.0
Masked Question Similarity	What are the different allergy types?	`SELECT DISTINCT allergytype FROM Allergy_type`	0.0
	What are the different card type codes?	`SELECT DISTINCT card_type_code FROM Customers_Cards`	0.0

	What are the different product sizes?	SELECT DISTINCT product_size FROM Products	0.0
	What are the different product colors?	SELECT DISTINCT product_color FROM Products	0.0
	What are the numbers of constructors for different nationalities?	SELECT count(*) , nationality FROM constructors GROUP BY nationality	1.1
	What are the number of different course codes?	SELECT count(DISTINCT crs_code) FROM CLASS	0.3
	What are the different film Directors?	SELECT DISTINCT Director FROM film	0.0
	What are the different membership levels?	SELECT count(DISTINCT LEVEL) FROM member	0.3
	What are the numbers of wines for different grapes?	SELECT count(*) , Grape FROM WINE GROUP BY Grape	1.1
Question Similarity	What is the primary conference of the school that has the lowest acc percent score in the competition?	SELECT t1.Primary_conference FROM university AS t1 JOIN basketball_match AS t2 ON t1.school_id = t2.school_id ORDER BY t2.acc_percent LIMIT 1	100.0
	What are the enrollment and primary conference for the university which was founded the earliest?	SELECT enrollment , primary_conference FROM university ORDER BY founded LIMIT 1	0.9
	What are the names of all teams?	SELECT Name FROM Team	0.2
	What are the names of colleges that have two or more players, listed in descending alphabetical order?	SELECT College FROM match_season GROUP BY College HAVING count(*) >= 2 ORDER BY College DESC	100.0
	What are the details of all organizations that are described as Sponsors and sort the results in ascending order?	SELECT organisation_details FROM Organisations AS T1 JOIN organisation_Types AS T2 ON T1.organisation_type = T2.organisation_type WHERE T2.organisation_type_description = 'Sponsor' ORDER BY organisation_details	100.0
	What are the nicknames of schools whose division is not 1?	SELECT Nickname FROM school_details WHERE Division != "Division 1"	1.0
	What are the different names of the colleges involved in the tryout in alphabetical order?	SELECT DISTINCT cName FROM tryout ORDER BY cName	0.6
	What are the names of the members and branches at which they are registered sorted by year of registration?	SELECT T3.name , T2.name FROM membership_register_branch AS T1 JOIN branch AS T2 ON T1.branch_id = T2.branch_id JOIN member AS T3 ON T1.member_id = T3.member_id ORDER BY T1.register_year	100.0
Random	Count the number of students who have advisors.	SELECT count(DISTINCT s_id) FROM advisor	0.3
	What is the number of aircraft?	SELECT count(*) FROM aircraft	0.5
	What is all the information about employees with D or S in their first name, ordered by salary descending?	SELECT * FROM employees WHERE first_name LIKE '%D%' OR first_name LIKE '%S%' ORDER BY salary DESC	100.0
	What is the id of the reviewer whose name includes the word "Mike"?	SELECT rID FROM Reviewer WHERE name LIKE "%Mike%"	1.0
	What is the average price for wines not produced in Sonoma county?	SELECT avg(price) FROM wine WHERE Appelation NOT IN (SELECT T1.Appelation FROM APPELLATIONS AS T1 JOIN WINE AS T2 ON T1.Appelation = T2.Appelation WHERE T1.County = 'Sonoma')	100.0
	Find the average price of wines that are not produced from Sonoma county.	SELECT avg(price) FROM wine WHERE Appelation NOT IN (SELECT T1.Appelation FROM APPELLATIONS AS T1 JOIN WINE AS T2 ON T1.Appelation = T2.Appelation WHERE T1.County = 'Sonoma')	100.0
	What are the names of all stations that have more than 10 bikes available and are not located in San Jose?	SELECT T1.name FROM station AS T1 JOIN status AS T2 ON T1.id = T2.station_id GROUP BY T2.station_id HAVING avg(bikes_available) > 10 EXCEPT SELECT name FROM station WHERE city = "San Jose"	100.0
	List the names of the customers who have once bought product "food".	SELECT T1.customer_name FROM customers AS T1 JOIN orders AS T2 JOIN order_items AS T3 JOIN products AS T4 ON T1.customer_id = T2.customer_id AND T2.order_id = T3.order_id AND T3.product_id = T4.product_id WHERE T4.product_name = "food" GROUP BY T1.customer_id HAVING count(*) >= 1	100.0

