Learning to Retrieve Iteratively for In-Context Learning

Yunmo Chen[†]   **Tongfei Chen**   **Harsh Jhamtani**   **Patrick Xia**   **Richard Shin**[‡]
Jason Eisner   **Benjamin Van Durme**
Microsoft

yunmo@jhu.edu, {tongfeichen, hjhamtani, patrickxia, jeisner, ben.vandurme}@microsoft.com

Abstract

We introduce *iterative retrieval*, a novel framework that empowers retrievers to make iterative decisions through *policy optimization*. Finding an optimal portfolio of retrieved items is a combinatorial optimization problem, generally considered NP-hard. This approach provides a learned approximation to such a solution, meeting specific task requirements under a given family of large language models (LLMs). We propose a training procedure based on reinforcement learning, incorporating feedback from LLMs. We instantiate an iterative retriever for composing in-context learning (ICL) exemplars and apply it to various semantic parsing tasks that demand synthesized programs as outputs. By adding only 4M additional parameters for state encoding, we convert an off-the-shelf dense retriever into a stateful iterative retriever, outperforming previous methods in selecting ICL exemplars on semantic parsing datasets such as SMCALFLOW, TREEDST, and MTOP. Additionally, the trained iterative retriever generalizes across different inference LLMs beyond the one used during training.

1 Introduction

A significant emergent capability of large language models (LLMs) is *in-context learning* (ICL; Brown et al., 2020), which facilitates few-shot learning. In ICL, a set of *exemplars*[1] is usually provided to build the mapping relationship between inputs and outputs. These exemplars can either be hand-crafted and fixed or retrieved from a training set. However, if retrieving from the dataset, the retrievers used in such applications are typically off-the-shelf models (e.g., Contriever (Izacard et al., 2022)) that do not consider interactions among retrieved items when

---

[†] Johns Hopkins University. Performed while interning at Microsoft.

[‡] Google. Performed while at Microsoft.

[1] An exemplar is a tuple of input and output, demonstrating the mapping relationship between the two.
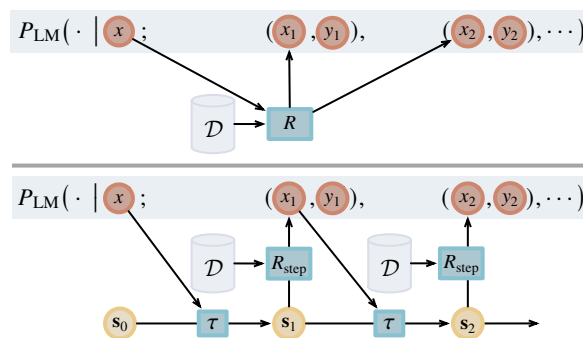

Figure 1: Above: ICL under a single retriever call. Below: ICL under our proposed iterative retriever.

multiple targets are required, nor the specific characteristics of the inference LLMs and downstream task requirements. Research (Gao et al., 2021; Liu et al., 2022; Lu et al., 2022, *i.a.*) has shown that ICL is sensitive to both the exemplars provided and their order within prompts. Off-the-shelf retrievers, which generally rank items based solely on semantic similarity (Lee et al., 2019; Reimers and Gurevych, 2019a, *i.a.*), do not ensure optimal conditions for either criterion, leading to suboptimal performance in downstream LLM generation. Hence, there is a need for a retriever capable of constructing a portfolio of items tailored to achieve optimal generation with LLMs.

We propose *iterative retrieval* to address this problem. Unlike traditional retrievers that perform a single call to obtain a list of similar items ordered by their similarities, iterative retrieval involves a sequence of retrieval calls, each using different query vectors. This makes the retriever stateful, maintaining an *internal state*. The process can be likened to navigating the encoding space of exemplars, with each step adjusting direction based on previously selected exemplars, thus building a trajectory of exemplar selections.

This approach can be formulated as *Markov decision processes* (MDPs). At each step, the *action* taken by the retriever is a retrieval call that fetches

(potentially multiple) documents from the dataset $\mathcal{D}$.[2] The policy is trained to optimally select exemplars at each step so that the overall trajectory maximizes the reward, leading to better ICL performance. By leveraging the LLMs as environments, we create simulators that allow a policy to roll out in the environment and receive feedback on the effectiveness of the composed prompts, measured by a reward (metric). Thus, exemplar selection and prompt composition can be framed as *policy optimization* aimed at maximizing rewards, which can be addressed through reinforcement learning.

We situate our study in in-context semantic parsing due to its difficulty, popularity, and practical value.[3] We instantiate an iterative retriever and investigate the performance of policy learning under this setup. Our contributions include:

- We propose a novel iterative retrieval framework that builds a portfolio of exemplars for ICL, considering both interactions among retrieved exemplars and their relationship with LLMs;

- We instantiate this iterative retriever for the in-context semantic parsing task and train its policy via reinforcement learning, demonstrating superior performance over strong baselines from prior work, thereby proving its effectiveness;

- Through a series of analyses, we provide insights into the behaviors of an iterative retriever initialized with an off-the-shelf retriever.

2 Overview of an Iterative Retriever

We consider the problem of *in-context learning* (ICL): given a dataset $\mathcal{D} = \{(x_i, y_i)\}_i$ of *exemplars*, a retriever R retrieves a sequence of exemplars $R(x)$ based on input query x and generate the answer y based on the distribution $P_{\text{LM}}(\cdot | x; R(x))$.

