FLASHBACK : Efficient Retrieval-Augmented Language Modeling for Fast Inference

Anonymous ACL submission

Abstract

Retrieval-Augmented Language Modeling (RALM) by integrating large language models (LLM) with relevant documents from an external corpus is a proven methodology for enabling the LLM to generate information beyond the scope of its pre-training corpus. Previous work by retrieving a set of tokens iteratively with retrieved content prepending to the input poses a high runtime issue, which degrades the inference efficiency of the LLMs because they fail to use the Key-Value (KV) cache efficiently. We propose FLASHBACK, a modular RALM designed to improve the inference efficiency of RALM with appending context pattern while maintaining decent performance after fine-tuning by Low-Rank Adaption. FLASHBACK appends retrieved documents at the end of the context for efficiently utilizing the KV cache. We also introduce the Marking Token as two special prompt tokens for marking the appending context during fine-tuning. Our experiments show that FLASHBACK can improve language modeling performance in perplexity metric. We proved the Marking Token is a usable add-on when fine-tuning models on specific context patterns. Via bypassing unnecessary re-computation, FLASHBACK achieves fast inference speed speed with long context input. The inference speed is up to $4\times$ faster than the prepending counterpart on a 7B LLM (Llama 2) in the runtime test.

1 Introduction

Large language models (LLMs) based on the Transformer architecture (Vaswani et al., 2023) such as GPT, Llama, and OPT, etc (Brown et al., 2020; Touvron et al., 2023; Zhang et al., 2022) require enormous computational resources to keep their knowledge updated (Meng et al., 2023). Thus, Retrieval-Augmented Language Modeling (RALM) has emerged as a popular approach, enabling content generation that leverages external corpora to extend beyond the knowledge inherent in the model's parameters, thereby reducing the computational cost of capturing up-to-date knowledge. The effectiveness of using RALM to enhance LLMs' performance has been proven by previous studies(Guu et al., 2020; Lewis et al., 2021; Izacard et al., 2022; Wang et al., 2023b).

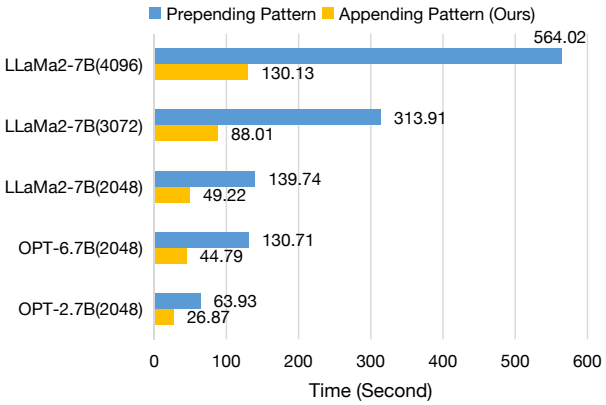

Figure 1: A series of inference tests with simulated input and retrieved content on OPT and Llama 2 for comparing Appending Pattern with Prepending Pattern. (Maximum sequence length is denoted under the model name)

Recent works utilizing retrieved content by simply prepending retrieved contents to the input without updating the language models exhibit the impressive potential of adapting off-the-shelf LLMs to the retrieved content (Shi et al., 2023; Ram et al., 2023). Other studies build RALM through the pre-training of LLM(Borgeaud et al., 2022), or engage in the continual pre-training of existing RALMs (Izacard et al., 2022; Wang et al., 2023a).

However, these works are introduced with limitations. First, the off-the-shelf LLMs are not inherently trained to incorporate retrieved content, and extensive pre-training of LLMs for building RALM incurs high computational costs (Lin et al., 2023). Second, although in-context methods have been effectively applied on off-the-shelf LLMs (Ram et al.,

[†]These authors contributed equally to this work.

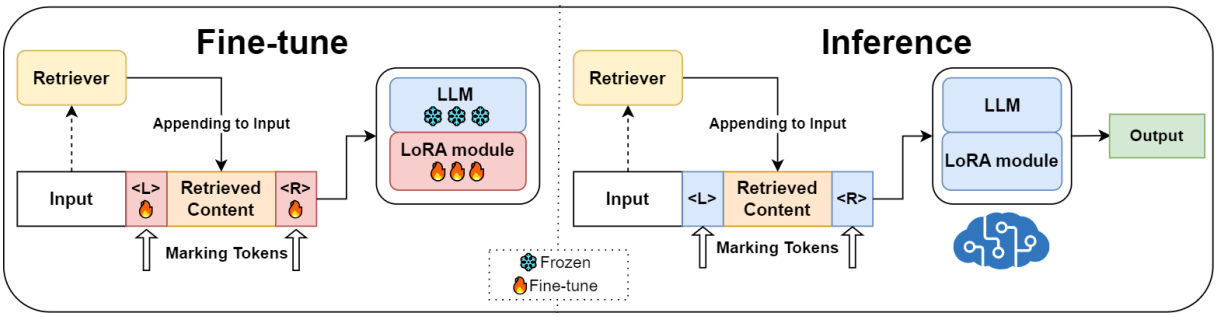

Figure 2: An illustration of the FLASHBACK pipelines. The left side of the diagram is the fine-tuning phase of our model with Marking Token and LoRA module. The inference procedure is on the right.

