Transformers are Multi-State RNNs

Matanel Oren[*,H] **Michael Hassid**[*,H,M] **Yossi Adi**[H,M] **Roy Schwartz**[H]

[H]The Hebrew University of Jerusalem [M]FAIR, AI at Meta

Abstract

Transformers are considered conceptually different compared to the previous generation of state-of-the-art NLP models—recurrent neural networks (RNNs). In this work, we demonstrate that decoder-only transformers can in fact be conceptualized as infinite multi-state RNNs—an RNN variant with unlimited hidden state size. We further show that pretrained transformers can be converted into *finite* multi-state RNNs by fixing the size of their hidden state. We observe that several existing transformers cache compression techniques can be framed as such conversion policies, and introduce a novel policy, TOVA,[1] which is simpler compared to these policies. Our experiments with several long range tasks indicate that TOVA outperforms all other baseline policies, while being nearly on par with the full (infinite) model, and using in some cases only 1/8 of the original cache size. Our results indicate that transformer decoder LLMs often behave in practice as RNNs. They also lay out the option of mitigating one of their most painful computational bottlenecks—the size of their cache memory. We publicly release our code.[2]

1 Introduction

Not so long ago, transformers (Vaswani et al., 2017) replaced recurrent neural networks (RNNs; Elman, 1990) as the go-to architecture for NLP. Transformers are considered conceptually different than RNNs, as they have direct access to each token in the sequence, rather than RNNs that maintain a recurring state of previous inputs. Recently, *decoders* became a dominant transformer variant (Touvron et al., 2023a; Jiang et al., 2023). These typically generate their output auto-regressively, where the generation of each token depends on the key and value computation of previous tokens.[3]

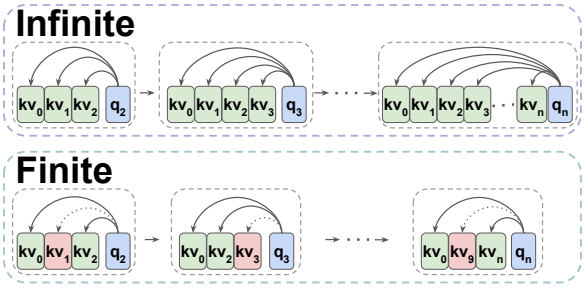

Figure 1: Top: transformers can be thought of as infinite multi-state RNNs (MSRNNs), with the key/value vectors corresponding to a multi-state that dynamically grows infinitely (green elements). Bottom: transformers behave in many cases as *finite* MSRNNs (bottom), which keep a fixed-size multi-state (here of size 2) by dropping one state (red element) at each decoding step.

In this work, we demonstrate that this auto-regressivity aligns with the core principle of RNNs—preserving a state from one step to the next one. Based on this observation, we formally redefine decoder-only transformers as a form of *multi-state* RNNs (MSRNN)—a generalized version of traditional RNNs. Importantly, as the number of previous tokens grows with each decoding step, transformers correspond to MSRNNs with an *infinite* number of states (Fig. 1, top). We continue by showing that transformers can be compressed into *finite* MSRNNs by limiting the number of tokens processed at each step (Fig. 1, bottom). We then consider previous work, which applied compression policies that effectively limit this capacity in pretrained transformer-base LLMs (Zhang et al., 2023; Xiao et al., 2023; Han et al., 2023). Our definition frames these works as converting pretrained transformers from infinite into finite MSRNNs.

We continue by proposing TOVA, a simpler

[*]Equal contribuation

[1]**T**oken **O**mission **V**ia **A**ttention. Literally "good" in Hebrew.

[2]https://github.com/schwartz-lab-NLP/TOVA

[3]These previous computations are often cached for efficiency purposes, referred to as KV caching (Radford et al., 2019; Pope et al., 2022). We note that the arguments we make in this work apply similarly to non-cached implementations.

yet more powerful MSRNN compression policy. TOVA selects which tokens to keep in the multi-state based solely on their attention scores. We evaluate TOVA on four long range tasks. Our results show that it outperforms all existing policies, and leads to minimal performance degradation compared to the infinite MSRNN, using, in some cases, as little as 1/8–1/4 of the context.

We finish by analyzing the tokens kept in memory by our method. Unlike previous work (Xiao et al., 2023; Zhang et al., 2023), we observe that not all recent tokens are important to keep in memory, and some can be safely dropped. Moreover, we show the importance of keeping the very first token in the sequence, and highlight other, perhaps surprising important tokens such as possessive nouns.

Our results shed light on the behavior of transformer decoder LLMs; while they are trained as infinite MSRNNs, they often perform in practice as finite MSRNNs. Our results also have practical benefits. Our proposed method substantially reduces memory consumption during inference, leading to up to 88% reduction in LLM cache size.

2 Background

We briefly introduce RNNs (Sec. 2.1) and transformers (Sec. 2.2). Throughout this work, we assume a model with a hidden dimension size d.

2.1 RNNs

Recurrent Neural Networks (RNNs; Elman, 1990) are a family of deep learning architectures that process sequential data in a recurrent manner. In the most general form, each layer l (often called a "cell") is modeled as a function f^l_{RNN} that receives at time t two inputs: x^l_t, a representation of the token t, and h^l_{t-1}, the hidden state from the previous time step. It then outputs two values: x^{l+1}_t, an updated token representation, and h^l_t, a new hidden state:

$$x^{l+1}_t, h^l_t = f^l_{\text{RNN}}(x^l_t, h^l_{t-1}) \quad (1)$$

h^l_t is used for the recurrent computation over the next token x^l_{t+1}, while x^{l+1}_t is used as input for the next layer. It is common, though not necessary, to set $x^{l+1}_t := h^l_t$, i.e., the input for the following layer and the hidden state are the same.

2.2 Transformers

Transformers (Vaswani et al., 2017) also process sequential data, but do so non-recurrently. A transformer layer f^l_{TRANS} takes as input a sequence of token representations: $X^l = (x^l_1, ..., x^l_t) \in \mathbb{R}^{t\times d}$, and returns a transformed representation:

$$X^{l+1} = f^l_{\text{TRANS}}(X^l) = \text{FF}^l\big(\text{SelfAttn}^l(X^l)\big) \quad (2)$$

Each transformer layer consists of two main components: self-attention (SelfAttn^l) and Feed-Forward (FF^l).[4] The former operates over the entire sequence, while the latter on each token individually. Self-attention projects the input into three matrices: $Q^l, K^l, V^l \in \mathbb{R}^{t\times d}$ and computes:[5]

$$X^l_{attn} = \text{Attn}(Q^l, K^l, V^l) \quad (3)$$

$$= \underbrace{\text{Softmax}\big(Q^l \cdot (K^l)^T\big)}_{A^l} \cdot V \quad (4)$$

where $A^l \in \mathbb{R}^{t\times t}$, the attention matrix, computes the interactions between tokens within a sequence.

