university of groningen / faculty of science and engineering / artificial intelligence

Exploring Retrieval in Hybrid SSM-Transformers

Bachelor's Project Thesis

Kurt Felix Michalak, s5142644, k.f.michalak@student.rug.nl,
Supervisors: Steven Abreu & Dr Herbert Jaeger

Abstract: Hybrid large language models (LLMs) combining state-space models (SSMs) with transformer self-attention layers offer promising computational efficiency while maintaining performance. However, the specific roles of different components remain unclear. This paper investigates retrieval capabilities in hybrid LLMs through systematic experiments on RecurrentGemma-2B, RecurrentGemma-9B, and Jamba-Mini-1.6. Using the Needle-in-a-Haystack benchmark and attention manipulation techniques, we demonstrate that retrieval depends exclusively on self-attention layers. Complete attention ablation causes total retrieval failure across all models, confirming that SSM layers do not contribute to retrieval. Methods to improve SSMs' retrieval abilities fail to recover retrieval capabilities in ablated models. Sparsification experiments reveal that attention layers can be significantly reduced without substantial performance degradation. Systematic attention weight manipulation shows that successful retrieval requires needle token exposure during generation and sufficient context during prefill or generation stages. These findings establish that self-attention layers serve as specialized retrieval modules while SSM and MLP layers handle general language capabilities.

1 Introduction

The transformer architecture (Vaswani et al., 2017), specifically the self-attention mechanism, is the basis for state-of-the-art (SOTA) models (DeepSeek-AI et al., 2025; Grattafiori et al., 2024; Gemma Team et al., 2024). However, FLOPs scaling quadratically with the length of the attention window makes transformer self-attention expensive during inference. So, recent advances in Large Language Models (LLMs) have increasingly focused on linear-attention models, especially state-space models (SSMs) (De et al., 2024; Gu & Dao, 2024; Dao & Gu, 2024; Qin et al., 2024). The FLOPs for processing a single token scale linearly with sequence length, which, computationally, makes them an attractive alternative to transformer-based models.

However, even modern SSMs are not yet capable enough to compete with transformer models, with their scores reaching a maximum of 90% of comparable transformers in performance benchmarks; depending on the task, these scores can drop down to 18% (De et al., 2024). Recent research on SSM-interpretability has highlighted the weaknesses of this architecture, primarily its lack of retrieval or copying ability (Arora, Eyuboglu, et al., 2024; Jelassi et al., 2024; Ben-Kish et al., 2025). This lacking ability is attributed to their *fuzzy memory* (Waleffe et al., 2024), where the in-context retrieval success rate decreases with the distance of to-be-retrieved token sequences to the end of the prompt. In contrast, for transformer-based models, this in-context retrieval is dependent on the attention implementation, and full accuracy can range from within a certain token window, e.g., the last 2048 tokens (sliding window attention), to the full sequence length (global attention).

During training, different heads of the self-attention mechanism (see Figure 3.1) learn specialized roles, while others learn more general roles (Voita et al., 2019). Recently, Yin & Steinhardt (2025) showed that specialized *function vector* (FV) heads (Todd et al., 2024; Hendel et al., 2023) contribute to the In-Context Learning (ICL) capability of transformer models, whereas special-

ized induction heads (Olsson et al., 2022; Elhage et al., 2021) learn pattern-matching functions enabling longer context retrieval. ICL describes the ability of a model to use in-context information (information contained in the prompt) to answer a question, understand a problem, or learn a behavior (Brown et al., 2020). The combination of these specialized and general heads is a strong driver of the performance of transformer-based architectures.

In an attempt to close the gap between transformers and SSMs, hybrid LLMs combine both layer architectures in a single model to reap the benefits of both: transformer-level retrieval and SSM-level efficiency. The strengths of the transformer architecture, namely, perfect retrieval, seem to complement the missing retrieval abilities of SSMs. Combining pure SSMs with self-attention in model architectures like RecurrentGemma (Botev et al., 2024; De et al., 2024) or Jamba (Lieber et al., 2024) appears to negate these weaknesses. They deliver similar capabilities to comparable transformer-based models. These performance gains were only achieved due to the (minimal) addition of self-attention mechanisms, based on the hypothesis that these mechanisms can complement the capabilities of pure SSMs. Now, the success of hybrid LLMs begs the question of the actual contributions of self-attention to the new capabilities and how they impact the latent space.

In a first attempt, Zani et al. (2025) investigated the impact of self-attention on hybrid LLM retrieval in an ablation study on RecurrentGemma-2B (referred to as *RG2B*). Their results suggest that hybrid LLM retrieval is solely dependent on self-attention, as a full ablation of this mechanism in RG2B leads to complete retrieval failure across all sequence lengths, such that not even the inferior *fuzzy memory* is retained. As a result of their findings, Zani et al. hypothesize that retrieval is learned as a specialized function in a small number of attention heads, similar to the occurrence of copying and induction heads in full transformers.

With this paper, we aimed to answer the following research questions regarding hybrid LLMs:

- Q1 Does the finding that hybrid LLMs rely on their attention layers for retrieval hold for hybrid LLMs other than the RG2B model tested by Zani et al. (2025)?
- Q2 Can methods for improving SSM's retrieval abilities (Arora, Timalsina, et al., 2024) be used to amplify SSM retrieval in hybrid LLMs?
- Q3 Can attention layers be sparsified to isolate their retrieval capability without degrading the overall performance of the language model?
- Q4 Under what conditions do attention layers in hybrid LLMs perform successful retrieval?

To answer Q1, we tested RecurrentGemma-9B (referred to as *RG9B*) and Jamba-Mini-1.6 (referred to as *Jamba*) on the Needle-In-A-Haystack (NIAH) task (Bai et al., 2024) with ablated and top-k-sparsified attention, similar to the experiments run by Zani et al. (2025). NIAH is a retrieval benchmark that tests the ability of a model to retrieve an exact string, the needle, from a previously presented context, the haystack. See Section 2.2 for more details.

To answer Q2, one has to answer two questions: "*Does only attention perform retrieval?*" and "*Does attention only perform retrieval?*". While the earlier question seems to be covered through Q1, Arora, Timalsina, et al. (2024) found that the prompting structure is a limiting factor for retrieval in SSMs, so they derived the *Just Read Twice* (JRT) method to circumvent this problem. We repeated all experiments from Q1 and applied the JRT method.

To answer Q3, we ran Jamba on two benchmarks: the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) and the Massive Multitask Language Understanding benchmark (MMLU) (Hendrycks et al., 2020). Both benchmarks measure general natural language processing capabilities, although MMLU was designed as a more difficult successor to GLUE. Jamba was run three times on this benchmark: once in the base configuration, once in the optimally sparsified configuration, and once in the ablated configuration. The optimally sparsified configuration used the highest possible self-attention sparsification that still showed close to full retrieval capabilities in NIAH; the ablated configuration fully sparsified the model, effectively removing the self-attention mechanism.

To answer Q4, we systematically manipulated the self-attention mechanism of RG2B to highlight the importance of different parts in the context for successful retrieval. Inference in most language

models is split into two stages, the *prefill* stage and the *generation* stage. During the prefill stage, all prompt tokens are processed in parallel to fill model activations, and the first token is generated. The generation stage follows after the prefill stage, and uses autoregressive sampling to predict the second and every following token after another. We ran a combination of different ablation methods during prefill and generation.

Since the research questions are partially dependent on each other, they will be addressed in order, one at a time. Sections 3-6 deal with Q1-4.

2 Methodology

2.1 Models

Across all four questions, we used three different models: RecurrentGemma-2B (RG2B), RecurrentGemma-9B (RG9B), and Jamba-Mini-1.6 (Jamba).

