Improving Retrieval-Augmented Generation without Taxonomy-based Error Categorization

Gongbo Zhang
Columbia University
gz2366@cumc.columbia.edu

Yifan Peng*
Weill Cornell Medicine
yip4002@med.cornell.edu

Chunhua Weng*
Columbia University
cw2384@cumc.columbia.edu

Abstract

Retrieval-Augmented Generation (RAG) improves the factual accuracy of large language model (LLM) outputs by grounding generation in external knowledge. Recent agentic RAG systems extend this paradigm with critical agents to evaluate model responses and iteratively refine outputs. However, most prior work implicitly assumes reliable critic feedback and focuses on planning strategies, while paying limited attention to the robustness of the error-correction process itself, which can be impacted by misaligned error categories and ineffective or incorrect corrections. Here, *we hypothesize that RAG performance can be improved without explicit error categorization*. We propose RePAIR, a response–action learning paradigm that directly maps flawed RAG outputs to error-mitigating action plans without relying on fine-grained error taxonomies and explicit critic supervision. Across multiple benchmarks, RePAIR consistently improves agentic RAG performance.

1 Introduction

Large language models (LLMs) achieve strong generative performance but remain prone to factual hallucinations, which hinder their reliable deployment. Retrieval-Augmented Generation (RAG) partially mitigates this issue by grounding model outputs in external knowledge (Lewis et al., 2020; Izacard et al., 2023; Mialon et al., 2023; Fan et al., 2024; Zhang et al., 2025; Xu et al., 2025). More recently, agentic RAG systems extend this paradigm by introducing critic agents that evaluate model outputs and guide iterative actions over retrieval and generation (Asai et al., 2024; Wang et al., 2023; Yao et al., 2023; Yan et al., 2024; Dong et al., 2025; Fang et al., 2026). In this work, we adopt the narrower definition of agentic RAG used in prior refinement-based systems, focusing on iterative corrective planning over structured retrieval and generation operations rather than open-ended tool use or web-scale environments.

Although critic-guided refinement can improve end-to-end performance, most approaches assume the critic is reliable and focus on *using* critic feedback for planning modules (Madaan et al., 2023; Yao et al., 2023; Shinn et al., 2023), error taxonomies (Dong et al., 2025), and multi-step control flows (Asai et al., 2024; Zhou et al., 2024; Yan et al., 2024; Kim and Lee, 2024). In practice, LLM-based critics frequently misidentify failure causes and recommend ineffective corrections that can sometimes degrade performance relative to non-agentic baselines (Zheng et al., 2023; Wang et al., 2024; Huang et al., 2024; Liu et al., 2023; Kim and Lee, 2024). These observations suggest that reliance on explicit error categorization may introduce additional sources of uncertainty under noisy retrieval and generation.

This raises a key question: *is explicit error categorization essential for effective RAG improvement*? Existing agentic RAG systems typically treat critic-generated error categories as a central intermediate representation for planning and correction (Asai et al., 2024; Dong et al., 2025; Yan et al., 2024; Yao et al., 2023; Shinn et al., 2023). We hypothesize that *explicit error categorization is not essential to improve RAG performance*.

To test this, we introduce **RePAIR**, a RAG Response-Action learning paradigm that directly maps flawed RAG outputs to effective actions without relying on detailed error taxonomies or explicit critic supervision. RePAIR treats error categorization as a latent process and learns a policy over actions conditioned on the RAG state, thereby unifying error categorization and planning into a single learning objective. Our goal is not to invalidate taxonomy-based approaches, but to demonstrate that competitive corrective performance can be achieved without requiring explicit intermediate

*Equal contribution corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 155–165

error categorization.

Experiments on three benchmark datasets show that RePAIR improves token-level F1 by 3.8 points (13.1%) over standard RAG and outperforms all agentic RAG baselines.

