CO-ADAPTIVE RULE ROUTING AND CONSTRAINING FOR STRUCTURED KNOWLEDGE RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured knowledge retrieval aims to acquire task-relevant evidence from structured knowledge sources for downstream task execution. Existing methods either rely on unconstrained retrieval or predefined query-rule supervision (e.g., author–paper for seeking papers in scientific discovery and anatomy–drug for precision medicine in biomedical analysis), which is rarely available in practice. To address this limitation, we propose **Co**-Adaptive **R**ule Routing and Constraining R**E**trieval (**CoRE**), a framework that alternately determines *which rule to route each query to* and *how to retrieve under that rule without requiring predefined query–rule pairs for training supervision*. Specifically, a routing agent assigns each query to a reasoning rule, and the corresponding retriever traverses the structured knowledge base and ranks evidence that satisfies the rule's constraints. Since the rule-routing agent and rule-constraining retriever are inherently interdependent, we formulate an alternate optimization that couples reinforcement learning for rule routing with contrastive learning for the rule-constraining retriever. This closed-loop learning enables both components to co-adapt without explicit query–rule supervision. Experiments on STaRK-QA datasets, spanning scientific discovery, biomedical reasoning, and E-commerce enterprise knowledge management domains, verify the superiority of our proposed CoRE. Ablations validate co-adapting both router and retriever, and generation-based routing demonstrates cross-dataset transferability and benefits from multi-domain pretraining with target-specific fine-tuning. Furthermore, marginalizing the routing policy over all queries of each dataset reveals the distribution of rule usage, informing automatic rule refinement and discovery. Our code is available at here.

1 INTRODUCTION

Retrieval over structured knowledge bases (Nickel et al., 2015) serves as the fundamental component of modern information systems, with tremendous real-world applications such as scientific discovery (Hope et al., 2020; Auer et al., 2020), biomedical reasoning (Bonner et al., 2022), and enterprise knowledge management (Raina & Gales, 2024; Pipitone & Alami, 2024). Structured knowledge retrieval requires adhering to explicit reasoning constraints, such as retrieving neighbors of specific relation types, traversing predefined meta-paths, or composing symbolic attributes. For example, a clinician searching for treatment options follows `Symptom` $\rightarrow$ `Diagnosis` $\rightarrow$ `Treatment` to infer appropriate interventions from a patient's observed symptoms and diagnosed condition (Wu et al., 2024b). A researcher tracing intellectual lineage may traverse `Paper` $\rightarrow$ `Author` $\rightarrow$ `Paper` to identify related works connected through common authorship (Wu et al., 2024b). A cybersecurity analyst investigating a potential security breach may follow `Alert` $\rightarrow$ `Host` $\rightarrow$ `Process` $\rightarrow$ `Vulnerability` to identify the root cause of suspicious activities and assess potential attack vectors (Noel et al., 2017). In order to support rule-constrained retrieval across all these domains, a natural question arises:

How can we identify reasoning rules and use them to constrain structured knowledge retrieval?

Existing structured knowledge retrieval can be broadly categorized into three groups. The first group uses structured query and representation methods, including SQL joins, graph traversal, and symbolic rule execution, to identify relevant records, paths, or substructures. Some also learn structure-conditioned embeddings for entity retrieval (He et al., 2021; Saxena et al., 2020). However, they typically use fixed queries or a single scoring pass rather than adaptively identifying and retrieving the next useful evidence. Stepping further, the second group integrates agentic workflows into structured

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

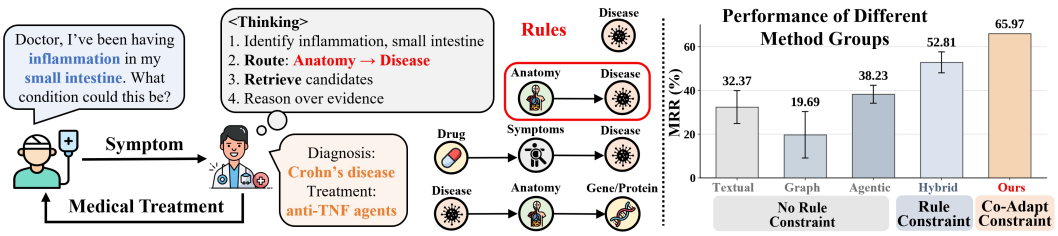

Figure 1: (a) Given a patient's biomedical query, structured knowledge retrieval is guided by the reasoning rule `Anatomy` → `Disease`, which a doctor gradually acquires through years of clinical experience. (b) Co-adapting the rule router and rule-constraining retriever further improves retrieval performance.

knowledge retrieval, where agents (Han et al., 2026; Bai et al., 2026b) dynamically explore the structured knowledge base and select subsequent actions based on the global query and intermediate observations. However, these agentic retrieval methods still struggle to adhere to domain-specific rule constraints. This limitation further motivates the third group, reasoning-constraining retrieval, which leverages predefined reasoning structures, such as meta-paths (Luo et al., 2023; Sun et al., 2023; Lei et al., 2025) and formal logic (Faltings et al., 2025), to guide structured retrieval. However, the underlying reasoning rule for each query often remains hidden in practice. Existing methods (Luo et al., 2023; Han et al., 2026; Lei et al., 2025) therefore rely on heuristics or LLM knowledge to construct query-rule pair supervision, which introduces construction bias and fails to capture the reasoning rationale required by the query, as empirically demonstrated in Appendix C

To address the lack of explicit query-rule pairs, we formulate reasoning-rule selection as a latent agentic routing problem, where the routing agent is learned directly from downstream feedback obtained by executing retrieval under the selected rule. This formulation introduces two key challenges. *First, rule routing and rule-constraining retrieval are inherently coupled.* The router determines the query-rule assignments used to train rule-constraining retrievers, while retrieval performance provides feedback to improve the routing policy. *Second, training rule router and rule-constraining retrievers require fundamentally different learning signals.* The router should identify the rule most likely to yield successful retrieval, whereas each retriever should learn to rank relevant evidence against irrelevant evidence under its assigned rule. To address these two challenges, we propose **Co**-Adaptive **R**ule Routing and Constraining R**E**trieval (**CoRE**), an alternate optimization framework that jointly adapts the rule-routing agent and rule-constraining retrievers. Specifically, the rule-routing agent is optimized via reinforcement learning using downstream retrieval feedback, while rule-constraining retrievers are optimized via contrastive learning over queries assigned by the rule-routing agent. Alternating between these two stages enables routing and retrieval to co-adapt, progressively identifying the appropriate reasoning rule for each query and improving retrieval under the corresponding rule constraints. Our contributions are summarized as follows:

- **Reasoning-constraining Retrieval without Assuming Query-Rule Pairs.** We formulate reasoning-rule selection as a latent routing problem and decompose structured retrieval into rule routing and rule-constraining retrieval, which requires no query-rule supervision.
- **Co-Adaptive Rule-Routing Agent and Rule-Constraining Retrievers.** We propose **CoRE**, an alternate optimization framework where the agentic router assigns each query to an appropriate reasoning rule to adapt the corresponding rule-constraining retriever, while retrieval performance provides feedback to adapt the routing policy.
- **Superior Performance and New Insights.** Experiments show that **CoRE** achieves the strongest average performance. Ablations validate both adaptation directions, and the token-generation router shows cross-dataset transferability and benefits from multi-domain pretraining. Moreover, the learned routing distributions further provide insights for automatic rule mining and refinement.

In the following, we review related work in Section 2 and formulate the rule-constraining structured retrieval in Section 3. We then introduce the CoRE framework and its alternate optimization in Section 4, present experimental results and analyses in Section 5, and conclude the paper in Section 6.

2 Related Work

Our work lies at the intersection of structured knowledge retrieval and constrained agentic reasoning, while additionally addressing the joint adaptation of agent policies and reasoning-constrained retrieval tools. Accordingly, we review related work from three perspectives: (1) Structured knowledge retrieval, (2) Constrained agentic reasoning for retrieval, and (3) Co-adaptation of agents and tools.

Structured Knowledge Retrieval. Early structured retrieval methods primarily improve how textual and relational signals are combined. MFAR (Li et al., 2024) adaptively aggregates retrieval scores across structured fields, while KAR (Xia et al., 2025) augments retrieval with LLM-generated query expansions and relation filtering. More recent methods introduce richer structural reasoning. MoR (Lei et al., 2025) generates a planning graph and combines structural traversal with textual matching, whereas HybGRAG (Lee et al., 2025) and AvaTaR (Wu et al., 2024a) use LLM feedback to refine retrieval strategies or tool use. These methods increasingly adapt how structure is used, but the structural strategy itself is typically generated or optimized separately from the retriever. Our method instead treats rule selection as latent and jointly learns which structural rule to use and how to retrieve under that rule from downstream retrieval feedback.

Reasoning-Constrained and Agentic Graph Retrieval. A complementary line of work moves structural reasoning from retrieval design into the LLM policy itself. RoG (Luo et al., 2023) first generates relation paths as explicit reasoning plans and then retrieves along those paths. ToG (Sun et al., 2023) makes this process interactive by selecting relations and entities during graph exploration. RoE (Han et al., 2026) and GraphDancer (Bai et al., 2026b) further learn such exploration policies through supervised and reinforcement learning. This progression shifts from predefined retrieval structure to learned traversal decisions, but also requires the LLM to perform low-level graph exploration. Our method separates the two levels: a routing agent selects a schema-valid reasoning rule, while a specialized retriever executes that rule and ranks the reachable evidence. The router is trained directly from retrieval outcomes rather than gold reasoning paths or query–rule labels.

Co-Adaptation of Agents and Tools. Recent self-improving agents (Jiang et al., 2025) go beyond optimizing a single policy by adapting interacting components in a feedback loop (Huang et al., 2026; Li et al., 2026). CoEvolve (Yang et al., 2026) couples agent learning with an evolving training distribution, WebEvolver (Fang et al., 2025) co-adapts a web agent and world model, and Tool-R0 (Acikgoz et al., 2026) jointly improves a task generator and tool-using solver. Our work brings this co-adaptation principle to structured retrieval. The rule router determines query assignments used for specialized training rule-constrained retrievers, while retriever performance provides the reward for improving the router.

3 Notation and Problem Formulation

In this section, we define key notation and formulate the structured knowledge retrieval problem.