Transformer decoders Decoders are the focus of this work. They mask the upper triangular part of the attention matrix in order to perform next-token prediction. When applying auto-regressive decoding, it is common to cache the K, V matrices in order to avoid recomputing the previous tokens.

3 Transformers as Multi-State RNNs

We start by formally defining a new RNN variant, Multi-State RNN (*MSRNN*; Sec. 3.1). We then demonstrate that transformers can be viewed as a special case of MSRNNs with an infinite number of states (Sec. 3.2). We continue by discussing *finite* MSRNNs, and present several policies for converting pretrained transformers into them (Sec. 3.3). Finally, we introduce TOVA—a novel and simple finite MSRNN policy (Sec. 3.4).

3.1 Multi-State RNNs

We define an MSRNN to be an RNN with a state *matrix* instead of a vector: $H^l_t \in \mathbb{R}^{g(t)\times d}$. The size of H^l_t is parameterized by a function g, which can either be constant (i.e., a fixed size matrix) or input-dependent.[6] The MSRNN equation corresponding to Eq. (1) is:

$$x^{l+1}_t, H^l_t = f^l_{\text{MSRNN}}(x^l_t, H^l_{t-1}) \quad (5)$$

[4] Layer normalization and skip connections are omitted.

[5] The attention mechanism typically uses multiple heads in each layer. We omit head subscripts for readability.

[6] Note that we could unroll the matrix and define it as a single vector in $\mathbb{R}^{g(t)\cdot d}$ and use the traditional RNN terminology, but we find it more convenient to think of it as a matrix.

We can interpret each row of H_t^l as a single-state, allowing us to think of H_t^l as a multi-state matrix. By defining $g(t) = 1$ for all t, MSRNN reduces to a standard (single-state) RNN.

3.2 Transformers are Infinite MSRNNs

Consider the case where $g(t) = t$, in which the number of single-states equals the number of input tokens in the corresponding time-step. In this setup, we can view the transformer as an MSRNN, where $H_t^l = (K_t^l, V_t^l)$ and the layer computation is:

$$(K_t^l, V_t^l) = \left(\begin{pmatrix} K_{t-1}^l \\ k_t^l \end{pmatrix}, \begin{pmatrix} V_{t-1}^l \\ v_t^l \end{pmatrix}\right) \quad (6)$$

$$x_t^{l+1} = \text{FF}^l\left(\text{Attn}^l(q_t^l, K_t^l, V_t^l)\right) \quad (7)$$

Where q_t^l, k_t^l, v_t^l are the self-attention projections of x_t^l, and each single-state of (K_t^l, V_t^l) corresponds to a specific token. Combined, we get the MSRNN equation for transformers:[7]

$$x_t^{l+1}, (K_t^l, V_t^l) = f_{\text{TRANS}}^l\left(x_t^l, (K_{t-1}^l, V_{t-1}^l)\right) \quad (8)$$

It should be noted that in practice, transformer models are trained up to a specific length and often struggle to extrapolate beyond that (Press et al., 2022). However, in theory, they possess the capacity to handle infinite-length inputs, and thus correspond to an *infinite* size MSRNN.

3.3 Converting Pretrained Transformers into Finite MSRNNs

Our view of transformers as MSRNNs with an infinite number of states raises an interesting question: do pretrained transformers actually make use of this infinite capacity? To address this question, we define a *finite* MSRNN by setting $g(t) = \min(t, k)$ for some constant k. In that case, when the number t exceeds k, a compression policy is applied in order to fit the context into the memory restriction. This policy can be pre-defined, or learned during training, like in some RNN variants (e.g., LSTM; Hochreiter and Schmidhuber, 1997).

Interestingly, we observe that several compression policies have been proposed, which could be used to convert an infinite MSRNN into a finite one. We note that none of them discussed the connection between their policy and RNNs.

[7]We note a practical difference between RNNs and transformer MSRNNs is that in the former the state is typically contextualized by the hidden representations in the current layer, while each state in the latter is based on the contextualization of the *previous* layer.

Window This policy implements a First In First Out (FIFO) strategy (Beltagy et al., 2020). When the multi-state reaches its capacity limit, the oldest state (i.e., the earliest token) is discarded, ensuring that only the most recent states are retained.

Window+i This policy extends the Window policy by also retaining the first i tokens, for some constant i. Previous work (Xiao et al., 2023; Han et al., 2023) has shown that Window+i strongly outperforms Window using as few as four early tokens.

H_2O This policy (Zhang et al., 2023), much like Window+i, keeps a fixed window of the recent tokens, as well as additional earlier tokens. Unlike Window+i, it dynamically selects the non-window tokens by aggregating the attention scores throughout the sequence, and keeping the tokens with the highest aggregated scores. The number of non-window tokens is typically set as half of the multi-state size. H_2O can operate head-wise or layer-wise (by averaging the attention scores across heads). Preliminary results (App. A) indicate that both variants perform similarly, so we follow Zhang et al. (2023) and use the head-wise version.

3.4 Our Proposed Policy: TOVA

The policies presented so far introduce strong inductive biases; they both devote a substantial part of the state towards the most recent tokens, and prefer tokens appearing early in the sequence.[8]

We introduce a new policy, Token Omission Via Attention (TOVA), which is simpler than previously proposed methods. TOVA retains the top states based on the attention weights of the last token only. At each decoding step, we consider the softmax-normalized attention scores from the current query to all the tokens currently in the multi-state, plus the current token. The token with the lowest score is dropped. See Fig. 2 for illustration.

TOVA makes fewer assumptions compared to the policies above: it neither fixes a window of recent tokens, nor favors early tokens. We note that it does introduce a weak recency bias, as early tokens require high attention scores in all subsequent decoding steps in order to be kept in the multi-state, and thus have a higher chance of being dropped. Our analysis (Sec. 6) shows that despite this bias, a substantial fraction of the recent tokens are dropped

[8]Note that H_2O aggregates the attention weights, which strongly favors initial tokens, as they accumulate more attention scores as the sequence progresses. This effectively leads to a policy similar to Window+i.

