# ReZero: Enhancing LLM search ability by trying one-more-time

**Anonymous ACL submission**

## Abstract

Retrieval-Augmented Generation (RAG) systems enhance Large Language Models (LLMs) but often falter when initial search queries fail. Existing approaches typically focus on query formulation or reasoning over results, lacking mechanisms to explicitly encourage persistence after a search failure. We introduce ReZero (Retry-Zero), a novel framework employing Group Relative Policy Optimization (GRPO) reinforcement learning to address this. ReZero incorporates a specific reward component, `reward_retry`, that directly incentivizes the LLM to retry search queries following an unsuccessful initial attempt, conditional on successful final answer generation. Experiments on the Apollo 3 mission dataset demonstrate ReZero's effectiveness: it achieved a peak accuracy of 46.88%, significantly outperforming a 25.00% baseline trained without the retry incentive. This highlights that rewarding persistence enhances LLM robustness in information-seeking scenarios where initial queries may prove insufficient.

## 1 Introduction

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by grounding them in external knowledge (Fan et al., 2024; Jeong et al., 2024; Kim et al., 2024). However, RAG effectiveness depends on the LLM-retriever interaction. Initial queries can fail, and complex tasks often require multi-step interaction (Trivedi et al., 2023; Yao et al., 2023).

Existing work often focuses on refining reasoning over retrieved documents (Madaan et al., 2023; Sun et al., 2025) or optimizing query generation for a single successful retrieval (Jiang et al., 2025). These methods may not sufficiently incentivize persistence when initial searches fail. The ability to recognize inadequacy and retry the search is crucial but underexplored.

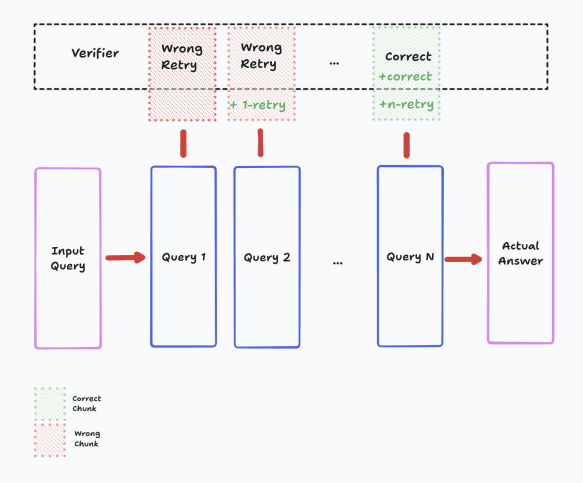


Figure 1: ReZero rewards the LLM for retrying search after failure.

To address this, we introduce ReZero (Retry-Zero), a novel framework improving LLM search ability by explicitly incentivizing persistence. We augment standard RAG reinforcement learning (RL) with a reward component specifically for retrying search queries after initial failures (Figure 1). Unlike approaches rewarding only final answer correctness or retrieval relevance, ReZero directly rewards the *process* of retrying, mirroring the human strategy: "if at first you don't succeed, try, try again."

ReZero uses RL to encourage exploration of different query strategies and learn when persistence is beneficial, avoiding premature abandonment or hallucination after a failed search.

Our contributions are: (1) ReZero, an RL framework rewarding search retries in RAG to foster persistence. (2) A novel reward function incentivizing this "retry" mechanism. (3) Positioning ReZero relative to work on RAG reasoning/query optimization, highlighting its unique focus on rewarding persistence.

## 2 Related Work

Our work intersects with research in Retrieval-Augmented Generation (RAG) and the application of Reinforcement Learning (RL) to enhance LLM reasoning and search.

### 2.1 Retrieval-Augmented Generation (RAG)

RAG systems (Fan et al., 2024; Jeong et al., 2024; Xu et al., 2024) ground LLMs in external knowledge. While early work used single retrieval steps, complex tasks often require multi-step RAG (Trivedi et al., 2023; Yao et al., 2023). Methods like Self-Ask (Press et al., 2023) and IRCoT (Trivedi et al., 2023) integrate reasoning (e.g., CoT) with iterative retrieval for incremental information gathering. However, these often assume effective retrieval or rely on prompting. ReZero distinctively targets the robustness of the retrieval interaction itself by directly encouraging retries after initial failures, a specific aspect of persistence less explored than sequential information gathering.