RG2B was used by Zani et al. (2025) as a comparatively small hybrid LLM that is easy to run. We used this model to provide comparability to previous results. This model consists of 2.03B parameters (excluding embedding parameters) spread across 26 layers, being eight repetitions of the pattern "2× SSM, 1× Attention" followed by two SSM layers. Each attention layer hosts ten attention heads.

RG9B was used as a model of a larger size that is comparable to RG2B. RG9B and RG2B are of the same structure and architecture, just that RG9B is bigger in scale with 7.53B parameters spread across 38 layers, being 12 repetitions of the pattern "2× SSM, 1× Attention" followed by two SSM layers. The attention layers of RG9B host 16 heads each.

Both RG models use sliding window attention with a sliding window size of 2048.

Jamba was used as a model of different architecture to compare to the RG models. Jamba uses global attention and Mixture of Experts (MoE) layers that systematically substitute Multilayer Perceptrons (MLPs). MoE layers are sparse alternatives to MLPs that only use a fraction of parameters during inference, effectively reducing the memory footprint of a model. Jamba consists of 51.03B total parameters (excluding embedding parameters), of which only 12B are active during inference, spread across 32 layers in a pattern of "3× SSM, 1× Attention 4× SSM". Each attention layer hosts 32 attention heads.

These three models built a diverse basis for experimental results, differing in their parameter sizes, attention implementation, and surrounding architecture.

All experiments were conducted on a high-performance cluster, utilizing a maximum of two AMD Zen 3 EPYC 7763 processors, eight NVIDIA A100 (40 GB HBM2) graphics cards, and 1024 GB of memory.

2.2 The NIAH Benchmark

The Needle-In-A-Haystack (NIAH) benchmark is built on the implementation by Bai et al. (2024). The task in NIAH is to retrieve a specific sentence from a context that is filled with irrelevant text. The to-be-retrieved information is called the *needle*, and the whole context, including the irrelevant text and the needle, is called the *haystack*. The benchmark evaluates the ability of a model to retrieve the needle from the haystack for multiple prompts of different lengths. Through strategic prompt generation, this benchmark can yield retrieval maps that unveil the retrieval performance of a model, depending on the position of the needle and the size of the haystack. See Figure 6.2 for examples of such a retrieval map. Following Bai et al. (2024), we used the same collection of essays from Paul Graham* to generate the haystack, as well as the same needle.

The needle was "The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day". The retrieval question was "What is the best thing to do in San Francisco?". See Appendix A for our prompt templates.

As an improvement over the scoring method used by Zani et al. (2025), which was a simple binary string matching evaluation, we implemented granular scoring. While still based on string matching, partial matches were allowed to yield partial scores. For each keyword of the needle that was present in the output string, the score was increased. The

*All essays can be found in their original form at https://www.paulgraham.com/articles.html, and are also part of the github repository for this project at https://github.com/lamalunderscore/retrieval-in-hybrid-ssms.

scores for all partial keywords added up to three. If the full needle string matched, the score was set to five. While examining outputs from test experiments on Jamba, we stumbled across a curious phenomenon: the grammatically imperfect needle was predicted with corrected grammar, thereby only scoring three points. To visualize this phenomenon, we added a rule: if the needle was predicted with corrected grammar, the score was set to four. See Table 2.1 for concrete strings and their scoring behavior.

Keyword	Score
"eat a sandwich"	increase by 1.0
"Dolores Park"	increase by 0.5
"sit in Dolores Park"	increase by 0.5
"sunny day"	increase by 1.0
"is to eat a sandwich and sit in Dolores Park on a sunny day"	set to 4.0
"The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day"	set to 5.0

Table 2.1: Keyword strings used in the evaluation of retrieval success, and their influence on the score. Notice the grammatical imperfection in the full needle string (last row).

3 Hybrid SSM-Transformers rely on self-attention for retrieval

3.1 Methodology

Zani et al. (2025) used 100 prompts to test retrieval in RG2B: ten prompt lengths on ten different depths at which the to-be-retrieved information was located in the prompt. The maximum prompt length was chosen as double the sliding window size of RG2B - namely, 4096 tokens.

RG9B has the same sliding window size as RG2B, so to ensure comparability to previous results, we used the same maximum prompt length of 4096 for RG9B. As Jamba features global attention, one expects the retrieval ability of the unmanipulated model to be similar across all token lengths. The only limiting factor for its retrieval ability is the prompt length it was trained on, affecting the effectively usable context length during inference. For Jamba-Mini-1.6, this context length is reported to be 256k tokens. Fully investigating the impact of attention head-sparsification on Jamba retrieval - testing prompt lengths up to 512k tokens - was not feasible, as it would have required hardware resources that were inaccessible for this project. For continuity reasons, we chose to test Jamba on the same maximum prompt length as RG9B, 4096 tokens. We used the same prompt distribution as Zani et al. (2025), meaning ten prompt lengths across ten different depths, amounting to 100 prompts in total.

For sparsification, the k attention heads with the lowest uniformity in their attention weight distribution were retained, and all other attention heads were ablated - this is called top-k sparsification and assumes $k \in \mathbb{N}$. Entropy was used as a uniformity measure and calculated as

$$H(A) = -\sum_t a_t \times log_2(a_t),$$

where A is the attention weight vector of an attention head, and a_t is the attention weight of token t in A. See Figure 3.1 for more details, as it indicates the parts of interest in the self-attention mechanism. The ablation acted on the attention output, based on the metrics calculated on the attention weights. Ablating a specific head here meant setting all values reflecting that head in the attention output to zero.

RG9B and Jamaba were run for $k \in [0, 16]$ and $k \in [0, 32]$ respectively. Note that $k = \text{num\_heads}$ (the highest tested k for both models) is equivalent to the non-sparsified model.

Zani et al. (2025) only applied sparsification during the generation stage. However, we ran all sparsification levels twice, once only during generation, and once during prefill as well as generation, because it provided a more detailed overview of the impact of attention sparsification on model output.

Since RecurrentGemma and Jamba have to be run via different means (Kaggle/custom loading or HuggingFace, respectively), we built an extendable library to create a unified interface for the manipulations mentioned in this paper, and creatively

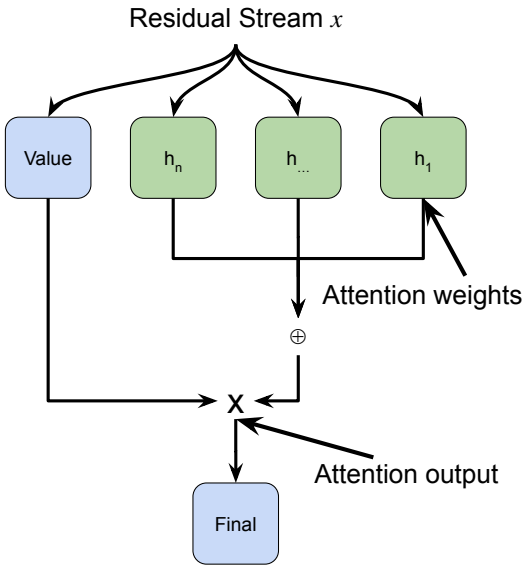


Figure 3.1: Structural overview of the self-attention mechanism. Projections are indicated as blue squares. Green squares represent the operations of a single head, outputting $A = QK^T$, where $Q = xW_Q$ and $K = xW_K$ are results of head-specific projections of the residual stream x. Consider Vaswani et al. (2017) for a detailed explanation of the attention mechanism.

called it ManipuLatte[†].

To increase comparability, the sparsification setup for RG2B used by Zani et al. (2025) was rerun on the updated procedure, and we also added a run that sparsified RG2B during both stages for continuity.

All models were tested without batching (using a batch size of one).

3.2 Results

As seen in Figure 3.2, all three models showed distinct but similar behaviors for sparsification.

