Token Memory Transformer with Infinite Context

Taize Sun[1], Katsuhide Fujita[1](✉), Konstantin Markov[2], and Shengbo Chang[3]

[1] Tokyo University of Agriculture and Technology, Koganei, Japan
katfuji@cc.tuat.ac.jp
[2] University of Aizu, Aizuwakamatsu, Japan
[3] Tsinghua University, Beijing, China

Abstract. This study proposes an infinite context Transformer model based on Token Memory, which aims to solve the problem of contextual limitation in long text processing. The core of this model is Token Memory, which stores the context memory for each token and provides the information during model generation. The model first splits a long text into segments and then generates local and global attention for each segment. Local attention is generated by the decoder-only Transformer and is discarded after each segment is completed. Global attention is generated by the Token Memory. The Token Memory is retained to provide information for the infinite context, and it is updated after the current segment is calculated. Combining local and global attention can achieve an infinite context. We trained and evaluated the proposed model on the PG-19, C4-en, and BookSum datasets. Our model reaches state-of-the-art results on language modeling tasks with long contexts of up to 1M tokens. Our approach enables large language models to handle infinite contextual tasks, adapting to the increasing demands placed on these models by people.

Keywords: Large Language Model · Infinite Context · Memory · Fine-Tuning · Neural Networks

1 Introduction

With the introduction of ChatGPT [1], natural language processing (NLP) models have provided great convenience for people. These models help humans answer questions, generate suggestions, brainstorm, and perform other tasks. However, some tasks require the model to have long context processing capabilities, such as summarizing long documents or holding very long conversations that consider the context.

The most widely used model for NLP is the Transformer [2], but the attention mechanism in Transformers exhibits quadratic complexity regarding memory usage and computation time. Therefore, most Transformer-based models are unlikely to have long context or output lengths due to massive consumption [3].

One common approach to solving the above problems is to split long inputs into smaller segments and use memory, which is the segment-level memory transformer (c.f.

D.-S. Huang et al. (Eds.): ICIC 2025, LNAI 15865, pp. 319–330, 2025.
https://doi.org/10.1007/978-981-95-0020-8_27

Sect. 2). Memory is effective for humans [4] and NLP models [5]. It allows the model to process long contexts at a lower cost by storing and retrieving contextual information. Based on the above information, we propose a machine learning model that uses Token Memory, which aims to solve the problem of context limitation in long text processing.

Currently, most language model tokenizers come with a vocabulary whose main function is to convert tokens into integers for further processing. Our work focuses on processing and storing the information that tokens pass through the model. This information can be utilized for processing long contexts. Our model can be roughly divided into three components. **First**, like most current works [5], we split the long context into shorter segments and send them to the model one by one. The model uses decoder-only Transformers to calculate the local attention of the current segment. **Second**, unlike what was done in previous works [6], we prepare the memory for each token in the tokenizer vocabulary of the model, which we call Token Memory. When a token appears for the first time, the model uses the query, key, and value from the multi-head attention to calculate and save the memory for this token, which we call Corresponding Memory. If the token reappears in a later segment, the query and Token Memory are used to calculate its global attention and then update the Token Memory. This method builds a permanent and queryable Token Memory that is continuously updated. Based on this Token Memory, the model remembers information about the tokens over a long context. **Third**, we added Segment Memory to the model to summarize the previous segment [7]. It is used to supply the previous segment's information when the first token of the current segment is generated.

In the experiments, our model solves the long context modeling task with a maximum length of 1M tokens. The model achieves state-of-the-art performance in the language modeling task on the PG-19 [8] and C4-en [9] datasets and performs well in the summary task on the BookSum [10] dataset.

