STARE at the Structure: Steering ICL Exemplar Selection with Structural Alignment

Anonymous ACL submission

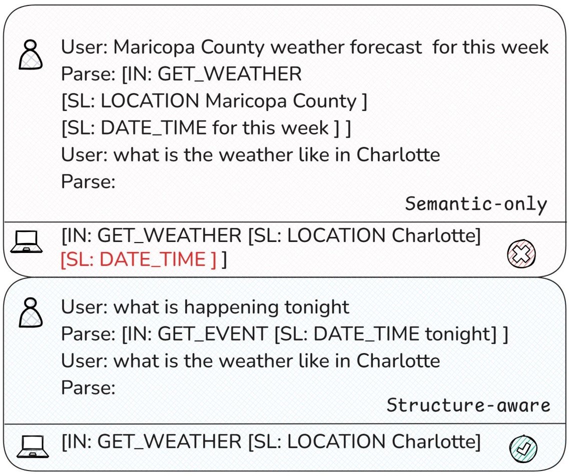
Abstract

In-Context Learning (ICL) has become a powerful paradigm that enables LLMs to perform a wide range of tasks without task-specific fine-tuning. However, the effectiveness of ICL heavily depends on the quality of exemplar selection. In particular, for structured prediction tasks such as semantic parsing, existing ICL selection strategies often overlook structural alignment, leading to suboptimal performance and poor generalization. To address this issue, we propose a novel two-stage exemplar selection strategy that achieves a strong balance between efficiency, generalizability, and performance. First, we fine-tune a BERT-based retriever using structure-aware supervision, guiding it to select exemplars that are both semantically relevant and structurally aligned. Then, we enhance the retriever with a plug-in module, which amplifies syntactically meaningful information in the hidden representations. This plug-in is model-agnostic, requires minimal overhead, and can be seamlessly integrated into existing pipelines. Experiments on four benchmarks spanning three semantic parsing tasks demonstrate that our method consistently outperforms existing baselines with multiple recent LLMs as inference-time models.

1 Introduction

Large language models (LLMs) have demonstrated remarkable few-shot capabilities by leveraging in-context learning (ICL) to perform a new task without parameter updates (Brown et al., 2020). Despite its effectiveness, prior work has shown that ICL performance is highly sensitive to the choice of exemplars (Liu et al., 2022; Rubin et al., 2022; Li and Qiu, 2023). Therefore, how to select meaningful exemplars becomes an active research area.

Exemplar selection methods can be broadly categorized into two aspects: proxy-task-based (Rubin et al., 2022; Shi et al., 2022; Li et al., 2023; Ye et al., 2023) and similarity-based approaches (Das et al., 2021; Hu et al., 2022; An et al., 2023). Proxy-task-based methods extensively query a proxy LLM to evaluate exemplar effectiveness. While effective, they tend to be computationally expensive and often lack generalizability across different models. Similarity-based methods, by contrast, rely on embedding-based metrics to select exemplars that closely match the query instance. However, they typically neglect essential structural information required for precise compositional generalization.



User: Maricopa County weather forecast for this week
Parse: [IN: GET_WEATHER
[SL: LOCATION Maricopa County]
[SL: DATE_TIME for this week]]
User: what is the weather like in Charlotte
Parse:

[IN: GET_WEATHER [SL: LOCATION Charlotte]
[SL: DATE_TIME]]

User: what is happening tonight
Parse: [IN: GET_EVENT [SL: DATE_TIME tonight]]
User: what is the weather like in Charlotte
Parse:

[IN: GET_WEATHER [SL: LOCATION Charlotte]

Figure 1: Comparison between semantic-only and structure-aware similarity based one-shot prompting with LLaMA3-8B.

As a result, existing strategies are suboptimal for structure-intensive tasks such as semantic parsing, which involves translating natural language utterances into structured, machine-executable forms, such as logical queries or database commands (Zelle and Mooney, 1996). These tasks require not only semantic coherence but also precise structural compatibility between exemplars and queries, as illustrated in Figure 1.

Meanwhile, many existing exemplar selection methods implicitly rely on the assumption that the model's learned representations are sufficient for assessing exemplar utility. However, recent interpretability studies suggest that LLM hidden states often encode richer, task-relevant signals than what is directly expressed in their outputs, revealing a gap between internal model knowledge and observable behavior (Wang et al., 2020; Kadavath et al., 2022; Burns et al., 2024).

To address these gaps, we propose **ST**ructure-**A**ware **R**etrieval of **E**xemplars (STARE), a retrieval framework designed for semantic parsing under the ICL paradigm. The framework comprises two key components: 1) a structure-aware retriever that jointly captures both semantic and structural characteristics, and 2) a lightweight plug-in module, Middle-Layer Injection (MLI), that enhances hidden representations with syntactically informative directions. MLI uses linguistic probes and singular value decomposition to identify and amplify syntactic and structural properties in intermediate layers, thereby enhancing the quality of exemplar retrieval. Additionally, the modular design of MLI allows it to be integrated with existing few-shot retrievers, which enables it to enhance semantic parsing performance across diverse scenarios.

To the best of our knowledge, few prior methods for semantic parsing under ICL explicitly incorporate linguistic structure into the retriever's representations to guide exemplar selection. Experimental results on four diverse semantic parsing benchmarks demonstrate that STARE consistently outperforms existing proxy-based and similarity-based methods, while maintaining lower training costs and exhibiting strong generalizability.

Our contributions can be summarized as follows:

- We propose STARE, a structure-aware exemplar selection framework for semantic parsing that integrates both semantic and structural criteria.
- We introduce Middle-Layer Injection (MLI), a lightweight, modular, and model-agnostic technique to enhance hidden representations for improved retrieval.
- The modular design of MLI allows easy integration with diverse retrieval frameworks, thereby improving the generalizability across tasks and models.
- Extensive experiments across four benchmarks demonstrate strong performance with lower training costs compared to proxy-task-based methods.

2 Related Work

ICL for Semantic Parsing Early work on semantic parsing with pre-trained models relied on encoder-decoder architectures augmented with schema-aware modules or constrained decoding to ensure well-formed outputs (Lin et al., 2020; Scholak et al., 2021; Qi et al., 2022). With the advent of stronger models, ICL-based methods emerged. Shin et al. (2021) showed that few-shot prompting with controlled rephrasings could guide models toward canonical forms before parsing, while Pasupat et al. (2021) introduced retrieval-augmented ICL. Later, Shin and Van Durme (2022) demonstrated that instruction-tuned models can perform direct mappings from natural language to structured forms, shifting the research focus toward improving exemplar selection within ICL setups.

