Moscow Institute of Physics and Technology
(National Research University)
Phystech-School of Applied Mathematics and Informatics
Master's Program
«Methods and Technologies of Artificial Intelligence»

Matvey Kairov

Efficient architectures for long context and reasoning

03.04.01 — Applied Mathematics and Physics

Literature Review

Supervised by:
Yuri Kuratov, PhD

Moscow

2024

Contents

1 Introduction **4**

2 Problem statement **5**

3 Literature review **7**

3.1 Sparse Attention 7

3.1.1 Longformer 8

3.1.2 Big Bird 9

3.2 Kernelized Attention 10

3.2.1 Linformer 10

3.2.2 ∞-former 11

3.3 Segment-level recurrence 11

3.3.1 Transformer-XL 12

3.3.2 Memformer 13

3.3.3 Recurrent Memory Transformer 13

3.3.4 Associative Recurrent Memory Transformer 14

3.4 Structured State-Space Models 15

3.4.1 Mamba 16

4 Conclusion **17**

Abstract

Transformer-based models show their effectiveness across multiple domains and tasks. The self-attention mechanism, which is the key part of the Transformer architecture, enables the generation of rich and context-aware representations of the processed sequence, but suffers from a quadratic computational complexity. This drawback becomes very apparent when Transformer-based models process very long sequences, which causes decreasing accuracy and inefficient memory management. This paper aims to establish an optimization task for long-context language modeling along with compiling existing research on the matter in order to choose the approach to long-context language modeling that is both the most relevant and most promising.

1 Introduction

Transformers [1] have made a significant impact on natural language processing (NLP) by enabling models to efficiently solve a multitude of tasks with state-of-the-art performance. The key component of the Transformer is the self-attention mechanism, which allows the model to integrate information from the entire sequence to each of its tokens, capturing both short-term and long-term dependencies and generating rich and context-aware representations of the sequence. Due to this effectiveness, Transformers have become the architecture of choice for many applications. Nevertheless, the attention mechanism has a number of notable limitations, especially concerning its handling of longer sequences.

The key challenge presented by the attention mechanism in this respect is complexity, which scales quadratically with the length of the input sequence. The result of this is a fast degradation of performance when handling longer sequences, especially connected to the usage of memory and computation time. This means that while traditional Transformers are highly effective on short and medium-length sequences, tasks like processing full-length documents, books, and time-series data require an unreasonably large amount of computations.

In response to these challenges, a number of approaches emerged, focused on the improvement of the efficiency of the attention operation. Although these approaches vary vastly, it is possible to broadly categorize them into three categories: sparse attention, kernelized attention, and memory-augmented recurrent attention. Each approach seeks to balance the model's ability to capture long-range dependencies with the need for computational efficiency, offering unique trade-offs in terms of scalability, accuracy, and memory usage.

In this review, we set for ourselves the following goals:

1. Review the existing approaches to optimizing the performance of transformer models on long sequences.

2. Evaluate the benefits and drawbacks of each such approach.

3. Determine which approach is both the most promising and the most relevant to the problem of handling long sequences.

2 Problem statement

The goal of long context language modeling is to train a model that predicts the next token x_n in a sequence $x = (x_1, x_2, \ldots, x_n)$, based on the context of previous tokens $x_1, x_2, \ldots, x_{n-1}$. This task involves capturing long-range dependencies to effectively model sequences with extended histories.

Given a sequence $x = (x_1, x_2, \ldots, x_N)$ from a corpus, formally, the goal is to optimize the parameters θ by maximizing the following log-likelihood over a set of training sequences $\{x^{(i)}\}$:

$$\mathcal{L}(\theta) = \sum_{i=1}^{M} \sum_{t=1}^{N_i} \log p_\theta \left(x_n^{(i)} | x_1^{(i)}, \ldots, x_{t-1}^{(i)} \right), \tag{2.1}$$

where M represents the number of sequencces in the corpus and N_i represents the length of the i-th sequence.

The parameters θ of the model are typically optimized using gradient-based optimization methods, such as Adam. The gradient of the log-likelihood with respect to θ is computed and used to update the model parameters iteratively:

$$\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \mathcal{L} \left(\theta^{(t)} \right), \tag{2.2}$$

where:

- $\theta^{(t)}$ are the model parameters at iteration t,
- η is the learning rate,
- $\nabla_\theta \mathcal{L} \left(\theta^{(t)} \right)$ is the gradient of the log-likelihood with respect to the parameters θ.