Our model mainly refers to infini-transformers [5], Memorizing Transformers [6], and LMEDR [7]. Compared with these models, our model is different in the following ways:

- Unlike Infini-attention, our model provides memory for each token instead of compressing the previous information into one matrix. When the token information is transmitted over long distances, it does not pass through the intermediate segment memories, and thus, the integrity of the information is preserved.
- Unlike Memorizing Transformers, our model considers the memory for all tokens. Therefore, our model does not discard any information about tokens and does not need to use kNN to search. Our model performs better with less memory.
- Unlike LMEDR, our model simplifies the two memory pools into one to accelerate the training and prediction speed when a large amount of memory is used. In addition, the memory is initialized and updated by the query, key, and value instead of randomly generated and backpropagated. This approach improves memory accuracy.

The contributions of this paper are summarized as follows:

- We propose a Transformer model based on Token Memory that can handle contexts of unlimited length. The Token Memory provides memory for each token and updates this memory when it processes lengthy texts, thereby enhancing the model's accuracy.

- Our proposed model was trained and tested on the PG-19 [8], C4-en [9] and BookSum [10] datasets. Our proposed model achieves state-of-the-art performance on long context language modeling and summarization tasks. The entire code for replicating the experiment is on https://github.com/Ozawa333/tokenmemory

2 Related Work

In this section, we discuss related segment-level memory models that can process long contexts.

Transformer-XL [11] solves the problem of context fragmentation by introducing a segment-level repetition mechanism and a new position encoding scheme, which enables the model to learn dependencies beyond a fixed-length context. Long-distance dependency modeling is achieved by sharing hidden states between different paragraphs, whereas relative position encoding is introduced to capture a wider range of context information. It is widely used in tasks that require long context understanding.

Compressive Transformer [12] models longer sequences by compressing past hidden states (memory). It inherits the idea of Transformer-XL but compresses rather than discards memories at each time step. It retains a longer context without losing important information; thus, it performs well in long text processing tasks.

Memorizing Transformers [6] allows models to acquire new knowledge by reading and memorizing new data. This approach uses an approximate k-nearest neighbor lookup to memorize and retrieve new information. It enhances the performance of the language models and enables them to update and expand their knowledge base continuously. In addition, it reduces the need for periodic retraining, thereby increasing the model's flexibility and adaptability. This method is especially beneficial for tasks that require dynamic updates to the knowledge base.

Recurrent Memory Transformer (RMT) [13] combines the advantages of the RNN [14] and the Transformer [2]. By introducing a memory mechanism to process long-sequence data, it captures both short-term and long-term dependencies, thereby improving the efficiency and effectiveness of long-sequence processing. It uses the memory capacity of RNN and the parallel computing capability of the Transformer to achieve efficient long-sequence dependency modeling, and it is widely used in tasks that require long-term dependency modeling.

AutoCompressors [15] reduces computing costs using automatic compression mechanisms while maintaining efficient performance. It significantly reduces the consumption of computing resources while maintaining the high-performance output of the model. Using automated compression methods optimizes the weights and structure of the model and reduces redundant calculations. AutoCompressors are suitable for tasks that require efficient computing.

Infini-Transformers [5] is designed to process sequences of infinite length; it can process ultralong sequences. It uses new positional encoding and attention mechanisms to ensure that the model can effectively capture and utilize infinitely long context information. It is suitable for tasks that require the processing of extremely long text sequences.

LMEDR [7] is not a Segment-Level Memory Transformer, but Entailment Relation Memory (ERM) in this work inspired our model. ERM is an external memory module that learns and stores the entailment relationships between premises and assumptions to ensure consistency in the conversation.

3 Method

Our model mainly maintains a Token Memory Pool for tokens. Each token in the tokenizer vocabulary has a Corresponding Memory in the Pool. A Corresponding Memory is calculated and updated when the model processes a token. Token Memory will provide additional information in long context generation. The structure of our model is shown in Fig. 1. On the left is the traditional Transformer decoder [2] that calculates local attention, and on the right is the memory module that we use to calculate global attention. First, we chunk the long input into segments and send the segments to the model in order. Second, we find the Corresponding Memory of the token and replace the memory of the first token with Segment Memory. Then, the Corresponding Memory is used to calculate global attention. After passing through a linear layer, we return global attention to the model and then plus it with local attention. Third, we calculate the memory update using the query, key, and value of the token, and then update the memory after the current segment calculation is complete. Fourth, we save the hidden states of the last token in the last layer as the Segment Memory and update after the current segment is calculated. After the current segment is complete, the model processes the next segment with memory. This section introduces each part of the model in detail.

