# Lightweight Retrieval Tuning for Black-Box Language Models

**Xiao-Wen Yang, Hong-Jie You, Peng-Xiao Song, Hao-Ran Hao, Jie-Jing Shao, Yu-Feng Li***
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
yangxw@lamda.nju.edu.cn, yhj13777@gmail.com, songpx@lamda.nju.edu.cn
hhr277133291@gmail.com, shaojj@lamda.nju.edu.cn, liyf@nju.edu.cn

## Abstract

Retrieval-augmented language models have demonstrated remarkable effectiveness, particularly in knowledge-intensive tasks. Previous studies on retrieval augmentation typically require tuning the parameters of language models or updating the vector datastore, resulting in huge computational costs. However, it becomes infeasible as the scale of language models and the vector datastore continues to increase, especially when language models are only accessible through APIs. Hence, we treat the language model as a black box and keep the vector datastore frozen. We propose a lightweight retrieval tuning technique by introducing a self-adapted similarity matching module, employing less than 1M parameters. Proximal Policy Optimization (PPO) is utilized to fine-tune the introduced parameters because the black-box language models cannot be trained end-to-end. Our approach exhibits great scalability as it can be employed in any scenario, regardless of the frozen vector datastore and the black-box language model. Moreover, our approach has high training efficiency, the speed bottleneck of which lies in the inference of the black-box language models. Experiments conducted on the MMLU and TrivialQA benchmarks demonstrate that our lightweight retrieval tuning technique significantly improves the performance of retrieval augmentation across different scales and architectures of language models. Specifically, our method improves InstructGPT's performance on the MMLU benchmark by 6%.

## 1 Introduction

Large language models (LLMs) such as GPT-4[1] have made remarkable progress, which are now capable of achieving performance levels close to that of humans across a wide range of downstream tasks [2, 3, 4]. Although language models can leverage general knowledge to perform well on some tasks such as machine translation [5], existing work points out that due to their limited domain-specific knowledge, LLMs may suffer from hallucination [6, 7] in certain knowledge-intensive downstream tasks, such as open-domain question answering [8]. Fortunately, retrieval-augmented language models [9, 10, 11] effectively alleviate this problem by retrieving targeted knowledge or prompts from an external datastore.

However, previous approaches to retrieval augmentation require fine-tuning language models and dense retrievers using downstream data [9, 12]. This necessitates access to the complete parameters of the language model and the retriever and the ability to fine-tune with limited computational resources. Unfortunately, most existing LLMs that exhibit excellent performance such as GPT4 [1] are only accessible via API calls. Even for open-source language models, a single GPU is insufficient to support loading all the parameters (#>100B). Therefore, considering language models as black

*Corresponding author

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

boxes is worthy of exploration. In addition, constructing a large vector datastore by storing embeddings of the source data using a pre-trained dense retriever becomes a trend [13, 14]. To adapt the dense retriever for downstream tasks by fine-tuning, the vector datastore must be frequently updated, resulting in high computational costs. While several research [15, 16, 17] has focused on enhancing retrieval augmentation capabilities for downstream tasks using black-box language models, they ignore the high costs of updating the large-scale vector datastore. The left side of Figure 1 illustrates that previous studies suppose that both language models and retrieval models can be trained, with relatively small model parameters and external corpus sizes.

In this paper, we introduce a lightweight retrieval tuning technique (LRT) for black-box language models. Our approach does not require updating the whole retrieval model, thus preserving the reusability of the vector datastore. We argue that due to the distribution shift in downstream tasks, the embedding corresponding to the query may not effectively retrieve the most suitable external documents. Therefore, specifically, we apply a linear layer to the querys embedding to obtain a new embedding for document matching. The matched documents and query are then fed into the black-box language model to obtain the final prediction. By evaluating the prediction, we design a reward based on the evaluation metric, which serves as a learning guide for the learnable parameters. We optimize this reward using the Proximal Policy Optimization (PPO) reinforcement learning algorithm [18]. Figure 1 demonstrates the difference between our method and previous methods. The advantage of our proposal is that it be flexibly applied to different black box models and different forms of external vector datastore.