As a whole, given a set of training sequences $\{x^{(i)}\}$, the training objective is to learn the parameters θ^* of the language model such that:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{M} \sum_{t=1}^{N_i} \log p_\theta \left(x_n^{(i)} | x_1^{(i)}, \ldots, x_{t-1}^{(i)} \right). \tag{2.3}$$

In essence, the optimization objective for language modeling is the same for both short and long contexts, with the primary difference being the scale of sequence length N. However, the task of handling long sequences introduces specific challenges. The model must be able to retrieve relevant information from earlier parts of the sequence, process

long-term dependencies, and handle large input contexts. In short contexts, models can attend to all tokens in the sequence at once, making all dependencies local. However, with longer contexts, fully attending to the entire sequence becomes increasingly impractical due to rising computational complexity and decreasing memory efficiency.

The inefficiency of Transformers comes from the self-attention operation. Attention is defined as:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V, \tag{2.4}$$

where Q, K and V (representing query, key and value), are all equal to the input sequence. For a sequence of length N and a model dimension d_k, this operation would require at least $N^2 \cdot d_k$ operations. Therefore, the computational complexity of the self-attention operation grows quadratically with the sequence length N, which can be expressed as $O(N^2)$. A major consideration for the current research are the directions for improving the computational and memory complexity of Transformer models through modifications to the attention mechanism.

Another consequence of increased context length is the different approaches to benchmarking a model's performance on long sequences. One such approach involves the **needle-in-a-haystack** task [2]. It can be described as finding relevant facts that are hidden within a vast corpus of information, requiring the model to perform complex reasoning across parts of the input thousands of tokens apart from each other. The implementations of these tasks can be found in benchmarks such as BABILong [3].

In general, a model capable of handling long sequences should be able to satisfy the following conditions:

- Memory and Long-Range Dependency Capture: Standard attention mechanisms are limited in their ability to capture dependencies across long sequences. While self-attention can encode relationships within a fixed context window, maintaining global dependencies over longer sequences requires significant memory and computational resources. The model should be able to take advantage of the ability of the attention mechanism to capture long-range dependencies without sacrifices in terms of scalability and efficiency.

- Balance between Memory and Efficiency: The model should not trade off memory efficiency for accuracy and vice versa. Approaches such as sparse attention, kernelized attention, and recurrent attention models offer various trade-offs between

memory usage, computational efficiency, and the ability to capture long-range dependencies. A key challenge is to design a system that balances these factors while maintaining high performance across diverse tasks.

The challenge, therefore, is to design a model that can efficiently manage both short-term dependencies, critical for language fluency, and long-term dependencies, which are key for understanding and retrieving relevant context in large sequences. The model must maintain contextual understanding and memory management across long sequences without sacrificing computational efficiency. This includes retaining relevant information from earlier parts of the sequence and retrieving it as needed to solve tasks that require reasoning over extended contexts.

3 Literature review

3.1 Sparse Attention

The techniques known as **Sparse Attention** can be summarized as methods of decreasing the computational complexity of the attention mechanism by reducing the share of the sequence that each token attends to. The key concepts, most frequently utilized in sparse attention include:

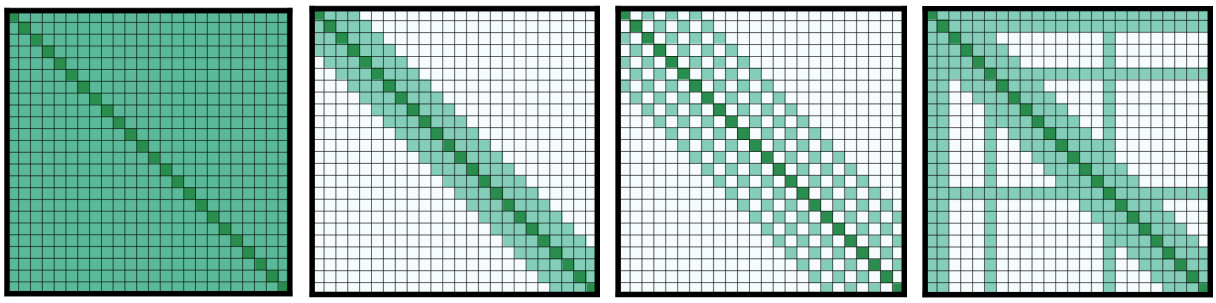
- **Local attention**: Tokens only attend to a relatively small window of neighboring tokens, reducing the total number of token pairs involved in computing attention. These tokens insure the capture of local dependencies.

