FrugalRAG: Learning to retrieve and reason for multi-hop QA

Abhinav Java
Microsoft Research
India

Srivathsan Koundinyan
Microsoft Research
India

Nagarajan Natarajan
Microsoft Research
India

Amit Sharma
Microsoft Research
India

Abstract

We consider the problem of answering complex questions, given access to a large unstructured document corpus. The de facto approach to solving the problem is to leverage language models that (iteratively) retrieve and reason through the retrieved documents, until the model has sufficient information to generate an answer. Attempts at improving this approach focus on retrieval-augmented generation (RAG) metrics such as accuracy and recall and can be categorized into two types: (a) fine-tuning on large question answering (QA) datasets augmented with chain-of-thought traces, and (b) leveraging RL-based fine-tuning techniques that rely on question-document relevance signals. However, efficiency in the number of retrieval searches is an equally important metric, which has received less attention. In this work, we show that: (1) Large-scale fine-tuning is not needed to improve RAG metrics, contrary to popular claims in recent literature. Specifically, a standard ReAct pipeline with improved prompts can outperform state-of-the-art methods on benchmarks such as HotPotQA. (2) Supervised and RL-based fine-tuning can help RAG from the perspective of frugality, i.e., the latency due to number of searches at inference time. For example, we show that we can achieve competitive RAG metrics at nearly half the cost (in terms of number of searches) on popular RAG benchmarks, using the same base model, and at a small cost (1000 examples).

1 Introduction

We study the problem of answering questions, such as "*Can a microwave melt Toyota Prius battery*?", given access to a large unstructured corpus like Wikipedia. The de facto approach is to use language models (LMs) with retrieval, i.e., the retrieval-augmented generation (RAG) paradigm [Jeong et al., 2024, Jiang et al., 2023, Chan et al., 2024, Asai et al., 2023]. However, answering complex questions often requires multi-hop reasoning and retrieval: the LM must iteratively decompose the query into sub-queries (e.g., "*melting point of Toyota Prius battery*"), retrieve relevant documents, and reason through them until it can answer the original query.

Most prior work focuses on the accuracy of generated answers, using (a) supervised fine-tuning (SFT) on large QA datasets [Asai et al., 2023, Chan et al., 2024, Hsu et al., 2024], or (b) reinforcement learning (RL) methods such as GRPO [Jin et al., 2025] and DPO [Hsu et al., 2024]. Both rely on tens or hundreds of thousands of training examples, either as <question, answer> pairs or <question, documents> signals. For instance, SearchR1 [Jin et al., 2025] applies GRPO to fine-tune Qwen-2.5-7B-Instruct [Yang et al., 2024] using 100k examples, while LeReT [Hsu et al., 2024] uses 90k examples with Llama3.1-8B [Grattafiori et al., 2024]. Although, some prior work [Su et al., 2024,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

Yao et al., 2024] focuses on the efficiency of RAG at inference time, they also do so by training auxiliary modules with large amount of synthetic labeled data. In practice, however, labeled data is scarce *and* inference latency is critical.

In this work, we challenge recent literature in three ways: **(1)** efficiency, i.e., the number of hops per query, is as important as accuracy; **(2)** the simple ReAct baseline [Yao et al., 2023, Shao et al., 2023], with optimized few-shot prompting, is competitive with recent RAG techniques while requiring only tens of examples (vs. tens of thousands); and **(3)** RL can further improve such baselines for efficiency using *just 1000 training examples*.

Our solution, **FRUGALRAG**, is a two-stage framework that avoids large-scale labels while enabling effective and efficient search. In the *first stage*, the model is trained to maximize evidence coverage by generating diverse queries across multiple hops. In the *second stage*, we post-train the model to decide when to stop retrieving and generate an answer, explicitly balancing retrieval cost with evidence sufficiency. Optimizing for coverage and efficiency jointly in one stage leads to unstable training: models either over-retrieve or stop too early. Our key insight is that coverage is best learned through iterative querying (e.g., ReAct), while stopping is naturally learned via RL. By decoupling these, FRUGALRAG achieves both high recall and efficiency.

We evaluate FRUGALRAG on HotPotQA [Yang et al., 2018], 2WikiMultiHopQA [Ho et al., 2020], and MuSiQue [Trivedi et al., 2022b], showing that FRUGALRAG achieves the highest recall and answer quality with fewer retrievals—finetuned on only 1k examples.

2 FrugalRAG: Two-stage finetuning framework

We describe FRUGALRAG, a novel framework for enhancing retrieval augmented generation in LMs by decoupling the evidence exploration stage from answer generation. FRUGALRAG demonstrates several key advantages over existing approaches – (1) *Requires only 1000 annotated training examples* which is a 100 times reduction in dataset size compared to existing works [Jin et al., 2025, Hsu et al., 2024, Chan et al., 2024], (2) *Dynamically adapts test-time compute* which results in low inference time latency and high retrieval recall, unlike existing fixed compute methods [Hsu et al., 2024].

Setup. Let Q denote a complex user question that requires multiple iterative retrievals to answer. Let f denote a reasoner language model (LM) that examines the current context and determines the next action. At hop h (where $1 \leq h \leq B$, and B is the maximum allowed number of hops), the model f produces a *thought–action-search query* triplet (T_h, A_h, S_h). The search query S_h is passed to a retriever $\mathcal{R}(\cdot)$, which returns a set of documents: $\mathcal{D}_h = \mathcal{R}(S_h)$ from the document index, say $\mathcal{I}$. The challenge is to produce targeted queries that achieve high recall within the hop budget. Importantly, training requires only ground-truth evidence documents Y, not final answers.

Stage 1: Exploration (Evidence Coverage). Multi-hop questions benefit from issuing diverse search queries for retrievals. We therefore finetune f to maximize evidence coverage by leveraging ReAct [Yao et al., 2023] rollouts. Our dataset is constructed by sampling multiple queries per hop and selecting the one with highest recall against Y. Unlike prior RAG works that require large synthetic corpora [Chan et al., 2024, Asai et al., 2023], we generate only 1k examples. We then finetune f on a mix of rollouts (90% without FINISH, 10% with) to obtain a base policy f_S that prioritizes exploration while retaining the ability to stop. This ensures high recall and prevents premature termination. Additional details of data generation and finetuning are presented in the Appendix B.1.

Stage 2: Controlling Test-Time Compute. Given a finetuned base policy f_S, we aim to adaptively control rollout length for inference efficiency. Since f_S tends to over-explore, the objective is to learn when sufficient evidence has been gathered and stop early. We formulate this as a reinforcement learning problem: unlike prior work that uses RL to scale retrievals [Jin et al., 2025], our focus is on reducing unnecessary retrievals.

Reward Design. We guide the policy towards the optimal rollout length, denoted by h^* – such that any additional steps do not improve the recall. To compute the reward, we first generate the complete rollout using the policy f_S and then evaluate it. Mathematically,

Table 1: Retriever-level metrics (Recall, Support F1, Search latency) across HotpotQA, 2WikiMultiHopQA, and MuSiQue. FRUGALRAG achieves the best tradeoff between performance and efficiency on all datasets using Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct. Green (**bold**) represents the best score and blue (underlined) represents the second best score. "-" indicates results are not present in the respective papers.