This retriever $R : \mathcal{X} \rightarrow \mathcal{D}^K$ retrieves an ordered list (of length K) of exemplars for the LM. The goal of the retriever R is to select a sequence of exemplars $((x_i, y_i))_{1 \le i \le K}$ such that the probability of the expected output y is maximized:

$$\arg\max_{(x_i, y_i) \in \mathcal{D}} P_{\text{LM}}(y | x; ((x_i, y_i))_{1 \le i \le K}). \quad (1)$$

However, this is a *combinatorial optimization* problem that is computationally infeasible to solve

---

[2] The action space is at least as large as $\mathcal{D}$.

[3] Code generation is considered one of the most useful but challenging techniques in the era of LLMs. Some semantic parsing tasks share structural similarity with code generation and program synthesis.

exactly. Much of prior work resort to selecting top-k exemplars based on a scoring function S:

$$R(x) = \underset{(x', y') \in \mathcal{D}}{\arg\text{top}_k} \; S(x, (x', y')) \quad (2)$$

Prior work has differed on the choice of the scoring function S: BM25 (Roy et al., 2023), coverage (Gupta et al., 2022), etc. However, such method did not model the interaction between the retrieved exemplars and the language model. We propose an iterative version, where we create a *retrieval state* s, and for each step i one exemplar $(x, y) \in \mathcal{D}$ is retrieved. This is an approximation to the optimization problem in Equation 1.

$$(x_i, y_i) \leftarrow R_{\text{step}}(s_i) \quad (3)$$
$$s_{i+1} \leftarrow \tau(s_i, (x_i, y_i)) \quad (4)$$

After K steps, the retrieved sequence would be $R_{\text{iter}}(x) = ((x_i, y_i))_{1 \le i \le K}$. This formulation of an *iterative retriever* naturally fits in the definition of a Markov decision process (MDP). Here, our decision process comprises of $(\mathcal{D}^*, \mathcal{D}, \tau, r)$, where

- The state set $\mathcal{D}^*$ contains exemplar sequences whose elements are in $\mathcal{D}$;

- The action set is just $\mathcal{D}$: each action selects one exemplar from $\mathcal{D}$. In theory, more than 1 exemplar can be selected at each step, but we proceed with just 1 exemplar for simplicity;

- The transition function $\tau : \mathcal{D}^* \times \mathcal{D} \rightarrow \mathcal{D}^*$ appends an exemplar to the existing sequence;

- The reward function $r : \mathcal{D}^* \times \mathcal{D} \rightarrow \mathbb{R}$ funnels signal from the LLM back to the retriever. It will be discussed in §4.

By situating our proposed iterative retriever under this RL scenario, we can utilize all sorts of RL techniques to train this retriever from the environment, which is the LLM itself. In the next section, we instantiate a neural iterative retriever and situate it under a common task, namely *semantic parsing*, under this ICL framework.

3 Instantiating an Iterative Retriever

We consider an *instance* of in-context learning, namely few-shot semantic parsing. Given a natural language query x, a model is expected to output a semantic representation y of x given a sequence of exemplars (see Figure 3).

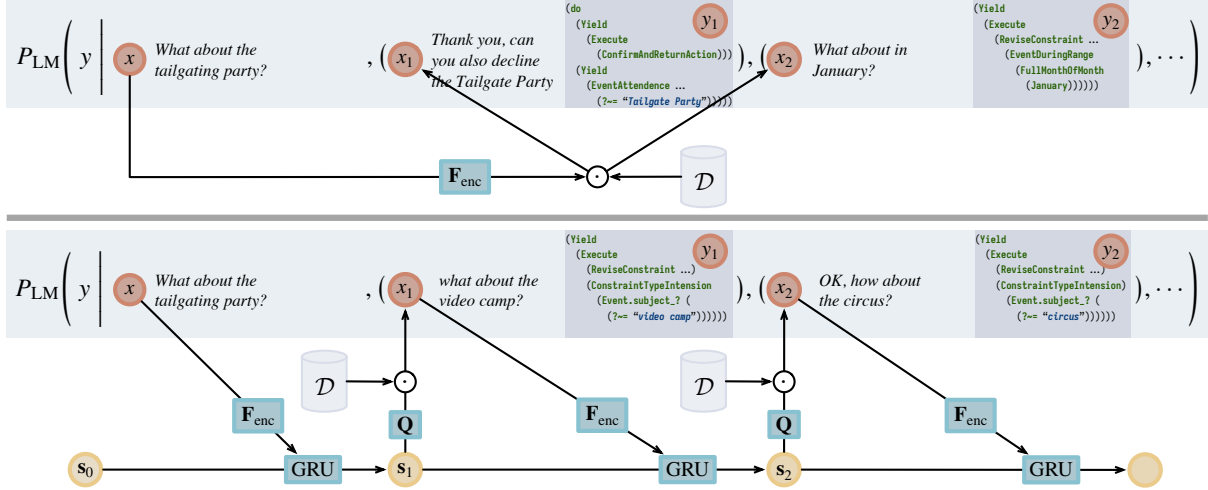We instantiate a *neural iterative retriever* based on the formulation we proposed above:

Figure 2: ICL prompt construction for an example in SMCALFLOW. *Above:* ICL with BM25 as the retriever. *Below:* An instance of our iterative retriever. BM25 retrieves examples that overlaps lexically with the query, whereas the trained iterative retriever is better at retrieving structurally similar exemplars since it is trained to maximize the probability of the LM generating the reference parse.