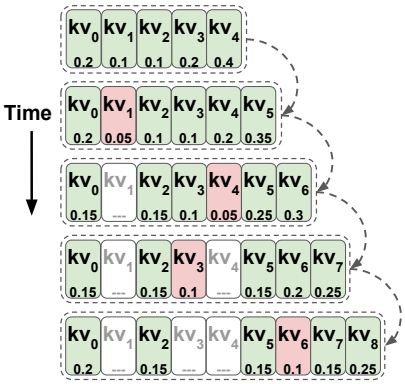

Figure 2: Illustration of the TOVA policy, which keeps a fixed-size multi-state (green cells). For a given attention layer, at each decoding step, the state with the lowest attention score is omitted (red cells, which become transparent in the following steps).

by TOVA, indicating that perhaps a fixed recent window is too strict. Further, some initial tokens are kept for thousands of decoding steps, indicating that they are indeed important for successful decoding (Xiao et al., 2023; Han et al., 2023).

We note that, much like H_2O, TOVA can operate head-wise or layer-wise. Unlike H_2O, here our preliminary results show a clear preference in favor of the layer-wise variant (App. A), which we therefore report. In the following, we show that TOVA dominates the other policies, obtaining near-similar results to the corresponding infinite MSRNN (a regular pretrained transformer).

4 Experimental Setup

Our goal is to check whether any compression policy (finite MSRNN) can match the performance of the full, infinite MSRNN model. To address this question, we evaluate the different compression policies (Sec. 3.3).[9] Below we describe the benchmarks we use (Sec. 4.1) and the transformer LLM families we experiment with (Sec. 4.2).

4.1 Long Range Evaluation

Our experimental setup focuses on long range evaluation, which would trigger the different policies. Below we describe the three types of long-range evaluations we employ: language modeling, long-range understanding, and generation of long texts.

[9]We highlight that our focus is on the capacity of off-the-shelf models, so we only consider policies that operate on pretrained LLMs and require no additional training. See Sec. 7 for approaches that do require training.

Language modeling We report perplexity on the PG-19 test set (Rae et al., 2020), a widely used benchmark for evaluating long range language models (So et al., 2022; Hutchins et al., 2022; Chen et al., 2023). PG-19 is composed of 100 full-length books of average length of 70k tokens.

Long range understanding We use two test sets from the ZeroSCROLLS benchmark (Shaham et al., 2023), each focusing on a different aspect of long range understanding: long range summarization and long range question answering (QA).

For the former, we use SQuALITY (Wang et al., 2022), a question focused summarization dataset. Following Shaham et al. (2023), we report the geometric mean of ROUGE-1/2/L scores (based on the ground truth summary). For the latter, we use QASPER (Dasigi et al., 2021), a QA dataset based on Semantic Scholar Open Research Corpus (S2ORC; Lo et al., 2020). As a QA task over long texts, this task can be considered a retrieval task, as the model needs to retrieve the relevant information from the text to answer the question. We follow Dasigi et al. (2021) and report F1 score. See App. C for the prompts used for both tasks.

Text generation We feed the models with prompts that solicit the generation of a long story. We sample 100 unique stories from each version of the model, using different seeds. Given the complexity of comparing two stories, we follow Chiang et al. (2023) and Zhou et al. (2023) and employ GPT-4 as an evaluator. For each seed, we compare the two generated stories by asking GPT-4 to evaluate which is better, reporting the average win rate for each approach. We drop cases where the model stops generating before reaching the memory limit, as both stories are identical. To account for GPT-4's positional bias (Wang et al., 2023), we present each pair of stories twice, alternating their positions, and only consider a "win" if the same approach is preferred in both cases. See App. C for the prompts used for generation and evaluation.

4.2 Models

We experiment with three state-of-the-art transformer decoder LLMs families: LLaMA-2 (Touvron et al., 2023b), Mistral (Jiang et al., 2023) and Yi (01-ai, 2023). Each family offers a ∼7B parameter version, which we use for evaluation.

For language-modeling, we use the vanilla versions of the models. For the long range understanding tasks, we also consider three fine-

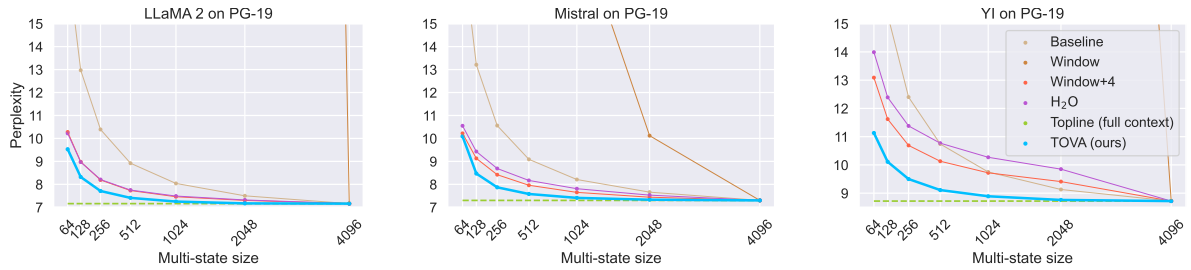

Figure 3: Perplexity results for the PG-19 test set. TOVA outperforms all other policies in all multi-state sizes, while maintaining comparable results to the full context topline using 1/8–1/4 of the context size.

tuned versions: LLaMA-2-chat (Touvron et al., 2023b), Mistral-Instruct (Jiang et al., 2023) and neural-chat,[10] which have been shown to excel in instruction-specific tasks. Lastly, for text generation, we use MythoLogic,[11] a LLaMA-2-13B version fine-tuned for story generation.

For all models and tasks, we use a maximal input-length of 4,096 tokens.[12] For the language modeling task, we split the texts into chunks of length 4,096, and apply efficient masking (see App. D). For the language understanding tasks, we truncate the end of the example (excluding prompt) if it exceeds 4,096 tokens as done in Shaham et al. (2023). All experiments are done using bfloat16 floating-point precision over Nvidia V100 GPUs.

5 Pretrained Transformers Act as Finite MSRNNs

We present our results of the different tasks: language modeling (Sec. 5.1), long-range understanding (Sec. 5.2), and long text generation (Sec. 5.3).

5.1 Language Modeling

We evaluate our base models over the language modeling task using the following policies: Window, Window+4, H_2O and our TOVA policy.[13] As a baseline, we run the models with a smaller sequence length, while not applying compression, which corresponds to an infinite MSRNN with a full sequence length smaller than 4,096. As a topline, we use the models with the full training sequence length (4,096), again without compression. We examine multi-state sizes in exponential scales of 2^j for $j \in \{6, 7, \ldots, 12\}$ (2^{12}=4,096).

Figure 3 presents the perplexity results of the different policies. Our TOVA policy outperforms all other policies using all three models in all multi-state sizes. Specifically, our policy maintains results within 0.5 perplexity points of the topline using a quarter (LLaMA-2 and Yi) or even one eighth (Mistral) of the full context length. In contrast, other policies require at least half of the full context length to reach comparable results.

