MINI-EFFICIENTRAG: A TINY REPRODUCTION OF AN EFFICIENT & ACCURATE RETRIEVAL AUGMENTED GENERATION SYSTEM

Ilia Hashemi Rad
Department of Electrical & Computer Engineering
Purdue University
West Lafayette, IN 47907, USA
ihashemi@purdue.edu

ABSTRACT

Retrieval-augmented generation (RAG) systems often struggle with multi-hop question answering, where a single retrieval step rarely surfaces all necessary evidence. EfficientRAG (Zhuang et al., 2024) addresses this by replacing per-hop large language model (LLM) query rewriting with lightweight token-classification modules (Labeler and Filter) that iteratively refine queries and filter irrelevant passages. In this TinyReproduction project, we adapt the official EfficientRAG pipeline to a single-GPU, notebook-based setting on the HotpotQA distractor benchmark, using Contriever for retrieval, DeBERTa-v3-based Labeler/Filter modules, and a locally served LLaMA-3-8B generator via vLLM. We reconstruct the synthetic data pipeline, train the Filter on the full released supervision and the Labeler on a reduced subset, and integrate the trained modules into an iterative RAG system. Compared to direct prompting, chain-of-thought prompting, naive one-shot RAG, and a SelfAsk-style baseline, our reproduced EfficientRAG retrieves fewer passages, attains higher retrieval recall, and achieves stronger end-to-end QA performance, while requiring only a single final LLM call per question. We also document the practical code patches and infrastructure changes needed to make EfficientRAG runnable and reproducible on a single A100 GPU, offering an educational reference for multi-hop RAG research.

1 INTRODUCTION

Large language models (LLMs) have recently achieved strong performance across a wide range of natural language tasks (OpenAI et al., 2024; Zhao et al., 2025; Touvron et al., 2023). However, they remain limited by the coverage and recency of their pre-training data and are prone to hallucinations, fluent but factually incorrect statements that undermine reliability in knowledge-intensive applications (Zhang et al., 2025; Huang et al., 2025). These issues are especially problematic when answers must be grounded in external sources such as up-to-date documents, long-tail facts, or domain-specific corpora.

Retrieval-augmented generation (RAG) mitigates these limitations by coupling LLMs with non-parametric memory: a retriever fetches relevant passages from an external corpus, and the generator conditions on the query plus retrieved evidence (Lewis et al., 2021; Gao et al., 2024). In its simplest form, RAG performs a single retrieval step using the original query, which works well when all necessary information is explicitly present and can be recovered with one semantic search. Many real-world questions, however, require multi-hop reasoning, where the model must integrate complementary facts from multiple documents. Datasets such as HotpotQA explicitly target this setting by requiring systems to retrieve and combine multiple supporting facts across Wikipedia articles (Yang et al., 2018). One-round RAG often fails here, and recent work therefore explores multi-round strategies that interleave reasoning and retrieval via question decomposition, query rewriting, or self-asking pipelines (Khattab et al., 2023; Ma et al., 2023; Press et al., 2023a; Shao et al., 2023), but these typically rely on repeated LLM calls at each hop, careful prompt engineering, and incur substantial latency and cost.

EfficientRAG proposes a different approach: replace LLM-based query rewriting with small learned controllers. It introduces a *Labeler*, a token-level classifier that identifies useful tokens in retrieved passages and decides whether the current evidence is sufficient to stop, and a *Filter*, which constructs the next-hop query by selectively incorporating newly discovered, informative tokens while discarding irrelevant content. Trained on synthetic supervision derived from LLM-generated decompositions, these modules allow the retriever to iteratively expand and refine queries without additional LLM calls at each hop, achieving competitive or better retrieval quality while reducing the number of retrieved passages and LLM invocations.

