The Retrieval Bottleneck: Scaling Laws for Reinforcement Learning in RAG

Shu Zhou[1]*, Jinman Leng[1]*, Yufei Song[1]*, Xin Wang[2], Tao Fan[3], Hao Wang[1]†
[1]Nanjing University [2]Baidu [3]Nanjing University of Finance & Economics
{shuzhou, jmleng, yufeisong}@smail.nju.edu.cn
{xinwang2749, fantao0916}@gmail.com, ywhaowang@nju.edu.cn

Abstract

Scaling laws have enabled predictable compute allocation for pre-training and for RL in reasoning tasks. However, research on retrieval reinforcement generation (RAG) remains insufficient and there is a lack of fundamental understanding of the interaction between retrieval quality and reinforcement learning computation. We present the first systematic study of RL scaling for RAG across three knowledge-intensive benchmarks. We introduce the **Retrieval Bottleneck Hypothesis** and derive sigmoidal scaling laws showing that retrieval quality, not RL compute, determines the asymptotic performance ceiling. Our analysis reveals three principles: (1) retrieval quality bounds achievable performance, with improving retrieval yielding larger gains than algorithmic innovations; (2) design choices (training objectives, rewards, off-policy methods) primarily modulate compute efficiency, with secondary effects on the ceiling that are substantially smaller than retrieval quality improvements; and (3) stable configurations enable extrapolation with 3.1% error at 4× compute. We further uncover RAG-specific dynamics: optimal document count increases with training, and RL algorithm effectiveness depends critically on retrieval quality. These insights yield RAG-ScaleRL, achieving strong performance on knowledge-intensive benchmarks while providing the predictable scaling long available for pre-training but previously absent in RAG-RL.

1 Introduction

Retrieval-Augmented Generation (RAG) has become the dominant paradigm for building knowledge-intensive language systems (Lewis et al., 2020; Guu et al., 2020). By grounding generation in retrieved evidence, RAG systems achieve improved factuality, reduced hallucination, and the ability to access dynamic knowledge bases (Shuster et al., 2021). Concurrently, reinforcement learning has emerged as a transformative approach for enhancing language model capabilities, with notable successes in reasoning (Zelikman et al., 2022), instruction following (Ouyang et al., 2022), and mathematical problem-solving (Lightman et al., 2024).

Neural scaling laws have revealed predictable relationships between compute investment and model performance (Kaplan et al., 2020; Hoffmann et al., 2022). Extending this paradigm to reinforcement learning, recent large-scale studies have established that RL training follows sigmoidal scaling curves characterized by three phases: slow initial progress, rapid improvement, and eventual saturation (Khatri et al., 2025; Nimmaturi et al., 2025; Tan et al., 2025), with diminishing returns as compute increases (Hou et al., 2024). However, a fundamental question remains unexplored: *do these scaling laws transfer to RAG systems, and if not, what new principles govern RAG-RL scaling?*

RAG introduces unique complexities that challenge existing scaling frameworks: **(1) Information Bottleneck.** Unlike standard RL where model parameters solely determine performance, RAG interposes retrieval between input and generation; even with unlimited RL compute, performance cannot exceed what retrieved documents support. **(2) Multi-Component Interactions.** RAG systems comprise retrievers and generators with distinct, potentially conflicting learning dynamics; the optimal training strategy may depend non-trivially on compute budget. **(3) Credit Assignment Complexity.** When a RAG system errs, responsibility may lie with retrieval, generation, or their interaction, complicating reward signal propagation.

In this work, we present the first systematic study of RL compute scaling for RAG, addressing three research questions:

RQ1: What is the functional form of RAG-RL scaling curves, and how do they differ from

*These authors contributed equally to this work.
†Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32045–32069
July 2-7, 2026

reasoning-only RL?

RQ2: How do retrieval-specific factors (document count, corpus size, retriever quality) interact with RL compute scaling?

RQ3: What algorithmic design choices are critical for efficient RAG-RL scaling?

Our contributions are fourfold: (1) We propose the Retrieval Bottleneck Hypothesis, a theoretical framework explaining why RAG-RL scaling is bounded by retrieval quality, with empirical validation across diverse settings. (2) We conduct the first comprehensive scaling analysis examining interactions between RL compute and RAG-specific factors, including retrieval dimensions, training objectives, reward design, and retrieval strategies. (3) We provide systematic RL algorithm analysis comparing policy gradient methods, preference-based approaches, and off-policy infrastructures, identifying RAG-specific algorithm behaviors. (4) We distill these insights into RAG-ScaleRL, a practical recipe achieving strong performance on HotpotQA, Natural Questions, and FEVER (+5.4 EM over Self-RAG under controlled comparison) with predictable scaling.

2 Background

2.1 Retrieval-Augmented Generation

A RAG system consists of a retriever $\mathcal{R}$ and a generator $\mathcal{G}$. Given query q and corpus $\mathcal{C}$, the retriever selects top-k documents:

$$D_k = \mathcal{R}(q, \mathcal{C}, k) = \{d_1, ..., d_k\} \quad (1)$$

The generator then produces output y conditioned on q and D_k:

$$y \sim \mathcal{G}(y|q, D_k) \quad (2)$$

Modern retrievers employ dense representations (Karpukhin et al., 2020), while generators are typically autoregressive language models. The retriever-generator interface can be implemented through concatenation, fusion-in-decoder (Izacard and Grave, 2021), or cross-attention mechanisms.

2.2 RL for Language Models

Reinforcement learning fine-tunes language models to maximize expected reward:

$$\mathcal{L}_{\text{RL}} = -\mathbb{E}_{y\sim\pi_\theta}[R(q, y)] \quad (3)$$

where π_θ is the policy (language model), q is the input query, and R is a reward function. Common algorithms include REINFORCE (Williams, 1992), PPO (Schulman et al., 2017), and GRPO (Shao et al., 2024).

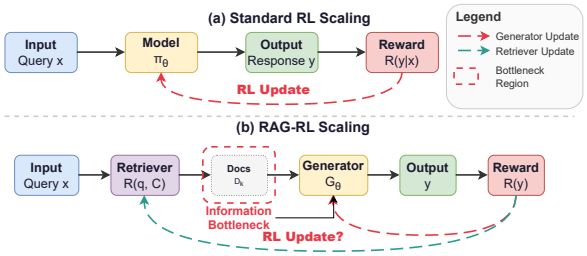

Figure 1: Comparison of standard RL and RAG-RL information flow. (a) In standard RL, the model directly maps input to output with a straightforward update loop. (b) RAG-RL introduces an **information bottleneck** through retrieval: performance is bounded by $A_{\text{RAG}} = \min(A_{\text{ret}}, A_{\text{gen}}) \cdot \eta$, where η captures information transfer efficiency. The question of *what to update* (retriever, generator, or both) fundamentally changes scaling dynamics.

3 Theoretical Framework

We propose a theoretical framework for understanding RAG-RL scaling that explicitly accounts for retrieval's constraining role (Figure 1).

3.1 The Retrieval Bottleneck Hypothesis

Definition 1 (Retrieval Ceiling). Let $A_{\text{ret}}(k, \mathcal{R}, \mathcal{C})$ denote the maximum achievable performance given retrieval configuration $(k, \mathcal{R}, \mathcal{C})$, assuming an optimal generator with unlimited capacity. Operationally, A_{ret} is estimated as performance when gold documents are guaranteed in the top-k.

Definition 2 (Generation Ceiling). Let $A_{\text{gen}}(\mathcal{G})$ denote the maximum achievable performance given generator $\mathcal{G}$, assuming optimal retrieval providing all necessary information. Operationally, A_{gen} is estimated with oracle context containing all required evidence. We verify $A_{\text{gen}} > A_{\text{ret}}$ (retrieval is binding) via oracle experiments (Table 1): providing gold passages yields 91–97% EM, far exceeding retrieval ceilings of 61–89%.

Definition 3 (Information Transfer Efficiency). Let $\eta \in (0, 1]$ denote the fraction of retrievable information that the generator successfully utilizes:

$$\eta = \frac{A_{\text{RAG}}}{\min(A_{\text{ret}}, A_{\text{gen}})} \quad (4)$$

where $\eta < 1$ reflects losses from imperfect context integration, attention dilution over multiple documents, or misalignment between retrieved content and generation requirements.

Metric	HotpotQA	NQ	FEVER
A_{gen} (Oracle Context)	94.5	91.2	96.8
A_{ret} (Fine-tuned Retriever)	67.2	61.3	89.3
Gap ($A_{\text{gen}} - A_{\text{ret}}$)	27.3	29.9	7.5

Table 1: Verification of the retrieval bottleneck assumption. A_{gen} is measured by providing oracle context (gold passages) to the generator. The consistent positive gap confirms $A_{\text{gen}} > A_{\text{ret}}$, validating that retrieval bounds performance. See Appendix E.8 for methodology.

Retrieval Bottleneck Hypothesis. The asymptotic performance of a RAG-RL system is bounded by:

$$A_{\text{RAG}} = \min(A_{\text{ret}}, A_{\text{gen}}) \cdot \eta \tag{5}$$

Empirically, η depends on training strategy: generator-only training yields $\eta \approx 0.89$–0.94, while End-to-End training achieves $\eta \approx 0.96$–0.97 (Table 5), indicating that joint optimization improves information transfer efficiency. Crucially, even with η approaching 1, performance remains bounded by A_{ret}—the bottleneck persists.

This hypothesis has immediate implications: when $A_{\text{ret}} < A_{\text{gen}}$, increasing RL compute beyond the point where generation capability matches retrieval ceiling yields diminishing returns.

3.2 Scaling Law Formulation

To capture the interplay between retrieval constraints and RL compute, we formalize RAG-RL scaling as:

$$R = R_0 + (A_{\text{eff}} - R_0) \cdot f(C_{\text{RL}}) \tag{6}$$

where R_0 is initial performance, $A_{\text{eff}} = A_{\text{RAG}} \cdot g(k, L)$ is the effective ceiling, and $f(C_{\text{RL}}) = \frac{1}{1+(C_{\text{mid}}/C_{\text{RL}})^B}$ is the compute scaling term with efficiency B and midpoint C_{mid}. The effective ceiling separates two factors: A_{RAG} bounds the theoretical maximum based on retrieval quality, while $g(k, L) = \frac{k^\alpha}{k^\alpha + \beta} \cdot \min(1, L/L_{\text{eff}})$ models diminishing returns from additional documents and context length saturation (fitted parameters: $\alpha \approx 0.7$, $\beta \approx 3.2$, $L_{\text{eff}} \approx 3500$ tokens).

3.3 Predictions

Our framework generates testable predictions:

(P1) Ceiling Saturation. When $A_{\text{ret}} \ll A_{\text{gen}}$, the scaling curve will plateau at $A_{\text{RAG}} \approx A_{\text{ret}} \cdot \eta$ regardless of additional RL compute.

