REVERSE ENGINEERING ATTENTION SINKS IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In autoregressive large language models (LLMs), the beginning-of-sequence (BOS) token tends to receive disproportionate amounts of attention. While past research has theorized about *why* these "attention sinks" form, we instead focus on characterizing *what* attention sinks are and *how* they're implemented. We posit that attention sinks are used to implement linear threshold-based gates to filter attention logits, and from this, a Heisenberg uncertainty principle follows: an attention head can either observe an attention sink, or observe its semantic meaning and position, but not both simultaneously. Inspired by this principle, we find that LLMs trained to accept sequences without a BOS token, which have been shown to form an attention sink at their first token, use Layer 0 to remove semantic and positional information from the first token, turning it into a replacement BOS so that it may function as the attention sink. We show how malicious prompts can fool these LLMs into acting as if each of multiple tokens are the unique attention sink. Finally, motivated by our findings, we propose minimal architectural modifications to alleviate these issues, resulting in LLMs that are comparable-or-better in performance, no longer form first-token sinks, and are more robust to repeated-token attacks.

1 WHAT IS AN ATTENTION SINK?

Attention sinks are a phenomenon in autoregressive large language models in which the beginning-of-sequence (BOS) token, despite carrying no information about the input sequence, often receives about as much attention weight as all other tokens combined. Prior work on attention sinks in LLMs largely focuses on heuristically explaining *why* they're necessary. The prevailing explanation is that each token in an attention head must distribute all of its attention mass somewhere, so allocating it to a semantically void token can serve as a no-op when the head finds nothing useful to retrieve (Bondarenko et al., 2023; Xiao et al., 2024; Barbero et al., 2025a). In this paper, we trace through the computations that enable LLMs to form and utilize attention sinks. We emphasize understanding *what* attention sinks are and *how* they're implemented, in order to inform us about how to train more interpretable and more reliable LLMs. As such, we place our emphasis on building heuristic mathematical intuitions for how attention sinks behave over developing rigorous proofs thereof.

1.1 ATTENTION SINKS AS NO-OPS

As evidence for the no-op explanation, the value vector for the attention sink token is significantly lower in magnitude than other value vectors (Bondarenko et al., 2023; Guo et al., 2024; Gu et al., 2025). In particular, as shown in Figure 1, heads with stronger attention sinks have smaller value vectors for the BOS token than for other tokens. In fact, training an LLM with attention head value vectors for the BOS token zeroed out has been shown to slightly improve performance (Xiao et al., 2024; Gu et al., 2025).

1.2 ATTENTION SINKS AS THRESHOLDS

We further this no-op account by positing that attention sink logits in fact function as thresholds used to filter attention logits. As initial motivation for this interpretation, consider an idealized version of softmax, namely argmax. Under argmax attention with attention logits $(s_0, s_1, \ldots, s_n)$, token 0 would be abnormally likely to receive attention when s_0 is larger than typical values of $s_1, \ldots, s_n$, with the implication that it's a token's s_i being greater than s_0 is not its typical event. Then, s_0 has a simple interpretation as a threshold: a ordinary token can only be attended to if its logit beats the baseline of s_0.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

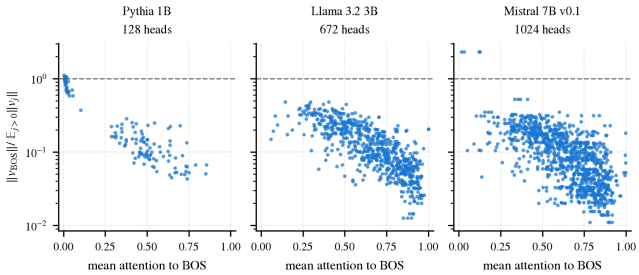

Figure 1: The ratio between the magnitude of the BOS token's value vector and those of value vectors for other tokens, scatterplotted, by attention head, against the mean attention weight that the BOS recieves in each head.

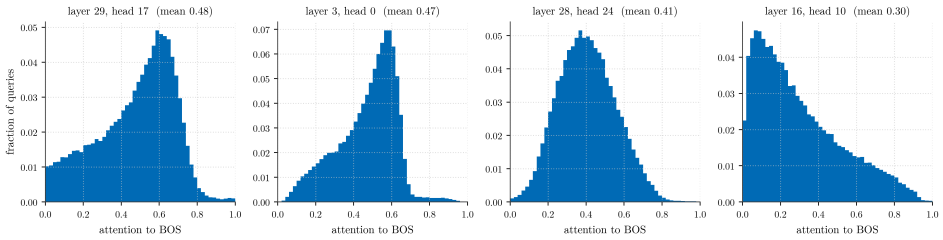

Figure 2: A histogram of how much attention weight the BOS receives for each of four randomly selected heads in Mistral-7B-v0.1.

Moving onto softmax attention, consider attention logits $(s_0, s_1, \ldots, s_n)$ with attention weights $(\alpha_0, \alpha_1, \ldots, \alpha_n) = \text{softmax}(s_0, s_1, \ldots, s_n)$. Observe that, using $\|(\alpha_1, \ldots, \alpha_n)\|_2$ as a proxy for the magnitude of the output of the attention head,

$$\begin{aligned}\|(\alpha_1, \ldots, \alpha_n)\|_2 &= \left\| \frac{\sum_{i=1}^n e^{s_i}}{\sum_{i=0}^n e^{s_i}} \cdot \text{softmax}(s_1, \ldots, s_n) \right\|_2 \\ &= \left(1 - \frac{1}{1 + \sum_{i=1}^n e^{s_i - s_0}}\right) \|\text{softmax}(s_1, \ldots, s_n)\|_2.\end{aligned}$$

Both of the factors in this product are upper bounded by 1, so for the magnitude of their product to be significant, both factors must simultaneously take on significant values. The former factor approaches its maximum value when $s_0 < \text{logsumexp}(s_1, \ldots, s_n)$; the latter when $\text{softmax}(s_1, \ldots, s_n)$ is sparse. But $\text{softmax}(s_1, \ldots, s_n)$ being sparse is precisely when $\text{logsumexp}(s_1, \ldots, s_n)$ is closest to $\max(s_1, \ldots, s_n)$ in value. In other words, the attention head's output will tend to be most significant precisely when there exists a logit that beats or is roughly on par with the s_0 threshold. This serves as further motivation for the view of attention sink logits as functioning like thresholds on other logits.