Model	Method	HotpotQA			2Wiki			MuSiQue		
		Recall	Sup. F1	Search	Recall	Sup. F1	Search	Recall	Sup. F1	Search
Qwen2.5-3B-Instruct	Naive	0	0	0	0	0	0	0	0	0
	CoT	0	0	0	0	0	0	0	0	0
	CoT+RAG (n=3)	58.56	66.29	1	32.87	46.16	1	19.95	23.61	1
	CoT+RAG (n=5)	63.20	69.94	1	36.38	49.46	1	22.73	25.52	1
	ReAct	65,13	73.96	1.88	39.76	52.67	2.30	25.85	26.81	2.68
	ReAct FS	71.47	78.64	3.33	45.77	57.67	3.99	30.44	29.09	4.32
	IR-COT	59.11	66.68	2.04	36.82	49.75	2.11	23.30	25.80	2.05
	Search-R1	-	-	-	-	-	-	-	-	-
	FRUGALRAG-Explore	**81.41**	**85.72**	5.90	**54.13**	**64.17**	5.91	**37.02**	**31.89**	5.87
	FRUGALRAG	<u>78.77</u>	<u>83.89</u>	2.62	<u>52.70</u>	<u>62.79</u>	4.02	<u>33.97</u>	<u>30.56</u>	4.25
Qwen2.5-7B-Instruct	Naive	0	0	0	0	0	0	0	0	0
	CoT	0	0	0	0	0	0	0	0	0
	CoT+RAG (n=3)	58.56	66.29	1	32.87	46.16	1	19.95	23.61	1
	CoT+RAG (n=5)	63.20	69.94	1	36.38	49.46	1	22.73	25.52	1
	ReAct	73.45	80.03	2.07	45.80	58.62	2.78	30.94	29.11	3.13
	ReAct FS	77.65	82.80	2.91	49.25	61.03	3.48	33.90	30.50	4.16
	IR-COT	58.91	66.52	2.01	36.73	49.61	2.02	23.08	25.71	2.02
	Search-R1	-	-	-	-	-	-		-	-
	FRUGALRAG-Explore	**84.19**	**87.63**	5.88	**55.01**	**64.97**	5.89	**37.11**	**31.88**	4.32
	FRUGALRAG	<u>83.86</u>	<u>87.43</u>	2.75	<u>54.38</u>	<u>64.32</u>	4.86	<u>36.71</u>	<u>31.86</u>	3.98

$$
\mathbf{R} = \begin{cases} \max\left(-R_{\max}, \min\left(\log\left(\dfrac{1-\Delta}{\Delta}\right), R_{\max}\right)\right), & \text{if } \Delta > 0 \wedge c \geq \tau \quad \text{(late stop)} \\ R_{\max} + \alpha \cdot \left(\dfrac{h^*}{B}\right), & \text{if } \Delta = 0 \wedge c \geq \tau \quad \text{(perfect stop)} \\ \max\left(-R_{\max}, \min\left(\log\left(\dfrac{1-\Delta}{\Delta}\right), 0\right)\right), & \text{if } c < \tau \quad \text{(early stop)} \end{cases} \tag{1}
$$

Here, $\Delta = (h_{\text{term}} - h^*)/B$ is the normalized difference, $R_{\max}$ caps the reward, and α scales the bonus for stopping exactly at h^* (larger for longer rollouts). A query is considered answerable if recall $c \geq \tau$, where τ is set by the performance of the base explore model f_S. Intuitively, the reward penalizes deviations from h^* in proportion to $|\Delta|$, with maximum reward when $h_{\text{term}} = h^*$. The final reward is given by the mean of the main reward $\mathbf{R}$ (Eq. 1) with the format reward $\mathbf{R_f}$, lying in $[-R_{\max} - 1,\ R_{\max} + \alpha + 1]$. Further details on τ and $\mathbf{R_f}$ are in Appendix B.2.

Optimization. Motivated by the recent success and memory efficiency of GRPO [Shao et al., 2024], we adopt it as our optimization algorithm. At each hop h, we sample v tuples $\{T_h^i, A_h^i, S_h^i\}_{i=1}^v$, and retrieve their corresponding documents $\mathcal{D}_h^i$, and deduplicate. We repeat this until we reach the maximum budget $h = B$, collecting sample tuples and documents. For each rollout i, we then compute a cumulative reward using Eq. 1, and backpropagate through every logit produce by the policy along that rollout. Finally, once f_S emits the FINISH token, we mask any further generations. The detailed steps are provided in Algorithm 1.

3 Experiments

The detailed experimental setup and additional analysis are provided in the Appendix C and Appendix G respectively. Next, we discuss our key findings.

Main Results. In Tables 1 and Table 2, we compare FRUGALRAG against strong baselines on HotPotQA, 2WikiMultiHopQA, and MuSiQue. For fair comparison, all baselines use the ColBERTv2 [Santhanam et al., 2021] retriever indexed on Wikipedia (See Appendix E). The key

Table 2: Answer-level metrics (F1, EM, Match, Search latency) across HotpotQA, 2WikiMultiHopQA, and MuSiQue. (Details same as Table 1).

Model	Method	HotpotQA				2Wiki				MuSiQue			
		F1	EM	Match	Search	F1	EM	Match	Search	F1	EM	Match	Search
Qwen2.5-3B-Instruct	Naive	8.33	1.24	22.41	0	9.84	0.62	37.64	0	4.36	0.20	7.36	0
	CoT	13.90	6.54	22.23	0	15.31	7.07	25.38	0	6.62	5.95	1.94	0
	CoT+RAG (n=3)	35.29	21.85	43.29	1.0	18.15	12.19	21.54	1	9.15	4.26	8.27	1
	CoT+RAG (n=5)	36.56	25.37	36.63	1.0	20.11	13.78	24.33	1	9.63	3.97	8.02	1
	ReAct	25.35	19.84	42.91	1.88	15.87	2.64	34.84	2.30	7.64	1.44	11.75	2.68
	ReAct FS	50.88	38.02	47.65	3.33	23.83	16.23	24.18	3.99	13.30	7.73	11.66	4.32
	IR-COT	39.61	28.85	36.47	2.04	24.73	18.90	25.90	2.11	11.58	5.54	10.26	2.05
	Search-R1	-	32.4	-	-	-	**31.90**	-	-	-	10.30	-	-
	FRUGALRAG-Explore	53.17	40.56	49.65	5.90	25.97	17.00	28.19	5.91	**18.84**	**12.12**	**15.92**	5.87
	FRUGALRAG	**55.81**	**42.88**	**51.20**	2.62	**30.01**	21.46	**31.49**	4.03	18.71	11.17	15.76	4.25
Qwen2.5-7B-Instruct	Naive	13.63	5.49	24.30	0	16.43	9.88	32.47	0	6.40	0.70	8.00	0
	CoT	23.64	14.07	24.96	0	24.24	18.30	29.60	0	10.80	3.59	8.06	0
	CoT+RAG (n=3)	35.67	22.20	44.01	1	14.95	6.34	30.38	1	10.52	4.00	12.53	1
	CoT+RAG (n=5)	36.53	22.70	46.54	1	16.82	7.28	33.27	1	11.51	4.79	12.53	1
	ReAct	29.75	11.85	56.28	2.07	21.28	6.80	43.80	2.78	10.69	3.56	20.39	3.13
	ReAct FS	53.59	42.10	53.20	2.91	41.74	32.90	42.50	3.48	20.26	11.91	18.99	4.16
	IR-COT	36.53	24.92	36.36	2.01	16.05	8.96	20.45	2.01	11.11	5.70	9.68	2.02
	Search-R1	-	37.00	-	-	-	**41.40**	-	-	-	14.60	-	-
	FRUGALRAG-Explore	61.84	47.87	57.31	5.88	42.55	33.07	45.61	5.89	22.09	13.19	**22.67**	4.32
	FRUGALRAG	**63.03**	**48.81**	**58.25**	2.75	**45.00**	36.13	**47.00**	4.86	**24.78**	**14.89**	21.84	3.98