Our contributions are three-fold:

- We propose a response–action learning formulation for RAG refinement that removes the need for explicit error categorization and unifies diagnosis and planning within a single policy.
- We introduce a two-phase training strategy that combines oracle-guided off-policy bootstrapping with on-policy refinement under deployment-matched conditions.
- We empirically show that competitive and stable performance can be achieved without explicit error taxonomies across multiple QA benchmarks.

2 Related Work

Our work bridges two directions: agentic RAG and evaluation-driven planning and reasoning in LLMs.

RAG improves factual reliability by grounding language model outputs in external knowledge (Lewis et al., 2020; Izacard et al., 2023; Mialon et al., 2023; Fan et al., 2024). Recent **Agentic RAG** explicitly categorizes errors and implements mitigating mechanisms. For example, Asai et al. (2024) integrated self-reflection via control tokens that regulate retrieval, relevance assessment, and filtering. Similarly, Zhou et al. (2024) separated cognition from metacognitive regulation by diagnosing knowledge-related failures and planning targeted corrections. Yan et al. (2024) employed an external evaluator to assess evidence reliability and trigger additional retrieval under noisy conditions. Across these approaches, critic-generated evaluations act as intermediate representations that guide subsequent retrieval or regeneration. Dong et al. (2025) formalized this design using a hierarchical error taxonomy combined with critic-guided action planning, while Ru et al. (2024) provided fine-grained diagnostic evaluations without learning policies. Collectively, these methods exemplify error categorization-based agentic RAG pipelines.

LLMs can also use evaluation signals to guide multi-step **planning and reasoning**. Prior work has framed reasoning as a search over intermediate states controlled by LLM-based evaluators (Yao et al., 2023), stored verbal self-critiques to inform subsequent actions Shinn et al. (2023), or applied iterative self-feedback to improve outputs (Madaan et al., 2023). In parallel, reinforcement learning approaches learn action policies from outcome-based signals for long-horizon planning (Yu et al., 2025; Li et al., 2025). Tool-augmented reasoning or multi-agent frameworks (Hong et al., 2024; Wu et al., 2023) emphasize general agent capabilities or structured coordination.

Although effective, these approaches rely on explicit error categorization or multi-step reasoning and assume reliable self-evaluation, which can lead to instability when feedback is unreliable. By contrast, our work directly learns response-to-action policies. This eliminates the need for predefined error taxonomies or external evaluators while still capturing latent error signals. Consequently, our method unifies error categorization and planning in a single, learnable framework, enabling more flexible and robust model behavior.

3 Methodology

We formulate RePAIR as a two-phase, preference-based policy learning problem. Given an initially failed RAG instance, the goal is to learn a policy that outputs a sequence of high-level RAG operations (i.e., a *plan*) whose execution produces a corrected answer. We distinguish an *off-policy* phase, where explicit correctness and error signals are available to bootstrap learning, from an *on-policy* phase, where the planner must infer failures solely from the raw RAG context.

3.1 RAG State and Plan Representation

Let q denote a user question, $D = \{d_1, \ldots, d_k\}$ the set of documents retrieved by a baseline retriever, and a_0 the initial answer produced by a RAG system. For training instances, we assume access to a ground-truth answer a_{gold}.

During off-policy training, the planner observes an oracle-derived binary correctness label $c \in \{0, 1\}$, obtained by comparing a_0 with a_{gold}, indicating whether the initial RAG answer is correct. When a_0 is incorrect, the planner additionally observes an incorrect reasoning trace r_0 produced by the baseline system. The planner, therefore, conditions on the augmented state $x_{\text{off}} = (q, D, a_0, c, r_0)$ during off-policy learning.

During on-policy training, we simulate the inference-time setting where gold answers are un-

available. The planner receives a coarse correctness estimate $\hat{c} \in \{0, 1\}$, inferred solely from the RAG response via an LLM-based judge, without access to a_{gold}. The planner thus conditions on the reduced state $x_{\text{on}} = (q, D, a_0, \hat{c})$ during on-policy learning. At inference time, the planner observes only the same reduced state and has no access to any explicit diagnostic or oracle signals.