Figure 3: Samples of (x, y) pairs for semantic parsing under different datasets used in this paper.

- The state of the MDP, i.e. the sequence of exemplars, is modeled by a fixed-length vector $\mathbf{s} \in \mathbb{R}^d$. The initial state $\mathbf{s}_0$ is a parameter.

- At each step 1 exemplar is retrieved. We define a *policy distribution* that picks one exemplar from the training set $\mathcal{D}$, similar to Lu et al. (2023):

$$\pi((x_i, y_i)|s_i) \propto \exp(\mathbf{Q}(\mathbf{s}_i) \cdot \mathbf{F}_{enc}(x_i)/\beta) \quad (5)$$

where $\mathbf{Q} : \mathbb{R}^d \to \mathbb{R}^d$ maps a state vector $\mathbf{s}_i$ to a query vector $\mathbf{q}_i$, $\mathbf{F}_{enc} : V^* \to \mathbb{R}^d$ is a text embedder that maps a text sequence into a vector, and β is a temperature hyperparameter. In our experiments, $\mathbf{F}_{enc}$ is initialized with the weights of Contriever (Izacard et al., 2022), a general-purpose text embedder trained for retrieval.

Under this policy, if we take greedy decoding, the retrieval step would just be

$$(x_i, y_i) \leftarrow R_{step}(\mathbf{s}_i) = \arg\max_{(x', y') \in \mathcal{D}} \pi((x_i, y_i)|s_i)$$
$$= \arg\max_{(x', y') \in \mathcal{D}} \mathbf{Q}(\mathbf{s}_i) \cdot \mathbf{F}_{enc}(x_i). \quad (6)$$

This is a *maximum inner product search* (MIPS) problem, and thus can be solved with a vector index such as FAISS (Douze et al., 2024).

- State transition is modeled by a gated recurrent unit (GRU; Chung et al., 2014) update:

$$\mathbf{s}_{i+1} \leftarrow \text{GRU}(\mathbf{s}_i, \mathbf{F}_{enc}(x_i)) \quad (7)$$

where the encoded vector of the retrieved exemplar x_i is passed to the GRU to update the state.[4]

Note that the only additional parameters we included in this neural iterative retriever is the state transition model, where we instantiate as a GRU.

This is different from a regular retriever, where a single retrieval call to the training set $R(x) = \arg\max_{(x,y) \in \mathcal{D}} \mathbf{q} \cdot \mathbf{F}_{enc}(x)$ is made. The iterative retriever navigates the encoding space of exemplars, adjusting the query vector $\mathbf{q}'$ at each step based on previously selected exemplars s, thus steering the search process to find new candidates. Figure 2 demonstrates the process of such an iterative retriever. This stateful design allows for optimized retrieval results through iterative interactions, incorporating signals from both external

---

[4] Using a Transformer decoder here results in more unstable training as we discovered in our experiments. See §6.1.

sources (LLMs) and internal states (previously retrieved items tracked via state transitions).

4 Training

Environment Simulator   To construct feedback (or reward) from the underlying LLMs, we treat LLMs as environments where actions are performed and evaluated. We design an iterative prompting schedule within this LLM environment to simulate the process of iterative retrieval and corresponding ICL prompt execution. At each step i, the current sequence of chosen exemplars, s_i, is turned into an LLM prompt using a predefined template,[5] then used for LLM generation. This schedule effectively simulates the real-world scenario of prompting LLMs, allowing us to observe various execution dynamics, such as generated hypotheses and their probabilities.

Reward Design   Technically, if the final task metric were available, it can be used directly as the reward to optimize for. However, such a reward is often too coarse to reflect differences in partially correct results. For example, if the metric is exact match, which is common in semantic parsing tasks, the reward would simply be the Kronecker delta $\delta(y^*, y)$, yielding 1 only if the prediction y exactly matches the reference y^*, and 0 otherwise.

Given that the LLM simulator provides access to the probabilities of generated sequences,[6] we employ a more general reward design that is not task-specific. Our reward leverages the LM completion probability of the reference sequence $P_{\text{LM}}(y^*|x)$ (Shin et al., 2021; Shi et al., 2023), which captures subtle changes in the likelihood of the LM generating the target sequence with respect to changes in the input x. In ICL, more exemplars typically result in better performance before reaching saturation. Inspired by Zhang et al. (2022), We further refine the reward to reflect the *increase in the likelihood* of the reference y^* given the prompt. This is a proxy value that measure *how much this exemplar contribute* to generating the reference parse. This design encourages the model to select exemplars that most significantly contribute to the final result

---

[5] Refer to Appendix A.3 for the template used in this work.
[6] This is generally accessible in many LLM inference implementations such as vLLM (Kwon et al., 2023). For OpenAI-style APIs, this can be accessed using the "echo" parameter.

given the existing exemplar sequence s_i:

$$r(s_i, x_i) = P_{\text{LM}}(y^* \mid x; s_i, (x_i, y_i)) \\ - P_{\text{LM}}(y^* \mid x; s_i). \quad (8)$$

Policy Optimization   We employ *proximal policy optimization* (PPO; Schulman et al., 2017) to train an iterative retriever for its stability and efficiency.[7] One core idea of PPO is to define a clipping term that controls the policy optimization process, so that variance is reduced. Given a trajectory $(x_1, \cdots, x_T)$, we have

$$\mathcal{L}_i^{\text{clip}}(\theta) = \hat{\mathbb{E}}_i\big[\min(\rho_i, \text{clip}_\varepsilon(\rho_i)) \cdot \hat{A}_i)\big], \quad (9)$$

where $\rho_i = \frac{\pi_\theta(x_i|s_i)}{\pi_{\theta_{\text{old}}}(x_i|s_i)}$ is a probability ratio between action x_i[8] performed against the current policy π_θ and old policy $\pi_{\theta_{\text{old}}}$ at state s_i, $\text{clip}_\varepsilon(\rho)$ clips ρ to be within $(1 - \varepsilon, 1 + \varepsilon)$ and $\hat{A}$ is the advantage.

