DSL-R1: From SQL to DSL for Training Multimodal Retrieval Agents across Structured and Unstructured Data with Reinforcement Learnings

Anonymous ACL submission

Abstract

Multimodal retrieval requires integrating both logical reasoning and semantic matching across diverse modalities such as text, image, and video. Traditional approaches rely on either symbolic query languages or vector-based similarity search, yet effective unification of the two remains rare. In this work, we propose a domain-specific language (DSL) that combines SQL-style logical operators with vector retrieval, enabling complementary strengths of symbolic precision and semantic coverage. We further introduce a reinforcement learning (RL) framework in which rule-based verification and retrieval feedback jointly optimize the DSL generation process, balancing execution correctness and semantic alignment. Evaluations on a large-scale business email corpus demonstrate that our DSL-RL framework achieves an average improvement of +12.3% on Hit@1/3, consistently outperforming both symbolic and vector-only baselines. DSL-R1 thus provides a simple, extensible, and reinforcement-aligned paradigm for multimodal retrieval, with potential for broader interactive AI applications.

1 Introduction

The explosive growth of multimodal data presents pressing challenges for retrieval systems. Symbolic query languages such as SQL (Zhong et al., 2017) or SPARQL (Berant et al., 2013) provide strong interpretability and precise logical control but struggle with unstructured modalities such as free-form text or document content. Conversely, vector-based models (e.g., CLIP (Radford et al., 2021), BLIP-2 (Li et al., 2023)) excel at semantic matching but lack the ability to enforce logical constraints and offer explainability. Bridging these two paradigms is critical for building retrieval systems that are both robust and interpretable.

To this end, we propose a DSL-driven retrieval framework that integrates SQL operators with vector search under a unified syntax. This design enables compositional queries that capture both logical constraints and semantic intents within a single, language-like interface. A key challenge lies in generating and executing DSL queries reliably under ambiguous or noisy user inputs. We address this by introducing a reinforcement learning framework that jointly optimizes symbolic correctness and retrieval utility through rule-based reward functions, enhancing both execution robustness and semantic coverage.

We evaluate our framework on a large-scale multimodal retrieval dataset constructed from business emails, which combines structured metadata (e.g., sender, recipient, timestamp) with unstructured text and attachments. Experiments using GRPO (Sun et al., 2024) and DAPO (Wang et al., 2025) optimization algorithms show substantial accuracy gains over symbolic-only and vector-only baselines, with DAPO offering the best trade-off between stability and efficiency.

Our main contributions are summarized as follows: **A multimodal retrieval dataset for business emails:** a large-scale benchmark integrating structured metadata with unstructured textual and visual content. **A DSL-agent reinforcement learning framework:** a unified DSL combining SQL-style logical operators with vector-based retrieval, optimized through rule-driven RL signals for correctness and semantic quality.

2 Related Work

2.1 Multimodal Retrieval

Vector-based multimodal retrieval approaches such as CLIP (Radford et al., 2021), ALIGN (Jia et al., 2021), BLIP-2 (Li et al., 2023), and Florence-2 (Yuan et al., 2024) have demonstrated strong performance in capturing semantic similarity across text–image and text–video modalities. In contrast, symbolic retrieval methods like SQL (Zhong et al., 2017) and SPARQL (Berant et al., 2013) offer struc-

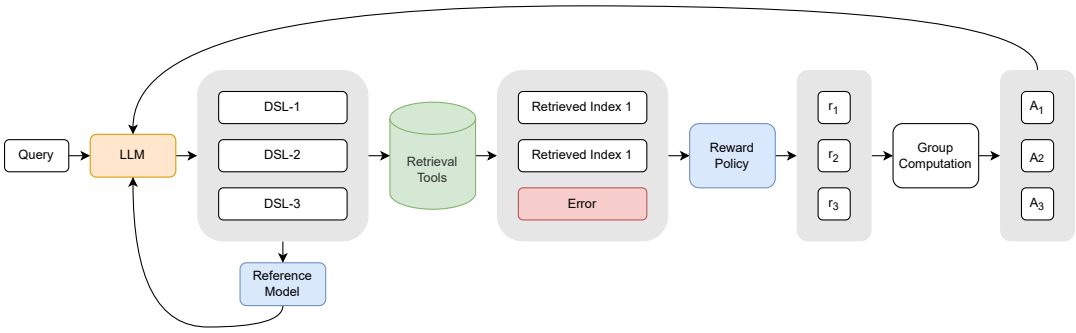

Figure 1: Overview of the reinforcement learning framework for DSL-based multimodal retrieval.

tural precision and interpretability but are less effective for unstructured or ambiguous inputs. Recent hybrid methods, including ColBERTv2 (Santhanam et al., 2022), ColPali (Kenton et al., 2024), and ColQwen (Chen et al., 2024), enhance dense retrieval with contextualized representations, while fusion-based systems such as FiD (Izacard and Grave, 2021), RePlug (Shi et al., 2023), and InstructBLIP (Dai et al., 2023) explore multimodal reasoning by integrating retrieval and generation. However, most existing works operate at the representation level, lacking an explicit symbolic layer to express compositional and logical dependencies. Our work differs by introducing a trainable domain-specific language (DSL) that unifies symbolic operators and vector retrieval within a single reasoning and optimization framework.