Let $\mathcal{O}$ denote a predefined set of high-level RAG operations, including REWRITE, DECOMPOSE, RETRIEVAL, REFINEDOC, and GENERATEANSWER. A *plan* is a variable-length sequence of operations $y = (o_1, \ldots, o_T)$, where each $o_t \in \mathcal{O}$ and T denotes the length of the plan. Our experiments focus on this structured action space to isolate the effect of removing explicit error categorization; extending the approach to more open and complex action spaces (e.g., tool use or API interaction) is left for future work.

3.2 Response-Action Plan Policy

Given a RAG instance x and a plan y, an executor deterministically applies the sequence of operations specified by y to produce a revised answer, $\hat{a}(y; x) = \text{Executor}(x, y)$. The revised answer is evaluated against the ground-truth answer a_{gold} using a scalar reward function $R(x, y)$. Concretely, $R(x, y) \in \mathbb{R}$ is defined as the token-level F1 score between $\hat{a}(y; x)$ and a_{gold}, as described in the experimental setup. The reward thus quantifies the effectiveness of the plan in correcting the initial RAG failure.

For each input x, we execute a set of candidate plans $\{y_1, \ldots, y_n\}$ and induce pairwise preferences by comparing their rewards, such that $y_i \succ y_j$, iff $R(x, y_i) > R(x, y_j)$, yielding preference triples (x, y^+, y^-), where y^+ denotes the plan with the higher F1 score.

We learn a conditional plan policy $\pi_\theta(y|x)$ that maps a RAG state to a distribution over plans, and define a reference policy π_{ref} as the initial pretrained model. The planner is trained using Direct Preference Optimization (DPO) (Rafailov et al., 2023). For each preference triple (x, y^+, y^-), the training objective is

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \log \frac{\pi_\theta(y^+|x)/\pi_{\text{ref}}(y^+|x)}{\pi_\theta(y^-|x)/\pi_{\text{ref}}(y^-|x)} \right), \tag{1}$$

where σ denotes the logistic function and β controls the strength of regularization toward the reference policy.

Preference construction and noise. We construct DPO preference pairs using relative reward rankings over candidate plans. In some cases, both plans may achieve low absolute rewards, which could introduce noisy supervision. However, our objective is to optimize relative preference: DPO encourages the model to favor plans that yield comparatively better outcomes under the same conditions. In practice, two factors mitigate the impact of such noise. First, if the learned policy fails to identify a superior plan, the system defaults to the vanilla RAG response, which serves as a lower bound. Second, even when absolute rewards are low, relative differences often reflect variations in grounding or reasoning quality rather than reinforcing systematic hallucinations. Nevertheless, filtering near-tied or uniformly low-reward pairs may further reduce noise and improve robustness, which is beyond the scope of the current work.

3.3 Two-Phase Training

We adopt a two-phase training strategy to address a trade-off between learning stability and deployment alignment. Learning corrective plans directly from deployment-style inputs can be unstable, as the reward signal provides only coarse feedback on the entire action sequence rather than step-wise supervision. The *off-policy* phase mitigates this issue by leveraging oracle-derived correctness signals and reasoning traces to bootstrap stable response–action learning. However, reliance on oracle signals introduces a train–test mismatch, since such information is unavailable at inference time. The *on-policy* phase removes these oracle signals and refines the planner under deployment-matched inputs, using only coarse correctness estimates derived from the RAG response. This phase improves robustness by aligning training conditions with inference (Appendix A).

4 Experimental Setup

We follow the experimental protocol of RAG-Critic (Dong et al., 2025) and evaluate RePAIR on three question answering benchmarks (Table 1): *Natural Questions* (NQ) (Kwiatkowski et al., 2019), *Wizard of Wikipedia* (WoW) (Dinan et al., 2019), and *2WikiMultiHopQA* (2Wiki) (Ho et al., 2020), using the same evaluation splits. These datasets cover single-hop factual QA, knowledge-grounded dialogue, and multi-hop reasoning, respectively.