Advantage $\hat{A}_i$ at step i describes how much better it is to take a specific action x_i at state s_i, over randomly selecting an action according to $\pi(x_i|s_i)$. To compute it, besides the neural model defined in §3, we follow common practice in reinforcement from human feedback (RLHF; Huang et al., 2024) to add a single linear layer to serve as a state-value function $V(s) = \mathbf{v} \cdot \mathbf{s}$ that maps states to values. Generalized advantage estimate (GAE; Schulman et al., 2016) is then used for variance-reduced advantage estimation atop the learned state-value function:

$$\hat{A}_i = \delta_i + (\gamma\lambda)\delta_{i+1} + \cdots + (\gamma\lambda)^{T-i+1}\delta_{T-1} \quad (10)$$
$$\delta_i = r_i + \gamma V(s_{i+1}) - V(s_i) \quad (11)$$

where r_i is the reward obtained at step i, γ is the discount factor, λ downweighs rewards corresponding to delayed effects. Following Schulman et al. (2017) on PPO in Actor-Critic style, we minimize the value function error term by a squared-error loss, with an additional entropy bonus term $-H$:

$$L_i^{\text{PPO}} = \mathbb{E}_i\big[\mathcal{L}_i^{\text{clip}}(\theta) + c_1\hat{A}_i^2(\theta) - c_2 H_{\pi_\theta}(s_t)\big] \quad (12)$$

where c_1, c_2 are coefficients.

State Transition Initialization   We found that initializing the state transition function as an *identity* function (i.e., states do not change at all:

---

[7] We experimented with various other RL algorithms (including policy gradient (Sutton et al., 1999) and advantage actor critic (A2C; Mnih et al., 2016)) and found that PPO is the most stable one for our scenario.
[8] x_i describes that the action of an iterative retriever is to retrieve an exemplar from a candidate set, hence $x_i \in \mathcal{D}$.
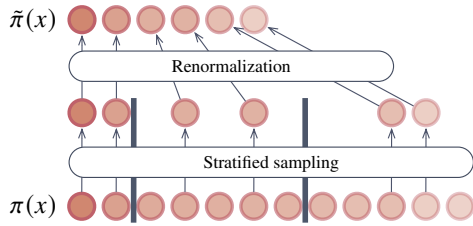
Figure 4: Stratified sampling employed in our approach. Our sampling method retains the top k/N_s samples and split the rest into $(N_s - 1)$ strata to perform stratified sampling. The resulting k samples are renormalized to construct action distribution.

$\tau_0(s, x) = s$) helps stabilizing training. This can be achieved in GRU by initializing the bias of the update gate to $b_z = -\infty$: this renders the GRU not forgetting and not updating.

Sampling & Collecting Experience In a single retrieval step, the retriever selects an exemplar from a candidate set, with the policy $\pi_\theta(x_i|s_i)$ defining a probability distribution over candidates $x_i \in \mathcal{D}$. In this RL simulation, it is crucial to sample different actions at each step to enable the model to explore various trajectories and benefit from those that yield higher rewards. However, constructing the entire distribution and sampling from it at each step is computationally infeasible, especially when the number of candidates exceeds 100K. Furthermore, these distributions often exhibit a long-tailed nature, where many candidates have low scores, suggesting that a significant portion of candidates may be less similar and potentially less useful for ICL.

To address these challenges and reduce the computational cost of sampling trajectories while managing the trade-offs between exploration and exploitation, we propose a stratified sampling (N_s strata) method to construct a modified policy $\tilde{\pi}$ that contains k candidates.

To start, we construct a buffer with top-B exemplars retrieved with Equation 5. Retain the top k/N_s samples in the policy. Split the rest into $(N_s - 1)$ strata, and sample k/N_s from each. Combine all these selected exemplars and renormalize these scores with softmax (with temperature β_{renorm}). This method enables the model to focus on more promising candidates while still allowing for exploration (see Figure 4 for an illustration).

During training, experience replay (Lin, 1992) is employed to improve training efficiency. To collect experience, we run inference with the current policy fixed on several training examples to generate trajectories. At each step, information such

as policy, reward, and value is recorded. These trajectories are stored in a replay buffer, then shuffled and split into mini-batches for policy optimization. This approach allows the same experiences to be replayed multiple times, reducing the number of required simulation runs.

5 Experimental Setup

Datasets We validate our pilot iterative retriever for ICL on a set of semantic parsing datasets, namely SMCALFLOW (Andreas et al., 2020), TREEDST (Cheng et al., 2020), and MTOP (English portion only; Li et al., 2021), following the BenchClamp benchmark (Roy et al., 2023). Samples of representations are shown in Figure 3. For statistics, see Appendix A.1.