2.2 DSLs in AI Systems

Domain-specific languages (DSLs) have been widely applied in program synthesis (Balog et al., 2017; Nye et al., 2021; Chen et al., 2021), data analysis (Wang et al., 2023; Zhang et al., 2024), and reasoning with language models (Jiang et al., 2023; Wei et al., 2022; Gupta et al., 2023). DSLs enable structured expressivity and verifiable reasoning but traditionally lack adaptability without external optimization or learning signals. Recent advances have explored combining DSLs with reinforcement learning or large language models to improve control and generalization (Xu et al., 2024; Yao et al., 2023; Gou et al., 2024). Our framework continues this direction by treating the DSL not as a static interface but as a learnable policy space, jointly optimized through rule-based rewards and retrieval feedback to achieve semantic alignment and robust execution in multimodal retrieval contexts.

3 Method

3.1 Overview

Our framework consists of three main *components*: (i) a *DSL-based multimodal retrieval agent*, which translates natural-language inputs into executable programs that combine structured operators with vector-based similarity search; (ii) a *data generation component*, which constructs multimodal retrieval corpora containing both structured and unstructured elements, and produces paired queries, programs, and gold targets to support supervised and evaluative training; and (iii) a *reinforcement learning (RL) component*, which optimizes the agent policy by leveraging execution-derived rewards and preference-based regularization. Together, these components ensure both interpretability and adaptability, enabling effective retrieval across heterogeneous multimodal data sources.

3.2 DSL Agent Design

The core of our system is a *DSL-based multimodal retrieval agent* that integrates symbolic and vector-based operators within a unified execution framework. Specifically, the DSL program is composed of two parts: (i) an SQL-like clause that handles structured conditions over metadata, and (ii) a *vector search query list* that retrieves candidate items from unstructured modalities.

Given a natural-language query x, the agent first generates a DSL program y containing both structured filters and vector queries. The vector search module computes nearest neighbors in the embedding space and returns a set of candidate primary keys $\{id_1, \ldots, id_k\}$. These candidate identifiers are then injected into the SQL execution engine as constraints, where structured predicates (e.g., date, category, author) are applied. The final result set is obtained by merging the outcomes of the SQL

filter with the candidate list returned by the vector search.

This design allows the agent to exploit the *precision of symbolic filtering* while simultaneously leveraging the *semantic coverage of vector similarity*, ensuring both interpretability and robustness in multimodal retrieval tasks.

3.3 Data Preparation

Our corpus $\mathcal{D}$ consists of email records $d = (m, u)$, where m denotes structured metadata (sender, date, category) and u represents unstructured content (subject, body, attachments). For each record, we randomly select 1–3 metadata attributes to form a structured filter F_s, retrieving a candidate set $\mathcal{C}_s = \{d' \in \mathcal{D} \mid m'(a_i) = m(a_i)\}$. A gold record $\hat{d}$ is then sampled from $\mathcal{C}_s$, and several semantic cues from its content u are used to construct a query embedding q. A vector search refines the candidate set to $\mathcal{C}_{su} = \{d' \in \mathcal{C}_s \mid \cos(q, u') \geq \tau\}$, where τ is a similarity threshold. The structured filter F_s and semantic query q are combined into an executable DSL program y that retrieves $\hat{d}$, while a corresponding natural-language query x is generated from the same attributes, forming a supervised triplet $(x, y, \hat{d})$ that couples symbolic constraints with semantic intent and provides verifiable supervision.

3.4 Reinforcement Learning

As shown in Figure 1, given a natural language query x, the policy model generates multiple DSL candidates $\{y_i\}_{i=1}^{G}$. Each candidate is executed with retrieval tools; valid results are scored by the reward function, and errors receive zero or negative scores. The group of rewards $\{r_i\}$ is then aggregated to compute group-relative advantages $\{A_i\}$, which guide policy updates.

We adopt two algorithms: GRPO (Sun et al., 2024) and its improved variant DAPO (Wang et al., 2025). GRPO enables efficient policy optimization without a value model, while DAPO further stabilizes training through dual-anchor regularization and adaptive reward scaling.

The simplified GRPO objective is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} \min\big(r_i A_i,\ \text{clip}(r_i) A_i\big) + \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \quad (1)$$

where $r_i = \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)}$ is the importance ratio and A_i the group-relative advantage.