### 2.2 Learning and Search for Reasoning in RAG

Recent methods apply RL and process supervision to improve RAG reasoning. For instance, (Sun et al., 2025) focuses on enhancing the *trustworthiness* of multi-step reasoning over retrieved documents using PRMs and PEMs with MCTS/KTO (Ethayarajh et al., 2024; Pang et al., 2024). In contrast, ReZero focuses on incentivizing retries during the search interaction itself if initial attempts fail, complementing approaches that prioritize reasoning quality given retrieved context.

Similarly, (Jiang et al., 2025) uses RL (PPO) to optimize *query generation or rewriting* for better retrieval metrics (Recall, NDCG). While DeepRetrieval optimizes the quality of a single query attempt for maximal retrieval success, ReZero encourages multiple attempts by directly rewarding the retry mechanism, focusing on persistence.

### 2.3 Reinforcement Learning for LLMs

RL is widely used for LLM alignment (RLHF) (Ouyang et al., 2022) and enhancing capabilities like reasoning (DeepSeek-AI et al., 2025) and tool use (Schick et al., 2023). Techniques like ReFT (Wu et al., 2024) apply RL based on outcome or process rewards. ReZero leverages this concept but introduces a novel reward signal specifically for the retry action in a search context, targeting persistence in information retrieval.

Self-correction methods (Huang et al., 2024; Madaan et al., 2023) also involve iterative improvement, but typically focus on refining the LLM's generated output (reasoning, answers) based on feedback. ReZero differs by encouraging retries of the external tool interaction (search), addressing failures at the information-gathering stage.

In summary, ReZero uniquely uses RL to incentivize search persistence in RAG. Unlike work focusing on reasoning quality (ReARTeR) or single-query optimization (DeepRetrieval), ReZero rewards the "retry" mechanism itself, aiming for more robust information seeking.

## 3 Methodology

### 3.1 Overview

ReZero is a reinforcement learning (RL) framework designed to enhance the search capabilities of large language models (LLMs) in retrieval-augmented generation (RAG) systems. Inspired by recent advancements in RL for reasoning tasks (DeepSeek-AI et al., 2025) and motivated by findings suggesting RL fosters better generalization compared to supervised fine-tuning, ReZero utilizes Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to explicitly incentivize persistence—rewarding the model for retrying search queries when initial attempts fail.

### 3.2 Reinforcement Learning Framework

ReZero operates within a search environment where the LLM interacts with an external retrieval system. We employ the Group Relative Policy Optimization (GRPO) algorithm, noted for its effectiveness in training LLMs for reasoning tasks without requiring a separate critic model. The RL loop involves standard components: the **State** (current conversation history and retrieved information), the **Action** (LLM generation, including thoughts, searches, or answers, and potentially retries), the **Reward** (scalar signal from evaluation functions acting as a self-teacher), and the **Policy** (LLM strategy, fine-tuned via GRPO).

### 3.3 Reward Functions

ReZero employs multiple reward functions to guide the training via GRPO. These evaluate correctness, format adherence, retrieval quality, search strategy, diversity, and crucially, search persistence. See

Appendix A for full details. The key functions include:

1. `reward_correctness`: Evaluates final answer accuracy and structure (binary).

2. `reward_format`: Checks adherence to conversational format and tag usage (binary).

3. **`reward_retry`**: This reward function encourages the model to persist when initial search attempts do not yield sufficient information. It assigns a positive reward for each subsequent `<search>` query issued after the first one within a single generation sequence (i.e., for retries). The magnitude of the reward could potentially diminish with each additional retry to encourage efficiency. Crucially, this reward is conditional on task completion, it is only awarded if the model's final generated output in the sequence includes the complete `<answer>...</answer>` tags. If the sequence concludes without a well-formed answer enclosed in these tags, the `reward_retry` component contributes zero to the total reward for that trajectory, regardless of how many retries were performed. This mechanism prevents the model from learning to accumulate reward simply by retrying repeatedly without ever successfully generating a final answer.

4. `reward_em_chunk`: Verifies correct chunk retrieval via exact match (binary).

5. `reward_search_strategy`: Evaluates the quality and flow of the search process (graded).

6. `reward_search_diversity`: Assesses query variety and penalizes repetition (graded).

### 3.4 Training Process

The LLM is fine-tuned directly from a pre-trained base model using GRPO within an interactive framework involving a search engine verifier (Snell et al., 2025), drawing parallels to setups studied for improving generalization via RL (Chu et al., 2025). The initial reference policy ($\pi_{\text{ref}}$) is the base model itself.