Baselines We compare our iterative retriever (henceforth denoted as ITERR) with a range of off-the-shelf retrievers, including **BM25** (Robertson and Zaragoza, 2009) and a dense encoder, **Contriever** (Izacard et al., 2022). Additionally, we benchmark against two strong baselines from prior work on improving exemplar selection:

- **EPR** (Rubin et al., 2022) is an efficient exemplar retrieval method for in-context learning (ICL) that leverages a scoring LM to label positive and negative training examples, then using this dataset for a contrastively learned dense retriever.

- **CEIL** (Ye et al., 2023) uses *determinantal point processes* (DPPs) to model the interaction between the given input and in-context exemplars.

For the EPR baseline, we replace the base dense retrieval encoder with Contriever instead of S-BERT (Reimers and Gurevych, 2019b) for fair comparison. Following Ye et al. (2023), we use the trained EPR model as initialization for CEIL. Similarly, the same EPR checkpoint is used to initialize the text encoder in ITERR. Note that in ITERR, we freeze the weights of the EPR encoder and *only train the GRU-based state transition function, policy network, and value network*, resulting in 4M more parameters compared to the original Contriever (110M → 114M).

For retrievers without iterative capabilities, we take only the top-k retrieved items and keeping their original ranks. For EPR, CEIL, and ITERR, we selected the best performing model checkpoints on the validation set. All generation is run with **10** exemplars; i.e. $k = 10$.

Generation with LLMs The inference LLM is essential for executing input prompts to generate responses. In our experiments, we use `Llama-2-7b` (Touvron et al., 2023) to build the environment simulator and train the policy using its signals. With the learned policy, we investigate both *intra*-family and *inter*-family generalization by replacing the inference LLMs. For models within the same Llama-2 family, we explore various model sizes and finetuned versions, including `CodeLlama-70b-Instruct` (Rozière et al., 2023), a model further fine-tuned for code generation. For inter-family experiments, we choose `Mistral-7b` (Jiang et al., 2023). For decoding configurations, we consistently use beam search with beam size 3 and sampling temperature 0.5.

Hyperparameters Please refer to Appendix A.2.

Evaluation Metrics We follow prior work in evaluating semantic parsing (Roy et al., 2023), where *exact match* at k (EM@k) is used. Exact match results for top-k decoded hypotheses reflects beam search decoding used in LLMs, where multiple parsing results are generated simultaneously.

However, EM is a stringent metric, penalizing even minor mismatches. For instance, a parse with a substructure reordered differently (a && b) from the reference (b && a) is still correct but would score zero under EM. This is problematic in semantic parsing, where target parses are compositional, making it important to assess the correctness of substructures. Since SMCALFLOW and TREEDST involve deeply nested structures, we also adopt *SMatch* (Cai and Knight, 2013), following Chen et al. (2023), to evaluate performance on substructures. SMatch is designed to evaluate AMRs (Langkilde and Knight, 1998). Generated code can be transformed to AMRs by treating each function's return value as an *entity* and each argument to a function as a *value*, where the parameter name is the *relation*. See Appendix B for details.

6 Results & Analyses

We evaluate the performance of different retrievers by comparing their downstream ICL performance on semantic parsing (Table 1). ITERR outperforms all baselines across three datasets on all metrics.

The gain in EM is intuitive since it aligns with the training objective, which involves the probability of generating target parses. The improvement in SMatch indicates that ITERR optimizes retrieval

Retriever	Exact Match			SMatch		
	@1	@2	@3	P	R	F
SMCALFLOW						
BM25	39.8	43.7	44.0	66.6	64.2	65.3
Contriever	44.0	48.5	48.9	68.4	66.8	67.6
EPR	48.5	52.0	52.3	73.3	76.7	75.0
CEIL	51.1	54.2	55.8	74.9	75.2	75.1
ITERR	**54.1**	**58.4**	**58.5**	**76.6**	**78.1**	**77.3**
TREEDST						
BM25	50.8	56.1	56.6	81.8	81.8	81.8
Contriever	54.7	60.4	61.0	83.3	82.5	82.9
EPR	54.0	58.2	58.8	84.7	83.4	84.0
CEIL	56.2	58.3	61.6	81.3	84.4	84.9
ITERR	**58.2**	**63.4**	**63.8**	**85.5**	**85.8**	**85.7**
MTOP						
BM25	57.4	63.2	63.9	-	-	-
Contriever	59.3	64.2	64.7	-	-	-
EPR	62.3	68.8	69.2	-	-	-
CEIL	63.6	69.4	69.8	-	-	-
ITERR	**63.9**	**70.9**	**71.0**	-	-	-

Table 1: Comparison of our approach, ITERR against baselines. "EM@k" denotes exact match at top-k; "P", "R" and "F" denote precision, recall, and F_1 score respectively. Experiment results are run with 10 exemplars in the prompt, averaged over 3 inference runs, and significance tests using paired t-test confirm that the improvements over Contriever, EPR, and CEIL are statistically significant ($p < 0.05$).

results to improve compositionality to some extent, even with a simple objective.[9]

Generalization across Inference LLMs ITERR benefits from interactive training with an underlying LLM. While training incurs costs, these can be minimized by training only once, ideally using a smaller LM. Hence in this section we investigate the generalization capabilities of ITERR trained with a smaller LM A, but used for generation under a larger LM B.

