Forget, Forget Me Not - Adaptive Memory for Large Language Models (LLMs)

Caroline Cunningham
Massachusetts Institute of Technology
ccunning@mit.edu

Clarise Han
Massachusetts Institute of Technology
clariseh@mit.edu

Yogesh Koirala
Massachusetts Institute of Technology
yogesh11@mit.edu

Aileen Liao
Massachusetts Institute of Technology
ai0liao@mit.edu

Daniel Prakah-Asante
Massachusetts Institute of Technology
doprakah@mit.edu

Abstract

Large Language Models (LLMs) face high computational costs due to processing extensive token sequences. To address this, we propose a Adaptive Token Retention (ATR) Model that selectively discards a percentage of input tokens while preserving task accuracy. Our approach aim to optimizes computational efficiency by reducing time complexity without compromising performance. When applied to the WikiText dataset, retaining 90% of tokens achieves comparable accuracy to full retention. By leveraging reinforcement learning-based masking strategies, our model enhances token selection, achieving lower perplexity and competitive ROUGE scores. ATR highlights the potential of token-level optimization to improve efficiency in LLMs.

1 Introduction

The advancements in Large Language Models (LLMs) have brought remarkable capabilities in natural language processing. However, these achievements come with substantial computational demand, due to the quadratic complexity $O(n^2)$ involved in processing token sequences within the transformer architecture [11]. This complexity limits LLMs' efficiency and scalability.

In this project, we propose an adaptive filtering approach to tackle this issue. At the first layer of the transformer, we aim to learn which tokens to selectively "forget", reducing the sequence length before the transformer processes the input. We propose evaluating the effects of our adaptive filtering approach to determine whether the model learns to forget superfluous information and if this forgotten information allows the model to focus more effectively on the task. Analyzing which tokens are forgotten in these prompts should provide an assessment of the distilled model's potential capabilities.

2 Background

Tay et al. [5] introduced Sparse Sinkhorn Attention, a mechanism that reduces the computational burden of transformer-based models by dynamically identifying and focusing on the most relevant tokens. This approach employs a differentiable sorting operation to create a sparse attention mask,

Preprint. Under review.

effectively concentrating computational resources on key tokens while ignoring less significant ones. The result is improved performance and efficiency, particularly in tasks where the input sequence contains a mix of essential and redundant information. This work highlights the potential of token-level optimization for scaling large models without compromising accuracy.

Chen et al. [4] explored active forgetting as a means to enhance accuracy in languages with limited training data. By strategically removing less relevant tokens, their approach improved model performance in underrepresented datasets, demonstrating the utility of selective forgetting for fine-tuning models in specialized contexts. These findings emphasize the broader applicability of token filtering strategies for enhancing both accuracy and efficiency in language models.

However, the use of forgetting as a deliberate strategy to reduce computational resources while simultaneously maintaining or enhancing accuracy remains underexplored. Building on these concepts, our approach introduces an augmented LLAMA model with layers specifically designed to filter out less significant tokens. By reducing token retention, our method lowers the computational cost while ensuring that the model retains its generalized capabilities. This demonstrates the potential for integrating efficient forgetting mechanisms into large-scale language models.

3 Code

The code and demo for this project is publicly available under the MIT License. You can access it at the following link: ATR_TinyML on GitHub.