Exemplar Selection for ICL Exemplar selection for ICL generally falls into two categories: unsupervised similarity-based and supervised learning-based methods. Unsupervised methods rely on predefined similarity metrics or static retrieval models without task-specific supervision. Common approaches used BM25 or sentence encoders like SBERT to compute semantic similarity between queries and candidate exemplars (Liu et al., 2022; Agrawal et al., 2023). Skill-KNN enhanced this by extracting task-relevant features to identify skill-overlapping exemplars (An et al., 2023). Wu et al. (2023) proposed a self-adaptive selection framework minimizing entropy under the MDL principle. Supervised methods use explicit training signals. Some defined task-specific metrics such as logical-form alignment (Das et al., 2021), slot transitions (Hu et al., 2022), or sparse SQL keyword encoding (Nan et al., 2023). Others trained retrievers with proxy LLM feedback: Efficient Prompt Retrieval (EPR) used binary labels (Rubin et al., 2022), while Unified Demonstration Retriever (UDR) incorporated ranking and hard negative mining (Li et al., 2023). Ye et al. (2023) modeled exemplar selection as subset selection via Determinantal Point Processes (DPPs).

Probing and Representation Intervention Probing is a common technique in LLM interpretability that trains diagnostic classifiers on hidden states to identify encoded linguistic properties and analyze their effects on generation (Adi et al., 2017; Conneau et al., 2018; Liu et al., 2019). A more advanced form, causal probing, intervenes

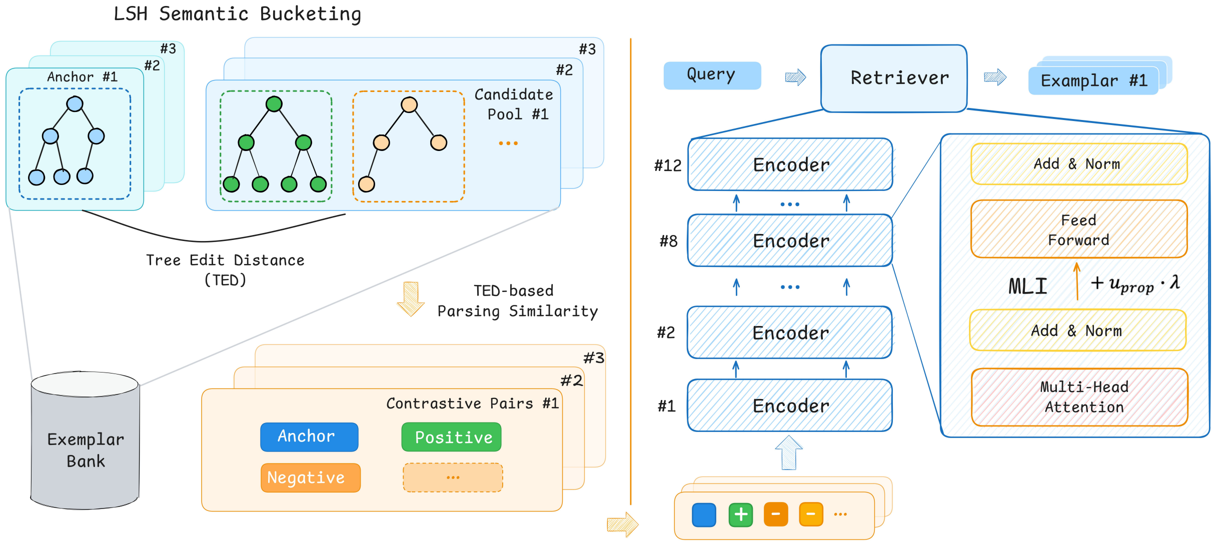


Figure 2: Overview of our proposed framework STARE. The backbone retriever is trained via contrastive learning using semantic and structural similarity signals. The MLI module injects linguistic directions into intermediate hidden states to enhance syntactic awareness.

in hidden representations to create counterfactuals and assess causal influence (Elazar et al., 2021; Ravfogel et al., 2021). While initially designed for analysis, such techniques have increasingly been repurposed to steer model behavior. Interventions on hidden states can affect grammatical agreement (Tucker et al., 2021), reduce bias (Levy et al., 2023), and enable semantic manipulations via vector arithmetic (Subramani et al., 2022). Li et al. (2024) introduced Inference-Time Intervention (ITI), which adjusts attention head activations to promote truthful generation. These findings underscore the potential of hidden-state interventions as a powerful tool for behavior control.

3 Methods

In this section, we introduce the overall methodology of our proposed framework, **ST**ructure-**A**ware **R**etrieval of **E**xemplars (STARE). The overview of STARE is illustrated in Figure 2. We begin by formulating the task in Section 3.1. Section 3.2 describes the backbone component of our framework, a finetuned retriever that jointly models semantic and structural similarity. Section 3.3 then introduces Middle-Layer Injection (MLI), a module that enhances the retriever's syntactic sensitivity by modifying internal representations.

3.1 Task Formulation

ICL enables LLMs to perform semantic parsing by conditioning on a set of exemplars $\mathcal{E} = \{(x_i, y_i)\}_{i=1}^{k}$, where each x_i is an input query and y_i its corresponding gold parse. Given a test input x_{test}, the model predicts an output $\hat{y}_{\text{test}}$ by maximizing the conditional probability:

$$\hat{y}_{\text{test}} = \arg\max_{y} P(y \mid x_{\text{test}}, \mathcal{E}; \theta), \qquad (1)$$

where θ denotes the frozen model parameters. Since ICL performance is sensitive to the exemplar set $\mathcal{E}$, our goal is to optimize its selection.

We aim to construct an effective retriever ϕ that captures both semantic similarity and structural alignment. For each test query x_{test} and candidate x_i from the training set $\mathcal{D}_{\text{train}}$, we compute embeddings $\phi(x_{\text{test}}), \phi(x_i) \in \mathbf{R}^d$, and select the top-$k$ exemplars based on cosine similarity:

$$\mathcal{E} = \text{TopK}\left(\cos(\phi(x_{\text{test}}), \phi(x_i))\right). \qquad (2)$$

3.2 Structure-Aware Retriever

In semantic parsing, both semantic context and structural form carry useful signals for exemplar selection. Semantically related exemplars help an LLM recall the appropriate domain knowledge and surface realizations of a parse, whereas structural correspondence provides the most direct guidance for generating a correct and executable output. Consequently, a two-stage strategy is adopted to construct contrastive pairs for retriever training: a coarse semantic bucketing step first collects a high-recall pool of candidates semantically relevant to the anchor exemplar, followed by an evaluation to distinguish structurally aligned exemplars from misaligned ones within the pool.

3.2.1 Semantic Bucketing

We compute semantic similarity between parsed outputs rather than input utterances, as logical forms and SQL queries more directly reflect compositional meaning. Since conventional off-the-shelf encoders are poorly suited to formal representations, a hashing-based strategy is adopted to group parses into semantically similar candidate pools.

In practice, each parse x is converted into a set of discrete features (e.g., normalized tokens, keywords, argument labels), denoted as $F(x)$, from which compact MinHash sketches (Broder, 1997) are generated, providing efficient and order-invariant approximations of Jaccard similarity. These signatures are stored in a Locality-Sensitive Hashing (LSH) index that enables sublinear retrieval of high-similarity candidates by hashing into multiple overlapping buckets. At query time, the anchor parse's signature is looked up in the LSH index to retrieve all parses with high approximate Jaccard similarity, avoiding exhaustive comparisons against the entire training set.