2023), our work verified that prepending long input with In-Context RALM poses high computational cost, which increases the inference time. To be specific, retrieved content changes with every retrieval stride tokens, and the computed key-value (KV) cache of the entire context is discarded, requiring re-computation for each change. Text generation process with long input context can substantially increase the inference runtime, and the attention KV cache grows even larger on LLM with hundreds of billions of parameters(Pope et al., 2022). Under these circumstances, the re-computation of the KV cache becomes a computationally intensive task, entailing significant costs for RALM with continuous retrieval(Ram et al., 2023).

In this paper, we propose FLASHBACK to overcome these issues we mentioned above, a method that utilizes off-the-shelf LLMs efficiently without further pre-training of LLMs as shown in Figure 2, to address the inefficiency issue by breaking the dependent of the input to retrieved content in the context. To be specific, instead of prepending the retrieved content directly to the input, we utilize the **Appending Context Pattern**, in which the retrieved content is appended at the end of the input (see Figure 3). In our method, the KV cache of each input is static during inference, and the re-computation of the KV cache takes place only in the part of the retrieved content. Although there are advantages to applying this context pattern, it also presents an issue that degrades the performance of the RALM. Our appending pattern breaks the semantic coherence of the context, which leads to a loss in perplexity when applying it to existing RALM (see Table 1). To solve this issue, inspired by Ren et al. (2023), FLASHBACK introduces tunable Marking Token to adapt to our appending context pattern for restoring comparable performance.

Notably, we use LoRA (Hu et al., 2021) to fine-tune the Marking Token while keeping both the retriever and the LLM frozen. Therefore, FLASHBACK is an adaptable approach that can cooperate with other RALM that perform retrieval for every n token, thereby aiming for efficient fine-tuning, inference and context-model alignment.

Our main contributions can be listed as follows:

- Our work analyzed the prepending context in the FLOPs analysis and experiments. We find it is inefficient in RALM with long input due to redundant re-computation of the KV-cache (see Figure 3 and section 3.3).
- We find the **Appending Context Pattern** can improve the inference efficiency of RALM. (Up to $4\times$ faster in 7B Llama 2 as shown in Figure 1)
- We exploit **Marking Token** that can be used for adapting the LLMs to new context patterns by efficient fine-tuning with LoRA instead of pre-training.

2 Background and Related Work

Retrieve-Read RALM Previous works (Borgeaud et al., 2022; Lin et al., 2023; Ram et al., 2023; Shi et al., 2023) have created distinct modules for document selection and document reading and this methodology can be defined as the Retrieve-Read RALM. In the recent RALM framework, particularly for those employing LLMs, the imperative is to align retrieved documents with the specific requirements of the LLMs (Gao et al., 2024). AAR, REPLUG, and UPRISE are fine-tuning retrievers with frozen LLMs to

achieve the alignment (Cheng et al., 2023; Shi et al., 2023; Yu et al., 2023). RETRO uses frozen retrievers to build their own RALM (Borgeaud et al., 2022). In-Context RALM uses a frozen retriever for document selection and a frozen LLM for document reading without undergoing additional training for either the LLM or the retriever (Ram et al., 2023). RAVEN (Huang et al., 2023) continually pre-train ATLAS (Izacard et al., 2022) for improving its in-context capabilities, which aligns the model to specific prompting strategy, and they also proposed Fusion-in-Context Learning for incorporating more in-context examples during its inference process.

Hence, our modeling is close to the genre of RALM that uses decoder-only LLMs **(Auto-regressive Models)** and **Retrieval-Read RALM** that uses distinct modules for document selection and document reading (Borgeaud et al., 2022; Ram et al., 2023; Lin et al., 2023; Shi et al., 2023). Furthermore, our implementation on input context is classified as **Input augmentation** by recent study(Asai et al., 2024). After all, this type of RALM has recently been defined as Modular RAG (Retrieval-Based-Generation), and it is increasingly becoming the dominant norm in the RAG domain (Gao et al., 2024).

3 Methodology

3.1 RALM with In-Context-Learning

In the In-Context RALM framework (Ram et al., 2023), an external corpus $\mathcal{C}$ is provided to the retriever. The retriever is responsible for determining the specific content within the corpus to be utilized as the retrieved content, and then the retrieved content can be concatenated with input to form a context that conditions the predictions of the LLM. Given a sequence of tokens $x_1, ..., x_n$ with length n, The probability of token sequence is represented by

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} p_\theta(x_i|[\mathcal{R}_\mathcal{C}(x_{<i}); x_{<i}]), \quad (1)$$

where θ is parameters of an auto-regressive model, $\mathcal{R}_\mathcal{C}(x_{<i})$ is retrieved content from corpus $\mathcal{C}$ using prefix $x_{<i}$ and $[\mathcal{R}_\mathcal{C}(x_{<i}); x_{<i}]$ denotes the concatenation of strings $\mathcal{R}_\mathcal{C}(x_{<i})$ and $x_{<i}$.

For the trade-off between runtime cost and model performance, retrieval stride s and query length ℓ are proposed for less frequently performing the retriever call (at least, every s > 1 token). Therefore, the retrieval would be performed for every s tokens, using the last ℓ tokens as designated retrieved content for input.