4 Approach

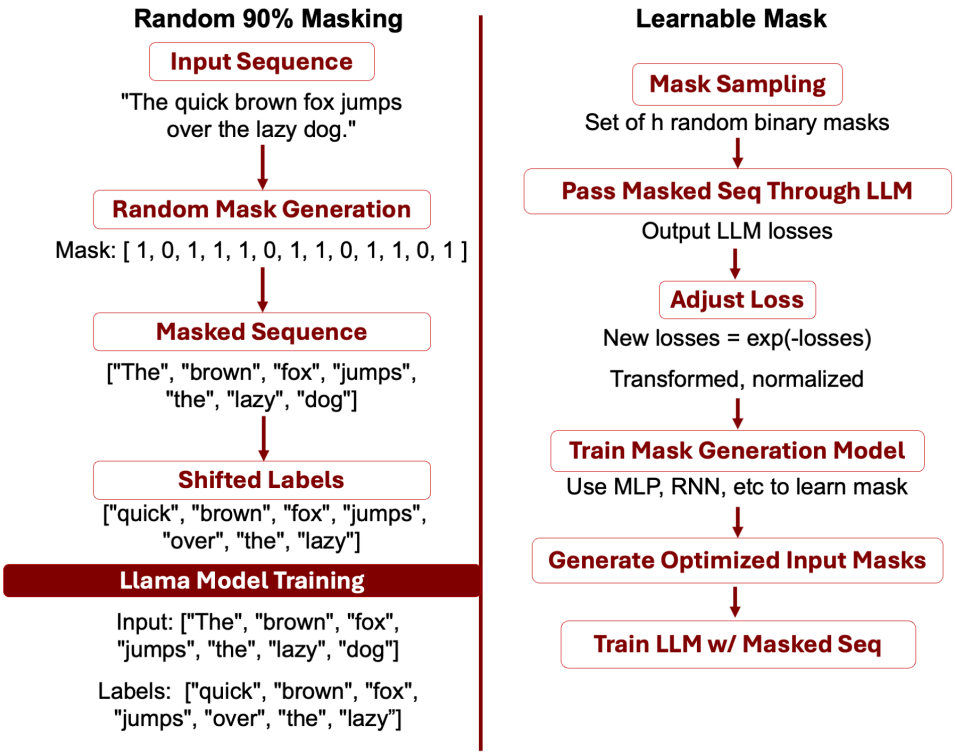

Figure 1: Diagram of Random vs Learnable Masking

4.1 Token Forgetting Mechanism

Before passing in token in to a transformer we introduce a "forgetting" layer that selectively filters out less significant tokens to reduce sequence length. This forgetting layer outputs a vector of binary variables that indicate whether a token should be retained or removed. Given an input sequence of length N with token embeddings represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, the forgetting layer produces a binary vector $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ where $f_i \in \{0, 1\}$. A value of $f_i = 1$ indicates that token i is retained, while $f_i = 0$ means it is forgotten. The downsampled sequence, $\mathbf{X}^*$, consists of only the tokens where $f_i = 1$. This approach reduces the effective sequence length to B, where $B < N$, leading to a computational complexity of $O(B^2)$ in each layer instead of $O(N^2)$.

We explore three methods for implementing this forgetting mechanism:

- **Independent Neural Network Filtering**: In this approach, each token embedding $\mathbf{x}_i$ is individually processed through a neural network (NN) to generate a score s_i, representing the token's importance. This score is then passed through a sigmoid activation function σ, which converts it into a retention probability. Finally, a threshold τ is applied to decide whether to retain or forget the token:

$$f_i = \begin{cases} 1, & \text{if } \sigma(s_i) > \tau, \\ 0, & \text{otherwise.} \end{cases}$$

  This method evaluates each token independently, focusing solely on individual token importance without considering relationships with neighboring tokens.

- **Contextual Convolutional Filtering**: Here, a convolutional neural network (CNN)[7] is first applied across the sequence to capture contextual dependencies among neighboring tokens. Let $\mathbf{y}_i$ denote the CNN output for token i, which encodes information from adjacent tokens within a local window. This output is then passed through a neural network layer and a sigmoid function to determine f_i:

$$f_i = \begin{cases} 1, & \text{if } \sigma(\text{NN}(\mathbf{y}_i)) > \tau, \\ 0, & \text{otherwise.} \end{cases}$$

  This approach enables context-aware token selection by accounting for local dependencies within the sequence, improving the decision-making process on token retention.

- **Sequential Dependency Filtering**: In this method, we use a small sequential model, such as a recurrent neural network (RNN) [6], to process the entire sequence $\mathbf{X}$ and generate binary retention decisions for each token. For each token i, the hidden state $\mathbf{h}_i$ from the sequential model encodes information from the surrounding sequence. The retention decision f_i is then determined by:

$$f_i = \begin{cases} 1, & \text{if } \sigma(\text{NN}(\mathbf{h}_i)) > \tau, \\ 0, & \text{otherwise.} \end{cases}$$

  This approach captures long-range dependencies across tokens, making it suitable for identifying complex relationships and patterns within the sequence to decide which tokens to retain.

Each method downsamples the sequence length from N to B , creating a compressed representation without sacrificing essential information.

4.2 Model Training

The process of removing tokens is not differentiable, We use reinforcement learning strategies, similar to those applied in Deep Q-Networks [8] and Proximal Policy Optimization [9], to optimize token retention decisions. The training process consists of two phases: an **exploration phase** and a **fine-tuning phase**.

4.2.1 Exploration Phase

In the exploration phase, the weights of the target LLM are frozen, and our objective is to train the forgetting layer to identify tokens that can be discarded with minimal negative impact on the LLM's performance. The process is as follows:

1. **Sampling Masks**: We generate random masks, discarding $X\%$ of the tokens. Each mask represents a subset of the input tokens to be passed through the frozen LLM.
2. **Loss Calculation**: The retained tokens are processed by the frozen LLM, and the loss is calculated using categorical cross-entropy. For a given mask M, the loss is denoted as $\mathcal{L}(M)$.
3. **Loss Weighting**:
$$w = e^{-\mathcal{L}(M)}$$
Here, w represents the weight assigned to the mask. Masks with higher losses are assigned smaller weights, whereas masks with lower losses receive larger weights.
4. **Normalization**: The weights w are normalized across all sampled masks to ensure comparability.
5. **Training the Forgetting Model**: The forgetting model generates its own predicted masks. For the same input tokens, we compute the Binary Cross-Entropy (BCE) loss between the predicted masks and the sampled masks, weighted by the normalized w values. The loss is calculated as follows:
$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} w_i \Big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big]$$
where:
   - N is the total number of tokens,
   - $y_i \in \{0, 1\}$ is the ground-truth label (from the sampled mask),
   - $\hat{y}_i \in [0, 1]$ is the predicted probability from the forgetting model,
   - w_i is the normalized weight for mask i.

