RMT for Context Length Extension

Ajay Krishnan Gopalan
Computer Science
New York University
ag8172@nyu.edu

Ishaan Shivhare
Electrical and Computer Engineering
New York University
iss6582@nyu.edu

Raghuram Cauligi Srinivas
Electrical and Computer Engineering
New York University
rc5428@nyu.edu

Vishwa Gohil
Electrical and Computer Engineering
New York University
vg2523@nyu.edu

Abstract

This paper evaluates the effectiveness of Recurrent Memory Transformer (RMT) [1] architecture in extending the context length of Large Language Models through memory-augmented processing. We implement an RMT system enhanced with Low-Rank Adaptation (LoRA) [2] and evaluate its performance on the Wikitext-2 dataset across various memory and segment configurations. Our results show that RMT achieves optimal performance with smaller segment configurations, demonstrating a 19% improvement in perplexity compared to vanilla GPT-2. However, we observe diminishing returns with increased memory sizes beyond 16 tokens, suggesting that careful configuration of memory and segment parameters is crucial for optimal performance. These findings provide practical insights for implementing memory-augmented transformers while maintaining computational efficiency. The implementation and code are available at https://github.com/Vishwa44/nlp_rmt.git.

1 Introduction

The exponential growth in the size and capabilities of Large Language Models (LLMs) has significantly improved their ability to handle tasks requiring extensive context comprehension. However, increasing the context length poses computational and memory challenges, especially for applications requiring real-time or resource-efficient processing. To address this, researchers have proposed various techniques, including memory-augmented architectures [1, 3].

The Recurrent Memory Transformer (RMT) is a notable approach that introduces memory cells to extend the context length by efficiently managing and reusing contextual information across segments [4]. This study evaluates the performance of RMT in extending the effective context length of LLMs. Specifically, we explore the interplay of memory size, segment sizes, and optimization techniques in training these models. We conduct experiments to assess their impact on model performance, aiming to provide actionable insights into scaling LLMs effectively without significant computational overhead [5].

2 Related Work

The challenge of extending context length in transformer models has seen several innovative approaches. The Recurrent Memory Transformer (RMT) [1] pioneered the integration of memory tokens with transformer architecture, while similar memory-based approaches were applied to specific

tasks like document-level machine translation [3]. Recent work has explored training-free methods for context extension, such as InfLLM's efficient context memory [4] and positional interpolation techniques [6]. The Infini-attention mechanism [7] further advanced this field by introducing methods for processing theoretically unlimited context lengths.

To address the computational challenges in fine-tuning large language models for extended context, several efficient adaptation techniques have been developed. LoRA [2] introduced low-rank adaptation as a parameter-efficient fine-tuning method, which was further extended by LongLoRA [5] to specifically tackle long-context scenarios. These approaches demonstrate the growing focus on developing both computationally efficient and practically viable solutions for extending context windows in large language models.

3 Methodology

The RMT algorithm is designed to extend the context length of transformer-based language models by introducing memory cells that carry contextual information across segments [1]. These memory cells enable the model to handle longer input sequences without drastically increasing computational complexity.

3.1 Traditional RMT

The RMT architecture processes an input sequence $X = \{x_1, x_2, \ldots, x_T\}$ divided into n segments $S = \{S_1, S_2, \ldots, S_n\}$, where each segment S_t has a fixed size b. Memory tokens M_t are used to carry contextual information across segments. For each segment t, the following operations are performed:

$$M_{t+1} = f_{\text{memory}}(M_t, S_t) \tag{1}$$
$$O_t = f_{\text{transformer}}(S_t, M_t) \tag{2}$$

Here, f_{memory} updates the memory tokens, and $f_{\text{transformer}}$ processes the input segment along with the memory tokens to generate the output O_t.

3.2 LoRA Fine-Tuning

To reduce computational overhead and enable efficient fine-tuning, we incorporate Low-Rank Adaptation (LoRA) [2]. LoRA adjusts the transformer's weights by decomposing them into low-rank updates:

$$W = W_0 + \Delta W, \quad \Delta W = A \cdot B \tag{3}$$

where W_0 is the pre-trained weight matrix, and ΔW is a low-rank matrix with rank r. This allows targeted adaptation without fully updating all weights, saving memory and computation.

3.3 Integrated Workflow

For a given segment S_t, the combined RMT and LoRA workflow is:

$$M_{t+1} = f_{\text{memory}}(M_t, S_t) \tag{4}$$
$$O_t = f_{\text{transformer}}(S_t, M_t + \Delta W) \tag{5}$$

This approach enables the model to learn efficiently across long sequences with minimal resource utilization [5].