In this TinyReproduction project, we aim to validate the core claims of EfficientRAG at smaller scale and in a more accessible setting. We focus on the HotpotQA distractor split and re-implement the main components of EfficientRAG using Jupyter notebooks that run end-to-end on a single A100 GPU in Google Colab. To keep the pipeline classroom-friendly, we operate on reduced supervision: we train the Filter on the full released synthetic data, train the Labeler on roughly 14% of its synthetic examples, and adjust model sizes and data volumes to keep training within a few hours. Along the way, we adapt and patch the official implementation to support HotpotQA, integrate with a local vLLM (Kwon et al., 2023) server hosting LLaMA-3, and handle practical issues such as token-label alignment, LM output normalization, and GPU memory constraints.

Our goal is to reproduce, under these constraints, the qualitative efficiency trend reported in the original paper: that a learned Labeler/Filter pair can guide multi-hop retrieval to retrieve fewer passages per question for a given recall level, use approximately a single LLM call per question for final answer generation, and maintain competitive end-to-end answer correctness. We therefore compare our reproduced EfficientRAG pipeline against several baselines, including direct prompting, chain-of-thought prompting, a direct retrieval R@10 system, and a Self-Ask-style multi-hop baseline, and show on a small HotpotQA test subset that the EfficientRAG-style pipeline can achieve higher retrieval recall with fewer retrieved chunks and improved exact match and F1. In doing so, we provide (i) a single-GPU, notebook-based reimplementation of EfficientRAG on HotpotQA, (ii) a record of the engineering adaptations required to make the original codebase practical in this setting, and (iii) an empirical demonstration that the key efficiency–accuracy trade-offs of EfficientRAG can be observed even with substantially reduced training data for the Labeler.

2 METHODOLOGY

In this section we describe how we adapt EfficientRAG to a single-GPU, notebook-based TinyReproduction on HotpotQA. Our goal is to preserve the core idea of the original framework, learned Labeler and Filter modules that control multi-hop retrieval without per-hop LLM calls, while simplifying the engineering so that the whole pipeline can be run and inspected in a small set of Jupyter notebooks. An overview of the reproduced architecture is shown in Figure 1, where EfficientRAG replaces LLM-based query rewriting with lightweight token-classification heads and an iterative retrieval loop built on top of a dense retriever.

2.1 PROBLEM SETUP AND DATASET

We focus on open-domain multi-hop question answering over Wikipedia using the HotpotQA distractor setting. Each example consists of a multi-hop question, a set of candidate paragraphs (including supporting and distractor documents), and an answer span. HotpotQA also provides sentence-level supporting facts, which we use as weak supervision when constructing labels for the Labeler and Filter. To keep training and evaluation feasible on a single NVIDIA A100 (80 GB), we work with reduced splits: a training subset for learning the Labeler and Filter, a smaller validation subset for model selection and diagnostics, and an independent test subset for final evaluation. For the Filter we use the full released synthetic supervision for HotpotQA; for the Labeler we train on approximately 14% of the available synthetic data to respect runtime constraints.

2.2 EFFICIENTRAG ARCHITECTURE

Following Zhuang et al. (2024), our system augments a standard retrieval-augmented generation pipeline with two learned components: a Labeler and a Filter. Given a question q, a dense retriever first returns the top-k passages ("chunks") from the corpus. For each chunk c, the Labeler processes

Table 1: training and validation statistics for the Filter and Labeler on HotpotQA.

Model	**Train Loss (start → end)**	**Val Acc**	**Val F1**	**Runtime (s)**
Filter	$0.0556 \rightarrow 0.0300$	0.904	0.904	73.2
Labeler	$0.3450 \rightarrow 0.2020$	0.981 (token)	0.901 (token)	68.7

the concatenated sequence (q, c) and predicts (i) token-level useful vs. non-useful labels and (ii) a chunk-level tag $\langle$Continue$\rangle$ or $\langle$Terminate$\rangle$. Both predictions are produced by lightweight heads on top of a frozen encoder (DeBERTa-v3): one linear layer applied to token embeddings for token labels, and another applied to a pooled sequence representation for the chunk tag. Chunks tagged as $\langle$Continue$\rangle$ are added to an evidence pool, and their useful tokens are passed to the Filter.