takeaway is that FRUGALRAG consistently outperforms the baselines on both answer and retrieval metrics, using competitive or significantly smaller number of searches on average.

The *ReAct Few Shot (FS) baseline* [Yao et al., 2023, Khattab et al., 2023] outperforms vanilla Retrieval Augmented Generation and ReAct without few shot prompts consistently. The optimized prompts enable ReAct FS to generate more search queries, and as a result achieve very strong performance. Strikingly, on HotPotQA, ReAct FS achieves (a) **77.44%** and **77.65%** recall with Llama3.1-8B-Instruct, Qwen2.5-7B-Instruct respectively, which is greater than the recall achieved by LeReT [Hsu et al., 2024] (77.1% using Llama3.1-8B) after significant finetuning (Table 3 in Appendix); and (b) comparable Exact Match (42.1) to that of the recently proposed methods such as Search-R1 [Jin et al., 2025] (43.3 using base, 37.0 using instruct); signifying the importance of building a strong baseline.

Table 1 and Table 2 demonstrate the overall *effectiveness of* FRUGALRAG-*Explore* achieving the highest Recall and Support F1 scores compared to all baselines. However, we note that FRUGALRAG-Explore is either best or second best in terms of both answer and recall metrics but introduces a very high latency compared to FRUGALRAG. Stage-2 finetuning significantly reduces the number of searches required while retaining the performance across three datasets. For instance, the search count on HotPotQA reduces on average by **-53.05%** across the three models and the F1 scores improve by **+3.5%** for a marginal drop in recall. A similar trend is observed in the case of 2Wiki and MuSiQue, where the search count reduces on average by **-19.87%** and **-21.17%** and the F1 scores improve by **+5.65%**, **+0.68%** respectively.

Compared to recent approaches such as LeReT Hsu et al. [2024], which evaluate performance on HotPotQA, FRUGALRAG achieves higher recall (**79.62** vs. **77.10**) with only a marginal increase of +0.96 in the average number of retrieval steps (See Tables 4, 3 in Appendix). Additionally, despite relying on an off-the-shelf answer generator g, FRUGALRAG outperforms reasoning-based baselines like Search-R1 Jin et al. [2025] on Exact Match in 2 out of 3 datasets. The lower performance on 2Wiki compared to Search-R1 is primarily due to the use of an off-the-shelf answer generator, whereas Search-R1 employs a trained model for answer generation.

Overall, our results highlight that our method strikes a significantly better efficiency-accuracy tradeoff, compared to strong baselines.

4 Conclusions

In this work, we argue that efficiency is an equally important metric to study in RAG solutions, besides the traditional RAG metrics such as retrieval performance and accuracy of the generated answers. We demonstrate that simple ReAct baseline that iteratively retrieves (by invoking the search tool) and reasons (to decide what search call to issue next) is quite competitive, especially if we can optimize its few-shot prompt using just tens of training examples. We propose a two-stage framework FrugalRAG that a) works with 1000 training examples, compared to state-of-the-art RAG techniques that use over 100,000 examples, and b) yet achieves competitive accuracies while also using far fewer search queries at inference time, on popular multi-hop QA datasets. The limitations and future work are discussed in the Appendix H.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*, 2024.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Sheryl Hsu, Omar Khattab, Chelsea Finn, and Archit Sharma. Grounding by trying: Llms with reinforcement learning-enhanced retrieval. *arXiv preprint arXiv:2410.23214*, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks, 2021. URL `https://arxiv.org/abs/2009.02252`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*, 2021.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*, 2023.

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

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the information needs of large language models, 2024. URL `https://arxiv.org/abs/2403.10081`.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022a.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022b.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. *arXiv preprint arXiv:2501.14342*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation, 2024. URL `https://arxiv.org/abs/2406.19215`.

5 Appendix

A Related Work

Which metric to optimize. Multi-hop QA involves two sub-tasks: retrieving relevant documents, and then answering the question based on the documents. Some methods report document retrieval-specific metrics such as recall [Hsu et al., 2024] whereas others report final answer metrics such as exact match [Jin et al., 2025]. Typically, a model is trained to optimize a particular metric (such as recall) and also evaluated on the same metric. For robustness, in this work we train on the recall metric and test on all metrics, including final answer metrics.

Prompting-based RAG approaches. With the recent advancements in the capabilities of large API-based LMs, some works explored prompting to call external search/retrievers at inference. Toolformer[Schick et al., 2023] uses a self-supervised objective to train an external model that decides to call tools (like Bing and Google search engines). ReAct [Yao et al., 2023] is another powerful prompting technique that allows the model to structure its outputs as thoughts, actions and observations, yielding significant improvements in the ability of LLMs to interact with external environments. Trivedi et al. [2022a] proposed IRCoT, another prompting strategy that alternates between chain-of-thought [Wei et al., 2022] steps and gathering evidence through retrievals. By using the intermediate traces, the IRCoT is able to decide what to retrieve by issuing the right search queries. Iter-RetGen [Shao et al., 2023] improves evidence gathering in multi-hop scenarios by combining retrieval and generation iteratively, such that a model's response is incorporated in the reasoning trace. However, both IRCoT [Trivedi et al., 2022a] and Iter-RetGen [Shao et al., 2023] rely on a fixed or predefined number of retrieval loops at inference, offering limited control over latency.

Finetuning-based techniques. A prevalent method for multi-hop QA using small LMs is supervised finetuning using reasoning traces from a large LM such as GPT-4 [Asai et al., 2023, Chan et al., 2024]. Other methods are trained to predict the next query to be retrieved [Chan et al., 2024]. Methods that scale the test-time compute that infer using multiple trajectories have also been proposed [Wang et al., 2025]. Recently, reinforcement learning-based techniques have been proposed that develop a reward based on outputting the ground-truth answer [Jin et al., 2025]. However, none of the techniques focus on efficiency of the solution. In fact, in Search-R1, the goal of RL is to increase the number of searches. Instead, we use RL to *decrease* the average number of searches done by our model.