1.3 Empirical Evidence for the Threshold Interpretation

In alignment with this furthered understanding of attention sinks, the attention weight given to the attention sink is quite inconsistent for each token, even when narrowing focus onto a specific attention head. Examples are shown in Figure 2.

1.4 Conclusion

The main takeaway of this section is that attention sinks are not just any kind of no-op: **attention sinks implement thresholds used to filter out other attention logits**. Because of this reason, in the rest of this paper, we will use the terms *attention threshold* and *attention sink logit* interchangeably.

2 How Are Thresholds Implemented in Attention Heads?

What determines the attention threshold? Considering a RoPE-based (Su et al., 2024) attention head with head dimension d, weight matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v$, and frequencies $\boldsymbol{\theta}$, where

$$\boldsymbol{W}_q = \begin{pmatrix} \boldsymbol{w}_{(1)}^q \\ \boldsymbol{w}_{(2)}^q \\ \vdots \\ \boldsymbol{w}_{(d/2)}^q \end{pmatrix} \in \mathbb{C}^{(d/2)\times D}, \qquad \boldsymbol{W}_k = \begin{pmatrix} \boldsymbol{w}_k^{(1)} \\ \boldsymbol{w}_k^{(2)} \\ \vdots \\ \boldsymbol{w}_k^{(d/2)} \end{pmatrix} \in \mathbb{C}^{(d/2)\times D},$$

$$\boldsymbol{W}_v = \begin{pmatrix} \boldsymbol{w}_v^{(1)} \\ \boldsymbol{w}_v^{(2)} \\ \vdots \\ \boldsymbol{w}_v^{(d/2)} \end{pmatrix} \in \mathbb{C}^{(d/2)\times D}, \qquad \boldsymbol{\theta} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{d/2} \end{pmatrix} \in \mathbb{R}^{d/2},$$

the attention logit from token m to token $n \le m$ is given by

$$\begin{aligned} \mathrm{Re}\left[\frac{1}{\sqrt{d}}\left\langle e^{m\cdot\mathrm{diag}(i\boldsymbol{\theta})}\boldsymbol{W}_q\boldsymbol{x}_m, e^{n\cdot\mathrm{diag}(i\boldsymbol{\theta})}\boldsymbol{W}_k\boldsymbol{x}_n\right\rangle\right] &= \frac{1}{\sqrt{d}}\sum_{j=1}^{d/2}\mathrm{Re}\left[\left\langle e^{mi\theta_j}\boldsymbol{w}_q^{(j)}\boldsymbol{x}_m, e^{ni\theta_j}\boldsymbol{w}_k^{(j)}\boldsymbol{x}_n\right\rangle\right] \\ &= \frac{1}{\sqrt{d}}\sum_{j=1}^{d/2}\mathrm{Re}\left[\left\langle e^{mi\theta_j}q_m^{(j)}, e^{ni\theta_j}k_n^{(j)}\right\rangle\right] \\ &= \frac{1}{\sqrt{d}}\sum_{j=1}^{d/2}\mathrm{Re}\left[q_m^{(j)}\overline{k_n^{(j)}}e^{\theta_j(m-n)i}\right]. \end{aligned} \tag{1}$$

The attention threshold is thus $\frac{1}{\sqrt{d}}\sum_{j=1}^{d/2}\mathrm{Re}\left[q_m^{(j)}\overline{k_0^{(j)}}e^{\theta_j mi}\right]$. For an LLM trained with a BOS, the value of $\boldsymbol{k}_0$ is input-independent. Hence, the attention threshold is determined by just two inputs: the absolute position m of the current token, and its query vector $\boldsymbol{q}_m$.

2.1 Keeping Thresholds Stable via Stashing Attention Sink Tokens in Low Frequency RoPE Dimensions

Treat the attention threshold as a function of m parameterized by $\boldsymbol{q}_m$; call it $t(m;\boldsymbol{q}_m)$. This is a sum of $d/2$ sinusoids in m, where the jth sinusoid has amplitude $\left|q_m^{(j)}\overline{k_0^{(j)}}\right|$ and angular velocity θ_j.

We posit that $t(m;\boldsymbol{q}_m)$ must be near-constant over $t(m;\boldsymbol{q}_m)$ over intervals of size at least $\Omega(N)$, where N is the LLM's context window. Otherwise, the LLM's behavior would have an arbitrary dependence on the input token's absolute position, even though distinct absolute positions $m \in [0.9N, N)$ have quite similar meanings.

Consider a RoPE subspace j for which $2\pi/\theta_j \ll N$, that is, for which the rotation incurred under a context window of length N is substantial. Our assumption then forces that the amplitude of the jth sinusoidal component of $t(m;\boldsymbol{q}_m)$ must be close to zero. If this were not the case, then the subspace would contribute its sinusoid's variance to the variance of $t(m;\boldsymbol{q}_m)$. But this would be problematic because it wouldn't have the ability to compensate for this additional variance by contributing negative covariance to other sinusoids in the sum, because it can be deduced from the product-to-sum rule that its term has near-zero covariance with other sinusoids. Hence, when $2\pi/\theta_j \ll N$, in order to suppress the influence of the jth RoPE subspace on the attention sink logit, we posit that the amplitude $q_m^{(j)}\overline{k_0^{(j)}}$ of the jth sinusoid must be approximately 0.