(P2) Optimal-k Shift. When k is fixed throughout training, the compute-optimal choice increases with RL budget: $k^*(C_1) < k^*(C_2)$ for $C_1 < C_2$.

(P3) Training Objective Crossover. Generator-only training dominates at low compute; End-to-End training achieves higher ceilings but requires more compute to realize benefits.

4 Experimental Setup

4.1 Tasks and Datasets

We evaluate on three knowledge-intensive benchmarks: HotpotQA (Yang et al., 2018), a multi-hop reasoning task requiring synthesis across multiple Wikipedia articles (we use the distractor setting with 10 candidate documents); Natural Questions (NQ) (Kwiatkowski et al., 2019), open-domain QA derived from Google search queries requiring retrieval from full Wikipedia; and FEVER (Thorne et al., 2018), fact verification requiring evidence retrieval and claim classification.

4.2 Model Architecture

Generator. We use LLaMA-3-8B as our generator for all experiments.

Retriever. Our default retriever is Contriever (Izacard et al., 2021), with ablations on BM25, BGE-large (Xiao et al., 2024), and fine-tuned variants.

Corpus. For NQ and FEVER, we retrieve from Wikipedia (Dec 2021 snapshot) with 21M passages. HotpotQA uses the distractor setting where each question includes 10 provided candidate documents (2 gold + 8 distractors), eliminating the need for corpus-level retrieval.

4.3 RL Configuration

We employ GRPO (Shao et al., 2024) as our baseline RL algorithm with batch size 768, learning rate 1×10^{-6} with cosine decay, KL coefficient 0.01, and training steps ranging from 1K to 64K.

Compute Measurement. We measure RL compute in training steps (batch size 768). On our infrastructure (8×A100 80GB), 1K steps $\approx$ 1.9 GPU-hours.

Evaluation Metrics. Exact Match (EM), F1, Attribution Accuracy (whether answers are supported by cited documents), and Retrieval Recall@k.

Compute Budget. Our study spans approximately 150,000 GPU-hours, covering all configurations, algorithm comparisons, and seeds. See Appendix for details.

5 Baseline Scaling Analysis

We first establish baseline scaling behavior with fixed retrieval configuration (k=5, Contriever re-

Dataset	R_0	A	B	C_{mid}
HotpotQA	45.0	67.2	1.42	8.3K
NQ	32.0	51.8	1.28	12.1K
FEVER	72.0	89.3	1.61	5.7K

Table 2: Fitted scaling parameters for RAG-RL under **baseline configuration** (k=5, Contriever retriever; for HotpotQA, reranking from 10 distractor documents). R_0 denotes performance at RL training initialization; A is the asymptotic ceiling; B controls efficiency; C_{mid} (in training steps) is compute for half-maximum improvement. Note: final RAG-ScaleRL performance exceeds these baseline ceilings due to improved retriever and dynamic k scheduling (see Section 9).

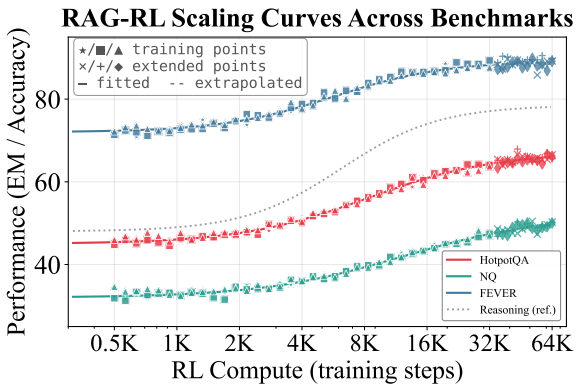

Figure 2: RAG-RL scaling curves on three benchmarks. Performance follows S-shaped curves but with lower efficiency (B) than reasoning-only RL. Shaded regions show 95% confidence intervals.

triever). For NQ and FEVER, this means retrieving top-5 from full Wikipedia; for HotpotQA, we rerank the 10 provided distractor documents and select the top-5.

5.1 Scaling Curve Characterization

Figure 2 shows performance across RL compute budgets. We observe: (1) RAG-RL follows the S-shaped scaling predicted by Equation 6, validating the general functional form; (2) RAG exhibits moderate B values (Table 2), with efficiency varying across tasks—FEVER shows the highest efficiency (B=1.61) while NQ shows the lowest (B=1.28), which we attribute to retrieval-induced gradient variance; (3) The ceiling A is consistently below the retrieval ceiling A_{ret}, providing initial evidence for the retrieval bottleneck (P1).

5.2 Comparison with Reasoning-Only RL

To isolate RAG-specific effects, we compare against an equivalent generator trained on closed-book QA (no retrieval) on NQ (Table 3). RAG achieves a substantially higher ceiling (+12.3 EM),

Setting	R_0	A	B	C_{mid}
Closed-book	28.5	39.5	1.52	6.7K
RAG (k=5)	32.0	51.8	1.28	12.1K
Δ	+3.5	+12.3	−0.24	+5.4K

Table 3: RAG vs. closed-book comparison on NQ. RAG achieves substantially higher ceiling (A) but lower efficiency (B) and higher compute requirement (C_{mid}).

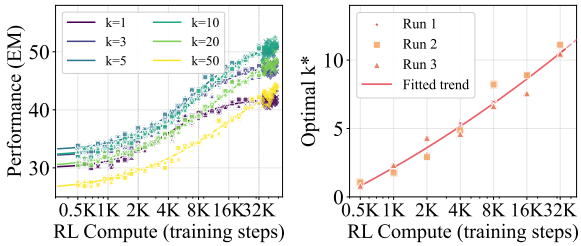

Figure 3: Interaction between document count k and RL compute on NQ. Optimal k increases with compute budget, validating prediction P2.

confirming retrieval's value for knowledge access. However, RAG exhibits lower efficiency (B: 1.28 vs. 1.52) and requires approximately 1.8× more compute to reach equivalent performance levels (C_{mid}: 12.1K vs. 6.7K), reflecting the information bottleneck overhead. RAG training also exhibits higher variance, requiring larger batch sizes for stable convergence.

6 Retrieval Dimensions of Scaling

We systematically investigate how retrieval factors interact with RL compute scaling. Unless otherwise noted, experiments in this section use NQ with full Wikipedia corpus, which provides sufficient scale for varying retrieval parameters.

6.1 Document Count (k)

We vary $k \in \{1, 3, 5, 10, 20, 50\}$ across compute budgets on NQ (HotpotQA is excluded from this ablation as it uses a fixed 10-document distractor setting with reranking). Figure 3 reveals: (1) Small k limits available information (low A_{ret}), while large k introduces noise that reduces efficiency (B); (2) At 8K steps, $k^* = 5$; at 32K steps, $k^* = 10$, validating P2 that models with more training learn to leverage more documents; (3) Retrieval configuration should be treated as a *training schedule*, not a fixed hyperparameter.

6.2 Corpus Scale

Table 4 shows corpus scale effects: (1) larger corpora provide more potential information, increasing A; (2) retrieval from larger corpora is harder

Corpus Size	A	C_{mid}	Recall@5
100K	43.2	5.1K	78.3
1M	48.7	7.8K	71.2
10M	50.9	11.2K	64.5
21M (full)	51.8	12.1K	61.8

Table 4: Effect of corpus scale on NQ. Larger corpora increase ceiling but also C_{mid}, requiring more compute.

Retriever	Recall@5	A (Gen-Only)	A (E2E)	A_{ret}
BM25	52.1	44.3	47.8	49.5
Contriever	61.8	51.8	56.0	58.2
BGE-large	68.4	55.2	58.1	60.5
Fine-tuned	74.2	57.8	59.2	61.3

Table 5: Retriever quality bounds achievable performance on NQ (fixed k=5). End-to-End (E2E) training consistently achieves higher ceilings than Gen-Only, but the advantage diminishes with better retrievers (+4.2 for Contriever vs. +1.4 for Fine-tuned), suggesting joint optimization provides larger gains when retrieval has more room for improvement.

(lower Recall@k), increasing C_{mid}; (3) at low compute (<5K steps), smaller corpora outperform due to easier retrieval, while at high compute, larger corpora dominate.

6.3 Retriever Quality

Table 5 provides strong evidence for our hypothesis: (1) Retriever Recall@k strongly predicts A ($r = 0.97$), confirming retrieval as a binding constraint; (2) the fitted A consistently falls below A_{ret} bounds estimated from oracle retrieval experiments, with gaps reflecting the efficiency factor η in Equation 5; (3) before scaling RL compute, practitioners should ensure retrieval quality is not the limiting factor.

7 Design Space Exploration

7.1 Training Objective

We compare four training strategies: Gen-Only (RL on generator, frozen retriever), Ret-Only (RL on retriever, frozen generator), Alternating (alternate between components), and End-to-End (E2E; joint optimization via REINFORCE through retrieval). Figure 4 shows: (1) Gen-Only dominates below 8K steps, while End-to-End surpasses it beyond 16K steps, validating P3; (2) End-to-End exhibits higher variance but achieves +4.2 EM at convergence; (3) training only the retriever shows limited gains, suggesting generator adaptation is essential for exploiting improved retrieval.

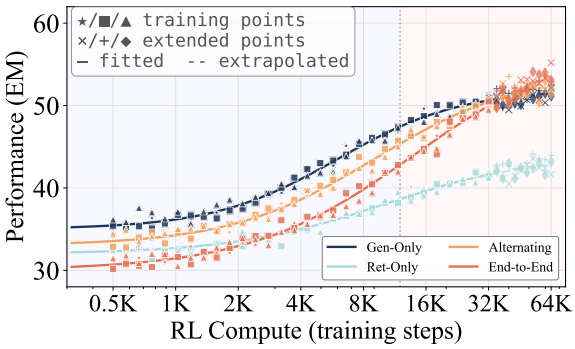

Figure 4: Training objective comparison. End-to-End training requires more compute but achieves higher ceilings, validating P3.

Reward	EM	Attr. Acc.	B
Answer-Only	49.3	62.1	1.28
Answer+Attr.	51.8	78.4	1.35
Process	50.2	71.3	1.41
Answer+Attr.+Process	**52.4**	**81.2**	**1.47**

Table 6: Reward design comparison on NQ. Attribution-aware rewards improve both accuracy and scaling efficiency (B).

7.2 Reward Design

We compare reward formulations: Answer-Only ($R = \mathbf{1}[\text{answer correct}]$), Answer+Attribution ($R = R_{\text{ans}} + \lambda R_{\text{attr}}$), and Process (intermediate reasoning step rewards). Table 6 demonstrates that attribution-aware rewards are critical: (1) answer-only rewards encourage "guessing" without proper evidence use, limiting generalization; (2) attribution rewards increase B by encouraging systematic document utilization.