The probability of the token sequence is represented by

$$p(x_1, ..., x_n) = \prod_{j=0}^{n_s-1} \prod_{i=1}^{s} p_\theta(x_{s\cdot j+i}|[\mathcal{R}_\mathcal{C}(q_j^{(s,\ell)}); x_{<s\cdot j+i}]), \quad (2)$$

where $n_s = n/s$ is the number of retrieval strides, $q_j^{(s,\ell)} = x_{s\cdot j-\ell+1}, ..., x_{s\cdot j}$ is the retriever input.

Our work followed this mechanism of retrieval besides using different context patterns because this mechanism uses distinct retrieve modules and readers (LLMs) so that they can be optimized independently.

3.2 Context Pattern

In the decoder-only transformer-based models, the computation of attention modules is related to the query of the current token and the key-value representations of preceding tokens. (Vaswani et al., 2023). During inference, the key-value of the current token would be saved in the KV cache, eliminating the requirement in the subsequent step. We find that prepending retrieved content to the input has been prevalent in previous methods(Ram et al., 2023; Shi et al., 2023). However, the key-value computed for previous tokens becomes obsolete, and a re-computation of the new key-value is required for each retrieval step since the retrieved content varies with each retrieval.

To avoid the extensive cost of re-computation, we remove the auto-regressive dependency between retrieved content and input in context by using an appending context pattern. Specifically, we append retrieved content to input (at the end of the input) as shown in Figure 3.

Then, the probability of the token sequence is represented by

$$p(x_1, ..., x_n) = \prod_{j=0}^{n_s-1} \prod_{i=1}^{s} p_\theta(x_{s\cdot j+i}|[x_{<s\cdot j+i}; \mathcal{R}_\mathcal{C}(q_j^{(s,\ell)})]). \quad (3)$$

In this case, for each input, the change of retrieved content $\mathcal{R}_\mathcal{C}(q_j^{(s,\ell)})$ would not require a re-computation of new value therefore preserving the computed value of previous tokens in KV cache $x_{<s\cdot j+i}$. (Algorithm pseudo code in Appendix A.)

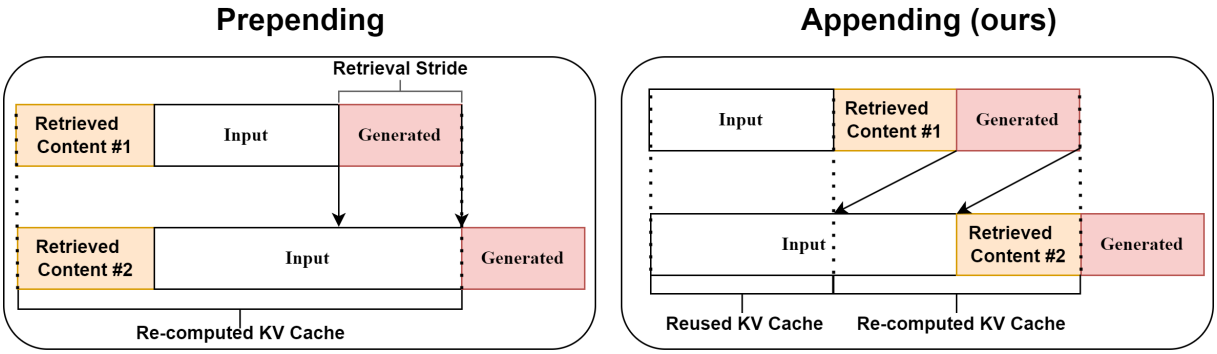

Figure 3: Comparison of prepending and appending context patterns in interval of change of retrieved content, arrow represents that generated tokens in past round will become a part of input in the next round. In the prepending context pattern,the key-value of all tokens in past round should be re-computed. By contrast, in appending context pattern, the key-value of input in past round can be reused in the next round.

3.3 FLOPs analysis of context pattern

We analyzed the Floating Point Operations (FLOPs) of re-computation. In the following analysis, the input batch size is denoted as b, the hidden size of the model as h, and the dimensions of the Key and Value vectors are also h, with a maximum sequence length of T, retrieval stride of s, and a model comprised of l layers, then the size of the input vector is given by $[b, i \cdot s, h]$. The Key and Value projection matrix dimensions are $[h, h]$, and the number of retrievals is denoted as i. Overall, the FLOPs of re-computation are formulated as:

$$C_0 = 2l \sum_{i=1}^{T/s} 2bish^2 = \frac{2T(T+s)bh^2l}{s}. \quad (4)$$

It increases quadratically as a function of the maximum sequence length T, and thus, it will increase dramatically when the maximum sequence length T is at a large value.