The Filter is a separate encoder-plus-head model that takes as input the current question q together with useful tokens aggregated from previous hops. It assigns binary labels to these tokens and constructs the next-hop query q' by injecting the selected tokens into under-specified parts of q. Retrieval, labeling, and filtering are repeated until no chunks are tagged $\langle$Continue$\rangle$ or a hop limit is reached. The final evidence pool is then concatenated into a compact context and passed, together with q, to a locally served LLaMA-3-8B generator for a single answer-generation call.

2.3 Synthetic data construction

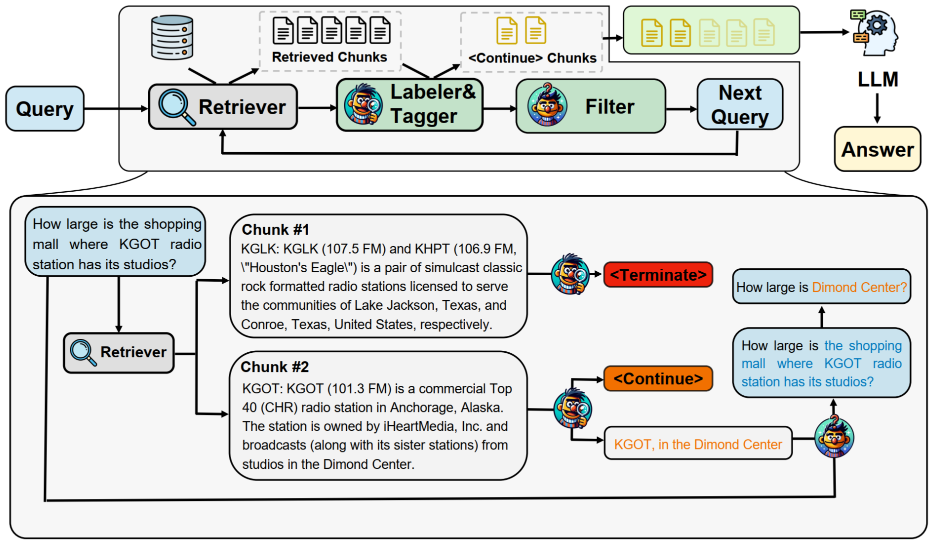
Training the Labeler and Filter requires supervision that is not directly provided by HotpotQA. We therefore reconstruct the synthetic data pipeline of EfficientRAG in a single notebook (`EfficientRAG.ipynb`), using a locally served LLaMA-3 model via vLLM. The pipeline has four stages: (1) multi-hop question decomposition, where the LLM rewrites each multi-hop question into a small set of single-hop sub-questions and their dependency structure; (2) token labeling, where the LLM highlights words in each supporting chunk that are crucial for answering the corresponding sub-question, from which we derive binary token labels; (3) next-hop question filtering, where the LLM generates next-hop queries that incorporate newly discovered entities or attributes, and we extract the minimal token changes between queries; and (4) negative sampling, where we retrieve hard negative chunks that are similar but not relevant and explicitly label them as $\langle$Terminate$\rangle$. The resulting synthetic instances provide token and chunk labels suitable for training both the Labeler and the Filter on HotpotQA.

2.4 Training details

We train the Labeler and Filter in `EfficientRAG_Training.ipynb` using cross-entropy losses. For the Labeler, we jointly optimize token-level and chunk-level classification; for the Filter, we optimize only the token-level head. Both models use a frozen DeBERTa-v3 encoder with small classification heads, a modest learning rate, a short warm-up schedule, and two training epochs. The Filter is trained on the full HotpotQA synthetic dataset, while the Labeler is trained on a subsample of about 14% of its synthetic instances to keep total training time around a few hours on a single A100. Training and validation statistics for both components are summarized in Table 1, showing that both models converge smoothly and reach strong validation performance despite the reduced training budget.

