Reinforcing Compositional Retrieval: Retrieving Step-by-Step for Composing Informative Contexts

Quanyu Long[*1] **Jianda Chen**[*1] **Zhengyuan Liu**[2] **Nancy F. Chen**[2]
Wenya Wang[1] **Sinno Jialin Pan**[1,3]
[1]Nanyang Technological University, Singapore
[2]Institute for Infocomm Research (I2R), A*STAR, Singapore
[3]The Chinese University of Hong Kong
{quanyu001, jianda001}@ntu.edu.sg

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities across numerous tasks, yet they often rely on external context to handle complex tasks. While retrieval-augmented frameworks traditionally focus on selecting top-ranked documents in a single pass, many real-world scenarios demand compositional retrieval, where multiple sources must be combined in a coordinated manner. In this work, we propose a tri-encoder sequential retriever that models this process as a Markov Decision Process (MDP), decomposing the probability of retrieving a set of elements into a sequence of conditional probabilities and allowing each retrieval step to be conditioned on previously selected examples. We train the retriever in two stages: first, we efficiently construct supervised sequential data for initial policy training; we then refine the policy to align with the LLM's preferences using a reward grounded in the structural correspondence of generated programs. Experimental results show that our method consistently and significantly outperforms baselines, underscoring the importance of explicitly modeling inter-example dependencies. These findings highlight the potential of compositional retrieval for tasks requiring multiple pieces of evidence or examples[1].

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable progress in recent years, mastering text generation and diverse problem-solving. Context often plays a critical role in grounding these models in task-relevant information: it helps incorporate domain-specific knowledge, clarify ambiguous queries, and provide demonstrative examples. This leads to the common adoption of retrieval-augmented frameworks (Lewis et al., 2020; Izacard et al., 2022b; Liu et al., 2022), which introduce a "retrieve" step to extract relevant context segments such as documents or examples to enrich the LLM's input, improving both the accuracy and reliability of its outputs.

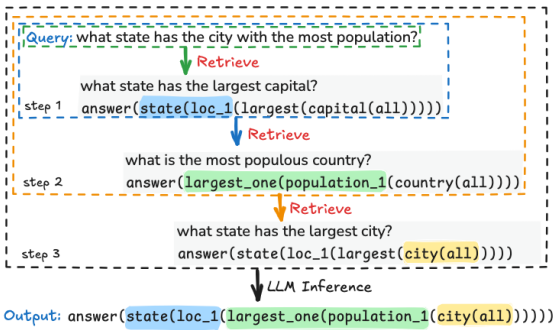

Figure 1: When generating the target semantic program of a query (parsing the query into the logical form), relying on single or repetitive demonstrative context cannot cover the ground-truth program local structure and lead to poor context augmentation quality. In order to capture the whole retrieval results' collective effect to facilitate better local structure coverage and program generalization, we solve this compositional retrieval problem by modeling it as a Markov Decision Process.

However, many complex tasks demand combining multiple pieces of evidence or examples of diverse semantics or structures. For instance, multi-hop queries often require several documents to accumulate all necessary insights. Another scenario, as illustrated in Figure 1, involves translating a query into a logical forms, where an LLM needs to assemble symbols or sub-trees from multiple demonstrative examples to construct the target program accurately. In this case, relying on a single retrieved example can only provide partial hints, while repetitive or redundant examples provide little additional information. Consequently, this compositional retrieval setting necessitates coordinated selection of examples to maximize collective utility of chosen composition.

[*]Equal contribution.
[1]Codes of this work are available at https://github.com/ruyue0001/Step-by-Step-Retrieval

Findings of the Association for Computational Linguistics: ACL 2025, pages 7633–7651
July 27 - August 1, 2025

To address these complexities, conventional top-k retrieval is insufficient, as it operates in a single-turn manner and fails to capture how newly selected items condition subsequent choices. Therefore, we propose a compositional retrieval paradigm that conducts retrieval in multiple steps, with each step adapting to the evolving context formed by previously retrieved items. This approach ensures that newly selected examples complement previously chosen ones, improving the overall coverage of retrieved content (Figure 1). Although recent works have explored sequential retrieval for selecting exemplary contexts (Liu et al., 2024a,b), they typically rely on a large scoring language model for constructing training sequences, incurring significant computational overhead.

To maximize the collective utility of the retrieved examples instead of selecting multiple high-scoring yet potentially redundant elements from a single global ranking, we formulate compositional retrieval as a Markov Decision Process (MDP), wherein each state reflects both the query and the previously chosen items. We adopt a tri-encoder architecture as the policy model, mapping each query and partial retrieval sequence to a probability distribution over candidate examples. To initialize this policy, we construct supervised fine-tuning data in a more efficient manner by maximizing sub-structure coverage without scoring models. In the second stage, we perform reinforcement learning (RL) and adopt a grouped policy optimization (Shao et al., 2024; DeepSeek-AI et al., 2025), reinforcing the policy using a task-specific reward based on local structure coverage of the generated logical form. This design captures the LLM's compositional needs and progressively enhances the quality of the assembled context.

We evaluate our method **R**einforcing **C**ompositional **R**etrieval (**RCR**) on compositional generalization semantic parsing benchmarks which requires generating new combinations of familiar structures and symbols, making it an ideal initial scenario to examine how context compositions impact program generation capabilities. Results show our method consistently outperforms top-k and sequential retrieval baselines. The observed improvements stem from our tri-encoder retriever's ability to model inter-example dependencies, enabling more effective context composition and enhancing program generation quality within the LLM. Our analysis further underscores the effectiveness of leveraging RL in refining retrieval strategies. We systematically evaluate different RL setups and advantage estimation methods, providing a comprehensive evaluation and insights into both its benefits and limitations in retrieval tasks. In summary, our key contributions are:

- We formalize a compositional retrieval process that explicitly accounts for coordinated selection of contexts via sequential conditional probabilities.
- We develop a tri-encoder retriever, modeling retrieval as a Markov Decision Process and factoring each selection step upon previously chosen examples.
- We introduce an efficient supervised fine-tuning step for data construction and subsequently refine the retriever through reinforcement learning, aligning it with the LLM's downstream performance.

2 Problem Formulation

In this work, we depart from conventional top-ranked retrieval which treat each candidate independently. Instead, we propose a compositional retrieval problem that aims to optimize the selection of the most influential context group, and each element within this group will contribute together to lead to a correct prediction.

2.1 Compositional Retrieval

The common retrieval-augmented framework mainly contains two components, a dense retriever to retrieve relevant contexts and a Large Language Model (LLM) to accept retrieved contexts for generating responses. Given a set of retrieved elements $\mathcal{Z}$, LLMs will condition on the combination of context $\mathcal{Z}$ and query x to generate the output y by prepending all k retrieved elements to x:

$$\mathcal{P}(y \mid x) \approx \sum_{\mathcal{Z}} \mathcal{P}_{\mathrm{LM}}(y \mid \mathcal{Z}, x)\, \mathcal{P}(\mathcal{Z} \mid x), \quad (1)$$

where $\mathcal{Z} = [z_1, \cdots, z_k]$, and $[\cdot]$ denotes text concatenation. In order to retrieve k elements, in standard dense retrieval, the likelihood of retrieving a single candidate z is computed as:

$$\mathcal{P}(z \mid x) \approx \frac{\exp(\mathrm{sim}(x, z))}{\sum_{z' \in \mathcal{C}} \exp(\mathrm{sim}(x, z'))}, \quad (2)$$

where $\mathrm{sim}(\cdot, \cdot)$ is a similarity function, which is often computed via inner product, $\mathrm{sim}(x, z) =$

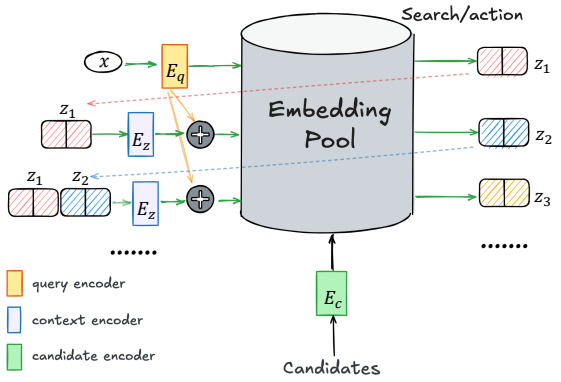

Figure 2: Tri-encoder model to retrieve step-by-step.