Notation. Consider a structured knowledge base $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ consisting of an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$, where each relation $r_{ij} \in \mathcal{R}$ connects a pair of entities $e_i, e_j \in \mathcal{E}$. Each entity e_i and relation r_{ij} is associated with its type and descriptive attributes, such as the textual descriptions used in this work. Examples include the abstract of a paper entity or the interaction description of a protein–protein relation. In many domains, effective retrieval follows recurring multi-step reasoning patterns that specify how entities and relations should be progressively traversed and composed. We represent these multi-step reasoning patterns as a set of reasoning constraint rules $\mathcal{K} = \{K | K = (K_1, K_2, \ldots, K_{L_K})\}$ where each rule K consists of an ordered sequence of L_K reasoning steps. Each step may correspond to an entity type or relation type that guides retrieval toward task-supporting evidence, e.g., a scientific literature search requires a reasoning constraint `Author`→`Paper`→`Venue`, and a biomedical diagnosis requires `Drug`→`Target`→`Disease`.

Problem Formulation. Given a user query Q, the objective is to learn a retrieval model $P_{\mathbf{\Theta}}$ parameterized by $\mathbf{\Theta}$ that retrieves supporting evidence from the structured knowledge base $\mathcal{G}$. Specifically, each training instance (Q^*, S^*) is sampled from the training data distribution $P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}}$, where $S^* \subseteq \mathcal{G}$ denotes the ground-truth evidence for answering Q^* and may correspond to a single answer entity, multiple entities, or an entity-relation subgraph. Formally, we aim to learn

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \mathbb{E}_{(Q^*,S^*)\sim P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}}}[P_{\mathbf{\Theta}}(S^* \mid Q^*, \mathcal{G})]. \tag{1}$$

Unlike prior structure retrieval methods Han et al. (2026); Lei et al. (2025), we do not assume the availability of query–rule pair supervision or access (i.e., joint distribution $P_{\mathbb{Q}\times\mathbb{K}}$). Instead, we assume only the availability of query–evidence pairs sampled from $P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}}$, where $\mathbb{S}$ denotes the ground-truth supporting evidence provided by the retrieval dataset. The reasoning rule that connects each query to its supporting evidence remains latent and should be inferred automatically.

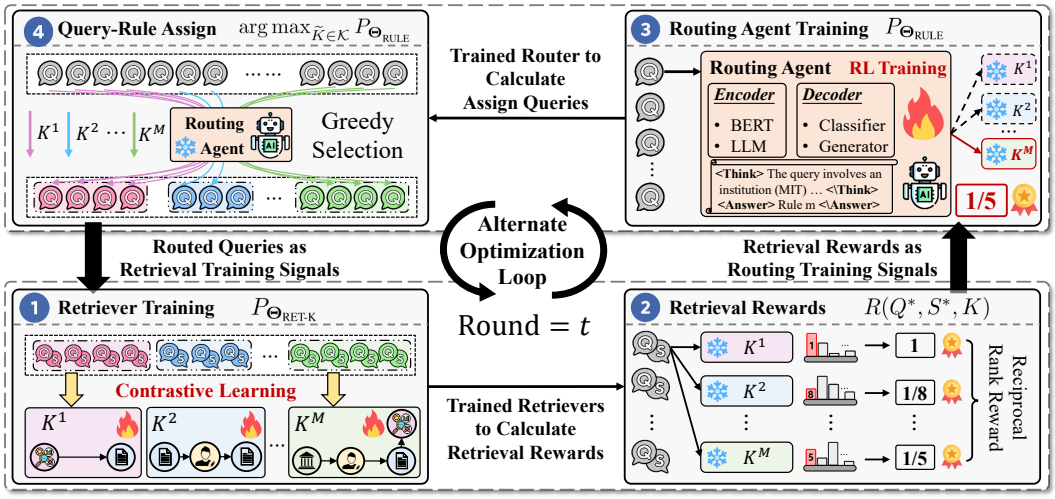

Figure 2: Overview of the CoRE framework. CoRE alternates between optimizing rule-constraining retrievers and the rule-routing agent. Given the current query–rule assignments, each retriever is first trained on queries assigned to its corresponding rule (①). The resulting retrieval performance is converted into rule-level reward signals (②), which are used to update the routing policy (③). The updated router then reassigns queries to rules (④), providing new supervision for the next round of retriever optimization. This alternating process enables the routing policy and rule-constraining retrievers to co-adapt through a closed feedback loop.

4 Framework

In this section, we first provide an overview of CoRE and then detail the design of the rule-routing agent module in Section 4.1 and the rule-constraining retrieval module in Section 4.2, respectively. Next, we present the alternative optimization together with the corresponding training strategies for both components in Section 4.3. Finally, we describe the inference procedure in Section 4.4.

As discussed earlier, a key challenge in rule-constraining structured knowledge retrieval is the absence of query–rule pair supervision, making it impossible to directly train a rule-routing policy through supervised learning. Instead, the available training data contains only observable query–evidence pairs (Q, S). Consequently, directly optimizing the retrieval objective $P_{\Theta}(S \mid Q, \mathcal{G})$ in Eq. (1) requires jointly inferring the latent reasoning rule $P_{\Theta_{\text{RULE}}}(K \mid Q, \mathcal{K})$ and retrieving evidence conditioned on the inferred rule, $P_{\Theta_{\text{RET-K}}}(S \mid Q, K, \mathcal{G})$. To address this challenge, we model the reasoning rule as a latent variable and optimize the factorized retrieval likelihood as:

$$\{\Theta^*_{\text{RET-K}}\}_{K=1}^{|\mathcal{K}|}, \Theta^*_{\text{RULE}} = \underset{\{\Theta_{\text{RET-K}}\}_{K=1}^{|\mathcal{K}|}, \Theta_{\text{RULE}}}{\arg\max} \mathbb{E}_{(Q^*,S^*)\sim P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}}} \sum_{K\in\mathcal{K}} \underbrace{P_{\Theta_{\text{RET-K}}}(S \mid Q, K, \mathcal{G})}_{\text{Rule-constraining Retriever}} \times \underbrace{P_{\Theta_{\text{RULE}}}(K \mid Q, \mathcal{K})}_{\text{Rule-routing Agent}} \quad (2)$$

where $P_{\Theta_{\text{RULE}}}(K \mid Q, \mathcal{K})$ denotes the rule routing policy that estimates the probability of assigning query Q to reasoning rule K from the rule space $\mathcal{K}$, and $P_{\Theta_{\text{RET-K}}}(S \mid Q, K, \mathcal{G})$ denotes the rule-constrained retriever that estimates the probability of retrieving evidence S from the structured knowledge source $\mathcal{G}$ conditioned on rule K.

Intuitively, the rule-routing policy acts as a high-level planner that selects the most suitable latent rule for each query, while each rule-constraining retriever specializes in retrieving evidence under its corresponding structural constraint. As shown in Figure 2, CoRE couples them through a four-stage loop. Starting from uniform query–rule assignments, ① each retriever is trained on the queries assigned to its rule; ② the trained retrievers execute the candidate rules, and the rank of the ground-truth evidence is used to construct routing rewards; ③ these rewards optimize the routing agent toward rules that produce better retrieval; and ④ the updated agent reassigns queries, providing new supervision for the next round of retriever training.

Importantly, the rule-routing policy and rule-constraining retrievers are intrinsically interdependent. The rule-routing policy determines the query–rule assignments that supervise retriever training, whereas retriever performance provides reward signals for refining the rule-routing policy. This mutual dependency precludes optimizing either component in isolation and motivates the alternative optimization framework presented in Section 4.3.

4.1 AGENTIC RULE ROUTING MODULE

The rule-routing module aims to infer the latent reasoning rule that best explains a query. Formally, given a query Q and a candidate rule set $\mathcal{K}$, the rule-routing module models the rule assignment distribution $P_{\mathbf{\Theta}_{\text{RULE}}}(K \mid Q, \mathcal{K})$. To accommodate both discriminative rule classification and generative rule prediction, we adopt a general encoder–decoder formulation: $P_{\mathbf{\Theta}_{\text{RULE}}}(K \mid Q, \mathcal{K}) = f_{\text{Decoder}}(f_{\text{Encoder}}(Q), \mathcal{K})$. The encoder transforms the query into a latent representation $\mathbf{h}_Q \in \mathbb{R}^d$, and the decoder maps $\mathbf{h}_Q$ into a rule-selection decision.

Encoder. We consider two encoder instantiations:

$$\mathbf{h}_Q = f_{\text{Encoder}}(Q) = \begin{cases} \text{BERT}(Q)_{\texttt{[CLS]}}, & \text{BERT-based encoder,} \\ \text{LLM}(Q)_{\text{Last}}, & \text{LLM-based encoder.} \end{cases} \tag{3}$$

The BERT-based encoder extracts the final hidden state of the `[CLS]` token to capture sentence-level semantic information. In contrast, the LLM-based encoder uses the final-layer hidden state of the last token, enabling richer contextual representations that better capture complex reasoning patterns.

Decoder. Given the query representation $\mathbf{h}_Q$, the decoder defines the rule-selection distribution as:

$$f_{\text{Decoder}}(\mathbf{h}_Q, \mathcal{K}) = \begin{cases} \text{Softmax}(\mathbf{W}\mathbf{h}_Q + \mathbf{b}), & \text{Classifier-based decoder,} \\ \prod_{t=1}^{T} P(y_t \mid y_{<t}, Q, \mathcal{K}), & \text{Generation-based decoder.} \end{cases} \tag{4}$$

For the classifier-based decoder, $\mathbf{W} \in \mathbb{R}^{|\mathcal{K}| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{K}|}$ are learnable parameters that produce a categorical distribution over the candidate rules in $\mathcal{K}$. A rule is subsequently selected through argmax selection or categorical sampling. Alternatively, the generation-based decoder formulates rule selection as a conditional generation task, where the generated sequence $Y = \{y_t\}_{t=1}^{T}$ is parsed into a rule assignment. The prompting format is provided in Appendix G.

Combining the encoder and decoder choices yields three variants of CoRE: **BERT-C**, which combines a BERT-based encoder with a classifier-based decoder; **LLM-C**, which combines an LLM-based encoder with a classifier-based decoder; and **LLM-G**, which combines an LLM-based encoder with a generation-based decoder. We do not instantiate a BERT-based generation variant, as coupling BERT with a generative decoder would introduce additional architecture design complexity and training choices beyond the scope of this study.