The LSH index is parameterized by a similarity threshold τ, which defines the target Jaccard similarity above which two parses are likely to collide in at least one bucket. Concretely, given a training instance's semantic parse p and its corresponding feature set $F(p)$, we aim to retrieve q such that

$$\mathrm{Jaccard}(F(p), F(q)) = \frac{|F(p) \cap F(q)|}{|F(p) \cup F(q)|} \geq \tau. \tag{3}$$

This yields a high-recall candidate pool $\mathcal{C}_p = \{q|\mathrm{Jaccard}(F(p), F(q)) \geq \tau\}$ for each parse p that is much smaller than the full dataset, and serves as a strong base for contrastive pair construction in the next stage.

3.2.2 Structure-Based Pair Filtering

Building on the candidate pool, contrastive pairs are next extracted by measuring structural correspondence between parses. The goal is to quantify how closely two compositional representations align in their tree topology.

To this end, each semantic parse is converted into a labeled tree and the normalized Zhang–Shasha tree edit distance (TED) (Zhang and Shasha, 1989) is computed. The way to construct tree structure for semantic parses is detailed in Appendix C.

Formally, let $T(p)$ and $T(q)$ be the trees for parses p and q. We compute $\mathrm{TED}(T(p), T(q))$ using unit edit costs. This distance is then normalized and converted into a similarity score:

$$\mathrm{sim}_{\mathrm{struct}}(p, q) = 1 - \frac{\mathrm{TED}(T(p), T(q))}{\max(|T(p)|, |T(q)|)} \in [0, 1] \tag{4}$$

Higher values indicate closer structural alignment.

For each anchor p with its corresponding candidate pool $\mathcal{C}_p$, the candidate with the highest $\mathrm{sim}_{\mathrm{struct}}$ to the anchor is designated as the positive example, while the least structurally similar candidates within the pool are selected as hard negatives. Additional negatives are randomly sampled from outside the candidate pool. This allows a combination of structure-aware positives and progressively challenging negatives for the contrastive pairs collected.

3.2.3 Training

With the structure-aware contrastive pairs, a BERT is fine-tuned as the exemplar retriever. The model is optimized to bring structurally aligned exemplar–query pairs closer in the representation space, while pushing apart structurally divergent or semantically irrelevant ones. We adopt a contrastive learning objective based on the InfoNCE loss (van den Oord et al., 2019), where each anchor is paired with one positive and multiple negatives. Sentence-level representations are obtained via mean pooling over the final hidden states of the encoder.

3.3 Middle-Layer Injection (MLI)

Recent studies have shown that certain knowledge and properties tend to be attenuated or forgotten as representations progress through deeper layers (Wallat et al., 2021). This raises a critical challenge in exemplar retrieval: while the retriever is fine-tuned using contrastive learning to align with a predefined similarity metric, it is not guaranteed that the final-layer representations optimally encode the most informative signals for exemplar selection.

Probing techniques offer a diagnostic tool for uncovering what features are encoded in intermediate representations, typically by training linear classifiers to predict certain properties from hidden states (Tenney et al., 2019; Hewitt and Manning, 2019). Prior work has explored using probing to identify task-relevant directions in the latent space, for example, directions associated with truthfulness or gender sensitivity. By intervening along these directions, either reinforcing or suppressing them, researchers have been able to increase a model's likelihood of generating truthful responses or reduce biased behavior (Levy et al., 2023; Li et al., 2024).

Inspired by this, we introduce Middle-Layer Injection (MLI), a method that intervenes in the internal representations of the retriever to amplify task-relevant linguistic abstractions, as illustrated in Figure 3. In the absence of ground-truth utility labels for exemplars, we instead extract directions in the model's latent space corresponding to well-established linguistic properties and inject these directions into intermediate layers. By enhancing the retriever's internal encoding of syntactic distinctions, MLI improves the alignment between latent representations and linguistic structure, ultimately leading to more effective exemplar selection.

Concretely, we focus on three widely studied linguistic properties: 1) **Part-of-Speech (POS) Tags**, which identify the syntactic category of each token (e.g., noun, verb), 2) **Dependency Labels (DEPS)**, which define grammatical relationships between words (e.g., subject, object), and 3) **Phrase Types (PT)**, which describe constituent structures (e.g., noun phrase, verb phrase). These properties are chosen because they span fine-grained lexical roles (POS), functional relations (DEPS), and higher-order syntactic structure (PT). Together, they reflect multiple levels of compositional meaning in natural language, making them particularly suitable for enhancing representations used in semantic parsing.

To extract directional signals, auxiliary datasets annotated with the relevant labels are leveraged to train linear probes (logistic regression classifiers) on hidden representations at a chosen layer N. Denote the representation for token w at layer N as $h_N \in \mathbf{R}^d$, a linear probe f_{task} for task $\in \{\text{POS}, \text{DEPS}, \text{PT}\}$ is obtained by minimizing $L(y, \hat{y})$, where $\hat{y} = f_{task}(h_N) = W_{task}h_N$ and y is the ground-truth label for token w. L is the cross-entropy loss function. After training, we denote $\hat{W}_{task} \in \mathbf{R}^{k \times d}$ as the final weight matrix of the classifier, where k is the number of labels.

Singular Value Decomposition (SVD) is then performed on $\hat{W}_{task}$: $\hat{W}_{task} = U\Sigma V^\top$. The first right singular vector V_1 from V is selected as the dominant direction u_{prop} encoding the linguistic property: $u_{\text{prop}} = V_1$.

To amplify this information in the model, the direction u_{prop} is injected at the N-th layer of the retriever by adjusting each token's hidden representation h:

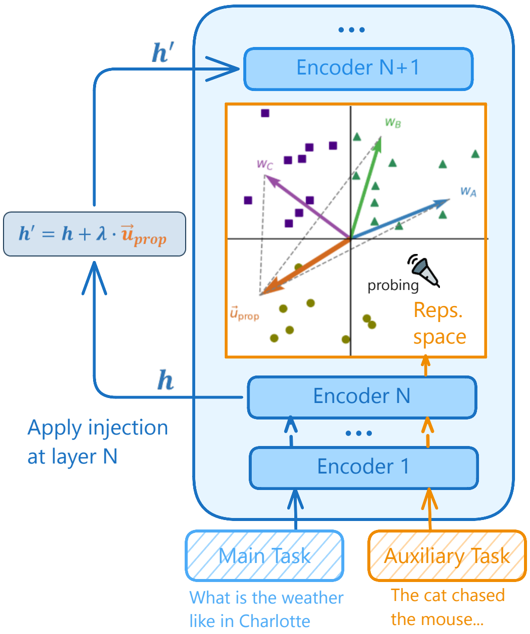

h'
Encoder N+1
$h' = h + \lambda \cdot \vec{u}_{prop}$
w_B
w_C
w_A
$\vec{u}_{\text{prop}}$
probing
Reps. space
h
Encoder N
Apply injection at layer N
Encoder 1
Main Task
What is the weather like in Charlotte
Auxiliary Task
The cat chased the mouse...