Table 9: **Examples of 8 retrieved demonstrations for different selection methods**. The inference exemplar is the mapping *What are the different conference names?* $\mapsto$ `SELECT DISTINCT conference_name FROM conference`. The QED score between the gold query and each retrieved query is shown as reference. Demonstrations retrieved with MARLO are metadata-agnostic. Although the question phrasing differs slightly and the domains are different, the SQL queries are structurally similar.

Table 10: **Reference GPT 4 execution accuracy (%) with ICL demonstration selection methods from literature**. Results that are unavailable or impossible to reproduce are omitted using ∅. [1] *In cases where our evaluation conflicts with reported results, we reports ours instead.*

LLM	Selection Method	№ demos.	Spider-dev	
			EX	EX†
GPT 4	Baseline (Pourreza and Rafiei, 2023)	0	72.9[1]	77.8
	Baseline (Gao et al., 2023a)	0	72.3	∅
	Random (Chen et al., 2023b)	32	73.2	∅
	Random (An et al., 2023)	4	76.1	∅
	Random (Gao et al., 2023a)	5	79.5	∅
	Random (Pourreza and Rafiei, 2023)	6	76.8[1]	77.8
	Question Similarity (An et al., 2023)	4	76.7	∅
	Question Similarity (Gao et al., 2023a)	5	79.9	∅
	Masked Question Similarity (Gao et al., 2023a)	5	82.0	∅
	Skills Similarity (An et al., 2023)	4	82.7	∅
	DAIL-SQL (Gao et al., 2023a)	9	83.1[1]	83.3[1]

A.4.3 Reported Results for GPT 4 in Text-to-SQL

Table 10 lists execution accuracy of GPT 4 in Text-to-SQL reported by previous studies that implemented similarity-based demonstration selection methods. Note that only results on Spider development set are included in the table since most studies do not report results on the test set. The DAIL-SQL results are shown as the SOTA benchmark.

A.4.4 Breakdown of Results by Query Difficulty

In Figure 5 we compare execution accuracy (**EX**†) across difficult levels on Spider-dev. For both Claude 2.1 and Mistral Large we see question understanding enabled by MARLO plays a pivotal role when generating the more difficult questions.

A.5 Beyond Example Selection

Current state-of-the-art Text-to-SQL systems[8] comprise multiple stages such as schema linking and specialized decoding strategies (Wang et al., 2023c) in addition to ICL with example selection. Hence, we include two additional experiments to assess the effectiveness of MARLO with the most commonly used decoding strategy — self-consistency (SC) — as well as an upper-bound (UB) estimate of the gains we can expect using an optimal strategy. We sample 10 queries using MARLO with a temperature of 0.7 and report results in Figure 6. SC leads to minor performance gains—in-line with results reported in literature. However, we observe a significant unrealized potential (+3-4%pt.) comparing UB and SC across all models. As it is not within the scope of this work, we encourage future work to explore preference optimization or voting strategies to boost LLM performance on this task.