1. **Traditional RAG approaches.** Early work in grounding generation with real world documents focused on end-to-end differentiable encoder-decoder pipeline REALM [Guu et al., 2020], which augments Masked-Language Modeling (MLM) with a latent retriever model, backpropagating through retrieval to learn both retriever and generator jointly. However, this approach incurs significant computational cost and has only been shown to work with relatively smaller models like T5 [Raffel et al., 2020]. Building on this, Lewis et al. [2020] proposed a general finetuning strategy, RAG-Token which demonstrated that join-training outperforms fixed dense retrieval and BM25.
2. **RL-based Retrieval Augmented Generation.** Recently, framing search query as an RL problem has received attention. LeReT [Hsu et al., 2024] performs preference optimization using diverse few shot prompts leveraging hundred-thousands of ground truth annotated documents. However, LeReT utilizes a fixed amount of compute per instance during inference and cannot be readily generalized to variable-hop scenarios. Similarly, concurrent works, Jin et al. [2025] and Chen et al. [2025] propose end-to-end RL-based optimization that only leverages the final answer annotation. These methods show that RL can effectively be used to teach the search query generator model *to issue more search queries* for multi-hop problems *without considering latency*. Our two-stage RL framework, by contrast, first explores without RL to maximize recall and then learns to stop at test time using RL.

B Detailed Methodology

Our overall framework is presented in Algorithm 1. Next, we describe FrugalRAG in detail.

Following the setup described in Sec 2, let $\mathcal{D}_0$ denote the initial context, either empty or initialized as $\mathcal{R}(Q)$. At hop h, the model has access to the context $Q \cup \bigcup_0^{h-1}\{\mathcal{D}_h, T_h, A_h, S_h\}$. This includes the original query Q, all previously retrieved documents, and the previously generated thought, action, search query triplets.

The process continues until the model outputs a special FINISH action, terminating after h_{term} hops (or at the budget limit B). At this point, a separate generator LM g is invoked to produce the final answer, conditioned on the original question Q and the full context accumulated up to termination. The central challenge for f is to iteratively construct highly targeted queries S_h such that the retriever $\mathcal{R}$ can surface a minimal yet sufficient set of documents to answer Q within the hop budget B.

FRUGALRAG requires access **only** to ground truth documents Y during training. It does **not** require supervision in the form of final answer annotations. These documents are used to provide fine-grained feedback signals. At inference time, FRUGALRAG relies solely on the input question Q, the document index $\mathcal{I}$, and a trained retriever $\mathcal{R}$.

Our key observation is that, with sufficient test-time compute, even base models can generate multiple, diverse search queries to address a question—following the ReAct paradigm (e.g., see Sec 3, Table 1). Therefore, the goal of our work is not to *scale up* test-time computation, as argued in prior work [Jin et al., 2025, Chen et al., 2025], but rather to *adaptively control* it based on the difficulty of each question. In the following subsections, we describe the two stages of our framework:

B.1 Stage 1: Evidence Coverage Maximization (Explore)

Gathering evidence plays a crucial role in answering multi-hop questions, which often require iterative retrieval and reasoning across multiple sources. Drawing on recent advances in test-time scaling, we observe that we can boost evidence coverage (i.e., recall) simply by letting the model f issue multiple search queries S_h at inference time. This approach sidesteps the need for massive supervised fine-tuning—instead, it harnesses the model's own generated rollouts to gather, and then integrate additional information. In the next section, we describe how we construct our training data and design our fine-tuning protocol to fully leverage this capability.

Training Dataset Generation. We choose ReAct [Yao et al., 2023] for generating rollouts. Here, a rollout is a set of outputs generated by the model f. In the standard ReAct setup, an off-the-shelf model f generates a thought-action pair $(T_h A_h)$ at each hop $h \in [1, B]$, where A_h is either a call to the retriever $\mathcal{R}$ or FINISH indicating the end of rollout. At each hop h, we generate samples $\{(T_h^1, A_h^1, S_h^1) \dots (T_h^n, A_h^n, S_h^n)\}$ using n bootstrapped prompts [Khattab et al., 2023] (See Appendix E). For each search query $S_h^i, i \in [1, n]$ we retrieve corresponding documents $\mathcal{D}_h^i = \mathcal{R}(S_h^i)$, then discard any documents already present in the context. We then compute recall against ground-truth labels and add the sample i that achieves the highest recall to the context for the next hop $h + 1$. This dataset generation strategy is simple and easily parallelizable. We conduct two separate runs – the standard ReAct framework where f is allowed to generate FINISH, and the second where f can only call the retriever. Although the former is more efficient and finishes before B search hops, we observe that the latter yields a significantly higher overall recall owing to a greater number of retrievals. Unlike previous work [Chan et al., 2024, Asai et al., 2023, Hsu et al., 2024, Jin et al., 2025] that generated orders of magnitude more data, we only generate 1000 examples during this step.

Supervised "Exploration" Finetuning (FRUGALRAG-Explore). Although the base model f without FINISH maximizes exploration, we cannot use it directly for reinforcement learning because it does not include FINISH. Consequently, during fine-tuning, we sample rollouts from both configuration described above, 90% without FINISH and 10% with it. We want to use supervised finetuning to build a strong base-policy for RL, that prioritizes exploration while ensuring that FINISH remains in the model's generation distribution. Hence, we finetune the model f to obtain our base policy f_S. At each iteration, the model predicts the next (T_h, A_h, S_h) tuple given the rollout which comprises interleaved thought-action-search tuples and retrieved documents represented as an ordered set $\{(\mathcal{D}_0, T_0, A_0, S_0), (\mathcal{D}_1, T_1, A_1, S_1) \dots (\mathcal{D}_{(h-1)}, T_{(h-1)}, A_{(h-1)}, S_{h-1})\}$ till $h - 1$, using standard cross-entropy error as the objective function. f_S has several notable advantages – **(1)** *off-the-shelf model f does not explore.* Despite prompt optimization, f is generally over-confident and predicts the answer without sufficient exploration. **(2)** *removing* FINISH *results in over-retrievals.* We observe that simply removing FINISH from the ReAct loop yields high recall even with the base model f, however, the model is forced to utilize the full budget for every question and cannot be post-trained for efficiency as it never generates a rollout with FINISH.

B.2 Stage 2: Controlling test-time compute with RL

Given a finetuned base policy model f_S, we propose a strategy that enables the model to generate extended rollouts only when required. This mechanism is crucial for inference-time efficiency, as it allows the model to adaptively determine the appropriate rollout length. Since f_S generally prioritizes exploration, our only goal is to learn when to sufficient evidence has been gathered, thereby reducing overall search latency during inference. Below we show how this problem can be formulated as a reinforcement learning task, as it requires evaluating and comparing different rollouts. However, unlike recent work using RL for scaling the number of retrievals [Jin et al., 2025], here our focus is to use RL to reduce the number of retrievals per question.