Figure 3: Illustration of Middle-Layer Injection (MLI). The vectors W_A, W_B, and W_C are rows of the probe weight matrix W. $\vec{u}_{\text{prop}}$ is the principal direction extracted from W via SVD.

$$h' = h + \lambda\, u_{\text{prop}} \tag{5}$$

where λ controls the intensity of the injection.

The injection layer N, augmentation task (POS, DEPS, or PT), and intensity λ are hyperparameters. The best configuration is selected by evaluating different combinations on a development set, ensuring that the enhancement provides tangible benefits to retrieval performance.

4 Experiments

4.1 Tasks

Our method is evaluated across four semantic parsing tasks, which span from intent and slot filling (MTop (Li et al., 2021)), task-oriented dialogue parsing (SMCalFlow (Andreas et al., 2020), TreeDST (Cheng et al., 2021)) and text-to-SQL (Spider(Yu et al., 2018)). Following standard practice, the training sets of these datasets are used as exemplar banks. Appendix A provides an overview of data splits and examples along with detailed dataset descriptions.

4.2 Baselines

Our method is compared against five recent exemplar selection baselines: Efficient Prompt Re-

Method	MTOP			SMCalFlow			TreeDST		
	LLaMA3	4o-mini	DS-V3	LLaMA3	4o-mini	DS-V3	LLaMA3	4o-mini	DS-V3
BM25	60.6	55.0	72.2	82.6	70.9	61.9	58.1	42.2	34.4
BERT	60.5	53.3	73.9	82.0	73.2	61.9	60.1	45.4	33.9
MLSM	63.6	56.9	73.3	83.0	74.5	61.0	58.1	41.9	33.5
EPR	67.0	58.3	73.3	82.9	73.5	63.5	60.3	**45.5**	36.1
CEIL	68.3	58.8	75.7	84.3	73.8	**63.7**	60.8	44.7	35.9
STARE	**69.5**	**59.4**	**78.8**	**86.9**	**74.8**	61.9	**62.1**	**45.5**	**37.1**

Table 1: Exact Match accuracy on MTop, SMCalFlow and TreeDST across different exemplar retrievers and inference models.

Method	3.5-turbo		4o-mini		DS-V3	
	EX	EM	EX	EM	EX	EM
Zero Shot	71.2	12.1	73.0	19.1	77.0	11.7
Random	73.8	38.6	74.6	50.1	81.7	60.4
BERT	75.5	54.4	74.2	58.4	82.3	70.5
EPR	73.4	48.4	74.7	55.9	82.5	67.0
CEIL	75.9	42.0	74.8	50.4	82.4	66.4
MLSM	75.3	41.6	75.2	59.9	83.4	69.4
Skill-KNN (cons.)	76.3	42.6	75.4	50.3	82.5	66.3
Skill-KNN (dist.)	76.8	43.0	72.9	49.1	82.2	63.2
Similarity-Div.	75.1	42.8	76.2	51.0	82.7	65.3
STARE	**77.0**	**60.3**	**76.9**	**65.4**	**84.5**	**74.0**

Table 2: Execution (EX) and Exact Match (EM) accuracy on the Spider dataset across different exemplar retrievers and inference models.

triever (EPR) (Rubin et al., 2022), Compositional Exemplars for In-context Learning (CEIL) (Ye et al., 2023), Multi-level Similarity Maximization (MLSM) (Liu et al., 2024), Skill-KNN (An et al., 2023), and Similarity-Diversity (Nan et al., 2023).

EPR and CEIL utilize proxy tasks that incorporate LLM feedback to assess the utility of exemplars. MLSM aggregates similarity signals across BERT layers as expert representations. Skill-KNN and Similarity-Diversity are tailored for text-to-SQL tasks. In addition, unsupervised retrieval baselines such as BM25 and BERT-based dense retrievers are included. Detailed descriptions and implementation details are provided in Appendix B.

4.3 Experimental Settings

Backbone Retriever A BERT encoder[1] is fine-tuned with InfoNCE loss (temperature 0.07) for at most three epochs using AdamW. For each anchor, we construct one positive pair, three hard negative pairs, and two random negative pairs.

Middle-Layer Injection For the auxiliary datasets, we use the English Universal Dependencies (UD) Treebank (McDonald et al., 2013) for part-of-speech (POS) and syntactic dependency (DEPS) labels, and the Penn Treebank (Marcus et al., 1993) for constituency parsing (Phrase Type, PT) labels. To mitigate overfitting, fine-grained labels are merged into broader categories; the final label sets are summarized in Table 6 in Appendix D. The selected properties and intensities under different settings are listed in Appendix G.

Inference LLMs For MTop, SMCalFlow, and TreeDST, LLaMA3-8B (Grattafiori et al., 2024), GPT-4o-mini (OpenAI, 2024), and DeepSeek-V3 (DeepSeek-AI et al., 2025) are used as inference models. For Spider, the same setting is used except that LLaMA3-8B is replaced with GPT-3.5-turbo (OpenAI, 2023). Detailed settings are provided in Appendix E.

Prompt Construction Following existing work, we use 20 exemplars for MTop, 5 for SMCalFlow, 10 for TreeDST, and 5 for Spider, ordered by ascending similarity to the test query. The prompts used for Spider incorporate schema linking in the format proposed by Nan et al. (2023), and adopt the system prompt from Lee et al. (2025). Full prompt templates and examples are provided in Appendix F.

Evaluation Exact Match (EM) is reported for all tasks, while Execution Accuracy (EX) is additionally reported for Spider, following the official evaluation script[2].

5 Results

5.1 Main Results

We report the main ICL performance of our proposed framework STARE across the four semantic parsing benchmarks. Results for MTop, SMCalFlow, and TreeDST are summarized in Table 1,

[1] https://huggingface.co/bert-base-uncased

[2] https://github.com/taoyds/test-suite-sql-eval.

Method	MTOP			SMCalFlow			TreeDST		
	LLaMA3	4o-mini	DS-V3	LLaMA3	4o-mini	DS-V3	LLaMA3	4o-mini	DS-V3
STARE (w/o MLI)	67.8	57.5	74.5	84.3	71.6	63.0	61.0	43.8	36.2
STARE	69.5	59.4	78.8	86.9	74.8	61.9	62.1	45.5	37.1
Δ (MLI Gain)	+1.7	+1.9	+4.3	+2.6	+3.2	-1.1	+1.1	+1.7	+0.9
BERT	60.5	53.3	73.9	82.0	73.2	61.9	60.1	45.4	33.9
BERT + MLI	64.4	58.7	76.2	87.5	73.9	62.2	61.6	46.1	35.7
Δ (MLI Gain)	+3.9	+5.4	+2.3	+5.5	+0.7	+0.3	+1.5	+0.7	+1.8
EPR	67.0	58.3	73.3	82.9	73.5	63.5	60.3	45.5	36.1
EPR + MLI	67.4	60.6	75.7	86.3	75.2	62.5	60.6	46.3	40.0
Δ (MLI Gain)	+0.4	+2.3	+2.4	+3.4	+1.7	-1.0	+0.3	+0.8	+3.9

Table 3: Effect of MLI on Exact Match accuracy across different exemplar retrievers and inference models on MTop, SMCalFlow and TreeDST.