The core training involves several steps: (1) **Iterative Interaction Loop (Rollout):** The LLM interacts with the verifier (Figure 1), potentially issuing multiple `<search>` queries within one sequence before generating an `<answer>`. (2) **Reward Calculation:** The completed sequence is evaluated using the reward functions (Section 3.3) to get a total trajectory reward. (3) **Policy Update (GRPO):** Rewards from multiple trajectories update the LLM parameters ($\theta$) via GRPO, comparing trajectory rewards to the batch average and stabilizing with KL divergence against $\pi_{\text{ref}}$. (4) **Noise Injection for Robustness:** Noise is added at the vector DB level during training to simulate imperfect retrieval, encouraging robust retry mechanisms.

This process directly fine-tunes the base LLM using RL, teaching it not only to answer correctly but also to strategically and persistently use the search tool, even when facing retrieval imperfections, without an intermediate supervised fine-tuning stage.

## 4 Experiments and Results

We implemented the ReZero framework (Section 3), fine-tuning a base LLM using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) within an interactive search environment (Snell et al., 2025), without intermediate supervised fine-tuning. We used the Apollo 3 mission dataset and the **Llama3.2-3B-Instruct** model (Grattafiori et al., 2024). Specific implementation details, dataset splits, and training parameters are provided in Appendix B.

To isolate the impact of the `reward_retry` mechanism, we compared two configurations: the **Baseline** (trained with all reward functions *except* `reward_retry`) and **ReZero** (trained with all rewards, including the crucial `reward_retry` component). Both models started from the same weights and used the same GRPO training procedure, differing only in the inclusion of the `reward_retry` signal.

Model performance was evaluated periodically on a held-out evaluation set using accuracy. The results (Figure 2) clearly indicate the effectiveness of the `reward_retry` component. The ReZero model achieved a peak accuracy of **46.88%** (at 250 steps), significantly outperforming the Baseline model's peak of **25.00%** (at 350 steps). ReZero also showed a faster initial learning rate. Both models exhibited a decline after their peaks, dropping to 0% accuracy later in training, suggesting potential RL instability or overfitting.

The substantial gap in peak accuracy (46.88%

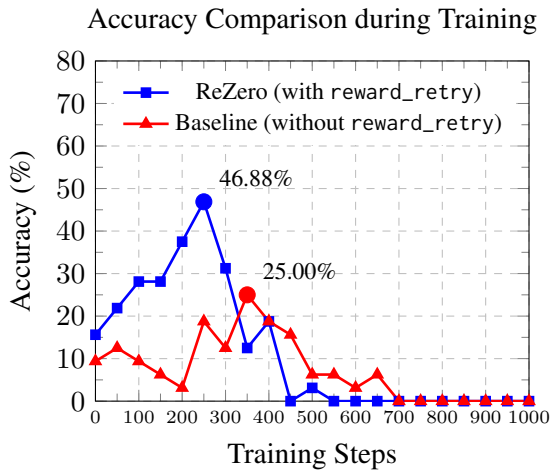


Figure 2: Comparison of evaluation accuracy between the ReZero model (incorporating the `reward_retry` component) and the Baseline model (lacking the retry incentive) over 1000 training steps on the held-out Apollo 3 dataset chunks. Peak accuracies are highlighted.

vs 25.00%) strongly suggests that explicitly rewarding the act of retrying search queries via the `reward_retry` function significantly enhances the model's ability to effectively utilize the search tool and ultimately arrive at correct answers, particularly in scenarios where initial queries might be insufficient.

## 5 Discussion

The nearly doubled peak accuracy of ReZero over the baseline (Section 4) strongly validates that explicitly rewarding search retries enhances LLM information retrieval. However, the subsequent performance decline in both models highlights challenges, likely related to RL training stability (e.g., GRPO instability, overfitting), requiring further investigation beyond the scope of this work (see Section 6).

## 6 Conclusion

We introduced ReZero, an RL framework using GRPO to enhance RAG system robustness by explicitly rewarding search persistence. Unlike methods focused on query formulation or reasoning, ReZero uses a `reward_retry` component to incentivize retrying after initial search failures, conditional on generating a final answer.

Experiments on the Apollo 3 dataset validate this approach. The ReZero model (with `reward_retry`) significantly outperformed a baseline lacking this incentive, achieving nearly double the peak accuracy (46.88% vs 25.00%) and faster initial learning. This confirms that rewarding retries improves the LLM's ability to overcome search failures.