DAPO further improves GRPO by normalizing advantages at the token level and using asymmetric clipping bounds. Its simplified loss is:

$$\mathcal{L}_{\text{DAPO}}(\theta) = -2\frac{1}{\sum_i |y_i|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i|} \min\big(r_{i,t}A_{i,t}, \text{clip}(r_{i,t}, 1-\epsilon_l, 1+\epsilon_h)\, A_{i,t}\big), \quad (2)$$

where $N = \sum_i |y_i|$ is the total number of tokens, $r_{i,t} = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|x, y_{i,<t})}$ is the token-level importance ratio, $A_{i,t}$ is the token-level advantage, and (ϵ_l, ϵ_h) are asymmetric clipping bounds that allow different tolerances for underestimation and overestimation.

These two strategies allow stable and efficient optimization of the DSL agent, with DAPO showing better convergence in practice.

3.4.1 Reward Function Design

To optimize the DSL-based retrieval agent, we design a composite reward function that balances structural correctness, execution reliability, semantic accuracy, and response efficiency. The total reward is formulated as

$$R = S_f + S_e + S_r + S_l,$$

where each component provides distinct feedback to guide policy optimization. **Format Reward** (S_f) encourages syntactic compliance by rewarding outputs that correctly encapsulate reasoning and retrieval logic within predefined DSL tags (e.g., `<query>...</query>`). **Execution Reward** (S_e) measures the executability of the generated DSL statement, assigning higher rewards to programs that can be successfully parsed and executed by the retrieval backend. **Result Reward** (S_r) evaluates the alignment between retrieved results and the reference set, computed as a function of retrieval precision and recall. **Length Reward** (S_l) penalizes overly long outputs to promote concise and latency-efficient responses, ensuring the agent maintains high utility under real-world inference.

As shown in Table 2, removing any single component causes a performance drop, confirming that each reward type contributes complementarily to the agent's learning dynamics.

4 Experiments

4.1 Setup

Datasets. We evaluate our model on large-scale text–image and text–video corpora with fine-grained alignment, focusing on the Email Retrieval

Table 1: **Main Results on the Email Retrieval Task.** Comparison across traditional retrieval methods, base LLMs, and our reinforcement learning–optimized DSL agents.

Category	Model	Hit@1	Hit@3	MRR	Latency (ms/query)
Traditional Retrieval	BM25	42.1	64.7	53.2	18
	Copali (ColQwen)	55.3	78.9	65.4	35
LLM Baselines (no RL)	Qwen3-4B (pretrained)	58.3	80.1	69.2	95
	Qwen3-8B (pretrained)	62.1	83.6	72.5	140
	GPT-4o (zero-shot API)	68.9	88.7	78.8	**30**
Ours (DSL-RL Agents)	Qwen3-4B + GRPO	69.5	88.9	78.6	55
	Qwen3-4B + DAPO	**71.7***	**90.6***	**80.9**	52
	Qwen3-8B + GRPO	75.8	92.9	84.6	82
	Qwen3-8B + DAPO	**77.9***	**94.2***	**86.1**	78

benchmark as a representative use case. Each query consists of a multimodal user request (text + attachment metadata) and multiple structured/unstructured candidates.

Baselines. We compare four groups: (1) *Traditional retrieval models* such as BM25 (**?**) and Copali/ColQwen (Kenton et al., 2024; Chen et al., 2024), (2) *LLM baselines* (Qwen3-4B, Qwen3-8B, GPT-4o (OpenAI, 2024)) without RL optimization, (3) *Hybrid models* that combine SQL-style symbolic filtering with vector retrieval (Izacard and Grave, 2021; Santhanam et al., 2022), and (4) our *DSL-RL agents* trained with GRPO (Sun et al., 2024) and DAPO (Wang et al., 2025).

Metrics. We report standard retrieval metrics—Hit@k, Mean Reciprocal Rank (MRR), and nDCG—alongside *Exec-Success*, which measures whether the generated DSL query can be executed without syntax or schema errors. Latency (ms/query) is also included to reflect runtime efficiency.

4.2 Main Results

Table 1 summarizes the overall results. Our DSL-RL agents outperform all baselines across all metrics, confirming the effectiveness of reinforcement learning in aligning the DSL generator with downstream execution quality. Compared with pretrained LLMs, RL optimization improves Hit@1 by 9.6–15.8 points and MRR by up to 13.6 points. Among the two optimization strategies, DAPO consistently yields higher accuracy and more stable convergence than GRPO on both the 4B and 8B backbones. In addition, reinforcement learning substantially reduces response latency by regularizing output length: *Qwen3-8B + DAPO* achieves 78 ms/query, a 45% reduction compared with the vanilla pretrained model.

Table 2: Ablation study of reward components on the Email dataset.