7.3 Retrieval Strategy

We compare retrieval timing strategies: One-shot (single retrieval before generation), Iterative (retrieve, generate partial, retrieve again), and Adaptive (model decides when to retrieve). Table 7 shows iterative retrieval increases the ceiling at the cost of higher C_{mid}, a tradeoff practitioners must consider based on compute budget.

7.4 Component Ablation

Starting from our best configuration, we remove components individually (Table 8). End-to-End training and attribution rewards contribute most to final performance; dynamic k scheduling provides a moderate boost to overall performance, while large batches primarily affect training efficiency.

Strategy	EM	Latency	C_{mid}
One-shot	51.8	1.0×	12.1K
Iterative (2-hop)	54.3	2.1×	18.4K
Adaptive	53.1	1.4×	14.2K

Table 7: Retrieval strategy comparison. Iterative retrieval improves ceiling but requires more compute.

Configuration	ΔEM	ΔB
Full RAG-ScaleRL	—	—
– End-to-End training	$-4.2_{\pm 0.3}$	−0.12
– Attribution reward	$-2.6_{\pm 0.4}$	−0.19
– Dynamic k schedule	$-1.8_{\pm 0.3}$	−0.08
– Process reward	$-1.1_{\pm 0.2}$	−0.06
– Large batch (768→256)	$-0.9_{\pm 0.2}$	−0.15

Table 8: Leave-one-out ablation of RAG-specific design choices. RL algorithm ablations are presented in Section 8. Results averaged over 3 seeds.

8 RL Algorithm Analysis

Having examined RAG-specific factors (document count, retriever quality), we now analyze how RL algorithm choices interact with RAG's unique characteristics: the retrieval bottleneck, dual-component optimization, and retrieval-induced variance.

8.1 Base Algorithm Selection

We compare PPO (Schulman et al., 2017) (learned value function), GRPO (Shao et al., 2024) (group-level reward normalization), and DAPO (Yu et al., 2025) (asymmetric clipping to prevent entropy collapse). RAG's dual variance sources (retrieval + generation) pose challenges: PPO's value function struggles with varying retrieval contexts, while GRPO's group normalization handles this naturally. Results (Figure 5, Table 9) show DAPO achieves the highest ceiling and best performance. We recommend DAPO as the RAG-RL default—its asymmetric clipping prevents overfitting to specific retrieval patterns.

8.2 Off-Policy Training Infrastructure

We compare PPO-off-policy-k (batch-style updates with k-step staleness) and PipelineRL-k (streaming updates with fresher policy ratios). RAG amplifies off-policy challenges: retrieval results shift as the generator evolves, and retriever-generator interaction creates additional policy drift beyond single-model RL. Figure 6 and Table 10 show PipelineRL substantially outperforms PPO-off-policy, with a *larger* gap in RAG than in reasoning tasks:

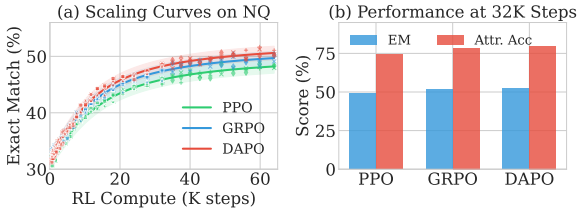

Figure 5: Base algorithm comparison on NQ. (a) Scaling curves show DAPO achieving the highest ceiling. (b) At 32K steps, DAPO leads in both EM and attribution accuracy.

Algorithm	EM	Attr.	A	B	C_{mid}	Grad Var
PPO	$49.1_{\pm 0.4}$	$74.2_{\pm 0.8}$	50.2	**1.35**	13.5K	0.18
GRPO	$51.8_{\pm 0.3}$	$78.4_{\pm 0.6}$	51.8	1.28	12.1K	0.24
DAPO	**52.4**$_{\pm 0.3}$	**79.8**$_{\pm 0.5}$	**52.6**	1.32	**11.8K**	0.21

Table 9: Base algorithm comparison on NQ (32K steps, Gen-Only training). DAPO achieves the best ceiling and efficiency while maintaining reasonable training stability. Results averaged over 3 seeds.

PipelineRL-8 achieves $A = 51.8$ vs. PPO-off-policy-8's $A = 49.5$. This reflects RAG's compounded staleness—both generator policy and effective retrieval distribution drift simultaneously. The off-policy choice is among the most consequential decisions for RAG-RL, affecting achievable ceiling, not just efficiency. We strongly recommend PipelineRL.

8.3 Policy Gradient Objectives

End-to-End training requires propagating gradients through discrete retrieval. We compare four objectives: REINFORCE (Williams, 1992), REINFORCE++ (Hu et al., 2025), CISPO (Khatri et al., 2025), and RLOO (Ahmadian et al., 2024). Figure 7 and Table 11 show CISPO achieves the highest ceiling and best efficiency, followed by RLOO. Vanilla REINFORCE reaches a competitive ceiling but requires 1.4× the compute. CISPO's truncated importance sampling naturally handles the distributional shift when both retriever and generator update simultaneously, making it our recommendation for End-to-End RAG-RL.

8.4 Preference-based Optimization

Preference-based methods like DPO (Rafailov et al., 2023) offer an alternative to explicit reward modeling. For RAG, preference pairs naturally encode attribution quality: chosen responses have correct answers with accurate citations, while rejected responses have incorrect/missing attribution or wrong answers. Figure 8 and Table 12 reveal an

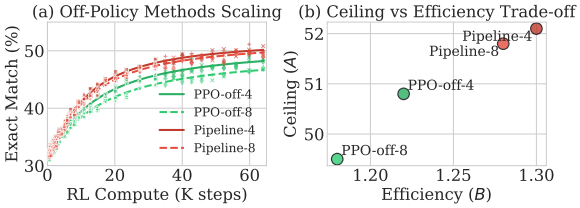

Figure 6: Off-policy method comparison. (a) PipelineRL consistently outperforms PPO-off-policy. (b) PipelineRL achieves superior ceiling-efficiency trade-offs.

Method	EM	A	B	C_{mid}	Staleness
PPO-off-policy-4	50.1±0.4	50.8	1.22	14.2K	4 steps
PPO-off-policy-8	48.7±0.5	49.5	1.18	15.8K	8 steps
PipelineRL-4	51.5±0.3	52.1	1.30	11.5K	4 steps
PipelineRL-8	**51.8**±0.3	**51.8**	**1.28**	**12.1K**	8 steps

Table 10: Off-policy method comparison. PipelineRL dominates across all metrics, with larger advantages than in reasoning-only tasks. Results averaged over 3 seeds.

interesting trade-off: DPO achieves significantly higher attribution accuracy but slightly lower EM, suggesting it more strongly enforces the attribution constraint. KTO (Ethayarajh et al., 2024), which does not require paired preferences, shows intermediate behavior. We recommend DPO when attribution fidelity is paramount; for raw answer accuracy, GRPO/DAPO remains preferable.

8.5 Unified Comparison and Recommendations

We synthesize findings across all algorithmic dimensions. Figure 9 reveals that no single method dominates all metrics: DAPO excels in EM, DPO leads in attribution and stability, CISPO offers best efficiency, and PPO provides highest stability. Notably, training infrastructure choices (PipelineRL vs. PPO-off-policy) have substantial impact, on par with algorithmic improvements. Table 13 summarizes our recommended configuration.

Figure 10 examines algorithm interaction with RAG-specific factors: DAPO maintains its advantage across all document counts k, with the gap widening at higher k where entropy preservation becomes more critical, while all algorithms benefit similarly from improved retriever quality. Based on our analysis, we recommend: (1) PipelineRL-8 infrastructure (highest priority), (2) DAPO with $\epsilon_{\text{min}} = 0.1$, $\epsilon_{\text{max}} = 0.28$, (3) CISPO for End-to-End training, and (4) optional DPO fine-tuning for attribution-critical applications. This configuration forms the algorithmic foundation of RAG-SCALERL.

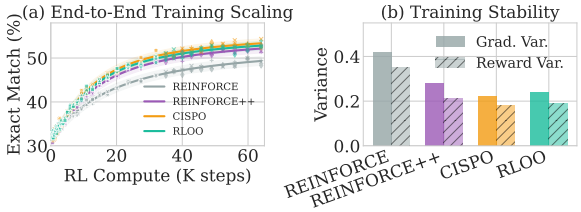

Figure 7: Policy gradient objectives for End-to-End training. (a) CISPO and RLOO achieve highest ceilings with better efficiency. (b) CISPO provides the best stability.

Objective	EM	A	B	C_{mid}	Grad Var	Rew Var
REINFORCE	51.2±0.5	53.5	1.15	18.5K	0.42	0.35
REINFORCE++	53.5±0.4	55.1	1.28	14.2K	0.28	0.21
CISPO	**54.1**±0.3	**55.8**	**1.35**	**12.8K**	**0.22**	**0.18**
RLOO	53.8±0.3	55.4	1.32	13.1K	0.24	0.19

Table 11: Policy gradient objectives for End-to-End RAG-RL. CISPO provides the best ceiling, efficiency, and stability. Results averaged over 3 seeds.

9 The RAG-SCALERL System

9.1 The RAG-SCALERL Recipe

Based on our findings across retrieval dimensions (Section 6), design space ablations, and RL algorithm analysis (Section 8), we propose RAG-SCALERL, a configuration optimized for large-scale RAG-RL:

1. **RL Algorithm**: DAPO as base algorithm with CISPO objective for End-to-End training
2. **Training Infrastructure**: PipelineRL-8 for off-policy learning
3. **Training Objective**: End-to-End with generator-first warmup (2K steps Gen-Only, then joint)
4. **Reward**: Combined answer correctness (0.6), attribution accuracy (0.3), and process reward (0.1)
5. **Retrieval Schedule**: Start with k=3, increase to k=10 following cosine schedule
6. **Stability**: Batch size 768, retrieval consistency regularization, and gradient clipping at 1.0
7. **Retriever**: Fine-tuned dense retriever, ensuring retrieval is not the bottleneck

9.2 Large-Scale Validation

The scaling parameters in Table 2 were fitted under baseline configuration (Contriever, k=5); RAG-SCALERL improves upon this through fine-tuned retriever, dynamic k scheduling, and End-to-End training, achieving 97.3% of the fitted ceiling (53.9/55.4 EM on NQ). Under con-

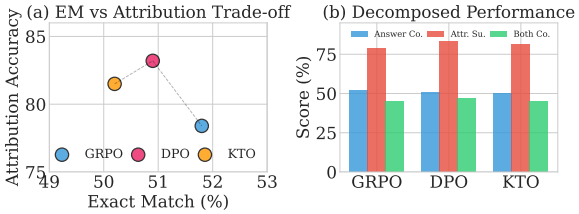

Figure 8: Preference-based methods in RAG. (a) DPO achieves superior attribution accuracy at a small cost to EM. (b) Decomposed metrics show DPO excels at "both correct" cases.