Reward Design. Our reward function is designed to guide the model toward discovering the optimal rollout length. To compute the reward, we first generate the complete rollout using the policy f_S and then evaluate it. Let h^* denote the optimal number of retrieval steps (or hops), defined as the point beyond which further retrievals do not improve the overall recall c. If the model terminates at a hop $h_{\text{term}} > h^*$, it incurs a penalty for redundant steps. This reward structure enables the model to explore adequately for complex queries while avoiding unnecessary computation for simpler ones. The reward equation is presented in Eq. 1.

Our reward penalizes deviations from the optimal stopping point in proportion to $|\Delta|$. The closer the model stops to h^*, the higher the reward, with the maximum attained when $h_{\text{term}} = h^*$. In addition to $\mathbf{R}$, we define a format reward $\mathbf{R_f}$ to enforce adherence to the ReAct-style format. If the output deviates from the expected format and results in failed retrievals, we assign a reward of -1; conversely, if the retrievals succeed, the model receives a reward of $+1$. This reward is averaged across all hops. The final reward is the mean of the main reward $\mathbf{R}$ and the format reward $\mathbf{R_f}$, yielding an overall reward in the range $[-R_{\max} - 1,\ R_{\max} + \alpha + 1]$.

Optimal Rollout Length. We define the optimal rollout length h^* as the minimum number of retrieval steps required to answer a query Q effectively. Any additional retrievals beyond h^* are considered redundant. Conversely, if the rollout has not yet gathered sufficient evidence to answer the query, it should continue retrieving. To balance retrieval efficiency and performance, we use FRUGALRAG-Explore as a reference policy, assuming it represents the best achievable performance within a fixed budget B. If another policy achieves the same recall performance as FRUGALRAG-Explore in fewer than B hops, we set h^* to that lower number of hops and compute the reward using Eq 1.

Optimization. Motivated by the recent success and memory efficiency of GRPO [Shao et al., 2024], we adopt it as our optimization algorithm. At each hop h, we sample v tuples $\{T_h^i, A_h^i, S_h^i\}_{i=1}^v$, and retrieve their corresponding documents $\mathcal{D}_h^i$, and deduplicate. We repeat this until we reach the maximum budget $h = B$, collecting sample tuples and documents. For each rollout i, we then compute a cumulative reward using Eq. 1, and backpropagate through every logit produce by the policy along that rollout. Finally, once f_S emits the FINISH token, we mask any further generations. The detailed steps are provided in Algorithm 1.

C Experimental Setup

Benchmarks. We conduct evaluation of FRUGALRAG using three widely adopted Multi-Hop RAG benchmarks– HotPotQA [Yang et al., 2018], 2WikiMultiHopQA [Ho et al., 2020] (2Wiki), and MuSiQue [Trivedi et al., 2022b], under their full-wiki setting using a ColBERT-v2 [Santhanam et al., 2021] retriever index over Wikipedia passages provided in official datasets. These datasets require models to perform reasoning across multiple documents to arrive at an answer. The **HotPotQA** benchmark comprises of of 7405 development examples requiring supporting evidence from Wikipedia abstracts. We report the results on the entire dev-set to access the models ability to perform end-to-end reasoning and retrieval. For the **2WikiMultiHopQA** benchmark, we utilize 12576 development examples, each coupled with its supporting evidence and document title. This evaluates a model's ability to hop between structured and unstructured wikipedia text. Finally, for **MuSiQue**, we test on its 2405 development examples of 2-4 hop questions derived from a composition of two-hop queries. We provide details of the datasets and the index in Appendix D.

Metrics. Most prior work focuses on the final answer accuracy [Jin et al., 2025] or document-level recall [Hsu et al., 2024]. In this work, we evaluate FRUGALRAG on three key parameters, namely, goodness of final answer, retrieval recall, and efficiency. To assess the final answer fidelity we report F1 Score, EM (Exact Match), and Match Score. **F1** score is the word level harmonic mean of precision and recall between predicted and ground truth string. **EM** requires the predicted string to exactly match the ground truth. **Match** requires the ground truth to be a substring of the generated answer. We also evaluate the alignment of supporting evidence with the retrieved documents by reporting **Recall** and **Support-F1** [Trivedi et al., 2022b]. Finally, we compute the **Latency** of FRUGALRAG as the total number of search operations it performs to answer a given query. In Sec 3, we introduce metric to realize the tradeoff between efficiency and performance (answer/retrieval), computed as the average performance per search. A detailed overview of metrics is presented in the Appendix F.

Baselines. We evaluate FRUGALRAG by comparing it against no-retrieval and retrieval-based baselines. The no-retrieval baselines include naive generation and chain-of-thought (CoT) prompting [Wei et al., 2022]. The retrieval-based baselines are vanilla retrieval-augmented generation (RAG) with CoT, ReAct [Yao et al., 2023], and optimized few-shot prompting with ReAct [Khattab et al., 2023]. We also compare FRUGALRAG to recently proposed approaches that leverage large-scale fine-tuning, including Search-R1 [Jin et al., 2025], LeReT [Hsu et al., 2024], IR-COT [Trivedi et al., 2022a], and CoRAG [Wang et al., 2025]. However, we note that these recent methods do not report all metrics that we consider and their metrics may not be directly comparable (due to varying model sizes and retrievers). Therefore, we conduct comparisons using the closest model scales and overlapping subsets of evaluation metrics.

Training. FRUGALRAG is model agnostic – we train Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Meta-Llama-3.1-8B-Instruct using our two-stage framework. In both the supervised finetuning and reinforcement learning (RL) stages, we leverage the TRL library [von Werra et al., 2020], and our ReAct pipeline with prompt bootstrapping is built on DsPy [Khattab et al., 2023]. For each dataset, we prepare the Stage 1 finetuning dataset with 1000 randomly sampled examples from the corresponding training split. Stage 1 (Sec. B.1) consists of full-parameter finetuning for a single epoch, using a learning rate of 2×10^{-5} and a weight decay of 0.01 for all models and datasets. We choose a maximum sequence length of 4096 during finetuning. In Stage 2 (Sec. B.2), we further train the finetuned models via GRPO [Shao et al., 2024]. For a detailed list of hyperparameters refer Appendix E.

D Dataset and Retrieval Index

We use the pre-processed Wikipedia abstracts index [1] provided by ColBERTv2 [Santhanam et al., 2021] for all our experiments on HotPotQA [Yang et al., 2018]. For each instance, we retrieve the top 3 documents and their titles and perform a maximum 6 retrievals. HotPotQA annotations consists of document title and evidence sentences which are used to compute the Recall and Supporting Document F1 respectively.

Since 2WikiMultiHopQA [Ho et al., 2020] and MuSiQue [Trivedi et al., 2022b] datasets are created using both the body and abstract of wikipedia articles we use the pre-processed dump of Wikipedia provided by Karpukhin et al. [2020] and index it using ColBERTv2 [Santhanam et al., 2021]. The generated index consists of 21M passages. For each instance, we retrieve top 5 documents and append it to our context. For experiments in Table 5, we use E5-Large provided by CoRAG Wang et al. [2025] indexed on KILT Petroni et al. [2021] which consists of 36 million passages, and retrieve top 5 documents for all datasets.