We compare RePAIR against a range of standard

Dataset	Task	Train	Test
NQ	Single-hop QA	79.1k	8.7k
2Wiki	Multi-hop QA	15.0k	12.5k
WoW	Dialogue Generation	63.7k	3.0k

Table 1: Benchmark dataset specifics.

and critic-based RAG baselines, including vanilla RAG without refinement, Self-Refine (Madaan et al., 2023), FLARE (Jiang et al., 2023), Self-RAG (Asai et al., 2024), MetaRAG (Zhou et al., 2024), and RAG-Critic (Dong et al., 2025). For fair and controlled comparisons, we adopt Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama3.1-8B (Meta, 2024) as a shared backbone across all methods, following the RAG-Critic configuration. Detailed experimental settings and configurations are provided in Appendices B and C.

We report token-level F1 on all datasets, computed as the lexical overlap between the predicted answer and the set of gold answers after normalization. For each example, the maximum F1 score across gold answers is used.

5 Results and Discussion

5.1 Main Results

Across all three benchmarks, RePAIR achieves the strongest overall performance (Table 2). It improves the average F1 by 3.8 points over the vanilla RAG baseline and outperforms all critic-centric agentic RAG methods. In contrast, prior approaches (e.g., Self-Refine, FLARE, and Self-RAG) show mixed or negative gains, sometimes underperforming vanilla RAG. RePAIR demonstrates stable improvements across datasets, achieving the best results on NQ and 2Wiki and the second-best on WoW, indicating robust effectiveness across both single-hop and multi-hop question answering. Compared to RAG-Critic, RePAIR also delivers consistently larger improvements without relying on explicit error taxonomies or critic supervision.

5.2 Impact of the Two-Phase Training

We examine the effectiveness of the two-phase training strategy by comparing three variants: off-policy only, on-policy only, and the full model (rows 7–9 in Table 2). Off-policy training alone underperforms because the planner relies on oracle correctness signals that are unavailable at inference time, resulting in a train–test mismatch. In contrast, on-policy training alone operates on deployment-style inputs but lacks sufficient supervision, making it difficult to learn effective corrective policies from sparse and noisy reward signals. The full two-phase RePAIR resolves this trade-off by first stabilizing learning with oracle-guided supervision and then refining the policy under inference-matched conditions. This combination yields better overall performance than either phase alone, indicating that both stages play a key role in effective training. These results suggest that LLM-based critics remain useful when providing coarse correctness signals, but fine-grained error categorization may introduce additional instability.

#	Method	NQ	WoW	2Wiki	Avg. (Δ)
1	*Standard RAG*	38.3	10.2	30.1	26.2
Agentic RAG					
2	Self-Refine	22.3	11.7	23.1	19.0 (↓7.2)
3	FLARE	19.8	4.2	22.7	15.6 (↓10.6)
4	Self-RAG	32.3	**17.4**	18.9	22.9 (↓3.3)
5	MetaRAG	40.2	6.2	29.2	25.2 (↓1.0)
6	RAG-Critic	**42.0**	11.6	33.1	28.9 (↑2.7)
Ours					
7	RePAIR (offline)	38.4	14.5	31.5	28.1 (↑1.9)
8	RePAIR (online)	38.8	14.5	33.0	28.8 (↑2.6)
9	RePAIR	40.3	15.3	**34.5**	**30.0** (↑3.8)

Table 2: Comparison between RePAIR and existing agentic RAG frameworks. Best token-level F1 scores are in **bold**, and second-best are underlined.

Dataset	Action	w/ Err.	w/o Err.	Δ (%)
NQ	Retrieval	310	138	-55.5
	Rewrite	12	0	-100.0
	Decompose	2	1	-50.0
	RefineDoc	3	0	-100.0
2Wiki	Retrieval	1563	491	-68.6
	Rewrite	86	43	-50.0
	Decompose	14	11	-21.4
	RefineDoc	20	7	-65.0
WoW	Retrieval	8	7	-12.5
	Rewrite	0	0	–
	Decompose	0	0	–
	RefineDoc	0	0	–

Table 3: Comparison of action usage with and without error categorization optimization across datasets.