Method	3.5-turbo		4o-mini		DS 3	
	EX	EM	EX	EM	EX	EM
STARE (w/o MLI)	76.5	57.0	76.8	60.7	83.0	72.1
STARE	77.0	60.3	76.9	65.4	84.5	74.0
Δ (MLI Gain)	+0.5	+3.3	+0.1	+4.7	+1.5	+1.9
BERT	75.5	54.4	74.2	58.4	82.3	70.5
BERT + MLI	75.6	56.3	74.3	61.1	83.9	72.0
Δ (MLI Gain)	+0.1	+1.9	+0.1	+2.7	+1.6	+1.5
EPR	73.4	48.4	74.7	55.9	82.5	67.0
EPR + MLI	76.6	57.1	75.5	60.7	83.8	73.0
Δ (MLI Gain)	+3.2	+8.7	+0.8	+4.8	+1.3	+6.0

Table 4: Effect of MLI on Execution (EX) and Exact Match (EM) accuracy across different exemplar retrievers and inference models on the Spider dataset.

while Table 2 presents Execution accuracy (EX) and Exact Match (EM) on the Spider dataset.

Our method STARE consistently outperforms all baselines, including proxy-task-based methods such as EPR and CEIL, except on SMCalFlow under DeepSeek-V3. On average, STARE yields a 1.35% gain over the strongest competing baseline across the first three tasks. On Spider, STARE achieves the best EX and EM across all inference models, improving over the best baseline by 0.9% (EX) and 5.0% (EM).

In contrast to EPR and CEIL, which depend on intensive LLM interactions to derive training supervision, STARE avoids reliance on the inference model during training. This not only improves efficiency, but also mitigates overfitting to biases introduced by the proxy model, which can compromise generalization to stronger inference models.

5.2 MLI as a Plug-in

To assess the contribution of Middle-Layer Injection (MLI) within our STARE framework, we first compare STARE with and without MLI to isolate the effect of linguistic augmentation. Furthermore, to examine the generalizability of MLI beyond our framework, MLI is applied as a plug-in module to two representative baselines: an unsupervised BERT retriever and the supervised Efficient Prompt Retriever (EPR). Results are shown in Table 3 (MTop, SMCalFlow, TreeDST) and Table 4 (Spider). MLI configuration details are listed in Appendix G.

Experimental results across MTop, SMCalFlow, and TreeDST show that MLI enhances STARE performance under all three inference LLMs, with an average improvement of 2.2%. The only exception is SMCalFlow when using DeepSeek-V3, where performance slightly declines. On the Spider dataset, MLI provides an average gain of 0.7% in execution accuracy and 3.3% in exact match.

Notably, MLI also improves retrieval performance when integrated with both BERT and EPR retrievers. In some cases, MLI enables a non-fine-tuned BERT retriever to achieve performance comparable to fully supervised retrievers, for example, on SMCalFlow with LLaMA3-8B and TreeDST with GPT-4o-mini. On the Spider dataset, MLI achieves up to an 8.7% boost in exact match when combined with EPR under GPT-3.5-turbo.

These results underscore MLI's versatility as a modular plug-in. It can be applied to both unsupervised and supervised retrievers and is especially useful in low-resource scenarios where retriever training is infeasible.

6 Ablation Study

To investigate how the choice of injection layer, the augmented linguistic property, and the augmentation intensity λ affect performance, a series of ablation studies are conducted by systematically varying these factors and analyzing their impact.

Injection Layer Injecting MLI into all 12 BERT layers is computationally costly and often redun-

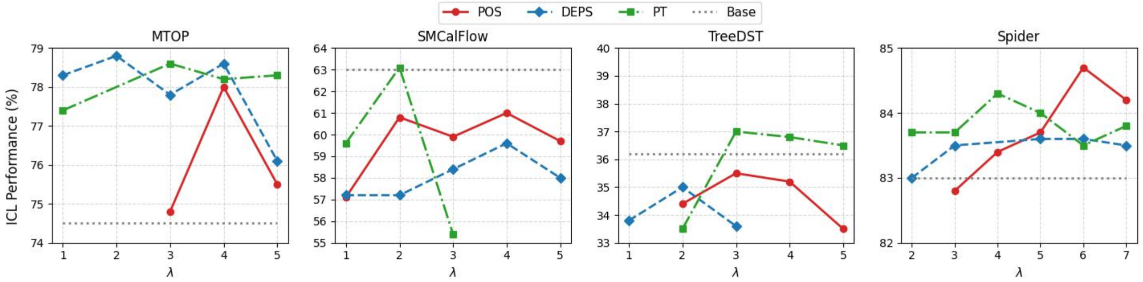


Figure 4: ICL performance on the dev sets of MTop, SMCalFlow, and TreeDST (reported as Exact Match accuracy), and Spider (reported as Execution accuracy), under varying MLI injection intensities λ for each linguistic property, using DeepSeek-V3 as the inference model.

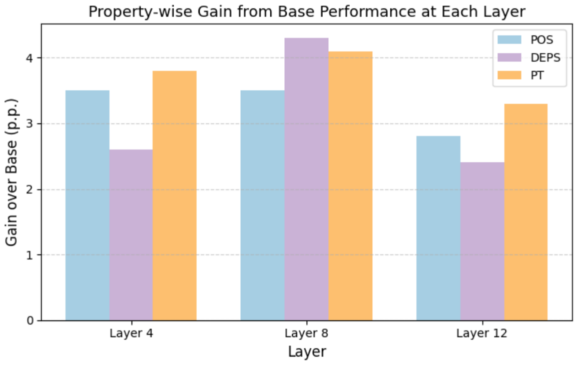


Figure 5: Effect of applying MLI at different injection layers on MTop performance, using DeepSeek-V3 as the inference model.

dant due to high inter-layer similarity from residual connections. To reduce overhead, we draw on BERTology findings that different layers capture distinct linguistic patterns: lower layers (1–4) encode lexical features, middle layers (5–9) capture syntax, and upper layers (10–12) model semantics and task-specific abstractions (Ethayarajh, 2019; Tenney et al., 2019; Jawahar et al., 2019). Based on this, Layers 4, 8, and 12 are pre-selected as injection candidates.

For each layer, we apply property-specific injections (POS, DEPS, PT) across a range of intensity values ($\lambda = 1, 2, 3, 4, 5$), and record the maximum performance gain. Figure 5 shows the results on MTop using DeepSeek-V3.