- **Global attention**: Some special tokens are able to attend to all other tokens in the sequence, ensuring the capture of long-range dependencies.

- **Attention patterns**: Attention patterns determine which parts of the sequence each token attends to. These include both local and global attention and other types of relations between tokens such as random attention. Attention patterns can be fixed (as in the case of local and global attention), which greatly increases the computational efficiency, but may lack the ability to capture some complex relationships between distant tokens. On the other hand, learned patterns require more computations, but the resulting patterns may be better suit for the task the model is trained to solve.

There is a number of methods that utilize sparse attention, of which we chose the following three.

3.1.1 Longformer



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 1: Longformer attention patterns [4].

Longformer [4] is a transformer model designed to address the inefficiencies of traditional self-attention mechanisms when processing long sequences. Standard transformers, which rely on a full attention mechanism, scale quadratically with respect to sequence length ($O(N^2)$), limiting their ability to handle long inputs due to the high computational and memory costs. Longformer mitigates this issue through the introduction of a sparse attention mechanism, enabling it to process much longer sequences efficiently.

Longformer employs a hybrid attention mechanism that combines local attention and global attention. In local attention, each token attends only to a fixed window of neighboring tokens, reducing the number of interactions and thus the computational cost. This results in an attention complexity of $O(N \cdot w)$, where w is the size of the local context window. The model also incorporates global attention, where selected tokens (such as the [CLS] token in classification tasks) attend to the entire sequence. This allows the model to capture global dependencies across the entire sequence, even when the token is far removed from others.

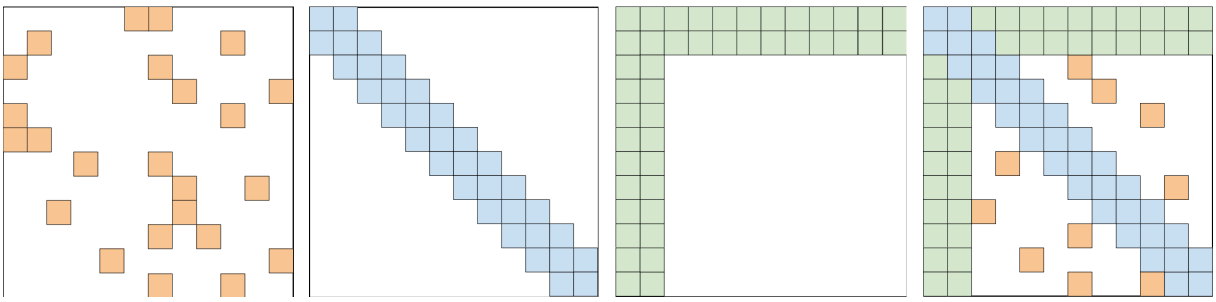
The combination of local and global attention strikes a balance, enabling Longformer to maintain computational efficiency while also preserving the ability to capture long-range dependencies.

The sparse attention mechanism allows Longformer to process sequences much longer than traditional transformers, which would be computationally infeasible due to their quadratic complexity. By attending only to a subset of tokens, both memory and com-

putational costs are drastically reduced. The model scales efficiently to long sequences, processing inputs that would be impractical for standard transformers, such as entire documents or long passages of text.

This reduction in computational complexity is a significant advantage for tasks involving long-range dependencies. Longformer's attention mechanism operates in linear time ($O(N)$), which allows for both efficient training and inference when working with large inputs.

3.1.2 Big Bird



(a) Random attention (b) Window attention (c) Global attention (d) BigBird

Figure 2: BigBird attention patterns [5].

Big Bird [5] is another transformer model designed to efficiently handle long sequences by addressing the limitations of standard self-attention, which becomes computationally expensive as the sequence length grows. Big Bird incorporates a sparse attention mechanism to reduce the quadratic complexity of self-attention, making it feasible to process long sequences while still maintaining the model's ability to capture long-range dependencies.