As to the other policies, as observed by Han et al. (2023) and Xiao et al. (2023), the Window policy performs quite poorly, while the Window+4 and H_2O policies obtain much better results, though still substantially lower than our TOVA. In light of these findings, we proceed to evaluate other tasks using two policies: Window+4 and TOVA.

5.2 Long Range Understanding

We next evaluate instruction-tuned LLMs on SQuALITY and QASPER.[14] As a baseline, we consider a setup where the model is presented with a truncated version of the example according to the MSRNN capacity. E.g., when considering a multi-state size of 1,024 tokens, the baseline uses the example truncated to 1,024 tokens (including the prompt). We examine multi-state sizes in exponential scales of 2^j for $j \in \{8, 9, \ldots, 12\}$.

Long range summarization Results for SQuALITY are presented in Fig. 4. Our TOVA policy consistently outperforms both baselines across all multi-state sizes and models. As in language modeling, using TOVA with a quarter (Mistral and Yi) or even one eighth (LLaMA-2) of the full context yields results within one point of the topline model.

Long range QA Figure 5 shows the QASPER results. The gap between TOVA and the baselines is large, in some cases reaching beyond 5 F1 points. Nonetheless, here TOVA needs half of the full context to perform within one F1 point of the topline.

[10] huggingface.co/Intel/neural-chat-7b-v3

[11] huggingface.co/Gryphe/MythoLogic-L2-13b

[12] Due to computational constraints, we defer extrapolation experiments to future work, and approximate the infinite MSRNN with a sequence length of 4,096 tokens.

[13] We ablate other policies using LLaMA-2-7B in App. A.

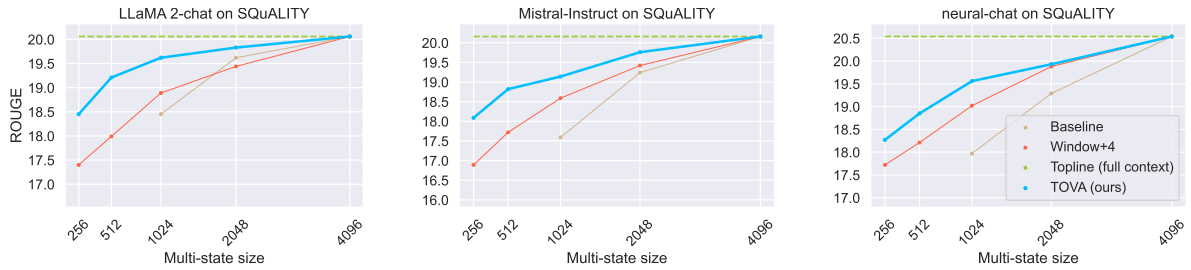

Figure 4: Geometric mean of ROUGE-1/2/L for SQuALITY. TOVA achieves within one point of the topline using $1/8 - 1/4$ of the multi-state size, while outperforming all other policies.

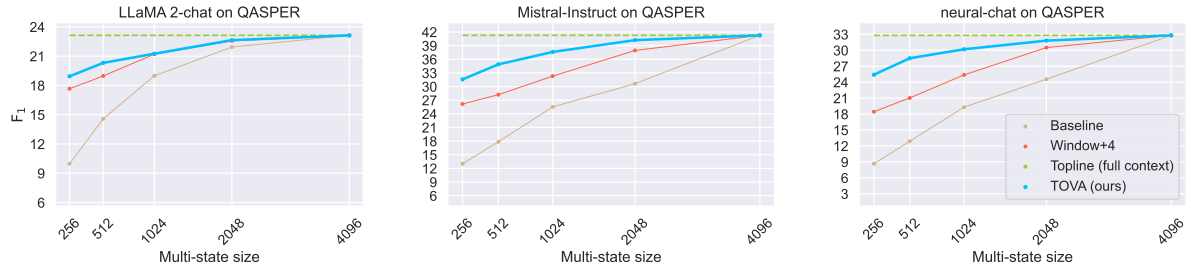

Figure 5: F1 score over QASPER benchmark. TOVA outperforms both baselines, but requires a half of the full multi-state size for obtaining comparable results to the topline.

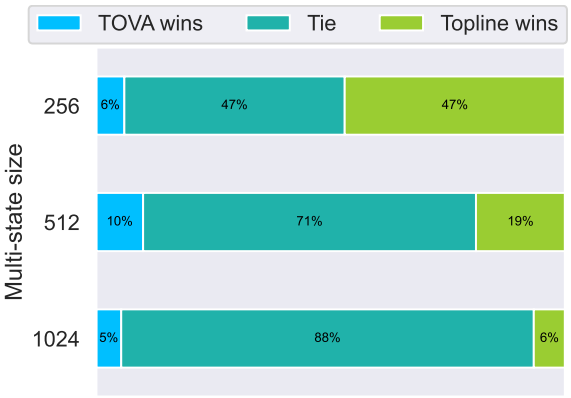

Figure 6: GPT-4 preference over stories generated by the infinite MSRNN and a finite one using TOVA.

5.3 Text Generation

Finally, we evaluate task generation. We first note that limiting the multi-state size makes the generated text shorter: the average story length for the full model is 1,566 tokens. This value is kept for a multi-state size of 1,024, but drops to 1,503 with 512 tokens and to 1,361 with 256 tokens.

Figure 6 shows the evaluation results of the stories using GPT-4. We observe that using a small multi-state size (256 tokens), our policy losses to the topline in 47% of cases, while winning or tying in the remaining cases. This loss rate decreases substantially to 19% with 512 tokens and further to only 6% with 1,024 tokens. Importantly, our policy is also preferred over the topline in 5–10% of the cases in all multi-state sizes considered.

5.4 Discussion

Our results indicate that transformer decoder LLMs, which are infinite MSRNNs, often behave empirically as *finite* MSRNNs: in 2/4 tasks, finite MSRNNs using as little as 1/8–1/4 of the tokens yield comparable results to the corresponding infinite MSRNN, despite being trained with the full context. The other two tasks, text generation and retrieval QA, seem to require longer contexts, though still maintain comparable performance using one half of the original context. This suggests that the conversion of a transformer into an RNN reintroduces the inherent challenges associated with RNNs, as they encounter difficulties with the retrieval of long range information (Hochreiter and Schmidhuber, 1997; Arjovsky et al., 2016).

6 Analysis

6.1 Which Tokens Matter?

We have shown so far that pretrained transformer LLMs, trained as infinite MSRNNs, empirically behave in many cases as *finite* MSRNNs. This effectively means that most of the tokens are dropped from memory as generation progresses. This section aims to shed light on this process, and characterize the tokens frequently kept, and those dropped. As our TOVA policy presents the best approxima-

[14] Base LLMs numbers are reported in App. B.