$\mathbf{E}_c(z)^\top \mathbf{E}_q(x)$. The final context $\mathcal{Z}$ is obtained by selecting the top-k candidates based by Eq. (2).

However, such top-k selection treats candidates independently,[2] disregarding the compositional quality of the retrieved set and failing to assess the joint utility of retrieved elements, which is crucial for tasks where multiple documents must interact to form an informative context. To address this, we define compositional retrieval as computing the probability of retrieving a set of candidates as a sequence of conditional probabilities:

$$\mathcal{P}(\mathcal{Z}\,|x) = \mathcal{P}(z_1|x)\prod_{i=2}^{k}\mathcal{P}(z_i|x, z_1, \ldots, z_{i-1}). \tag{3}$$

We explicitly model dependencies among retrieved elements, where each selection influences subsequent choices. Unlike traditional methods that independently select top ranked elements in a single turn and unable to estimate the compositional retrieval probability, this approach optimizes the overall composition by accounting for both individual relevance and inter-example interactions, leading to more collective context selection.

2.2 Overview of the MDP Modeling

We further formalize the proposed step-by-step retrieval process as a Markov Decision Process (MDP). In compositional program generalization (an example in Figure 1), prior works (Levy et al., 2023; An et al., 2023) show that increasing structural coverage (i.e., sub-trees of the program tree) of the expected program within the demonstrative context improves in-context learning (ICL) performance, since high-coverage contexts enhance generalization by exposing the model to essential program symbols (e.g., predicates and logical operators). Thus, the goal of our decision process is to select program examples that provide more expected symbols and larger structures coverage within the whole context, while minimizing redundant selections with limited utility.

State At each step t, state is defined as $s_t = [x, z_1, \cdots, z_{t-1}]$, representing the partial retrieval sequence, where x is the input, and z_i is i-th selected program example.

Policy To compute the conditional probability in Eq. (3), we avoid using a single encoder to process the concatenated input $[x, z_1, \cdots, z_{t-1}]$, which may weaken the signal input x. Instead, we employ a query encoder $\mathbf{E}_x$ and a context encoder $\mathbf{E}_z$ to separately encode x and each selected example z_i (Figure 2). Given a candidate pool $\mathcal{C} = \{c_j\}_{j=1}^{N}$, we introduce a third encoder $\mathbf{E}_c$ to encode candidate examples, the selection logit for candidate c_j at step t are computed as:

$$\mathrm{q}(x, \mathcal{Z}_{t-1}, c_j) = \mathbf{E}_c(c_j)^\top(\mathbf{E}_x(x) + \lambda\sum_{i=1}^{t-1}\mathbf{E}_z(z_i)), \tag{4}$$

where λ is a weighting factor, and $\mathcal{Z}_{t-1} = [z_1, \cdots, z_{t-1}]$ represents selected list at time step $t-1$. The policy distribution is then given by: $\pi_\theta(\cdot|x, z_{i<t}) = \mathrm{softmax}(\mathbf{q}(x, \mathcal{Z}_{t-1}, \cdot)/\tau)$, τ is a scaling temperature.

Action At step t, an action involves selecting an example z_t from $\mathcal{C}$ by sampling from $\pi_\theta(\cdot|x, z_{i<t})$. Once selected, an action is removed from the candidate pool to prevent duplicate selection.

Reward After k retrieval steps, the selected examples serve as context for prompting an LLM in an ICL manner to generate a response $\hat{y}$:

$$\hat{y}_i \sim \mathcal{P}_{\mathrm{LLM}}(\hat{y}_i|[\underbrace{z_1, z_2, \cdots, z_k}_{\text{context}}, x], \hat{y}_{1:i-1}). \tag{5}$$

We define a reward based on local structural similarity between the generated and reference programs. Following Levy et al. (2023), we first anonymize programs by replacing values (e.g., strings, numbers) with constants, as these are typically irrelevant for structural coverage. Each program is then represented as a tree, where nodes

[2]The selection of each element from a probability distribution is not necessarily independent when considering rank dependence, i.e. once choosing one element, the remaining elements are selected from a constrained set without replacement using re-normalization. However, once considering rank dependence, it inherently become a sequential selection.

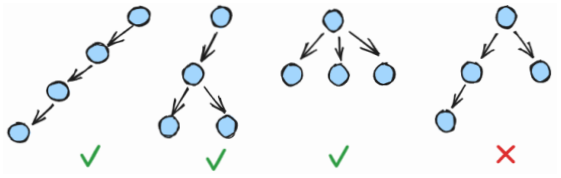

Figure 3: Local structures when size $l = 4$. Following Levy et al. (2023), we exclude more distant familial ties.

correspond to program symbols and edges capture parent-child relationships between functions and their arguments. A local structure is defined as a subgraph where all leaf nodes share a sibling relationship, meaning it includes only parent-child and sibling connections while excluding cousin or uncle relations. The size of a local structure l is defined as the number of nodes in this sub-graph (see Figure 3). Additional details are provided in Appendix B, with examples in Table 4.

We extract local structures of size $\leq l$ from both the generated program $\hat{y}$ and the ground-truth program y, and then collect all the extracted structures into a set. We denote such extraction as a function $LS^l(\cdot)$. The reward is then computed as the Jaccard similarity between their local structures:

$$r(o) = \frac{|LS^l(\hat{y}) \cap LS^l(y)|}{|LS^l(\hat{y}) \cup LS^l(y)|}, \tag{6}$$

where $o = [x, z_1, \cdots, z_k]$ is the final observation. This formulation naturally penalizes the generation of overly lengthy programs.

3 Reinforcing Compositional Retrieval

The training of our propose tri-encoder sequential retriever contains two stages, supervised fine-tuning (SFT) and reinforcement learning (RL).

3.1 Stage 1: SFT

To construct sequential training data and establish a strong initial policy, we propose an intuitive and efficient data construction method. Given an input x and a candidate pool $\mathcal{C}$, we first construct the training data for step 1. A high-quality program example should maximize local structure coverage, as broader coverage reduces the model's need to generate new structures and provides insights into structural fusion. Therefore, we greedily sample the candidate in the pool $\mathcal{C}$ that maximizes local structure coverage relative to the ground-truth y.

For subsequent steps, intuitively, an optimal composition should introduce diverse symbols and local structures to enhance generalization. Therefore, retrieval should prioritize covering unseen structures. More specifically, as shown in Figure 1, after expanding the set of retrieved examples (encode by $\mathbf{E}_z$ in Figure 2), the retriever **should select examples that can cover previously uncovered structures** in y. We iteratively sample candidates from $\mathcal{C}$ that maximize coverage of these remaining structures. Algorithm 1 presents the data construction procedure for SFT stage.

Algorithm 1 Data construction for SFT

Input: data instance (x, y), example pool $\mathcal{C}$, total step k, maximum local structure size l, local structure extractor $LS^l(\cdot)$, uncovered local structure set $\mathcal{S}$, selected examples list $\mathcal{Z}_t$ at step t.