Big Bird introduces a hybrid sparse attention mechanism that combines local attention, global attention, and random attention:

- Local Attention: Similar to Longformer, Big Bird uses a sliding window mechanism for local attention, where each token attends only to tokens within a fixed window. This reduces the number of attention pairs significantly compared to traditional transformers.

- Global Attention: Some tokens, like the [CLS] token or other important tokens in the input, attend to all other tokens in the sequence. This ensures that critical context can be captured across the entire sequence, maintaining global dependencies.

- Random Attention: A key innovation in Big Bird is the use of random attention, where tokens are randomly selected to attend to each other. This helps the model capture long-range dependencies that might not be covered by local or global attention alone.

The combination of these attention types reduces computational complexity while still allowing Big Bird to model complex dependencies across the sequence.

By using sparse attention, Big Bird reduces the attention complexity from $O(N^2))$ to $O(N \cdot \sqrt{N})$, significantly improving both memory efficiency and computational speed. This makes Big Bird suitable for processing sequences that are far longer than those manageable by traditional transformers, such as long documents, full-text corpora, and large time-series data.

3.2 Kernelized Attention

Kernelized attention is an approach designed to approximate the traditional self-attention mechanism in transformers in order to reduce its computational complexity with respect to the sequence length. Kernelized attention replaces the standard dot-product attention with an approximation using kernel methods, enabling more efficient computation, especially when handling long sequences. This approach lends a number of benefits to the Transformer architecture, namely the linear or near-linear computational complexity, however one of its notable drawbacks is the lower precision, which is unavoidable with most approximations. For this review, we considered three characteristic approaches, each representing a distinct direction in kernelized attention methods.

3.2.1 Linformer

Linformer [6] is a transformer architecture designed to efficiently process long sequences by addressing the high computational and memory costs associated with traditional self-attention. The central innovation of Linformer is the use of low-rank matrix factorization to approximate the attention matrix, reducing the time and memory complexity from $O(N^2)$ to $O(N \cdot k)$, where k is a fixed, much smaller rank. This makes it feasible to process much longer sequences without incurring the prohibitive computational overhead of standard transformers.

The main limitation of Linformer lies in the approximation of the attention mechanism. By using a low-rank approximation, the model may lose fine-grained dependencies

between distant tokens, which could affect tasks requiring intricate interactions across the sequence. Additionally, while the low-rank approximation reduces the computational burden, it introduces some degree of approximation error, potentially reducing performance on tasks that require precise attention.

3.2.2 ∞-former

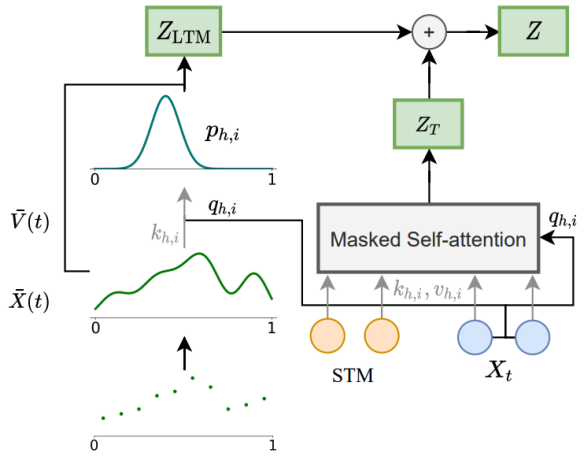


Figure 3: ∞-former's attention diagram [7].

∞-former, or Infinite Memory Transformer [7], implements a continuous-space attention mechanism, which enables the model to maintain an unbounded memory capacity. This architecture employs a set of basis functions to represent memory, allowing the model to update and retrieve information dynamically as new data is processed. Specifically, ∞-former utilizes 512 basis functions to facilitate the storage and retrieval of long-term memories, thereby enhancing the model's ability to recall relevant information from previous interactions.