E Training Details

Algorithm 1 shows the overall training framework of FRUGALRAG. We plan to publicly release our code soon and have attached a copy of the codebase for review in the meantime. Below, we discuss each step along with their implementation details.

Few-Shot Prompt Optimization Details. We leverage DSPy [Khattab et al., 2023] for automatic few-shot prompt generation following LeReT [Hsu et al., 2024]. Specifically, we use 50 training examples ($\mathcal{L}_{\text{init}}$) with the BOOTSTRAPFEWSHOTWITHRANDOMSEARCH method, which uses the

[1] `https://downloads.cs.stanford.edu/nlp/data/colbert/baleen/wiki.abstracts.2017.tar.gz`

LM f to generate few-shot examples, selecting the best performing ones for subsequent prompting. We select 4 best performing few-shot prompts from a total of 15 candidate sets using the sum of answer EM and answer passage match. Answer EM checks for an exact string-match between the generated and actual answer, and passage match checks if the actual answer is present in the retrieved passages. This step is crucial because it facilitates dataset generation using diverse rollouts and ensures the answer format is followed by the model. For this step, we serve our model on one GPU using VLLM [Kwon et al., 2023]. For all experiments involving Qwen2.5, we utilize the 7B-Instruct variant for prompt optimization. The optimized prompts are then reused without modification for the 3B variant.

Dataset Generation Details. For each few-shot prompt p_i, the model f generates a tuple (T_h^i, A_h^i, S_h^i) representing a candidate output for the next hop. As described in Sec. B.1, we evaluate all candidate tuples at hop h and select one with the highest recall. This selected candidate is then used as the context for the next hop and the process is repeated till budget B (optionally till the selected candidate action A_h indicates FINISH). We set the budget $B = 6$, where the initial retrieval step is always $\mathcal{R}(Q^{(j)})$ with $Q^{(j)}$ denoting the original user utterance. The generated dataset is denoted by $\mathbf{D}$. For all experiments involving Qwen2.5, we utilize the 7B-Instruct variant along with its prompts to generate the dataset. For further improving results, we can repeat few shot prompt optimization and dataset generation using different base models.

Supervised "Explore" Finetuning Details. We use the standard next token prediction loss given by:

$$\max_f \mathbb{E}_{(x,y)\sim\mathbf{D}} \log p_f(x|y) \tag{2}$$

where $y = (T_h, A_h, S_h)$ and $x = Q^{(j)} \cup \{T_k, A_k, S_k, \mathcal{D}_k\}_{k=0}^{h-1}$ sampled from the generated dataset $\mathbf{D}$.

We train the model f for 1 epoch using a batch size of 4 and apply gradient accumulation of 2 steps, resulting in an effective batch size of 8. Optimization is performed using AdamW [Loshchilov and Hutter, 2017] with a learning rate of 2×10^{-5}. We use a linear learning rate scheduler with a warmup phase of 20 steps. The training is performed using 8 H100 80GB GPUs.

Controlling test-time compute with RL. Our RL step employs GRPO for fine-tuning the base policy f_S. Specifically, following the notation in DeepSeekMath [Shao et al., 2024], for each question $Q^{(j)}$, we sample a group of outputs $\{o_h^1, o_h^2, \ldots, o_h^v\}$ at hop h, where v is set to 8. We optimize our base policy f_S using the standard GRPO objective using the cumulative rollout reward as defined in Eq. 1. We use a KL divergence penalty with weight 0.1 since we have a trained base policy, and set the maximum reward $R_{\max} = 2.0$ for stability. Generation is limited to a maximum of 256 completion tokens and the maximum prompt size is 1024. Training is conducted using DeepSpeed-Zero2 [Rasley et al., 2020] and 7 H100 GPUs (where 1 is exclusively reserved for sampling). We set the learning rate to 10^{-6}. Due to the long prompt (which includes retrieved documents from previous hops), we use a total batch size of 48. We train FRUGALRAG for 400 steps across datasets and models, and report the performance using the final checkpoint.

F Metrics

F1. is the harmonic mean of the precision and recall measured at the word-level, and is given by

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{TP}_{\text{ans}}}{\mathrm{TP}_{\text{ans}} + 0.5 * (\mathrm{FP}_{\text{ans}} + \mathrm{FN}_{\text{ans}})} \tag{3}$$

where TP_{ans} denotes correctly predicted words, FP_{ans} represents the extra words, and FN_{ans} are the missing words in the generated answer.

Exact Match. is the exact match between the normalized generated answer string and normalized ground truth answer. It is 100% if the two strings match exactly, and 0 otherwise.

Match Score. measures the accuracy of generated answer by checking if the ground truth answer string is present in the generated answer string. It is 100% if the ground truth string is in the generated answer, and 0 otherwise.

Recall. is a retrieval metric, that measures the percentage of ground truth documents retrieved by the model. It is given by the ratio of correctly retrieved document titles TP_{doc} and the total number of

Algorithm 1: Our novel two-stage framework, FRUGALRAG consists of **(1)** *Dataset Generation* and *Supervised "Explore" Finetuning*, and **(2)** *Controlling test-time compute wth RL.*

Input: Labeled dataset $\mathcal{L} = \{(Q^{(j)}, Y^{(j)})\}_{j=1}^{1000}$, $\mathcal{L}_{\text{init}} = \{(Q^{(j)}, Y^{(j)})\}_{j=1}^{50}$, retriever $\mathcal{R}$, base LM f, budget B, max hops m, number of samples v

`// Prompt Optimization`
Perform prompt optimization on f using $\mathcal{L}_{\text{init}}$ to obtain few-shot prompts $\{p_1, \dots, p_n\}$;

`// Dataset Generation`
Initialize finetuning dataset: $\mathbf{D} \leftarrow []$;
for $Q^{(j)}, Y^{(j)}$ *in* $\mathcal{L}$ **do**
    Initialize buffer: `main_rollout` $\leftarrow []$;
    Initialize $\mathcal{D}_0 \leftarrow \mathcal{R}(Q^{(j)})$ or $\emptyset$;
    Initialize T_0, A_0;
    $\mathcal{H}_0 \leftarrow \{Q^{(j)}, T_0, A_0, \mathcal{D}_0\}$ `// stores previous context`
    Append $\mathcal{H}_0$ to `main_rollout`;
    for $h = 1$ *to* m **do**
        for i *in* $1 \dots n$ **do**
            for $h = 1$ *to* B **do**
                $(T_h^i, A_h^i, S_h^i) \leftarrow f(\mathcal{H}_{h-1}^i; p_i)$;
                `// occurs in 10% of calls`
                if $A_h^i =$ FINISH **then**
                    break
                $\mathcal{D}_h^i \leftarrow \mathcal{R}(S_h^i)$;
                Remove duplicate retrievals from $\mathcal{D}_h^i$;
                $\mathcal{H}_h^i \leftarrow \mathcal{H}_{h-1}^i \cup \{T_h^i, A_h^i, S_h^i, \mathcal{D}_h^i\}$;
        Evaluate all $\{\mathcal{D}_h^i\}_{i=1}^n$ (recall against ground truth $Y^{(j)}$);
        Select best-performing trajectory $\mathcal{H}^*$;
        Append $\mathcal{H}^*$ to `main_rollout`;
    Append each hop from `main_rollout` to $\mathbf{D}$;