3.4 Marking Token and Fine-tuning Choice

Since LLMs are not aligned explicitly to our appending pattern, we use the Marking Token and Low-Rank Adaption (LoRA) (Hu et al., 2021) to adapt them to the appending pattern while keeping origin model weights frozen so that the alignment is achieved without modifying the inherent ability of the LLMs. Marking Tokens are represented as two special prompt tokens denoting <MARK_L>,<MARK_R> which are added into the vocabulary of the model, and they are used for marking the boundary of the retrieved content in context (see Figure 2). Then, we fine-tune the model using LoRA with Marking Tokens applied for adapting the appending pattern. LoRA is a parameter-efficient fine-tuning (PEFT) for fine-tuning a subset of parameters in the pre-trained model while targeting for obtaining comparable performance to full model fine-tuning. Recent study suggests that forgetting is inevitable in fine-tuning but the PEFT method can facilitate less forgetting during the fine-tuning process which does not heavily damage the knowledge integrity of the pre-trained LLM (Kalajdzievski, 2024). In FLASHBACK, we opt for LoRA to demonstrate how fine-tuning a relatively minimal set of model weights enables our model to adapt to the appending context pattern while keeping pre-training model weights frozen. Our experiments show that FLASHBACK has significantly boosted the inference speed, and it can still maintain comparable perplexity (PPL) after fine-tuning. During fine-tuning, all parameters are frozen except for the embedding of <MARK_L>, <MARL_R>, and LoRA modules applied to attention layers.

3.5 FLOPs analysis of Marking Token and LoRA

If the rank of weight matrices of LoRA is r, then the projection matrix dimensions are $[h, r]$. The average length of retrieved content is d. We only equip LoRA modules, two projection matrices, for Key and Value matrices. Overall, the FLOPs of LoRA modules in appending context pattern can be formulated as:[1]

[1]We only consider multiplication operations and omit some addition operations.

$$C_1 = 2l(4bThr + bTh) + 2l\sum_{i=1}^{T/s}[4b(d+s)hr + b(d+s)h] = \frac{2l(4r+1)bhT(d+2s)}{s} \quad (5)$$

Combining equation 4 and 5, the decreament of FLOPs when using appending context pattern with LoRA is:

$$C_{\text{decreament}} = C_0 - C_1 = \frac{2lTbh\left[(T+s)h - (4r+1)(d+2s)\right]}{s} \quad (6)$$

3.6 Retriever

Our runtime test used a sparse model, the BM25 (Robertson and Zaragoza, 2009), and QA task is tested with DPR retriever (Karpukhin et al., 2020) for demonstrating our idea. In practice, FLASHBACK intrinsically supports switching to other retrievers in a plug-and-play manner.

4 Experiments

4.1 Run-time Improvement

Setup We test OPT, GPT-2, and Llama2 in different model sizes and employ simulated retrieved content and input in inference speed tests. Our experiment encompassed tests under varying lengths of input. Additionally, it involved examining the inference time under two distinct context patterns. 4.5.

Results In Figure 4, we compare the appending pattern (in-context), prepending pattern (in-context), and **FlashBack** in OPT models. We scale the model size from 125M to 6.7B. The acceleration of inference time is more effective in large models that have more layers and larger hidden sizes. In Figure 5, we scale the max sequence length from 1088 to 3968 and observe significant improvement, as Llama 2 has a larger max sequence length, which is 4096 tokens. (Note: The test results for inference speed may vary depending on the hardware used and setting; our runtime tests are conducted on a single A100-80GB GPU setting to P0 state)

4.2 Language Modeling

Setup We test FLASHBACK on Wikitext-2 (Merity et al., 2016) dataset, and three datasets: Arxiv, Freelaw, and Stackexchange from the Pile (Gao et al., 2020). We report token-level average perplexity as the evaluation metric (excluding <MARK_L>,<MARK_R>). Our experiments use models of OPT (1.3B-6.7B) and GPT-2 (124M-1.5B). We compare FLASHBACK with the appending context pattern without fine-tuning and also the prepending context pattern. (For smaller models that are under 774M parameters, the results are in Appendix C Table 8)

Results Table 1 shows the results of different setups on the dev set. We can see that the appending context pattern applied directly on LLMs through the in-context method generates higher perplexity compared with the model using the prepending context pattern. Our result on fine-tuning with appending context patterns by using LoRA demonstrates that the model can adapt to such patterns after fine-tuning. Notably, with both the Marking Token and LoRA applied, the difference of perplexity results between two distinct context patterns becomes progressively alleviated as the size of the test model increases. We also test GPT-2 from 124M to 1.5B to validate this observation. (See Table 8 in D)

4.3 Language Modeling with Continuous Retrieval

Setup As the extra run-time cost of LoRA in comparison to the full-parameter is acceptable, we employ it in the following experiments. We continue to test FLASHBACK on Wikitext-2 (Merity et al., 2016) dataset and design a multiple retrieval scenario. Specifically, given a token chunk $x_{1,\ldots,n_c}$, we split it by retrieve stride to get prefixes: $x_{1,\ldots,s}, x_{1,\ldots,2s}, \ldots, x_{1,\ldots,n_c-s}$, where n_c is the chunk length and s is retrieve stride. Then we use each prefix with corresponding retrieved passages to predict the next s tokens. We report token-level average perplexity as the evaluation metric (excluding <MARK_L>,<MARK_R>).

Results Like results on the language modeling task, a similar trend is in Table 2. Both Marking Token and LoRA are necessary to improve model performance. The performance gap between context patterns decreases as the model size increases.