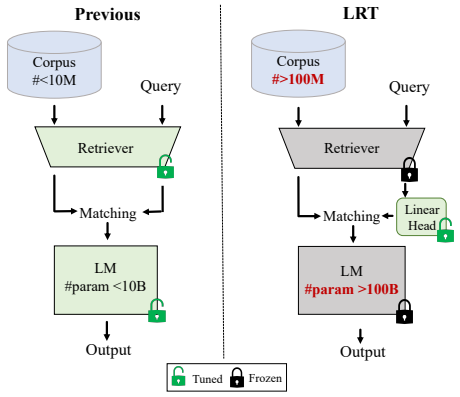


Figure 1: Comparison of our method LRT and previous methods. Previous work trains the retriever and the language model of small scale jointly. The LRT requires neither re-indexing the large vector datastore nor updating the parameters of the large language model, it solely involves training a linear layer, making it highly lightweight.

Our experiments show that LRT can improve the performance of black box language models on a series of downstream benchmarks. Specifically, LRT improves the performance of the InstructGPT by 6% over zero-shot on the MMLU dataset, using only 0.59M trainable parameters. The advantage of our method is that it does not need to fully fine-tune the pre-trained retriever in the case of the language model black box. According to our estimation, the number of parameters trained using our method is generally less than 1 million for most scenarios. This is considerably smaller than the parameter sizes of language models and retrieval models. Consequently, our method is more lightweight when compared to previous approaches.

## 2 Related Work

Retrieval-based methods to provide external knowledge to language models have been proved effective in a range of downstream knowledge-intensive tasks [9, 19, 11] and can effectively alleviate the language model's hallucination. Prior research often involves training retrievers and language models separately. KNN-LM [20] utilizes $k$-nearest neighbors retrieval on a set of tokens to derive token distributions, which are then ensembled with the language model's predictions. RETRO [10] uses a fixed retriever to only fine-tune language models with back-propagation. Some methods [19, 9, 11, 21] aim to update the retriever-LM system comprehensively by adopting an end-to-end approach. These methods all require obtaining parameters of the language model and then updating, which cannot be applied to black-box language models.

Recently, efforts have been made to address fine-tuning the retriever for the black-box language model. REPLUG [15] utilizes a frozen language model to provide probability on different documents, thereby supervising the retrieval model. AAR [16] employs FiDAtt scores to manually construct positive and negative documents for a given query and then adjusts the retriever through

contrastive loss. Some [17] employing in-context learning to select retrieval documents also make progress. Due to the requirement of adjusting the retrieval encoder in these methods, it becomes necessary to periodically update the whole datastore which incurs a significant computational cost.

## 3 Lightweight Retrieval Tuning

Retrieval-augmented language models utilize a retriever to retrieve effective prompts from external corpus to improve the generalization performance of language models. Specifically, we have an external corpus, denoted as $\mathcal{D} = \{d_1, ..., d_m\}$, and a query $q$ to the language model. Following previous work, we use a dense retriever $E$ to encode each document and the query. Then the similarity between the query and the document embedding is computed by the dot product, $\text{Sim}(q, d) = E(q)^{\mathrm{T}} \cdot E(d)$. The top-$k$ documents with the highest similarity score are retrieved and concatenated with the query to form the input of LM: $x = \text{concat}(d_{i_1}, ..., d_{i_k}, q)$. We eliminate the need for on-the-fly computation during retrieval by precomputing the embeddings of documents beforehand. To ensure the retrieval model effectively retrieves optimal documents, it is crucial to obtain high-quality embeddings. In this regard, we employ Contriever [12] which achieves desirable embedding distribution through contrastive pre-training. However, given that downstream tasks exhibit shifted distributions and various large models possess distinct prompt preferences, it becomes imperative to adapt the retrieval process for specific tasks.

Due to the limitations of black-box language models, updating the language model specifically for downstream tasks is not always feasible. Some methods address this issue by fine-tuning the retrieval model using data from downstream tasks. However, updating the embeddings of documents frequently can be costly due to the large size of the external corpus. Additionally, updating the vector datastore can increase the generalization risk, resulting in performance degradation. To overcome these challenges, we introduce a lightweight retrieval tuning technique (LRT).