5.3 Analysis of Planner Action Usage

We analyze the planner's action usage before and after optimization, excluding the final answer-generation step (Table 3). Across all datasets, optimization without explicit error categorization leads to substantially more concise action sequences, with large reductions in retrieval, query rewriting,

RAG Input State
Question:
Which act governs the working of banking companies in India?
Retrieved Passages:
Banking Regulation Act, 1949; Companies Act 2013; Reserve Bank of India Act, 1934...
Initial RAG Response:
To determine which act governs...

Golden label error categorization
High-level:
Incomplete/Missing Response; Inaccurate/Misunderstood Response; Irrelevant/Off-Topic Response; Erroneous Information
Fine-grained:
Content–Context Misalignment; Entity/Concept Confusion; Specificity/Precision Errors; Erroneous Retrieval
Planned Correction:
GENERATEANSWER

Incorrectly predicted error categorization
High-level:
Incomplete Information; Irrelevant Information
Fine-grained:
Insufficient or Incomplete Information Retrieval; Irrelevant Information Retrieval
Planned Correction:
REWRITEQUERY, RETRIEVAL, GENERATEANSWER

RePAIR Planned Correction:
GENERATEANSWER

Figure 1: An example of correct vs. incorrect error categorization. Misclassification results in ineffective refinement.

decomposition, and document refinement. Notably, these reductions do not compromise performance; RePAIR maintains comparable or improved QA results, indicating that effective behavior can be learned without frequent auxiliary actions. Importantly, actions such as query rewriting or document refinement remain valuable but are more effective when applied selectively rather than being triggered by potentially unreliable fine-grained error analysis. Overall, the optimized planner learns a more efficient policy that avoids redundant interventions while still correcting errors.

5.4 Case Analysis

Figure 1 illustrates how error categorization affects planning in agentic RAG. For a factual question on banking regulation in India, the retriever returns passages that cover multiple related statutes, while the initial RAG response fails to identify the correct governing act. With accurate error labels, this failure is recognized as a generator issue, prompting direct regeneration of the answer. In contrast, an incorrect error categorization misattributes the issue to the retriever, triggering unnecessary query rewriting and additional retrieval before regenerating the answer. By avoiding fine-grained misclassification, RePAIR directly selects the appropriate action, yielding more concise and effective plans.

6 Conclusion

We presented RePAIR, a response-action learning paradigm for agentic RAG that eliminates the need for explicit error categorization. By directly optimizing action policies from response–action preferences, RePAIR reduces dependence on brittle critic judgments and yields more stable refinement under noisy retrieval and generation conditions. Empirical results across multiple benchmarks demonstrate that RePAIR consistently outperforms critic-centric approaches in both accuracy and efficiency.

Limitations

While RePAIR demonstrates consistent gains over critic-centric agentic RAG baselines, our study has several limitations. First, experiments are restricted to three open-domain QA benchmarks and a fixed set of high-level RAG operations; performance and stability in other domains or with richer action spaces remain to be explored. Additionally, although RePAIR shows improved stability compared to critic-driven approaches, we do not comprehensively characterize failure modes under noisier and more adversarial retrieval conditions. Addressing these limitations is an important step toward understanding the generality of response–action learning in agentic RAG systems.

Acknowledgements

This work was supported by the National Library of Medicine [grant numbers R01LM014344, R01LM014573].

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Guanting Dong, Jiajie Jin, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2025. Rag-critic: Leveraging automated critic-guided agentic workflow for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),*

ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 3551–3578. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *CoRR*, abs/2401.08281.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 6491–6501. ACM.

Yilu Fang, Gongbo Zhang, Fangyi Chen, Yifan Peng, and Chunhua Weng. 2026. A critical evaluation of generative query expansion on biomedical literature retrieval. *Journal of the American Medical Informatics Association*, page ocag037.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7969–7992. Association for Computational Linguistics.