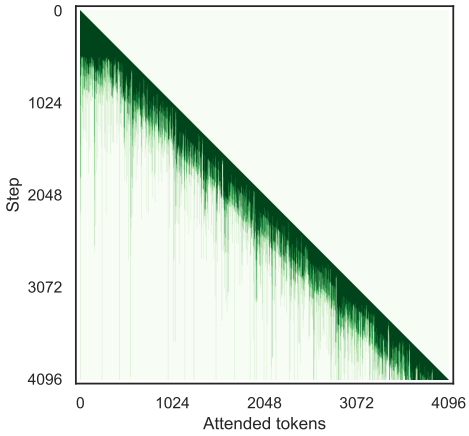

Figure 7: An illustration of the tokens kept by our policy in the last layer of LLaMA-2-7B on one PG-19 example. Each row represents a decoding step, and each column is a token attended to.

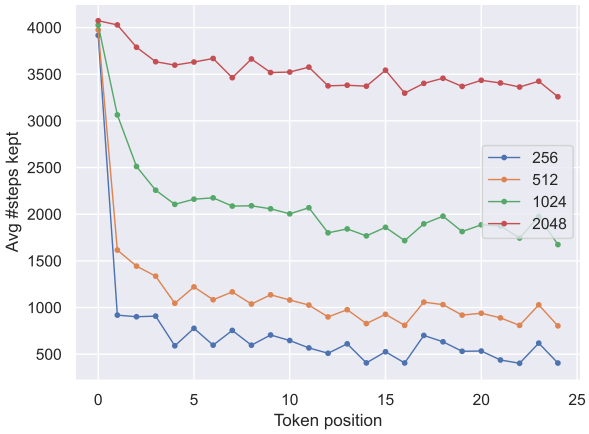

Figure 8: The average number of steps a token is kept in the multi-state when applying TOVA as a function of token position. Different lines are different multi-state sizes. The very first token is kept through the entire context, while next tokens are dropped far earlier.

tion of the full model, we analyze its results. The analysis below uses the LLaMA-2-7B model, with 31 instances from PG-19.

Recency is *not* all you need We first observe that, much like most compression policies (Sec. 3.3), TOVA preserves recent tokens. Figure 7 illustrates the tokens kept by TOVA in the final layer for one example from PG-19, using multi-state size of 512.[15] The figure shows a clear window trend, which indicates the importance of recent tokens for decoding. Nonetheless, we also observe that many older tokens are kept. To quantify this, we compute the proportion of recent tokens out of the tokens kept in the multi-state, averaging across the PG-19 examples, layers, and positions. We find that only 73-76% of the tokens are recent, the rest being older. This suggests that while recent tokens are important, they are far from sufficient. Importantly, unlike previous work that handcrafted the recent window, our method identifies it automatically. We next turn to investigate which tokens from the far history tend to be kept. We study two dimensions: the position of the token, and its content.

First token matters Figure 8 shows the number of steps kept (averaged across layers and examples) for the first 25 tokens. As observed by previous work (Han et al., 2023; Xiao et al., 2023), we find that the very first token is crucial for the model: it is kept until the end of the sequence across all multi-state sizes. However, other initial tokens (e.g., positions 2–4) are far less important. While they are kept for longer than the next tokens, they are dropped far faster than the first one.

Tag \ Multi-state size	256	512	1024	2048
Avg.	249	481	897	1537
POS	1134	1393	1736	2061
”	845	1101	1413	1774
$	329	724	1276	2123
)	379	670	1161	1558
.	350	645	1117	1677
NNPS	321	578	1042	1671
\n	303	550	969	1538

Table 1: Mean number of steps tokens are kept in the multi-state with TOVA, grouped by POS-tags. Columns represent the multi-state size. Here we report the tokens kept the longest, see full table in App. F.

Not all tokens are equally kept As indicated by Fig. 7, some tokens last much longer than others. To study this phenomenon, we map each token to its part-of-speech tag (POS-tag) using NLTK (Bird et al., 2009), and plot the tags that last longest in Tab. 1.[16] Our results show that, as observed by previous work (Clark et al., 2019; Zhang et al., 2023; Ge et al., 2023), punctuation and other special symbols tend to be kept. However, we also identify other tokens that tend to stay longer, e.g., possessive nouns (POS) and proper nouns (NNPS). Studying the role of these tokens is an exciting research direction, which we defer to future work.

[15] See App. E for the illustrations of all layers.

[16] We present the full table for all POS-tags in App. F.

6.2 Increased Batch Size using TOVA

As discussed in Sec. 2.2, caching the K, V matrices in transformer auto-regressive decoding is common in current frameworks, as is trades-off decoding speed for runtime memory. Importantly, the memory factor is determined by two elements: the model size (e.g., number of layers, hidden dimension), and the batch size. As the former is fixed, caching effectively limits the inference batch-size.

When decoding transformers as finite MSRNNs, the cached K, V matrices are compressed. As a result, reducing the multi-state size to $1/m$ of the original size can increase the batch size by a factor of m, dramatically enhancing hardware utilization. As shown in Sec. 5, TOVA allows limiting the multi-state to $1/8$-$1/4$ of the full LLMs context length without major loss in performance, therefore enabling up to an eightfold increase in batch size.

7 Related Work

Transformers and RNNs Several works have tried to bridge the gap between RNNs and transformers. Hutchins et al. (2022) employed a hybrid transformer-RNN approach that attends to recent tokens and to further hidden states simultaneously. Katharopoulos et al. (2020) and Sun et al. (2023) substituted the self-attention layer with a convolution layer that can be applied in a recurrent manner. Peng et al. (2023) adjusted the self-attention layer to maintain an RNN nature at inference time.

Perhaps most relevant to this work, ABC (Peng et al., 2022) presented transformer models with bounded memory. They also showed that several transformer variants such as Linformer (Wang et al., 2020) and Window attention can be interpreted as instances of their framework. However, these models typically treat the memory as a single state and not as multi-state, meaning there is no explicit mapping from tokens to states. Importantly, unlike our approach, both ABC and the other works above require a dedicated training procedure, and cannot operate on existing transformer-based LLMs.

New RNN variants Recent work aimed to revive RNNs in NLP. The most prominent work is S4 (Gu et al., 2022) and its successors (Gupta et al., 2022; Mehta et al., 2023; Gu and Dao, 2023), which elevate state spaces to form linear RNNs. Other works introduced novel RNN variants that train effectively while reducing inference cost (Merity, 2019; Orvieto et al., 2023; Yang et al., 2023).