The injection at Layer 8 consistently achieves the highest gains across all properties. We attribute this to its intermediary position, allowing injected signals to propagate effectively while still benefiting from well-encoded linguistic features. In contrast, early injection may dilute the signal, and late injection may limit downstream utilization.

MLI Configurations To determine the most effective linguistic property for injection and the optimal intensity λ, controlled experiments are conducted on the development set. For each of the three properties (POS, DEPS, and PT), we sweep over a range of λ values and observe the resulting ICL performance. Figure 4 illustrates this tuning process when DeepSeek-V3 is used as the inference model[3]. The dashed line represents the performance of our backbone retriever without MLI. Typically, we observe that the characteristic trend of contributing properties is an initial increase followed by a decline as λ grows. We select the configuration that achieves the highest development performance for each task and apply it in test-time evaluation.

The observed trend indicates that there is an optimal injection strength. A small λ may fail to meaningfully influence the representation, while overly aggressive injection risks disrupting the structural integrity of the latent space.

7 Conclusion

We present Structure-Aware Retrieval of Exemplars (STARE), a novel framework for in-context learning exemplar retrieval for semantic parsing. STARE combines a backbone retriever based on semantic hashing and dependency tree representations with a modular enhancement strategy, Middle-Layer Injection (MLI). MLI serves as a lightweight yet effective augmentation mechanism, which can also be integrated into various retrieval pipelines. Our method achieves state-of-the-art performance across multiple semantic parsing benchmarks, while maintaining low training costs and demonstrating strong generalizability.

[3] For optimal presentation, we omit results for those λ falling out of the range in the figure

Limitations

Our Middle-Layer Injection (MLI) method assumes a linear mechanism of injecting linguistic properties, treating each direction independently in the hidden space. This simplification may overlook potential nonlinear interactions, such as those between part-of-speech tags and syntactic dependencies, which may be important in certain retrieval scenarios and warrant further investigation in future work. In addition, our evaluation focuses solely on exemplar selection, without comparisons to reasoning-centric methods such as chain-of-thought prompting or to fine-tuning-based approaches, which may limit the scope of our current analysis to retrieval-specific settings.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks.

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*.

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*.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, and Jean et al. Crawford. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8.

Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, pages 21–29. IEEE Computer Society.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2024. Discovering latent knowledge in language models without supervision. *Preprint*, arXiv:2212.03827.

Jianpeng Cheng, Devang Agrawal, Hector Martinez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Shaona Ghosh, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid O Seaghdha, and Anders Johannsen. 2021. Conversational semantic parsing for dialog state tracking.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

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*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, and Ethan Perez et al. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Jimin Lee, Ingeol Baek, Byeongjeong Kim, and Hwanhee Lee. 2025. Safe-sql: Self-augmented in-context learning with fine-grained example selection for text-to-sql. *Preprint*, arXiv:2502.11438.

Tal Levy, Omer Goldman, and Reut Tsarfaty. 2023. Is probing all you need? indicator tasks as an alternative to probing embedding spaces. *Preprint*, arXiv:2310.15905.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Preprint*, arXiv:2306.03341.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing.

Hui Liu, Wenya Wang, Hao Sun, Chris Xing Tian, Chenqi Kong, Xin Dong, and Haoliang Li. 2024. Unraveling the mechanics of learning-based demonstration selection for in-context learning. *Preprint*, arXiv:2406.11890.

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)*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, and Slav Petrov. 2013. Universal dependencies: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, pages 1659–1666. European Language Resources Association (ELRA).

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. 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*.

OpenAI. 2023. Gpt-3.5 turbo. `https://platform.openai.com/models/gpt-3.5-turbo`. Accessed: 2025-05-12.

OpenAI. 2024. Gpt-4o-mini. `https://platform.openai.com/models/gpt-4o-mini`. Accessed: 2025-05-12.

Panupong Pasupat, Yuan Zhang, and Kelvin Guu. 2021. Controllable semantic parsing via retrieval augmentation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Emmanouil Antonios Platanios, Adam Pauls, Subhro Roy, Yuchen Zhang, Alexander Kyte, Alan Guo, Sam Thomson, Jayant Krishnamurthy, Jason Wolfe, Jacob Andreas, and Dan Klein. 2021. Value-agnostic conversational semantic parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Shauli Ravfogel, Grusha Prasad, Tal Linzen, and Yoav Goldberg. 2021. Counterfactual interventions reveal the causal effect of relative clause representations on agreement prediction. In *Proceedings of the 25th Conference on Computational Natural Language Learning*.

Stephen Robertson and Hugo Zaragoza. 2009. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

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*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. 2022. XRICL: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Richard Shin and Benjamin Van Durme. 2022. Few-shot semantic parsing with language models trained on code. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline.

Mycal Tucker, Peng Qian, and Roger Levy. 2021. What if this modified that? syntactic interventions with counterfactual embeddings. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 862–875, Online. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Jonas Wallat, Jaspreet Singh, and Avishek Anand. 2021. Bertnesia: Investigating the capture and forgetting of knowledge in bert. *Preprint*, arXiv:2106.02902.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *Preprint*, arXiv:2010.11967.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. *Preprint*, arXiv:2212.10375.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning.

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*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

A Datasets

Table 5 presents the data splits and example instances for each dataset. Detailed descriptions are provided below.

MTop MTop (Li et al., 2021) is a multilingual task-oriented semantic parsing dataset spanning diverse user intents and domains. We focus on the English subset. We use the full training set and sample 1K examples each from the original validation and test sets to form our splits.

SMCalFlow SMCalFlow (Andreas et al., 2020) is a conversational semantic parsing dataset serialized in a LISP-style format (Lispress). Its structured representation enforces well-formedness and supports generalization in low-data settings, making it ideal for testing compositional generalization in dialogue. We sample 5K/1K/1K examples from its original training, validation and test datasets as our splits.

TreeDST TreeDST (Cheng et al., 2021) is a task-oriented dialogue dataset representing dialogue states as hierarchical trees. We use its Lispress serialization version (Platanios et al., 2021), which captures compositional dependencies across intents, domains, and slots, better reflecting real-world dialogue complexity. We sample 5K/1K/1K examples from its original training, validation and test datasets as our splits.

Spider Spider (Yu et al., 2018) is a large-scale text-to-SQL dataset. It covers a wide range of domains and compositional SQL structures, providing a rigorous testbed for text-to-SQL exemplar selection methods. Its different splits do not share any databases. Following standard practice, we evaluate on the development set since the test set is not publicly released.

B Baselines

Efficient Prompt Retriever (EPR) EPR (Rubin et al., 2022) constructs contrastive training pairs by applying one-shot prompting to every training instance, using a proxy LLM to approximate exemplar utility. Specifically, a simple retriever is first used to retrieve a candidate pool, and top-K and bottom-K candidates are selected based on proxy model scores. A BERT-based retriever is then fine-tuned via contrastive learning. Following the original setup, we use GPT-Neo (2.7B) as the proxy model. We then retrieve 15 candidates for each training instance using BM25, and set $K = 5$ for pair construction.