A.6 Additional Training Cost & Inference Latency

It is important to recognize that improved performance on the Text-to-SQL task using general-purpose LLMs often comes at the expense of additional training cost and inference latency. The requirement to learn aligned embeddings by means of an additional fine-tuning process sets this approach apart from most other example selection baselines. However, compared to the approaches that rely on generic embeddings, MARLO is able to achieve considerable performance gains at the expense of minimal training cost overhead by fine-tuning a relatively small encoder. Since the embedding dimension remains the same across all selection methods, MARLO does not incur additional inference latency compared to other example selection methods that rely on vector search. However, compared to other approaches, such as Skill-KNN or DAIL-SQL[8] that require preliminary or additional LLM

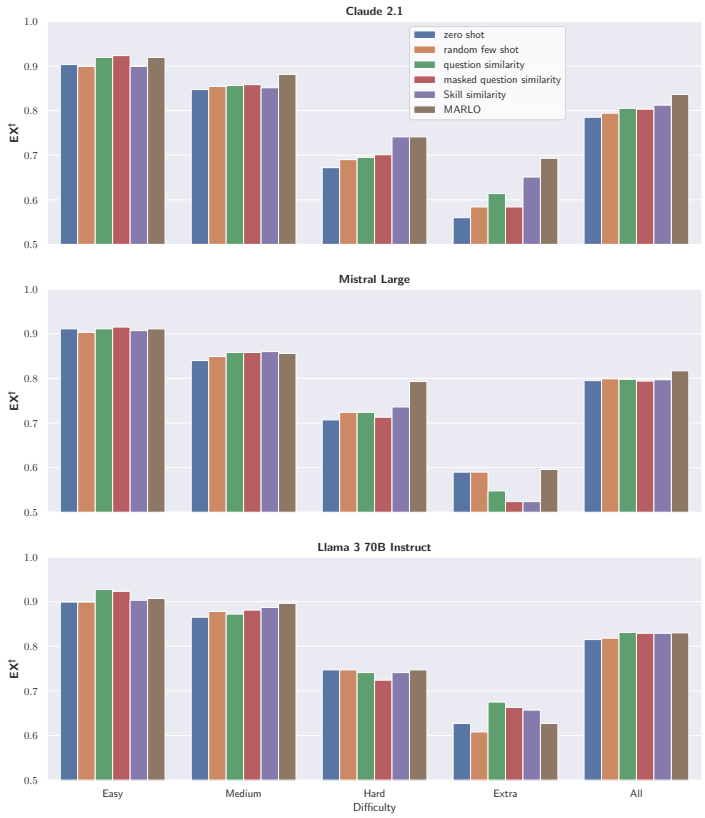


Figure 5: **Execution accuracy (%) on Spider-dev by difficulty level.** With Claude 2.1 and Mistral Large, MARLO outperforms all other demonstration selection methods across all difficulty levels, particularly on more difficult questions, implying the examples to selects contributes to better question understanding in more complex settings.

calls during inference, MARLO is able to achieve comparable performance with considerably lower inference latency.

Figure 6: **Execution accuracy (%) of MARLO sampling 10 predictions**. Self-consistency (SC) uses majority-voted predictions. Upper-bound (UB) uses the best predictions post-hoc. While SC leads to negligible gains, UB implies LLMs possess unrealized potential.

LLM	Voting Method	Spider-dev		Spider-test	
		EX	EX†	EX	EX†
Claude 2.1	-	80.8	83.6	81.2	84.0
	SC	80.7	84.1	81.9	84.5
	UB	86.4	88.3	87.4	88.6
Mistral Large	-	80.0	81.7	81.7	83.2
	SC	81.1	82.8	82.3	83.7
	UB	85.8	87.6	86.1	87.8
Llama 3 70B Instruct	-	82.2	83.0	83.2	83.8
	SC	83.3	84.0	83.7	84.5
	UB	88.0	88.7	87.1	87.9