4.4 Question Answering

As improvement to the perplexity of the model result in performance improvement in downstream tasks, perplexity is as a proxy metric for improved general-purpose abilities. In the In-Context RALM, retrieved passages boost model performance both

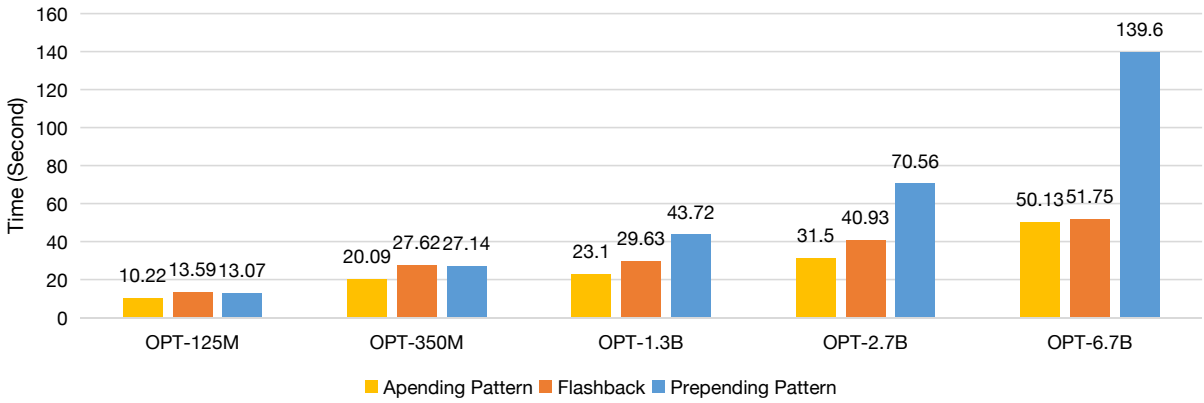

Figure 4: Time for OPT models generating a sequence of length 2048 tokens (including 128 tokens for retrieved content), given a input of 512 tokens, with retrieval stride of 16.

Model	A./P.	LoRA	M.T.	Wikitext-2	Arxiv	Freelaw	Stackexchange
OPT-1.3B	N.			16.76	9.64	8.58	7.78
	P.			15.02	9.59	8.35	7.63
	P.	✓		10.23	8.33	7.47	6.87
	P.	✓	✓	**10.22**	**8.31**	**7.45**	**6.87**
	A.			81.57	51.94	53.73	42.99
	A.	✓		12.65	11.38	10.11	9.13
	A.	✓	✓	**10.49**	**8.72**	**7.85**	**7.17**
OPT-2.7B	N.			14.50	8.72	7.74	6.99
	P.			13.14	8.68	7.56	6.88
	P.	✓		9.23	7.69	6.80	6.23
	P.	✓	✓	**9.23**	**7.68**	**6.78**	**6.22**
	A.			76.41	50.68	51.78	42.48
	A.	✓		11.58	10.88	9.31	8.51
	A.	✓	✓	**9.52**	**8.08**	**7.19**	**6.53**
OPT-6.7B	N.			12.30	7.74	6.94	6.22
	P.			11.20	7.73	6.83	6.15
	P.	✓		**8.23**	**6.98**	6.19	5.58
	P.	✓	✓	8.24	6.99	**6.18**	**5.58**
	A.			68.31	46.53	48.33	40.25
	A.	✓		10.54	10.92	9.27	8.04
	A.	✓	✓	**8.59**	**7.43**	**6.64**	**5.94**

Table 1: Perplexity (the lower the better) of OPT models on the entire validation set of WikiText-2 dataset and partial test set of Arxiv, Freelaw and Stackexchange datasets. The retrieval stride is set to 16. **A.** and **P.** refers to the appending and prepending context pattern respectively. **M.T.** refer to the Marking Tokens. And **N.** refers to none retrieval.

on upstream and downstream tasks–language modeling and question answering, with perplexity and EM score as metrics, respectively. Although FlashBack can improve models on language modeling tasks, it is unclear whether it affects the downstream ability of models.

Setup We test FlashBack in different settings (see Table 3) on Natural Questions (NQ) (Kwiatkowski et al., 2019) and TriviaQA(Joshi et al., 2017) open-domain question answering datasets to observe the effect of context pattern and the marking token in the QA task.

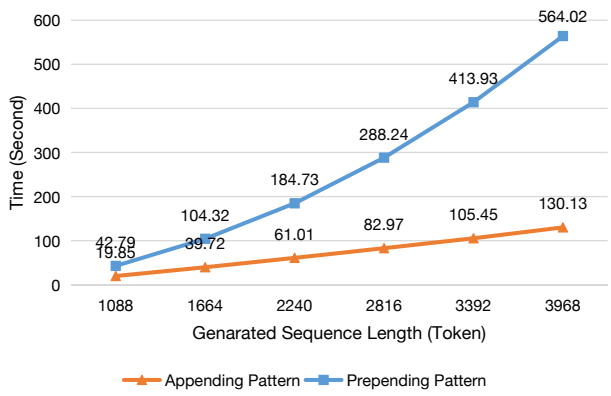

Figure 5: Time for Llama2-7B generating sequences with different length, given a input of 512 tokens, with retrieval stride of 16.