3.1 Local Multi-head Attention Calculation

Our model is based on a decoder-only Transformer [2]. The local multi-head attention is calculated in each decoder layer. The local attention [11] is focused only on the current segment, and the local attention of the previous segment is discarded. Similar to most other Transformer-based models, our model calculates the query, key, and value states from the input $X \in \mathbb{R}^{N \times d_{model}}$ or the previous layer:

$$\mathrm{Q} = \mathrm{XW_Q}, \mathrm{K} = \mathrm{XW_K}, \mathrm{V} = \mathrm{XW_V} \tag{1}$$

where $Q, K, V \in \mathbb{R}^{N \times d_{model}}$, $W\_ \in \mathbb{R}^{d_{model} \times d_{model}}$ are trainable matrices, and N is the length of the input X. Then, we split the last dimension dmodel for the local multi-head attention calculation:

$$Q, K, V \in \mathbb{R}^{H \times N \times d_{head}} \tag{2}$$

where H is the number of attention heads and d_{head} is the dimension of the attention heads. Q, K, and V are used for the local multi-head attention calculation [2]:

$$A_{local} = softmax\left(\frac{QK^T}{\sqrt{d_{head}}}\right)V \tag{3}$$

where $A_{local} \in \mathbb{R}^{H \times N \times d_{head}}$ and K^T means transposing the last two dimensions of key K.

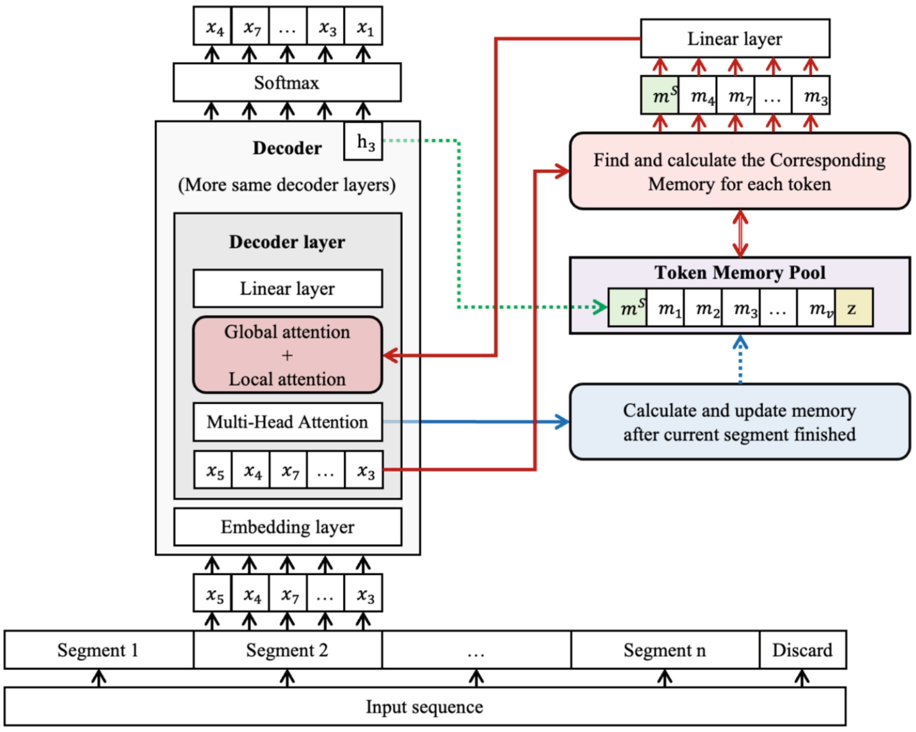
x_4 x_7 … x_3 x_1
Softmax
Decoder
h_3
(More same decoder layers)
Decoder layer
Linear layer
Global attention + Local attention
Multi-Head Attention
x_5 x_4 x_7 … x_3
Embedding layer
x_5 x_4 x_7 … x_3
Segment 1
Segment 2
…
Segment n
Discard
Input sequence
Linear layer
m^S m_4 m_7 … m_3
Find and calculate the Corresponding Memory for each token
Token Memory Pool
m^S m_1 m_2 m_3 … m_v z
Calculate and update memory after current segment finished

Fig. 1. The architecture of Token Memory Transformer.

