TARM: Token Averaging Recurrent Memory Transformers

Rahul Vigneswaran K
CS23MTECH02002

Peketi Divya
Assigned TA

Prof. C. Krishna Mohan
Course Instructor

Done as part of the coursework on Visual Computing (CS6450).

Abstract

Recurrent Memory Transformers (RMT) are well know for their ability to learn very long-term dependencies and general-purpose memory processing in various applications. However, their capacity too is limited by the size of the memory tokens. In this paper, we propose a novel approach, TARM to increase the memory capacity of RMTs by implementing an exponential moving average on the memory tokens. Our approach allows for improved capture of longer-term dependencies without increasing the size of the memory tokens. We evalute the effectiveness of this approach on language modelling tasks and demonstrate that it outperforms the original RMT model on similar settings. We also show TARM makes the training much more stable.

1. Introduction

Recurrent Memory Transformers [1] have shown great potential in capturing long-term dependencies and general-purpose memory processing. However, the size of the memory tokens limits the memory capacity of these models, and increasing the size of these tokens can result in higher computational complexity. In this work, we propose a novel approach to address this limitation by implementing an exponential moving average on the memory tokens. The following are the key contributions:

1. We introduce a novel token based memory method called TARM that increases the capacity of the memory token without actually increasing the memory token size.

2. We show how TARM increases the stability of the training which enables faster convergence.

3. We evaluate the effectiveness of this approach on language modelling tasks and demonstrate that it outperforms the original RMT model on similar settings.

2. Methodology

Our proposed method involves using an exponentially weighted moving average which would stablize and smooth out the memory tokens over time. Incidently this would enable the model to access more information from previous segments without actually increasing the size of the memory tokens.

Let the memory tokens at time t be denoted by M_t, and the exponentially weighted memory tokens be denoted by E_t. The formula for updating the exponentially weighted memory tokens at time t is given by:

$$E_t = \alpha \times M_t + (1 - \alpha) \times E_{t-1} \tag{1}$$

where α is the weight used in the moving average, and E_{t-1} is the exponentially weighted memory tokens from the previous time step.

To initialize the exponentially weighted memory tokens, we can set E_0 to be equal to the initial memory tokens M_0. During training, the model uses the exponentially weighted memory tokens E_t instead of the original memory tokens M_t for processing and passing information between segments of the long sequence using recurrence.

3. Experimental setup

The experiments are designed in such a away to evalute the ability of the approach to preserve long-term dependencies across multiple segments.

Our implementation is based on [1] for the ease of reproducibility. WikiText-103 [3] experiments use 16-layer Transformers (10 heads, 410 hidden size, 2100 intermediate FF). We used Adam [2] optimizer with linear schedule learning rate starting from 0.00025 for 200,000 steps for WikiText-103. Vocabulary contains 267735 words for Wikitext-103 tokenizers. For Wikitext-103 an input context length was set to 160 tokens.

Experiments for Quadratic Equations task use 6 layer Transformers (6 heads, 128 hidden size, 256 intermediate FF). We used Adam [2] optimizer with constant learning rate 1e-4 with reduction on plateau with decay factor of 0.5.

Task	Dataset	Metric	RMT (From Paper)	RMT (Reproduced)
LM*	WT-103	PPL ↓	**23.99**	24.291 (↑0.301)
Step-by-Step	Quadratic Equations	Accuracy ↑	99.8	**99.9** (↑0.1)

Table 1. Comparison of reproduced RMT results on various tasks. The best results are shown in bold, and the second-best results are underlined. The symbol ↑ and ↓ indicate the improvement and degradation of the result compared to the reproduced RMT model. LM* denotes language modeling task. PPL denotes perplexity.

Task	Dataset	Metric	RMT (Reproduced)	TARM (Ours)
LM*	WT-103	PPL ↓	28.301	**24.493** (↓3.808)

Table 2. Results comparison of RMT and TARM on Language Modelling task. The best results are shown in bold, and the second-best results are underlined. The symbol ↑ and ↓ indicate the improvement and degradation of the result compared to the reproduced RMT model. LM* denotes language modeling task. PPL denotes perplexity.

Task	Dataset	Metric	α	PPL
LM*	WT-103	PPL ↓	0.2	**24.493**
LM*	WT-103	PPL ↓	0.7	24.603
LM*	WT-103	PPL ↓	1.0	28.301

Table 3. Ablation of the hyperparamter α. The best results are shown in bold, and the second-best results are underlined. The symbol ↑ and ↓ indicate the improvement and degradation of the result compared to the reproduced RMT model. LM* denotes language modeling task. PPL denotes perplexity.

For the reproduction of the RMT results, we train the model for 400,000 following [1]. Due to compute constraints, we were able to run TARM only for 120,000 steps, so we compare the corresponding best RMT result for the same number of steps.

3.1. Dataset

3.1.1 Language Modelling Task

WikiText-103 [3] is used for word-level language modeling and contains 103M words from English Wikipedia articles.

3.1.2 Quadratic equations Task

This dataset consists of equations with integer coefficients and step-by-step solutions using the discriminant. Process of equation generation is started from uniformly sampling real roots x_1, x_2 from -100 to 100. The answer of an equation is represented as x_1, x_2. Next, we find the equation as multiplication of two parentheses $(x - x_1)(x - x_2) = 0$, which is expanded to $x^2 - (x_1 + x_2)x + x_1x_2 = 0$. Next, we multiply all coefficients by a random natural number α from 1 to 10. The final equation form is $\alpha x^2 - \alpha(x_1 + x_2)x + \alpha x_1 x_2 = 0$. A dataset sample is made of these stages in reversed order. We also provide a string with the discriminant calculation to help find the equation roots. 20 percent of equations in the dataset do not have real roots.