4.2 RULE-CONSTRAINING RETRIEVAL MODULE

Given a query Q and a reasoning rule K selected by the rule-routing agent, the corresponding rule-constraining retriever identifies evidence that is both semantically relevant to the query and structurally consistent with the selected rule. Formally, the likelihood of retrieving candidate evidence S_i for query Q under reasoning rule K is defined as $P_{\mathbf{\Theta}_{\text{RET-K}}}(S_i \mid Q, K, \mathcal{G}) = \frac{\exp(\Phi(Q, S_i, K))}{\sum_{S_j \in \mathcal{C}(Q,K)} \exp(\Phi(Q, S_j, K))}$, where $\mathcal{C}(Q, K)$ is the candidate evidence obtained by constrained depth-first-search (DFS) under rule K, and $\Phi(Q, S_i, K)$ is a rule-conditioned relevance score measuring the compatibility between query Q, candidate evidence S_i, and rule K. $P_{\mathbf{\Theta}_{\text{RET-K}}}$ provides a general formulation of rule-constrained retrieval. We instantiate the framework for knowledge graph retrieval, as experimented in Section 5.

For rule-constrained knowledge graph retrieval, the key is to assign each candidate entity a relevance score that reflects both its compatibility with the query and its selected rule. We therefore define the rule-conditioned relevance score $\Phi(Q, S_i, K)$ by combining rule-constrained structural information with query-dependent relevance signals. To incorporate the rule constraint, a rule K specifies a sequence of entity types, such as *Anatomy* $\rightarrow$ *Disease* $\rightarrow$ *Drug*. The retriever performs constrained graph traversal following this sequence, which restricts the search space to entities reachable under K. For each resulting candidate S_i, let $(v_{i,1}, \ldots, v_{i,L})$ denote the traversal path leading to it, where $v_{i,L} = S_i$. To incorporate query relevance, we measure the BM25 similarity between the query and the textual description of each entity along this path: $\mathbf{x}_i = [\text{BM25}(Q, v_{i,1}), \ldots, \text{BM25}(Q, v_{i,L})]$. The resulting vector $\mathbf{x}_i$ therefore summarizes how relevant the entire rule-constrained path is to the query. Finally, we combine the query-dependent path features with the representation of the selected rule. Specifically, $\mathbf{x}_i$ and $f_{\text{embed}}(K)$ are provided as inputs to a learnable scoring function: $\Phi(Q, S_i, K) = f_{\text{score}}(\mathbf{x}_i, f_{\text{embed}}(K))$, where $f_{\text{embed}}(K)$ represents the selected rule embedding and f_{score} is parameterized by $\mathbf{\Theta}_{\text{RET-}K}$. In this way, $\Phi(Q, S_i, K)$ jointly captures query relevance along the constrained traversal path and the semantics of the selected rule.

4.3 ALTERNATIVE OPTIMIZATION

Since the rule routing policy determines the latent query–rule assignments that supervise rule-constraining retrievers, while retrieval performance in turn provides feedback for improving rule routing, we therefore formulate their co-adaptation as a bi-level optimization problem and optimize them through an alternate optimization strategy (Bai et al., 2026a; Tarzanagh et al., 2024) that iteratively updates the rule-constraining retrievers and rule routing policy.

$$\boldsymbol{\Theta}^{t+1} = \begin{cases} \boldsymbol{\Theta}^{t+1}_{\text{RET-K}} = \arg\max_{\boldsymbol{\Theta}_{\text{RET-K}}} \mathcal{L}_{\text{RET}}(\{\boldsymbol{\Theta}^{t}_{\text{RET-}K^m}\}_{m=1}^{|\mathcal{K}|}, \boldsymbol{\Theta}^{t}_{\text{RULE}}), \quad \text{with } \boldsymbol{\Theta}^{t}_{\text{RULE}} \text{ fixed}, \\ \boldsymbol{\Theta}^{t+1}_{\text{RULE}} = \arg\max_{\boldsymbol{\Theta}_{\text{RULE}}} \mathcal{L}_{\text{RULE}}(\boldsymbol{\Theta}^{t}_{\text{RULE}}, \{\boldsymbol{\Theta}^{t+1}_{\text{RET-}K^m}\}_{m=1}^{|\mathcal{K}|}), \text{ with } \{\boldsymbol{\Theta}^{t+1}_{\text{RET-}K^m}\}_{m=1}^{|\mathcal{K}|} \text{ fixed}. \end{cases} \tag{5}$$

where $\mathcal{L}_{\text{RET}}$ and $\mathcal{L}_{\text{RULE}}$ denote the optimization objectives for the rule-constraining retrievers and the rule routing policy, respectively. The specific formulations are introduced next.

Rule-constraining Retrieval Optimization. With the rule routing policy fixed, each rule-constrained retriever is trained on the queries assigned to its rule by that policy. Let $\arg\max_{\widetilde{K}\in\mathcal{K}} P_{\boldsymbol{\Theta}^t_{\text{RULE}}}(\widetilde{K} \mid Q^*)$ denote the rule assigned to training pair (Q^*, S^*). Therefore, given a fixed rule assignment, the marginalization over all candidate rules in Eq. (2) reduces to the assigned rule, yielding the following:

$$\mathcal{L}_{\text{RET}} = \mathbb{E}_{(Q^*,S^*)\sim P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}}} \sum_{K\in\mathcal{K}} \mathbb{1}(K = \arg\max_{\widetilde{K}\in\mathcal{K}}\{P_{\boldsymbol{\Theta}^t_{\text{RULE}}}(\widetilde{K}|Q^*)\}) P_{\boldsymbol{\Theta}^t_{\text{RET-K}}}(S^*|Q^*, K, \mathcal{G}) \tag{6}$$

where $P_{\boldsymbol{\Theta}^t_{\text{RULE}}}(\widetilde{K}|Q^*)$ and $P_{\boldsymbol{\Theta}^t_{\text{RET-K}}}(S^*|Q^*, K, \mathcal{G})$ are calculated following Sections 4.1-4.2.

Rule routing Optimization. With the rule-constrained retrievers fixed, we next optimize the rule routing policy using retrieval feedback. Intuitively, the policy should assign high probability to reasoning rules that lead to successful evidence retrieval and low probability to rules that result in poor retrieval performance. For each training query Q^*, the rule routing policy samples a reasoning rule $K \sim P_{\boldsymbol{\Theta}^t_{\text{RULE}}}(\cdot \mid Q^*)$ which is subsequently evaluated by the optimized rule-constrained retrievers:

$$\mathcal{L}_{\text{RULE}} = \mathbb{E}_{(Q^*,S^*)\sim P^{\text{Train}}_{\mathbb{Q}\times\mathbb{S}},\ K\sim P_{\boldsymbol{\Theta}^t_{\text{RULE}}}(\cdot|Q^*)} \ell_{\text{Rank}}(Q^*, S^*, K, \{\boldsymbol{\Theta}^{t+1}_{\text{RET-K}}\}_{K\in\mathcal{K}}). \tag{7}$$

The ranking objective ℓ_{Rank} is defined based on the retrieval rank of the ground-truth evidence. Given a training pair (Q^*, S^*) and a reasoning rule K, the corresponding rule-constrained retriever ranks all candidate evidence in $\mathcal{C}(Q^*, K)$ as in Section 4.2. Let $\text{rank}(S^* \mid Q^*, K)$ denote the ranking position of the ground-truth evidence S^* among these candidates. We convert this ranking position into a scalar retrieval reward using Reciprocal Rank: $R(Q^*, S^*, K) = \frac{1}{\text{rank}(S^*|Q^*,K)}$. Depending on the policy parameterization, ℓ_{Rank} is optimized differently: classifier-based policies rank over all candidate rules, whereas generation-based policies optimize over sampled rollouts.

- **Classifier-based Rule Routing.** For classifier-based policies, the model predicts a distribution over all candidate rules, allowing rewards to be evaluated for every $K \in \mathcal{K}$. We compute a rule-wise advantage $A(Q^*, S^*, K) = R(Q^*, S^*, K) - \frac{1}{|\mathcal{K}|}\sum_{K'\in\mathcal{K}} R(Q^*, S^*, K')$, and optimize the expected advantage under the policy: $\ell_{\text{Rank}} = \sum_{K\in\mathcal{K}} P_{\Theta_{\text{RULE}}}(K \mid Q^*) A(Q^*, S^*, K)$.
- **Generation-based Rule Routing.** For generation-based policies, evaluating all candidate rules is computationally prohibitive. Instead, we sample a rollout set $\{K_1, \ldots, K_N\} \sim P_{\Theta_{\text{RULE}}}(\cdot \mid Q^*)$ and compute group-relative advantages from their Reciprocal Rank-based retrieval rewards. Following GRPO, the policy is optimized using $\ell_{\text{Rank}} = \frac{1}{N}\sum_{i=1}^{N} \min(r_i A_i, \text{clip}(r_i, 1-\epsilon, 1+\epsilon)A_i)$, where r_i is the policy ratio and A_i is the group-relative advantage.

4.4 INFERENCE

At inference time, given a query Q, the optimized rule routing policy first identifies the most probable reasoning rule. Conditioned on the selected rule, the corresponding rule-constrained retriever constructs a candidate evidence set $\mathcal{C}(Q, K^*)$, assigns each candidate a rule-conditioned relevance score $\Phi(Q, S_i, K^*)$, and returns the highest-ranked evidence:

$$K^* = \arg\max_{K\in\mathcal{K}} P_{\boldsymbol{\Theta}_{\text{RULE}}}(K \mid Q) \qquad \hat{S} = \text{TopK}_{S_i\in\mathcal{C}(Q,K^*)}\, \Phi(Q, S_i, K^*). \tag{8}$$

The retrieved evidence set $\hat{S}$ is subsequently provided to downstream task execution.

Table 1: Retrieval performance on AMAZON, MAG, and PRIME, comparing baselines with CoRE-BERT-C, CoRE-LLM-C, and CoRE-LLM-G. Best and second-best results are **bold** and underlined, respectively.