3.2 Memory Initialization

Similar to previous work, such as the RNN [14] and LSTM [16], we maintain a recurrent memory state to track long contexts efficiently. Our model's memory usage increases with the number of different tokens that appear, but it never exceeds the size of the model tokenizer vocabulary. Our Token Memory is a variable-length parameterized matrix [17] that stores the Corresponding Memory of each token that has appeared in past segments. The model allocates Token Memory for all heads in every layer, defined as:

$$M \in \mathbb{R}^{l \times H \times v \times d_{head}} \tag{4}$$

where l is the number of layers and v is the number of tokens encountered thus far. The Token Memory M matrix is initialized to all zeros.

3.3 Global Memory Attention Calculation and Injection

We find the Corresponding Memory for each token in each layer, and we then replace the memory of the first token with Segment Memory. (Some elements will be defined in the following subsection because these values are initially zero).

$$M_{s-1}^{C} \leftarrow M_{s-1} \tag{5}$$

where $M_{s-1}^{C} \in \mathbb{R}^{H \times N \times d_{head}}$.We calculate the global attention:

$$A_{global} = \frac{M_{s-1}^{C}}{\sigma(Q) z_{s-1}} \tag{6}$$

where $A_{global} \in \mathbb{R}^{H \times N \times d_{head}}$, $z_{s-1} \in \mathbb{R}^{H \times d_{head} \times 1}$ is the normalization term [18], and σ is the element-wise ELU + 1 activation function [19]. Unlike previous work that used learned gating scalar [5], we pass A_{global} through a linear layer before feeding it back into the decoder for injecting global attention and mitigating potential issues arising from context turbulence [3]. Then, A_{global} and A_{local} are added together and merged from the d_{head} to d_{model}. Finally, we passed the above result through another linear layer to complete the current decoder layer.

$$O = \left(A_{global} W_A + A_{local}\right) W_O \tag{7}$$

where $O \in \mathbb{R}^{N \times d_{model}}$. $W_A \in \mathbb{R}^{H \times d_{head} \times d_{head}}$ and $W_O \in \mathbb{R}^{d_{model} \times d_{model}}$ are trainable matrices.

3.4 Memory Update

After the current segment calculation is complete, the new Corresponding Memory will be updated to the Token Memory for the token. Our approach is similar to that of previous work [5], as this work has demonstrated the superiority of this method. However, there are two differences. First, we utilize the position embeddings for the Q and K vectors that are used to calculate the Token Memory. Second, we directly use the current Q, K, and V of each token in the multi-head attention mechanism of each layer to compute the updated memory:

$$M_s \leftarrow M_{s-1}^{C} + \sigma(Q)(\sigma\left(K^T\right)V) \tag{8}$$

According to previous experience [18], the norm method is important to training stability, so we let the model sum K along the second dimension to compute and update the normalization term for the calculation of memory:

$$z_s \leftarrow z_{s-1} + \sum\nolimits_{n=1}^{N} \sigma(K_n) \tag{9}$$

where $z_s \in \mathbb{R}^{H \times 1 \times d_{head}}$. Then, we transpose the last two dimensions of the normalization term $z_s \in \mathbb{R}^{H \times d_{head} \times 1}$.

3.5 Segment Memory

This part uses a method similar to that idea in previous work [7], which uses Segment Memory to summarize the previous segment's information and to maintain contextual consistency. The Segment Memory comes from the hidden state $h^N \in \mathbb{R}^{1 \times d_{model}}$ of the final token of the final decoder layer output, which contains all the token information of the segment due to the structure of the decoder. Splitting the d_{model} dimension of the hidden state produces the Segment Memory.

$$M_s^S \leftarrow h_{s-1}^N \tag{10}$$

where $M_s^S \in \mathbb{R}^{H\times 1\times d_{head}}$. It is also updated to the memory pool after the current segment ends.