Listing 2: **Example SQL generation prompt with a single in-context example**. The language model generates the predicted query by autoregressively completing the prompt until the close SQL XML tag (</sql>) is generated. For zero-shot inference, the example blocks and introductory heading are simply omitted.

```
Human: Paying careful attention to the table and column names in the given metadata,
↪  provide a correct SQLite query to answer the given question. Enclose your query in
↪  '<sql></sql>' XML tags.

Here are some examples:

<example>
Metadata:
<metadata>
CREATE TABLE bank (
branch_ID int PRIMARY KEY,
bname varchar(20),
no_of_customers int,
city varchar(10),
state varchar(20))
CREATE TABLE customer (
cust_ID varchar(3) PRIMARY KEY,
cust_name varchar(20),
acc_type char(1),
acc_bal int,
no_of_loans int,
credit_score int,
branch_ID int,
state varchar(20),
FOREIGN KEY(branch_ID) REFERENCES bank(branch_ID))
CREATE TABLE loan (
loan_ID varchar(3) PRIMARY KEY,
loan_type varchar(15),
cust_ID varchar(3),
branch_ID varchar(3),
amount int,
FOREIGN KEY(branch_ID) REFERENCES bank(branch_ID),
FOREIGN KEY(Cust_ID) REFERENCES customer(Cust_ID))

1 sample row from "bank" table:
"""
bname: downtown
```

```
branch_ID: 2
city: Salt Lake City
no_of_customers: 123
state: Utah
"""
1 sample row from "customer" table:
"""
acc_bal: 2000
acc_type: saving
branch_ID: 2
credit_score: 30
cust_ID: '1'
cust_name: Mary
no_of_loans: 2
state: Utah
"""
1 sample row from "loan" table:
"""
amount: 2050
branch_ID: '1'
cust_ID: '1'
loan_ID: '1'
loan_type: Mortgages
"""
</metadata>
Question: Count the number of bank branches.
SQL Query: <sql>SELECT count(*) FROM bank</sql>
</example>

Metadata:
<metadata>
CREATE TABLE Ref_Template_Types (
Template_Type_Code CHAR(15) NOT NULL,
Template_Type_Description VARCHAR(255) NOT NULL,
PRIMARY KEY (Template_Type_Code)
)
CREATE TABLE Templates (
Template_ID INTEGER NOT NULL,
Version_Number INTEGER NOT NULL,
Template_Type_Code CHAR(15) NOT NULL,
Date_Effective_From DATETIME,
Date_Effective_To DATETIME,
Template_Details VARCHAR(255) NOT NULL,
PRIMARY KEY (Template_ID),
FOREIGN KEY (Template_Type_Code) REFERENCES Ref_Template_Types (Template_Type_Code)
)
CREATE TABLE Documents (
Document_ID INTEGER NOT NULL,
Template_ID INTEGER,
Document_Name VARCHAR(255),
Document_Description VARCHAR(255),
Other_Details VARCHAR(255),
PRIMARY KEY (Document_ID),
FOREIGN KEY (Template_ID) REFERENCES Templates (Template_ID)
)
CREATE TABLE Paragraphs (
Paragraph_ID INTEGER NOT NULL,
Document_ID INTEGER NOT NULL,
Paragraph_Text VARCHAR(255),
Other_Details VARCHAR(255),
PRIMARY KEY (Paragraph_ID),
FOREIGN KEY (Document_ID) REFERENCES Documents (Document_ID)
)
```

```
1 sample row from "Ref_Template_Types" table:
"""
Template_Type_Code: CV
Template_Type_Description: CV
"""
1 sample row from "Templates" table:
"""
Date_Effective_From: '1996-02-04 11:27:24'
Date_Effective_To: '1995-09-19 22:27:48'
Template_Details: ''
Template_ID: 11
Template_Type_Code: BK
Version_Number: 6
"""
1 sample row from "Documents" table:
"""
Document_Description: z
Document_ID: 33930
Document_Name: How Google people work
Other_Details: null
Template_ID: 1
"""
1 sample row from "Paragraphs" table:
"""
Document_ID: 651512
Other_Details: null
Paragraph_ID: 243399026
Paragraph_Text: Indonesia
"""
</metadata>
Question: How many paragraphs in total?

Assistant: SQL Query: <sql>
```