Model	A./P.	LoRA	M.T.	Perplexity↓
OPT-1.3B	P.			26.30
	P.	✓		14.82
	P.	✓	✓	14.79
	A.	✓		15.41
	A.	✓	✓	15.20
OPT-2.7B	P.			22.73
	P.	✓		13.14
	P.	✓	✓	13.14
	A.	✓		13.72
	A.	✓	✓	13.49
OPT-6.7B	P.			19.51
	P.	✓		11.65
	P.	✓	✓	11.68
	A.	✓		12.18
	A.	✓	✓	11.98

Table 2: Perplexity of OPT models on the language modeling with continuous retrieval task, evaluated on the validation set of WikiText-2. The retrieval stride is set to 16.

Results Our results show the context pattern does not evidently affect the model performance in the QA tasks after fine-tuning the model. However, the Marking Token can improve the model's performance in QA tasks, especially for the test with the appending context pattern.

4.5 Implementation Details

Run-time Improvement We randomly sample from a discrete uniform distribution $U\{500, 1000\}$ to get simulated token sequences to be the pseudo input and pseudo retrieved content. We maintain the length of simulated retrieved content to a fixed value of 128, then test models with different maximum sequence lengths. (Setup details are shown in Appendix Table 5). For the retrieval stride, all models are set to 16. All run-time experiments are conducted by using a single NVIDIA A100 GPU.

A./P.	LoRA	M.T.	EM↑ NQ	EM↑ TriviaQA
P.	✓		17.0	43.3
P.	✓	✓	17.5	**44.7**
A.	✓		18.7	42.5
A.	✓	✓	**20.3**	43.0

Table 3: QA Results of NQ, TriviaQA test set for OPT-6.7B. The model is fine-tuned on the language modeling with continuous retrieval task. We test with DPR retriever and keep top-2 retrieved documents.

Language Modeling For every s token in the train set (refer to target), we sample from a discrete uniform distribution $U\{T/2, T-s\}$ to obtain the start position of the target, where s is retrieval stride and T is the maximum sequence length. For Wikitext-2, Wikitext-103, a superset of Wikitext-2, is used as a retrieval corpus. To avoid data leakage that target outputs are included in retrieved contents, we filter out retrieved content with titles that match Wikitext-2 during data construction. Owning to the much larger size of the Pile dataset, for Arxiv, Freelaw, and Stackexchange, the validation split and 80% of the test split is a mixture as a retrieval corpus. The rest of 20% of the test split is divided into two parts equally as the training and the test dataset, respectively. We employ the BM25 retrieval and configure the query length to 16, indicating the utilization of the preceding 32 tokens from the target as input for the retriever. We use the next token prediction objective for language modeling experiments. The loss is computed on target tokens in context.

Training Details The trainable parameters for all models include two token embedding of <MARK_L> and <MARK_R>, and LoRA modules applied to all attention layers. The rank of weight matrices of LoRA is set to 16. The batch size is set to 16. The learning rates for GPT-2 models (124M-1.5B) are specified as [4e-4, 2e-4, 1e-4, 5e-5]. Correspondingly, the learning rates for OPT models (1.3B-6.7B) are specified as [1e-4, 5e-5, 5e-5]. We train for 5K additional steps, with 10% warm-up steps and a cosine learning rate schedule.

For running on a single 24GB RAM GPU, it is feasible to use the bf16 format on the OPT-6.7B model and set the maximum sequence length to 512.

In our implementation, we use the Transformers library (Wolf et al., 2020) and BM25 retrieval from the Pyserini library (Lin et al., 2021). All fine-tuning experiments are conducted by using 4×24GB RAM GPUs.

5 Ablation Study

Marking Token We have already tested the model using an appending context pattern that is solely fine-tuned with LoRA without Marking Token to compare with FLASHBACK. In Table 1, test results of models with Marking Tokens and LoRA fine-tuning have lower PPL, which means fine-tuning the Marking Token could further boost the performance of the RALM and, therefore, meet our expectations. Furthermore, we also tested whether the Marking Token and LoRA module add additional inference costs to our modeling. Our results in Figure 4 show that FLASHBACK is slightly slower than the model without these modules but the difference is tiny enough to be ignored compared to the overall increase of the inference speedup. In the QA test (Table 3), the results show the marking token can be effective on model performance on both context patterns.

Model	Perplexity Retrieval Stride		
	16	32	64
GPT2-124M	28.11	23.64	21.80
GPT2-355M	16.46	15.66	15.80
GPT2-774M	13.46	13.12	13.24
GPT2-1.5B	12.59	12.28	12.24
OPT-125M	24.73	22.42	21.26
OPT-350M	17.29	16.71	16.71
OPT-1.3B	11.51	11.29	11.31
OPT-2.7B	10.70	10.38	10.23
OPT-6.7B	9.40	9.10	9.09

Table 4: Perplexity (the lower the better) of GPT models and OPT models on the validation set of WikiText-2 dataset, varying retrieval stride, by using FLASHBACK.