3 Experiments

We evaluate our TinyReproduction of EfficientRAG on the HotpotQA distractor split (Yang et al., 2018) using a single-GPU setup. Here we briefly describe the experimental environment, baselines, and evaluation metrics; additional implementation details (including repository patching, notebook structure, and dependency configuration) are provided in Appendix A.1. Quantitative results and analysis are presented in Section 4.



Retrieved Chunks
<Continue> Chunks
Query
Retriever
Labeler& Tagger
Filter
Next Query
LLM
Answer
How large is the shopping mall where KGOT radio station has its studios?
Retriever
Chunk #1
KGLK: KGLK (107.5 FM) and KHPT (106.9 FM, \"Houston's Eagle\") is a pair of simulcast classic rock formatted radio stations licensed to serve the communities of Lake Jackson, Texas, and Conroe, Texas, United States, respectively.
Chunk #2
KGOT: KGOT (101.3 FM) is a commercial Top 40 (CHR) radio station in Anchorage, Alaska. The station is owned by iHeartMedia, Inc. and broadcasts (along with its sister stations) from studios in the Dimond Center.
<Terminate>
<Continue>
KGOT, in the Dimond Center
How large is Dimond Center?
How large is the shopping mall where KGOT radio station has its studios?

Figure 1: overview of the EfficientRAG framework within an iterative RAG pipeline: retrieved chunks are tagged as ⟨continue⟩ or ⟨terminate⟩, useful tokens such as "KGOT in the Dimond Center" are preserved, and the filter constructs the next-hop query (e.g., "How large is Dimond Center?") from the original question and previously annotated tokens until all chunks are tagged ⟨terminate⟩ or the maximum number of iterations is reached.

3.1 Experimental Setup

All experiments are conducted in a Google Colab environment with one NVIDIA A100 GPU (80 GB). We use `facebook/contriever-msmarco` (Izacard et al., 2021) as the dense retriever to index Wikipedia paragraphs from the HotpotQA distractor setting, and fine-tune DeBERTa-v3-based Labeler and Filter modules on the synthetic supervision described in Section **??**. For answer generation, we run `meta-llama/Meta-Llama-3-8B-Instruct` via a local vLLM OpenAI-compatible server, which is also used during data synthesis. At test time, implemented in `EfficientRAG_Inference.ipynb`, we apply the trained Labeler and Filter in an iterative retrieval loop with top-$k = 10$ passages per hop, stopping when no chunks are tagged ⟨Continue⟩ or a hop limit is reached, and then call the generator once per question with the final evidence pool. Details of the Colab environment, model caching, and code patches needed to make the original repository work with LLaMA-3 and modern libraries are given in Appendix A.1.

3.2 Baselines

We compare our reproduced EfficientRAG system against four baselines that share the same retriever and generator wherever retrieval is involved. **Direct** prompts LLaMA-3-8B to answer the question without retrieval. **CoT** uses chain-of-thought prompting to encourage step-by-step reasoning. **Direct-R@10** is a one-shot RAG baseline where Contriever retrieves the top-10 passages and the generator conditions on them in a single pass. **SelfASK** is an iterative self-asking baseline inspired by Press et al. (2023b), where the model generates intermediate sub-questions and answers, each of which triggers an additional retrieval step. Our **EfficientRAG** configuration uses the same retriever and generator but replaces per-hop LLM query rewriting with the learned Labeler and Filter, as illustrated in Figure 1.

3.3 Evaluation Metrics

We evaluate systems along two axes: retrieval effectiveness and answer correctness. For retrieval, we report Recall@K with respect to the gold supporting paragraphs provided by HotpotQA and the average number of retrieved chunks per question K; for one-shot methods such as Direct-R@10,

Table 2: Retrieval performance on the HotpotQA test subset. We report Recall@K with respect to gold supporting paragraphs, and the average number of retrieved chunks per question (K).

Method	Recall@K	K (avg. chunks)
Direct-R@10	69.86	10.00
SelfASK	71.21	25.45
EfficientRAG (ours)	**79.74**	**7.29**

K is fixed by design, while for iterative methods such as SelfASK and EfficientRAG, K counts the total number of unique passages retrieved across all hops. These retrieval metrics are reported in Table 2. For end-to-end question answering, we follow the official HotpotQA evaluation and compute exact match (EM), token-level F1, and answer accuracy (Acc), using the standard normalization of answers; the QA metrics for all methods are summarized in Table 3.