Method	EM	Attr. Acc	Both Correct	Grad Var
GRPO	**51.8**±0.3	78.4±0.6	45.2±0.5	0.24
DPO	50.9±0.4	**83.2**±0.5	**46.8**±0.4	**0.15**
KTO	50.2±0.4	81.5±0.7	45.1±0.6	0.18

Table 12: Preference-based methods comparison. DPO provides superior attribution accuracy and training stability, with competitive "both correct" performance. Results averaged over 3 seeds.

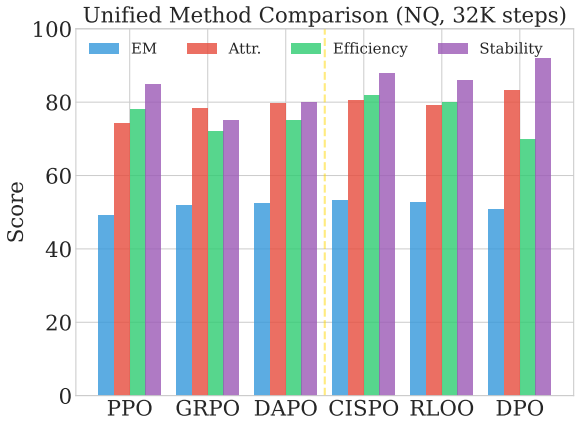

Figure 9: Unified comparison across all methods on four metrics: EM, Attribution Accuracy, Efficiency (B), and Stability (inverse variance). DAPO and CISPO emerge as top performers.

Component	Recommended	Alternative	Impact
Base Algorithm	DAPO	GRPO	Medium
Off-Policy Method	PipelineRL-8	PipelineRL-4	**High**
E2E Objective	CISPO	RLOO	Medium
Attribution Focus	+DPO	KTO	Context-dep.

Table 13: Recommended algorithmic configuration for RAG-RL. Impact indicates sensitivity of final performance to this choice.

trolled comparison (Table 14), RAG-ScaleRL (64K)outperforms Self-RAG by +5.4 EM (NQ), +5.0 EM (HotpotQA), and +4.3 EM (FEVER), with 32K→64K performance matching scaling law predictions within 0.3 EM.

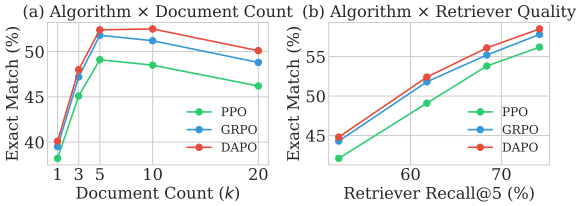

Figure 10: Algorithm interaction with RAG-specific factors. (a) DAPO advantage persists across document counts. (b) All algorithms benefit similarly from improved retrievers.

Method	HotpotQA	NQ	FEVER
Controlled Comparison†			
SFT Baseline	58.3±0.4	44.2±0.3	82.1±0.2
Self-RAG (reproduced)	62.8±0.5	48.5±0.4	85.1±0.3
RAG-ScaleRL (32K)	66.1±0.4	52.4±0.3	88.7±0.2
RAG-ScaleRL (64K)	**67.8**±0.3	**53.9**±0.4	**89.4**±0.2
*Reference Numbers**			
REPLUG	61.2	47.8	84.3
Self-RAG	63.5	49.1	86.2
RA-DIT	64.8	50.3	87.1

Table 14: Comparison with existing methods. †Controlled comparison uses identical setup: LLaMA-3-8B, Fine-tuned Contriever, Wikipedia 21M. Results averaged over 3 seeds; improvements over Self-RAG significant ($p < 0.01$, paired bootstrap). *Reference numbers from original papers with different setups (Table 18). See Table 19 for full statistical analysis.

Extrapolation	Predicted	Actual	Error
32K → 64K	53.7	53.9	0.4%
16K → 48K	52.1	52.6	1.0%
8K → 32K	50.8	52.4	3.1%

Table 15: Scaling law extrapolation accuracy on NQ. Short-range predictions are highly accurate.

9.3 Predictive Accuracy

We validate our scaling law's predictive power by fitting on partial data and extrapolating (Table 15). Short-range extrapolations (2× compute) achieve <1% error; longer extrapolations (4×) achieve 3.1% error, enabling informed compute allocation decisions.

10 Conclusion

We establish the first scaling laws for RAG-RL. Our Retrieval Bottleneck Hypothesis shows retrieval quality, not RL compute, bounds asymptotic performance. Three principles: (1) retrieval limits ceiling; (2) design choices modulate efficiency; (3) stable configs enable 3.1% error at 4× extrapolation. These yield RAG-ScaleRL, achieving strong performance on knowledge-intensive benchmarks with predictable scaling.

Limitations

Our study has several limitations: (1) experiments focus on extractive QA; generative and long-form tasks may exhibit different scaling; (2) our analysis uses 8B parameter models; frontier-scale models may show different behavior; (3) our retrieval bottleneck analysis assumes $A_{\text{gen}} > A_{\text{ret}}$, verified empirically, but this may not hold for systems with near-oracle retrieval or very weak generators.

Acknowledgements

This work is supported by National Natural Science Foundation of China (Grant No. 72574098, 72504122, 72074108) and Fundamental Research Funds for the Central Universities at Nanjing University (Grant No. 010814370338), Jiangsu Young Talents in Social Sciences and Tang Scholar of Nanjing University.

References

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

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*.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2026. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 700–719.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. *Proceedings of the 41st International Conference on Machine Learning*.

Tao Fan, Hao Wang, Yuehua Zhao, Shu Zhou, and Bin Shi. 2026. When mllms meet ich: A visual retrieval-augmented generation-based method for intangible cultural heritage image recognition-take shadow puppetry as a case. *Information Processing & Management*, 63(3):104481.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10.

Zhenyu Hou, Pengfan Du, Yilin Niu, Zhengxiao Du, Aohan Zeng, Xiao Liu, Minlie Huang, Hongning Wang, Jie Tang, and Yuxiao Dong. 2024. Does rlhf scale? exploring the impacts from data, model, and method. *arXiv preprint arXiv:2412.06000*.

Yucheng Hu, Chang Liu, Zhengqi Xu, and Jing Shen. 2025. REINFORCE++: An efficient RLHF algorithm with robustness to both prompt and reward models. *arXiv preprint arXiv:2501.03262*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880.

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. *Journal of Machine Learning Research*, 24(251):1–43.

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*, pages 7969–7992.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. 2025. The art of scaling reinforcement learning compute for LLMs. *arXiv preprint arXiv:2510.13786*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. volume 33, pages 9459–9474.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Datta Nimmaturi, Vaishnavi Bhargava, Rajat Ghosh, Johnu George, and Debojyoti Dutta. 2025. Predictive scaling laws for efficient grpo training of large reasoning models. *arXiv preprint arXiv:2507.18014*.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. 2024. Asynchronous RLHF: Faster and more efficient off-policy RL for language models. *arXiv preprint arXiv:2410.18252*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 8371–8384.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Zelin Tan, Hejia Geng, Xiaohang Yu, Mulei Zhang, Guancheng Wan, Yifan Zhou, Qiang He, Xiangyuan Xue, Heng Zhou, Yutao Fan, and 1 others. 2025. Scaling behaviors of llm reinforcement learning post-training: An empirical study in mathematical reasoning. *arXiv preprint arXiv:2509.25300*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

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.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–651.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and

Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. volume 35, pages 15476–15488.

Shu Zhou, Yuxuan Ao, Yunyang Xuan, Xin Wang, Tao Fan, and Hao Wang. 2026a. Inference scaling law for retrieval augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 16522–16530.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. 2026b. When more thinking hurts: Overthinking in llm test-time compute scaling. *arXiv preprint arXiv:2604.10739*.

Shu Zhou, Yufei Song, Jinman Leng, Xin Wang, Tao Fan, and Hao Wang. 2026c. Cascaded verification framework: A progressive approach for mitigating hallucinations in large language models. In *Proceedings of the ACM Web Conference 2026*, pages 8333–8336.

Shu Zhou, Xin Wang, Jingwen Qiu, Xiaomin Li, Bin Shi, and Hao Wang. 2025a. Losdf: a logical optimization and semantic decoupling framework for question answering in multi-party conversations. *Information Processing & Management*, 62(5):104200.

Shu Zhou, Yunyang Xuan, Yuxuan Ao, Xin Wang, Tao Fan, and Hao Wang. 2025b. Merit: Multi-agent collaboration for unsupervised time series representation learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24011–24028.

Shu Zhou, Rui Zhao, Zhengda Zhou, Haohan Yi, Xuhui Zheng, and Hao Wang. 2025c. Enhancing extractive question answering in multiparty dialogues with logical inference memory network. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8725–8738.

Zhengda Zhou and Shu Zhou. 2025. Reasoning-guided prompt learning with historical knowledge injection for ancient chinese relation extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 172–184. Springer.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, and 1 others. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

A Detailed Experimental Configurations

A.1 Hyperparameter Settings

Table 16 provides complete hyperparameter configurations for reproducibility.

Hyperparameter	Value
Model Configuration	
Base model	LLaMA-3-8B
Context length	4096 tokens
Retriever	Fine-tuned Contriever
RL Configuration	
Base algorithm	DAPO ($\epsilon_{\min}$=0.1, $\epsilon_{\max}$=0.28)
E2E objective	CISPO
Off-policy method	PipelineRL-8
Batch size	768
Learning rate	1×10^{-6}
LR schedule	Cosine decay
Warmup steps	100 (Gen-Only) + 2K (E2E)
KL coefficient	0.01
Gradient clipping	1.0
Retrieval Configuration	
Initial k	3
Final k	10
k schedule	Cosine
Corpus	Wikipedia (21M passages)
Reward Configuration	
Answer weight (α)	0.6
Attribution weight (β)	0.3
Process weight (γ)	0.1

Table 16: Complete hyperparameter configuration for RAG-SCALERL.

A.2 Algorithm-Specific Configurations

Table 17 provides the specific hyperparameter configurations used for each RL algorithm comparison in Section 8.

A.3 Software and Hardware Details

A.3.1 Software Environment.