Either $q_m^{(j)} \approx 0$ or $k_0^{(j)} \approx 0$ would suffice for this. However, by equation 1, the only reason that $q_m^{(j)} \approx 0$ would suffice is because it would suppresses the influence of jth RoPE subspace on *all* attention logits entirely, not just the attention sink logit. This would thereby reducing the effective dimensionality of the attention head's latent query-key space from d to at most $d-2$, and d is already significantly less than the model's embedding dimension D.

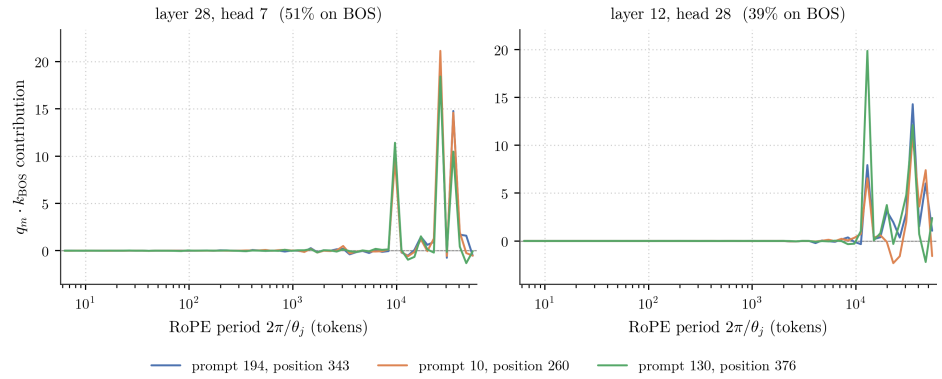

Figure 3: For Mistral-7B-v0.1, we plot the contribution to sink formation of each RoPE period. Empirically, low frequency dimensions appear to have significantly more contribution, matching our proposed theory.

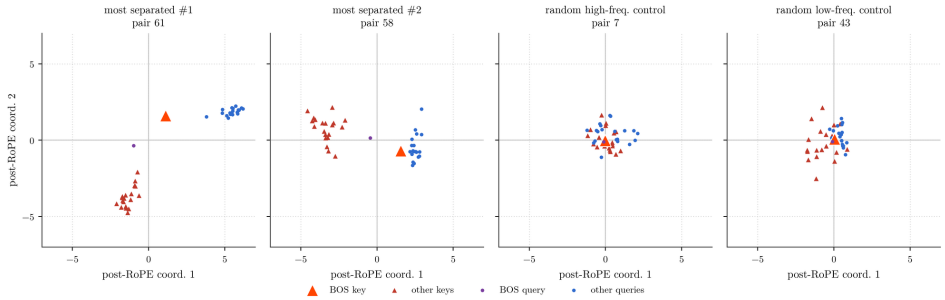

Figure 4: An attention head in Mistral-7B-v0.1 was randomly chosen. Then, two RoPE subspaces j with high separability between the BOS token's key and other tokens' keys were chosen, in addition to two random subspaces j. Scatterplots of $e^{\theta_j mi} q_m^{(j)}$ and $e^{\theta_j ni} k_n^{(j)}$ across different values of m and n are then shown from a randomly chosen input sentence.

As such, the more useful implementation of $q_m^{(j)} \overline{k_0^{(j)}} \approx 0$ is for the model to have $k_0^{(j)} \approx 0$ when j does not correspond to a sufficiently low-frequency RoPE subspace. In other words, $\boldsymbol{k}_0$, the key used to retrieve the attention sink token, can only have nontrivial components in low-frequency RoPE subspaces where $\theta^{(j)} = O(N^{-1})$.

2.2 Another Lens into Stashing

An alternative way to view this conclusion is from the perspective of holding the reference frame of query vectors fixed, and rotating the key vector correspondingly, that is, expressing the logit from token m to token $n \leq m$ as

$$\text{Re}\left[\frac{1}{\sqrt{d}}\left\langle e^{m\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{W}_q\boldsymbol{x}_m, e^{n\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{W}_k\boldsymbol{x}_n\right\rangle\right] = \text{Re}\left[\frac{1}{\sqrt{d}}\left\langle \boldsymbol{q}_m, e^{-(m-n)\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{k}_n\right\rangle\right]. \quad (2)$$

In this view, the attention logit is proportional to the inner product between non-rotating query vectors $\boldsymbol{q}_m$, and rotating key vectors $e^{-m\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{k}_0$, where the amount of rotation is proportional to $m - n$. In this view, the advantage of stashing the attention sink token's key in RoPE subspaces where $\theta^{(j)} = O(N^{-1})$ would be to prevent the key from rotating, as then, the key vector for the sink token remains stable for the duration of the context window.

2.3 Empirical Evidence for Stashing

We observe stashing behavior when plotting the coordinates of query vectors $\boldsymbol{q}_m$ and key vectors $\boldsymbol{k}_n$: Figure 4 demonstrates that the BOS key only has a presence in low-frequency RoPE subspaces. In these dimensions, tokens' query vectors align with the BOS key (except for its query vector, which is meaningless), and tokens' key vectors anti-align with their query vectors.

2.4 CONCLUSION

One main takeaway of this section is the precise way in which LLM weights learn to stop the rotation $e^{-m\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{k}_0$ of the BOS key: **the sink token's key is stashed in low-frequency RoPE subspaces**, such that it doesn't experience significant rotation in different in-distribution context lengths.