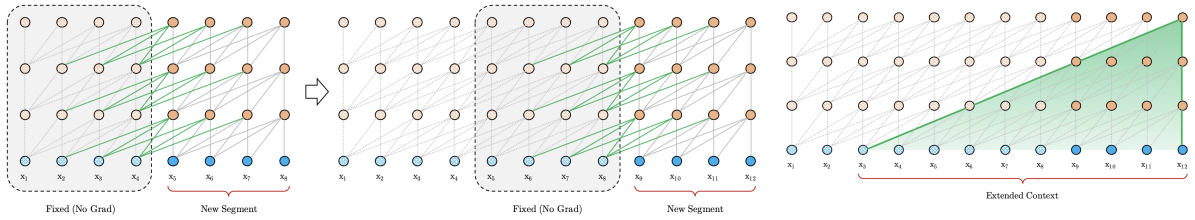
∞-former's novel approach allows for $O(n)$ computational complexity and, theoretically, unlimited memory storage with a complexity of $O(1)$ thanks to its continuous attention mechanism, however this memory is limited by floating point precision.

3.3 Segment-level recurrence

Another way of addressing limitations presented by Transformers was proposed in the concept of segment-level recurrence. It is implemented by partitioning the input sequence into manageable segments and introducing mechanisms to retain and utilize information across these segments. This approach effectively reduces computational overhead while

preserving the model's ability to capture long-range dependencies. By recursively inputting segments to the model, the model maintains contextual continuity without the necessity to attend to the entire sequence simultaneously. This memory can be implemented through hidden states or explicit memory vectors that are updated as new segments are processed. Consequently, the model achieves a balance between computational efficiency and the retention of essential contextual information across segments.

3.3.1 Transformer-XL



(a) Training phase.

(b) Evaluation phase.

Figure 4: Illustration of the Transformer-XL model with a segment length 4 [8].

Transformer-XL [8] represents a significant advancement in addressing the limitations of traditional transformer models concerning long-term dependency modeling and computational efficiency. The architecture extends the conventional transformer by incorporating a segment-level recurrence mechanism, enabling the model to capture dependencies beyond the fixed-length context window. This is achieved through a novel recurrence strategy that maintains hidden states across segments, thereby facilitating the flow of information over longer sequences without incurring the quadratic complexity typically associated with self-attention mechanisms.

A key innovation in Transformer-XL is the introduction of the relative positional encoding scheme. Unlike absolute positional encodings used in standard transformers, relative positional encodings allow the model to generalize across different segment lengths and positions by encoding the relative distances between tokens. This enhances the model's ability to capture positional relationships dynamically, improving its performance on tasks requiring an understanding of long-term dependencies. The relative positional encoding also contributes to better scalability, as it reduces the dependency on fixed-length segments and allows the model to adapt to varying sequence lengths more effectively.

Overall, Transformer-XL's integration of segment-level recurrence and relative posi-

tional encoding marks a substantial improvement in transformer architecture, enabling more effective processing of long sequences. Its ability to maintain contextual information across segments without excessive computational costs makes it a valuable model for various natural language processing tasks that demand the handling of extensive and complex textual data.

3.3.2 Memformer

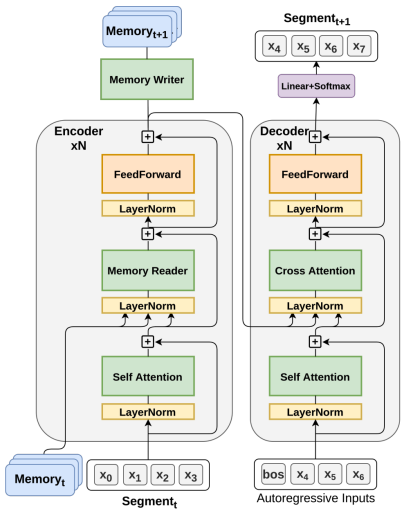


Figure 5: Memformer

Memformer [9] introduces an innovative approach to enhancing transformer architectures through the integration of external memory mechanisms. The model aims to address the challenges associated with processing long sequences by providing a persistent memory component that stores and retrieves relevant information across different segments. This memory-augmented architecture allows Memformer to maintain and utilize contextual information over extended sequences without solely relying on the transformer's inherent self-attention mechanism.

The core idea behind Memformer is to decouple the memory storage from the transformer's computational graph, enabling efficient access and updates to the memory content. This separation allows the model to retrieve pertinent information from the memory based on the current input segment, thereby facilitating the retention of long-term dependencies. The memory is structured in a way that supports both read and write operations, ensuring that the model can dynamically incorporate new information while preserving essential contextual details from previous segments.