In the following experiments, ITERR is trained with `Llama-2-7b` as the environment, but used for (a) *intra-family LMs*: variants within the `Llama-2` model family; and (b) *inter-family LMs*: Mistral (Jiang et al., 2023) from a different model family. We follow the setup described in §6, substituting only the LLM. As shown in Figure 5, ITERR significantly outperforms (> 1% gain) baselines for 75% of the settings and is comparable to a prior strong baseline (within 1% in absolute performance) for 15% of settings, demonstrating its generalization

---

[9] While a more dedicated reward design, such as incorporating various linearizations of target structures, might further enhance ITERR's performance. This work focuses on demonstrating the framework's effectiveness rather than dedicatedly optimizing for a specific task design.
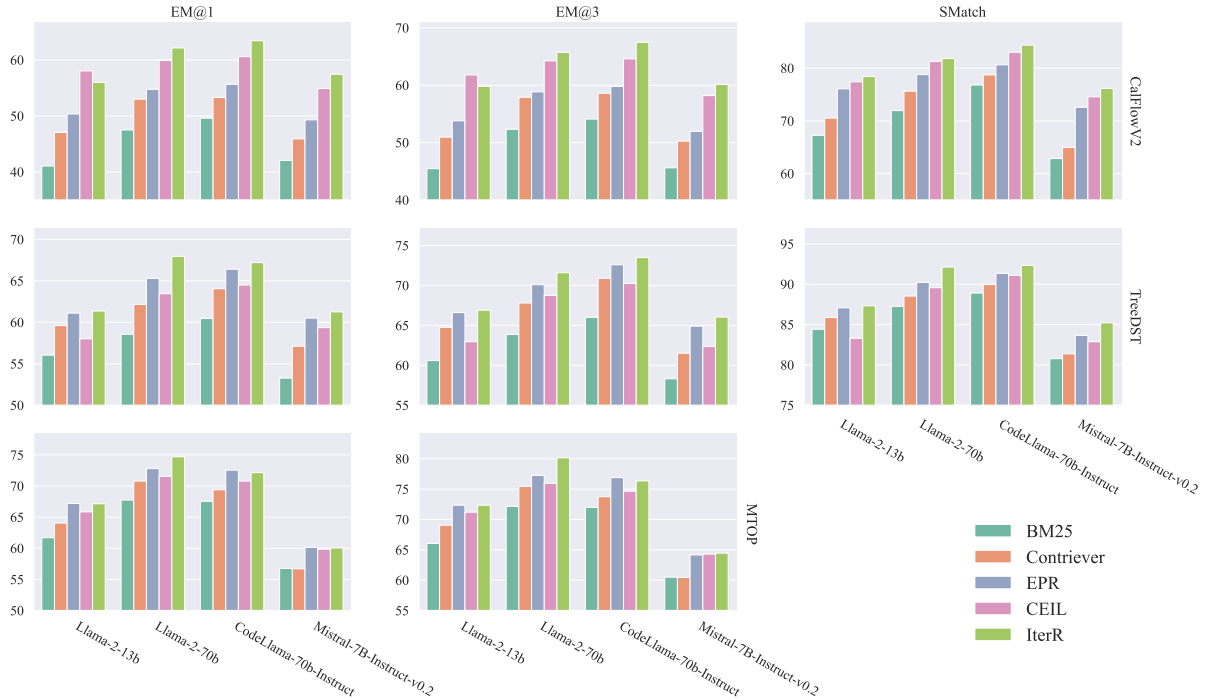
Figure 5: Performance comparisons on using various LLMs for inference (top row: SMCALFLOW; mid: TREEDST; bottom: MTOP). Our ITERR used in these experiments are trained with `Llama-2-7b` but performs retrieval of ICL exemplars used on other LLMs.

within and beyond its own model family.

In intra-family generalization, performance metrics improve with larger model sizes, and ITERR consistently outperforms all baselines. This improvement is most evident with larger models such as `Llama-2-70b` and `CodeLlama-70b-Instruct`. For inter-family generalization, ITERR maintains its advantage across datasets, though this is less pronounced than within the same model family. This is expected, as the signal from LLM simulator is more representative for models sharing the same pre-training procedure. Notably, with Mistral, Contriever performs worse than BM25 on MTOP, but ITERR still shows improvement. This suggests that ITERR, comprising a frozen EPR and additional GRU layers, can learn task-specific abilities not present in the vanilla EPR.

ICL & Number of Exemplars We investigated how the performance of ITERR changes with the number of exemplars ($\{1, \cdots, 10\}$) used for ICL on the SMCALFLOW dataset (Figure 6). ITERR consistently outperforms baseline models across various metrics and numbers of exemplars, with one exception for the EM@3 metric when using 6 exemplars. This aligns with our training objective, where actions that boost performance at each step receive higher advantages. ITERR achieves

comparable performance with fewer exemplars.

CEIL shows a similar trend in EM, but its SMatch performance lags significantly, indicating poorer quality in near-miss predictions compared to ITERR. Practically, this means our method allows for a trade-off between performance and cost, enabling effective ICL with fewer exemplars and reducing the number of tokens processed by LLMs.

6.1 Ablation Study

Variant	EM@1	SMatch-F
Contriever	44.0	67.6
ITERR	**54.1**	**77.3**
− *EPR intialization*	45.1	68.8
− *GRU; + Transformer decoder*	50.1	75.1
− *Stratified sampling*	52.3	75.7

Table 2: Results on ablation study. − *EPR intialization* indicates the model is trained from Contriever instead of a EPR finetuned checkpoint. + *Transformer decoder* replaces GRU with a Transformer decoder. − *Stratified sampling* replaces the stratified sampling described in Figure 4 with sampling directly from the buffer.