3.6 Loss Function

Our model converts all tasks to language modeling tasks. We used Teacher Forcing. Given $X = (x_1, x_2, \ldots x_N)$, the model predicts the next token based on the input tokens, so the loss function is:

$$loss = -\sum\nolimits_{n=1}^{N} P(x_n)\log P\left(x_n'|x_1, x_2, \ldots, x_{n-1}\right) \tag{11}$$

where x_n' is the output of the model, and $P(x)$ represents the probability distribution of x in the vocabulary as calculated by a linear layer.

4 Experiment

We modified a large language model using the Sect. 3 method to address long context language modeling tasks. We trained and evaluated the model using long context datasets and compared the model with other models.

4.1 Datasets

We selected texts having an appropriate length from the following datasets for training and evaluation.

- PG-19 [8] is a language modeling benchmark created by DeepMind. It consists of a collection of books extracted from the Project Gutenberg library, specifically those published before 1919. Commonly used for model evaluation with long contexts. We used its train split with 13.7K texts for the training, and we used its test split with 100 texts for evaluation.
- C4 [9] dataset is a massive, cleaned version of the Common Crawl web crawl corpus. It was used to train the T5 text-to-text Transformer models. We use its train split with 16K texts for training and test split with 3.6K texts for evaluation on the en subset.
- BookSum [10] is a dataset that was designed for long-form narrative summarization. It aims to address the limitations of most current text summarization datasets. BookSum contains source texts from the literary domain, such as novels, plays, and stories, as well as highly abstract human-written summaries. We used its train split with 330 texts for training and its test split with 25 texts for evaluation.

4.2 Implementation Details

We trained our model based on **Sheared-LLaMA-1.3B** [20]. This model was pruned and further pre-trained from Llama2 [21] by removing layers, heads, and intermediate hidden dimensions. Data from different domains in the RedPajama [22] dataset were dynamically loaded to continue pre-training the model. This model had 16 attention heads of dimension 128, 24 hidden layers, and 2048 model dimensions. The maximum input length of the model was 4096, so we also set the segment length to 4096.

We set the input length to (32K, 512K, 1M), which was (8, 128, 256) times the segment length. We tested the learning rates (1e-2, 1e-3, 1e-4) using a decay cosine scheduler and gradient accumulation steps (8, 128, 256) with a batch size of one. We used cross-entropy to calculate the loss and Adafactor [23] to optimize the model. The average full parameter fine-tuning time was 12 h with Pytorch on three NVIDIA RTX A6000 Ada Generation. Algorithm 1 is our training algorithm.

4.3 Language Modeling Task

Our model was trained for about 23K steps on PG-19 [8] and C4-en [9]. We used perplexity to evaluate the model. If a model is confident in predicting many words (high probability), its perplexity will be low; conversely, if a model is unsure about the prediction of a word (low probability and wide distribution), its perplexity will be high. Due to the insufficient length of C4-en, we were not able to train and test the results of 512K and 1M input lengths. We found that the model performed best when the gradient accumulation steps were 8 and the learning rate was 1e-3.

Table 1 shows the evaluation results for our model compared with other segment-level language models. On PG-19 [8], our model achieves state-of-the-art results for all input lengths. This demonstrates the effectiveness of the Token Memory in handling long context language modeling tasks. On C4-en [9], we tested only the results of 32K input length due to insufficient data length, and our model again achieved state-of-the-art results.