Limited KV cache and window attention Window attention (Beltagy et al., 2020; Zaheer et al., 2020) is a simple way of constructing a transformer with limited cache requirement. As mentioned in Sec. 3.3, the H_2O policy (Zhang et al., 2023) can be used to limit the cache size of transformers. A very recent followup work (Ge et al., 2023) showed that manually caching specific tokens like "." and "," further boosts H_2O performance. We showed that TOVA does so without manually selecting tokens (Sec. 6.1). Anagnostidis et al. (2023) introduced a learned approach over LLMs that limits the cache consumption of transformers. Lastly, Xiao et al. (2023) and Han et al. (2023) showed that the Window+i policy enables models to extrapolate beyond their training sequence length. The application of TOVA for extending the context size of LLMs is an intriguing topic for future research.

Simplifying transformers Previous work has shown that many transformer attention heads can be pruned (Michel et al., 2019; Li et al., 2021). Hassid et al. (2022) showed that the dynamic attention in transformers can be replaced with static weights without major drop in performance. Several works replaced the attention mechanism in transformers with an efficient variant (Liu et al., 2021; Lee-Thorp et al., 2022). We show that transformer decoders can reduce to finite MSRNNs.

8 Conclusion

In this work, we redefined decoder transformers as a form of multi-state RNNs (MSRNN) with an infinite multi-state size. We highlighted that limiting the number of token representations transformers can handle at each step is equivalent to compressing it from infinite to finite MSRNNs.

We then introduced TOVA, a conceptually simple compression method that selects which tokens to keep using their attention scores. Our findings highlight its superior performance compared to existing compression policies. Moreover, we showed that in many cases, TOVA performs comparably to the infinite MSRNN model, while requiring 1/8–1/4 of the multi-state size. Notably, our results demonstrate that, although transformers are not trained as such, they often function as finite MSRNNs.

Our findings shed light on the inter-working of transformers, and their connections to RNNs. They also have practical value—they can dramatically reduce the LLM cache size by up to 88%.

Limitations

Evaluating models on long text generation is computationally expensive and might limit others from reproducing our results. Further, the evaluation of such task is extremely complicated, even for humans. We therefore resort to GPT-4 to compare the output of our TOVA policy compared to the topline model (Sec. 5.3). We recognize that this is far from perfect, and will most likely not catch the full breadth of evaluating text quality. Finally, our evaluation framework focuses on English tasks. It is not unlikely that languages with more flexible word order will make different use of the attention mechanism, and thus potentially require a larger multi-state size.

Ethics Statement

Our work has the potential to dramatically reduce the memory footprint of transformer LLMs, thereby potentially increasing their adoption by users with limited hardware access.

This work does not collect any new data, and only uses open source models, and public data collected by other sources.

Acknowledgements

We thank Miri Varshavsky Hassid for the great feedback and moral support. This work was supported in part by NSF-BSF grant 2020793.

References

01-ai. 2023. 01-ai/yi-6b. https://github.com/01-ai/Yi.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. Dynamic context pruning for efficient and interpretable autoregressive transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. arXiv:2004.05150.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. arXiv:2306.15595.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@NeurIPS 2023)*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. arXiv:2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. arXiv:2111.00396.

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. LM-Infinite: Simple on-the-fly length generalization for large language models. arXiv:2308.16137.

Michael Hassid, Hao Peng, Daniel Rotem, Jungo Kasai, Ivan Montero, Noah A. Smith, and Roy Schwartz. 2022. How much does attention actually attend? questioning the importance of attention in pretrained transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1403–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. arXiv:2310.06825.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2022. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4296–4313, Seattle, United States. Association for Computational Linguistics.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to MLPs. In *Advances in Neural Information Processing Systems*, volume 34, pages 9204–9215. Curran Associates, Inc.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2023. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*.

Stephen Merity. 2019. Single headed attention RNN: Stop thinking with your head. arXiv:1911.11423.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. arXiv:2303.06349.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2022. ABC: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7469–7483, Dublin, Ireland. Association for Computational Linguistics.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. arXiv:2211.05102.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. arXiv:2305.14196.

David R. So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. 2022. Primer: Searching for efficient transformers for language modeling. arXiv:2109.08668.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. arXiv:2307.08621.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R. Bowman. 2022. SQuALITY: Building a long-document summarization dataset the hard way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1156, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. arXiv:2305.17926.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. arXiv:2006.04768.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. arXiv:2309.17453.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2023. Gated linear attention transformers with hardware-efficient training. arXiv:2312.06635.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. arXiv:2306.14048.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. arXiv:2305.11206.

A Ablation over policies

We ablate all policies presented in Sec. 3.3 with the language modeling task, specifically we examine: Window, Window+i policies for $i \in \{1, 4\}$, H_2O policy for both per layer and per head approaches and our TOVA policy for both per layer and per head approaches. We also combine our approach with additionally fixing the first i tokens using $i \in \{1, 4\}$. We consider the same baseline policy as in Sec. 5.1. We use the LLaMA-7B as the backbone model.

Our results are presented in Tab. 2. As shown in Sec. 5.1 the Window policy fails, while the Window+$1/4$ policies maintain much better results (with Window+4 performs slightly better). The two H_2O policies—head/layer produce similar results. Regarding our TOVA policies, the head version performs worse than former policies in most Multi-State sizes, while the layer version outperforms all other policies. We attribute this difference to the more robust selection mechanism employed by the layer version, which requires agreement among all heads to determine the importance of specific tokens. Lastly, when we enhance our TOVA policy with the explicit preservation of i initial tokens, the results remain relatively unchanged, implying that our policy inherently retains the crucial tokens.

B Long Range Understanding with base models

Fig. 9 and Fig. 10 shows the results for base models over the SQuALITY and QASPER benchmarks respectively.

C Prompts

The prompts used for the different evaluations through this work are presented in Tab. 3. To evaluate the generated texts, using GPT-4, we use the gpt-4-0613 version.

D Language Modeling - Experimental Details

To effectively parallelize the language modeling task for all tokens in the sequence, we modify the attention mask to incorporate the different MSRNN policies presented in Sec. 3. Specifically, for Window and Window+i policies, we apply a static masking, as the reduced tokens are independent with respect to the attention computation. For H_2O and TOVA, we adjust the mask according to the attention weights of the relevant layer.

E TOVA illustration for the full model

Fig. 11 and Fig. 12 shows illustrations for which tokens are attended (X axis) at each step (Y axis) for every layer of LLaMA-2-7B, when applying TOVA with multi-state size of 512, on next-token prediction over one PG-19 example.

F Full POS-tags analysis

The full version of Tab. 1 presented in Tab. 4.