Kiseung Kim and Jay-Yoon Lee. 2024. RE-RAG: improving open-domain QA performance and interpretability with relevance estimator in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 22149–22161. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Woosuk Kwon, Zhuohan Li, Ying Zhuang, Yinan Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, and Ion Stoica. 2023. Efficient memory management for large language model serving with vllm. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. 2025. In-the-flow agentic system optimization for effective planning and tool use. *CoRR*, abs/2510.05592.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2356–2362. ACM.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2511–2522. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,

Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Meta. 2024. The llama 3.1 model family. *arXiv preprint arXiv:2407.21783*.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Trans. Mach. Learn. Res.*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9440–9450. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155.

Zihan Xu, Haotian Ma, Yihao Ding, Gongbo Zhang, Chunhua Weng, and Yifan Peng. 2025. Natural language processing in support of evidence-based medicine: A scoping review. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 21421–21443. Association for Computational Linguistics.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *CoRR*, abs/2401.15884.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *CoRR*, abs/2407.10671.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

Gongbo Zhang, Zihan Xu, Qiao Jin, Fangyi Chen, Yilu Fang, Yi Liu, Justin F. Rousseau, Ziyang Xu, Zhiyong Lu, Chunhua Weng, and Yifan Peng. 2025. Leveraging long context in retrieval augmented language models for medical question answering. *npj Digital Medicine*, 8(1):239.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yujia Zhou, Zheng Liu, Jiajie Jin, Jian-Yun Nie, and Zhicheng Dou. 2024. Metacognitive retrieval-augmented large language models. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1453–1463. ACM.

A Detailed RePAIR Algorithm Design

We adopt a two-phase training strategy that leverages diagnostic supervision when available, while avoiding reliance on such signals at inference time. The *off-policy* phase uses explicit correctness labels and reasoning traces to bootstrap stable response–action learning, whereas the *on-policy* phase removes these signals to align training with deployment conditions and ensure robustness without explicit error categorization.

In the **off-policy phase** (Algorithm 1), we assume access to a static log of RAG instances with correctness labels and reasoning traces:

$$\mathcal{D}_{\text{off}} = \{(x_{\text{off}}^{(i)}, a_{\text{gold}}^{(i)})\}_{i=1}^{N}. \tag{2}$$

For each $x_{\text{off}}^{(i)}$, a teacher model proposes candidate plans $\{y_1^{(i)}, \ldots, y_{n_i}^{(i)}\}$ conditioned on the augmented state $(q^{(i)}, D^{(i)}, a_0^{(i)}, c^{(i)}, r_0^{(i)})$. Each plan is executed to obtain a revised answer $\hat{a}(y_j^{(i)}; x_{\text{off}}^{(i)})$ and evaluated using the reward $R(x_{\text{off}}^{(i)}, y_j^{(i)})$. These scores are used to construct preference triples $(x_{\text{off}}^{(i)}, y^{(i,+)}, y^{(i,-)})$ and minimized using the DPO loss in Eq. (1), yielding an off-policy optimized planner π_θ^{off}.

In the **on-policy phase** (Algorithm 2), we further refine the planner *without* providing access to reasoning traces r_0. We consider a dataset

$$\mathcal{D}_{\text{on}} = \{(x_{\text{on}}^{(i)}, a_{\text{gold}}^{(i)})\}_{i=1}^{M}, \tag{3}$$

where $x_{\text{on}}^{(i)} = (q^{(i)}, D^{(i)}, a_0^{(i)}, \hat{c}^{(i)})$. For each $x_{\text{on}}^{(i)}$, the current planner π_θ (initialized from π_θ^{off}) generates one or more candidate plans $\{y_1^{(i)}, \ldots, y_{k_i}^{(i)}\}$. These plans are executed and evaluated using the same reward function $R(x_{\text{on}}^{(i)}, y_j^{(i)})$, enabling preference-based optimization under deployment-matched conditions.