`// Stage 1: Supervised "Explore" Finetuning`
$f_S \leftarrow$ Fine-tune f on $\mathbf{D}$ using standard next-token prediction `// See Eq. 2`

`// Stage 2: Controlling test-time compute with RL`
for $Q^{(j)}, Y^{(j)}$ *in* $\mathcal{L}$ **do**
    for $h = 1$ *to* m **do**
        Generate v sample tuples $\{T_h^i, A_h^i, S_h^i, \mathcal{D}_h^i\}_{i=1}^v$ for hop h ;
    for $i = 1$ *to* v **do**
        Compute reward $R^i \leftarrow \mathbf{R}(\{\mathcal{D}_h^i\}_{h=1}^m, Y^{(j)}, f_S)$ `// See Eq. 1`
        Backpropagate loss on $\{T_h^i, A_h^i, S_h^i\}_{h=1}^m$ using R^i;

Table 3: Retriever-level metrics (Recall, Support F1, Search latency) across HotpotQA, 2Wiki-MultiHopQA, and MuSiQue. FRUGALRAG achieves the best tradeoff between performance and efficiency on all datasets using Llama3.1-8B-Instruct. Green (**bold**) represents the best score and blue (underlined) represents the second best score. "-" indicates results are not present in the respective papers.

Model	Method	HotpotQA			2Wiki			MuSiQue		
		Recall	Sup. F1	Search	Recall	Sup. F1	Search	Recall	Sup. F1	Search
Llama3.1-8B-Instruct	Naive	0	0	0	0	0	0	0	0	0
	CoT	0	0	0	0	0	0	0	0	0
	CoT+RAG (n=3)	58.56	66.29	1	36.44	49.49	1	22.29	25.61	1
	CoT+RAG (n=5)	63.23	69.97	1	36.44	49.48	1	22.82	25.55	1
	ReAct	73.04	79.91	3.83	44.11	56.39	4.37	30.03	29.00	4.82
	ReAct FS	77.44	83.03	4.79	48.83	60.04	5.46	33.74	30.75	5.48
	LeReT	77.10	-	2.0	-	-	-	-	-	-
	FRUGALRAG-Explore	**83.11**	**86.96**	5.96	51.79	**62.35**	6.0	**37.11**	**32.27**	5.99
	FRUGALRAG	79.62	84.47	2.96	**51.87**	62.13	5.38	33.90	30.86	4.31

Table 4: Answer-level metrics (F1, EM, Match, Search latency) across HotpotQA, 2WikiMultiHopQA, and MuSiQue. FRUGALRAG achieves the best tradeoff between performance and efficiency on all datasets using Llama3.1-8B-Instruct. Green represents the best score and blue represents the second best score. "-" indicates results not present in the paper.

Model	Method	HotpotQA				2Wiki				MuSiQue			
		F1	EM	Match	Search	F1	EM	Match	Search	F1	EM	Match	Search
Llama3.1-8B-Instruct	Naive	14.17	5.26	31.03	0	8.00	0.9	43.16	0	4.28	0.24	13.28	0
	CoT	27.23	15.11	32.39	0	18.73	9.58	33.60	0	12.10	4.33	11.93	0
	CoT+RAG (n=3)	42.30	28.93	45.92	1	21.54	13.37	32.88	1	14.35	7.01	14.41	1
	CoT+RAG (n=5)	42.54	28.59	47.26	1	21.49	13.32	32.98	1	14.51	7.26	14.41	1
	ReAct	37.39	22.09	49.72	3.83	23.22	11.15	38.62	4.37	12.50	5.83	15.11	4.83
	ReAct FS	55.38	42.57	50.29	4.79	35.74	28.18	38.76	5.46	18.73	11.21	15.80	5.48
	LeReT	-	**52.5**	-	2.0	-	-	-	-	-	-	-	-
	FRUGALRAG-Explore	60.58	46.67	55.21	5.96	**44.72**	**37.39**	**46.31**	6.0	**23.44**	**15.56**	**19.70**	5.99
	FRUGALRAG	**62.95**	49.54	**56.28**	2.96	42.78	35.30	44.46	5.38	21.23	13.00	17.14	4.31

ground truth document titles $\text{TP}_{\text{doc}} + \text{FN}_{\text{doc}}$. We measure recall following LeReT Hsu et al. [2024], using document titles. The document-level recall is given by –

$$\text{Recall} = \frac{\text{TP}_{\text{doc}}}{\text{TP}_{\text{doc}} + \text{FN}_{\text{doc}}} \tag{4}$$

Sup. F1. measures the word-level F1 score (3) between the ground truth evidence sentences and those retrieved from the documents. Following Trivedi et al. [2022b], we compute the average F1 score across the ground truth evidence sentences, comparing them with corresponding retrieved documents. The evidence sentences are are provided in all three datasets. Supporting Document F1 or Sup. F1 is serves as a more reliable metric for retrieval with 2WikiMultiHopQA and MuSiQue since it considers fine-grained evidence rather than just the document titles.

G Additional Analysis

In this section, we present a discussion on the various properties of FRUGALRAG and demonstrate its effectiveness through ablation studies. For our analysis, we choose HotPotQA (2 hops) and MuSiQue (upto 4 hops) datasets due to the managable size of their datasets.

Adaptive vs Fixed Budget Search. A simple way to make ReAct baselines more efficient is to limit their number of searches to a fixed budget. *Does variable compute (searches) really help in handling questions of varying difficulty?* To get an estimate of the query difficulty and search requirements, we look at the histogram of number of searches in the trajectories of FRUGALRAG-Explore up until when 100% Recall is attained. Fig. 1a illustrates that most additional searches beyond a certain point are redundant (Estimated Optimal Searches), and the utility of each additional query diminishes

Table 5: Retriever-level metrics (Recall, Support F1, Search latency) for HotpotQA, 2WikiMultiHopQA, and MuSiQue using **Llama-3.1-8B-Instruct**. Green denotes improvement and red denotes decrease, over the test-time scaling baseline CoRAG. For CoRAG, we choose the first 30 documents from the provided inference files.

Retriever, Index	Method	Dataset	Recall	Sup. F1	Search
E5, KILT	CoRAG [Wang et al., 2025]	HotpotQA	74.55	70.35	6.00
		2Wiki	69.73	66.72	6.00
		MuSiQue	43.25	34.41	6.00
E5, KILT	FRUGALRAG-Explore	HotpotQA	75.82	72.10	6.00
		2Wiki	83.77	78.05	5.99
		MuSiQue	47.53	35.30	6.00
E5, KILT	**FRUGALRAG**	HotpotQA	76.33 (1.78)	72.18 (1.91)	5.52 (0.48)
		2Wiki	84.75 (15.01)	78.55 (11.83)	5.67 (0.33)
		MuSiQue	47.39 (4.14)	35.37 (0.96)	5.16 (0.84)
ColBERT, Wiki	FRUGALRAG-Explore	HotpotQA	83.11	86.96	5.96
		2Wiki	51.79	62.35	6.00
		MuSiQue	37.11	32.27	5.99
ColBERT, Wiki	**FRUGALRAG**	HotpotQA	79.62	84.47	2.96
		2Wiki	51.87	62.13	5.38
		MuSiQue	33.90	30.86	4.31

exponentially, from the perspective of Recall. So, we can expect ReAct baselines with fixed (small) search budget to attain competitive performance.