$\mathcal{S} \leftarrow LS^l(y)$
$\mathcal{Z}_0 \leftarrow []$
for $t = 1, \cdots, k$ **do**
  $c_t \leftarrow \arg\max_c |LS^l(c) \cap \mathcal{S}|, c \in \mathcal{C}$
  Append $(x, \mathcal{Z}_{t-1}, c_t)$ to $\mathcal{D}^{SFT}$
  $z_t \leftarrow c_t$
  $\mathcal{Z}_t \leftarrow \mathcal{Z}_{t-1} + [z_t]$
  $\mathcal{S} \leftarrow S - LS^l(z_t)$
end for

Output: Training Data $\mathcal{D}^{SFT}$ for (x, y).

Unlike existing data construction methods that rely on a scoring LLM (often distinct from the inference model) to encode selected examples and rank training data based on similarity scores (Liu et al., 2024a,b), our approach eliminates the computational overhead of LLM forward passes while maintaining a more interpretable selection process.

After constructing the training data, each instance $(x, \mathcal{Z}_{t-1}, c_t) \in \mathcal{D}^{SFT}$ is used to train the tri-encoder retriever. The objective is to maximize the selection logit of the candidate c_t, treating it as the positive example given input x and previously retrieved examples $\mathcal{Z}_{t-1}$. For negative samples $\mathcal{N}$, we incorporate both in-batch negatives and hard negatives. Hard negatives are selected using a two-step process: 1) retrieve top-H most similar example to x using BM25 (Robertson et al., 2009), 2) rank these H candidates by their local structure coverage relative to the ground-truth y and select the bottom-B (high-similarity but low-coverage) candidates as hard negatives. The tri-encoder is trained using the InfoNCE loss (Oord et al., 2018;

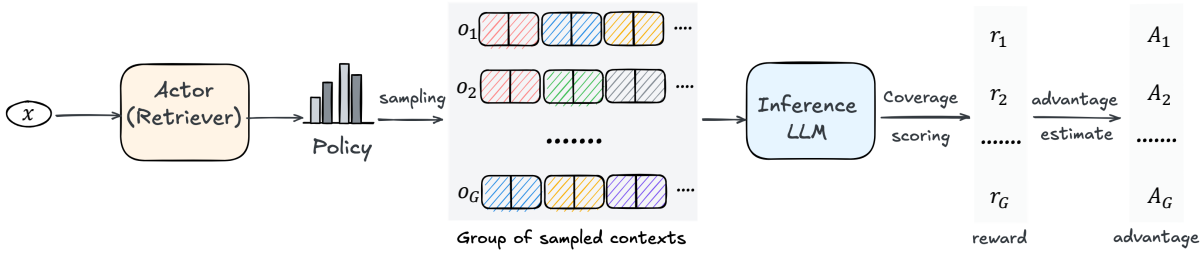

Figure 4: Group Relative Policy Optimization (GRPO) process for optimizing the sequential retriever.

Chen et al., 2020):

$$\mathcal{L}(x, \mathcal{Z}_{t-1}, c_t, \mathcal{N}) = -\log \frac{\exp\left(\mathrm{q}(x, \mathcal{Z}_{t-1}, c_t)\right)}{\sum_{c_j \in \{\mathcal{N} \bigcup c_t\}} \exp\left(\mathrm{q}(x, \mathcal{Z}_{t-1}, c_j)\right)}, \quad (7)$$

Where $\mathrm{q}(x, \mathcal{Z}_{t-1}, c_j)$ is computed by Eq. (4).

3.2 Stage 2: RL

Although the learned initial policy ensures high coverage of the expected output program, it does not guarantee correct generation by the inference LLM. The retriever should also capture the LLM's preferences and align accordingly. With a sequential retriever, we collect a chain of retrieved examples over k steps and then feed the concatenation of the prompt and test utterance $[\mathcal{Z}_k, x]$ to a LLM (Eq. (5)). After generating the full response (Eq. (6)), we obtain a reward. This reward signal eliminates the need for pairwise preference data, allowing reinforcement learning to optimize retrieval based on aggregated evaluations across multiple example chains.

To achieve this, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) , which enables efficient policy alignment with the LLM without requiring a separately trained value model. For each query x, we sample a group of outputs $\{o_1, \cdots, o_G\}$ using the tri-encoder with the old policy $\pi_{\theta_{old}}$ and obtain a corresponding group of rewards $\{r_1, \cdots, r_G\}$. The policy is then optimized by maximizing the following objective:

$$\mathcal{J}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\Bigg(\min\Big(r_\theta(o_i)A_i, \mathrm{clip}\big(r_\theta(o_i), 1-\epsilon, 1+\epsilon\big)A_i\Big) - \beta D_{KL}(\pi_\theta \parallel \pi_{ref})\Bigg), \quad (8)$$

where $r_\theta(o_i)$ is the probability ratio between the new and old policy, $r_\theta(o_i) = \frac{\pi_\theta(o_i|x)}{\pi_{\theta_{old}}(o_i|x)}$. The clipping function clip truncates the ratio between the range $[1-\epsilon, 1+\epsilon]$, stabilizing updates. The advantage function A_i computed using normalized group rewards:

$$A_i = \frac{r_i - \mathrm{mean}(\{\mathrm{r}_1, \cdots, \mathrm{r}_G\})}{\mathrm{std}(\{r_1, \cdots, r_G\})}. \quad (9)$$

To prevent policy drift, a KL-divergence regularization term constrains the updated policy to remain close to the initial policy from Stage 1:

$$D_{KL}(\pi_\theta||\pi_{ref}) = \frac{\pi_{ref}(o_i|x)}{\pi_\theta(o_i|x)} - \log\frac{\pi_{ref}(o_i|x)}{\pi_\theta(o_i|x)} - 1. \quad (10)$$

After reinforcement learning, we encode the entire candidate pool $\mathcal{C}$ using the trained candidate encoder $\mathbf{E}_c$. For a test input x_{test}, retrieval follows a greedy decoding strategy, where at each step t, we select the candidate with the highest probability under the learned policy $\pi_\theta(\cdot|x_{\text{test}}, z_{i<t})$. We find that beam search provides only marginal improvements for sequential retrieval, a similar observation noted in Liu et al. (2024a). The retrieval process continues until k examples are collected, forming the context for LLM inference. Evaluation is then performed on the LLM's final generated program.

4 Experiments

4.1 Datasets

We conduct our experiments on several Compositional Generalization tasks for generating semantic parsing program. This task inherently involves generating new combinations of familiar structures and symbols, explicitly highlighting the compositional nature. This explicitness makes it an ideal initial scenario to examine how context compositions impact model generation capabilities.