B Details of Experimental Settings

All experiments were conducted in a Python environment based on PyTorch. Model training and inference were implemented using the HuggingFace ecosystem, with distributed training and preference optimization supported by `accelerate`, `deepspeed`, `trl`, and parameter-efficient fine-tuning via `peft` (Wolf et al., 2020; Rasley et al., 2020). Fast inference was enabled by `vllm` (Kwon et al., 2023). Retrieval components relied on FAISS and dense embedding toolkits, with sparse retrieval

Algorithm 1 Off-Policy DPO Training for RAG Planning

Require: Off-policy dataset $\mathcal{D}_{\text{off}}$, executor, scoring function R, reference policy π_{ref}

$\pi_\theta \leftarrow \pi_{\text{ref}}$
$\mathcal{P}_{\text{off}} \leftarrow \emptyset$
for each $(x_{\text{off}}, a_{\text{gold}}) \in \mathcal{D}_{\text{off}}$ **do**
  $x_{\text{off}} = (q, D, a_0, c, r_0)$
  Use a teacher model to propose candidate plans $\mathcal{Y}$ conditioned on x_{off}
  for each $y_j \in \mathcal{Y}$ **do**
    $\hat{a}_j \leftarrow \text{Executor}(x_{\text{off}}, y_j)$
    $s_j \leftarrow R(x_{\text{off}}, y_j)$
  end for
  Induce preferences over $\mathcal{Y}$ using scores $\mathcal{S}$ (e.g., by ranking) and construct one or more triples $(x_{\text{off}}, y^+, y^-)$
  Add all resulting triples to $\mathcal{P}_{\text{off}}$
end for
Train π_θ on $\mathcal{P}_{\text{off}}$ using the DPO loss in Eq. (1)
Denote the resulting planner as π_θ^{off}
return π_θ^{off}

Algorithm 2 On-Policy DPO Refinement for RAG Planning

Require: On-policy dataset $\mathcal{D}_{\text{on}}$, executor, scoring function R, reference policy π_{ref}, initialized planner π_θ^{off}, number of training iterations T

Initialize planner policy $\pi_\theta \leftarrow \pi_\theta^{\text{off}}$
for $t = 1$ to T **do**
  $\mathcal{P}_{\text{on}} \leftarrow \emptyset$
  for each $(x_{\text{on}}, a_{\text{gold}}) \in \mathcal{D}_{\text{on}}$ (or a minibatch) **do**
    $x_{\text{on}} = (q, D, a_0, \hat{c})$
    Sample or decode a set of candidate plans $\mathcal{Y}$ from current policy $\pi_\theta(\cdot|x_{\text{on}})$
    for each $y_j \in \mathcal{Y}$ **do**
      $\hat{a}_j \leftarrow \text{Executor}(x_{\text{on}}, y_j)$
      $s_j \leftarrow R(x_{\text{on}}, y_j)$
    end for
    Induce preferences over $\mathcal{Y}$ using scores $\mathcal{S}$ and construct one or more triples $(x_{\text{on}}, y^+, y^-)$
    Add all resulting triples to $\mathcal{P}_{\text{on}}$
  end for
  Update π_θ on $\mathcal{P}_{\text{on}}$ using the DPO loss (Eq. (1))
end for
return π_θ

baselines supported by `pyserini` and BM25 utilities (Lin et al., 2021; Douze et al., 2024).

C DPO Optimization and Configuration.

We trained the DPO objective using DeepSpeed ZeRO Stage 3 to enable memory-efficient optimization. Batch-related parameters, including the global training batch size, per-GPU micro-batch size, and gradient accumulation steps, were set to `auto` to allow DeepSpeed to adaptively determine optimal values based on available hardware resources. Parameter persistence thresholds were set to retain frequently accessed parameters in GPU memory when feasible, whereas 16-bit weights were collected only at model save time to minimize runtime overhead. Logging and diagnostic options were kept lightweight, with periodic step-level reporting and wall-clock breakdown disabled.