Our theory of stashing directly challenges Ruscio et al. (2025)'s claim that because $\boldsymbol{R}_0 = \boldsymbol{I}$, the attention sink token has a privileged computational status in the attention computation that enables it to act as a geometric anchor for other tokens. However, it provides a mechanistic explanation for Wertheimer et al. (2026)'s finding that RoPE causes tokens to lose separability at context lengths longer than the training context length. We thus endorse their proposal, RoPE-ID, for what set of RoPE frequencies to use when training a model that needs to generalize past its in-distribution context window: to stop $e^{-m\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{k}_0$ from rotating, only use angular velocities which are either $\omega(N^{-1})$ or 0.

An important implication of stashing is that by equation 2, the attention logit collapses to just its RoPE-less variant:

$$\text{Re}\left[\frac{1}{\sqrt{d}}\left\langle \boldsymbol{q}_m, e^{-m\cdot\text{diag}(i\boldsymbol{\theta})}\boldsymbol{k}_0\right\rangle\right] \approx \text{Re}\left[\frac{1}{\sqrt{d}}\left\langle \boldsymbol{q}_m, \boldsymbol{k}_0\right\rangle\right] = \frac{1}{\sqrt{d}}\left\langle \boldsymbol{q}_m, \boldsymbol{k}_0\right\rangle_{\mathbb{R}}. \tag{3}$$

Crucially, because this is constrained to be a linear function, parameterized by $\boldsymbol{k}_0$, but solely dependent upon $\boldsymbol{q}_m$, which itself is a linear function of $\boldsymbol{x}_m$, it is invariant to the semantic and positional information of the first token. Hence, a second main takeaway of this section is what we from now on refer to as a **Heisenberg uncertainty principle for attention sinks**: within a single attention head, the attention sink token can either be observed to be an attention sink, or its semantic and positional information can be observed, but not both simultaneously (cf. Barbero et al. (2025b)).

3 HOW IS STASHING IMPLEMENTED IN LLMS WITHOUT A BOS TOKEN?

LLMs with a BOS token form attention sinks, but so do those trained without a BOS token, which have been shown to form attention sinks on their first token (Xiao et al., 2024; Gu et al., 2025; Barbero et al., 2025a). For as many questions Section 2 settles about how LLMs *with* a BOS implement attention sinks, it raises several new questions about how LLMs *without* a BOS can.

To answer these questions, we analyze Qwen3-8B, a specific BOS-free LLM that we use as a case study. The first of these questions is: given the Heisenberg uncertainty principle for attention sinks, as described in Section 2.4, how can an LLM use the first token as an attention sink when it carries not just potentially important positional information but also definitively important semantic information? In particular if the attention threshold is $\frac{1}{\sqrt{d}}\left\langle \boldsymbol{q}_m, \boldsymbol{k}_0\right\rangle_{\mathbb{R}}$, a linear function in $\boldsymbol{q}_m$ parameterized by $\boldsymbol{k}_0$, how can this threshold be kept independent of the first token when the first token is not input-independent?

3.1 QWEN3-8B LEARNS A BOS CONCEPT AND OVERWRITES THE FIRST TOKEN WITH IT

It turns out that the first token's embedding *morphs* into being nearly input-independent. Taking the embeddings of 500 different tokens when inputted into the first position, the cosine similarity between the different values of $\boldsymbol{x}_0$ (sampled after layer 17, the midway point through the model) comes out to $\approx 0.999911 \pm 0.000045$. That is, in lieu of having a BOS, Qwen3-8B seems to implicitly learn its own BOS direction, and replaces the first token's direction with this learned direction. But this only raises more questions: If the first token's semantic information is overwritten, how can the model output an input-*dependent* prediction for the second token, and how can subsequent tokens that need to do next-token prediction respond to the semantic information provided by the first token?

3.2 WHERE THE OVERWRITE HAPPENS

The latter of these questions is easier to answer. Figure 5 shows that the embedding of the first token doesn't become input-independent until after the MLP in layer 6. This way, attention heads up to those in layer 6 can mix in the informational content of the first token, before the first token's input embedding becomes invariant, allowing it to utilize the semantic information of the first token in the first seven attention layers.

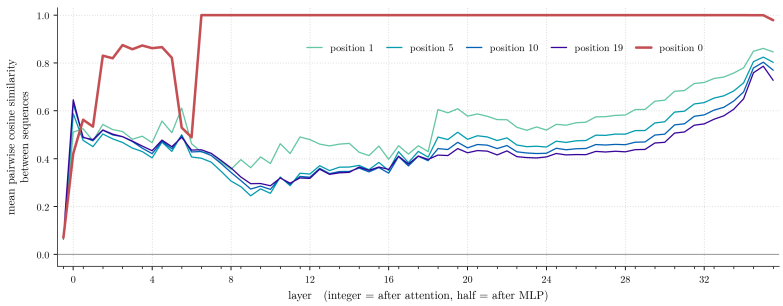

Figure 5: Mean cosine similarity in Qwen3 8B between sampled values of the embedding $\boldsymbol{x}_i$ of the token with index i as a function of the layer, shown for various i.

Table 1: Qwen3-8B Layer 6 Neuron Typical Activations

Neuron	Activation on token 0	Activation on other tokens
#422	≈ 1110	0.86 ± 0.89
#5723	≈ 5700	0.36 ± 0.38
#8518	≈ -1350	-0.65 ± 0.67

Layer 6's MLP accomplishes input-independence for the first token by writing a single direction at very large magnitude ($\approx 10^4$) to the residual stream. We attribute this to three neurons: neurons 422, 5723, and 8518. Typical activations of these neurons for the first token and for other tokens are shown in Table 1; they achieve massive activations similar to those found by Sun et al. (2024).

The MLP output weight vectors for each of these neurons are of magnitude 1.45-1.5, and the cosine similarities between them are all > 0.995. Combined, they consistently collectively write their shared direction at a magnitude of $\approx 10^4$ to the residual stream when they're active. In Qwen's pre-norm architecture, this drowns out not just all previous writes, but also future writes to the residual stream for the first token, thereby keeping the first token's representation stable for all future attention heads.