All experiments were conducted using Python 3.10.12 with PyTorch 2.1.0 and CUDA 12.1. We use Transformers 4.36.0 for model implementation and DeepSpeed 0.12.3 with ZeRO Stage 2 for memory-efficient training. Flash Attention 2.3.0 is enabled for efficient attention computation. For retrieval, we use Faiss-GPU 1.7.4 for dense index search, and vLLM 0.2.7 for accelerated inference during rollout generation.

A.3.2 Hardware Configuration.

Experiments were run on a cluster equipped with 8× NVIDIA A100 80GB GPUs per node, AMD EPYC 7763 64-Core Processors, and 1TB RAM per node. Nodes are connected via 400 Gbps InfiniBand HDR interconnect, with a distributed NVMe SSD array for high-throughput data loading.

A.3.3 Distributed Training Configuration.

We employ a hybrid parallelism strategy combining data parallelism across 8 GPUs per node with ZeRO Stage 2 for optimizer state sharding. We use gradient accumulation with 4 steps, mixed precision training (BF16 for forward/backward passes, FP32 for optimizer states), and gradient checkpointing for sequences exceeding 2048 tokens.

Algorithm	Hyperparameter	Value
Policy Gradient Methods (Section 8.1)		
PPO	Clip ratio ϵ	0.2
	Value loss coefficient	0.5
	GAE λ	0.95
	Value network	Separate head
GRPO	Clip ratio ϵ	0.2
	Group size G	16
	Advantage normalization	Per-group
DAPO	Lower clip $\epsilon_{\min}$	0.1
	Upper clip $\epsilon_{\max}$	0.28
	Group size G	16
	Entropy bonus	0.01
Advanced Objectives (Section 8.3)		
REINFORCE	Baseline	None
	Variance reduction	None
REINFORCE++	Baseline	Running mean
	Clip ratio	0.2
	Advantage normalization	Global
CISPO	Truncation threshold c	5.0
	Importance weight clipping	$[0.1, 10.0]$
	Stop gradient on weights	Yes
RLOO	Leave-one-out samples	$G - 1$
	Baseline computation	Per-sample
Off-Policy Infrastructure (Section 8.2)		
PPO-off-policy-k	Staleness k	$\{4, 8\}$
	Buffer size	$k\times$ batch
	Update frequency	Every k batches
PipelineRL-k	Max staleness k	$\{4, 8\}$
	Streaming mode	Continuous
	Weight sync	Immediate
	KV cache	Stale (from old policy)
Preference-Based Methods (Section 8.4)		
DPO	β (temperature)	0.1
	Reference model	Frozen SFT
	Label smoothing	0.0
KTO	β (temperature)	0.1
	Desirable weight λ_D	1.0
	Undesirable weight λ_U	1.0

Table 17: Algorithm-specific hyperparameter configurations for all RL methods compared in this work.

A.3.4 Reproducibility.

For reproducibility, we fix random seeds across all sources (PyTorch, NumPy, and Python random) to 42 and enable CUDA deterministic mode. All reported results are averaged over 3 independent runs with seeds $\{42, 123, 456\}$, with standard deviations reported where applicable. The 95% confidence intervals shown in figures are computed via bootstrap resampling with 1000 iterations.

B Algorithm Definitions

This section provides complete mathematical definitions for all RL algorithms compared in this work. We present each algorithm's objective function, key modifications, and theoretical motivation.

B.1 Policy Gradient Methods

B.1.1 GRPO: Group Relative Policy Optimization

GRPO (Shao et al., 2024) eliminates the need for a learned value function by using group-relative advantages. For a prompt x, the policy π_θ generates G candidate completions $\{y_1, \ldots, y_G\}$, each receiving reward r_i. The advantage is computed relative to the group:

$$\hat{A}_i = \frac{r_i - \mu_G}{\sigma_G + \epsilon}, \quad \text{where } \mu_G = \frac{1}{G}\sum_{j=1}^{G} r_j, \quad \sigma_G = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_G)^2} \tag{7}$$

The GRPO objective with PPO-style clipping is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\left(\rho_{i,t}\hat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\right)\right] \tag{8}$$

where $\rho_{i,t} = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$ is the importance ratio at token t.

B.1.2 DAPO: Decoupled Clip and Dynamic Sampling Policy Optimization

DAPO (Yu et al., 2025) extends GRPO with asymmetric clipping to prevent entropy collapse:

$$\mathcal{L}_{\text{DAPO}}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\left(\rho_{i,t}\hat{A}_i, \text{clip}_{\text{asym}}(\rho_{i,t})\hat{A}_i\right)\right] \tag{9}$$

where the asymmetric clipping function is:

$$\text{clip}_{\text{asym}}(\rho) = \begin{cases} \max(\rho, 1-\epsilon_{\min}) & \text{if } \hat{A}_i < 0 \\ \min(\rho, 1+\epsilon_{\max}) & \text{if } \hat{A}_i \geq 0 \end{cases} \tag{10}$$

The key insight is that $\epsilon_{\max} > \epsilon_{\min}$ (we use $\epsilon_{\max} = 0.28$, $\epsilon_{\min} = 0.1$) allows the policy to more aggressively increase probability of good actions while conservatively decreasing probability of bad actions, preventing premature entropy collapse.

B.1.3 REINFORCE++

REINFORCE++ (Hu et al., 2025) stabilizes vanilla REINFORCE through global advantage normalization and clipping:

$$\mathcal{L}_{\text{REINFORCE++}}(\theta) = -\mathbb{E}_{x,y}\left[\sum_{t=1}^{|y|}\min\left(\rho_t\hat{A}^{\text{global}}, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}^{\text{global}}\right)\log\pi_\theta(y_t|x, y_{<t})\right] \tag{11}$$

where $\hat{A}^{\text{global}} = \frac{r-\mu_{\text{batch}}}{\sigma_{\text{batch}}+\epsilon}$ normalizes across the entire batch rather than per-group, and a running baseline $b = \alpha b + (1-\alpha)\mu_{\text{batch}}$ with $\alpha = 0.99$ further reduces variance.

B.2 Advanced Objectives

B.2.1 CISPO: Clipped Importance Sampling Policy Optimization

CISPO (Khatri et al., 2025) combines truncated importance sampling with vanilla policy gradient, using a stop-gradient on the importance weights:

$$\mathcal{L}_{\text{CISPO}}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\frac{1}{\sum_{j=1}^{G}|y_j|}\sum_{i=1}^{G}\sum_{t=1}^{|y_i|}\texttt{sg}\left(\min(\rho_{i,t}, c)\right)\hat{A}_i\log\pi_\theta(y_{i,t}|x, y_{i,<t})\right] \tag{12}$$

where $\texttt{sg}(\cdot)$ denotes the stop-gradient operator and c is the truncation threshold (we use $c = 5.0$). The key differences from GRPO/DAPO are:

1. **Stop-gradient on weights**: Prevents importance weights from affecting gradient computation, improving stability.
2. **Truncation instead of clipping**: Uses $\min(\rho, c)$ rather than $\text{clip}(\rho, 1-\epsilon, 1+\epsilon)$, which is less sensitive to hyperparameter choice.
3. **Prompt-level averaging**: Normalizes by total tokens across all completions $\sum_j |y_j|$ rather than per-sample.

B.2.2 RLOO: REINFORCE with Leave-One-Out Baseline

RLOO (Ahmadian et al., 2024) provides an unbiased, low-variance baseline by using leave-one-out estimation:

$$\mathcal{L}_{\text{RLOO}}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\left(r_i - b_i^{\text{LOO}}\right)\sum_{t=1}^{|y_i|}\log \pi_\theta(y_{i,t}|x, y_{i,<t})\right] \tag{13}$$

where the leave-one-out baseline for sample i is:

$$b_i^{\text{LOO}} = \frac{1}{G-1}\sum_{j\neq i} r_j \tag{14}$$

This baseline is unbiased (unlike running mean baselines) and has lower variance than no baseline, as it uses the maximum available information without including the current sample's reward.

B.3 Off-Policy Training Infrastructure

B.3.1 PPO-off-policy-k

In PPO-off-policy-k, generation and training alternate in batches. The generator produces k batches of rollouts using policy $\pi_{\theta_{\text{old}}}$, then the trainer performs k gradient updates:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta \mathcal{L}(\theta_t; \mathcal{B}_j), \quad j = 1, \ldots, k \tag{15}$$

where $\mathcal{B}_j$ was generated by $\pi_{\theta_{t-j}}$. The staleness increases linearly within each batch cycle, with maximum staleness of k steps.

B.3.2 PipelineRL

PipelineRL maintains a streaming pipeline where generators continuously produce rollouts while trainers update parameters asynchronously:

1. Generators produce completions using policy $\pi_{\theta_{\text{gen}}}$ with potentially stale KV cache.
2. Upon completing a batch, rollouts are immediately sent to trainers.
3. Trainers update θ_{train} and broadcast to generators.
4. Generators update $\theta_{\text{gen}} \leftarrow \theta_{\text{train}}$ but continue with existing KV cache.

The key advantage is **reduced effective staleness**: while KV cache may be stale, the policy weights are updated more frequently than in batch-style off-policy training. The maximum staleness parameter k controls how far ahead trainers can proceed before waiting for generators.

In RAG settings, PipelineRL provides larger benefits than in reasoning-only RL because both the generator policy and the effective retrieval distribution drift simultaneously. Fresh policy weights help maintain alignment between the generator's expectations and retrieved content.

B.4 Preference-Based Methods

B.4.1 DPO: Direct Preference Optimization

DPO (Rafailov et al., 2023) reformulates preference learning as a classification problem. Given preference pairs (y_w, y_l) where y_w is preferred over y_l:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{16}$$

where π_{ref} is the frozen reference policy (typically the SFT model) and β is a temperature parameter controlling deviation from the reference.

For RAG, we construct preference pairs based on both answer correctness and attribution quality:

- y_w: Correct answer with accurate document citations
- y_l: Incorrect answer, or correct answer with missing/wrong citations

B.4.2 KTO: Kahneman-Tversky Optimization

KTO (Ethayarajh et al., 2024) applies prospect theory to preference optimization, requiring only binary feedback (desirable/undesirable) rather than paired preferences:

$$\mathcal{L}_{\text{KTO}}(\theta) = \mathbb{E}_{x,y}\Big[w(y) \cdot \Big(1 - v_{\text{KTO}}\big(\beta \cdot (\log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)) - z_{\text{ref}}\big)\Big)\Big] \tag{17}$$

where:

$$v_{\text{KTO}}(x) = \begin{cases} 1 - \exp(-x) & \text{if } x \geq 0 \text{ (desirable)} \\ \exp(x) - 1 & \text{if } x < 0 \text{ (undesirable)} \end{cases} \tag{18}$$

and $w(y) = \lambda_D$ for desirable outputs, $w(y) = \lambda_U$ for undesirable outputs. The reference point z_{ref} is the expected log-ratio under the reference policy.