GeoQuery. GeoQuery (Zelle and Mooney, 1996; Tang and Mooney, 2001) is a corpus of 880 questions related to U.S. geography. We follow the splits introduced by Shaw et al. (2021), which include: 1) Template split: Target programs are

	GeoQuery								COVR-10
	i.i.d.	Template 1	Template 2	Template 3	TMCD 1	TMCD 2	TMCD 3	Length	avg.
Cover-LS (Levy et al., 2023)	70.97	64.05	46.67	59.88	59.57	52.89	65.05	49.24	66.72
Top-k Retrieval									
Random	11.43	9.34	8.46	7.88	9.09	8.48	6.67	3.03	24.56
BM25	72.14	40.36	26.89	44.85	46.67	42.42	48.79	27.88	1.74
BERT	77.50	43.97	26.28	45.75	**55.45**	41.81	54.24	29.09	25.14
Contriever	69.28	41.86	27.79	30.30	50.30	40.61	53.33	30.91	28.17
EPR (Rubin et al., 2022)	67.50	44.28	26.89	35.76	45.45	40.91	53.03	27.88	27.82
Sequential Retrieval									
se^2 (Liu et al., 2024a)	74.28	30.42	26.28	41.21	49.69	42.12	54.54	31.51	29.46
se^2+ tri-encoder	77.50	54.81	26.58	43.03	50.61	43.03	55.15	31.52	30.44
RCR w/o SFT (Ours)	72.86	39.76	25.68	30.91	46.97	39.39	53.64	28.79	26.72
RCR w/o RL (Ours)	77.85	58.73	28.39	45.76	50.61	43.94	57.58	32.12	33.64
RCR (Ours)	**78.21**	**59.64**	**33.83**	**48.48**	51.52	**44.24**	**59.09**	**33.03**	**36.18**

Table 1: Exact-match accuracy on GeoQuery and COVR-10. Results highlight the advantages of a tri-encoder design and compositional retrieval via a Markov Decision Process.

anonymized into templates, then randomly partitioned into training and test sets (Finegan-Dollak et al., 2018). 2) TMCD split: The distribution of compounds in the training data is made maximally different from that of the test data (Keysers et al., 2020). 3) Length split: The test set contains longer program sequences than the training set.

COVR-10. COVR-10 (Bogin et al., 2022) is a synthetic dataset featuring a variable-free functional language. It comprises 10 compositional grammar splits, each of which includes distinct local structures in the test set that do not appear in training. Following prior setups (Levy et al., 2023), we aggregate results by averaging across the 10 splits. More details are provided in Appendix A.

4.2 Experiment Setup

Models We initialize all three encoders in our proposed retriever with `bert-base-uncased` embeddings (Devlin et al., 2019). To compare with previous results, however, Codex (`code-davinci-002`) (Chen et al., 2021; Ouyang et al., 2022) has been deprecated. As recommended by OpenAI, we replace it with `gpt-3.5-turbo-instruct` for our experiments.

Evaluation Following prior work, we use exact match accuracy as the main metric for evaluation. Results are averaged over 3 random seeds unless stated otherwise.

Model setup In our experiments, we set the total number of steps to $k = 4$, which represents the number of examples within the context. For coverage calculation, we consider a maximum local structure size of $l = 4$. In the proposed tri-encoder used in RCR, we set $\lambda = 0.1$ in Eq. (4). The scaling temperature τ for computing the RL policy is set to 0.2. The clip ratio ϵ in GRPO is set to 0.2, and the coefficient β of the KL divergence is set to 0.04 (Eq. (8)).

Training details In the SFT stage, we construct the hard negatives set by retrieving the top-50 examples using BM25 and selecting the five examples with the lowest coverage for the instance $(x, \mathcal{Z}_{t-1})$. From these five examples, one hard negative is randomly sampled. We use a batch size of 64 for SFT and train for 120 epochs with a learning rate of 1×10^{-5}, employing a linear learning rate decay scheduler. In the RL stage, we train the model with a batch size of 16 and a group size of 32 in GRPO. The model is trained for 8 epochs with a learning rate of 1×10^{-6}.

4.3 Baselines

We evaluate our method against a variety of baselines using the same inference LLM but different retrieval strategies. Both learning-free and learning-based retrievers are considered, including top-k and sequential approaches.

Cover-LS. Cover-LS (Levy et al., 2023) is an oracle setting in which the gold program is assumed to be accessible at test time. A BM25 retriever (Robertson et al., 2009) then scores and selects examples covering the symbols of the gold program.

Top-k methods. **Random** imply samples k examples from the candidate pool $\mathcal{C}$. **BM25** (Robertson et al., 2009) is a sparse retriever that scores can-

didate matches based on lexical overlaps with the query. **BERT** uses a `bert-base-uncased` encoder (Devlin et al., 2019) to embed both the query and candidate examples, retrieving the top-k by similarity (Eq. 2). **Contriever** (Izacard et al., 2022a) follows a similar setup to BERT but replaces the encoder with a pre-trained "facebook/contriever" model that is trained in an unsupervised manner. **EPR** (Rubin et al., 2022) is a learned retriever specifically for in-context example selection, where positive and negative pairs are sampled from a ranked candidate set (scored by a scoring LM's log-likelihood of the ground-truth output).

Sequential retrieval methods se^2 (Liu et al., 2024a) is a bi-encoder retriever that sequentially selects in-context examples, leveraging a large scoring LM to rank candidates. Unlike EPR, it constructs training data by iteratively extending the context sequence. We also adapt this method to our tri-encoder architecture (denoted as "se^2+tri-encoder"), while keeping all other se^2 settings unchanged. **Reinforcing Compositional Retrieval (RCR)** is our proposed method, trained via two stages (SFT and RL). We also report ablations: "RCR w/o SFT" and "RCR w/o RL".

RL variations We compare different advantage estimations A_i in Eq. (8). **NB** denotes reinforce without baseline, directly uses the reward $r(i)$ for training ($A_i = r_i$). **Remax** (Li et al., 2024) estimates a baseline by greedily sampling a sequence $o' \in \arg\max \pi_\theta(\cdot|x)$ and calculating the associated reward value, the advantage is $A_i = r_i - r(o')$. **RLOO** (Ahmadian et al., 2024) computes an unbiased return estimate by removing the contribution of each sampled sequence from the group, the advantage is calculated by $A_i = r_i - \frac{1}{k-1}\sum_{j \neq i} r_j$. **GRPO** applies the group advantage formulation from Eq. (9). While GRPO directly includes a KL-divergence term in its objective (Eq. 4), we implement the other variants adding KL penalty to the reward (Ouyang et al., 2022).

4.4 Results and Analysis

Main results and ablation studies. Table 1 presents our main experimental results. Overall, our proposed RCR method consistently outperforms both top-k and sequential baselines on most splits, underscoring the importance of explicitly modeling inter-example dependencies through a sequential selection framework. We further analyze the contribution of our two-stage training pipeline via two ablations: 1) "RCR w/o RL" omits the RL phase, relying solely on SFT. Although this setting already surpasses many baselines, adding RL yields a significant performance boost, highlighting the impact of task-specific rewards in refining the policy and aligning the retriever with the LLM's preferences. 2) "RCR w/o SFT". Here, we eliminate SFT and directly train with RL. Although competitive with certain baselines, its weaker results suggest that sequential retrieval benefits substantially from a well-initialized policy.

Dataset	w/o RL	NB	Remax	RLOO	GRPO
i.i.d.	77.85	68.57	72.14	75.71	**78.21**
Template 1	58.73	10.84	30.12	55.72	**59.64**
Template 2	28.39	6.04	19.63	26.58	**33.83**
Template 3	45.76	28.48	29.39	48.18	**48.48**
TMCD 1	50.61	44.54	47.57	50.30	**51.52**
TMCD 2	43.94	37.87	43.63	42.12	**44.24**
TMCD 3	57.58	51.51	55.75	58.18	**59.09**
Length	32.12	29.09	28.48	28.78	**33.03**

Table 2: Comparison of various RL advantage estimators on the GeoQuery dataset. GRPO consistently achieves the highest accuracy across all splits.

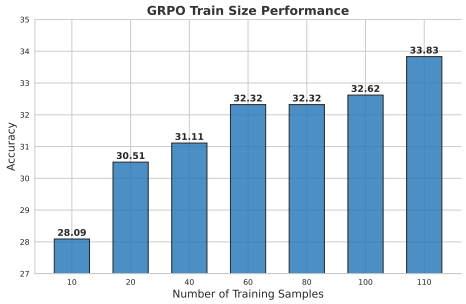

Figure 5: Performance over various numbers of training samples in GRPO. Results are evaluated in GeoQuery-Template2.