Compositional Exemplars for In-context Learning (CEIL) CEIL (Ye et al., 2023) builds on the EPR framework but improves selection granularity by modeling exemplar interactions. Instead of scoring one-shot prompts individually, it adopts Determinantal Point Processes (DPP) to select exemplar subsets that jointly maximize compositional contribution. As with EPR, we use BM25 for candidate pre-selection, GPT-Neo (2.7B) as the scoring model, and score 10 randomly sampled subsets of 16 examples for each training instance.

Multi-level Similarity Maximization (MLSM) MLSM (Liu et al., 2024) proposes to leverage different abstraction levels captured across BERT layers for exemplar selection. Redundant layers are first filtered using CKA-based clustering, and each selected layer acts as an expert capturing similarity at a distinct level. For each query, MLSM aggregates similarity scores from multiple layers with learned weights, optimizing for agreement across experts. The method is fully unsupervised and designed to enhance task-agnostic generalization without relying on task-specific labels.

Skill-KNN Skill-KNN (An et al., 2023) proposes a two-stage retrieve framework for text-to-SQL in-context learning. First, for each query, a frozen LLM is prompted to rewrite the input into a skill-based description that captures task-relevant features in natural language. These rewritten skills are then embedded with an off-the-shelf encoder, and examples with similar skills are retrieved. To address the sensitivity of rewriting to prompt order, two variants are proposed: a consistency-based variant, which aggregates multiple rewrites via mean pooling, and a distinctiveness-based variant, which selects based on the most distinctive match. In our experiments, we use GPT-4o-mini to generate 5 candidate skill descriptions per input and evaluate both variants. We use `bert-base-uncased` as the off-the-shelf encoder for embedding the skill descriptions.

Similarity-Diversity Similarity-Diversity (Nan et al., 2023) proposes selecting exemplars by balancing similarity and diversity among demonstrations. First, candidates are filtered by retrieving examples with similar SQL structure complexity, using the difficulty-level categorization from the Spider dataset. Then, to promote diversity, a sparse

Dataset	Train/Dev/Test	Example
MTop	15,567/1,000/1,000	**Sentence:** Whats weather forecast for tomorrow? **Parsing:** [IN:GET_WEATHER [SL:DATE_TIME for tomorrow]]
SMCalFlow	50,000/1,000/1,000	**Sentence:** What does my schedule look like on Thursday? **Parsing:** (Yield (FindEventWrapperWithDefaults (EventOnDate (NextDOW (Thursday)) (^(Event) EmptyStructConstraint))))
TreeDST	50,000/1,000/1,000	**Sentence:** Hi my assistant, where is the Westin hotel? **Parsing:** (plan (^(Hotel) Find :focus (Hotel.location_? (^(String) always)) :object (Hotel.hotelName_? (?= "Westin"))))
Spider	7,000/1,000/1,034	**Sentence:** How many available features are there in total? **Parsing:** SELECT count(*) FROM Other_Available_Features

Table 5: Overview of datasets used for semantic parsing experiments.

encoding of the predicted SQL query is computed, and k-means clustering is applied over the discrete representations to select diverse exemplars. In the original setup, an approximate SQL prediction by baseline text-to-SQL models is used for difficulty categorization and sparse encoding. To avoid introducing noise from imperfect preliminary models, we instead use the ground-truth SQL queries for encoding in our experiments, representing an upper-bound variant of this method.

BM25 Retriever BM25 (Robertson and Zaragoza, 2009) is a sparse retrieval baseline that scores exemplar candidates by computing lexical similarity with the test query. Specifically, it compares each candidate's input utterance to the test query using term frequency and inverse document frequency, and selects the Top-K highest-scoring candidates.

BERT Retriever The BERT retriever encodes both exemplars and test queries using a pre-trained BERT model and selects those with the highest cosine similarity in embedding space. Despite being unsupervised, it captures richer semantic signals than token-level matching methods like BM25, and serves as a lightweight neural baseline for retrieval.

C Tree Construction

To enable structure-based similarity computation, we convert each semantic parse into a labeled tree.

For MTop, SMCalFlow and TreeDST, we use a bracket-based parser to recursively construct trees, where each non-terminal label becomes a parent node, and its enclosed spans are attached as children. The resulting tree captures the hierarchical structure of the parse.

For SQL Queries (Spider), we parse SQL queries into Abstract Syntax Trees (ASTs) using `sqlglot`, and prune them by retaining only clause-level nodes and essential fields to form a structural skeleton.

In both cases, each node's label corresponds to its operator or clause type, and children reflect its compositional arguments.

Property	Final Merged Labels
POS	ADJ, ADP, ADV, AUX, CCONJ, DET, INTJ, NOUN, NUM, PART, PRON, PROPN, PUNCT, SCONJ, SYM, VERB, X
DEPS	ACL, ACL:RELCL, ADVMOD, AUX, CASE, COMP, COMPOUND, CONJ, CSUBJ, DEP, DET, EXPL, GOESWITH, LIST, MARK, MOD, NMOD, NSUBJ, OBJ, OBL, ORPHAN, PUNCT, REPARANDUM, ROOT, VOCATIVE
PT	SBAR, UCP, ADVP, O, WHADVP, NAC, INTJ, NX, CONJP, QP, SBARQ, S, ADJP, FRAG, SQ, LST, PRT, PP, X-HLN, VP, X, WHADJP, WHPP, NP, WHNP, SINV, PRN

Table 6: Final label sets used for linguistic probing after merging fine-grained categories.

D Final Merged Labels of Auxiliary Datasets

Table 6 shows the merged linguistic labels for the three properties in the auxiliary datasets. Specifically, the POS and DEPS labels are derived from the UD Treebank (McDonald et al., 2013), which contains 207,230 tokens, while the PT labels are sourced from the Penn Treebank (Marcus et al., 1993), comprising 100,676 tokens.

E Inference Settings

For all experiments, we use greedy decoding, with the temperature set to 0.0 to ensure deterministic generation. The maximum generation length is capped at 200 new tokens, excluding the input prompt. To ensure robustness, each experiment is run with three different random seeds. The final reported results are obtained by averaging over these three runs.

F Prompt Examples

F.1 Prompt Example for MTop, SMCalFlow, TreeDST

(Same prompt template is used for MTop, SMCalFlow and TreeDST while the following example is instantiated with MTop)

Below are examples of converting user utterances into MTop semantic parses:

Example 1

User: Remind me about shopping for school on tax free weekend.
Parse: [IN:CREATE_REMINDER [SL:PERSON_REMINDED me] [SL:TODO shopping for school] [SL:DATE_TIME on tax free weekend]]

Example 2

User: Remind me to bake cookies tomorrow night for the bake sale
Parse: [IN:CREATE_REMINDER [SL:PERSON_REMINDED me] [SL:TODO [IN:GET_TODO [SL:TODO bake cookies] [SL:DATE_TIME tomorrow night] [SL:TODO the bake sale]]]]

Example 3

User: Remind me to tell Angie I am bringing the salad for bible study on Friday
Parse: [IN:CREATE_REMINDER [SL:PERSON_REMINDED me] [SL:TODO tell Angie I am bringing the salad for bible study] [SL:DATE_TIME on Friday]]

. . .