   The objective encourages the forgetting model to learn a strategy that minimizes performance degradation while aligning with the sampled masks.

4.2.2 Fine-Tuning Phase

In the fine-tuning phase, the forgetting layer is frozen and its learned masks are applied to prune the input tokens. The shortened sequences are then used to further train the LLM. This process enables the LLM to fine-tune its responses based on the reduced input while maintaining accuracy.

4.2.3 Balancing Exploration and Exploitation

To balance exploration and exploitation during the training process, we employ a scheduler. Initially, the model explores with a higher probability of using random masks. Over time, this probability decreases, allowing the model to focus on exploiting the learned token retention strategies.

4.2.4 Label Reformulation

When working with truncated input sequences, it is essential to adjust the labels to ensure they align correctly with the modified input. For example:

- **Original Input**: ["a", "b", "c", "d", "e"]
- **Original Labels**: ["b", "c", "d", "e", "f"]

If a token, such as "c", is forgotten during token pruning:

- **Modified Input**: ["a", "b", "d", "e"]
- **Adjusted Labels**: ["b", "c", "e", "f"]

This adjustment ensures that the output of the model remains consistent with the expected labels, even after certain tokens are removed. By reformulating the labels to correspond to the pruned sequence, we maintain alignment between the input and target outputs, preserving the integrity of the training process.

5 Experiments

To facilitate our experiments and analyze degradation across models, we adjusted the threshold to fix the token retention rate at 90%, meaning that 10% of the input tokens were masked during both training and evaluation. This masking process was guided by the output of the forgetting layer, where the tokens with the 10% highest logits were identified for removal. The remaining 90% of tokens were retained and processed by the model.

5.1 Experimental Setup

Model: All experiments were conducted using the LLaMA-3.2-1B model.

Strategies and Models: We evaluated the following token retention strategies and their respective models:

Category	Method	Description
Baseline	**No Fine-Tuning**	The model was tested without any fine-tuning and with 100% of input tokens retained.
	Fine-Tuned (100% Tokens)	The model was fine-tuned on the WikiText dataset, trained with 100% of input tokens retained.
Random Masking	**Random 90%**	A random masking strategy where 10% of input tokens were randomly masked.
Learnable Masking	**MLP**	A Multi-Layer Perceptron (MLP) was trained to mask 10% of the input tokens.
	GRU	A Gated Recurrent Unit (GRU) model was trained to mask 10% of the input tokens.
	Bi-GRU	A Bidirectional Gated Recurrent Unit (Bi-GRU) model was trained to mask 10% of the input tokens.

Table 1: Methods for Testing and Masking Strategies

5.2 Evaluation

The models were trained on a dataset consisting of 10,000 examples from the WikiText corpus, with training proceeding until convergence to ensure optimal performance. To monitor training progress and prevent overfitting, 20% of the dataset was reserved as a validation set. Convergence was defined as the point when the validation loss no longer improved over the training loss for three consecutive epochs (patience = 3 epochs). This approach ensured that the model was well-tuned to the data without over-optimizing on the training set.

All training and evaluation procedures were conducted using sequences of a fixed length of 512 tokens to maintain consistency and comparability across experiments. For evaluation, a separate set of 100 examples from the WikiText dataset was used. This evaluation dataset provided an independent measure of the models' performance, enabling a fair comparison of the effectiveness of various masking strategies and their impact on predictive accuracy.

6 Results

The results of these experiments will be evaluated using perplexity, ROUGE, and BLEU scores to assess the effectiveness of each masking strategy. Perplexity will provide a measure of how well the model predicts token probabilities, offering insights into its general language modeling performance. ROUGE and BLEU scores, though traditionally used for multi-text evaluation tasks such as summarization or translation, are employed here to assess the semantic similarity of the output compared to the expected text.