Category	Retrieval Baseline	AMAZON				MAG				PRIME				AVERAGE			
		H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR
Textual	BM25	40.62	62.18	50.89	50.59	23.11	44.47	44.02	33.23	11.96	25.67	28.31	18.35	25.23	44.11	41.07	34.06
	Ada	39.16	62.73	53.29	50.35	29.08	49.61	48.36	38.62	12.63	31.49	36.00	21.41	26.96	47.94	45.88	36.79
	Multi-Ada	40.07	64.98	55.12	51.55	25.92	50.43	50.80	36.94	15.10	33.56	38.05	23.49	27.03	49.66	47.99	37.33
	DPR	15.29	47.93	44.49	30.20	10.51	35.23	42.11	21.34	4.46	21.85	30.13	12.38	10.09	35.00	38.91	21.31
Graph	QAGNN	26.56	50.01	52.05	37.75	12.88	39.01	46.97	29.12	8.85	21.35	29.63	14.73	16.10	36.79	42.88	27.20
	ToG	–	–	–	–	13.16	16.17	11.30	14.18	6.07	15.71	13.07	10.17	9.62	15.94	12.18	12.18
Hybrid	AvaTaR	49.87	69.16	60.57	58.70	44.36	59.66	50.63	51.15	18.44	36.73	39.31	26.73	37.56	55.18	50.17	45.53
	KAR	**54.20**	68.70	57.24	61.29	50.47	69.57	60.28	58.65	30.35	49.30	50.81	39.22	45.01	62.52	56.11	53.05
	MFAR*	41.20	70.00	58.50	54.20	49.00	69.60	71.70	58.20	40.90	62.80	68.30	51.20	43.70	67.47	66.17	54.53
	HYBGRAG	–	–	–	–	**65.40**	75.31	65.70	69.80	28.56	41.38	43.58	34.49	46.98	58.35	54.64	52.15
	MoR	52.19	74.65	59.92	62.24	58.19	78.34	75.01	67.14	36.41	60.01	63.48	46.92	48.93	71.00	66.14	58.77
Agentic	RoG	40.93	62.79	36.96	49.25	24.52	45.33	30.93	35.03	14.87	22.41	23.28	19.36	26.77	43.51	30.39	34.55
	RoE	42.90	64.17	37.42	50.96	27.18	46.62	31.41	37.76	14.18	33.80	37.93	23.63	28.09	48.20	35.59	37.45
	GraphDancer	46.27	67.43	65.91	55.18	29.33	64.67	66.97	44.36	18.67	29.33	44.53	28.54	31.42	53.81	59.14	42.69
CoRE (Ours)	BERT – C	50.71	**76.19**	**61.39**	**62.46**	63.37	**87.78**	**83.51**	**74.17**	**49.34**	**76.40**	**80.81**	**61.27**	**54.47**	**80.12**	**75.24**	**65.97**
	LLM – C	50.83	75.55	60.85	62.21	60.24	83.03	77.64	71.53	45.97	74.12	79.18	58.57	52.35	77.57	72.56	64.10
	LLM – G	50.26	73.24	60.92	58.64	59.14	82.39	77.92	69.25	35.54	55.81	59.01	44.94	48.31	70.48	65.95	57.61

5 EXPERIMENT

This section aims to experimentally verify the effectiveness of CoRE. We first introduce experimental settings and then conduct experiments to answer the following questions:

- § 5.2 - $\mathbf{Q}_1$: *How does CoRE compare with existing structured retrieval methods?*
- § 5.3 - $\mathbf{Q}_2$: *How do router adaptation and retriever adaptation affect the performance?*
- § 5.4 - $\mathbf{Q}_3$: *Can the token-generation routing agents transfer learned routing knowledge across datasets and benefit from multi-domain pretraining?*
- § 5.5 - $\mathbf{Q}_4$: *How do the routing agent and retrievers co-adapt across alternate rounds?*

5.1 EXPERIMENTAL SETUP

Datasets and evaluation. We evaluate on three STaRK-QA (Wu et al., 2024b) datasets: AMAZON, MAG, and PRIME. We define 6, 8, and 65 rules by sampling all ordered entity-type sequences connected by dataset-schema-permitted relations in the traversal direction. Following our motivation, training uses only query–evidence pairs, while query-rule assignments remain latent and are inferred by the routing agent. We report Hits@1, Hits@5, Recall@20, and MRR.

Model variants. Following Section 4.1, we study three routing architectures: CoRE-BERT-C, which uses a BERT encoder and classification head; CoRE-LLM-C, which uses an LLM encoder and classification head; and CoRE-LLM-G, which generates rule selections as tokens. All variants use the same rule-constraining retriever design and undergo three alternate optimization rounds. The generation-based router is initialized from Qwen2.5-3B-Instruct and trained with eight sampled rollouts per query on four NVIDIA H200 GPUs.

Baselines. We compare against textual retrievers, including BM25 (Robertson et al., 2009), Ada (OpenAI, 2023), Multi-Ada (OpenAI, 2023), and DPR (Karpukhin et al., 2020); graph-based methods, including QAGNN (Yasunaga et al., 2021) and ToG (Sun et al., 2023); hybrid methods, including AvaTaR (Wu et al., 2024a), KAR (Xia et al., 2025), MFAR (Li et al., 2024), HYBGRAG (Lee et al., 2025), and MoR (Lei et al., 2025); and agentic graph methods, including RoG (Luo et al., 2023), RoE (Han et al., 2026), and GraphDancer (Bai et al., 2026b).

Further details, including model implementation and hyperparameter settings, are in Appendix A.

5.2 $\mathbf{Q}_1$-MAIN RESULTS

Table 1 shows that CoRE-BERT-C achieves the strongest average performance.

- **Compared with textual and graph-based methods**, all CoRE variants outperform the textual and graph-based baselines. These gains support using structural rule constraints for retrieval, particularly on MAG and PRIME, where textual matching alone performs poorly.
- **Compared with hybrid methods**, CoRE-BERT-C outperforms MoR, achieving the highest average performance among evaluated hybrid methods. While MoR relies on LLM-generated constraints that may not align with downstream retrieval needs, CoRE learns rule routing directly from retrieval feedback. This advantage is more pronounced on PRIME, where its larger domain-specific rule space makes it harder for LLMs to select appropriate rules based on their general knowledge alone.

Table 2: Ablation of routing agent and retriever adaptation. **w/o Agent-Adapt** fixes the router to a pre-trained or zero-shot model while continuing retriever training. **w/o Retriever-Adapt** replaces learned rule-constraining retrievers with fixed BM25-traversal scoring while continuing routing agent training. Best results are in **bold**.

Method	AMAZON		MAG		PRIME		Average	
	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR
CoRE-BERT-C	**50.71**	**62.46**	**63.37**	**74.17**	**49.34**	**61.27**	**54.47**	**65.97**
w/o Agent-Adapt (unadapted BERT)	14.63	19.83	28.23	35.05	5.15	8.15	16.00	21.01
w/o Agent-Adapt (GPT-5-mini)	26.70	32.72	28.23	31.87	4.14	6.66	19.69	23.75
w/o Retriever-Adapt (BM25-traversal)	49.87	61.45	55.51	65.45	38.86	49.70	48.08	58.87

- **Compared with agentic graph methods**, all CoRE variants achieve higher average scores than RoG, RoE, and GraphDancer. One explanation is that agentic methods rank evidence through step-by-step graph exploration based on intermediate node observations, whereas CoRE first constrains the search space with a reasoning rule and then ranks the fetched evidence using a rule retriever, which is more effective in these text-rich graphs.
- **Compared among CoRE variants**, CoRE-BERT-C achieves the strongest average performance, followed by CoRE-LLM-C and CoRE-LLM-G. With the same classification head, BERT's stronger results suggest that its sentence-level representation is better suited to rule routing in this setting. Both classification-based routers outperform generation-based routing on average, likely because they use reward feedback from all candidate rules rather than sampled rollouts. Limited rollouts provide less reward feedback and higher-variance gradient estimates, reducing training efficiency (Cheng et al., 2020). This limitation is most shown on PRIME, where eight rollouts per query cover only a small portion of its 65-rule space. Generation-based routing nevertheless supports cross-dataset transfer and multi-domain foundational pretraining, as in Section 5.4.

5.3 Q_2-Ablation Study of Co-Adapting Router and Retriever

Since CoRE co-adapts rule routing and rule-constrained retrieval, we ablate each adaptation direction in Table 2 and further examine their joint benefit in Appendix E due to space limitations.

Without routing agent adaptation. Replacing the learned routing agent with a fixed pre-trained BERT or zero-shot GPT-5-mini policy substantially reduces performance across all three datasets, despite continued retriever training. This suggests that poorly aligned query-rule assignments limit both rule routing quality and rule-constraining retriever specialization.

Without retriever adaptation. We replace the learned rule-constraining retrievers with fixed BM25-traversal scoring, ranking candidates under the selected rule by the mean query-node BM25 similarity along their traversal paths. This preserves the rule-constraining candidate space and BM25 relevance signals while removing learnable rule-conditioned scoring. The routing agent is still trained on the resulting retrieval rewards. Performance drops most on PRIME, followed by MAG and AMAZON. This aligns with the rule distributions in Figure 6: PRIME contains more diverse multi-hop rules that benefit from learning how to combine relevance signals across traversal steps, whereas the less-node-hop rules common on MAG and AMAZON rely largely on a few BM25 scores.

5.4 Q_3-Transferability of Generation-based Rule Routing

Unlike fixed classification heads, generation-based routers leverage LLMs to generate rules in the token space, enabling natural transfer to unseen rule sets across datasets. We therefore examine whether CoRE-LLM-G transfers across datasets and further benefits from multi-domain pretraining.

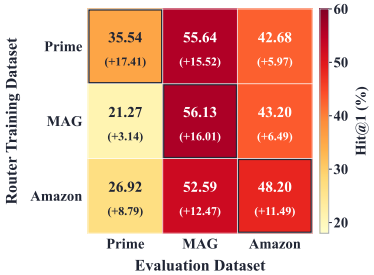

Figure 3: Cross-dataset H@1 of CoRE-LLM-G. Parentheses show gains over zero-shot Qwen with trained retrievers.

Cross-dataset transfer. We independently train CoRE routers on AMAZON, MAG, and PRIME and evaluate each router across all three datasets. We report H@1 together with its improvement over an unadapted Qwen router, using the same trained retrievers for a controlled comparison. During cross-dataset transfer, only task description, graph schema, and candidate rules are replaced; the router remains fixed and is paired with target-domain retrievers. Figure 3 shows that all transferred routers outperform zero-shot Qwen, suggesting that retrieval-feedback training learns transferable mappings between query semantics and reasoning patterns rather than memorizing dataset-specific query-rule mappings.

Foundation routing agent. Motivated by the cross-dataset transfer results above, we next ask whether routing experience from different domains can be combined to build a stronger shared router. We use Qwen2.5-32B-Instruct in this study to provide greater model capacity for consolidating heterogeneous routing knowledge across multiple domains. We compare four settings: unadapt Qwen w. TR uses the base Qwen model without routing training; CoRE-LLM-G trains a separate router on each dataset; CoRE-LLM-G Pre jointly trains one router for one epoch on the combined AMAZON, MAG, and PRIME training sets and directly evaluates it on each dataset; Finally, CoRE-LLM-G Pre-FT further fine-tunes this shared router on each target dataset. All four settings use the same Qwen2.5-32B-Instruct backbone and the same trained retrievers for the corresponding evaluation dataset.