4.4 Summarization Task

We concatenated the chapters and summaries in BookSum [10] into a new text by using the keyword "Summarization:". In addition, due to data length limitations, we selected new texts having lengths between 32K and 64K, and we then truncated them to 32K for the training and evaluation. Our model was trained for 330 steps. We use ROUGE [24] to evaluate the models. The ROUGE evaluation mainly measured the model's performance by comparing the similarity between the generated text and the reference text. Common ROUGE indicators include ROUGE-N, which calculates the number of identical n-grams. ROUGE-L evaluates the similarity based on the longest common subsequence. Our model performed best at a learning rate of 1e-3 with 128 gradient accumulation steps.

```
Algorithm 1: Training detail
    Input: The dataset for language modeling
    Parameter: LLaMA with Token Memory
1:  for texts in dataset do
2:      for segment in texts do
3:          Calculate $A_{local}$ by Eq. (3)
4:          Search $M^C_{s-1}$ and $M^S_{s-1}$ by Eq. (5)
5:          Calculate $A_{global}$ by Eq. (6)
6:          Assemble $A_{local}$ and $A_{global}$ for output by Eq. (7)
7:          Update $M$ and $z$ by Eq. (8), Eq. (9), and Eq. (10)
8:          Backpropagation by Eq. (11)
```

Table 1. Language modeling results comparison of our model and others on PG-19 and C4-en. All the models used the optimal settings. Some data come from [5].

Model	Input length	PG-19	C4-en
Transformer-XL [11]	-	11.88	-
Memorizing Transformers [6]	-	11.37	13.64
RMT [13]	8k	13.27	-
Infini-Transformer [5]	32k	9.65	-
Ours	32k	**9.35**	**4.89**
	512k	**8.83**	-
	1m	**8.04**	-

Table 2 shows a comparison with other segment-level language models. Even when compared with a model that is designed specifically for summarization [25, 26] with a long context extension [0], our model takes the lead on some indicators. However, when it is compared with the latest segment-level Memory Transformer Infini-Transformer [5], it only takes the lead on ROUGE-1. This suggests that although the model is good at finer details, it may require further optimization to improve its ability to capture the overall gist of the text and the more complex relationships within the text.

4.5 Ablation Study

Table 3 shows the performance of the model on PG-19 using different context lengths. As the input length increases, the model performs better, which shows that as the tokens that have appeared gradually fill the memory, the performance of the model improves. In addition, although the input length increases by 16-fold and 32-fold for 32K input length, the increase of memory is only 2.8-fold and 3.1-fold. The above result indicates

Table 2. Summarization result comparison of our and others on BookSum. Some data come from [5].

Model	ROUGE-1	ROUGE-2	ROUGE-L
BART + Unlimiformer[25]	36.8	8.3	15.7
PRIMERA + Unlimiformer[26]	37.9	8.2	16.3
Infini-Transformer[5]	40.0	**8.8**	**17.9**
Ours	**43.2**	7.6	16.7

that many words are repeated in the dataset, which enhances the advantage of our model as the input length increases.

Table 3. Language modeling results of different context lengths on PG-19.

Context length	Memory pool	Token Perplexity
32k	3857	9.35
512k	10837	8.83
1M	12135	8.04

In the summarization task, the result in Table 4 shows that removing Token Memory and Segment Memory decreased almost all the performance. This indicates that Token Memory and Segment Memory play a crucial role in improving the detail of the generated summaries and maintaining contextual consistency. When disabling the Token Memory, the increase of ROUGE-2 scores may suggest that our model focuses too much on single token information and loses the overall information of the sentence.

Table 4. Summarization results of our model including w/o memory on BookSum.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
Original	**43.29**	7.64	**16.74**	**41.57**
w/o Token Memory	39.65	**7.66**	15.91	37.97
w/o Segment Memory	36.26	6.89	16.33	34.61

5 Conclusion

In this study, we introduced an innovative infinite context Transformer model based on Token Memory to overcome the limitations of traditional Transformer models in processing long texts. Our approach effectively combined local and global attention mechanisms, which then enabled the model to process texts of unlimited length while maintaining high performance. Through experiments on the PG-19, C4-en, and BookSum datasets, our model demonstrated state-of-the-art performance and excelled particularly on tasks that involved long contexts of up to 1M tokens.