We will refer to the model versions that are only sparsified during the generation stage as *generation versions*, and to the versions that are sparsified during both stages as *prefill versions*.

Jamba scored the highest across all models, reaching 100% accuracy until $k = 27$. The prefill and generation versions behave very similarly, both first decline slowly and steadily until $k = 6$, and then sharply drop in their scores. The prefill version shows an advantage of around five points until $k = 6$, but also drops more sharply afterwards.

RG9B scored the second best across the three models. Both versions showed a dip that starts at $k = 16$ and ends around $k = 3$, only that the prefill version bottoms out deeper. Accuracy sharply dropped after $k = 2$ for the generation version, and after $k = 3$ for the prefill version.

RG2B scored the worst overall, although the generation version was comparable to the generation version of RG9B at low k. For this model, the two different versions also showed drastically different behaviors. While the generation version showed a steady increase in accuracy until $k = 1$, the prefill version almost immediately dropped to below 20% accuracy, where it then stayed until $k = 1$. The accuracy of both versions sharply drops again after $k = 1$.

All models and versions shared a common behavior: there was a tipping k after which accuracies dropped quickly to zero percent. This k ranged from 5 to 1, and was higher for models with a higher number of attention heads.

3.3 Discussion

Retrieval failed for all models at $k = 0$, which confirmed that the findings made by Zani et al. (2025) also hold on other hybrid LLMs. However, all three models showed slightly different tipping points in generation versions, after which the accuracy started to decrease drastically: $k = 4$ for Jamba, $k = 2$ for RG9B, and $k = 1$ for RG2B. This difference was expected to a degree, as all three models are different, either in size or architecture, and different training runs can yield differently performing models even if the underlying structure is identical. Still, the overall behavior is similar across all three models. Note that our results showed the tipping point for RG2B at $k = 1$, whereas Zani et al. (2025) identified this tipping point to be around $k = 2$. This difference can be attributed to our improved scoring method and prompt structure.

Regarding the prefill versions, the difference in behaviors was striking, especially looking at the behavior of RG2B. It seems that the prefill sparsification made the effects more prominent in Jamba and RG9B, whereas it made RG2B fail even at low

[†]Available at https://github.com/lamalunderscore/manipulatte

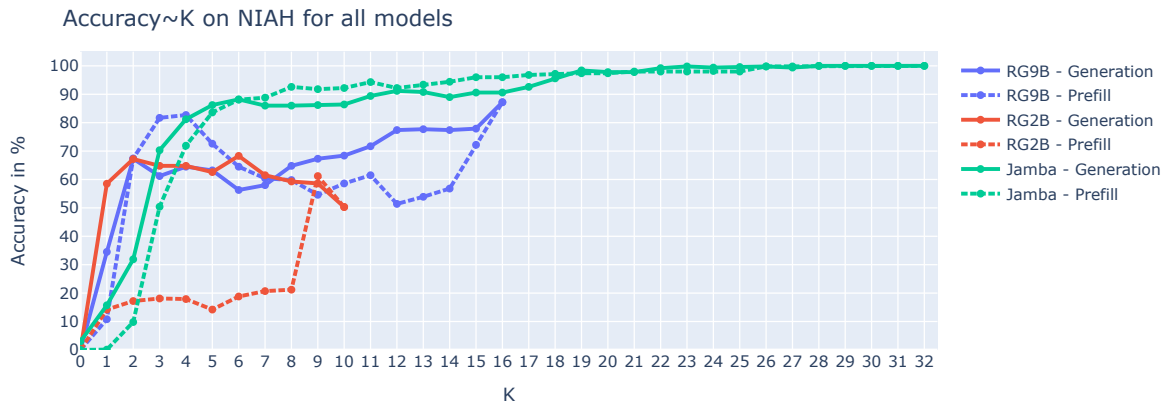


Figure 3.2: Accuracy as a function of k in top-k sparsification on standard NIAH for *generation* and *prefill versions*. Accuracy was approximated by the average score across all 100 prompts, relative to the maximum score (5). Note that scoring was neither linear nor continuous (see Table 2.1 for interpretation guidance). Read from right to left for increasing sparsity.

sparsification levels.

To understand why RG2B was impacted by prefill sparsification, it is important to consider the impact of sparsification on the internals of the model. Any manipulation of the computations during a forward pass will alter the residual stream and therefore yield, compared to the original activations, an imperfect version of all downstream activations. This means that every layer activation after the first manipulation is imperfect, and that every further manipulation likely pushes subsequent layer activations further away in the latent space. This also means that every query-, key- and value-projection is imperfect. Since the RG models used a cache for the values of key- and value-projections (kv-cache) to optimize inference time, the query-projection will use these imperfect activations for every subsequent forward pass during that inference run, increasing the impact of the manipulation. Especially important for this cache is the prefill stage, as it fills the kv-cache with all projections of the prompt tokens. Hence, we would expect the RG models to be more impacted by prefill sparsification.

Furthermore, approximately the last third of layers contributes the most decisive information in transformer-based models (Belrose et al., 2023), the most important self-attention layers act on the most manipulated activations. Additionally, the number of heads per self-attention layer should be considered, as the relative impact of incrementing k by one differs, being higher for smaller models (with fewer heads per layer). Jamba was the biggest model, but featured the fewest self-attention layers. RG9B has three times as many self-attention layers but has about half the active parameters; RG2B has twice as many self-attention layers as Jamb but only a sixth of the active parameters.

All things considered, it is likely that the detrimental impact of prefill sparsification on RG2B's retrieval accuracy originated in the fact that it used a kv-cache, which was further amplified by the comparatively small size of the model.

The impact of activation manipulations on different model structures could be investigated in the future by comparing runs with and without using a kv-cache, as well as training different configurations of the same model architecture that capture a wide diversity in combinations of the number of layers and heads per layer.

4 Attempting to recover SSM-retrieval capabilities

4.1 Methodology

In line with Arora, Timalsina, et al. (2024), we applied the JRT method by simply repeating the `context` and `question` parts once (see Appendix

A).

The goal of this experiment was to amplify potentially hidden retrieval capabilities in the SSM layers. Results from runs with higher k would not yield any insights, as successful retrieval was already accomplished through self-attention. Investigating low k sparsification minimized the effect of self-attention retrieval and so isolated the potential retrieval performed through SSM layers. Hence, for both models, we only ran NIAH with JRT applied for $k \in \{0, 1, 2\}$. Note that $k = 2$ already showed close to full retrieval performance without applying JRT. Since we were comparing the JRT retrieval results to the findings by Zani et al. (2025), we used the generation versions for all models - RG2B, RG9B, and Jamba.

4.2 Results

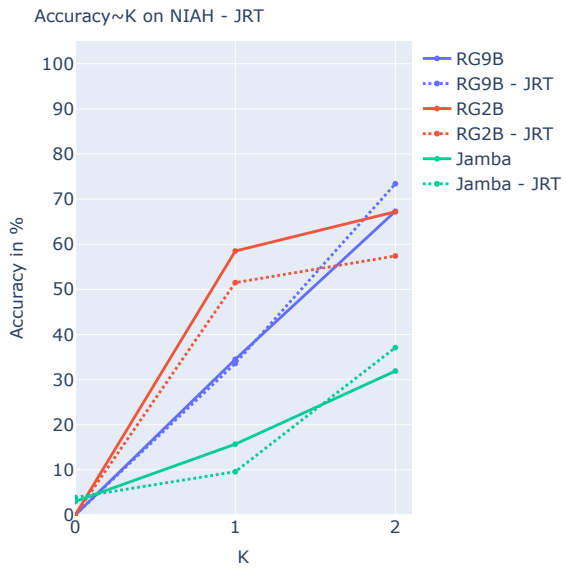


Figure 4.1: Accuracy as a function of k in top-k sparsification on NIAH with and without JRT applied, for all models in the generation version. Accuracy was approximated by the average score across all 100 prompts, relative to the maximum score (5). Note that scoring was neither linear nor continuous (see Table 2.1 for interpretation guidance). Read from right to left for increasing sparsity.