DPO training was performed using a single-stage schedule with a preference scaling coefficient $\beta = 0.1$. Models were trained for one epoch using the AdamW optimizer with a learning rate of 5×10^{-6} and a linear warmup over the first 10% of training steps. We used a per-device batch size of 1 with gradient accumulation over two steps, yielding a small effective batch size consistent with prior DPO setups. Input sequences were truncated to a maximum length of 4096 tokens, with prompts capped at 2048 tokens. Mixed-precision training was enabled using bfloat16 when hardware support was available, and attention kernels were selected adaptively, with optional support for FlashAttention when available.

D Case Analysis

Figure 1 illustrates a representative case highlighting how error categorization influences planning in agentic RAG. Given a factual question about banking regulation in India, the retriever surfaces passages mentioning several related statutes, including the *Banking Regulation Act, 1949*, the *Companies Act, 2013*, and the *Reserve Bank of India Act, 1934*. The initial RAG response fails to directly identify the governing act, resulting in an erroneous answer. Under the golden error categorization, this failure is correctly attributed to an answer-level error rather than a retrieval deficiency, leading to a direct correction via answer regeneration. In contrast, an incorrectly predicted error categorization attributes the failure to insufficient or irrelevant retrieval, triggering unnecessary query rewriting and additional retrieval steps before regenerating the answer. REPAIR avoids such redundant actions by bypassing fine-grained misdiagnosis and directly selecting the appropriate action, demonstrating how accurate or taxonomy-free error handling can yield more efficient and effective plans.

E Prompt Templates

We adopt a system prompt and a user prompt to guide the optimization of the RAG process. Both prompts are adapted from those used in RAG-Critic and are modified to better align with our settings. Specifically, the user prompt provides the RAG state, including the question, retrieved documents, prior model response, and error signal, and instructs the agent to generate only the minimal sequence of function calls necessary to resolve the error. The prompts emphasize concise, action-oriented planning without reliance on explicit error taxonomies.

Listing 1: System prompt for RAG optimization agent.

```
You are an agent tasked with optimizing a
    Retrieval-Augmented Generation process. The
    goal is to improve the model's predictions
    by addressing issues flagged in the
    error_type. You are given the results from
    an initial RAG process, including a query, a
     list of retrieved documents, a prediction,
    and the identified error type. Your task is
    to optimize the current RAG process by
    selecting the appropriate functions and
    generating the corresponding Python code to
    fix the problem.

Available Functions

1. Retrieval(query: str, topk: int) -> List[str]
   Purpose: Retrieves the top-k most relevant
     documents for a given query.
   Parameters:
     - query (str): input query
     - topk (int): number of documents
   Returns:
     - list of documents sorted by relevance

2. RewriteQuery(query: str, instruction: str) ->
     List[str]
   Purpose: Rewrite the query to better match
     relevant documents.
   Instructions:
     - "clarify": make the query more specific
     - "expand": add context or related terms

3. DecomposeQuery(query: str) -> List[str]
   Purpose: Decompose the query into more
     specific sub-queries.

4. RefineDoc(query: str, doc: str, instruction:
     str) -> str
   Purpose: Refine a document when it is not
     directly relevant.
```

```
   Instructions:
     - "explain"
     - "summarize"

5. GenerateAnswer(query: str, docs: List[str],
                  additional_instruction: str =
    None) -> str
   Purpose: Generate the final answer using the
    selected documents.

You can directly use the provided variables as
    inputs to the functions. You may freely
    combine functions to improve performance.
```

Listing 2: User prompt for RAG optimization.

```
Given the following information:

question = "{question}"
doc_list = {doc_list}
previous_pred = "{previous_pred}"

Error type of previous prediction:
{error_type}

Please carefully read the provided question,
    document list, previous answer, and the
    error type given by a teacher model. Your
    task is to generate Python code that calls
    the relevant functions to optimize the
    current RAG process and resolve the
    identified error.

The generated code should:
- Contain only function calls (no
    implementations)
- Use a minimal and necessary sequence of
    function executions
- End with: final_answer = GenerateAnswer(...)

Only output the code. Do not provide
    explanations.
```