Tri-encoder architecture brings notable gains. We also compare "se^2+ tri-encoder" with se^2 to assess the impact of our tri-encoder architecture. Adopting the tri-encoder in se^2 markedly improves results, suggesting that separating the encoding of query and retrieved examples is beneficial, especially for program parsing, where appending retrieved programs directly to the query can obscure the original query signal and promote repetitive selection (e.g., selecting items that mirror already retrieved examples due to the addition to the context of original query). Moreover, while se^2 relies on GPT-Neo-2.7B (Black et al., 2022) to rank candidates, our approach leverages an efficient

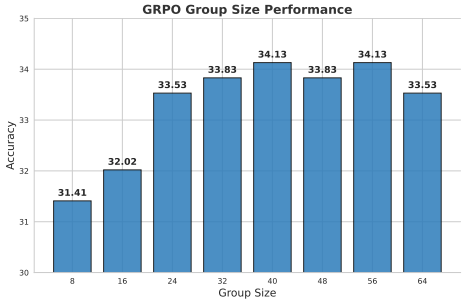

Figure 6: Performance over various group size in GRPO. Results are evaluated in GeoQuery-Template2.

data construction strategy based on maximizing sub-structure coverage, forgoing the computational overhead of scoring-based systems. As evidenced by the comparison of "RCR w/o RL" versus "se^2+ tri-encoder", our method achieves higher accuracy under a more efficient training paradigm.

Normalized advantage yields greater stability. Table 2 compares different advantage-estimation strategies for reinforcement learning. We observe that GRPO consistently provides the highest accuracy across all GeoQuery splits, indicating that its group-based advantage formulation is more stable and robust compared to alternative methods. By contrast, NB uses raw rewards without a baseline, yields noticeably lower scores, particularly on the more challenging Template 1 and 2 splits.

Impact of training sample size in RL stage Figure 5 shows how accuracy varies with the number of training samples in our RL stage. Initially, a significant improvement is observed as the number of training samples increases from 10 to 40, demonstrating the strong impact of additional data in the early stages. performance gains become more incremental, with accuracy stabilizing between 60 and 100 samples, where additional data contributes less noticeably. Further increasing the training size to 100 and 110 samples still leads to continued progress. This pattern indicates that the RL stage will continue to benefit from additional training data, albeit at a diminishing rate.

Impact of group size in GRPO The experiment results in Figure 6 show that in GRPO, accuracy improves with larger group sizes up to a threshold, after which gains diminish. Smaller groups perform worse, while moderate sizes yield significant improvements. Beyond an optimal size, further increases offer little benefit and may introduce inefficiencies. Balancing group size for both accuracy and computational efficiency is key to maximizing GRPO performance in our experiments.

5 Related Work

Early example selection methods retrieve semantically similar demonstrations (Liu et al., 2022), but semantic proximity alone does not always yield optimal exemplars. To improve selection, Rubin et al. (2022) propose training a dense retriever with a scoring LM, ranking candidates by the log-likelihood of the ground-truth output. Extensions further refine this retriever across multiple tasks (Li et al., 2023; Wang et al., 2024). Other studies highlight the importance of diversity in retrieval, such as structure or label diversity (An et al., 2023; Long et al., 2024; Levy et al., 2023; Ye et al., 2023). However, many of these approaches treat each candidate independently, overlooking inter-example dependencies that influence ICL effectiveness.

Recent empirical studies (An et al., 2023; Long et al., 2024) reveal that **diversity** (e.g., structure diversity, label diversity) also play an important role in demonstration selection. Levy et al. (2023) select diverse demonstrations that aims to collectively cover all of the required structures in the semantic parsing program. Ye et al. (2023) learns to select diverse example set based on the conditional determinantal point process. However, previous works (Rubin et al., 2022; Wang et al., 2024) overlook the interplay between in-context examples as they treat each candidate example independently. This may be suboptimal as the in-context examples can influence each other, previous work indicate that ICL is sensitive to the order of in-context examples (Lu et al., 2022). To capture the **exemplar dependency**, recent works (Liu et al., 2024b) and (Liu et al., 2024a) propose a data construction method by sequentially select and extend the in-context sequence from a LLM, the constructed dependency-aware data will be leverage to train a retriever via contrastive loss.

Different from indirect method that learning dependency via constructed training data, Zhang et al. (2022) directly frame ICL problem as a decision making problem and use offline Q-learning to actively select examples from unlabeled set to label. However, this method does not learn a retriever to acquire effective examples for each test instance, during the inference, this method will traverse the entire training set to greedly choose an action based on the Q-value which is produced by a scoring LM as well. Therefore, this method is costly when

action space (i.e., training dataset) is large and it lacks diversity. Although our method is also decision making, our method presents key differences with Zhang et al. (2022), as our method is retrieval-based, allowing sample and inference efficiency and to explore diversity. A similar work framing retrieval as MDP is proposed by Chen et al. (2024), their reward is primarily likelihood-oriented (white-box), whereas ours focuses on response-oriented (black-box). Besides, our reward design focus heavily on structural features which is crucial for program generation.

A comprehensive comparison of related methods and their properties is provided in Appendix C, with a detailed summary in Table 5.

6 Conclusion

In this work, we introduce compositional retrieval, a retrieval paradigm that models the selection of multiple interdependent examples. To achieve this, we propose a tri-encoder sequential retriever, formulating retrieval as a MDP and training it in two stages SFT via efficient data construction and RL for policy refinement. Experiments highlight the potential of compositional retrieval for tasks requiring multiple sources of evidence and offer insights into the benefits and limitations of RL in retrieval.

7 Limitations

While our approach effectively models compositional retrieval, it has several limitations. First, similar to generative retrieval, our retriever cannot dynamically incorporate new entries after training. Second, as retrieval steps increase, computational costs grow, and learning stability decreases due to compounding errors in sequential selection. To mitigate this, we focus on retrieval with a small number of steps, balancing efficiency and effectiveness. Future work could explore scalable retrieval mechanisms that maintain stability over longer sequences.

8 Acknowledgment

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (023618-00001, RG99/23), and the NTU Start-Up Grant (#023284-00001), Singapore.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. 2023. How do in-context examples affect compositional generalization? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11027–11052.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119, pages 1597–1607.

Yunmo Chen, Tongfei Chen, Harsh Jhamtani, Patrick Xia, Richard Shin, Jason Eisner, and Benjamin Van Durme. 2024. Learning to retrieve iteratively for in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7156–7168.

Daya Guo DeepSeek-AI, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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 (Volume 1: Long Papers)*, pages 4644–4668.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2024. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *Forty-first International Conference on Machine Learning*.

Haoyu Liu, Jianfeng Liu, Shaohan Huang, Yuefeng Zhan, Hao Sun, Weiwei Deng, Furu Wei, and Qi Zhang. 2024a. se^2: Sequential example selection for in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5262–5284.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B 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.

Wenhan Liu, Yutao Zhu, and Zhicheng Dou. 2024b. Demorank: Selecting effective demonstrations for large language models in ranking task. *arXiv preprint arXiv:2406.16332*.

Quanyu Long, Yin Wu, Wenya Wang, and Sinno Jialin Pan. 2024. Does in-context learning really learn? rethinking how large language models respond and solve tasks via in-context learning. In *First Conference on Language Modeling*.

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.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *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*, pages 2655–2671.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic

parsing approach handle both? 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 922–938.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

Liang Wang, Nan Yang, and Furu Wei. 2024. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 39818–39833.