4. Results

4.1. Reproduction

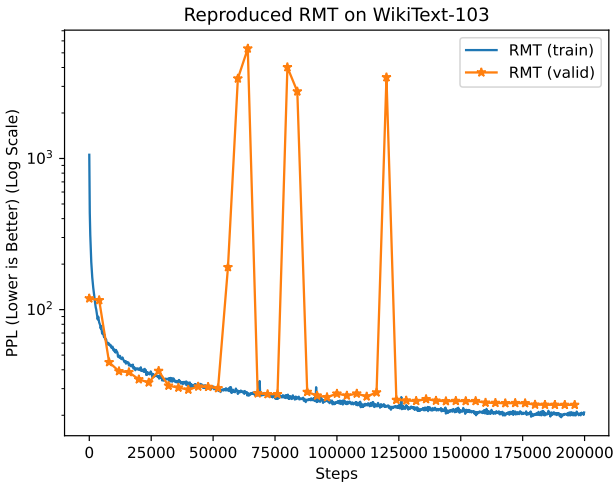

Figure 1. Reproduced step-wise perplexity score based on both training and validation sets. We use a log-scale for y-axis for better visibility of the lower scores which are obstructed by the outliers.

We begin by first reproducing the RMT results. As shown in Table 1, the reproduced result of LM task is 0.301 worse when compared to the results provided in [1]. For Step-by-Step task, the reproduced version performs slightly better. Since, difference in the reproduced results for both tasks is marginal, it is acceptable to establish that the RMT's results were successfully reproduced. We also show in figure 1 the reproduced training and validation scores across steps.

4.2. TARM

As mentioned in 2, due to compute constraints, we were able to train TARM only for 120,000. So we compare the

RMT's results which was trained for the same step count. As shown in 2, TARM performs very well when compared to RMT. The decrease in perplexity is 3.808. This reinforces our earlier claim that TARM enables the model to access more information from previous segments without actually increasing the size of the memory tokens.

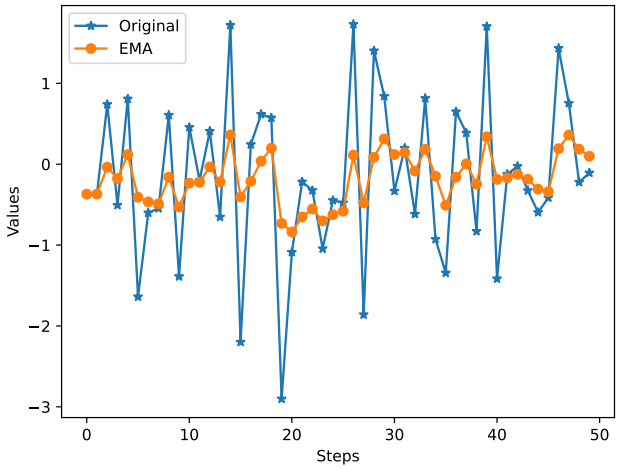

Figure 2. Demonstration of how the Exponential Moving Average (EMA) component used in TARM helps in stabilizing the erratic change in values.

Incidently, this also increases the stability of the training in general, leading to faster convergence. First we try to understand how the exponential moving average (EMA) in general improves the stability. From Figure 2, we can see that even though the original values is erratic and is jumping around randomly, with an $\alpha = 0.7$, the EMA has dampening the values down makes it smoother. This also translates to our use case exceptionally. From Figure 3, we can see that TARM is much smoother when compared to RMT over steps. TARM ($\alpha = 0.7$) converges much smoother and faster than anything else. While TARM ($\alpha = 0.2$) does seem erratic at the beginning, it too smoothen out towards the end of the training while also performing better.

5. Conclusion

Our proposed approach of using exponentially weighted memory tokens provides a simple yet effective way to increase the effective memory capacity of RMTs without increasing the size of the memory tokens. Our experiments demonstrate that this approach leads to significant improvements in the model's ability to capture and utilize long-term dependencies on language modeling tasks while simultaneously increasing the stability of training. We believe that this approach has the potential to be applied to other applications that require extensive memory processing and long-term dependencies.

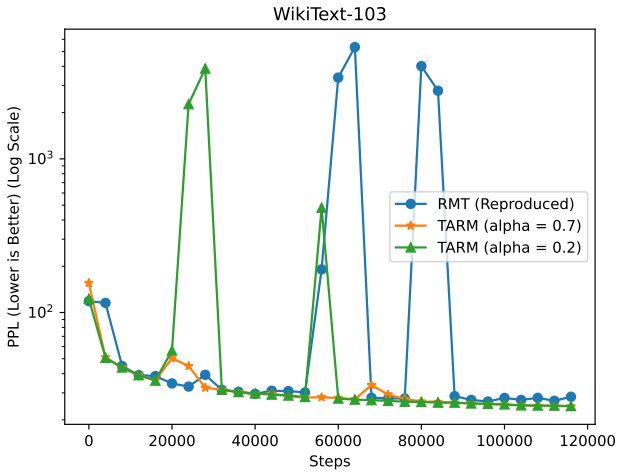

Figure 3. Figure shows how TARM stabilizes the validation accuracy over steps and also leads to faster/better convergence when compared to RMT.

As future work, we aim to substantiate the efficacy of TARM by conducting experiments on a variety of tasks and also show how the performance varies as we increase the number of segements.

References

[1] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022. 1, 2

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. 1, 2