Policy \ Multi-state size	64	128	256	512	1024	2048	4096
Baseline	17.65	12.97	10.39	8.92	8.04	7.50	**7.16**
Window	4812.27	4025.01	3275.58	2184.62	1001.29	240.17	**7.16**
Window+1	10.20	8.97	8.22	7.76	7.50	7.33	**7.16**
Window+4	10.28	8.98	8.19	7.73	7.46	7.30	**7.16**
H_2O-head	10.22	8.97	8.21	7.75	7.49	7.32	**7.16**
H_2O-layer	10.20	8.97	8.22	7.76	7.50	7.33	**7.16**
TOVA-head	11.13	9.55	8.69	7.90	7.52	7.27	**7.16**
TOVA-layer	**9.53**	**8.32**	**7.71**	**7.41**	**7.25**	**7.17**	**7.16**
TOVA-layer+1	**9.53**	**8.31**	**7.71**	**7.41**	**7.25**	**7.17**	**7.16**
TOVA-layer+4	9.63	**8.33**	**7.72**	**7.41**	**7.25**	**7.17**	**7.16**

Table 2: Perplexity over the PG-19 set using varying multi-state sizes (maximal number of states used). Our TOVA policy dominates the table in all multi-state sizes.

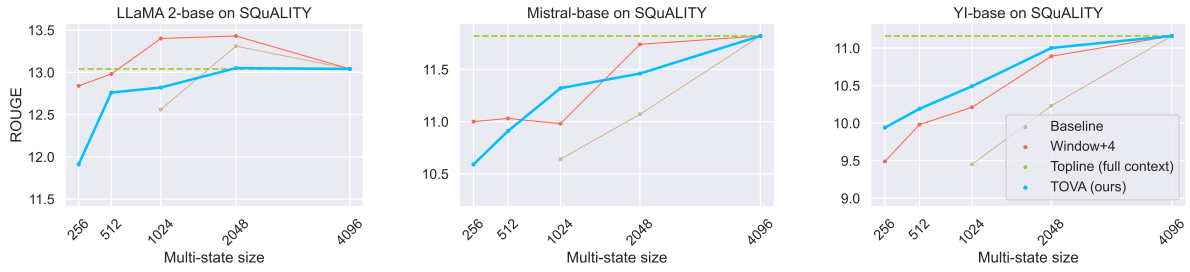

Figure 9: Geometric mean of ROUGE-1/2/L for SQuALITY using base models.

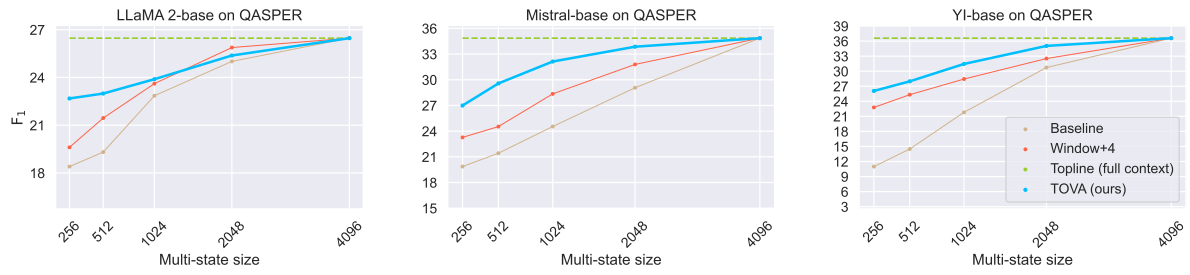

Figure 10: F1 score over QASPER benchmark using base models.

Task	Prompt
SQuALITY	{Story} Answer the question in a paragraph. Question: {Question} Answer:
QASPER	{Article} Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable". Question: {Question} Answer:
Story Generation	### Instruction: Write a very long story (at least 4,000 words). The story should include at least 20 named characters, spans 3 countries and 9 cities, at least 10 chapters and should have a lot of plot twists. ### Response:
GPT- Evaluation	Help me decide which response is better given the prompt: {Prompt body for story generation} Which of the following responses is better (the responses are separated by '————————'): Response (A): {First Response} ———————— Response (B): {Second Response} Comparing these two responses, which response is better (A), (B) or (C) for equal quality? please select one and only one option, be as concisely as you can, using a single phrase.

Table 3: Prompts used for our experiments.