Our model's ability to retain and update memory for each token ensured that no information was discarded, thereby improving the accuracy of long context language modeling. This advancement introduces new possibilities for applications that require extensive contextual understanding, such as document summarization and long-form content generation.

Our future work will focus on further optimizing the model's memory management and exploring its application in more diverse NLP tasks.

Acknowledgments. This research was supported by a tuition exemption from the Tokyo University of Agriculture and Technology.

Disclosure of Interests. The authors have no conflicts of interest to declare.

References

1. Liu, Y., et al.: Summary of chatgpt-related research and perspective towards the future of large language models. Meta-Radiol., 100017 (2023)
2. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inform. Process. Syst. **30** (2017)
3. Brown, T.B.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
4. Collins, A.M., Quillian, M.R.: Retrieval time from semantic memory. J. Verbal Learn. Verbal Behav. **8**(2), 240–247 (1969). https://doi.org/10.1016/S0022-5371(69)80069-1, https://www.sciencedirect.com/science/article/pii/S0022537169800691
5. Munkhdalai, T., Faruqui, M., Gopal, S.: Leave no context behind: Efficient infinite context transformers with infini-attention. arXiv preprint arXiv:2404.07143 (2024)
6. Wu, Y., Rabe, M.N., Hutchins, D., Szegedy, C.: Memorizing transformers. arXiv preprint arXiv:2203.08913 (2022)
7. Chen, R., Wang, J., Yu, L.C., Zhang, X.: Learning to memorize entailment and discourse relations for persona-consistent dialogues. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 12653–12661 (2023)
8. Rae, J.W., Potapenko, A., Jayakumar, S.M., Hillier, C., Lillicrap, T.P.: Compressive transformers for long-range sequence modelling. arXiv preprint (2019), https://arxiv.org/abs/1911.05507
9. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv e-prints (2019)
10. Kryściński, W., Rajani, N., Agarwal, D., Xiong, C., Radev, D.: Booksum: A collection of datasets for long-form narrative summarization (2021)
11. Dai, Z.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
12. Rae, J.W., Potapenko, A., Jayakumar, S.M., Lillicrap, T.P.: Compressive transformers for long-range sequence modelling. arXiv preprint arXiv:1911.05507 (2019)
13. Bulatov, A., Kuratov, Y., Burtsev, M.: Recurrent memory transformer. Adv. Neural. Inf. Process. Syst. **35**, 11079–11091 (2022)
14. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)

15. Chevalier, A., Wettig, A., Ajith, A., Chen, D.: Adapting language models to compress contexts. arXiv preprint arXiv:2305.14788 (2023)
16. Graves, A., Graves, A.: Long short-term memory. Supervised sequence labelling with recurrent neural networks, pp. 37–45 (2012)
17. Schlag, I., Munkhdalai, T., Schmidhuber, J.: Learning associative inference using fast weight memory. arXiv preprint arXiv:2011.07831 (2020)
18. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are rnns: fast autoregressive transformers with linear attention. In: International Conference on Machine Learning, pp. 5156–5165. PMLR (2020)
19. Clevert, D.A.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
20. Xia, M., Gao, T., Zeng, Z., Chen, D.: Sheared llama: accelerating language model pre-training via structured pruning. arXiv preprint arXiv:2310.06694 (2023)
21. Touvron, H., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
22. Computer, T.: Redpajama: An open source recipe to reproduce llama training dataset (2023). https://github.com/togethercomputer/RedPajama-Data
23. Shazeer, N., Stern, M.: Adafactor: adaptive learning rates with sublinear memory cost. In: International Conference on Machine Learning. pp. 4596–4604. PMLR (2018)
24. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out, pp. 74–81 (2004)
25. Lewis, M.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
26. Xiao, W., Beltagy, I., Carenini, G., Cohan, A.: Primera: pyramid-based masked sentence pre-training for multi-document summarization. arXiv preprint arXiv:2110.08499 (2021)
27. Bertsch, A., Alon, U., Neubig, G., Gormley, M.: Unlimiformer: long-range transformers with unlimited length input. Adv. Neural Inform. Process. Syst. **36** (2024)