4 Results

In this section, we report the performance of our TinyReproduction of EfficientRAG on the HotpotQA distractor split. We first summarize the dev-set behavior of the learned Labeler and Filter, then present retrieval results, and finally end-to-end question answering performance. All experiments use the single-GPU setup and baselines described in Section 3.

4.1 Training performance of Labeler and Filter

Table 1 summarizes the training and validation behavior of the Filter and Labeler on HotpotQA. Both models converge smoothly within two epochs, with the Filter reaching strong validation accuracy and F1 despite a relatively small training loss range, and the Labeler achieving high token-level accuracy and F1 even though it is trained on only about 14% of the available synthetic data. Notably, the Labeler's validation scores are comparable to those of the Filter while using substantially fewer training instances, suggesting that the synthetic token-level supervision is highly informative. The short runtimes for both components indicate that, under our TinyReproduction regime, training the controllers is a minor cost compared to running the full data-synthesis and inference pipelines.

4.2 Retrieval Performance

Table 2 reports retrieval performance on the HotpotQA test subset. Direct-R@10, a standard one-shot RAG baseline, retrieves a fixed ten chunks per question and achieves a Recall@K of 0.6986. The SelfASK baseline, which issues multiple follow-up questions and retrieval calls, slightly improves recall to 0.7121 but at the cost of reading many more passages on average (25.45 chunks per question). In contrast, our reproduced EfficientRAG pipeline achieves a substantially higher recall of 0.7974 while retrieving only 7.29 chunks per question on average. This confirms, in a compact single-GPU setting, the central efficiency trend reported in the original EfficientRAG paper: learned hop control and query refinement can focus retrieval on informative passages and avoid over-fetching distractors, yielding both higher recall and lower passage consumption compared to naive RAG and LLM-driven iterative baselines.

4.3 End-to-End Question Answering Performance

End-to-end QA results on the same HotpotQA test subset are summarized in Table 3. As expected, answering without retrieval is challenging: the Direct baseline obtains EM/F1/Acc of $0.2606/0.2920/0.2303$, reflecting the difficulty of multi-hop questions when the model must rely solely on its parametric knowledge. Chain-of-thought prompting (CoT) provides a modest improvement (EM/F1/Acc of $0.2788/0.3284/0.2667$) by encouraging more structured reasoning, but it still lacks access to explicit supporting evidence. Incorporating retrieval via Direct-R@10 leads to a more substantial gain, with EM/F1/Acc of $0.3688/0.4272/0.3688$, showing the benefit of grounding the answer in retrieved passages even without iterative refinement.

Table 3: End-to-end question answering performance on the HotpotQA test subset. We report exact match (EM), token-level F1, and answer accuracy (Acc).

Method	**EM**	**F1**	**Acc**
Direct	26.06	29.20	23.03
CoT	27.88	32.84	26.67
Direct-R@10	36.88	42.72	36.88
EfficientRAG (ours)	**52.29**	**57.73**	**50.98**

Our reproduced EfficientRAG system outperforms all of these baselines across all metrics, achieving EM/F1/Acc of $0.5229/0.5773/0.5098$. This is a large improvement over Direct-R@10 and SelfASK-style iterative prompting, despite using only a single final LLM call per question and fewer retrieved chunks on average. These results support the intuition that providing the generator with a smaller set of highly relevant passages, rather than a large, noisy context, can lead to better downstream QA performance, even when using a relatively small 8B LLaMA-3 model and a reduced amount of training data for the Labeler.

Taken together, the retrieval and QA results demonstrate that our TinyReproduction successfully recovers the qualitative behavior of EfficientRAG in a much smaller, notebook-friendly setting: it retrieves fewer chunks per question, attains higher recall, and delivers stronger end-to-end performance than multiple strong baselines that rely on either naive RAG or LLM-driven iterative reasoning alone.