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-Volume 2*, pages 1050–1055.

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.

A Dataset details

Dataset	Split	Train	Development	Test
GeoQuery	Standard	600	-	280
	Template1	438	110	332
	Template2	439	110	331
	Template3	440	110	330
	TMCD1	440	110	330
	TMCD2	440	110	330
	TMCD3	440	110	330
	Length	440	110	330
COVR-10	Each split	3000	-	500

Table 3: Dataset sizes

GeoQuery. GeoQuery (Zelle and Mooney, 1996; Tang and Mooney, 2001) is a widely used semantic parsing dataset containing 880 natural language questions about U.S. geography. Each query is labeled with a corresponding logical form that captures the precise meaning of the question. Following Shaw et al. (2021), we adopt three different data splits designed to test various aspects of compositional generalization:

- Template Split: In this split, target logical forms are anonymized into templates, and then these templates are randomly partitioned between the training and test sets (Finegan-Dollak et al., 2018). This procedure ensures that the model cannot simply memorize specific entity names and must instead learn the underlying template patterns.
- TMCD Split: The TMCD split (Keysers et al., 2020) is constructed so that the distribution of compounds (sub-expressions in the logical forms) in the training set differs significantly from that in the test set. This presents a more challenging scenario, as the model must generalize to unseen or underrepresented logical substructures that do not appear (or rarely appear) in the training examples.
- Length Split: Here, test set queries and their corresponding logical forms are systematically longer than those in the training set. By ensuring a length discrepancy, this split evaluates whether the model can generalize to deeper or more complex logical forms that exceed the length of any training example.

While some previous works average performance results across multiple runs or across different TMCD/template variants, we present individual split results to show precisely how our model behaves in each compositional scenario.

COVR-10. COVR (Bogin et al., 2022) is a synthetic dataset designed for testing compositional generalization in a variable-free functional language. Unlike GeoQuery, which focuses on a real-world domain (U.S. geography) with a relatively small set of queries, COVR generates a more diverse and systematically controlled set of expressions.

In particular, COVR-10 consists of 10 different compositional grammar splits, each crafted such that the test set includes certain local structures (sub-functional forms) that never appear during training. This forces a model to extrapolate beyond the specific structures seen in training and handle previously unseen or novel combinations of language constructs. Across these splits, the dataset's design systematically varies the presence or arrangement of operations like *largest*, *filter*, or *logical connectives* in the target expressions. Each split highlights a different angle of compositional difficulty. By evaluating on all 10 splits individually, we can observe how robustly our approach adapts to a range of compositional challenges.

Statistics are provided in Table. 3.

We use 110 samples for the GRPO training stage across all benchmarks for a fair comparison. Specifically, we split the training dataset into two subsets: 110 samples for the GRPO stage and the remaining samples as retrieval candidate pool. Due to the limited number of total samples in the GeoQuery dataset (e.g., only 440 samples in GeoQuery Template 3, TMCD, and Length tasks as indicated in Table. 3), increasing the GRPO training sample size would significantly reduce the candidate set, adversely impacting retrieval performance. Therefore, we maintained 110 GRPO training samples to ensure sufficient candidates remain available for effective retrieval.

B Local Structure

Table 4 demonstrates several sample local structures with size ≤ 4 extracted from a program. We collect all such structures (from size 1 to 4) into a set, denoted by $LS^4(x)$.

Formally, following Bogin et al. (2022); Levy et al. (2023), we define *local structures* as small, connected subgraphs of its parse tree. Given a target program y, we first convert it into a tree $T = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ is labeled with a program symbol (e.g., a function, operator, or constant). The edge set $\mathcal{E}$ encodes parent-child relationships, corresponding to function–argument links in y. Next, following Levy et al. (2023), we introduce sibling relations to form a graph $G = (\mathcal{V}, \mathcal{E} \cup \mathcal{E}_{\text{sib}})$. Concretely, for each parent node p in T with children $(c_1^p, \ldots, c_{N_p}^p)$, we connect consecutive children with edges (c_i^p, c_{i+1}^p), thus collecting them in $\mathcal{E}_{\text{sib}}$. This step reflects the intuition that sibling nodes often appear as parallel arguments for the same function or operator.

A local structure of size n is then any connected subgraph $G_{\text{LS}} \subseteq G$ with exactly n nodes, satisfying the condition that any pair of nodes (x, y) in G_{LS} is connected by a sibling edge in $\mathcal{E}_{\text{sib}}$ if and only if both x and y are leaf nodes within G_{LS}. In simpler terms, these subgraphs capture hierarchical (parent-child) and lateral (sibling) relationships while excluding more distant familial ties (e.g., "cousins" or "uncles" in the broader tree).

C Comparison to Related Works

Table 5 summarizes key differences between our approach and several existing in-context learning (ICL) methods across multiple dimensions: whether the retriever is learned, whether a separate scoring LM is required, whether the method optimizes the entire composition of in-context examples, whether it uses reward modeling, whether it supports few-shot training, how it handles exemplar dependency, and whether it considers diversity in retrieval.

Early example selection strategies often rely on semantic similarity between the query and candidate examples (Liu et al., 2022). However, semantic proximity alone may overlook other factors crucial for downstream performance (e.g., structural coverage in logic-based tasks). To address this, Rubin et al. (2022) propose training a dense retriever guided by a **scoring LM**, where the log-likelihood of the ground-truth output ranks each candidate example.

Other research emphasizes **diversity** in retrieval. Levy et al. (2023) cover essential symbols in a semantic parsing program by selecting demonstration examples collectively, while Ye et al. (2023) adopt a conditional determinantal point process to learn a diverse example set. Nonetheless, these methods primarily treat each candidate independently, so interactions among selected examples are often overlooked.

Recent work addresses this **exemplar dependency** by constructing sequential training data that captures how each newly selected example influences future choices. For instance, Liu et al. (2024b) and Liu et al. (2024a) iteratively extend in-context examples based on a large LM's feedback, then train a retriever with contrastive loss on this data. Although this approach integrates exemplar interdependence into the training set, it still relies on a scoring LM at construction time and does not formulate retrieval as a direct decision process.

Our proposed method directly models compositional retrieval as a MDP, yielding a fully learned retriever that is both efficient and capable of inter-example conditioning. We do not require an external scoring LM, relying instead on a tri-encoder architecture and a reward signal based on local structure coverage. This design allows us to optimize over the entire in-context composition (instead of independently scoring

Dataset	COVR-10
Utterance	*What is the number of black mouse that is playing with dog ?*
Program	`count ( with_relation ( filter ( black , find ( mouse ) ) , playing with , find ( dog ) ) )`
Anonymized Program	`count ( with_relation ( filter ( ANON_TYPE_VALUE , find ( ANON_ENTITY ) ) , ANON_RELATION , find ( ANON_ENTITY ) ) )`

Size	**Local structures**
1	count with_relation filter black find mouse playing with dog
2	<root> → count count → with_relation filter → black filter → find find → dog find → mouse playing → with with_relation → filter with_relation → find with_relation → playing black ↔ find filter ↔ playing playing ↔ find
3	<root> → count → with_relation count → with_relation → filter count → with_relation → find count → with_relation → playing filter → find → mouse filter → black ↔ find with_relation → filter → black with_relation → filter → find with_relation → find → dog with_relation → playing → with with_relation ↔ filter ↔ playing with_relation → playing ↔ find filter ↔ playing ↔ find
4	<root> → count → with_relation → filter <root> → count → with_relation → find <root> → count → with_relation → playing count → with_relation → filter → black count → with_relation → filter → find count → with_relation → find → dog count → with_relation → playing → with count → with_relation → filter ↔ playing count → with_relation → playing ↔ find with_relation → filter → find → mouse with_relation → filter → black ↔ find with_relation → filter ↔ playing ↔ find

Table 4: Local structures of different sizes for a specific example (→ denotes parent-child relations, ↔ denotes sibling relations)

each example) and supports training with few data samples. Furthermore, unlike prior work that treats example dependency indirectly through data construction alone, our method explicitly captures it in both the model architecture and the training objective. Finally, our approach naturally promotes diversity by sequentially selecting items to maximize coverage, thereby improving ICL effectiveness without incurring the computational overhead of large scoring LMs.