However, performance declined after the peak for both models, indicating potential RL training instability or overfitting challenges common in RL (Chu et al., 2025), suggesting a need for further optimization. A key limitation acknowledged in Section 6 is the evaluation on a single domain; generalizability needs further validation.

Future work should focus on validating ReZero across diverse datasets and stabilizing the RL training. Exploring `reward_retry` refinements and integrating ReZero with complementary RAG techniques are also promising directions.

In conclusion, ReZero demonstrates that directly rewarding persistence improves LLM effectiveness in information seeking, highlighting the value of incorporating human-like problem-solving strategies into RAG frameworks.

## Limitations

Our study has several limitations. Firstly, the performance of both ReZero and the baseline model declined significantly after reaching peak accuracy during the 1000-step training process. This suggests potential challenges with the stability of the Group Relative Policy Optimization (GRPO) algorithm over extended training, possible overfitting to training data chunks, or suboptimal reward balancing, highlighting a need for further investigation into optimizing the RL training dynamics for sustained performance.

Secondly, our experiments were conducted solely on the Apollo 3 mission dataset. While this provided a controlled comparison, it represents a specific domain. Therefore, the generalizability of ReZero's observed performance benefits to diverse knowledge domains, query types, and task complexities requires further validation through evaluation on a broader range of datasets.

## References

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *Preprint*, arXiv:2501.17161.

dCaples. 2022. Autodidact. https://github.com/dCaples/AutoDidact. Accessed: 2025-04-11.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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 '24, page 6491–6501, New York, NY, USA. Association for Computing Machinery.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

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*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *Preprint*, arXiv:2503.00223.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. *Preprint*, arXiv:2404.13081.

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 *Thirty-seventh Conference on Neural Information Processing Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason E Weston. 2024. Iterative reasoning preference optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Song Yang, and Han Li. 2025. Rearter: Retrieval-augmented reasoning with trustworthy process rewarding. *Preprint*, arXiv:2501.07861.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval

with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. ReFT: Representation fine-tuning for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

## A Reward Function Details

This appendix provides the detailed descriptions of the reward functions used in ReZero, supplementing Section 3.3.

1. `reward_correctness`: Evaluates the final answer's accuracy against a ground-truth, checks response structure validity, and outputs a binary reward. The binary reward is determined by the model itself, acting as a self-judge (LLM-as-a-Judge) (Gu et al., 2025) using the base model's capabilities.

2. `reward_format`: Ensures adherence to the required conversational format and tag usage (e.g., tag sequence, valid markup), outputting a binary reward.

3. `reward_retry`: (See Section 3.3 for full description in main text).

4. `reward_em_chunk`: Verifies if the correct information chunk was retrieved by comparing the content in `<information>` tags against a ground-truth chunk using exact matching, outputting a binary reward.

5. `reward_search_strategy`: Evaluates the quality of the search process by checking adherence to a desired conversational flow: initiating a broad `<search>`, analyzing retrieved `<information>` (verified by specific keywords within `<think>` tags), executing subsequent refined `<search>` queries based on this analysis, and finally synthesizing an `<answer>` grounded in the analyzed information. Outputs a graded score (0.0-1.0) based on the successful execution of these sequential phases.

6. `reward_search_diversity`: Assesses the variety within the sequence of <search> queries used during generation. It rewards distinct query concepts and semantic dissimilarity between queries (measured using normalized string comparison). Bonus rewards are allocated for the effective use of diverse search operators (e.g., site:, "", OR). Penalties are applied to discourage submitting exact duplicate or highly similar queries, promoting broader exploration. Outputs a graded score (0.0-1.0) rewarding unique queries and operator diversity, while penalizing repetition and high similarity.

## B Experimental Setup Details

This appendix provides details for the experiments described in Section 4.

- **Base Model:** Llama3.2-3B-Instruct (Grattafiori et al., 2024).
- **Dataset:** Apollo 3 mission dataset, divided into 341 chunks (309 training, 32 evaluation).
- **Training Procedure:** Group Relative Policy Optimization (GRPO) (Shao et al., 2024) within an interactive environment using a search engine as a verifier (Snell et al., 2025). No intermediate supervised fine-tuning.
- **Hardware:** Single NVIDIA H200 GPU.
- **Duration:** 1000 steps (approx. 3 epochs over training data).
- **Libraries:** Implementation utilized `unsloth` and borrowed from (dCaples, 2022).