We further conduct ablation study on components of an iterative retriever, focusing on the SMCALFLOW dataset and use `Llama-2-7b` while changing the configuration of the iterative retriever. Results are reported in Table 2.
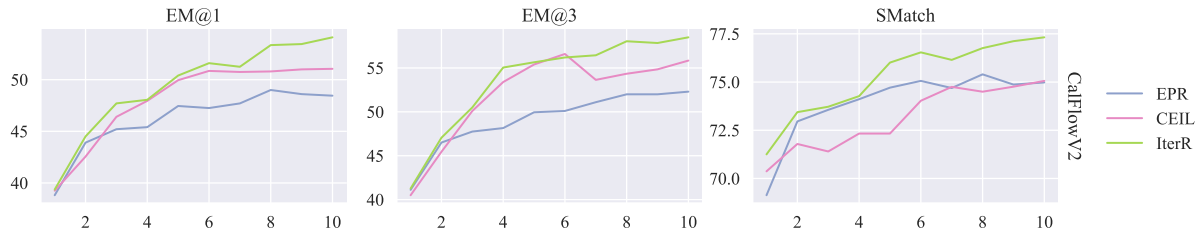
7162

Figure 6: Performance comparisons across the various numbers of exemplars used for ICL.

EPR Initialization Although we follow prior work in using EPR as initialization for $\mathbf{F}_{enc}$, our iterative retriever is agnostic to the choice of base encoders for similarity search. Even without EPR initialization, our training procedure still improves performance against Contriever ($\approx 1\%$ gain under Contriever, but $\approx 6\%$ gain under EPR). We see that ITERR benefits more when using EPR initialization, significantly outperforming the baselines. We hypothesize that this advantage stems from two sources: (1) EPR is fine-tuned on the target dataset, making it more domain-specific; (2) EPR restructures the action space, subsequently enhancing sample efficiency in RL training.

State Transition with Transformer Decoder In §3, we parameterize the state transition function in the iterative retriever with a GRU. To explore alternatives, we conducted an ablation experiment by replacing the GRU with a more powerful Transformer decoder, configured with 3 layers, 1024 hidden dimensions, with learnable positional encodings. Despite the increased expressiveness of the Transformer decoder, we observed a performance drop. During training, employing the warmup technique (Xiong et al., 2020) led to a trivial solution where the policy learned to predict a nearly fixed trajectory across test examples. Disabling the warmup stabilized the training but did not improve performance. Developing a stabilized approach to train the Transformer decoder as a state encoder is beyond the scope of this work, as our focus is on demonstrating the overall framework of iterative retrieval rather than optimizing a specific model for the state transition function. Notably, even with the less expressive GRU, our iterative retriever successfully learns a policy that retrieves a more optimized sequence of ICL exemplars.

Effectiveness of Stratified Sampling To collect diverse experience from policy rollouts, we introduce a stratified sampling method (described in §4) that balances the trade-off between exploration

and exploitation. We found that sampling from the raw policy in Equation 5 results in a significant performance drop. Additionally, qualitative examination of several such distributions revealed a preference for exploitation over exploration, as similar items at the top of the retrieved list all had higher probabilities.

7 Additional Related Work

LLMs as Environment in RL Lu et al. (2023) used policy gradient to learn a dense retriever for ICL exemplar retrieval, but the state does not contain previously selected examples, and thus is not iterative and unable to model exemplar order. Zhang et al. (2022) used *Q*-learning RL for ICL exemplar reordering, with a similar reward design like ours. However, the proposed method does not extend to exemplar *retrieval*, since the policy space is too large to be handled by *Q*-learning.

Few-shot Semantic Parsing Few-shot semantic parsing using LLMs has shown impressive capabilities in understanding new examples with minimal training data (Shin et al., 2021; Shin and Van Durme, 2022). However, these parsers often struggle with generalization and fail to parse unobserved local structures due to their limited access to information encoded through exemplars (Bogin et al., 2022). To this end, recent research has explored various approaches to improving exemplar selection. EPR (Rubin et al., 2022) used a proxy LM to score outputs from an unsupervised retriever, enabling better training of a dense retriever. Oren et al. (2021), Gupta et al. (2022), and Levy et al. (2023) emphasize learning to select exemplars based on particular criteria, such as diversity measures and coverage of local structures, to enhance compositional generalization. While these approaches have shown performance improvements in semantic parsing tasks, these are highly based on heuristics constructed from researcher's experience. Our approach could be seen as an *automated* version (through RL) of seeking

information useful for semantic parsing.

Multi-hop Question Answering There has been related work in iterative retrieval in multi-hop QA. Qi et al. (2019) iteratively generates text queries based on paragraphs retrieved, thus can be considered as a discrete, non-neural version of our MDP. Asai et al. (2020); Xiong et al. (2021) trained a dense, neural iterative retriever with oracle paths as direct supervision. Additionally, IRCoT (Trivedi et al., 2023) utilizes an iterative retriever to perform chain-of-thought (CoT) in LLMs, in which the agent is the LLM, thus not trained.

8 Conclusion

We proposed *iterative retrievers* that iteratively builds a prompt to perform in-context learning. Such retrievers are framed as Markov decision processes and trained via policy optimization from LLM feedback, where the policy directs which exemplar to append to the existing exemplar sequence. Experiments on semantic parsing demonstrated performance gain of iterative retrievers over various datasets and state-of-the-art baselines, showing that they are able to construct prompts that improves in-context learning and downstream LLM generation.