Retrieval Stride Allocating the number of retrieval strides for each iteration is a design choice that is related to model performance in both perplexity and inference time. As we mentioned above in section 3.1, the In-Context RALM (Ram et al., 2023) choose to use a cherry-picked retrieval stride to balance the performance trade-off. They choose to use a relatively small stride value to have better generation (lower perplexity) without adding extensive inference cost. However, our experiments (see Table 4 and 6 in appendix) on varying retrieval strides on FLASHBACK implied a different opinion. We find using 32 or 64 as retrieval stride does not degrade the model's performance. When testing with a larger model such as OPT-6.7B, the test with a higher retrieval stride has the lowest perplexity compared to the test using same model with lower retrieval stride. Therefore, the trade-off between perplexity and inference time may not be a particularly evident phenomenon for some cases. As a consequence, we made a bold speculation that it might be an advantage for FLASHBACK for increasing the retrieval stride without intensively degrading the model's perplexity and still maintaining high-speed inference in long context input scenarios. This phenomenon may requires further demonstration in the future.

6 Conclusion

Prepending retrieved contents to the input without modification of model architecture has been a practicable method for LM to use external knowledge. However, the prepending context pattern hinders RALM from efficiently generalizing at a considerable context length with long input.

This paper presented the FLASHBACK method, enabling RALM with faster inference speed while keeping competitive model performance. We proved that the appending context pattern is more efficient than the prepending counterpart. Besides, our experiments demonstrated that LLMs can adapt to new context patterns using Marking Tokens with adaption fine-tuning (LoRA). Its performance in generation quality is closely equivalent to the LLM using a prepending counterpart as the size of the model increases.

Limitations

FlashBack is specifically designed for RALM methods that use multiple retrievals by every n tokens. Other works use retrieval strategies like retrieval once or retrieval by every n text chunks are not covered in our paper and require further study. In our experiments, the retrieval stride is a fixed value

in each test, but it can be a dynamic variable, so there might be extra improvements. In the run-time comparison experiment, due to constrained computational resources, we used simulated (pseudo) data for testing. It is because we did not find an appropriate public downstream task that fulfills the following requirements: 1) long context input, 2) multiple retrieval, and 3) generated text becomes a new part of the context. QA task generally comes with a short question which is a short input so the burden of RALM mainly lies on retrieved documents rather than input. However, we do believe this type of scenario exists in some applications where long input is required with multiple retrievals. Additionally, we did not test our modeling on LLMs over the size of 7B parameters since they generally require extensive computational resources.

Furthermore, in FLASHBACK, we did not scale the number of retrieved contents to a large value. It is practicable to generalize appending context patterns to more retrieved contents (for example, using the framework of REPLUG(Shi et al., 2023)).

Ethics Statement

This research explores the Retrieval-Augmented Generation (RAG) application leveraging Large Language Models (LLMs). We call this method as Retrieval-Augmented Language Modeling (RALM).

The data utilized in training and evaluating the RALM was obtained from publicly available datasets and sources adhering to ethical guidelines, ensuring respect for user privacy and consent. The potential biases may exist inherently in both the external corpus and the LLMs. Using FLASHBACK for real applications needs extra precautions to avoid reinforcing stereotypes or propagating misinformation through the generation process.

References

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen tau Yih. 2024. Reliable, adaptable, and attributable language models with retrieval.

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. 2022. Improving language models by retrieving from trillions of tokens.

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. 2020. Language models are few-shot learners.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Denvy Deng, and Qi Zhang. 2023. Uprise: Universal prompt retrieval for improving zero-shot evaluation.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Jie Huang, Wei Ping, Peng Xu, Mohammad Shoeybi, Kevin Chen-Chuan Chang, and Bryan Catanzaro. 2023. Raven: In-context learning with retrieval augmented encoder-decoder language models.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Damjan Kalajdzievski. 2024. Scaling laws for forgetting when fine-tuning large language models.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, page 2356–2362, New York, NY, USA. Association for Computing Machinery.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2023. Ra-dit: Retrieval-augmented dual instruction tuning.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models.

Siyu Ren, Qi Jia, and Kenny Q. Zhu. 2023. Context compression for auto-regressive transformers with sentinel tokens.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval, 3:333–389.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. 2023a. Instructretro: Instruction tuning post retrieval-augmented pretraining.

Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, Anima Anandkumar, and Bryan Catanzaro. 2023b. Shall we pretrain autoregressive language models with retrieval? a comprehensive study.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Appendix

A FlashBack Algorithm

Algorithm 1 In-Context RALM with FLASHBACK

Given retrieve stride s, retrieved query length ℓ, and maximum sequence length T.
Given auto-regressive model p_θ, retriever q, and initial prompt sequence $x_0, ..., x_t$.
Initialise: $n \leftarrow t$, retrieved evidence $e \leftarrow$ None, Key-Value Cache $K \leftarrow$ None.

while $n \leq T$ **do**
  The retriever is called for every stride s tokens.
  if $(n - t \mod s) = 0$ **then**
    $e \leftarrow q(x_{n-\ell,..,n})$ ▷ Call retriever
    $l_e \leftarrow$ length of e
    Merge past generated tokens to the context, using truncated KV cache:
    $p_n, K_{1,...,l_e+n-1} \leftarrow p_\theta(x|[x_{1,...,\tilde{n}-1}; x_{\tilde{n},...,n-1}; e]; K_{1,...,\tilde{n}-1})$
    $\tilde{n} \leftarrow n$ ▷ Save last retriever call position
  else
    $p_n, K_{1,..,l_e+n-1} \leftarrow p_\theta(x|[x_{1,..,\tilde{n}-1}; e; x_{\tilde{n},...,n-1}]; K_{1,..,l_e+n-2})$
  end if
  $x_n \sim p_n(x)$
  $n \leftarrow n + 1$
end while

B Experiment Setup

Models	Length of input	Length limitation
GPT2	256	1024
OPT	512	2048
Llama2	512	4096

Table 5: The length of input and the length limitation we used in run-time improvement experiments.