5 Conclusion

In this work, we presented a TinyReproduction of EfficientRAG on the HotpotQA distractor benchmark, implemented entirely in a single-GPU, notebook-based setting. By reconstructing the synthetic data pipeline, fine-tuning DeBERTa-v3-based Labeler and Filter modules, and integrating them with a Contriever retriever and a locally served LLaMA-3-8B generator, we showed that the core ideas of EfficientRAG carry over to a much smaller and more constrained environment. Our experiments indicate that the reproduced system can retrieve fewer passages while achieving higher recall, and can improve end-to-end multi-hop QA performance compared to direct prompting, chain-of-thought prompting, naive one-shot RAG, and a SelfAsk-style baseline.

At the same time, our study highlights practical challenges in reproducing state-of-the-art RAG systems, including codebase fragility, dependency drift, and the cost of large-scale synthetic data generation. These constraints forced us to operate on reduced training subsets and a single dataset, which limits the generality of our conclusions. Nevertheless, we believe that the resulting Colab-friendly pipeline, along with the documented patches and design choices, provides a useful and accessible testbed for future work on efficient multi-hop RAG, including scaling to additional datasets, exploring alternative lightweight controllers, and studying robustness and fairness properties of retrieval-augmented LLMs.

References

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, January 2025. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2021. URL https://arxiv.org/abs/2112.09118.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023. URL https://arxiv.org/abs/2212.14024.

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, 2021. URL https://arxiv.org/abs/2005.11401.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.322. URL https://aclanthology.org/2023.emnlp-main.322/.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL `https://aclanthology.org/2023.findings-emnlp.378/`.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023b. URL `https://arxiv.org/abs/2210.03350`.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.620. URL `https://aclanthology.org/2023.findings-emnlp.620/`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL `https://aclanthology.org/D18-1259/`.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models, 2025. URL `https://arxiv.org/abs/2309.01219`.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and

Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Efficientrag: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3392–3411, 2024.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

This appendix provides additional details about the environment, code modifications, and notebook structure used in our TinyReproduction.

Environment and dependencies. We clone the official `efficientrag-official` repository in a Google Colab notebook and install dependencies required for dense retrieval, training, and local LLM serving, including `faiss-cpu`, `transformers`, `accelerate`, `deepspeed`, `vllm`, and `wandb`. Large model weights (Contriever, DeBERTa-v3, LLaMA-3) are cached in a local directory to avoid repeated downloads. We pin `transformers` and related libraries to compatible versions to ensure DeBERTa-v3 and training scripts run without breaking changes.

Code patching. Several patches are required to make the original repository work in this setting. We fix deprecated Trainer arguments (e.g., replacing `evaluation_strategy` with `eval_strategy`), update imports for modules such as `StableDropout` to match newer `transformers` layouts, and add missing HotpotQA-specific logic (normalization of contexts, prompt templates for query decomposition and next-hop construction). For local LLaMA-3 integration, we implement an OpenAI-compatible wrapper that routes all language-model calls to a vLLM server at `http://127.0.0.1:8000/v1`, and harden the response-parsing layer to handle minor format deviations (e.g., nested dictionaries or unexpected fields).

Notebook structure. We organize the pipeline into three main notebooks. `EfficientRAG.ipynb` handles synthetic data construction: it launches the vLLM server, normalizes HotpotQA records, runs multi-hop question decomposition, token labeling, next-hop query generation, and negative sampling, and produces the synthetic datasets for the Labeler and Filter. `EfficientRAG_Training.ipynb` downloads the released synthetic data for HotpotQA, applies training-related patches, and fine-tunes the Filter on the full dataset and the Labeler on a $\sim 14\%$ subset, saving checkpoints to persistent storage. `EfficientRAG_Inference.ipynb` rebuilds the corpus index with Contriever, loads the trained Labeler and Filter checkpoints, and executes the full iterative retrieval and QA pipeline on a held-out test subset, as used in our experiments.