As seen in Figure 4.1, the accuracies on NIAH with JRT applied were very similar to the accuracies without JRT applied. The JRT versions only scored slightly higher at $k = 2$ for RG9B and Jamba, but for RG2B at $k = 2$, the JRT version scored notably lower. At $k = 0$, the RG versions scored identically with zero percent retrieval accuracy, and the Jamba versions scored similarly, around three percent.

See Appendix B for a comparison between retrieval maps for standard and JRT prompting.

4.3 Discussion

Applying JRT during NIAH does not recover the retrieval ability of any model. With this experiment, we wanted to test if indeed only the self-attention layers perform retrieval, as hypothesised by Zani et al. (2025). These results, together with the results from Q1 (see Section 3.2), provide further strong evidence that retrieval in hybrid LLMs is exclusively implemented through self-attention layers. This also further solidifies the hypothesis that during training, only self-attention layers learn the retrieval function, so much so that SSM layers do not even develop the common *fuzzy memory* (Waleffe et al., 2024).

To investigate these training dynamics, one could identically train a pure SSM and a hybrid version of the same SSM that only adds self-attention layers. After it was confirmed that the hybrid model indeed failed retrieval tasks with ablated self-attention, the state transition matrices of the SSM layers of both models should be compared. Since the pure SSM layers will have learned a retrieval function (at least for fuzzy memory), but the layers of the hybrid SSM will not, contrasting the state transition matrices could uncover the retrieval mechanism in the pure SSM layers. Uncovering the retrieval mechanism used in SSMs would provide an example approach for mechanistic SSM analysis, furthering the field of Mechanistic Interpretability.

5 Benchmarking sparsified and ablated self-attention

5.1 Methodology

Recent advances in benchmarking LLMs included the LM-Eval-Harness (LME) in Gao et al. (2024).

LME is a library that implements easy benchmarking of, among others, HuggingFace models and allows for the extension of custom wrappers. However, writing a custom wrapper for Recurrent-Gemma was not feasible due to time constraints. For this reason, we only benchmarked Jamba, as it is provided by HuggingFace.

The MMLU benchmark provided by LME is evaluated using the log likelihood of multiple-choice options right after the prefill stage. We chose to only run the benchmarks on prefill-sparsified Jamba, since only this version would impact the evaluated likelihoods of multiple-choice tokens.

First, Jamba was loaded and then initialized for sparsification using $k = 5$ via ManipuLatte. We chose $k = 5$, as experiments for Q1 showed that this is the lowest k for Jamba in the prefill version that retains close to full retrieval capabilities (see Figure 3.2). The loaded model was then exposed to the LME library to run the benchmarks (see Appendix C).

MMLU is tested in a five-shot configuration (Hendrycks et al., 2020), and GLUE in a zero-shot configuration (Wang et al., 2019). We followed the same procedure.

n-shot evaluations prepend n examples of questions and their answers to every prompt.

5.2 Results

Figure 5.1 shows the average achieved scores per benchmark, per model configuration. See Appendix C for subtask results.

On GLUE, the base configuration of Jamba performed best, but the ablated and sparsified versions only scored slightly lower (6.6 and 9.1 points, respectively). Although only marginally, the ablated version scored higher than the sparse version (2.8 points).

On MMLU, the base configuration performed best again. The sparse version only scored six points below the base configuration, whereas the ablated version scored around 42 points below the base configuration.

For the more challenging MMLU benchmark, the sparse performance only declined 7.4% (relatively), despite the same configuration showing a more substantial 16.4% decrease in NIAH (see Figure 3.2). On the one hand, since the average decrease on MMLU is much lower than the decrease in retrieval capabilities, this suggests that the self-attention layers do not host other critical and exclusive capabilities beyond retrieval. On the other hand, the ablation ($k = 0$) completely failed on MMLU, dropping close to the random guessing baseline, which could be an indicator that self-attention layers play a role outside of retrieval, and that their contribution is crucial to the model's inner workings. However, the GLUE results contradicted this theory by showing that the ablated configuration and the sparse configuration score similarly (and close to the base configuration) on GLUE, with the ablated configuration even coming out on top on average.

Figure 5.1: Scores for Jamba-Mini-1.6, in an ablated configuration ($k = 0$), a sparse configuration ($k = 5$), and the base configuration, on GLUE and MMLU.

5.3 Discussion

The results provide evidence that the retrieval capability of self-attention layers can be isolated through sparsification. However, the effect of sparsification depends significantly on the complexity and nature of the evaluated task.

The big difference in the effect of sparsification and ablation between MMLU and GLUE highlights the necessity for retrieval capabilities for more difficult language modeling tasks, whereas general lan-

guage capabilities do not depend on retrieval to such a degree.

Still, the finding that the sparsified configuration performed worse than the ablated configuration hints at a suboptimal sparsification method. There are two likely explanations for this observation. First, the sparsification metric may be inadequately selecting which attention heads to preserve and ablate, preserving heads that potentially introduce noise while ablating the ones that are crucial for retrieval. Second, building on the first explanation, an incomplete retrieval mechanism may be worse than no retrieval at all, as a broken retrieval function that was learned to be crucial for certain tasks could lead to inconsistent or misleading information flow. The effect of such a broken retrieval function could range from simply introducing random noise (as mentioned in the first explanation) to systematically moving the activation vector in a wrong direction in the latent space, potentially pushing values out of distribution.

The findings from GLUE highlight the necessity for a more sophisticated sparsification method. Instead of investigating attention heads behaviorally, Olsson et al. (2022) structurally investigated attention heads through an offline analysis that examines the weights of different circuits in the self-attention architecture. Such an analysis could provide insights into which heads are implementing which kind of capabilities and functions before running inference. This way, instead of dynamically choosing the attention heads to ablate at run time, a model could be sparsified statically, based on the characteristics of different heads, and with more care.

6 Manipulating Attention Weights

6.1 Methodology

As previous experiments showed that the influence of interventions on self-attention yields similar results across models (see Section 3.2), this experiment was only run on RG2B because of time and compute constraints.

To isolate the effect of ablating attention weights, we used the base configuration, without sparsification ($k = 10$ for RG2B). Sparsity would only lead to confounding variables since the focus here is on the full self-attention mechanism and how it implements retrieval.

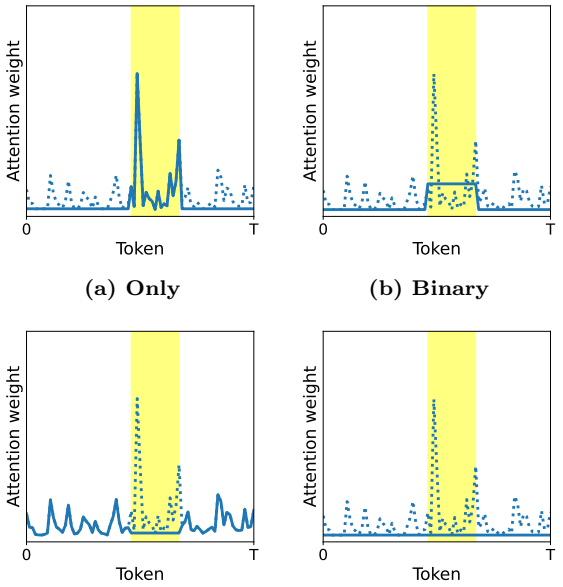
There were four different ablation methods used for this experiment: *Null*, *Only*, *Binary*, and *Omit*.

Null: Does as the name suggests and nullifies every attention weight, having the same effect as a $k = 0$ sparsification (see Figure 6.1d).