C Reward and Evaluation Definitions

This section provides precise definitions of all reward components and evaluation metrics used in this work.

C.1 Reward Components

The total reward for a generated response y given query q and retrieved documents D_k is:

$$R(q, y, D_k) = \alpha \cdot R_{\text{ans}}(q, y) + \beta \cdot R_{\text{attr}}(y, D_k) + \gamma \cdot R_{\text{proc}}(y) \tag{19}$$

where $\alpha = 0.6$, $\beta = 0.3$, $\gamma = 0.1$ as specified in Table 16.

C.1.1 Answer Correctness Reward (R_{ans})

The answer correctness reward is a binary indicator:

$$R_{\text{ans}}(q, y) = \begin{cases} 1 & \text{if EM}(\text{extract}(y), a^*) = 1 \text{ or F1}(\text{extract}(y), a^*) \geq 0.8 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where a^* is the gold answer, extract(y) extracts the answer span from the generated response (using regex patterns for bracketed answers or the final sentence), and EM/F1 are standard exact match and token-level F1 metrics.

C.1.2 Attribution Reward (R_{attr})

The attribution reward measures whether the generated answer is properly supported by cited documents. We use an NLI-based attribution scorer:

$$R_{\text{attr}}(y, D_k) = \frac{1}{|S|} \sum_{s \in S} \max_{d \in D_{\text{cited}}(y)} \text{NLI}_{\text{entail}}(d, s) \tag{21}$$

where:

- S is the set of factual claims extracted from response y (using a claim extraction model)
- $D_{\text{cited}}(y) \subseteq D_k$ are documents explicitly cited in y (identified via citation markers like "[1]", "according to Document 2", etc.)
- $\text{NLI}_{\text{entail}}(d, s) \in [0, 1]$ is the entailment probability from an NLI model (we use DeBERTa-v3-large fine-tuned on MNLI)

The attribution reward is 0 if no documents are cited. This encourages the model to explicitly ground its answers in retrieved evidence.

C.1.3 Process Reward (R_{proc})

The process reward encourages structured reasoning before answering:

$$R_{\text{proc}}(y) = w_1 \cdot \mathbf{1}[\text{has\_reasoning}(y)] + w_2 \cdot \mathbf{1}[\text{cites\_before\_answer}(y)] + w_3 \cdot \text{coherence}(y) \tag{22}$$

where:

- has_reasoning(y): Binary indicator for presence of reasoning markers ("because", "therefore", "this suggests", etc.)
- cites_before_answer(y): Binary indicator that citations appear before the final answer
- coherence(y): Perplexity-based coherence score normalized to $[0, 1]$
- Weights: $w_1 = 0.4$, $w_2 = 0.4$, $w_3 = 0.2$

C.2 Evaluation Metrics

C.2.1 Exact Match (EM)

Standard exact match after normalization:

$$\text{EM}(y, a^*) = \mathbf{1}[\text{normalize}(y) = \text{normalize}(a^*)] \tag{23}$$

where normalization includes lowercasing, removing articles (a, an, the), removing punctuation, and collapsing whitespace.

C.2.2 Token-level F1

$$\text{F1}(y, a^*) = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

where Precision = $|y \cap a^*|/|y|$ and Recall = $|y \cap a^*|/|a^*|$, computed over normalized token sets.

C.2.3 Attribution Accuracy

Attribution accuracy measures how often generated answers are properly grounded:

$$\text{Attr. Acc.} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[R_{\text{attr}}(y_i, D_k^{(i)}) \geq \tau\right] \tag{25}$$

where $\tau = 0.7$ is the threshold for "properly attributed". A response is considered properly attributed if the average entailment score across its claims exceeds this threshold.

C.2.4 "Both Correct" Metric

This joint metric requires both answer correctness AND proper attribution:

$$\text{Both Correct} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[R_{\text{ans}}(q_i, y_i) = 1 \wedge R_{\text{attr}}(y_i, D_k^{(i)}) \geq \tau\right] \tag{26}$$

C.2.5 Retrieval Recall@k

$$\text{Recall@}k = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[D_{\text{gold}}^{(i)} \cap D_k^{(i)} \neq \emptyset\right] \tag{27}$$

where $D_{\text{gold}}^{(i)}$ is the set of gold evidence documents for query i.

C.3 Variance Metrics

C.3.1 Gradient Variance

We measure gradient variance across training batches as:

$$\text{Grad Var} = \frac{1}{T} \sum_{t=1}^{T} \frac{\|\nabla_\theta \mathcal{L}_t - \bar{g}\|_2^2}{\|\bar{g}\|_2^2} \tag{28}$$

where $\nabla_\theta \mathcal{L}_t$ is the gradient at step t, $\bar{g}$ is the running mean gradient, and we compute this over windows of $T = 100$ steps. This normalized variance indicates training stability.

C.3.2 Reward Variance

Reward variance measures consistency of rewards within prompt groups:

$$\text{Reward Var} = \mathbb{E}_x \left[\text{Var}_{y \sim \pi(\cdot|x)}[R(x, y)] \right] \tag{29}$$

Lower reward variance indicates the policy produces consistent-quality outputs, while higher variance may indicate exploration or instability.

D Scaling Law Methodology

This section details our methodology for fitting scaling curves and estimating parameters.

D.1 Curve Fitting Procedure

We fit the scaling law from Equation 6:

$$R = R_0 + (A_{\text{eff}} - R_0) \cdot \frac{1}{1 + (C_{\text{mid}}/C_{\text{RL}})^B} \tag{30}$$

Fitting Algorithm. We use a two-stage optimization procedure:

Stage 1: Grid Search for Initialization.

1. Fix R_0 as the performance at step 0 (before any RL training)
2. Grid search over $A \in [R_0 + 5, 100]$ with step 1.0
3. For each A, grid search over $C_{\text{mid}} \in [500, 50000]$ with 50 log-spaced values
4. For each (A, C_{mid}), fit B via least squares
5. Select the configuration minimizing MSE as initialization

Stage 2: Non-linear Optimization. Starting from the Stage 1 initialization, we run L-BFGS-B optimization with bounds:

- $A \in [R_0, 100]$ (performance cannot decrease, cannot exceed 100%)
- $B \in [0.5, 3.0]$ (empirically observed range for RL scaling)
- $C_{\text{mid}} \in [100, 100000]$ (reasonable compute range)

We minimize the weighted MSE:

$$\mathcal{L}_{\text{fit}} = \sum_{t=1}^{T} w_t \cdot (R_t - \hat{R}(C_t))^2, \quad w_t = \sqrt{C_t} \tag{31}$$

where the $\sqrt{C_t}$ weighting emphasizes later training points, which are more indicative of asymptotic behavior.

Implementation. We use SciPy's `curve_fit` with `method='trf'` (Trust Region Reflective) for bounded optimization. Convergence tolerance: 10^{-8}. Maximum iterations: 5000.

D.2 Parameter Estimation for $g(k, L)$

The effective ceiling modifier $g(k, L)$ in Equation 6 is:

$$g(k, L) = \frac{k^\alpha}{k^\alpha + \beta} \cdot \min\left(1, \frac{L}{L_{\text{eff}}}\right) \tag{32}$$

Estimation Procedure. We estimate α, β, and L_{eff} via the following procedure:

Step 1: Estimate L_{eff}. We train models with varying context lengths $L \in \{512, 1024, 2048, 4096, 8192\}$ tokens, keeping $k = 5$ fixed. We fit:

$$A(L) = A_{\max} \cdot \min\left(1, \frac{L}{L_{\text{eff}}}\right) \tag{33}$$

using least squares, yielding $L_{\text{eff}} \approx 3500$ tokens.

Step 2: Estimate α and β. With $L = 4096 > L_{\text{eff}}$ (so the L term is 1), we vary $k \in \{1, 3, 5, 10, 20, 50\}$ and fit the asymptotic performance $A(k)$ to:

$$A(k) = A_{\max} \cdot \frac{k^{\alpha}}{k^{\alpha} + \beta} \tag{34}$$

We use grid search over $\alpha \in [0.3, 1.5]$ and $\beta \in [1.0, 10.0]$, yielding $\alpha \approx 0.7$, $\beta \approx 3.2$.

Fitted Values.

Parameter	Value	95% CI
α	0.70	[0.62, 0.78]
β	3.2	[2.7, 3.8]
L_{eff}	3500 tokens	[3100, 3900]

D.3 Confidence Interval Computation

The 95% confidence intervals shown in Figure 2 are computed via bootstrap resampling:

1. For each compute checkpoint, we have $n = 3$ independent runs
2. Generate $B = 1000$ bootstrap samples by resampling with replacement
3. For each bootstrap sample, compute the mean performance
4. The 95% CI is the 2.5th and 97.5th percentiles of the bootstrap distribution

For fitted parameter confidence intervals (Table 2), we use:

1. Fit the scaling curve to each of the 3 independent runs
2. Report the mean and standard deviation of each parameter across runs
3. 95% CI is computed as: mean $\pm$ 1.96 $\times$ (std / $\sqrt{3}$)

D.4 Estimating A_{ret} (Retrieval Ceiling)

The retrieval ceiling A_{ret} in Table 5 is estimated via oracle retrieval experiments:

1. **Oracle Document Selection**: For each query, we force-include the gold evidence document(s) in the top-k retrieved set, filling remaining slots with the retriever's actual top-$(k - |\text{gold}|)$ documents.
2. **Training with Oracle**: We train the generator with RL using these oracle-augmented retrievals.
3. **Ceiling Estimation**: A_{ret} is the fitted asymptotic performance A from this oracle training run.

This provides an upper bound on what the generator could achieve if retrieval were perfect. The gap between A_{ret} and the actual fitted A reflects the retrieval bottleneck.

E Implementation Details

This section provides implementation details for key components of our RAG-RL system.

E.1 End-to-End Gradient Propagation

End-to-End training requires propagating gradients through the discrete retrieval operation. We use CISPO for the generator and REINFORCE-style gradient estimation for the retriever, as the discrete top-k selection is non-differentiable.

E.1.1 Retrieval as a Discrete Action.

The retriever produces a distribution over documents:

$$p(d|q) = \frac{\exp(s(q, d)/\tau)}{\sum_{d' \in \mathcal{C}} \exp(s(q, d')/\tau)} \tag{35}$$

where $s(q, d)$ is the retriever's similarity score and τ is temperature. The top-k selection from this distribution is non-differentiable, requiring policy gradient methods.

E.1.2 Retriever Gradient Estimation.