Formally, we define the dataset for the downstream task $S = \{(q_1, y_1), (q_2, y_2), ..., (q_n, y_n)\}$. We have a black-box language model LM and a dense retriever $E$. The output probability is computed from total probability law, $p(y \mid q) = \sum_{d \in \mathcal{D}} p(y \mid \text{concat}(d, q)) \cdot \lambda(d, q)$. Where $p(y \mid \text{concat}(d, q))$ denotes the output probability of the language models and $\lambda(d, q)$ is estimated by the dense retriever, $\lambda(d, q) = \frac{e^{E(q)^{\mathrm{T}} E(d)/\tau}}{\sum_{d' \in \mathcal{D}} e^{E(q)^{\mathrm{T}} E(d')/\tau}}$. We select the top-$k$ documents with high similarity scores and denote them as $\mathcal{D}'$ to replace $\mathcal{D}$ due to the large size of the external corpus. $\tau$ is the temperature coefficient.

Given that the parameters of the language model and the dense retriever are unavailable, it is not possible to directly fine-tune $p$ and $\lambda$. Furthermore, we are unable to obtain the exact probability of the language model, as $\text{LM}(\cdot)$ merely represents a sampling result from this probability distribution. So we changed the form of $\lambda$ and introduced the learnable layer $f_\theta$.

$$\lambda_\theta = \frac{e^{f_\theta(E(q))^{\mathrm{T}} E(d)/\tau}}{\sum_{d' \in \mathcal{D}'} e^{f_\theta(E(q))^{\mathrm{T}} E(d')/\tau}} \tag{1}$$

We believe that the pre-trained dense retriever provides an excellent embedding space where different types of documents can be well distinguished. However, due to the distribution shift of downstream tasks, the embedding of the query is not able to be effectively aligned with the most suitable documents. Therefore, we only apply $f_\theta$ to the embedding of the query. To fully leverage the performance of the pre-trained retriever, we adopt a linear layer, i.e. $f_{W,b}(x) = Wx + b$. We initialize matrix $W$ as an identity matrix and $b$ as a zero vector. This guarantees that, in the initial training phase, retrieved documents are acquired from the pre-trained retriever, which serves as a reliable initialization. Additionally, it ensures that adjusted embedding does not deviate significantly from the original embedding.

After introducing the learnable layer, our objective is to perform maximum likelihood estimation on the training set. However, as the language model remains a black box, $p(y \mid q)$ remains intractable. Considering that the output of a language model is a sampling of $p(y \mid q)$, comparing it with the grounding labels provides an indication of the quality of the retrieved documents. As a result, we can treat the language model as a reward model and $\lambda_\theta$ as the policy model, using reinforcement learning algorithms to indirectly optimize our objective. We employed the Proximal Policy Optimization (PPO) for updating parameters. Detailed explanations can be found in the appendix.

Table 1: Results on MMLU and TriviaQA dataset for Flan-T5 and LLaMA-7B. The **bold** score means the best performance.

| Models | #Parameters | Methods | MMLU | | | | | TriviaQA | |
|---|---|---|---|---|---|---|---|---|---|
| | | | ALL | Hum. | Soc. Sci. | STEM | Other | F1 | EM |
| Flan-T5-Base [24] | 250M | Zero-shot | 36.3 | 38.2 | **40.9** | **29.7** | 39.0 | 8.8 | 5.0 |
| | | Retrieval | 35.5 | **38.9** | 39.9 | 27.0 | 39.7 | 30.7 | 23.9 |
| | | LRT(Ours) | **36.7** | 38.5 | 40.6 | **29.7** | **40.8** | **34.4** | **27.1** |
| Flan-T5-Large [24] | 780M | Zero-shot | 44.9 | 43.5 | 52.4 | 36.7 | 50.2 | 17.9 | 13.1 |
| | | Retrieval | 45.5 | 45.9 | **53.2** | 36.5 | 50.3 | 34.2 | 27.4 |
| | | LRT(Ours) | **45.8** | **46.3** | 52.5 | **37.0** | **50.7** | **38.4** | **31.0** |
| Flan-T5-XL [24] | 3B | Zero-shot | 51.0 | 54.9 | 57.6 | 36.3 | **60.6** | 31.2 | 26.4 |
| | | Retrieval | 50.9 | **55.2** | **58.0** | 36.7 | 59.2 | 37.8 | 31.5 |
| | | LRT(Ours) | **51.6** | **55.2** | 57.5 | **38.1** | **60.6** | **43.7** | **36.8** |
| LLaMA-7B [25] | 7B | Zero-shot | 30.4 | 31.1 | 27.8 | 28.6 | **34.4** | 61.6 | 52.1 |
| | | Retrieval | 31.2 | 28.5 | **34.6** | **29.5** | 32.9 | 63.8 | 53.9 |
| | | LRT(Ours) | **31.6** | **35.6** | 31.3 | 27.5 | 33.3 | **64.5** | **54.6** |