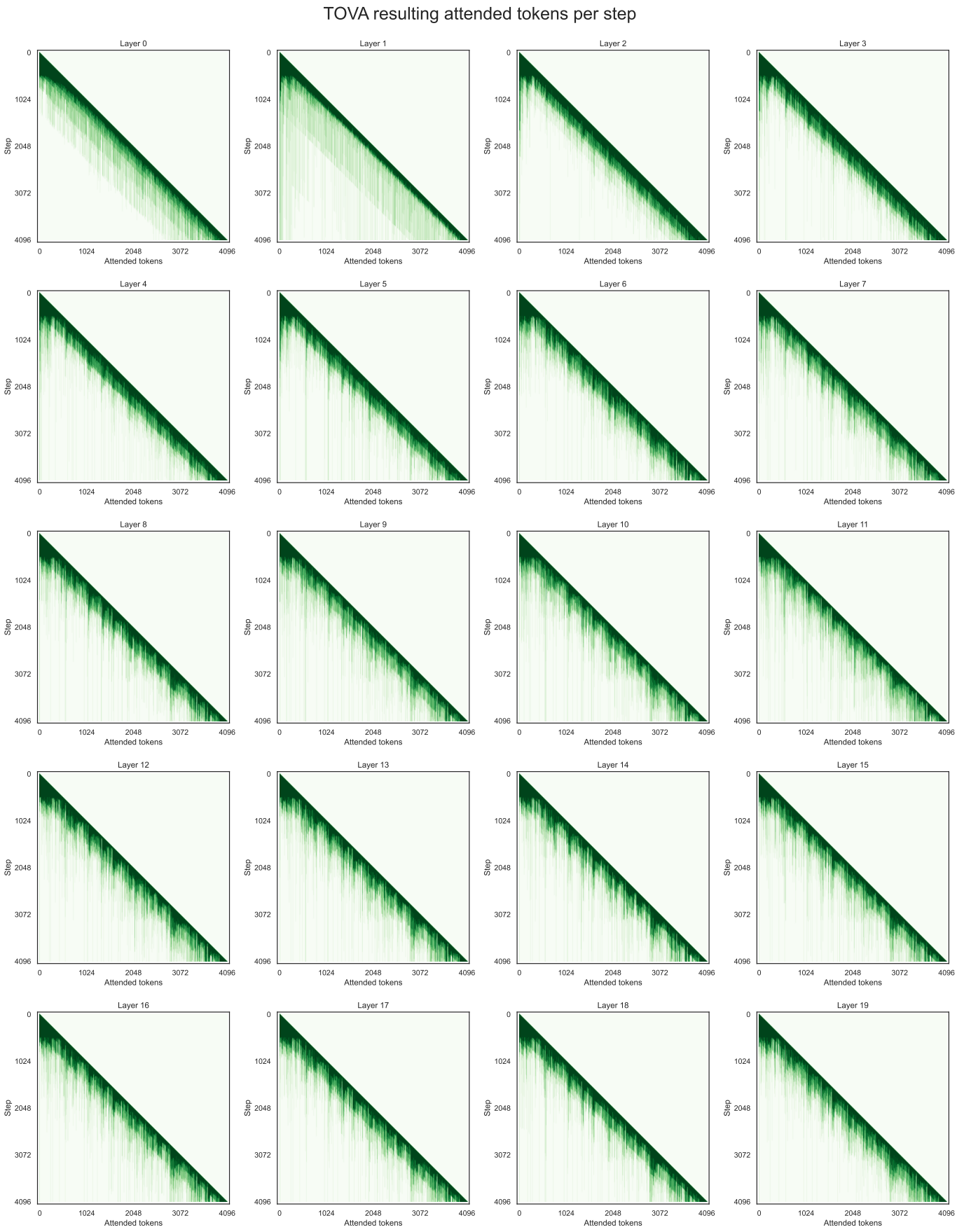

Figure 11: Attended tokens with TOVA. Layers 0-19.

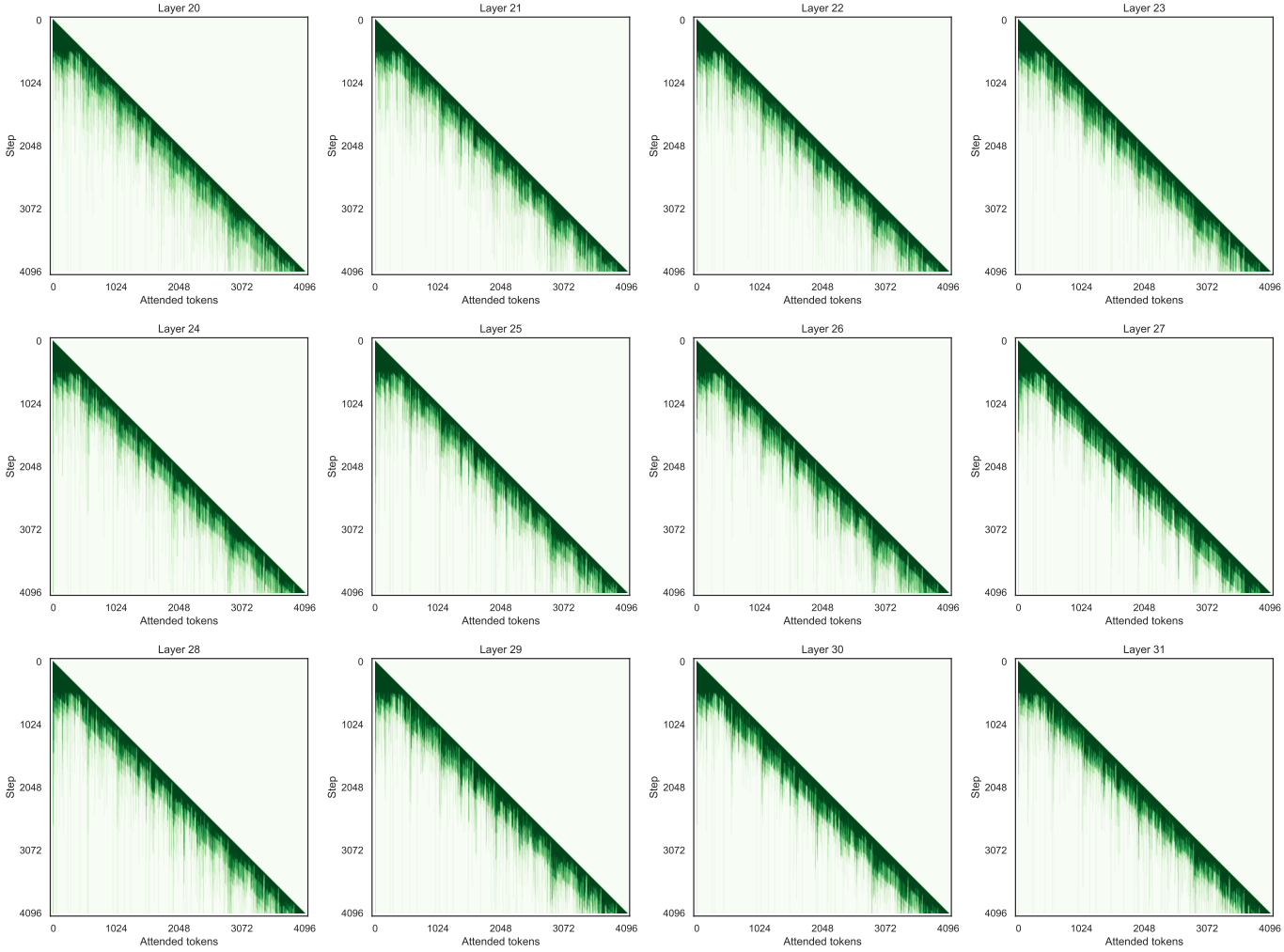

Figure 12: Attended tokens with TOVA. Layers 20-31.

Tag \ Multi-state size	256	512	1024	2048
Avg.	249	481	897	1537
POS	1134	1393	1736	2061
”	845	1101	1413	1774
$	329	724	1276	2123
)	379	670	1161	1558
.	350	645	1117	1677
NNPS	321	578	1042	1671
\n	303	550	969	1538
WP$	255	539	1121	1920
CD	301	537	940	1557
NN	270	527	983	1628
NNS	270	526	978	1618
NNP	270	517	951	1613
FW	253	511	903	1444
:	243	492	940	1570
JJ	240	480	918	1598
VBP	244	478	882	1504
JJS	220	475	953	1689
UH	233	474	870	1412
SYM	231	471	893	1482
WDT	223	462	903	1604
VBN	230	462	887	1549
EX	244	461	847	1461
RB	223	459	892	1566
,	236	453	840	1454
VBG	221	445	858	1523
RBS	210	441	878	1645
VBZ	219	440	844	1492
CC	217	437	862	1546
VBD	217	432	827	1493
VB	214	426	817	1457
PRP	217	424	794	1432
RP	207	417	811	1485
WRB	207	415	800	1502
WP	199	405	803	1506
JJR	195	403	782	1413
RBR	183	397	821	1566
PDT	181	391	756	1362
IN	190	385	760	1408
PRP$	189	383	745	1386
DT	190	379	734	1363
MD	177	378	754	1392
TO	182	368	734	1363

Table 4: Mean number of steps a token lasts, grouped by POS-tags.