Methods	Scoring LM	Black-box LM	Compositional Optimization	Reward Modeling	Few-Shot Training	Exemplar Dependency	Diversity
CSL-Aug (Qiu et al., 2022)	✗	✗	✓	✗	✗	✗	✗
EPR (Rubin et al., 2022)	✓	✗	✗	✗	✗	✗	✗
Active-RL (Zhang et al., 2022)	✓	✗	✓	✗	✓	✓	✗
Cover-LS (Levy et al., 2023)	✗	✗	✓	✗	✗	✗	✓
CEIL (Ye et al., 2023)	✓	✗	✓	✗	✗	✗	✓
LLM-R (Wang et al., 2024)	✓	✗	✗	✓	✗	✗	✗
Se^2 (Liu et al., 2024a)	✓	✗	✗	✗	✗	✓	✓
ITERR (Chen et al., 2024)	✓	✗	✓	✓	✓	✓	✓
RCR (**Ours**)	✓	✓	✓	✓	✓	✓	✓

Table 5: Comparison of different in-context learning (ICL) methods in terms of several desirable properties. **Scoring-LM** means whether the in-context examples are ranked by a language model (LM) to obtain the training signal. **Black-box LM** means whether the in-context examples are ranked by a language model (LM) without accessing the model probability. **Compo Optim**, compositional optimizing method, which scores and optimizes over the entire in-context examples set and can generalize well according to compositionality. **Reward Modeling** trains a reward model to capture the differences of in-context examples and provide fine-grained LM preference. **Few-Shot Training** means the example selection model is trained with only few data samples. **Exemplar Dependency** means that selection of in-context examples is based on the conditional probability of already selected examples. **Diversity** represents the method considers increasing the diversity of selected set of examples. For detailed differences compared to previous works, see Section 5 and Appendix B for more discussion.

A similar work is proposed by Chen et al. (2024), while both works share the perspective of framing retrieval as MDP, our approaches differ in key aspects: First, Chen et al. (2024) use a GRU-based retriever, while we adopt a tri-encoder architecture designed for better control of state and candidate representations during sequential retrieval. Second, their reward is primarily likelihood-oriented (white-box), whereas ours focuses on task-specific structural correctness (black-box), our black-box setting is more general than white-box setting. Besides, our reward design focus heavily on structural features which is crucial for program generation. Third, we adopt SFT and GRPO (a customized policy optimization), which complements their approach and reflects different priorities in training efficiency and stability.

D Example Cases of Tasks

In Table 4, we provide prompts example in few-shot ICL manner for each task, We add special prefixes "source:" and "target:" for retrieved source-target pairs and separate them with break lines.

E Case Study

Table 7 presents various methods for retrieving program examples. BM25 and EPR are top-k retrieval methods designed to find the most similar examples. However, a key limitation of these methods is that they may retrieve examples containing the same program, leading to a lack of diversity and potentially failing to cover the gold program. se^2 employs sequential retrieval, selecting examples one by one, which introduces some level of diversity. Nevertheless, it still primarily focuses on retrieving similar examples and does not provide sufficient diversity to fully cover the gold program. In contrast, our proposed method, RCR, leverages coverage-aware learning using SFT and RL to retrieve a more diverse set of examples, effectively improving coverage of the gold program.

Table 6: Prompts examples for LLM inference.

Datasets: Geoquery

Utterance: *what is the highest point in states bordering georgia*

Gold Program: `answer(highest(place(loc_2(state(next_to_2(stateid('value')))))))`

Source: *which capitals are in the states that border texas*

Target: `(capital(loc_2(state(next_to_2(stateid('value'))))))`

Source: *what is the largest city in states that border california*

Target: `answer(largest(city(loc_2(state(next_to_2(stateid('value')))))))`

Source: *what are the capitals of the states that border texas*

Target: `answer(capital(loc_2(state(next_to_2(stateid('value'))))))`

Source: *which state has the lowest point that borders idaho*

Target: `answer(state(loc_1(lowest(place(loc_2(next_to_2(stateid('value'))))))))`

Source: *what is the highest point in states bordering georgia*

Target:

Datasets: COVR-10

Utterance: *Either there is round animal or the color of mouse that is looking at cat is equal to round*

Gold Program: `or ( exists ( filter ( round , find ( animal ) ) ) , eq ( query_attr [ color ] ( with_relation ( find ( mouse ) , looking at , find ( cat ) ) ) , round ) )`

Source: *Either the shape of animal is equal to triangle or the shape of square cat that is looking at animal is equal to brown*

Target: `or ( eq ( query_attr [ shape ] ( find ( animal ) ) , triangle ) , eq ( query_attr [ shape ] ( with_relation ( filter ( square , find ( cat ) ) , looking at , find ( animal ) ) ) , brown ) )`

Source: *Either the color of square triangle animal that is looking at brown animal that is looking at gray mouse is equal to triangle or the shape of black gray cat is equal to square*

Target: `or (eq (query_attr [color] (with_relation (filter (square , filter (triangle , find (animal))) , looking at , with_relation (filter (brown , find (animal)) , looking at , filter (gray , find (mouse))))) , triangle) , eq (query_attr [shape] (filter (black , filter (gray , find (cat)))) , square))`

Source: *Either the color of white mouse is equal to square or the shape of white dog that is looking at triangle dog is equal to the color of square triangle mouse*

Target: `or ( eq ( query_attr [ color ] ( filter ( white , find ( mouse ) ) ) , square ) , eq ( query_attr [ shape ] ( with_relation ( filter ( white , find ( dog ) ) , looking at , filter ( triangle , find ( dog ) ) ) ) , query_attr [ color ] ( filter ( square , filter ( triangle , find ( mouse ) ) ) ) ) )`

Source: *Either the color of black cat is equal to gray or none of round black dog are looking at mouse that is looking at cat*

Target: `or ( eq ( query_attr [ color ] ( filter ( black , find ( cat ) ) ) , gray ) , none ( filter ( round , filter ( black , find ( dog ) ) ) , with_relation ( scene ( ) , looking at , with_relation ( find ( mouse ) , looking at , find ( cat ) ) ) ) )`

Source: *Either there is round animal or the color of mouse that is looking at cat is equal to round*

Target:

Dataset	GeoQuery
Utterance	*what is the highest point in states bordering georgia*
Gold Program	`answer(highest(place( loc_2(state(next_to_2(stateid('value')))))))`
BM25	source: *what is the highest point in the us* target: `answer(highest(place(loc_2(countryid('value')))))` source: *what is the highest point in the usa* target: `answer(highest(place(loc_2(countryid('value')))))` source: *what states have no bordering state* target: `answer(exclude(state(all), next_to_2(state(all))))` source: *what is the highest point in the united states* target: `answer(highest(place(loc_2(countryid('value')))))` source: *what is the highest point in states bordering georgia* target:
EPR	source: *what is the highest point in montana* target: `answer(highest(place(loc_2(stateid('value')))))` source: *what is the highest point in new mexico* target: `answer(highest(place(loc_2(stateid('value')))))` source: *what is the highest point in rhode island* target: `answer(highest(place(loc_2(stateid('value')))))` source: *what is the highest point in virginia* target: `answer(highest(place(loc_2(stateid('value')))))` source: *what is the highest point in states bordering georgia* target:
se^2	source: *what state that borders texas is the largest* target: `answer(largest( state(next_to_2(stateid('value')))))` source: *what states border states that border mississippi* target: `answer(state(next_to_2( state(next_to_2(stateid('value')))))))` source: *what states border states that border states that border florida* target: `answer(state(next_to_2(state(next_to_2( state(next_to_2(stateid('value'))))))))))` source: *what are the capitals of the states that border texas* target: `answer(capital( loc_2(state(next_to_2(stateid('value')))))))` source: *what is the highest point in states bordering georgia* target:
RCR	source: *which capitals are in the states that border texas* target: `answer(capital( loc_2(state(next_to_2(stateid('value'))))))` source: *what is the highest point in the us* target: `answer(highest(place( (loc_2(countryid('value')))))` source: *what is the largest city in states that border california* target: `answer(largest(city( loc_2(state(next_to_2(stateid('value')))))))` source: *what are the capitals of the states that border texas* target: `answer(capital( loc_2(state(next_to_2(stateid('value'))))))` source: *what is the highest point in states bordering georgia* target:

Table 7: Program examples produced with various retrieval methods for a specific test example. Each prompt contains $k = 4$ examples.

F Additional Results

F.1 GRPO Training Data Size

To analyze the effect of more GRPO training data, we ran experiments on the COVR-10 dataset. The results below show a trend where increasing the number of GRPO training samples improves performance initially and then gradually plateaus. This suggests that our method is capable of scaling with more training samples, but the marginal gains diminish after a certain point, indicating a convergence behavior. We believe this pattern aligns with typical reinforcement learning settings, where policy improvement benefits from more examples, but only up to the point where the additional data becomes less informative or redundant.

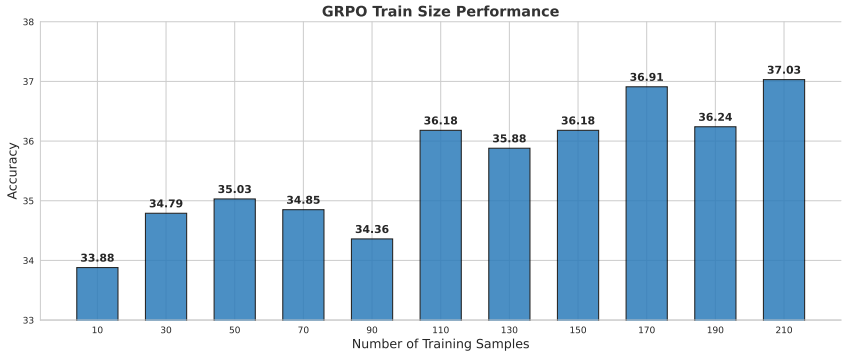

Figure 7: Performance over various numbers of training samples in GRPO. Results are evaluated in COVR-10.

F.2 Correctness-based Reward

Program generation task enables the use of a continuous and informative reward (local structure coverage, Eq. (6)), significantly benefiting RL training and analysis. Although our main focus is Compositional Generalization tasks for generating semantic parsing program, we conduct additional experiments using a simpler, binary reward formulation $r \in \{0, 1\}$, reflecting correctness of the model's final output. This discrete reward design aligns with use cases such as multi-hop question answering or classification tasks.

	GeoQuery							
	i.i.d.	Template 1	Template 2	Template 3	TMCD 1	TMCD 2	TMCD 3	Length
Coverage reward	**78.21**	**59.64**	33.83	**48.48**	**51.52**	44.24	**59.09**	**33.03**
Correctness reward	77.85	59.03	**36.85**	47.87	50.60	**45.15**	58.48	32.42

Table 8: Exact-match accuracy on GeoQuery using different rewards.

Results in Table 8 demonstrate our method's robust performance, even with a binary correctness reward that is less informative than our original continuous reward, underscoring the broader applicability and generalization potential of our approach.

F.3 Many-shot Setting for Compositional Generalization

As larger context sizes enable LLMs to process extensive information directly, potentially reducing explicit retrieval needs. To evaluate if the parsing program generation task can be solved by more in-context examples and longer contexts, we compare our retrieval-based approach against a many-shot setting in our ICL scenario. The results are summarized below:

Table 9 demonstrates that while increasing the number of in-context examples does improve performance up to a certain point (10 $\rightarrow$ 100 examples), further addition can negatively impact performance due to redundancy or repetitive information (as performance drops slightly from 63.21% at 100 examples to 62.50% at 150 examples). This observation aligns with our discussion in the main text where we note that repetitive or redundant examples may provide limited additional information or even degrade

#examples	10	20	100	150	4 (our trained RCR)
i.i.d. split	21.42	35.71	63.21	62.50	**78.21**

Table 9: Exact-match accuracy on GeoQuery i.i.d. split using different number of examples k (affecting testing only).

model performance in program generation. Our retrieval-based method, by carefully selecting fewer but more informative examples (4 steps), significantly outperforms the many-shot long-context approach, highlighting the efficiency and effectiveness of explicit compositional retrieval.

F.4 Trade-off Between Retrieval Accuracy and Efficiency

We conducted a detailed analysis of how the number of retrieval steps impacts both performance and computational efficiency. Figure 8 demonstrates our method's performance across varying numbers of retrieval steps, as well as the time spent when evaluating the model on the entire test split:

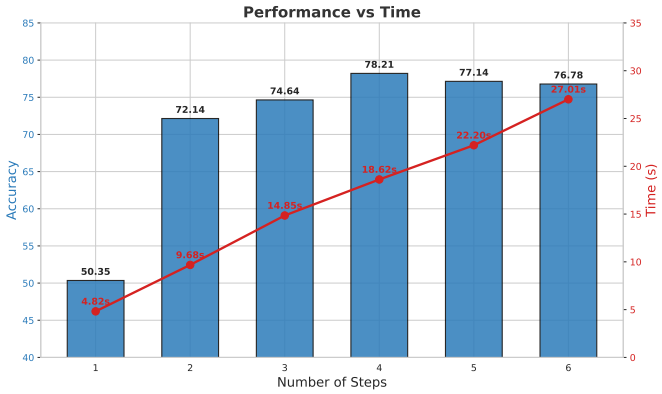

Figure 8: Exact-match accuracy on GeoQuery i.i.d. split using different number of steps k (affecting both training and testing), we report the time spent when evaluating the model on the entire test split.

As illustrated in Figure 8, our method achieves optimal performance with just 4 retrieval steps, which significantly reduces the computational overhead compared to longer sequences. Fewer steps directly translate to less computational time and lower costs.

Additionally, the specific design of our tri-encoder architecture further mitigates computational concerns. Notably, during sequential retrieval, as we seperately encode the input and retrieved candidates using two different encoder and add two embeddings at the top of the encoders, embeddings for the query need to be computed only once since the query remains unchanged throughout the retrieval process. Specifically:

- At step 0, we compute the query embedding.
- At each subsequent step $t > 0$, only embeddings for newly retrieved candidates at step $t - 1$ are computed.

Therefore, the complexity of our retriever scales linearly as $O(n)$, where n is the number of retrieval steps. This linear scaling ensures efficiency even when extending to more steps. In summary, our approach balances retrieval accuracy with computational efficiency by achieving optimal performance within very few steps. We appreciate the reviewer's feedback, which allowed us to better highlight and clarify these important points.