Limitations

In our instantiation of the iterative retriever, at each step a single exemplar is retrieved. One could envision multiple exemplars being retrieved at each step, thus making the RL trajectory shorter. This could make RL training easier and inference faster.

Our reward design depends on a particular linearization of the target structure. A more structured reward function may exhibit better training behavior and lead to better performance.

The encoder for queries in the iterative retriever is frozen in our current setup. A trainable query encoder that receives feedback from LLMs may be desired, but we left that for future work.

While we believe that semantic parsing / code generation is one of the most useful but challenging task for LLMs, as such is a representative task for ICL research, we have not tested the effectiveness of iterative retrievers under other LLM tasks.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy Mc-Govern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2731–2747, Abu Dhabi, United Arab Emirates. 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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Yunmo Chen, William Gantt, Tongfei Chen, Aaron White, and Benjamin Van Durme. 2023. A unified view of evaluation metrics for structured prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12868–12882, Singapore. Association for Computational Linguistics.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders

Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Structurally diverse sampling for sample-efficient training and comprehensive evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4966–4979, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. 2024. The N+ implementation details of RLHF with PPO: A case study on tl;dr summarization. *CoRR*, abs/2403.17031.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

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

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: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Long Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8:293–321.

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.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.

Inbar Oren, Jonathan Herzig, and Jonathan Berant. 2021. Finding needles in a haystack: Sampling structurally-diverse training sets from synthetic data for compositional generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10793–10809, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. 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 2590–2602, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019a. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. 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.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Subhro Roy, Samuel Thomson, Tongfei Chen, Richard Shin, Adam Pauls, Jason Eisner, and Benjamin Van Durme. 2023. Benchclamp: A benchmark for evaluating language models on syntactic and semantic parsing. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

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

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652.

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*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*, pages 5417–5425, Seattle, United States. Association for Computational Linguistics.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1057–1063. The MIT Press.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu,

Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 39818–39833. PMLR.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9134–9148, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Experiment Details

A.1 Dataset Statistics

Dataset	Train	Dev	Test
SMCALFLOW	108,753	12,271 (500 used)	13,496
TREEDST	121,652	22,910 (500 used)	22,841
MTOP	15,667	2,235 (500 used)	4,386

Table 3: Dataset statistics.

A.2 Hyperparameters

Name	Search Bounds
Encoder	{ GTR-T5, **Contriever**, SBert }
Learning rate	$\{5 \times 10^{-6}, 1 \times 10^{-5}, \mathbf{3 \times 10^{-5}}, 5 \times 10^{-5}\}$
LR scheduler	{ **reduce-on-plateau**, cosine-annealing }
State transition	{ **GRU**, LSTM, Transformer Decoder}
β_{renorm}	$\{0.5, 1.0, \mathbf{5.0}, 10.0\}$
c_1	$\{0.1, 0.3, \mathbf{0.5}, 0.7\}$
c_2	$\{0, 0.005, \mathbf{0.01}, 0.05, 0.1, 0.15\}$
γ	0.99
λ	0.95
Action buffer size	768
PPO ratio cutoff	1.2
PPO batch size	128
Replay buffer size	2048
Avg. training time	24 hrs
GPU used	4 Nvidia V100 32 GB
# of parameters[*]	~114M (w/ 110M frozen)

Table 4: Hyperparameters and other reproducibility information for ITERR. β_{renorm} is the temperature used to create a renormalized action distribution. c_1 and c_2 are coefficients used in the PPO loss. γ and λ are discount factors used in GAE.

A.3 Prompt Template

The prompt template used across all our experiments is shown in Table 5.

Let's translate what a human user says into what a computer might say.
Human: x_1 Computer: y_1
$\ldots$
Human: x_N Computer: y_N
Human: x Computer:

Table 5: Prompt template used in our experiments. This template will be instantiated as prompts when filled with retrieved exemplars $R(x) = ((x_1, y_1), \cdots, (x_N, y_N))$ and the test example x.

B SMatch Evaluation

For evaluation of semantic parse or code generation on partial results, we utilize SMatch (Cai and Knight, 2013). Generated code can be transformed to AMRs by treating each function's return value as an *entity* and each argument to a function as a *value*, where the parameter name is the *relation*. An example is given below.

Consider the following parse in SMCALFLOW, expressed in Lisp:

```
(Yield
  :output (Event.start
    :obj (FindNumNextEvent
      :constraint (Event.subject_?
        :obj (?~= "staff␣meeting"))
      :number 1L)))
```

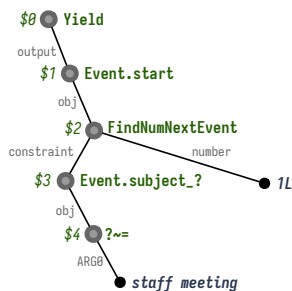This will be transformed into the following AMR:

Figure 7: Example AMR based on the previous parse.

This AMR can be easily converted to the following triples.

```
instance($0, Yield)
output($0, $1)
instance($1, Event.start)
obj($1, $2)
instance($2, FindNumNextEvent)
constraint($2, $3)
instance($3, Event.subject_?)
obj($3, $4)
instance($4, ?~=)
ARG0($4, "staff␣meeting")
number($2, 1L)
```