## 4 Experiments

Following prior work [15], we choose MMLU [22] and TriviaQA [23] as the downstream tasks. We consider language models of two architectures: encoder-decoder models and decoder-only models. For encoder-decoder models, we consider Flan-T5 [24] models due to their strong performance on a wide range of downstream tasks. As for the decoder-only models, we consider LLaMA-7B [25] and InstructGPT[26] (we use the text-davinci-002) as representatives. We compared the zero-shot and vanilla retrieval augmentation baselines with our LRT method. Since InstructGPT has longer input token lengths than other language models, we also compared the few-shot baseline under the few-shot setting (LRT w/ FS) on MMLU .

Table 2: Results for InstructGPT on MMLU

| Methods | MMLU | | | | |
|---|---|---|---|---|---|
| | ALL | Hum. | Soc.Sci. | STEM | Other |
| Zero-Shot | 59.5 | 65.9 | 68.0 | 44.0 | 66.3 |
| Few-Shot | 62.0 | 65.4 | 74.2 | 48.1 | 66.2 |
| Retrieval | 62.3 | 64.4 | 71.1 | 52.7 | 65.1 |
| LRT | 62.7 | 65.7 | 71.1 | 53.7 | 64.3 |
| LRT w/ FS | **65.5** | **69.5** | **75.7** | **54.6** | **67.1** |

Table 1 reports the results of our method LRT compared with the baselines on MMLU and TriviaQA for Flan-T5 and LLaMA-7B. We report the accuracy of 4 sub-domains and the average accuracy across all sub-domains (ALL) on MMLU. We report the EM (Exact Match) and F1 scores for TriviaQA. Our approach demonstrates a significant performance improvement compared to the zero-shot baseline and the retrieval baseline, particularly in the TriviaQA tasks. This highlights the effectiveness of our algorithm. Table 2 reports the results for InstructGPT on MMLU. Our approach achieves a 6% performance.

Our approach is extremely lightweight in terms of computational requirements. Assuming the embedding size of the datastore is noted as dim, and considering that we only have one linear layer as our learnable parameter, the total number of parameters is $\text{dim}(\text{dim}+1)$. For Contriever, its output dimension is 768, resulting in a total parameter count of 590,592, approximately 0.59M, which is negligible compared to the scales of language models. Our method only needs one forward pass, and the computational bottleneck is determined solely by the inferencing cost of the black-box language model. Furthermore, since we load the vector datastore before training, the index remains unchanged throughout the training process which saves huge computational costs. This significantly demonstrates that our approach outperforms other retrieval-augmentation methods.

## 5 Conclusion

In conclusion, this paper presents a novel approach for tuning retrieval-augmented language models that addresses the challenges of computational costs and scalability. Our proposed technique provides a practical and efficient solution for improving retrieval augmentation in knowledge-intensive tasks. It does not require adjusting the parameters of language models or updating the vector datastore, allowing for training with black-box language models. The experimental results on MMLU and TriviaQA demonstrate the effectiveness of our approach. Across different scales and various architectures of language models, our lightweight retrieval tuning technique significantly improves the performance of retrieval augmentation.

## References


[1] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[4] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[5] Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study. *CoRR*, abs/2301.07069, 2023.

[6] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *CoRR*, abs/2302.04023, 2023.