Only: With this method, the attention weights for all tokens, except for the needle tokens, get nullified. The needle tokens retain their original values (see Figure 6.1a).

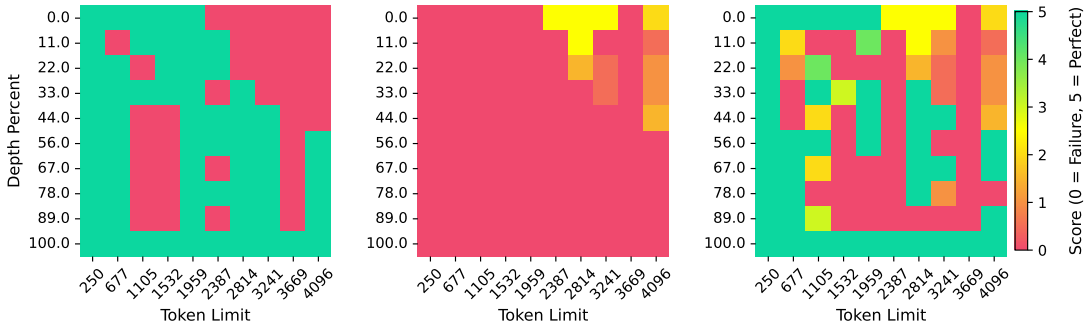
Binary: Similarly to *Only*, all attention weights except for the needle tokens get nullified. With this method, however, every attention weight of the needle tokens gets assigned the same value, namely the average weight across needle tokens (see Figure 6.1b).

Omit: This method is the inverse of *Only*, only nullifying the attention weights of the needle tokens, and all other attention weights retaining their original value (see Figure 6.1c).



(a) Only **(b) Binary** **(c) Omit** **(d) Null**

Figure 6.1: Simplified representation of different manipulation methods. The original attention weights are the *dotted blue* line, and the modified attention weights are the *solid blue* line. The token needles are indicated by the *yellow highlight*.



(a) Keep-Omit **(b) Omit-Keep** **(c) Keep-Keep (Base)**

Figure 6.2: Retrieval maps for RG2B on NIAH with the manipulations Keep-Omit, Omit-Keep, and Keep-Keep applied. The x-axis shows the used prompt length, the y-axis the depth of the needle, 0% being the very end of the prompt and 100% being the very beginning of the prompt.

Additionally, consider *Keep* as an indicator for no manipulation.

Every manipulation method can be applied during the prefill stage as well as the generation stage, and there is no need for both stages to be subject to the same method. We apply every prefill-generation method combination in a manner of increasing severity and run NIAH on those configurations.

Note that we exclude *Null* as a method for the generation stage, as this scenario is equivalent to full sparsification using $k = 0$. Full sparsification leads to a complete retrieval failure (see Figure 3.2), so further manipulating self-attention would only lead to the same result.

Since the position of the needle tokens in the tokenized context was different for every prompt in NIAH, the necessary positions were recalculated before every prompt when applying *Only*, *Omit*, or *Binary*.

Combinations were named "*Generation mode - Prefill mode*", so "Omit-Null" means applying *Omit* during generation, and *Null* during prefill.

6.2 Results

Table 6.1 gives an overview of the accuracies scored in NIAH for different combinations of prefill and generation manipulation. Omitting the needle tokens during generation led to a drastic drop in accuracy, with Omit-Keep reaching 3.7% accuracy, and all other combinations scoring zero or close to zero percent accuracy. Applying the Binary method during generation yielded the same accuracies as applying the Omit method; all combinations applying binary during prefill scored zero percent accuracy. The combinations Keep-Omit, Keep-Only, and Keep-Null all improved the accuracy relative to the base configuration, with 63.0%, 70.6%, and 68.8%, respectively. When only keeping the needle tokens during prefill, not manipulating the prefill or omitting the needle tokens also improved the scores relative to the base configuration, with 85.7% and 53.0% respectively. Only-Keep reached the highest score across all combinations. Note that every sparsification combination that did not severely reduce retrieval capabilities yielded better scores than the base model (Keep-Keep).

For a complete overview of all retrieval maps, see Appendix D.

Prefill \ Generation	Keep	Omit	Only	Binary
Keep	50.3%	3.7%	**85.7%**	3.7%
Omit	63.0%	0.0%	53.0%	0.0%
Only	**70.6%**	0.1%	18.2%	0.1%
Binary	0.0%	0.0%	0.0%	0.0%
Null	68.8%	0.0%	17.0%	0.0%

Table 6.1: NIAH results for all tested manipulation combinations on RecurrentGemma-2B. Marked as **overall best**, **second best** and third best.

6.3 Discussion

The results suggest that successful retrieval necessitates, firstly, exposure of the needle tokens during generation, and secondly, enough exposure of tokens that provide sufficient context either during prefill or during generation. The first point is obvious from the results when applying Omit during generation. Every combination, including Omit-Keep, fails retrieval. To deduce the second point, the key combinations to look at are Only-Omit and Only-Only, as well as Keep-Omit, Keep-Only, and Keep-Null. Only-Omit shows that, even when only exposing the needle tokens during generation, only exposing the structure around the needle during prefill suffices to facilitate successful (above baseline) retrieval capabilities; when not exposing the structure during prefill, like in Only-Only, retrieval fails. The other three combinations, especially Keep-Null, show that the tokens exposed during prefill are irrelevant, likely because enough tokens are exposed during generation.

Additionally, the low accuracy scores when applying Binary showed that the values of the attention weights are important for successful retrieval. It is not enough to expose necessary tokens more than unnecessary ones, but the exact structure and relations matter.

Investigating the retrieval maps of different combinations revealed further details, refining the previous hypothesis. The retrieval maps for Keep-Omit (Figure 6.2a) and Omit-Keep (Figure 6.2b) complement each other to make up the retrieval map of Keep-Keep (Figure 6.2c). This means that Omit-Keep, which was previously labeled as a retrieval failure, was in fact not a failure but only a partial success.

The phenomenon we can see here is likely related to the 2048 token sliding window attention that the RG models employ. Observe that the triangle of partially successful retrievals in the top right corner of Figure 6.2b starts at a prompt length of 2387, which is the first tested prompt length after 2048 tokens. The combinations of prompt length and needle depth that showed successful retrieval all have one thing in common: the needle was outside of the 2048 token sliding window of attention exposure.

This is only possible because the RG models actually only employ sliding window attention during the generation stage, but not during the prefill stage. The fact that there is even partially successful retrieval means that either the first output token (which is generated by prefill) carries a large enough bias, or the kv-cache generated during prefill encoded enough implicit information to make the model partially retrieve correct information despite this information not being exposed to the self-attention mechanism during further inference steps. It also means that the retrieval function learned by the model can generalize beyond 2048 tokens. These retrieval maps show that retrieval capabilities beyond the sliding window in RG2B are strictly dependent on all necessary tokens being exposed during the prefill stage, as Omit-Omit and Omit-Only do not show this behavior. They also show that the retrieval capabilities inside the sliding window are dependent on the tokens exposed during the generation stage.

Since the Binary method yielded such detrimental results, we examined the outputs of corresponding combinations. The outputs showed that when Binary was applied during prefill, the generated tokens were mostly nonsensical. When Binary was applied during generation, the generated tokens made sense, but were off topic. See Appendix E for a selection of pearls of wisdom generated by confused hybrid models.

Just as the effect of sparsifying during the prefill stage as discussed in Section 3.3, applying Binary during prefill likely has a similar effect on the kv-cache, which produces nonsensical tokens during generation. This would also explain why applying Binary only during generation did not result in such catastrophic failures, as the kv-cache was prefilled with unmanipulated keys and values.