3.3 Unpacking the First Token's Embedding

We now return to the question of how the model can output an input-independent prediction for the second token. As we can see from Figure 5, we can trace where this happens to the last transformer layer's MLP, which takes in a nearly input-independent representation for the first token, yet puzzlingly outputs an input-dependent one.

Unlike in Section 3.2, we estimate that consistently explaining at least 90% of the write performed by last layer's MLP on the first token requires least 170 of its neurons. However, the total magnitude of vector written by these 170 neurons on the sink token is typically around ≈ 2500, compared to just ≈ 7 on non-sink tokens. That is, the model has $\gtrapprox 170$ neurons devoted specifically to the task of unpacking the tiny variations in the first token's representation. Why would a model have so many neurons devoted to just the first token? We believe that this comes from a combination of two reasons. Firstly, the activation sensitivity required to perform this task is why these neurons need to be specialized specialized, because they'd have to gate out non-sink tokens aggressively in order to avoid dominating the output for non-sink tokens. Secondly, many of these neurons are required to perform the task well, because if only d neurons are assigned to the task, a projection of their activations must reside in some d-dimensional subspace, limiting the expressiveness of the output for the first token.

3.4 The Fallibility of This System

This system for forming attention sinks relies on distinguishing tokens which have undergone mixing from tokens which only self-attend. However, it does not reliably do so. There are around 90-110 tokens in Qwen 3 8B's vocabulary which, when placed at the front of a sequence, are not detected by the neurons in Layer 6's MLP which create attention sinks. Attention maps for two example prompts starting with these tokens are shown in Figure 6.

Another issue with this system is that whether a token has only mixed with itself is an imperfect proxy for whether it's the first token: if we were to use a prompt that begins with a repeated token

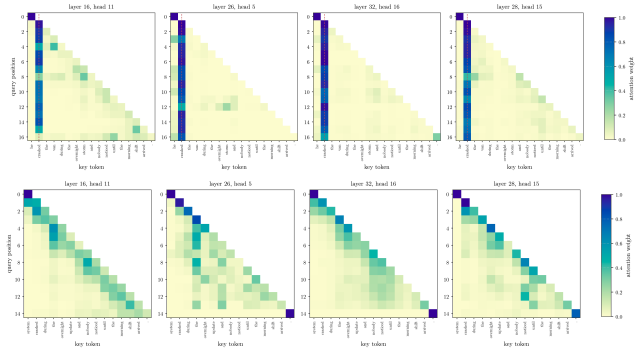

Figure 6: We select four random heads in Qwen 3 8B, and show attention maps for “he” and “system” (both without leading spaces). Tokens like “he” often result in attention sinks forming on the second token, while others, like “system”, singlehandedly prevent the model from forming attention sinks, severely degrading model performance (see Appendix A.2).

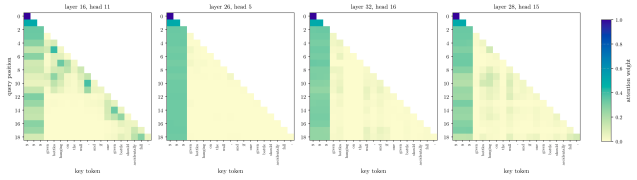

Figure 7: Attention maps for the same heads in 6 for a prompt beginning with “999”, showing the effect of repeated tokens on creating multiple attention sinks.

several times, all of the repeated tokens would become attention sinks, artificially inflating attention thresholds. An example such attention map is shown in Figure 7. This is similar to Yona et al. (2025)’s repeated token attack, except that due to the LLM’s lack of a BOS, decoy attention sinks form within a single repetition.

3.5 Conclusion

As a BOS-free LLM, the steps that Qwen3-8B go through to stash the attention sink are:

1. Separate the first token’s embedding from the rest of the tokens’ embeddings by the time Layer 6’s MLP is reached, all while **transferring** semantic and positional information from the first token into other tokens in order to prepare for its incoming overwrite,
2. At layer 6, use 3 neurons reading and writing roughly the same direction, effectively a **learned BOS direction**, to inject a high magnitude write into the residual stream, **removing** the identity of token 0, so that it may be used as an attention sink.
3. At the last layer, **recover the overwritten information** from the sink token using a large collection of neurons **specifically devoted** to just the first token.

Step 1 is especially brittle; it does not generalize consistently to out-of-distribution first tokens, and it’s vulnerable to repeating the first token.

4 Fixing BOS-free LLMs

4.1 Our Proposal: Learned Linear Thresholds

In Section 1, we discussed how the attention sink logit is an extra term inserted into softmax computations, functioning as a threshold used to filter out other tokens’ logits. In Section 2, we discussed how attention heads are forced to use low-frequency RoPE dimensions to simulate thresholds linear in query vectors, imposing the limitation that the semantic and positional information of a token cannot be used if it’s used instead as an attention sink, which we’ve described as being a Heisenberg uncertainty principle for attention sinks. And in Section 3, we discussed how LLMs’ weights, in the face of this Heisenberg uncertainty principle, coordinate across multiple different layers to flag the first token, remove its semantic and positional information, and finally bring back its semantic information.