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

[8] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781, 2020.

[9] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[10] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240, 2022.

[11] Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. *CoRR*, abs/2208.03299, 2022.

[12] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022, 2022.

[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[14] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, Zhenshan Cao, Yanliang Qiao, Ting Wang, Bo Tang, and Charles Xie. Manu: A cloud native vector database management system. *Proc. VLDB Endow.*, 15(12):3548–3561, 2022.

[15] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652, 2023.

[16] Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2421–2436, 2023.

[17] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *CoRR*, abs/2302.00083, 2023.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[19] Patrick S. H. 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 Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[20] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[21] Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5657–5673, 2022.

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[23] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1601–1611, 2017.

[24] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416, 2022.

[25] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[26] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

# A Details of Lightweight Retrieval Tuning Technique

## A.1 Training Framework

We demonstrate the training framework of the lightweight retrieval tuning method in Figure 2.

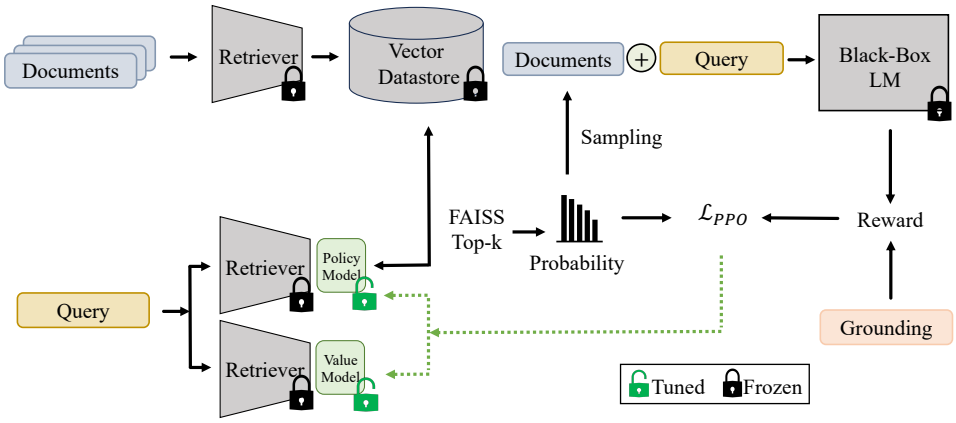


Figure 2: The training framework of LRT. The green dashed line indicates the direction of backpropagation.

## A.2 Proximal Policy Optimization

Formally, our LRT technique can be modeled as a Markov Decision Process (MDP) $S, A, P, R, \gamma$. The state space $S$ is a continuous embedding space obtained by mapping the query through the retriever. The action space is the set of the top-$k$ retrieved documents and the transition probability $P$ is determined by $\lambda_\theta$. The reward function $R$ is given by evaluating the prediction of the language model. The discount factor denoted as $\gamma$ is assigned a value of 0 in our setting, indicating that we are only concerned with maximizing the one-step reward.

To fine-tune the trainable layer, we employ the Proximal Policy Optimization (PPO) algorithm. Recently, PPO has played a significant role in aligning LLMs with human preferences. One of its advantages is that the penalty term in the algorithm ensures that the updated parameters do not deviate significantly from the old parameters, thereby adapting the model to downstream tasks without sacrificing generalization ability.

We consider $\lambda_\theta$ as the policy network, where the action space has a size of $k$ and is a subset of the corpus $\mathcal{D}$. The loss function of the PPO algorithm is defined as follows:

$$\begin{aligned} \mathcal{L}_{policy} &= \min\{rA, \text{clip}(r, 1-\epsilon, 1+\epsilon)A\} \\ r &= \frac{p(d \mid q)}{p_0(d \mid q)} = \frac{\lambda_\theta(d, q)}{\lambda_{\theta^{(0)}}(d, q)} \end{aligned} \tag{2}$$

where $\theta^{(0)} = (W^{(0)}, b^{(0)}) = (I, 0)$, which is fixed for sampling while $\theta$ will be updated during training. We construct a value network $V_\phi$. It takes a state $E(q)$ as input and outputs an estimated state value $V_\phi(E(q))$. Then the advance $A$ calculated by:

$$A = R - V_\phi(E(q))$$

The reward represents the quality of the document and can be defined by evaluating the correctness of the language models prediction. Additionally, drawing inspiration from RLHF, we add a negative KL divergence term to the reward to prevent policy network forgetting. We formulate the reward as:

$$R\text{=Evaluation Metric} - \beta \log(\frac{\lambda_\theta}{\lambda_\theta^{(0)}}) \tag{3}$$