Overall, the Binary method was not thoroughly investigated before employing it, but rather it was implemented in a way that most efficiently applied the idea of unifying attention weights across a range of tokens. For example, upon examining attention weights for the first activations in the generation stage, it became obvious that the range of the needle tokens always started with a large peak in the weights on the exact token that would have to be predicted (also visible in Figure 6.1). The behavior of the attention weights throughout the generation process was not further investigated. It is likely that this peak moves across the range of needle tokens, always being located on the token corresponding to

the correct prediction. A behavior like this would mean that the method used to apply the idea of unifying and binarizing attention weights was inadequate. This limitation could be overcome in future research.

7 Conclusion

This paper investigates the role of self-attention mechanisms in hybrid language models through systematic experiments on RecurrentGemma-2B, RecurrentGemma-9B, and Jamba-Mini-1.6, addressing four key research questions about retrieval capabilities and sparsification potential.

Our experiments demonstrate that retrieval in hybrid LLMs depends exclusively on self-attention layers. Ablation of self-attention resulted in complete retrieval failure across all tested models, confirming that SSM layers do not contribute to retrieval, as even the *fuzzy memory* observed in pure SSMs (Waleffe et al., 2024) is absent. The Just Read Twice method (Arora, Timalsina, et al., 2024) failed to recover any retrieval capabilities in sparsified models, further confirming this exclusive dependence.

The retrieval capabilities of self-attention could be isolated without detrimental effects on the model, as shown by our benchmarks on different model versions. Still, the effect of self-attention sparsification was dependent on the tested task.

We showed that successful retrieval requires two conditions: to-be-retrieved tokens must be exposed during generation, and sufficient context must be available during prefill or generation. Furthermore, the exact attention weight values matter, which means that binarizing attention weights post-hoc leads to retrieval failure.

These findings suggest that hybrid architectures can be optimized through targeted attention sparsification without significant performance degradation on non-retrieval tasks. The exclusive specialization of attention for retrieval indicates that these components could be designed specifically for this function, potentially reducing computational overhead and increasing explainability.

Our entropy-based sparsification method showed suboptimal performance in some cases, indicating the need for more sophisticated approaches such as structural circuit analysis of the self-attention mechanism, as was performed by Olsson et al. (2022). Future work may investigate training dynamics that lead to this component specialization and explore whether self-attention layers can be substituted by a light-weight retrieval-only mechanism in hybrid architectures.

This work establishes that self-attention serves as a specialized retrieval module in hybrid models, while SSM layers handle general language capabilities. Understanding this division of functionalities is crucial for designing efficient hybrid architectures that achieve transformer-level performance with improved computational efficiency, for developing more explainable substitutes to self-attention, and to understand the magic behind transformers. The exclusive dependence on attention for retrieval provides clear guidance for optimizing these architectures through targeted sparsification strategies and points out pathways for future research.

References

Arora, S., Eyuboglu, S., Zhang, M., Timalsina, A., Alberti, S., Zinsley, D., ... Ré, C. (2024, February). *Simple linear attention language models balance the recall-throughput tradeoff.* arXiv. Retrieved 2024-12-24, from `http://arxiv.org/abs/2402.18668` (arXiv:2402.18668 [cs]) doi: 10.48550/arXiv.2402.18668

Arora, S., Timalsina, A., Singhal, A., Spector, B., Eyuboglu, S., Zhao, X., ... Ré, C. (2024). *Just Read Twice: closing the recall gap for recurrent language models.* arXiv. Retrieved 2024-12-25, from `http://arxiv.org/abs/2407.05483` (arXiv:2407.05483 [cs]) doi: 10.48550/arXiv.2407.05483

Bai, Y., Lv, X., Zhang, J., He, Y., Qi, J., Hou, L., ... Li, J. (2024). *LongAlign: A Recipe for Long Context Alignment of Large Language Models.* Retrieved from `https://arxiv.org/abs/2401.18058`

Belrose, N., Furman, Z., Smith, L., Halawi, D., Ostrovsky, I., McKinney, L., ... Steinhardt, J. (2023). *Eliciting Latent Predictions from Transformers with the Tuned Lens.* Retrieved from `https://arxiv.org/abs/2303.08112`

Ben-Kish, A., Zimerman, I., Abu-Hussein, S., Cohen, N., Globerson, A., Wolf, L., & Giryes, R. (2025, April). *DeciMamba: Exploring the Length Extrapolation Potential of Mamba.* arXiv. Retrieved 2025-06-24, from http://arxiv.org/abs/2406.14528 (arXiv:2406.14528 [cs]) doi: 10.48550/arXiv.2406.14528

Botev, A., De, S., Smith, S. L., Fernando, A., Muraru, G.-C., Haroun, R., ... Frietas, N. d. (2024, August). *RecurrentGemma: Moving Past Transformers for Efficient Open Language Models.* arXiv. Retrieved 2025-01-05, from http://arxiv.org/abs/2404.07839 (arXiv:2404.07839 [cs]) doi: 10.48550/arXiv.2404.07839

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. *CoRR*, *abs/2005.14165*. Retrieved from https://arxiv.org/abs/2005.14165

Dao, T., & Gu, A. (2024, May). *Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality.* arXiv. Retrieved 2025-01-05, from http://arxiv.org/abs/2405.21060 (arXiv:2405.21060 [cs]) doi: 10.48550/arXiv.2405.21060

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., ... Gulcehre, C. (2024, February). *Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models.* arXiv. Retrieved 2025-01-05, from http://arxiv.org/abs/2402.19427 (arXiv:2402.19427 [cs]) doi: 10.48550/arXiv.2402.19427

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., ... Zhang, Z. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.* Retrieved from https://arxiv.org/abs/2501.12948

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., ... Olah, C. (2021). A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*. (https://transformer-circuits.pub/2021/framework/index.html)

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., ... Zou, A. (2024, 07). *The Language Model Evaluation Harness.* Zenodo. Retrieved from https://zenodo.org/records/12608602 doi: 10.5281/zenodo.12608602

Gemma Team, Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., ... Andreev, A. (2024, October). *Gemma 2: Improving Open Language Models at a Practical Size.* arXiv. Retrieved 2025-07-01, from http://arxiv.org/abs/2408.00118 (arXiv:2408.00118 [cs]) doi: 10.48550/arXiv.2408.00118

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... Ma, Z. (2024). *The Llama 3 Herd of Models.* Retrieved from https://arxiv.org/abs/2407.21783

Gu, A., & Dao, T. (2024, May). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces.* arXiv. Retrieved 2025-01-05, from http://arxiv.org/abs/2312.00752 (arXiv:2312.00752 [cs]) doi: 10.48550/arXiv.2312.00752

Hendel, R., Geva, M., & Globerson, A. (2023). *In-Context Learning Creates Task Vectors.* Retrieved from https://arxiv.org/abs/2310.15916

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020). Measuring Massive Multitask Language Understanding. *CoRR*, *abs/2009.03300*. Retrieved from https://arxiv.org/abs/2009.03300

Jelassi, S., Brandfonbrener, D., Kakade, S. M., & Malach, E. (2024, June). *Repeat After Me: Transformers are Better than State Space Models at Copying.* arXiv. Retrieved 2024-12-24, from http://arxiv.org/abs/2402.01032 (arXiv:2402.01032 [cs]) doi: 10.48550/arXiv.2402.01032

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., ... Shoham, Y. (2024, July). *Jamba: A Hybrid Transformer-Mamba Language Model.* arXiv. Retrieved 2025-01-05, from http://arxiv.org/abs/2403.19887 (arXiv:2403.19887 [cs]) doi: 10.48550/arXiv.2403.19887

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., ... Olah, C. (2022).