In order to simplify this convoluted implementation of linear thresholds, we propose adding a threshold logit t with no corresponding value vector into each attention softmax computation, computing attention weights as

$$(\alpha_0, \ldots, \alpha_n) = \frac{(e^{\alpha_0}, \ldots, e^{\alpha_n})}{e^t + \sum_{i=0}^{n} e^{\alpha_i}}, \tag{4}$$

where t denotes the threshold logit. Based on equation equation 3, we propose that t be an affine function of the *unrotated* query vector. That is, we propose learning threshold keys $\boldsymbol{k}_{\text{sink}} \in \mathbb{C}^{d/2}$ and threshold logit biases $b_{\text{sink}} \in \mathbb{R}$, and using threshold logits $t = b_{\text{sink}} + \frac{1}{\sqrt{d}} \langle \boldsymbol{q}_m, \boldsymbol{k}_{\text{sink}} \rangle_{\mathbb{R}}$ in attention computations via equation 4. This is most similar to Gu et al. (2025)'s technique of learning a key vector for the BOS; the differences are the choice to use a bias and the adaptation to RoPE-based attention heads.

We from now on refer to this architectural modification as QCT-Softmax (Query-Conditioned Threshold Softmax). We hypothesize that QCT-Softmax can be used to train BOS-free models which (1) perform better than models lacking this architectural modification, (2) do not form attention sinks on their first token, and (3) are robust to repeated first-token attacks.

4.2 Experimental Setup

We test our proposed architecture by training three models using the exact setup of Gu et al. (2025) from their released code, a 10-layer LLaMA-style model (around 59M non-embedding parameters) trained on a 20B tokens of the Pile with an matched weight initialization and data order across all compared methods (Gao et al., 2020). Full architecture and optimization details are in Appendix A.2.

Baseline: As a baseline, we use a standard softmax model without a BOS.

Linear Thresholds: We train a model QCT-Softmax that is otherwise identical. Each attention head, in addition to learning weight matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{C}^{(d/2)\times D}$, learns a threshold key $\boldsymbol{k}_{\text{sink}} \in \mathbb{C}^{d/2}$ and a threshold logit bias $b_{\text{sink}} \in \mathbb{R}$, and the softmax computation is modified as described in Section 4.1.

Scalar Thresholds (i.e., gpt-oss-style attention sinks): We also test learning a scalar, query-independent threshold $b_{\text{sink}} \in \mathbb{R}$ without a learned key $\boldsymbol{k}_{\text{sink}} \in \mathbb{C}^{d/2}$; this is mathematically equivalent to gpt-oss's implementation of explicit attention sinks (OpenAI, 2025a;b)

4.3 Experimental Results

We compared the learned models head-to-head on three metrics, each mirroring a hypothesis formulated in Section 4.1. Mirroring hypothesis 4.1, we evaluated model performance on a held-out validation dataset. Results are shown in Figure 8. Mirroring hypothesis 4.1, we plotted the average attention weight of the first token on sequences of length $N \in \{64, 128, 256\}$. Results are shown in Figure 9. Mirroring hypothesis 4.1, we estimated the KL divergence of each model from its normal predictions when the first token is repeated n extra times. Results are shown in Figure 10.

4.4 Conclusion

As shown in Figure 8, QCT-Softmax resulted in comparable-or-better performance to the alternatives we tested. As shown in Figure 9, QCT-Softmax is the only modification that fully **removes the attention sink** on the first token. And as shown in Figure 10, QCT-Softmax is significantly more **robust** than the alternatives we tested to repeated first-token attacks; we believe that due to removing sinks, QCT-Softmax also avoids failure modes of first token sinks. In addition to these benefits, it is **parameter-efficient**, only requiring d extra parameters per attention head, and **computationally efficient**, only requiring an extra logit in each softmax, when compared to alternative effective attention sink removal techniques such as gated attention (Qiu et al., 2025).

5 Conclusion and Discussion

We hope that our account contributes to an increased understanding of what attention sinks are and what various unusual computations LLMs can perform to implement them. We furthermore hope that our proposal for a method to cleanly remove attention sinks is adopted, not just for the sake of robustness and efficiency, but also for the sake of improving the interpretability of trained LLMs.

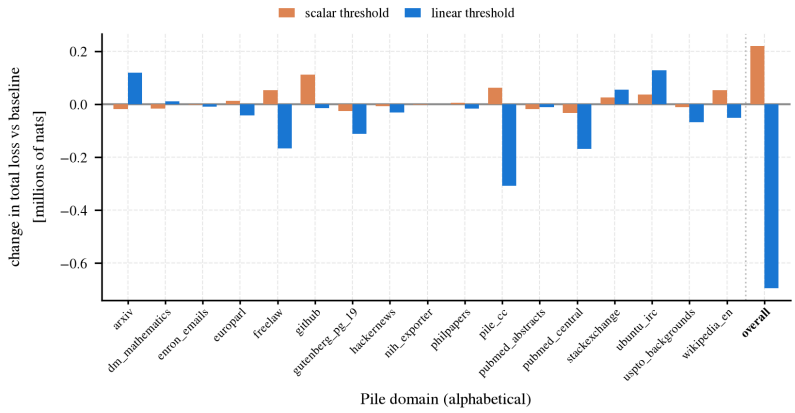

Figure 8: Change in combined Pile loss on its full validation and test splits of the explicitly threshold-based models, relative to the threshold-less baseline. No hyperparameters were tuned, so the test split is used only for evaluation

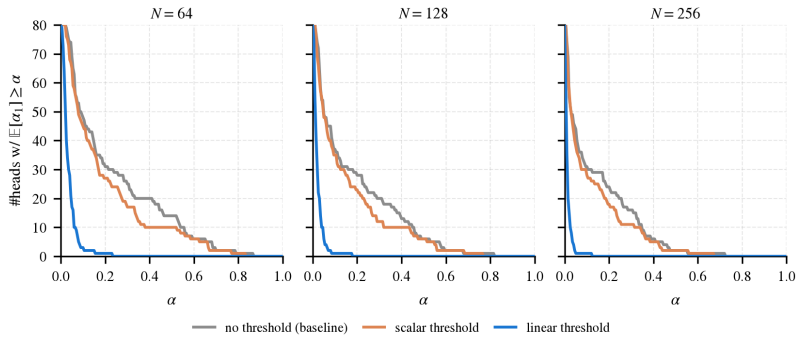

Figure 9: Number of heads whose average attention weight on the first token exceeds α, for $\alpha \in [0, 1]$.