Table 3: Cases study on TriviaQA. The color green represents the correct retrieval, while the color red indicates the wrong.

| Queries | Original Retrieval Document | Original Retrieval Answer | LRT Retrieval Document | LRT Answer |
|---|---|---|---|---|
| Which English county is the setting for most of painter John Constable's works? | ... Portrait of Maria Bicknell, Mrs. John Constable (1816) Tate Gallery, London ... Weymouth Bay: Bowleaze Cove and Jordon Hill (181617) National Gallery, London ... | Gallery, London (✗) | ... In his youth, Constable embarked on amateur sketching trips in the surrounding Suffolk and Essex countryside, which was to become the subject of a large proportion of his art... | Suffolk (✓) |
| The River Tigris rises in which country? | ...The Tigris is 1,850km long, rising in the Taurus Mountains of eastern Turkey about 25km southeast of the city of Elazig and about 30km from the headwaters of the Euphrates... | sailed up the Tigris and the Shatt al-Arab... (✗) | ...The Tigris is heavily dammed in Iraq and Turkey to provide water for irrigating the arid and semi-desert regions bordering the river valley... | Turkey (✓) |

The design of the evaluation metric depends on the downstream tasks. For example, we can use either accuracy or F1 score for classification task. $\beta$ is the hyperparameter, which is set to 1.0 during our training. To train the value network, we have the value loss:

$$\mathcal{L}_{value} = \big(R - V_\phi(E(q))\big)^2 \tag{4}$$

As a result, The final loss function consists of two components: the policy loss and the value loss.

$$\mathcal{L}_{PPO} = \mathcal{L}_{policy} + \mathcal{L}_{value} \tag{5}$$

By iteratively optimizing the loss function, we can retrieve documents that better align with downstream tasks, thereby improving the performance of the language model in a black-box scenario.

# B Experimental Supplement

## B.1 Task Description

**MMLU** is a multitask language understanding benchmark, which contains multi-choice question-answering subtasks in 57 topics. These topics can be divided into four categories: the humanities, STEM, the social sciences, and others. Due to insufficient computing resources, we use their development set as our training set, and their validation set as our test set. In addition, We use their train set as our external corpus (#>100M). We evaluate the accuracy of these four categories on our test set separately and use the average accuracy of all subtasks as our final accuracy metric.

**TriviaQA** is a reading comprehension dataset, which is composed of question-answer-evidence triples. Each question has six evidence documents on average. We use the Wikipedia domain to train and test. We randomly select 2000 samples from their training set as our training set and keep 10% of the training set as our test set. We use all the evidence documents of our training set and test set as the external knowledge base. The F1 score and Exact Match (EM) on our test set are reported.

## B.2 Training Details

For all our experiments, we set $\tau = 1$ for simplicity. To speed up the training process, we use FAISS for efficient similarity search and pre-compute the embeddings of documents in the corpus. Given a query $q$, we first compute its embedding, and then get its final embedding after it is passed through our linear head. We retrieve the top-10 (i.e. $k = 10$) documents of the query from the FAISS index and compute the similarity scores. Our value network is a three-layer MLP with 128 nodes in the hidden layer. We use the Adam optimizer to train our networks. In the PPO settings, we set $\epsilon = 0.1$. For MMLU, the 'Evaluation Metric' in the reward function is accuracy, and for TriviaQA is F1-score. All our experiments are conducted on an NVIDIA A800.