(a) (b) (c)

Figure 1: (a) Frequency of number of searches (HotPotQA, Qwen2.5-3B-Instruct): FRUGALRAG estimates the optimal searches whereas FRUGALRAG-Explore uses a fixed budget. (b-c) FRUGALRAG achieves a higher F1 score compared to ReAct FS using different budgets while being more efficient. (Zoom in for a better view)

We investigate the utility of variable compute in Fig 1b with Qwen3.5-3B-Instruct and in Fig 1c with Qwen3.5-7B-Instruct. We consider the best baseline (ReAct FS), and give it a search budget B, and ensure that it does not generate FINISH till the entire budget B is consumed. This allows us to directly compare the accuracy of FRUGALRAG that uses variable budget per query with fixed-budget baselines. Using Qwen2.5-3B-Instruct, we find that FRUGALRAG consistently outperforms ReAct across all budgets $B = 2, 3, 4, 5, 6$ in terms of F1 score, while also being more efficient. On HotPotQA, FRUGALRAG achieves an F1 of **55.81** compared to ReAct FS ($B = 6$) at 53.17. Similarly, on MuSiQue, FRUGALRAG reaches **18.71**, slightly improving over ReAct's 18.84, but with fewer retrieval steps on average. This trend also holds for the larger Qwen2.5-7B-Instruct model. On HotPotQA, FRUGALRAG achieves a significantly higher F1 of **63.03**, compared to 57.59 from ReAct FS ($B = 6$), while requiring only around **2.5 searches** on average. On MuSiQue, FRUGALRAG reaches **24.78** (with roughly **4 avg searches**) compared to ReAct FS ($B = 6$) at 17.82, again demonstrating superior performance and efficient retrieval (Detailed Tables are provided in the Appendix).

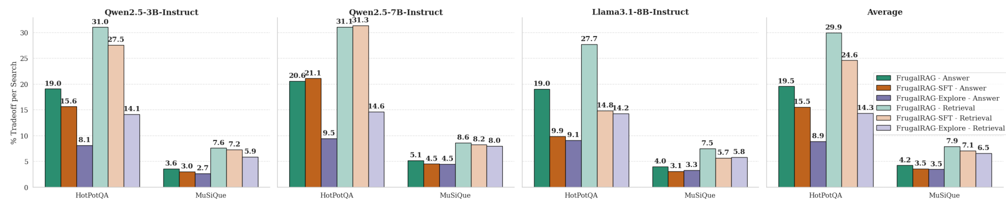

Figure 2: FRUGALRAG on average outperforms fixed budget and SFT based baselines, demonstrating the effectiveness of both Stage-1 finetuning and Stage-2 learning to control test time compute. (Zoom in for a better view). The plots show the % Tradeoff metrics defined in Section G.

Impact of RL based training. To capture the trade-off between answer quality and retrieval cost, we define two key metrics:

- **% Tradeoff (Answer)** quantifies how effectively each retrieval contributes to answer quality:

$$\%\text{Tradeoff}_{\text{Answer}} = 100 \times \frac{\text{F1} + \text{EM} + \text{Match}}{3 \times \text{Searches}}$$

- **% Tradeoff (Retrieval)** measures how well each retrieval contributes to evidence quality:

$$\%\text{Tradeoff}_{\text{Retrieval}} = 100 \times \frac{\text{Recall} + \text{Support F1}}{2 \times \text{Searches}}$$

These metrics allow us to jointly evaluate answer accuracy and evidence quality relative to the number of retrieval steps, enabling a comparison of efficiency across models and budgets.

To assess the impact of the RL-finetuning (Stage 2), we conduct an ablation study by comparing FRUGALRAG against a simple baseline, called FRUGALRAG-SFT. This baseline is trained for 1 epoch on the dataset generated during Stage 1, but with one key modification. Instead of sample 90% rollouts without FINISH, we sample all traces where the model outputs FINISH on its own.

In Fig. 2, we demonstrate that FRUGALRAG *on average across all models* is better on both $\text{Tradeoff}_{\text{Retrieval}}$ and $\text{Tradeoff}_{\text{Answer}}$. FRUGALRAG outperforms both FRUGALRAG-SFT and FRUGALRAG-Explore by a significant margin using Qwen2.5-3B-Instruct and Llama3.1-8B-Instruct on both the datasets as illustrated in Fig. 2. We note that the FRUGALRAG-SFT performance is comparable to FRUGALRAG with Qwen2.5-7B-Instruct.

Comparison with CoRAG. We compare FRUGALRAG against the recent state-of-the-art test-time scaling approach CoRAG [Wang et al., 2025]. CoRAG jointly trains the reasoning and the answer generation models on 100K multi-hop examples. In contrast, FRUGALRAG relies on an off-the-shelf answer generator, which limits its ability to match the exact answer formats expected by benchmark evaluation scripts. So, we adopt the following approach for fair comparison.

We replicate CoRAG's experimental setup using the E5-large retriever[2] and KILT Petroni et al. [2021] index. We fine-tune FRUGALRAG with Llama3.1-8B-Instruct separately on each dataset only using 1000 examples. For retrieval evaluation, we use the publicly released CoRAG outputs[3] to report Recall and Support F1 using all the retrieved documents, allowing us to compare the decomposition capabilities of FRUGALRAG and CoRAG. Concretely, we use the context document ids upto B=6 searches (30 documents) and extract the titles and passages to compute the recall and support F1 respectively.

In Table 5, we present retrieval-level metrics of CoRAG and FRUGALRAG variants on three datasets. The results clearly demonstrate that FRUGALRAG, when using the E5 retriever, significantly outperforms CoRAG (B=6) in both Recall and Support F1. We present a detailed list of hyperparamters for this run in Appendix E. This highlights the robustness and generalization ability of FRUGALRAG across different retrievers and index corpora. Notably, the number of retrieval queries is substantially higher on KILT [Petroni et al., 2021] compared to Wikipedia, which is expected given KILT's larger document collection (36 million vs. approximately 5 million for HotPotQA, 20 million for 2Wiki/MuSiQue using ColBERTv2 [Santhanam et al., 2021]).

[2] `https://huggingface.co/intfloat/e5-large-v2`
[3] `https://huggingface.co/datasets/corag/`

H Limitations and Future Work

Even though our method uses a small number of examples for training, it has some limitations in the analysis, and leaves room for future work in the following aspects: (a) in terms of generalization to new domains with no access to training data, and (b) in terms of coping with different types of retrievers (e.g., can we strike better efficiency-accuracy trade-off with a more powerful retriever?).