Example 19

User: Remind me to make chicken dip for the watch party tomorrow.
Parse: [IN:CREATE_REMINDER [SL:PERSON_REMINDED me] [SL:TODO make chicken dip for the watch party] [SL:DATE_TIME tomorrow]]

Example 20

User: Remind me to make the cookies for the bake sale.
Parse: [IN:CREATE_REMINDER [SL:PERSON_REMINDED me] [SL:TODO make the cookies for the bake sale]]

Query

User: Remind me to make bars for the picnic on Sunday.
Parse:

F.2 Prompt Example for Spider

Below are examples of database schema and text-to-SQL generation for Spider:

/* Given the following database schema: */

```
CREATE TABLE IF NOT EXISTS "flight" ( "flno" text, "origin" text, "destination" text, "distance" text, "departure_date" text, "arrival_date" text, "price" text, "aid" text, PRIMARY KEY ("flno"), FOREIGN KEY ("aid") REFERENCES "aircraft"("aid") );
CREATE TABLE IF NOT EXISTS "aircraft" ( "aid" text, "name" text, "distance" text, PRIMARY KEY ("aid") );
CREATE TABLE IF NOT EXISTS "employee" ( "eid" text, "name" text, "salary" text, PRIMARY KEY ("eid") );
CREATE TABLE IF NOT EXISTS "certificate" ( "eid" text, "aid" text, PRIMARY KEY ("eid"), FOREIGN KEY ("aid") REFERENCES "aircraft"("aid"), FOREIGN KEY ("eid") REFERENCES "employee"("eid") );
```

/* Answer the following: How many employees do we have? */

SQL Query: `SELECT count(*) FROM employee;`

/* Given the following database schema: */

```
CREATE TABLE IF NOT EXISTS "Activity" ( "actid" text, "activity_name" text, PRIMARY KEY ("actid") );
CREATE TABLE IF NOT EXISTS "Participates_in" ( "stuid" text, "actid" text, FOREIGN KEY ("actid") REFERENCES "Activity"("actid"), FOREIGN KEY ("stuid") REFERENCES "Student"("StuID") );
CREATE TABLE IF NOT EXISTS "Faculty_Participates_in" ( "FacID" text, "actid" text, FOREIGN KEY ("actid") REFERENCES "Activity"("actid"), FOREIGN KEY ("FacID") REFERENCES "Faculty"("FacID") );
CREATE TABLE IF NOT EXISTS "Student" ( "StuID" text, "LName" text, "Fname" text, "Age" text, "Sex" text, "Major" text, "Advisor" text, "city_code" text, PRIMARY KEY ("StuID") );
CREATE TABLE IF NOT EXISTS "Faculty" ( "FacID" text, "Lname" text, "Fname" text, "Rank" text, "Sex" text, "Phone" text, "Room" text, "Building" text, PRIMARY KEY ("FacID") );
```

/* Answer the following: How many faculty do we have? */

SQL Query: `SELECT count(*) FROM Faculty;`

...

/* Given the following database schema: */

```
CREATE TABLE IF NOT EXISTS "artist" ( "Artist_ID" text, "Name" text, "Country" text, "Year_Join" text, "Age" text, PRIMARY KEY ("Artist_ID") );
CREATE TABLE IF NOT EXISTS "exhibition" ( "Exhibition_ID" text, "Year" text, "Theme" text, "Artist_ID" text, "Ticket_Price" text, PRIMARY KEY ("Exhibition_ID"), FOREIGN KEY ("Artist_ID") REFERENCES "artist"("Artist_ID") );
CREATE TABLE IF NOT EXISTS "exhibition_record" ( "Exhibition_ID" text, "Date" text, "Attendance" text, PRIMARY KEY ("Exhibition_ID"), FOREIGN KEY ("Exhibition_ID") REFERENCES "exhibition"("Exhibition_ID") );
```

/* Answer the following: How many artists do we have? */

SQL Query: `SELECT count(*) FROM artist;`

/* Given the following database schema: */

```
CREATE TABLE IF NOT EXISTS "stadium" ( "Stadium_ID" text, "Location" text, "Name" text, "Capacity" text, "Highest" text, "Lowest" text, "Average" text, PRIMARY KEY
("Stadium_ID") );
CREATE TABLE IF NOT EXISTS "singer" ( "Singer_ID" text, "Name" text, "Country" text, "Song_Name" text, "Song_release_year" text, "Age" text, "Is_male" text, PRIMARY KEY ("Singer_ID") );
CREATE TABLE IF NOT EXISTS "concert" ( "concert_ID" text, "concert_Name" text, "Theme" text, "Stadium_ID" text, "Year" text, PRIMARY KEY ("concert_ID"), FOREIGN KEY ("Stadium_ID") REFERENCES "stadium"("Stadium_ID") );
CREATE TABLE IF NOT EXISTS "singer_in_concert" ( "concert_ID" text, "Singer_ID" text, PRIMARY KEY ("concert_ID"), FOREIGN KEY ("Singer_ID") REFERENCES "singer"("Singer_ID"), FOREIGN KEY ("concert_ID") REFERENCES "concert"("concert_ID") );
```

/* Answer the following: How many singers do we have? */

SQL Query:

Inference Model	**Method + MLI**	**MTOP**	**SMCalFlow**	**TreeDST**	**Spider**
LLaMA3-8B	STARE	PT-5	DEPS-4	DEPS-1.5	–
	BERT + MLI	POS-5	PT-1.5	DEPS-1.5	–
	EPR + MLI	POS-6	PT-4	DEPS-4	–
GPT-3.5-turbo	STARE	–	–	–	POS-5
	BERT + MLI	–	–	–	POS-5
	EPR + MLI	–	–	–	POS-5
GPT-4o-mini	STARE	POS-6	PT-2.5	PT-0.5	POS-5
	BERT + MLI	POS-5	DEPS-3	DEPS-1.5	POS-5
	EPR + MLI	DEPS-6	DEPS-3	DEPS-1.5	POS-5
DeepSeek-V3	STARE	POS-4	PT-2	PT-3	POS-6
	BERT + MLI	POS-5	PT-0.5	PT-2	POS-5
	EPR + MLI	DEPS-6	PT-2	PT-2	POS-5

Table 7: MLI configurations (property - intensity) for each inference model across datasets. "–" indicates the method is not evaluated for that task-model pair.

G MLI Configuration

Table 7 details the MLI configurations (linguistic property and injection intensity) applied to different exemplar selection methods for each dataset under various inference models. Configurations marked with "–" indicate settings that were not evaluated. All injections are applied at Layer 8 of the base retriever.