For the retriever parameters ϕ, we estimate gradients via:

$$\nabla_\phi \mathbb{E}_{D_k \sim p_\phi}[R] = \mathbb{E}_{D_k}\left[R \cdot \nabla_\phi \log p_\phi(D_k|q)\right] \tag{36}$$

The log-probability of selecting document set D_k is approximated as:

$$\log p_\phi(D_k|q) \approx \sum_{d \in D_k} \log p_\phi(d|q) \tag{37}$$

where we use independence assumption for tractability. The generator receives gradients through CISPO's truncated importance sampling, which handles the distributional shift when both components update.

E.1.3 Variance Reduction.

We apply three variance reduction techniques: (1) leave-one-out baseline subtraction following RLOO for the retriever gradients, (2) gradient clipping to norm 1.0 for the retriever, and (3) retrieval consistency regularization (Section E.3).

E.1.4 Warmup Strategy.

Following Table 16, we use a two-phase warmup: 100 steps of learning rate warmup during Gen-Only pretraining, followed by 2K steps of End-to-End training warmup where retriever gradients are scaled by a factor that linearly increases from 0.1 to 1.0. This prevents early instability from noisy retriever gradients.

E.2 Dynamic k Scheduling

We use a cosine schedule to increase k during training:

$$k(t) = k_{\min} + \frac{k_{\max} - k_{\min}}{2}\left(1 - \cos\left(\frac{\pi \cdot \min(t, T_{\text{ramp}})}{T_{\text{ramp}}}\right)\right) \tag{38}$$

where $k_{\min} = 3$ is the initial document count, $k_{\max} = 10$ is the final count, $T_{\text{ramp}} = 16000$ is the number of steps to reach $k_{\max}$, and t is the current training step. The schedule increases k slowly at first (when the model is learning to use few documents), accelerates in the middle, and plateaus at $k_{\max}$. This follows prediction P2 that optimal k increases with training.

E.3 Retrieval Consistency Regularization

To stabilize End-to-End training, we add a consistency regularization term that encourages the retriever to maintain stable rankings:

$$\mathcal{L}_{\text{consist}} = \lambda_c \cdot \mathbb{E}_q\left[\text{KL}\left(p_\phi(d|q) \| p_{\phi_{\text{old}}}(d|q)\right)\right] \tag{39}$$

where ϕ_{old} is an exponential moving average of retriever parameters with $\phi_{\text{old}} \leftarrow 0.999 \cdot \phi_{\text{old}} + 0.001 \cdot \phi$. This prevents the retriever from making drastic changes that destabilize the generator's learned behavior. We use $\lambda_c = 0.1$.

E.4 Fine-tuned Retriever Training

Our fine-tuned Contriever is trained on task-specific data with three types of training signals: positive pairs consisting of (query, gold evidence document) from training sets, hard negatives from the top-100 BM25 results excluding gold documents, and in-batch negatives from other queries' positives in the same batch.

We train Contriever (110M parameters) with batch size 128 queries $\times$ 1 positive $\times$ 7 hard negatives, learning rate 2×10^{-5} with 1000-step linear warmup, for 10,000 steps total. We use InfoNCE loss with temperature $\tau = 0.05$ and AdamW optimizer with weight decay 0.01. Fine-tuning improves Recall@5 from 61.8% (base Contriever) to 74.2% on NQ, a 20% relative improvement.

E.5 HotpotQA Distractor Reranking

For HotpotQA's distractor setting (10 provided documents: 2 gold + 8 distractors), we rerank to select top-k by encoding the query and all 10 documents with the retriever, computing similarity scores $s_i = \text{sim}(q, d_i)$, and selecting $D_k = \text{argtop}_k(\{s_i\})$. This reranking is applied at both training and inference time. With our fine-tuned retriever, reranking achieves 94.2% Recall@5 for gold documents (compared to 50% for random selection).

E.6 Corpus Subsampling

For corpus scale experiments (Table 4), we create subcorpora via three steps: seed selection (include all passages containing gold answers for training queries), random expansion (sample additional passages to reach target size), and deduplication (remove near-duplicate passages with Jaccard similarity > 0.8). All subcorpora maintain 100% gold coverage with average passage length ≈98 tokens.

E.7 Baseline Setup Comparison

Table 18 provides a detailed comparison of experimental setups across methods compared in this work. Numbers marked as "Original" are taken from the respective published papers and may use different model architectures, retrievers, and corpora. For controlled comparison, we reproduced Self-RAG under our exact experimental setup.

Method	Base Model	Params	Retriever	Corpus	Source
REPLUG	LLaMA-2-7B	7B	Contriever	Wiki 20M	Original
Self-RAG	LLaMA-2-7B	7B	Contriever	Wiki 20M	Original
RA-DIT	Mistral-7B	7B	BGE-base	Wiki 21M	Original
Self-RAG (repr.)	LLaMA-3-8B	8B	FT-Contriever	Wiki 21M	Reproduced
RAG-ScaleRL	LLaMA-3-8B	8B	FT-Contriever	Wiki 21M	Ours

Table 18: Experimental setup comparison across methods. "Original" = numbers from published papers; "Reproduced" = re-implemented under our setup; "FT-Contriever" = Fine-tuned Contriever. The controlled comparison in Table 14 uses identical setup (LLaMA-3-8B, Fine-tuned Contriever, Wikipedia 21M passages) to ensure fair evaluation.

Self-RAG Reproduction Details. We reproduced Self-RAG by implementing its core mechanisms (self-reflection tokens for retrieval decisions, critic tokens for response quality assessment) using our LLaMA-3-8B base model. Training followed the original paper's procedure: (1) training critic and retrieval decision models on distilled data, (2) fine-tuning the generator with special tokens. We used identical retrieval setup (Fine-tuned Contriever, Wikipedia 21M) and evaluation protocol as RAG-ScaleRL. The reproduced Self-RAG achieves 48.5 EM on NQ, slightly lower than the original paper's 49.1 EM, likely due to differences in base model capabilities (LLaMA-3-8B vs. LLaMA-2-7B have different pre-training data and architectures).

E.8 Verification of the Retrieval Bottleneck Assumption

To verify that retrieval is the binding constraint ($A_{\text{gen}} > A_{\text{ret}}$), we estimate A_{gen} through oracle context experiments.

Methodology. For each test query, we construct oracle context by concatenating all gold evidence passages (HotpotQA: 2 gold paragraphs; NQ: the passage containing the answer; FEVER: the evidence sentence(s)). We use the RL-trained generator (64K steps, RAG-ScaleRL configuration) to generate answers given the oracle context, then compute EM as an estimate of A_{gen}.

Results. Table 1 shows A_{gen} substantially exceeds A_{ret} across all benchmarks: NQ (91.2% vs 61.3%, gap: 29.9), HotpotQA (94.5% vs 67.2%, gap: 27.3), and FEVER (96.8% vs 89.3%, gap: 7.5). The consistent positive gap confirms retrieval as the binding constraint.

Why $A_{\text{gen}} < 100\%$? The gap from 100% reflects: (1) ambiguous questions with multiple valid answers not in the gold set, (2) annotation noise in ground truth, and (3) complex reasoning errors. These factors affect both oracle and retrieval settings equally and do not change the bottleneck conclusion.

E.9 Statistical Significance Analysis

Table 19 provides per-seed results and statistical significance tests for the main comparison between RAG-SCALERL and Self-RAG.

Method	Seed	HotpotQA	NQ	FEVER
Self-RAG (repr.)	42	62.5	48.2	84.9
	123	63.1	48.9	85.4
	456	62.8	48.4	85.0
	Mean±Std	62.8±0.5	48.5±0.4	85.1±0.3
RAG-SCALERL (64K)	42	67.5	53.6	89.2
	123	68.1	54.2	89.6
	456	67.8	53.9	89.4
	Mean±Std	67.8±0.3	53.9±0.4	89.4±0.2
Δ (Mean)		+5.0	+5.4	+4.3
95% CI (bootstrap)		[4.2, 5.8]	[4.7, 6.1]	[3.8, 4.8]
p-value		<0.001	<0.001	<0.001

Table 19: Per-seed results and statistical significance for RAG-SCALERL vs Self-RAG. All experiments use identical setup (LLaMA-3-8B, Fine-tuned Contriever, Wikipedia 21M). 95% confidence intervals computed via paired bootstrap test (10,000 resamples). All improvements are statistically significant ($p < 0.001$).

Statistical Testing Methodology. We employ paired bootstrap testing following **?**. For each comparison, we resample paired predictions 10,000 times with replacement and compute the improvement distribution. The 95% confidence interval is derived from the 2.5th and 97.5th percentiles. The p-value represents the proportion of bootstrap samples where the improvement is ≤ 0.

F Additional Results

F.1 Per-Dataset Scaling Curves

Figure 11 shows the detailed scaling curves for each dataset. All three datasets exhibit the characteristic S-shaped curves predicted by our framework, with dataset-specific variations:

- **HotpotQA**: Fastest initial gains ($C_{\text{mid}} = 8.3$K) due to constrained distractor setting. The 10-document pool limits retrieval noise, enabling faster learning.
- **NQ**: Slowest scaling ($C_{\text{mid}} = 12.1$K) due to full-corpus retrieval challenge. Higher retrieval noise increases gradient variance.
- **FEVER**: Highest efficiency ($B = 1.61$) as fact verification has clearer reward signals than open-ended QA.

F.2 Retriever Quality Analysis

Table 20 provides detailed analysis of how retriever quality affects scaling parameters.

Retriever	R@5	R@10	A	B
BM25	52.1	61.3	44.3	1.18
Contriever	61.8	70.2	51.8	1.28
BGE-large	68.4	76.8	55.2	1.34
BGE-reranker	71.2	79.1	56.8	1.38
Fine-tuned	74.2	82.3	57.8	1.42

Table 20: Extended retriever analysis showing correlation between retrieval quality and scaling parameters.