In-context learning and induction heads. *Transformer Circuits Thread*. (https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html)

Qin, Z., Yang, S., Sun, W., Shen, X., Li, D., Sun, W., & Zhong, Y. (2024, April). *HGRN2: Gated Linear RNNs with State Expansion.* arXiv. Retrieved 2024-07-10, from http://arxiv.org/abs/2404.07904 (arXiv:2404.07904 [cs]) doi: 10.48550/arXiv.2404.07904

Todd, E., Li, M., Sharma, A. S., Mueller, A., Wallace, B. C., & Bau, D. (2024). Function Vectors in Large Language Models. In *The Twelfth International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=AwyxtyMwaG

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. Retrieved 2025-07-01, from https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). *Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned.* Retrieved from https://arxiv.org/abs/1905.09418

Waleffe, R., Byeon, W., Riach, D., Norick, B., Korthikanti, V., Dao, T., ... Catanzaro, B. (2024, June). *An Empirical Study of Mamba-based Language Models.* arXiv. Retrieved 2024-12-25, from http://arxiv.org/abs/2406.07887 (arXiv:2406.07887 [cs]) doi: 10.48550/arXiv.2406.07887

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.* Retrieved from https://arxiv.org/abs/1804.07461

Yin, K., & Steinhardt, J. (2025). *Which Attention Heads Matter for In-Context Learning?* Retrieved from https://arxiv.org/abs/2502.14010

Zani, D., Michalak, F., & Abreu, S. (2025). Contextual Sparsity as a Tool for Mechanistic Understanding of Retrieval in Hybrid Foundation Models. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference.* Retrieved from https://openreview.net/forum?id=TGWzg86kYv

A Prompt templates used in NIAH

For base models that were not instruction-tuned, we used the following template to style the prompts:

```
CONTEXT:
<haystack>

QUESTION:
<retrieval question>

ANSWER: Here is the most relevant sentence in the context:
```

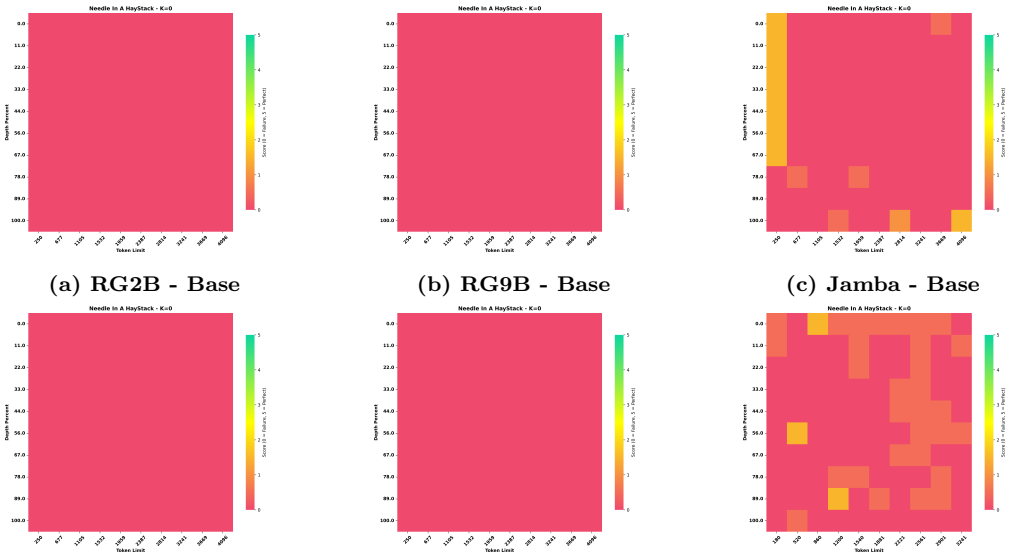
For instruction-tuned models, we changed the template to:

```
CONTEXT:
<haystack>

QUESTION:
<retrieval question> Output the most relevant sentence in the context, word by word!
```

B Comparing retrieval maps for JRT and standard NIAH

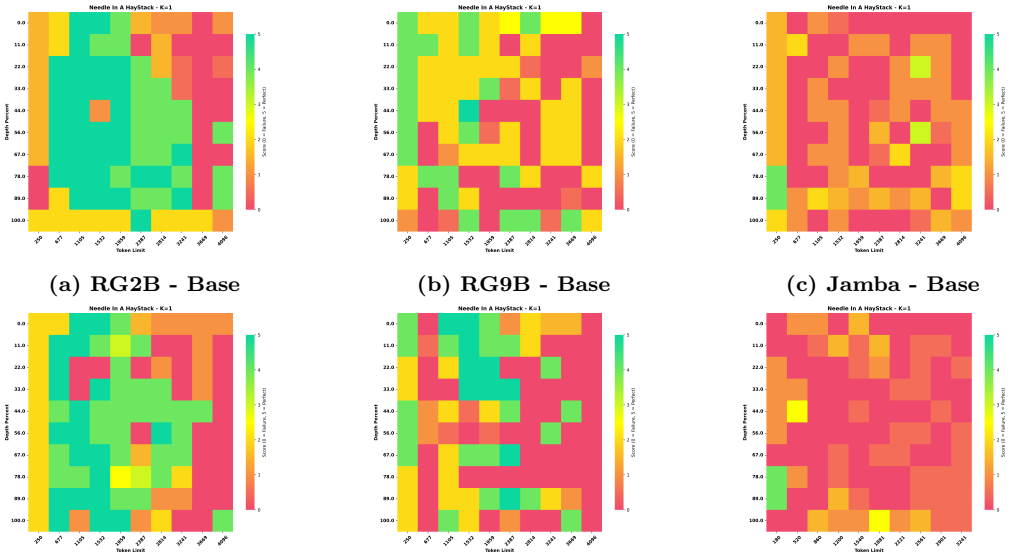
In Section 4, we are testing if applying JRT improves retrieval capabilities for top-k-sparsified RG2B, RG9B and Jamba. Figure B.1, B.2, and B.3 compare the retrieval maps of standard NIAH and JRT-applied NIAH on the same k on all models. Refer to Figure 6.2 for an explanation of the axes.



(a) RG2B - Base **(b) RG9B - Base** **(c) Jamba - Base**

(d) RG2B - JRT **(e) RG9B - JRT** **(f) Jamba - JRT**

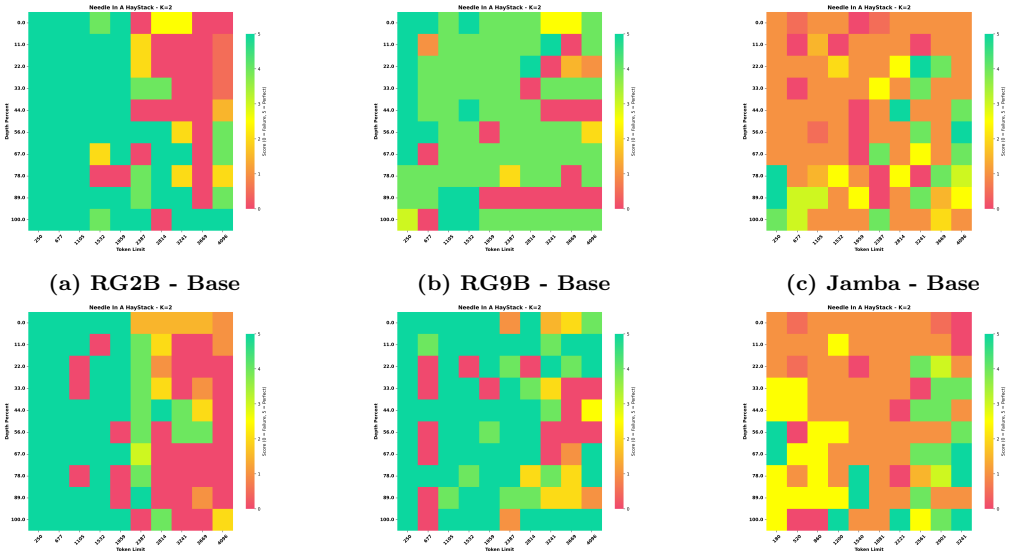
Figure B.1: Retrieval maps for RG2B, RG9B and Jamba at $k = 0$ on NIAH with and without JRT applied.