Memformer is able to achieve $O(N)$ computational comlexity.

3.3.3 Recurrent Memory Transformer

The Recurrent Memory Transformer (RMT)[10] extends the framework of Transformer-XL by implementing a segment-level attention mechanism without global memory. The key difference between RMT and Transformer-XL is their handling of memory tokens. In Transformer-XL, the input of the layer n consists of the last m states from the cached

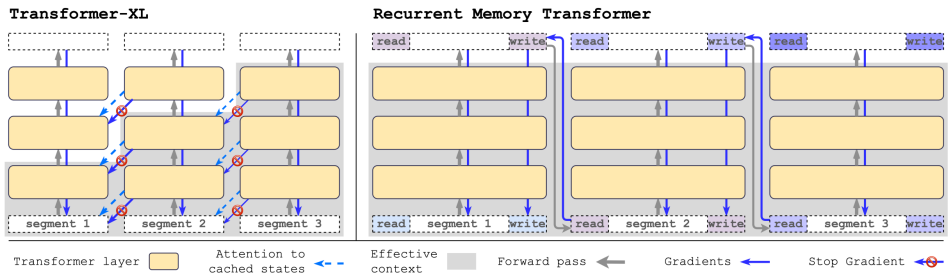


Figure 6: Comparison of Recurrent Memory Transformer (RMT) and Transformer-XL architectures [10].

memory and output of the previous Transformer layer for the current segment τ:

$$\widetilde{H}_{\tau}^{n-1} = \left[SG(M_{-m:}^{n-1} \circ H_{\tau}^{n-1}\right], \tag{3.1}$$

where SG stands for Stop Gradient and $\circ$ denotes concatenation. On the other hand, in Recurrent Memory Transformer the input is augmented with special **[mem]** tokens, tokens, processed in a standard way along with the sequence of tokens. Each memory token is a real-valued vector. m memory tokens are added at the beginning of the segment tokens representations H_{τ}^{0} and the sae m tokens are added at the end:

$$\widetilde{H}_{\tau}^{0} = \left[H_{\tau}^{mem} \circ H_{\tau}^{mem} \circ H_{\tau}^{0}\right], \ \bar{H}_{\tau}^{N} = \text{Transformer}\left(\widetilde{H}_{\tau}^{0}\right), \ \left[H_{\tau}^{read} \circ H_{\tau}^{N} \circ H_{\tau}^{write}\right] := \bar{H}_{\tau}^{N}, \tag{3.2}$$

here N is the number of Transformer layers.

Both memory and recurrence in the RMT are based only on global memory tokens. It allows to use a pre-trained backbone model which can be kept unchanged and make RMT memory augmentation compatible with any model from the Transformer family.

RMT's computational complexity is $O(N)$ and it's memory usage is as small as the memory required for the recurrent memory cell.

3.3.4 Associative Recurrent Memory Transformer

The Associative Recurrent Memory Transformer (ARMT) extends the general idea behind the Recurrent Memory Transformer by including an associative memory mechanism in the form of the Associative Block, as seen in Figure 7c. In the ARMT, each layer l processes input segments X_s^l and updates the memory tokens M_{s-1}^l from the previously processed segment by combining them with the output of the current associative block (as seen in Figure 7b). The updated memory and sequence embeddings are then fed to the

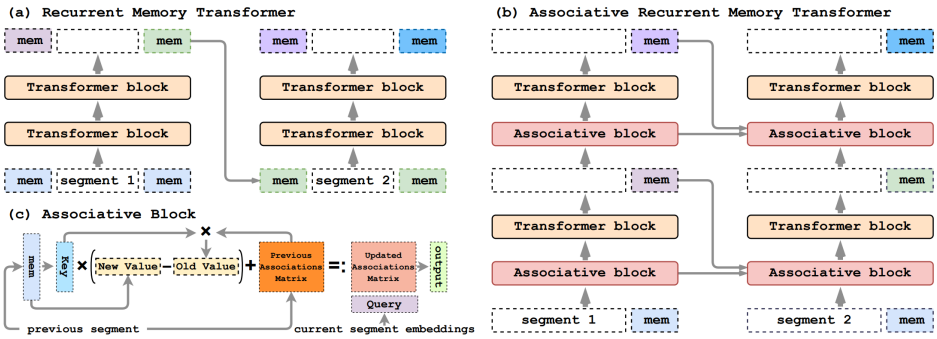


Figure 7: RMT, Associative Block and ARMT [11].

next layer, where the associative memory continues to evolve over the course of processing multiple segments.