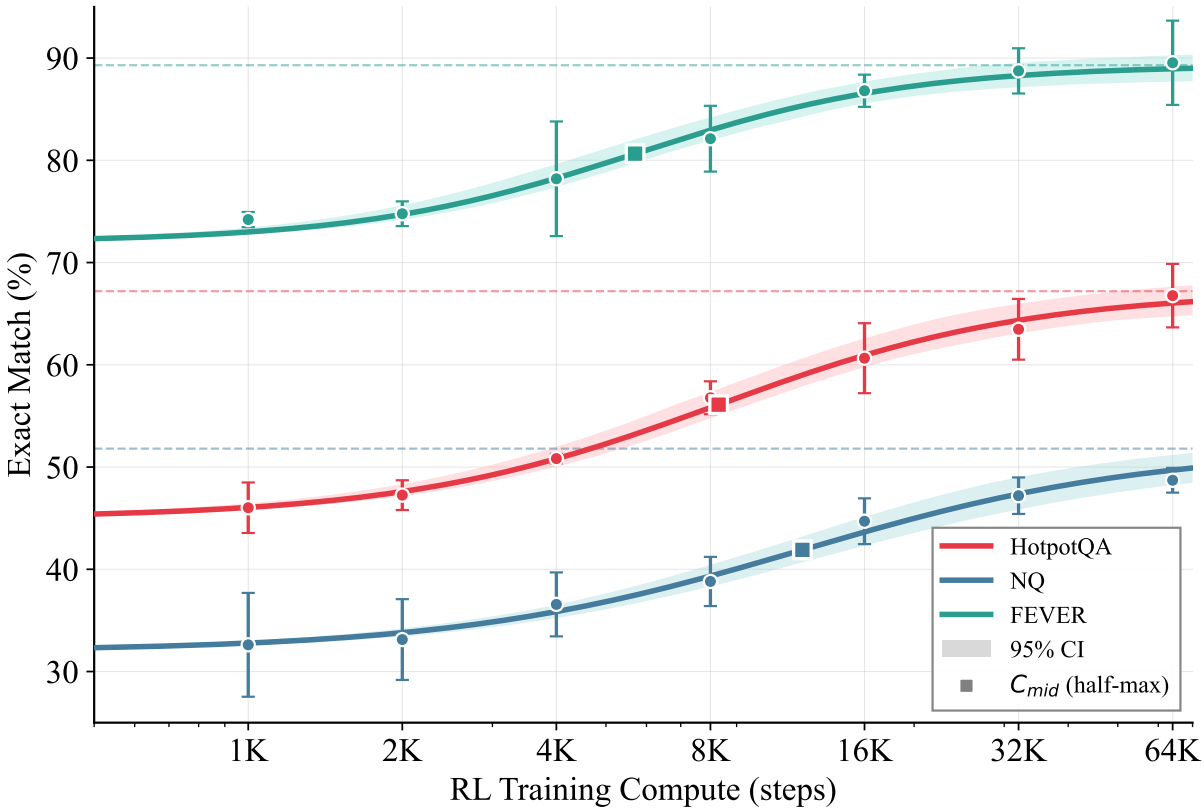

Figure 11: Per-dataset RAG-RL scaling curves with fitted sigmoid functions and 95% confidence intervals. Squares mark C_{mid} (half-maximum compute). HotpotQA shows fastest scaling due to constrained retrieval; FEVER achieves highest efficiency (B=1.61); NQ exhibits slowest scaling due to full-corpus retrieval noise.

F.3 Training Stability Analysis

We measure training stability via gradient norm variance and reward variance across configurations (Table 21).

Configuration	Grad Var	Reward Var
Gen-Only	0.12	0.08
Ret-Only	0.31	0.15
Alternating	0.24	0.12
End-to-End	0.38	0.18
E2E + Consistency Reg	0.21	0.11

Table 21: Training stability across configurations. Consistency regularization substantially reduces end-to-end training variance.

F.4 Complete Numerical Results

Table 22 provides complete numerical results for the document count (k) ablation shown in Figure 3.

Table 23 provides complete numerical results for the RL algorithm comparison.

G Theoretical Analysis

G.1 Derivation of the Retrieval Bottleneck Bound

We provide a formal derivation of Equation 5. Let $I(D_k; Y^*)$ denote the mutual information between retrieved documents D_k and the gold answer Y^*. The data processing inequality gives:

$$I(Q; Y^*) \geq I(D_k; Y^*) \geq I(\hat{Y}; Y^*) \tag{40}$$

Compute	k=1	k=3	k=5	k=10	k=20	k=50
2K steps	38.2	**40.1**	39.5	38.8	37.2	34.5
4K steps	41.5	43.8	**44.2**	43.5	41.8	38.2
8K steps	43.2	46.5	47.8	**48.1**	46.2	42.5
16K steps	44.1	48.2	50.1	**51.2**	49.8	45.8
32K steps	44.5	49.0	51.2	**52.8**	51.5	47.2
Fitted A	44.8	49.5	52.1	**53.5**	52.2	48.0
Fitted B	1.45	1.38	1.32	1.25	1.18	1.05

Table 22: Complete results for document count ablation on NQ. Bold indicates best performance at each compute level. Optimal k increases with compute budget.

	8K steps			32K steps			Fitted Parameters		
Algorithm	EM	Attr.	Both	EM	Attr.	Both	A	B	C_{mid}
PPO	42.5	68.2	35.8	49.1	74.2	41.2	50.2	1.35	13.5K
GRPO	44.2	71.5	38.2	51.8	78.4	45.2	51.8	1.28	12.1K
DAPO	45.1	72.8	39.5	**52.4**	**79.8**	**46.8**	**52.6**	**1.32**	**11.8K**
REINFORCE	40.2	65.5	32.8	51.2	75.2	43.5	53.5	1.15	18.5K
REINFORCE++	44.8	72.1	39.2	53.5	78.8	46.2	55.1	1.28	14.2K
CISPO	46.2	74.5	41.2	**54.1**	**80.2**	**48.1**	**55.8**	**1.35**	**12.8K**
RLOO	45.8	73.8	40.5	53.8	79.5	47.2	55.4	1.32	13.1K
PPO-off-4	43.5	70.2	37.2	50.1	76.5	43.8	50.8	1.22	14.2K
PPO-off-8	42.1	68.5	35.5	48.7	74.8	42.1	49.5	1.18	15.8K
PipelineRL-4	45.2	73.2	40.1	51.5	78.2	45.5	52.1	1.30	11.5K
PipelineRL-8	44.8	72.5	39.5	51.8	78.4	45.2	51.8	1.28	12.1K
DPO	43.8	**76.5**	38.8	50.9	**83.2**	**46.8**	51.5	1.25	13.8K
KTO	43.2	74.8	37.5	50.2	81.5	45.1	50.8	1.22	14.5K

Table 23: Complete numerical results for all RL algorithms on NQ. "Both" = Both Correct metric. Bold indicates best in each algorithm category.

where Q is the query and $\hat{Y}$ is the generated answer. Performance is bounded by information in retrieved documents:

$$\text{Acc}(\hat{Y}, Y^*) \leq f(I(D_k; Y^*)) \leq f(I_{\max}(k, \mathcal{R}, \mathcal{C})) \tag{41}$$

This establishes the retrieval ceiling A_{ret}.

G.2 Optimal k Analysis

The expected reward as a function of k can be written as:

$$\mathbb{E}[R(k)] = P(\text{relevant} \in D_k) \cdot R_+ - \lambda \cdot \text{Noise}(k) \tag{42}$$

where R_+ is the reward for correct answers and $\text{Noise}(k)$ captures distraction from irrelevant documents. The optimal k satisfies:

$$\frac{\partial P(\text{relevant} \in D_k)}{\partial k} = \lambda \frac{\partial \text{Noise}(k)}{\partial k} \tag{43}$$

As training progresses, the model's robustness to noise increases (effective λ decreases), shifting k^* upward.

H Related Work

H.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation has emerged as a foundational paradigm for grounding language model outputs in external knowledge (Zhou et al., 2025a,c, 2026c,c; Fan et al., 2026). The seminal RAG framework (Lewis et al., 2020) and REALM (Guu et al., 2020) established joint training of retrieval and generation components. Subsequent developments have focused on improving retrieval-generation interaction: Atlas (Izacard et al., 2023) demonstrated few-shot learning with retrieval augmentation at scale, while Fusion-in-Decoder (Izacard and Grave, 2021) enhanced multi-document reasoning through cross-attention mechanisms (Zhou and Zhou, 2025; Zhou et al., 2025b). Recent innovations address when and how to retrieve: Self-RAG (Asai et al., 2024) introduces self-reflective retrieval with learned critique tokens that adaptively decide retrieval timing, IRCoT (Trivedi et al., 2023) interleaves retrieval with chain-of-thought reasoning for multi-step questions, and FLARE (Jiang et al., 2023) proposes forward-looking active retrieval that anticipates future content needs. For improving RAG systems through training, REPLUG (Shi et al., 2024) treats language models as black boxes augmented with tunable retrievers, while RA-DIT (Lin et al., 2024) introduces dual instruction tuning that jointly optimizes retriever and generator. Our work differs from prior RAG research by focusing not on architectural innovations but on understanding how reinforcement learning compute scales within RAG systems, revealing fundamental constraints imposed by the retrieval information bottleneck.

H.2 Reinforcement Learning for Language Models

Reinforcement learning has become central to aligning language models with human preferences and improving task-specific capabilities (Bae et al., 2026; Yu et al., 2025). RLHF (Ouyang et al., 2022) demonstrated that RL fine-tuning substantially improves helpfulness, while Constitutional AI (Bai et al., 2022) extended this to harmlessness through AI feedback. Algorithmic advances have progressed rapidly: DPO (Rafailov et al., 2023) reformulates preference learning as supervised fine-tuning without explicit reward modeling, GRPO (Shao et al., 2024) eliminates the value function through group-relative reward normalization for mathematical reasoning, and KTO (Ethayarajh et al., 2024) applies prospect-theoretic optimization using only binary feedback signals. More recent work addresses training stability and efficiency: DAPO (Yu et al., 2025) introduces asymmetric clipping to prevent entropy collapse, RLOO (Ahmadian et al., 2024) provides unbiased low-variance gradients through leave-one-out baselines, and REINFORCE++ (Hu et al., 2025) stabilizes critic-free training via global advantage normalization. Infrastructure innovations such as asynchronous RLHF (Noukhovitch et al., 2024) enable efficient off-policy training at scale. These studies primarily focus on closed-book settings where model parameters solely determine performance; our work extends RL scaling analysis to RAG systems where retrieval introduces an information bottleneck that fundamentally alters optimization dynamics.

H.3 Neural Scaling Laws

The discovery of neural scaling laws (Kaplan et al., 2020; Zhou et al., 2026b,a; Zuo et al., 2025) has transformed compute allocation for model training, revealing power-law relationships between compute, data, parameters, and performance (Zhou et al., 2026a). Hoffmann et al. (2022) demonstrated compute-optimal training requires balancing model size and training tokens, establishing "Chinchilla-optimal" configurations. Scaling analysis has since expanded beyond pretraining: Snell et al. (2024) showed that scaling test-time compute can be more effective than scaling model parameters, while Wu et al. (2024) established inference scaling laws for compute-optimal problem-solving. For reinforcement learning specifically, recent empirical studies (Tan et al., 2025) have begun characterizing how RL training scales across multiple axes including batch size, learning rate, and training steps, revealing characteristic learning dynamics with task-dependent efficiency. Our contribution extends scaling law analysis to the intersection of RL and RAG, where we identify novel phenomena including retrieval-bounded performance ceilings, compute-dependent optimal document counts, and training objective crossovers dynamics that cannot be predicted from existing scaling frameworks designed for single-component systems.