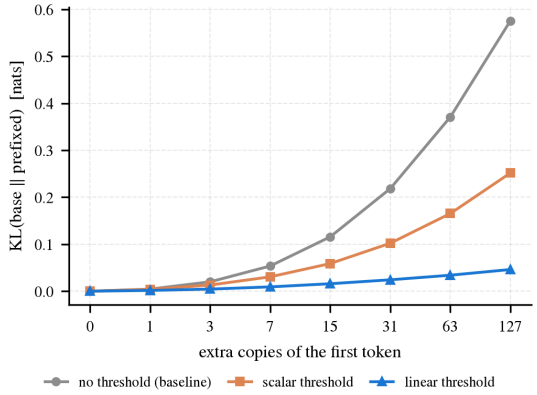

Figure 10: KL divergence from normal predictions vs. number of extra repeats of the first token.

AI USE STATEMENT

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

All code written for the purpose of this project was generated by AI, including the code to produce all figures.

We have not used AI assistance to ideate, develop, refine, or explain our theoretical models, mathematical claims, heuristic arguments, and hypotheses. AI assistance was used in experimental design, though almost exclusively for the purpose of automating microscopic design decisions. The application of AI towards generating synthetic data and towards writing rigorous proofs were not relevant for this work.

With the exclusions of the appendices and bibliography, AI was not used to write, draft, or format this paper, nor consulted when writing this paper for any purposes other than proposing this paper's title, simulating reviewer feedback, and aiding in our search through existing literature to identify which of our findings were novel.

REPRODUCIBILITY STATEMENT

If our paper is accepted for ICLR, we will prepare and post a public GitHub repository for reproducing our experiments.

REFERENCES

Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do LLMs attend to the first token? In *Conference on Language Modeling (COLM)*, 2025a. URL https://arxiv.org/abs/2504.02732.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. Round and round we go! what makes rotary positional encodings useful? In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=GtvuNrk58a.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Advances in Neural Information Processing Systems*, volume 36, pp. 75067–75096, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10781. Code: https://github.com/sail-sg/Attention-Sink.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in LLMs. *arXiv preprint arXiv:2410.13835*, 2024.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025a.

OpenAI. gpt-oss: Reference implementation. https://github.com/openai/gpt-oss/blob/main/gpt_oss/triton/attention.py#L193, 2025b. Explicit learned attention sinks in the Triton attention kernel.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025. URL https://arxiv.org/abs/2505.06708.

Valeria Ruscio, Umberto Nanni, and Fabrizio Silvestri. What are you sinking? A geometric approach on attention sink. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2508.02546.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models, 2024. URL https://arxiv.org/abs/2402.17762.

Davis Wertheimer, Aozhong Zhang, Derrick Liu, Penghang Yin, and Naigang Wang. Frayed RoPE and long inputs: A geometric perspective. *arXiv preprint arXiv:2603.18017*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453.

Itay Yona, Ilia Shumailov, Jamie Hayes, Federico Barbero, and Yossi Gandelsman. Interpreting the repeated token phenomenon in large language models, 2025. URL https://arxiv.org/abs/2503.08908.

Table 2: Pretraining setup, shared by all arms and matching the released configurations of Gu et al. (2025).

Architecture	
Layers / hidden size	10 / 768
Heads / head dim	8 / 96 (MHA)
MLP	SwiGLU, 1536
Norm	RMSNorm, pre-norm
Position	RoPE, base 10^4
Context length	2048
Vocabulary	50,432 (GPT-NeoX)
Bias / dropout	none / 0
Parameters	136M (59M non-emb.)

Training	
Steps	20,000
Batch	512×2048 ($\approx$1M tokens)
Optimizer	AdamW, $\beta = (0.9, 0.95)$
Weight decay	0.1
Learning rate	4×10^{-4}, constant†
Warmup	100 steps
Grad. clipping	1.0
Precision	bf16 mixed
Hardware	1$\times$ A100 80GB

† Gu et al. (2025) describe a cosine schedule, but their released configurations set the minimum learning rate equal to the peak, so the schedule is constant after warmup. We match the released configurations.

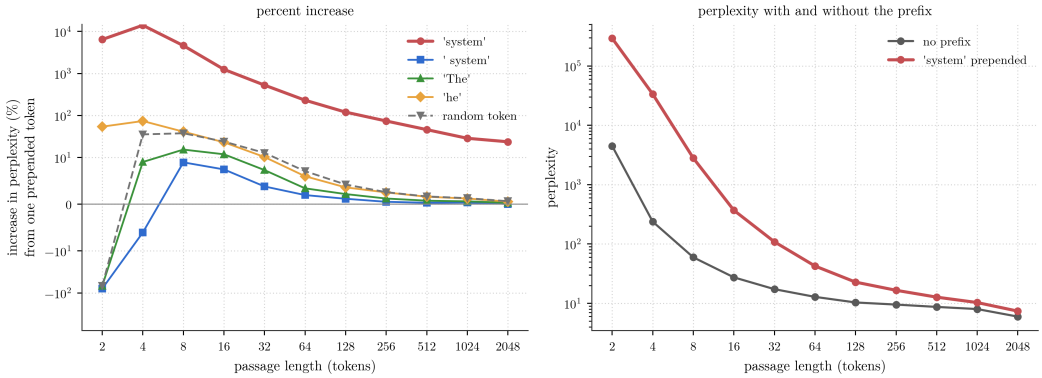

Figure 11: Percent perplexity increase and absolute perpexity on 512 random sequences from the Pile of varying token length, when prepending "system" and other tokens. The random token baseline uses an uniform random draw over the vocabulary for each depicted data point.