(a) RG2B - Base (b) RG9B - Base (c) Jamba - Base

(d) RG2B - JRT (e) RG9B - JRT (f) Jamba - JRT

Figure B.2: Retrieval maps for RG2B, RG9B and Jamba at $k = 1$ on NIAH with and without JRT applied.



(a) RG2B - Base (b) RG9B - Base (c) Jamba - Base

(d) RG2B - JRT (e) RG9B - JRT (f) Jamba - JRT

Figure B.3: Retrieval maps for RG2B, RG9B and Jamba at $k = 2$ on NIAH with and without JRT applied.

C LME Benchmark

In Section 5, we tested Jamba using LME. It is an instruction-tuned model, so evaluation was run with the `apply_chat_template` and `fewshot_as_multiturn` flags. Figures C.1 and C.2 show the performance

of Jamba in the ablated, sparsified and base version per subtask on GLUE and MMLU.

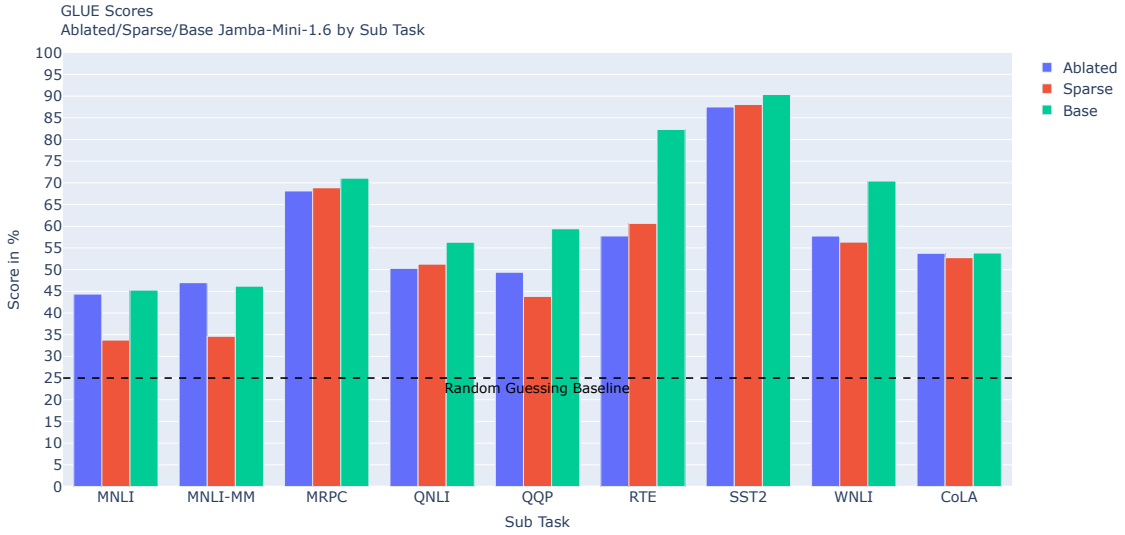


Figure C.1: GLUE benchmark Sub Task scores of the ablated ($k = 0$), sparse ($k = 5$) and base configuration.

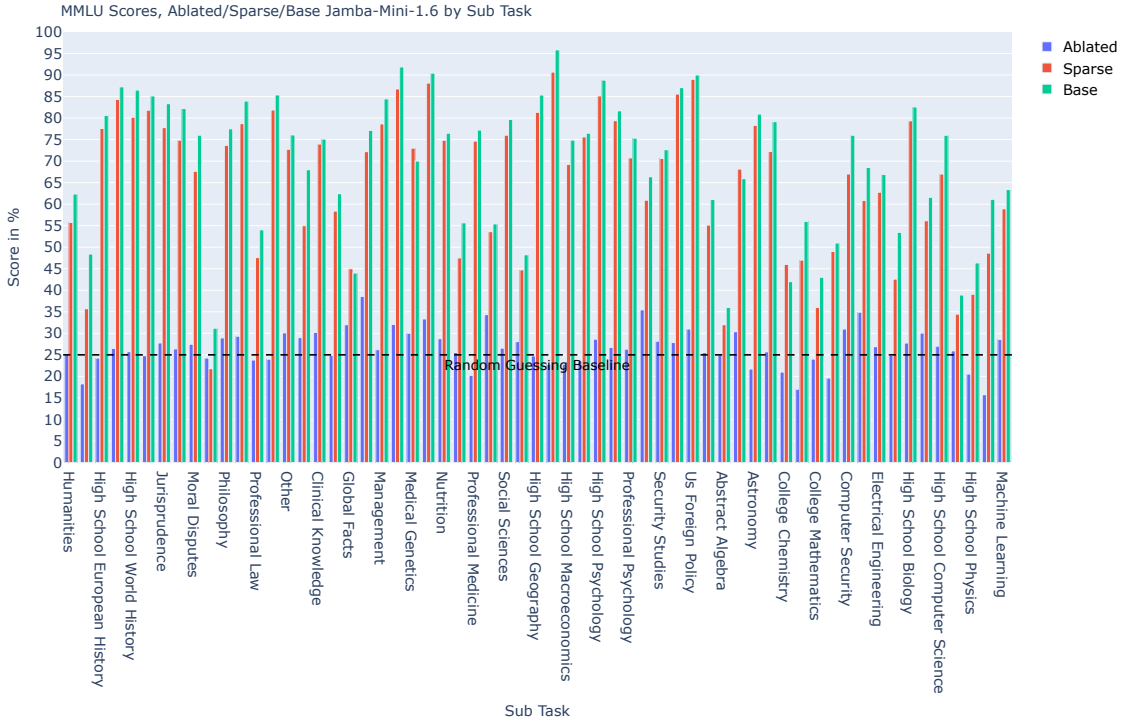


Figure C.2: MMLU benchmark Sub Task scores of the ablated ($k = 0$), sparse ($k = 5$) and base configuration.

D Retrieval maps for attention weight manipulation

Figure D.1: Retrieval maps for all manipulation combinations used in Section 6.

In Section 6, we systematically manipulated the attention weights in RG2B during NIAH benchmarks. Figure D.1 shows the retrieval maps of those benchmarks, structured in the same way as Table 6.1 for easy comparison.

E Pearls of Wisdom

Investigating and experimenting with manipulated models often results in amusing outputs. Since life is about joy, and reading some of those outputs definitely caused us joy, we distilled a collection of output strings to share our laughter with the world. Some quotes are also absurdly wise, which led us to believe they were part of the Paul Graham essays that were used to fill the context in the NIAH benchmark. However, none of the generated strings were found in the context strings. All following quotes were generated by sparsified or manipulated RG2B or RG9B, in response to the question: "What is the best thing to do in San Francisco?"

The best way to get to the top of the mountain is to start by walking.
- Mark Twain
The first thing I want to say is that I am not a native English speaker

---

RG2B - Binary-Keep

"The most important thing is to be kind. It is the little things that make the difference."
- Unknown

RG2B - Binary-Keep

The first sentence is a sentence that is not a sentence. It is a sentence that has been written in a way that is not a sentence.

RG2B - Binary-Null

"The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day on a sunny day."
The above is a joke.

RG2B - Only-Keep

I am a programmer. I am a programmer. I am a programmer. I am a programmer. I am a programmer. I am a programmer. I am a programmer.

RG2B - $k = 0$, sparse prefill

You need humility to know when to use qualification.

RG9B - $k = 4$