3.4 Algorithm

The training process for RMT is outlined in Algorithm 1:

Algorithm 1: Training RMT with Memory Sweeps

Input: Memory size m, input size i, block size b, number of segments n
Output: Trained RMT model
Initialize model with m, b, i, and n;
Tokenize dataset and group texts into chunks of size b;
for *each epoch* **do**
  for *each batch in training data* **do**
    Load input S_t and memory Memory_t;
    Compute Memory_{t+1} and Output_t;
    Calculate loss and backpropagate;
  end
  Evaluate on validation data;
end

3.5 Compute Efficiency

The use of memory cells reduces the need for processing the entire context repeatedly, resulting in significant computational savings. The integration of LoRA further minimizes memory overhead during training, allowing experiments to be conducted on limited hardware resources [5].

4 Experiments and Results

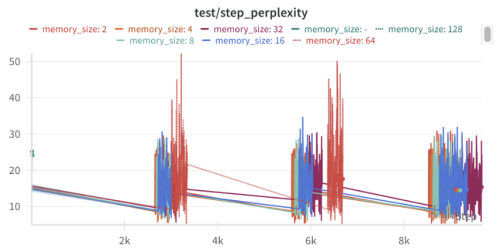

Figure 1: Training loss curves across epochs, demonstrating convergence patterns for different memory configurations

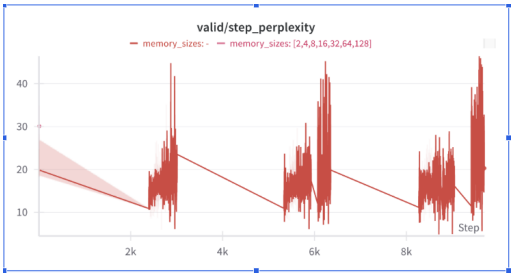

Figure 2: Validation loss trends showing model generalization capabilities and stability across training epochs

We conducted two primary experiments using the Wikitext-2 dataset to evaluate RMT's performance: (1) analyzing the impact of segment numbers on shorter sequences, and (2) investigating memory size effects on longer contexts [6]. The dataset was preprocessed, tokenized, and segmented according to each experimental configuration. Training ran for 10 epochs using a batch size of 1 and learning rate of 1×10^{-4}, with model performance evaluated using perplexity on the validation dataset.

Config	Size	nseg	ppl
GPT-2	1024	1	29.94
RMT	1024	2	24.29
RMT	1024	4	25.89
RMT	1024	6	25.67

Table 1: Impact of Segment Configuration

Config	Size	Mem	ppl
RMT	4096	2	17.37
RMT	4096	4	17.86
RMT	4096	8	17.40
RMT	4096	16	17.99
RMT	4096	32	18.05
RMT	4096	64	19.18

Table 2: Effect of Memory Size

Our segment analysis reveals that RMT achieves optimal performance with nseg=2, showing a 19% improvement in perplexity (24.29) compared to vanilla GPT-2 (29.94) for 1024-token inputs [1].

Additional segments (nseg=4,6) lead to performance degradation, suggesting that excessive segmentation introduces unnecessary overhead in information processing for shorter sequences.

In the memory size experiment with a 4096 input size, we see better performance at mem_size = 2 [4]. We observe a degradation as memory size increases, this could likely be attributed to increased complexity in the model's memory management or potential overfitting.

5 Further Work

Future research will focus on comparing RMT-enhanced small language models with larger models that have naturally longer context windows [7]. Key directions include developing dynamic memory allocation mechanisms, optimizing memory configurations for specific tasks, and exploring synergies with other efficiency-improving techniques [5]. Additionally, we plan to investigate the effectiveness of fine-tuning RMT-enhanced models on task-specific datasets to evaluate their performance against larger models, potentially establishing a more resource-efficient approach to handling extended context lengths.

6 Conclusion

Our results demonstrate that RMT architecture combined with LoRA [2] effectively extends context length in smaller language models, achieving a 19% perplexity improvement over vanilla GPT-2 and a 42% improvement over larger input sizes and while being 50% more efficient [1] compared to models with similar input sizes. The combination of memory tokens and segmentation enables theoretically infinite input sizes, while LoRA integration ensures efficient task-specific adaptation. These findings suggest that memory-augmented architectures offer a promising path toward scalable language models without the computational demands of larger systems.

References

[1] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[3] Yukun Feng, Feng Li, Ziang Song, Boyuan Zheng, and Philipp Koehn. Learn to remember: Transformer with recurrent memory for document-level machine translation. *arXiv preprint arXiv:2205.01546*, 2022.

[4] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory, 2024.

[5] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models, 2024.

[6] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.

[7] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.