Reward Function	Accuracy (%)
Qwen3-8B (pretrained)	62.1
$S_f + S_e + S_r + S_l$	**77.9**
- w/o S_f (Format Reward)	75.3 ($\downarrow$2.6)
- w/o S_e (Execution Reward)	75.6 ($\downarrow$2.3)
- w/o S_r (Result Reward)	77.0 ($\downarrow$0.9)
- w/o S_l (Length Reward)	75.9 ($\downarrow$2.0)

4.3 Ablation Studies

Table 2 quantifies the contribution of each reward component. Removing any term reduces accuracy, confirming their complementary effects. Format and execution rewards (S_f, S_e) are essential for generating valid and executable DSLs, while the result reward (S_r) enhances semantic correctness and the length reward (S_l) controls verbosity for faster responses. These rewards yield a 15.8-point improvement over the pretrained baseline.

Qualitative ablations show consistent trends: removing the SQL filter lowers precision by losing attribute constraints, excluding the vector retrieval module hurts recall on paraphrased queries, and omitting the feedback signal slows convergence and weakens cross-domain generalization.

5 Conclusion

We introduced DSL-RL, unifying SQL logic and vector retrieval in a single DSL, trained with RL using LLM-simulated feedback. Across benchmarks, DSL-RL improves accuracy and robustness; DSPO provides the best stability. Future work will integrate human-in-the-loop feedback.

6 Limitations

While our DSL-RL framework demonstrates strong retrieval performance and generality, several limitations remain. First, the current system operates in a single-turn setting and does not yet support multi-turn query refinement or conversational retrieval, which would require temporal context modeling and dialogue-state tracking. Second, the framework currently adopts a single-agent reinforcement process; extending it to a multi-agent or collaborative setting, such as coordinated query generation, execution, and verification, could further enhance interpretability and robustness. Third, although our DSL syntax unifies symbolic and vector-based retrieval, the present experiments are limited to textual and structured inputs. Extending the framework to handle multimodal inputs including images, audio, and videos, as well as cross-modal reasoning, remains an open direction. Finally, our rule-based reward design simplifies supervision but may constrain adaptability when scaling to more complex and heterogeneous real-world datasets.

References

Matej Balog and 1 others. 2017. Deepcoder: Learning to write programs. In *ICLR*.

Jonathan Berant and 1 others. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Mark Chen and 1 others. 2021. Program of thoughts prompting: Debugging reasoning in language models step-by-step. In *NeurIPS*.

Zhenyu Chen and 1 others. 2024. Colqwen: Column-aware dense retrieval via multimodal embedding alignment. *arXiv preprint arXiv:2405.08212*.

Wenliang Dai and 1 others. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*.

Jianyu Gou and 1 others. 2024. Llmcompiler: Converting natural language to optimized code via reinforcement learning. *arXiv preprint arXiv:2402.09231*.

Ananya Gupta and 1 others. 2023. Programmatic reasoning with large language models. *arXiv preprint arXiv:2310.06136*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Chao Jia and 1 others. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*.

Albert Jiang and 1 others. 2023. Program-aided language models for reasoning and verification. In *ICLR*.

Jacob Kenton and 1 others. 2024. Colpali: Contextual late interaction for multimodal dense retrieval. *arXiv preprint arXiv:2403.12345*.

Junnan Li and 1 others. 2023. Blip-2: Bootstrapped language-image pre-training with frozen image encoders and large language models. In *ICML*.

Maxwell Nye and 1 others. 2021. Show your work: Scratchpads for intermediate computation with language models. In *ICLR*.

OpenAI. 2024. Gpt-4o: Multimodal large language model. *Technical Report*.

Alec Radford and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Keshav Santhanam, Omar Khattab, and 1 others. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *NAACL*.

Weizhe Shi and 1 others. 2023. Replug: Retrieval-augmented black-box language models. In *NeurIPS*.

Yi Sun and 1 others. 2024. Group relative policy optimization for large language model alignment. *arXiv preprint arXiv:2402.10821*.

Liang Wang and 1 others. 2023. Dsl-sql: A unified domain-specific language for compositional data analytics. In *SIGMOD*.

Liang Wang and 1 others. 2025. Dapo: Dual-anchor policy optimization for efficient llm fine-tuning. *arXiv preprint arXiv:2503.14476*.

Jason Wei and 1 others. 2022. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*.

Yutong Xu and 1 others. 2024. Rl-code: Reinforcement learning for code generation and optimization. *arXiv preprint arXiv:2403.08711*.

Shunyu Yao and 1 others. 2023. React: Synergizing reasoning and acting in language models. In *ICLR*.

Lu Yuan and 1 others. 2024. Florence-2: Advancing a unified representation for a variety of vision tasks. *arXiv preprint arXiv:2405.13216*.

Wei Zhang and 1 others. 2024. Datadsl: A domain-specific language for declarative data transformation. *arXiv preprint arXiv:2401.10212*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. In *ACL*.