C Language Modeling With More Retrieved Content

Inspired by REPLUG(Shi et al., 2023), we append each retrieved content separately to the input and ensemble respective output probabilities.

Setup We use the same setup as section 4.2, except varying the number of retrieved content.

Model	Time (Second) Retrieval Stride		
	16	32	64
GPT2-124M	6.16	5.67	5.89
GPT2-355M	10.68	10.37	10.35
GPT2-774M	16.08	15.97	15.77
GPT2-1.5B	21.71	21.23	20.26
OPT-125M	10.72	10.09	10.10
OPT-350M	20.01	19.9	19.69
OPT-1.3B	22.13	20.72	19.79
OPT-2.7B	30.71	29.10	28.17
OPT-6.7B	50.25	43.79	41.06

Table 6: Time for GPT models and OPT models generating sequences with respective maximum context length, varying retrieval stride.

Model	Perplexity # Of Retrieved Content			
	1	2	4	8
GPT2-124M	25.00	24.06	24.65	25.72
GPT2-355M	16.05	15.58	15.91	16.50
GPT2-774M	13.28	12.94	13.21	13.67
GPT2-1.5B	12.49	12.19	12.43	12.83
OPT-125M	23.44	22.53	23.09	23.97
OPT-350M	16.89	16.31	16.67	17.29
OPT-1.3B	11.45	11.10	11.32	11.67
OPT-2.7B	10.48	10.14	10.32	10.63
OPT-6.7B	9.34	9.05	9.20	9.34

Table 7: Perplexity (the lower the better) of GPT models and OPT models on the validation set of WikiText-2 dataset, varying the number of retrieved content, by using FLASHBACK.

Results From the test results of varying the number of retrieved contents, the perplexity of all models we tested slightly worsens as the number of retrieved contents increases. We find the best perplexity comes with setting the number of retrieved contents to 2. We speculate the reasoning behind this phenomenon is that the second retrieved document is likely to be highly relevant to the first retrieved document. However, this experiment is only for demonstrating that FLASHBACK can be used for multi-document retrieval. Further improvement on multi-document retrieval is not our focus in this paper and it can be a direction in later work.

D Perplexity Results of GPT2 Model

We test the GPT2 models on various model size in perplexity metrics. The results are presented in table 8.

Model	A./P.	LoRA	M.T.	Wikitext-2	Arxiv	Freelaw	Stackexchange
GPT2-124M	N			30.69	15.19	16.65	14.76
	P.			26.00	14.98	16.00	14.43
	P.	✓		17.46	11.46	11.55	9.02
	P.	✓	✓	17.41	11.44	11.53	9.00
	A.			72.29	46.59	56.32	50.71
	A.	✓		25.56	22.35	24.31	17.41
	A.	✓	✓	19.14	14.21	15.01	10.25
GPT2-355M	N			22.13	11.79	12.01	10.07
	P.			19.26	11.68	11.63	9.91
	P.	✓		13.33	9.46	9.00	7.13
	P.	✓	✓	13.33	9.44	8.97	7.13
	A.			59.50	42.33	47.23	37.00
	A.	✓		17.05	18.60	18.54	13.18
	A.	✓	✓	14.37	10.51	9.94	7.81
GPT2-774M	N.			19.11	10.69	11.29	9.74
	P.			16.71	10.58	11.08	9.59
	P.	✓		11.74	8.56	8.03	6.45
	P.	✓	✓	**11.69**	**8.52**	**8.03**	**6.43**
	A.			56.15	42.44	49.25	39.61
	A.	✓		14.40	16.83	16.35	11.99
	A.	✓	✓	**12.13**	**8.95**	**8.47**	**6.78**
GPT2-1.5B	N.			17.32	9.87	10.61	9.03
	P.			15.36	9.81	10.49	9.01
	P.	✓		10.84	8.07	7.45	6.01
	P.	✓	✓	**10.83**	**8.04**	**7.43**	**5.98**
	A.			53.30	41.72	49.25	40.25
	A.	✓		13.75	16.48	15.20	11.50
	A.	✓	✓	**11.32**	**8.57**	**8.01**	**6.35**

Table 8: Perplexity (the lower the better) of OPT models on the entire validation set of WikiText-2 dataset and partial test set of Arxiv, Freelaw and Stackexchange datasets. The retrieval stride is set to 16. **A.** and **P.** refers to the appending and prepending context pattern respectively. **M.T.** refer to the Marking Tokens. And **N.** refers to none retrieval.