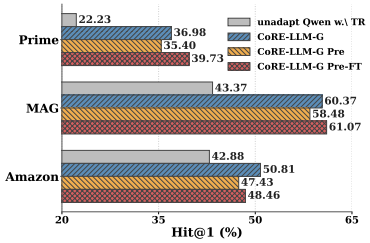

Figure 4: H@1 of CoRE-LLM-G under different training using Qwen2.5-32B-Instruct as the base router. Multi-domain pretraining followed by target-specific fine-tuning performs best.

As shown in Figure 4, the jointly pretrained routing CoRE-LLM-G Pre consistently outperforms unadapt Qwen w. TR, indicating that retrieval feedback from different domains provides useful shared knowledge for rule selection. Its performance remains slightly below dataset-specific training CoRE-LLM-G, suggesting that each domain still has its own routing patterns that are difficult to capture through shared pretraining alone. However, once further fine-tuned on the target dataset, the CoRE-LLM-G Pre-FT achieves the best generation-based routing performance overall. These results suggest a natural *pretrain-then-specialize* paradigm: first learn general routing knowledge across domains, and then adapt it to the reasoning patterns of each target domain.

5.5 Q_4-Alternate Optimization Dynamics

Figure 5 shows clear co-adaptation across three rounds: router training sharply improves H@1 from about 21% to 58%, and the resulting assignments further boost the next retriever update above 62%. Later rounds provide smaller refinements as the two components approach convergence. The inset distributions show how the router differentiates candidate rules without query-rule supervision. Starting from a uniform distribution, after the Round 1 rule-routing training, the agent assigns the highest probability to `IAP` (institution→author→paper) and `PAP` (paper→author→paper), while nearly eliminating the longer `IAPF` (institution→author→paper→field of study) and `PAPF` (paper→author→paper→field of study) rules. By Round 3, `IAP` remains dominant, while more probability shifts toward `PAP` and `PP` (paper→paper). This indicates that the router identifies useful institution-, authorship-, and citation-based patterns and refines their use as the retrievers improve. This can also support a future direction toward automatic rule discovery as new datasets continually emerge, where high-reward rules guide rule expansion while consistently unused rules are pruned.

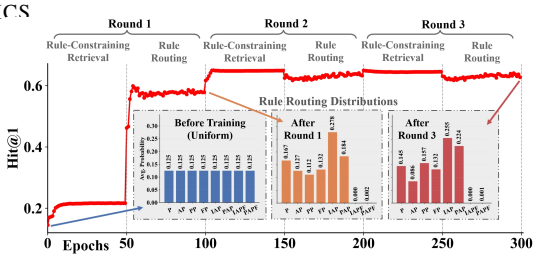

Figure 5: Optimization dynamics of CoRE-BERT-C on MAG. Each round alternates between retriever and router adaptation; the insets show the routing distributions before training and after Rounds 1 and 3.

A case study demonstrating that general LLMs struggle to identify the appropriate rules without explicit domain rule supervision is also provided in the Appendix C due to space limitations.

6 Conclusion

In this work, we study a novel problem of structured knowledge retrieval under rule-based constraints. We introduce CoRE, a structured knowledge retrieval framework that jointly learns rule routing and rule-constraining retrieval without explicit query–rule supervision. CoRE treats rule selection as a latent decision and employs alternating-based bi-level optimization to jointly specialize rule-constraining retrievers and optimize the routing agent through retrieval feedback. Experiments demonstrate that CoRE consistently outperforms existing baselines, while ablation studies confirm the complementary importance of routing-agent optimization and rule-constraining retriever adaptation. Analysis of the optimization dynamics further reveals how routing and retrieval progressively co-adapt through iterative feedback. These findings also motivate automatic rule refinement, where high-reward routing patterns guide the generation and adjustment of reasoning rules.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript review and language polishing. In particular, we leveraged generative AI to provide feedback following the ICLR review rubric, including identifying potential strengths and weaknesses, and used this feedback to further revise and improve the manuscript. We also used generative AI to assist with writing Python code for figure generation and visualization. We have carefully reviewed all AI-assisted content and take full responsibility for the final manuscript, including all text, claims, code, figures, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY

We provide the code and datasets necessary to reproduce the results shown in Figures 3, 4, 5, and Tables 1, 2 in the GitHub repository: link. The repository contains the full CoRE implementation, data preprocessing pipelines, training and evaluation scripts, experiment configurations, hyperparameters, and scripts for reproducing the reported tables and figures.

REFERENCES

Emre Can Acikgoz, Cheng Qian, Jonas Hübotter, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Tool-R0: Self-evolving LLM agents for tool-learning from zero data. *arXiv preprint arXiv:2602.21320*, 2026.

Sören Auer, Allard Oelen, Muhammad Haris, Markus Stocker, Jennifer D'Souza, Kheir Eddine Farfar, Lars Vogt, Manuel Prinz, Vitalis Wiens, and Mohamad Yaser Jaradeh. Improving access to scientific literature with knowledge graphs. *Bibliothek Forschung und Praxis*, 44(3):516–529, 2020.

Xiaoning Bai, Shangzhi Zeng, Jin Zhang, and Lezhi Zhang. Alternating gradient-type algorithm for bilevel optimization with inexact lower-level solutions via moreau envelope–based reformulation. *SIAM Journal on Optimization*, 36(1):350–380, 2026a.

Yuyang Bai, Zhuofeng Li, Ping Nie, Jianwen Xie, and Yu Zhang. Graphdancer: Training llms to explore and reason over graphs via curriculum reinforcement learning. *arXiv preprint arXiv:2602.02518*, 2026b.

Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William L Hamilton. A review of biomedical datasets relating to drug discovery: a knowledge graph perspective. *Briefings in Bioinformatics*, 23(6):bbac404, 2022.

Ching-An Cheng, Xinyan Yan, and Byron Boots. Trajectory-wise control variates for variance reduction in policy gradient methods. In *Conference on Robot Learning*, pp. 1379–1394. PMLR, 2020.

Felix Faltings, Wei Wei, and Yujia Bao. Enhancing retrieval systems with inference-time logical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 449–463, 2025.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. WebEvolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8959–8975. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.454.

Haoyu Han, Kai Guo, Harry Shomer, Yu Wang, Yucheng Chu, Hang Li, Li Ma, and Jiliang Tang. Reasoning by exploration: A unified approach to retrieval and generation over graphs. In *Proceedings of the ACM Web Conference 2026*, pp. 2408–2418, 2026.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 553–561, 2021.

Tom Hope, Jason Portenoy, Kishore Vasan, Jonathan Borchardt, Eric Horvitz, Daniel S Weld, Marti A Hearst, and Jevin West. Scisight: Combining faceted navigation and research group detection for covid-19 exploratory scientific search. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 135–143, 2020.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. In *International Conference on Learning Representations*, volume 2026, pp. 130770–130790, 2026.

Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, et al. Adaptation of agentic ai. *arXiv e-prints*, pp. arXiv–2512, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N Ioannidis, Huzefa Rangwala, and Christos Faloutsos. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 879–893, 2025.

Yongjia Lei, Haoyu Han, Ryan A Rossi, Franck Dernoncourt, Nedim Lipka, Mahantesh M Halappanavar, Jiliang Tang, and Yu Wang. Mixture of structural-and-textual retrieval over text-rich graph knowledge bases. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18306–18321, 2025.

Jiaxi Li, Ke Deng, Yun Wang, Jingyuan Huang, Yucheng Shi, Qiaoyu Tan, Jin Lu, and Ninghao Liu. Online skill learning for web agents via state-grounded dynamic retrieval. *arXiv preprint arXiv:2606.04391*, 2026.

Millicent Li, Tongfei Chen, Benjamin Van Durme, and Patrick Xia. Multi-field adaptive retrieval. *arXiv preprint arXiv:2410.20056*, 2024.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*, 2023.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

Steven Noel, Deborah Bodeau, and Rosalie McQuaid. Big-data graph knowledge bases for cyber resilience. In Alexander Kott and Gabi Dreo Rodosek (eds.), *Proceedings of the NATO IST-153/RWS-21 Workshop on Cyber Resilience*, volume 2040 of *CEUR Workshop Proceedings*, pp. 6–21, Munich, Germany, 2017. CEUR-WS.org. URL `https://ceur-ws.org/Vol-2040/paper2.pdf`.

OpenAI. Openai embeddings api, 2023. [Software].

Nicholas Pipitone and Ghita Houir Alami. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*, 2024.

Vatsal Raina and Mark Gales. Question-based retrieval using atomic units for enterprise rag. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pp. 219–233, 2024.

Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4498–4507, 2020.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*, 2023.

Davoud Ataee Tarzanagh, Parvin Nazari, Bojian Hou, Li Shen, and Laura Balzano. Online bilevel optimization: Regret analysis of online alternating gradient methods. In *International Conference on Artificial Intelligence and Statistics*, pp. 2854–2862. PMLR, 2024.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37: 25981–26010, 2024a.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *Advances in Neural Information Processing Systems*, 37: 127129–127153, 2024b.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4275–4286, 2025.

Shidong Yang, Ziyu Ma, Tongwen Huang, Yiming Hu, Yong Wang, and Xiangxiang Chu. CoEvolve: Training LLM agents via agent-data mutual evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23015–23036. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1055.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 535–546, 2021.

Appendix

Table of Contents

A Experiment Setting Details **14**

B Oracle Rule Distribution Analysis **15**

C Case Study **16**

D Different Training Strategies for Classification Router Agents **17**

E Extended Ablation Study **18**

E.1 Fixing the Retriever, Varying the Router. **18**

E.2 Fixing the Router, Varying the Retriever. **18**

F Inference Time Efficiency Comparison **19**

G Generation-Based Router Prompt **20**

G.1 Generation Routing Agent Prompts. **20**

G.2 Dataset-Specific Candidate Rules **20**

A EXPERIMENT SETTING DETAILS

We provide additional implementation details for the rule-constraining retrievers and the three routing variants used in CoRE: BERT-C, LLM-C, and LLM-G. Unless otherwise specified, all variants use the same rule sets, retriever architecture, candidate construction, and evaluation protocol.

Rule Space and Candidate Retrieval. We construct rule sets from the schema of each STaRK-QA dataset, resulting in 6 rules for AMAZON, 8 for MAG, and 65 for PRIME. For each query–rule pair, graph traversal under the corresponding rule produces a candidate pool, from which up to 200 candidate entities are retained for ranking. Each rule is represented by a 384-dimensional embedding. At inference time, the router selects one rule, and only the corresponding rule-constraining retriever is used to rank candidates within the resulting structural search space.