A Appendix

A.1 Experimental Setup

All method runs share the pretraining setup in Table 2, which matches the released configurations of Gu et al. (2025). Each run uses the same data seed and sees identical batches in the same order. Runs took $\approx$ 60 hours on a single A100 with 35-40 GB peak memory.

A.2 Perplexity Damage from the Lowercase Token "system"

A token in Qwen3 8B, "system", is able to singlehandedly prevent the formation of sinks in the rest of the sequence as previously shown in Figure 6. Tokens similar to it, like system with a space prep-ended or capitalized "System", form attention sinks. Figure 11 depicts perplexity damage from prepending "system" to normal completions. The damage from the "system" token is significantly higher and catastrophic compared to matched controls. This result is a demonstration the fragility of Qwen's learned implementation of token-0 sinks.

A.3 Quantifying Stashing Across Models

Section 2 makes two predictions about the key of the sink token (the BOS token, or the first token in a BOS-free model). First, its attention logit should stay nearly constant as the query position changes. Second, its key should place most of its magnitude in the lowest-frequency RoPE pairs,

since those barely rotate within the context window. We test both predictions in two models trained with a BOS token, Mistral-7B-v0.1 and Llama-3.2-3B, and in the BOS-free Qwen3-8B.

Score stability. For a given head and key position n, let $\boldsymbol{q}_m$ and $\boldsymbol{k}_n$ denote the query and key after RoPE is applied. We measure how consistent the logit $\boldsymbol{q}_m \cdot \boldsymbol{k}_n$ is across query positions m with

$$S_n = \frac{(\mathbb{E}_m[\boldsymbol{q}_m \cdot \boldsymbol{k}_n])^2}{\mathbb{E}_m\left[(\boldsymbol{q}_m \cdot \boldsymbol{k}_n)^2\right]}, \tag{5}$$

where m ranges from $n+2$ to $L-1$ for a sequence of length L. S_n lies in $[0, 1]$. It equals 1 when the logit is identical for every query position, and is near 0 when the logit oscillates around zero and averages out.

Low-frequency share. RoPE rotates each key in $d/2$ two-dimensional pairs, with pair j rotating at angular frequency θ_j. Let $E_j(\boldsymbol{k})$ be the squared norm of $\boldsymbol{k}$ restricted to pair j, and let $\mathcal{L}$ be the third of pairs with the longest periods $2\pi/\theta_j$ (the top tercile of $\log_{10}$ period). We define

$$f_{\text{low}}(\boldsymbol{k}) = \frac{\sum_{j \in \mathcal{L}} E_j(\boldsymbol{k})}{\sum_j E_j(\boldsymbol{k})}. \tag{6}$$

Because RoPE only rotates vectors within each pair, E_j, and hence f_{low}, is the same before and after RoPE. A key whose magnitude is spread evenly across pairs has $f_{\text{low}} = 1/3$. The models use different RoPE bases, so the low-frequency band covers different periods: above 2,650 tokens for Mistral, above 54,410 for Qwen3, and above 1.1M for Llama-3.2, whose RoPE scaling stretches the lowest frequencies.

Results. All quantities are computed per head and per sequence, then averaged. We use $K = 32$ passages of $L = 2048$ tokens each. For Mistral and Llama we prepend the model's BOS token; Qwen3 has no BOS, so its passages are left unchanged. Table 3 averages over the K passages and over every query head in every layer (32×32 for Mistral, 28×24 for Llama, and 36×32 for Qwen3). The $n > 0$ columns also average over eight key positions sampled log-uniformly from $[1, L-8)$ and held fixed across passages and models: $n \in \{2, 3, 12, 16, 40, 54, 135, 2040\}$.

In all three models, the sink key's logit is more stable across query positions than that of other keys ($S_0 \geq 0.91$, compared with 0.48–0.90), and 84–99% of its magnitude lies in the lowest-frequency third of RoPE pairs, compared with 41–66% for other keys. The pattern holds for Qwen3-8B, whose first token is an ordinary input token, consistent with Section 3's finding that the model converts it into a replacement BOS.

We average over all heads rather than only strong-sink heads because stashing concerns whether the threshold varies with position, not how much attention it absorbs; the argument in Section 2 applies to any head whose threshold is not negligible.

Table 3: Score stability S and low-frequency share f_{low} for the sink key ($n = 0$) and other keys ($n > 0$), at $L = 2048$. Higher values mean a more position-stable logit and more magnitude in slowly rotating RoPE pairs.

	S_0	$S_{n>0}$	$f_{\text{low}}(\boldsymbol{k}_0)$	$f_{\text{low}}(\boldsymbol{k}_{n>0})$
Mistral-7B-v0.1	0.98	0.87	0.99	0.66
Llama-3.2-3B	0.99	0.90	0.89	0.41
Qwen3-8B	0.92	0.48	0.84	0.49

Limits of the stability measure. S_n can only detect position-driven variation from pairs that rotate appreciably within the measured window. At $L = 2048$, pairs at the fast edge of the low-frequency band complete up to 77% of a cycle in Mistral, 4% in Qwen3, and 0.2% in Llama-3.2, whose RoPE scaling places this band beyond 10^6 tokens. For Mistral, S_0 therefore partly tests positional stability, and the sink logit remains stable ($S_0 = 0.98$). For Qwen3 and especially Llama, S_0 mainly reflects whether queries project consistently onto the sink key, since its low-frequency components cannot

rotate within any context we can evaluate. This is expected: these pairs are, by design, the ones that do not rotate within realistic contexts, which is precisely why storing the sink key there yields a position-independent logit. The low-frequency share f_{low} does not depend on L, since the energy in each pair is unchanged by rotation, and we regard it as the primary evidence for stashing.