The associative block is similar in principle to the Linformer [6], but with a focus on memory tokens. After each segment, memory tokens are transformed into keys and values using a linear mapping and stored in a key-value memory. This memory is updated through a process of associating the current memory tokens with their respective keys and updating the values accordingly. This process ensures that the model can recall and modify previous memory associations, enabling it to maintain context across long sequences. The memory updates are driven by importance scalars β_i, which control the weight of the new value added to the memory, and a non-linearity function φ is used to compute the associations.

The ARMT maintains all of the benefits that came with the RMT, including the constant memory requirements and linear complexity of computations, however it far surpasses RMT in the needle-in-a-haystack task, namely outperforming RMT on the BABILong benchmark.

3.4 Structured State-Space Models

Structured State-Space Models (SSMs) are mathematical frameworks used to model dynamic systems that evolve over time. SSMs describe the evolution of a system's state (the set of variables that capture the system's behavior) through a set of linear or non-linear equations. The state is typically updated over time based on system dynamics, with both observations and control inputs influencing the system's evolution.

SSMs introduce a number of key concepts:

- **State Transition**: The system's state evolves over time according to a state tran-

sition function, often represented as

$$x_t = Ax_{t-1} + Bu_t + w_t, \tag{3.3}$$

where x_t is the state at time t, A is the state transition matrix, u_t is the control input, and w_t is process noise.

- **Observation Model**: The observed data (or outputs) are typically related to the state through an observation model, such as

$$y_t = Cx_t + v_t \tag{3.4}$$

where y_t is the observation at time t, C is the observation matrix, and v_t is measurement noise.

- **Noise Terms**: The process and observation models incorporate noise terms w_t and v_t, representing uncertainties in the system's dynamics and measurements, respectively. These are usually assumed to be Gaussian with known covariance.

These models are used in a number of tasks, but, most importantly, they have found an application in language modeling.

3.4.1 Mamba

Mamba [12] integrates SSMs with a selection mechanism that enables content-based reasoning and selective information processing along the sequence length. This selection mechanism tailors the SSM parameters according to the input, allowing the model to dynamically adapt its behavior based on the content of the sequence. This enhancement is crucial for handling tasks like selective copying and induction heads, which require the model to focus on relevant inputs and ignore irrelevant ones.

One of the achievements that enabled Mamba is the hardware-aware algorithm for selective SSMs which was proposed in the same paper as the model itself. It reduced the practical computational complexity of these models to almost-linear performance, however the drawback comes in the shape of high memory consumption, which leads to inefficient processing of very long sequences. Nevertheless, as demonstrated by Kuratov et al. [3], Mamba shows outstanding performance on tasks over 100000 tokens long.

4 Conclusion

The landscape of natural language processing has been revolutionized by Transformer-based models, particularly through their ability to efficiently capture long-term dependencies and generate rich context-aware representations of sequences. However, the self-attention mechanism employed by these models faces significant challenges when dealing with extremely long sequences, leading to decreased performance and inefficient resource management. To overcome these obstacles, several architectural improvements have been proposed, each aiming to balance the model's capability to handle long-range dependencies with the need for computational efficiency.

This literature review examined various approaches to optimizing Transformer models for long-context language modeling. We explored methods such as sparse attention, kernelized attention, segment-level recurrence and structured state-space models, evaluating their benefits and drawbacks. Each of these approaches presents its own benefits and drawbacks.

Firstly, sparse attention methods, although effective in reducing computational complexity, lacks the ability to efficiently capture long-term dependencies essential for complex reasoning tasks involving long sequences. Its reliance on local windows and limited global attention hinders its performance in tasks that necessitate connecting information across vast distances. Furthermore, despite improvements in computational efficiency, sparse attention methods still face significant trade-offs in memory usage and overall accuracy, making them less ideal for scenarios requiring robust long-context reasoning.