Rule-constraining Retrievers. Each rule is associated with a lightweight retriever that scores candidates using the path-level relevance features introduced in Section 4. The candidate feature vector and rule embedding are independently projected into 256-dimensional hidden representations and jointly mapped to a scalar relevance score. Retriever training uses pairwise ranking with one positive candidate and up to 200 negative candidates per query. We optimize the retrievers with AdamW using a learning rate of 5×10^{-5}, weight decay 0.01, and batch size 512. Each retriever is trained for at most 50 epochs, and the checkpoint with the best validation Hit@1 is retained.

The downstream supervision for router optimization is derived from rank of ground-truth evidence under each rule-constraining retriever. For query q and rule k, we define routing reward as reciprocal rank, $R(q, k) = 1/\operatorname{rank}(q, k)$, such that higher reward corresponds to better retrieval under selected rules. For queries with multiple correct entities, we use the best rank among correct answers.

BERT-C Router. The classification-based BERT router uses `bert-base`. Queries are tokenized to a maximum length of 128 tokens, and the final `[CLS]` representation is passed to a dataset-specific linear classification head over the candidate rules. Both the encoder and classification head are fine-tuned jointly. The router is trained with AdamW using batch size 256, learning rate 1×10^{-5}, and weight decay 0.01. For each query, the training target is the rule with the highest downstream retrieval reward, i.e., $y_q = \arg\max_k \bar{R}(q, k)$.

LLM-C Router. The LLM classification router uses Qwen2.5-3B as the query encoder. Inputs are truncated to 256 tokens, and the final hidden state of the last non-padding token is passed to a linear rule-classification head. We adapt the LLM backbone using LoRA on the query, key, value, and output projection layers while training the classification head directly. We use a learning rate 1×10^{-5} for LoRA parameters, with LoRA rank 32 and scaling factor 64 for AMAZON and MAG; the selected PRIME configuration uses rank 16 and scaling factor 32. Training is performed in BF16.

Unlike BERT-C, which uses only the highest-reward rule as a hard target, LLM-C exploits the complete rule-wise reward vector. Let $\bar{R}_q$ denote the mean reward across candidate rules for query q. The router is optimized using

$$\mathcal{L}_{\text{LLM-C}} = -\frac{1}{B} \sum_q \sum_k \log p_\theta(k \mid q)\big(R(q, k) - \bar{R}_q\big) - \gamma H[p_\theta(\cdot \mid q)].$$

Thus, rules with above-average retrieval rewards are promoted, while below-average rules are suppressed. The entropy regularization term encourages exploration over candidate rules during training. At inference time, the rule with the highest predicted probability is selected.

LLM-G Router. The generation-based router uses `[final backbone]` and formulates rule selection as conditional generation. The input contains the task instruction, natural-language query, and textual descriptions of all candidate rules. The model generates a reasoning trace followed by a single rule identifier in the format `<think> reasoning </think><answer> rule </answer>`. The model is adapted with LoRA on the query, key, value, and output projections, using rank 32, scaling factor 64, and dropout 0.05.

The router is optimized with group-relative policy optimization (GRPO). For each query, we sample eight rule-selection trajectories and evaluate the selected rules using their corresponding frozen retrievers. The reward combines format correctness and downstream retrieval quality as $R(q, y) = 0.2R_{\text{format}}(y) + R(q, k(y))$, where $k(y)$ is the rule extracted from the generated response. Invalid rule identifiers receive no retrieval reward. We use BF16 training, a global prompt batch size of 32,

eight generations per prompt, and three training epochs. The KL coefficient is 0.01, and generation uses temperature 0.9 during training. At inference time, decoding is deterministic and the selected rule is extracted from the `<answer>` field.

For the verified PRIME configuration, the generation-based router is additionally warm-started with supervised fine-tuning on filtered teacher demonstrations before GRPO. Demonstrations are retained only when they satisfy the required output format and select a rule whose retriever ranks a correct entity near the top of the candidate list. This warm-up primarily teaches the required generation format and provides an initial routing policy before optimization with downstream retrieval feedback.

Bi-Level Training. All CoRE variants are optimized for three bi-level rounds. Within each round, the routing policy is fixed while the rule-constraining retrievers are updated using the current query–rule assignments. The retrievers are then frozen, and their retrieval ranks provide the feedback used to update the routing policy. The updated router subsequently reassigns queries to rules for the next round. Both router and retriever parameters are warm-started from the preceding round rather than reinitialized, allowing routing and retrieval to progressively co-adapt.

Evaluation. For each query, the learned router selects a single reasoning rule, after which the corresponding rule-constraining retriever ranks the candidates reachable under that rule. We report Hit@1, Hit@5, Recall@20, and mean reciprocal rank (MRR). Classification-based routers select the highest-probability rule, while the generation-based router uses the rule identifier produced by deterministic greedy decoding.

Computational Resources. Experiments were conducted on a server equipped with four NVIDIA H200 NVL GPUs. Generative SFT and GRPO training use all four GPUs with distributed data parallelism, whereas rule-constraining retriever training and the discriminative routing models use a single GPU per run.

B Oracle Rule Distribution Analysis

We analyze how frequently each retrieval rule is optimal under the trained round-1 rule-constraining retrievers. The MAG, Amazon, and PRIME datasets contain 8, 6, and 65 candidate rules, respectively.

For each query q and rule r, we compute the mean reciprocal rank over its ground-truth answers G_q as $S(q,r) = \frac{1}{|G_q|} \sum_{a \in G_q} 1/\operatorname{rank}_{q,r}(a)$, where an answer absent from the top-200 candidates contributes zero. The oracle set $O_q = \{r : S(q,r) = \max_{r'} S(q,r')\}$ contains all rules achieving the best retrieval score for q. A uniquely optimal rule receives full credit, while each rule in a k-way tie receives $1/k$ credit. We denote the resulting unique and tied frequencies by U_r and T_r, respectively, and define the total fractional oracle frequency as $F_r = U_r + T_r$. Thus, each query contributes total credit one and $\sum_r F_r = 1$ within each dataset.

Figure 6 reveals three characteristics of the rule space. **First, oracle utility is highly query-dependent rather than concentrated on a single universally effective rule**. MAG distributes most of its oracle credit among several citation-, authorship-, and institution-based paths, while Amazon exhibits an even broader balance across its smaller rule set. PRIME, despite containing substantially more candidate rules, concentrates useful retrieval behavior in only a subset of them, leaving many rules rarely selected by the oracle. **Second, a substantial portion of oracle credit arises from ties**, indicating overlap between the retrieval behaviors captured by different rules. This is particularly visible for several MAG and Amazon rules whose oracle contribution is dominated by tied cases. In PRIME, some rules receive oracle credit exclusively through ties: they match the best retrieval quality for certain queries but never provide a uniquely better result. Under the current experts and reward, such rules contribute little distinctive retrieval capability and may therefore be redundant with other rules. **Finally, the combination of highly uneven usage and strong oracle overlap suggests that the predefined rule space may be over-complete**. Rules with consistent unique oracle credit capture specialized retrieval patterns, whereas rules that are rarely optimal or only tied-optimal are natural candidates for consolidation or pruning. This analysis therefore not only motivates query-specific routing, but also suggests a direction for automatic rule-space refinement, where oracle utility and uniqueness can guide the retention, pruning, or merging of candidate rules.

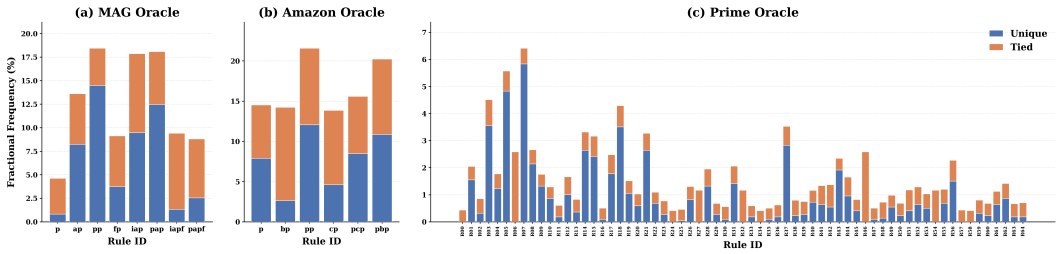

Figure 6: Rule-wise fractional oracle frequency across MAG, Amazon, and PRIME. Each stacked bar decomposes a rule's oracle credit into uniquely optimal (**Unique**) and tied-optimal (**Tied**) cases. For a k-way tie, each optimal rule receives $1/k$ credit, so the frequencies sum to 100% within each dataset.

C Case Study

A central motivation of CoRE is that the reasoning rule for a query is *latent* and cannot be reliably inferred from query semantics alone. When explicit query–rule supervision is unavailable, a natural alternative is to induce rule labels with a strong LLM and train the router on these pseudo-labels. However, such labels are determined only from the query and rule descriptions. The LLM does not observe how each rule-constraining retriever is instantiated, which evidence it can reach, or how well it ranks the ground-truth evidence. As a result, a rule that is semantically plausible may not be the rule that works best for downstream retrieval.

Figure 7 examines this gap on MAG using GPT-5 as a strong semantic router. Figure 7(a) compares GPT-5-selected rules with the *downstream oracle rule*, defined as the rule whose trained retriever achieves the highest reward for each query. The off-diagonal mass reveals a clear mismatch between semantic routing and retrieval utility. We identify two representative patterns: **semantic–retriever misalignment**, where the LLM's interpretation is reasonable but the corresponding retriever does not realize it effectively, and **semantic mis-grounding**, where the router itself maps the query to an inappropriate structural constraint. The former is more common; for example, the `iapf`→`iap` mismatch alone appears in 190 queries.

Semantic–retriever misalignment. Figure 7(b) shows a query containing both an institution and a topical constraint. GPT-5 therefore selects the more specific `iapf` rule (*institution* → *author* → *paper* ← *field of study*), which is semantically reasonable. However, the field branch retrieves field nodes whose paper sets do not overlap with the institution-derived candidates containing the ground-truth papers. Consequently, no valid `iapf` path reaches the answers. In contrast, the broader `iap` rule (*institution* → *author* → *paper*) successfully retrieves the two ground-truth papers at ranks 2 and 4. The key issue is therefore not incorrect semantic interpretation, but a mismatch between semantic specificity and actual retriever capability: adding a seemingly relevant constraint can reduce coverage when that constraint cannot be grounded reliably.