## B.3 Deeper Analysis

**Effectiveness of PPO algorithm** To demonstrate the effectiveness of the Proximal Policy Optimization algorithm, we record the training curves. The results of three Flan-T5 models on the TriviaQA task are reported. The upper of Figure 3 illustrates the variation curves of policy loss

and value loss during the training process. These losses decrease as the number of training epochs increases, indicating that the PPO algorithm functions properly. The lower of Figure 3 presents the performance changes in the training and validation sets. We test the validation set every 5 epochs. As the number of training epochs increases, the reward values on the training set consistently rise, implying that performance improvements can be achieved by optimizing our trainable module. On the validation set, there is also a tendency for performance improvement. However, due to the distribution shift between the training set and the validation set, the curves may fluctuate, and even the training exhibit overfitting. In our experiments, we implement early stopping to alleviate overfitting.

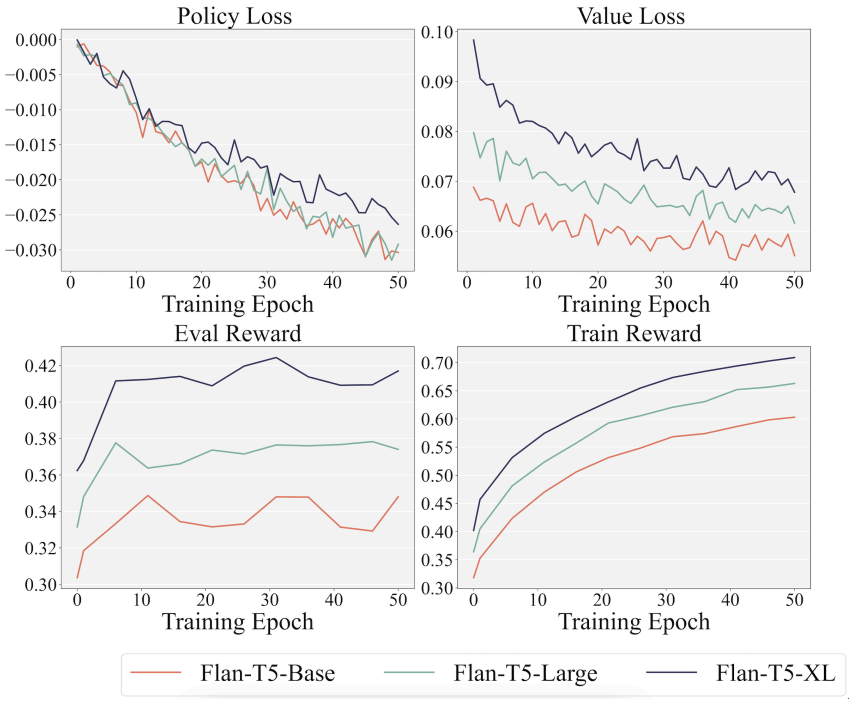


Figure 3: The training curve on TriviaQA of Flan-T5 models. The two above charts demonstrate the variations in policy and value loss and the two below charts showcase the changes in reward for the training and validation sets.

**Why does LRT improve performance?** We further investigated the reasons for performance improvement in LRT. On TriviaQA, we observe two typical cases, as shown in Table 3. The first case demonstrates that the original retrieval baseline fails to retrieve the document containing the final answer thus providing the wrong answer. However, after our training, the retrieval module can accurately identify the corresponding content from Wikipedia. Therefore, the final prediction is correct. In the second case, although the original retrieval baseline retrieves a document covering the final answer, due to the limitations of the language model itself, it fails to recognize the correct answer and makes an incorrect prediction. Our approach, on the other hand, is able to find an alternative document containing the correct answer, which is aligned with the preferences of the language model. Thus the model can make the correct prediction. These results demonstrate that our LRT technique can not only retrieve more relevant documents but also adapt better to the preferences of the language model itself.

## C Limitations

Due to resource constraints and geographical limitations, our work is not executed on larger-scale models, including open-source models such as LLaMA-33B and API-based models like PaLM2 and GPT4. In addition, considering the limited computational resources, we have not utilized the entire original training dataset during our training process. Instead, we have divided a subset of the training set. This allows for potential enhancements to our method. Furthermore, as a result of the

restricted number of trainable parameters within our method, there is a reduction in the expected generalization performance. We will explore novel approaches that are not only lightweight but also demonstrate strong generalizability.