Kernelized attention offers a promising approach to reduce the computational complexity of self-attention, but it comes with significant drawbacks when applied to long-context reasoning. The use of approximations can lead to a loss of important information and reduced representation quality, negatively impacting the model's performance on complex tasks. Additionally, while kernelized attention may improve computational efficiency, it often does not translate into substantial memory savings, and its effectiveness can be limited in real-world applications requiring precise and detailed long-range dependencies.

While Mamba presents a strong approach for certain aspects of long-context language modeling with its ability to handle selective information processing and content-based reasoning, it falls short in several critical areas. Its high memory consumption and significant computational overhead make it less suitable for extremely long sequences and resource-constrained environments. Additionally, the dependence on specialized hardware

optimizations limits its flexibility across different platforms. These drawbacks collectively suggest that, despite its merits, Mamba is not the optimal choice for robust and universally applicable long-context language modeling solutions.

This leaves us with recurrent attention methods. Indeed, recurrent attention allows for very effective memory management and scales linearly regardless of the task. RMT and ARMT stand out noticeably in this class of models, with great memory retention across extremely long contexts, with ARMT even surpassing Mamba on 128000 tokens (not to mention the fact that it scores 76% across the first 5 tasks from BABILong on 50 million tokens). This performance, along with the capacity for transfer learning, strongly suggests that RMT and ARMT are the most relevant and promising architectures for the task of handling long sequences.

References

1. *Vaswani A.* Attention is all you need // Advances in Neural Information Processing Systems. — 2017.
2. *Kamradt G.* Needle in a Haystack. — 2023. — https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.
3. BABILong: Testing the Limits of LLMs with Long Context Reasoning-in-a-Haystack / Y. Kuratov, A. Bulatov, P. Anokhin, I. Rodkin, D. Sorokin, A. Sorokin, M. Burtsev. — 2024. — arXiv: 2406.10149 [cs.CL]. — URL: https://arxiv.org/abs/2406.10149.
4. *Beltagy I.*, *Peters M. E.*, *Cohan A.* Longformer: The Long-Document Transformer. — 2020. — arXiv: 2004.05150 [cs.CL]. — URL: https://arxiv.org/abs/2004.05150.
5. Big Bird: Transformers for Longer Sequences / M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, A. Ahmed // Advances in Neural Information Processing Systems. Vol. 33 / ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin. — Curran Associates, Inc., 2020. — P. 17283–17297. — URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.
6. Linformer: Self-attention with linear complexity / S. Wang, B. Z. Li, M. Khabsa, H. Fang, H. Ma // arXiv preprint arXiv:2006.04768. — 2020.
7. *Martins P. H.*, *Marinho Z.*, *Martins A. F. T.* ∞-former: Infinite Memory Transformer. — 2022. — arXiv: 2109.00301 [cs.CL]. — URL: https://arxiv.org/abs/2109.00301.
8. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context / Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, R. Salakhutdinov. — 2019. — arXiv: 1901.02860 [cs.LG]. — URL: https://arxiv.org/abs/1901.02860.
9. Memformer: A Memory-Augmented Transformer for Sequence Modeling / Q. Wu, Z. Lan, K. Qian, J. Gu, A. Geramifard, Z. Yu. — 2022. — arXiv: 2010.06891 [cs.CL]. — URL: https://arxiv.org/abs/2010.06891.
10. *Bulatov A.*, *Kuratov Y.*, *Burtsev M.* Recurrent memory transformer // Advances in Neural Information Processing Systems. — 2022. — Vol. 35. — P. 11079–11091.

11. Associative Recurrent Memory Transformer / I. Rodkin, Y. Kuratov, A. Bulatov, M. Burtsev. — 2024. — arXiv: 2407.04841 [cs.CL]. — URL: https://arxiv.org/abs/2407.04841.

12. *Gu A.*, *Dao T.* Mamba: Linear-time sequence modeling with selective state spaces // arXiv preprint arXiv:2312.00752. — 2023.

13. *Kitaev N.*, *Kaiser Ł.*, *Levskaya A.* Reformer: The efficient transformer // arXiv preprint arXiv:2001.04451. — 2020.

14. Rethinking attention with performers / K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, [et al.] // arXiv preprint arXiv:2009.14794. — 2020.