Semantic mis-grounding. Figure 7(c) shows the complementary case where the routing decision itself is incorrect. The query explicitly mentions "Seiberg–Witten theory," a valid field-of-study anchor, yet GPT-5 selects direct paper retrieval `p` and treats the term as ordinary text. Without the field-level structural signal, the ground-truth paper is buried among many topically similar papers and ranks 110th. The `fp` rule instead grounds the named theory to the corresponding field node and raises the same paper to rank 7. This illustrates a genuine grounding error: the router fails to exploit a structural constraint that is both semantically justified and useful for retrieval.

These cases highlight why static LLM-induced query–rule labels are insufficient for learning a retrieval-oriented router. A good router must capture not only which rule *sounds* appropriate, but which rule works well with the available retrievers and knowledge graph. CoRE addresses this by making routing and retrieval mutually dependent: retriever performance provides the reward for optimizing the routing policy, while updated routing decisions determine the supervision used to specialize the retrievers. The resulting bi-level optimization directly aligns upstream rule selection with downstream retrieval effectiveness.

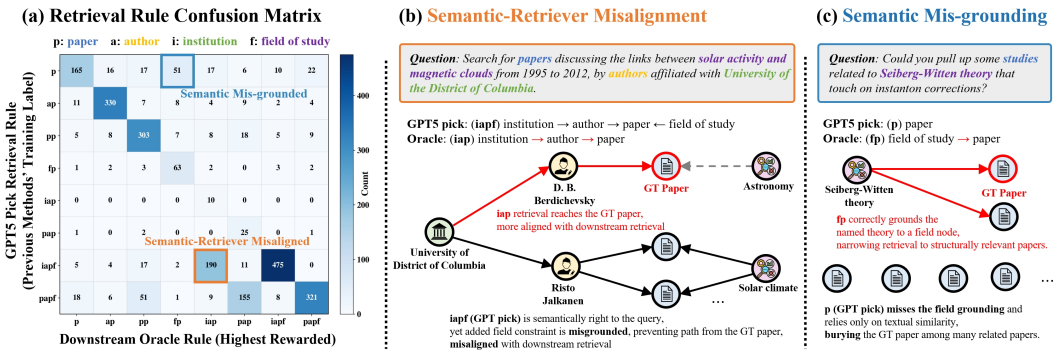

Figure 7: Case study of GPT-5 retrieval-rule routing failures on MAG. (a) **Confusion matrix** between GPT-5-selected rules and highest-reward oracle rules. (b) **Semantic–retriever misalignment**: GPT-5 selects the semantically plausible `iapf` rule, but the added field constraint is misgrounded by the retriever and prevents access to the ground-truth paper, while the broader `iap` path succeeds. (c) **Semantic mis-grounding**: GPT-5 treats the explicit field "Seiberg–Witten theory" as free-text content and selects `p`, whereas `fp` correctly grounds the field and narrows retrieval to structurally relevant papers.

D Different Training Strategies for Classification Router Agents

Motivated by the preceding case study, a natural question is whether the router can be trained more simply by treating the highest-reward rule as a pseudo-label. Since classification-based routing allows the retrieval reward of every candidate rule to be evaluated for each query, these rewards can either be converted into supervised targets or optimized directly. We therefore compare three training strategies while keeping the router architecture, retrievers, and co-adaptation procedure unchanged. Let $r_m = 1/\mathrm{rank}(S^* \mid Q, K_m)$ denote the reciprocal ground-truth rank under rule K_m, and $p_m = P_\theta(K_m \mid Q)$ the router probability. **(1) Hard-label SFT.** We select the highest-reward rule $m^* = \arg\max_m r_m$ and optimize $\mathcal{L}_{\mathrm{hard}} = -\log p_{m^*}$. This retains only the current best rule and discards reward differences among the remaining rules. **(2) Soft-label SFT.** We convert the reciprocal-rank rewards into a soft target distribution $q_m = \exp(r_m)/\sum_j \exp(r_j)$ and optimize $\mathcal{L}_{\mathrm{soft}} = -\sum_m q_m \log p_m$, preserving relative rule-level retrieval utility. **(3) RLFT.** Instead of converting rewards into fixed targets, RLFT directly maximizes expected retrieval utility, using advantages to reinforce above-average rules and suppress lower-reward ones.

As shown in Figure 8, performance improves consistently from hard-label SFT to soft-label SFT and further to RLFT across all retrieval metrics. This trend indicates that preserving richer rule-level reward information is more effective than collapsing it into a single pseudo-label, while directly optimizing retrieval utility provides an additional benefit over fitting a reward-derived target distribution.

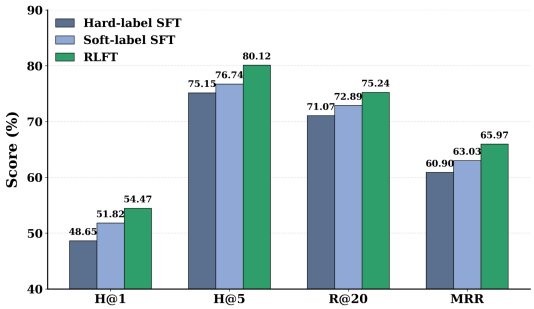

Figure 8: Comparison of training strategies for the BERT-C router, averaged over AMAZON, MAG, and PRIME. RLFT consistently outperforms hard-label and soft-label SFT across all retrieval metrics.

The difference is also intuitive from their optimization signals. Hard-label SFT only distinguishes the highest-reward rule from all others, regardless of whether its advantage is marginal or substantial. In contrast, RLFT uses the relative retrieval advantage of each rule, producing small updates when rules perform similarly and stronger updates when their retrieval utilities differ substantially. It therefore preserves richer information from the reward landscape instead of reducing it to a binary winner-versus-non-winner signal.

This distinction is particularly important under co-adaptation, where router assignments determine the training data for each rule-constraining retriever. Overly confident early routing can prematurely concentrate supervision on a few rules, limiting the improvement of alternatives as retriever quality adapts. RLFT mitigates this by keeping preferences weak when reward differences are small and strengthening them only as retrieval advantages become clearer.

E Extended Ablation Study

The main ablation in Table 2 shows that removing either router adaptation or retriever adaptation degrades performance. However, this result alone does not fully distinguish whether the gain comes from *co-adaptation* or simply from using stronger individual components. We therefore conduct two complementary studies that fix one component while varying the other. Together, they separate the effects of routing quality, retrieval quality, and their interaction. Unless otherwise noted, results are averaged over the three STaRK-QA datasets (AMAZON, MAG, and PRIME).

E.1 Fixing the Retriever, Varying the Router

We first isolate the routing effect by fixing the rule-constraining retriever to the one trained under CoRE-LLM-G and replacing only router with three zero-shot general-purpose LLMs: Qwen3B, GPT-5-mini, and GPT-5. We compare these decoupled router substitutions against CoRE-LLM-G, whose router is optimized directly from retrieval feedback.

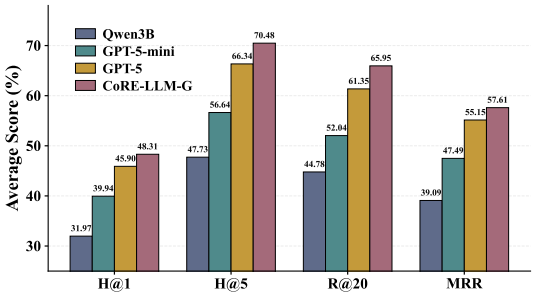

Figure 9: Performance with a fixed trained retriever and different routers, averaged over AMAZON, MAG, and PRIME. Stronger zero-shot LLM routers improve retrieval, but CoRE-LLM-G remains best across all metrics, showing the benefit of retrieval-feedback-based router adaptation.

Figure 9 reveals a clear trend. Stronger general-purpose LLMs provide progressively better rule routing, indicating that general reasoning capability transfers to structured rule selection to some extent. However, the improvement saturates before reaching CoRE-LLM-G. Even GPT-5, despite being substantially stronger than the backbone used by CoRE-LLM-G, remains below the co-adaptively trained router on every retrieval metric. This gap highlights an important distinction between *semantic rule plausibility* and *retrieval-effective rule selection*. A general-purpose LLM can infer which rule appears appropriate from the query semantics, but it does not observe how well each rule is instantiated by the downstream retriever or how effectively that retriever ranks the ground-truth evidence. In contrast, CoRE directly optimizes the routing policy using retrieval outcomes from the paired retriever. As a result, the learned router is calibrated not only to the query, but also to the empirical utility of each rule under the current retrieval system.

Thus, the gain from CoRE cannot be explained by simply replacing the router with a more capable off-the-shelf model. What matters is aligning routing decisions with downstream retrieval behavior through the closed feedback loop.

E.2 Fixing the Router, Varying the Retriever

We next study the complementary effect of retrieval quality. For each router policy (Qwen3B, GPT-5-mini, and GPT-5), we compare three retrieval strategies under the selected rule constraint: *BM25-last*, which scores candidates using only the terminal entity on the reasoning path; *BM25-avg*, which aggregates BM25 relevance across the full path; and *Trained*, the learnable rule-conditioned retriever used in CoRE. We additionally include plain BM25 over the full knowledge base as an unconstrained reference. Figure 10 shows two consistent trends. First, introducing a rule constraint produces the largest qualitative shift in performance. Even simple BM25-based scoring under a routed rule improves over unconstrained BM25, indicating that restricting retrieval to a structurally compatible candidate space is already highly beneficial. This supports the central premise of CoRE: identifying an appropriate reasoning rule is not merely an auxiliary classification problem, but directly determines which evidence can be reached.

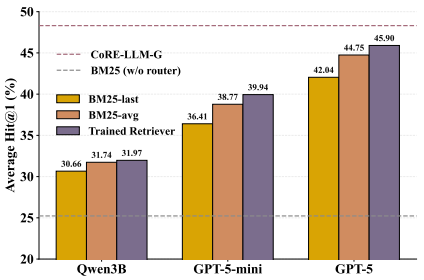

Figure 10: Average Hit@1 across datasets under different router–retriever combinations. Rule-constraining retrieval consistently improves over unconstrained BM25, while stronger retrievers provide additional gains within the selected rule space. CoRE-LLM-G remains best overall.

Second, once the rule is fixed, retrieval quality still matters. For every router, performance consistently improves from BM25-last to BM25-avg and further to the trained retriever. This suggests that structural restriction and relevance ranking play complementary roles. The router narrows the search space according to the inferred reasoning pattern, while the retriever learns how to rank candidates within that space by exploiting relevance signals along the traversal path.

The gains from retriever improvement are generally smaller than the gains obtained from imposing the rule constraint itself, indicating that routing is the dominant source of improvement over fully unconstrained retrieval. Nevertheless, the consistent benefit of the trained retriever shows that rule selection alone is insufficient: after the correct structural region is identified, the model still benefits from learning how different path-level signals should be combined.

Importantly, the two axes remain complementary rather than substitutable. A stronger router continues to benefit from a stronger retriever, and a stronger retriever continues to benefit from better routing. Yet even the strongest decoupled combination does not match CoRE-LLM-G. This remaining gap is precisely what the co-adaptive objective is designed to address.

Taken together, the two studies clarify the source of CoRE's improvement. Rule routing provides the primary structural advantage by determining the reachable evidence space; retriever adaptation refines ranking within that space; and co-adaptive optimization aligns the two so that routing decisions and retriever specialization reinforce each other. The benefit therefore does not arise from simply scaling either component in isolation.

F Inference Time Efficiency Comparison

We further examine the inference efficiency of CoRE against the agentic graph reasoning method GraphDancer on MAG. We run each method on 1,000 questions and report the mean inference time per question, decomposed into *routing / LLM generation* and *retrieval / tool execution* costs.

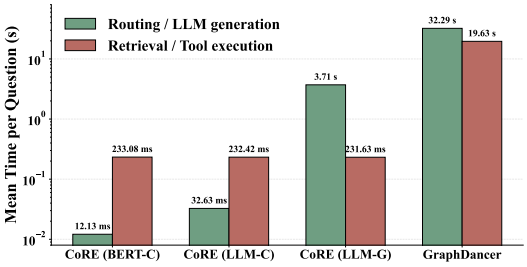

Figure 11: Efficiency comparison on MAG in terms of mean inference time per question. We decompose the cost into routing/LLM generation and retrieval/tool execution. CoRE substantially reduces inference cost compared with step-wise agentic graph exploration, particularly when using classifier-based routing. The y-axis is shown on a logarithmic scale.

As shown in Figure 11, a clear efficiency gap emerges between CoRE and step-wise agentic exploration. The classifier-based variants, CoRE-BERT-C and CoRE-LLM-C, introduce only negligible routing overhead, leaving rule-constraining retrieval as the dominant inference cost. Notably, their retrieval costs are nearly identical, indicating that changing the routing architecture has little effect on the downstream retrieval procedure once a rule has been selected.

CoRE-LLM-G exhibits a different cost profile: its retrieval cost remains comparable to the classifier-based variants, while its routing cost increases substantially due to autoregressive rule generation. This suggests that the additional computational cost of generative routing is concentrated in the routing stage rather than in structured retrieval itself. Nevertheless, its overall inference cost remains substantially below that of GraphDancer.

In contrast, GraphDancer incurs high costs in both components. Its LLM generation cost reflects repeated reasoning over intermediate graph observations, while its tool-execution cost accumulates across multiple exploration steps. The resulting gap highlights a fundamental architectural difference: CoRE performs a single high-level routing decision and delegates subsequent structured search to a specialized rule-constraining retriever, whereas agentic exploration repeatedly interleaves reasoning and graph operations throughout the inference trajectory.

Overall, the results show that rule routing provides a computationally efficient alternative to iterative graph exploration. In particular, classifier-based CoRE achieves strong structured retrieval with minimal decision overhead, while generation-based routing trades additional inference cost for its ability to operate over transferable, tokenized rule representations.

Table 3: Statistics of the STaRK datasets, and generation-based routing configuration.

	STARK-AMAZON	STARK-MAG	STARK-PRIME
#entity types	4	4	10
#relation types	5	4	18
#entities	1,035,542	1,872,968	129,375
#relations	9,443,802	39,802,116	8,100,498
#tokens	592,067,882	212,602,571	31,844,769
#queries	9,181	13,407	11,302
train	5,915	7,994	6,162
val	1,547	2,665	2,241
test	1,719	2,748	2,899
Rule Keys	`A–F`	`A–H`	`R00–R64`
# Rules	6	8	65

G GENERATION-BASED ROUTER PROMPT

G.1 GENERATION ROUTING AGENT PROMPTS

The generation-based router selects one rule from a dataset-specific candidate set. Each input consists of a natural-language query and a set of reasoning rules represented as schema-level traversal paths, such as `Author` → `Paper` and `Gene/Protein` → `Target` → `Drug`. We use the following prompt template across all datasets.

Generation-Based Routing Prompt

System Message

You are a `{DOMAIN}` knowledge graph retrieval rule selector. Given a query and a set of candidate rules, select the traversal path that best matches the query.
Consider the target entity, anchor entity, relation semantics, and traversal direction. Select the simplest rule that fully captures the query.

Output Format

```
<think>[brief reasoning]</think>
<answer>[rule key]</answer>
```

User Message

```
Query: {QUERY}

Available retrieval rules:
{KEY_1}: {RULE_1}
{KEY_2}: {RULE_2}
...
{KEY_N}: {RULE_N}
```

Select the single best retrieval rule.

Here, `{DOMAIN}` denotes the dataset domain, `{QUERY}` denotes the input query, and each `{KEY}`: `{RULE}` pair specifies a valid output key and its corresponding traversal path. We extract the prediction from the `<answer>` field and regard it as valid only if it matches one of the provided rule keys. Table 3 summarizes the dataset-specific configurations.

G.2 DATASET-SPECIFIC CANDIDATE RULES

The candidate rule set for each dataset is listed below. Each rule defines the schema-level traversal used to constrain retrieval. PRIME contains a substantially larger candidate set. We therefore use a multi-page table to improve readability. Long traversal paths are automatically wrapped, preferentially around relation boundaries.

Table 4: Candidate retrieval rules for Amazon.

Key	Traversal Path
A	Product
B	Brand → Product
C	Product → Product
D	Category → Product
E	Product → Category → Product
F	Product → Brand → Product

Table 5: Candidate retrieval rules for MAG.

Key	Traversal Path
A	Paper
B	Author → Paper
C	Paper → Paper
D	Field of Study → Paper
E	Institution → Author → Paper
F	Paper → Author → Paper
G	Institution → Author → Paper ← Field of Study
H	Paper → Author → Paper ← Field of Study

Table 6: Candidate retrieval rules for PRIME.

Key	Traversal Path
R00	Anatomy → Expression Present → Gene/Protein
R01	Anatomy → Expression Absent → Gene/Protein
R02	Pathway → Interacts With → Gene/Protein → Expression Absent → Anatomy
R03	Gene/Protein → Target → Drug
R04	Disease → Associated With → Gene/Protein
R05	Gene/Protein → PPI → Gene/Protein
R06	Gene/Protein → Associated With → Disease
R07	Disease → Parent-Child → Disease
R08	Effect/Phenotype → Phenotype Present → Disease
R09	Effect/Phenotype → Phenotype Absent → Disease
R10	Disease → Contraindication → Drug
R11	Gene/Protein → Expression Absent → Anatomy
R12	Drug → Contraindication → Disease

Continued on next page

Table 6 continued

Key	Traversal Path
R13	Biological Process → Parent-Child → Biological Process → Interacts With → Gene/Protein → Expression Absent → Anatomy
R14	Drug → Synergistic Interaction → Drug
R15	Pathway → Parent-Child → Pathway
R16	Drug → Off-Label Use → Disease
R17	Molecular Function → Interacts With → Gene/Protein
R18	Pathway → Interacts With → Gene/Protein
R19	Cellular Component → Interacts With → Gene/Protein
R20	Effect/Phenotype → Associated With → Gene/Protein → Expression Absent → Anatomy
R21	Disease → Indication → Drug
R22	Effect/Phenotype → Side Effect → Drug
R23	Exposure → Linked To → Disease
R24	Effect/Phenotype → Associated With → Gene/Protein
R25	Gene/Protein → Expression Present → Anatomy
R26	Drug → Indication → Disease
R27	Gene/Protein → Interacts With → Pathway
R28	Biological Process → Interacts With → Gene/Protein → Expression Absent → Anatomy
R29	Drug → Synergistic Interaction → Drug → Transporter → Gene/Protein
R30	Gene/Protein → Transporter → Drug
R31	Biological Process → Interacts With → Gene/Protein
R32	Gene/Protein → Interacts With → Cellular Component
R33	Gene/Protein → Enzyme → Drug
R34	Drug → Enzyme → Gene/Protein
R35	Disease → Off-Label Use → Drug
R36	Drug → Transporter → Gene/Protein → Interacts With → Exposure
R37	Drug → Target → Gene/Protein → Interacts With → Cellular Component
R38	Disease → Associated With → Gene/Protein → Expression Absent → Anatomy
R39	Cellular Component → Interacts With → Gene/Protein → Expression Absent → Anatomy
R40	Disease → Phenotype Present → Effect/Phenotype
R41	Disease → Associated With → Gene/Protein → Target → Drug
R42	Pathway → Interacts With → Gene/Protein → Interacts With → Biological Process
R43	Disease → Parent-Child → Disease → Phenotype Present → Effect/Phenotype

Continued on next page

Table 6 continued

Key	Traversal Path
R44	Pathway → Interacts With → Gene/Protein → Target → Drug
R45	Drug → Transporter → Gene/Protein → Expression Present → Anatomy
R46	Gene/Protein → Associated With → Effect/Phenotype
R47	Gene/Protein → Carrier → Drug
R48	Gene/Protein → PPI → Gene/Protein → Interacts With → Pathway
R49	Anatomy → Expression Present → Gene/Protein → Associated With → Disease
R50	Cellular Component → Interacts With → Gene/Protein → Target → Drug
R51	Biological Process → Interacts With → Gene/Protein → Interacts With → Pathway
R52	Exposure → Interacts With → Gene/Protein
R53	Drug → Target → Gene/Protein
R54	Gene/Protein → Interacts With → Biological Process
R55	Drug → Synergistic Interaction → Drug → Target → Gene/Protein
R56	Molecular Function → Interacts With → Gene/Protein → Target → Drug
R57	Biological Process → Parent-Child → Biological Process
R58	Gene/Protein → Transporter → Drug → Side Effect → Effect/Phenotype
R59	Effect/Phenotype → Phenotype Present → Disease → Associated With → Gene/Protein
R60	Molecular Function → Interacts With → Gene/Protein → Expression Absent → Anatomy
R61	Anatomy → Expression Present → Gene/Protein → Enzyme → Drug
R62	Effect/Phenotype → Phenotype Present → Disease → Indication → Drug
R63	Disease → Associated With → Gene/Protein → Interacts With → Pathway
R64	Cellular Component → Interacts With → Gene/Protein → Interacts With → Pathway