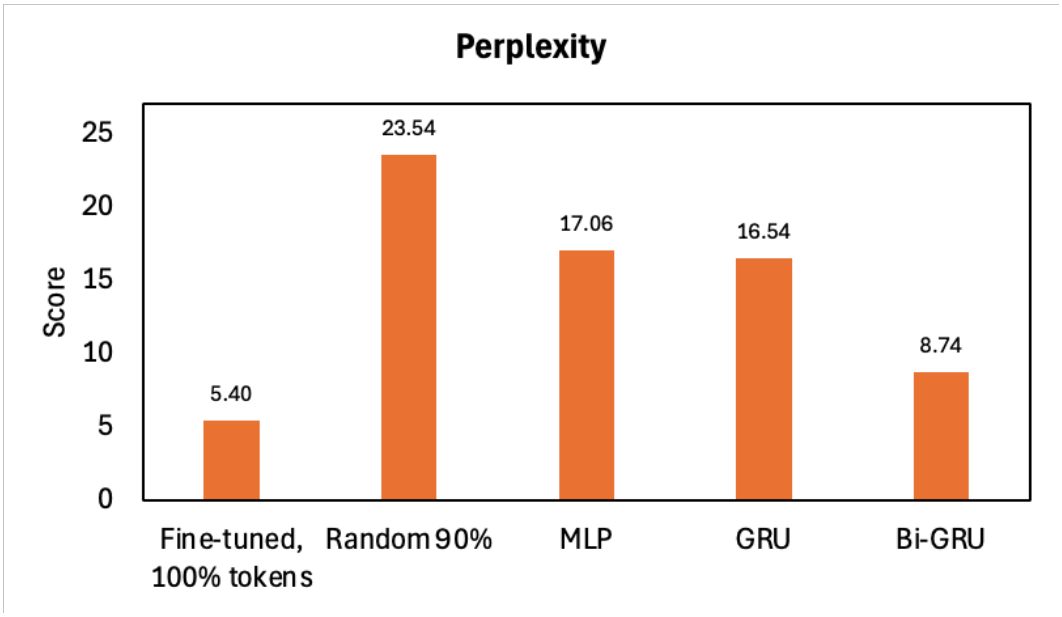

Figure 2: Perplexity scores for model variations

The use of ROUGE and BLEU scores in this context allows us to evaluate whether the reduced input sequences result in outputs that retain the intended meaning and coherence. This analysis helps determine if the masking strategies preserve not just syntactic accuracy but also semantic fidelity. By combining these metrics, we aim to provide a comprehensive evaluation of the trade-offs between computational efficiency and model accuracy across different token retention strategies.

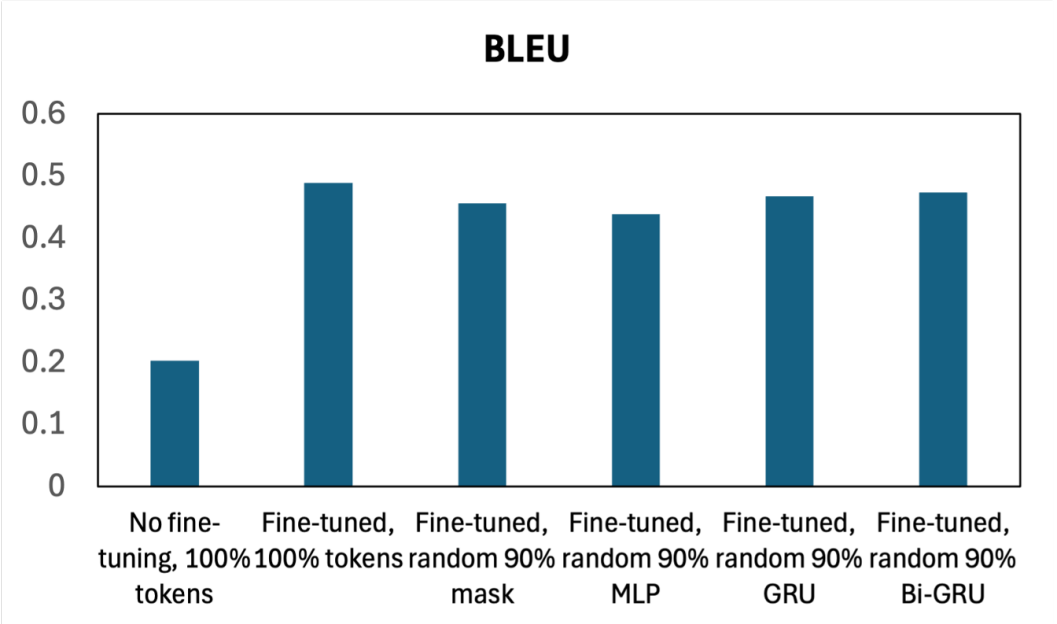

Figure 3: BELU scores for model variations

The results demonstrate that the best fine-tuned model with 100% token retention achieved the lowest perplexity of 5.40, highlighting its superior predictive accuracy. However, when token retention was reduced to 90%, the Bi-GRU-based masking strategy stood out by achieving a perplexity of 8.74, significantly outperforming other reduced-token models. This indicates that the Bi-GRU effectively

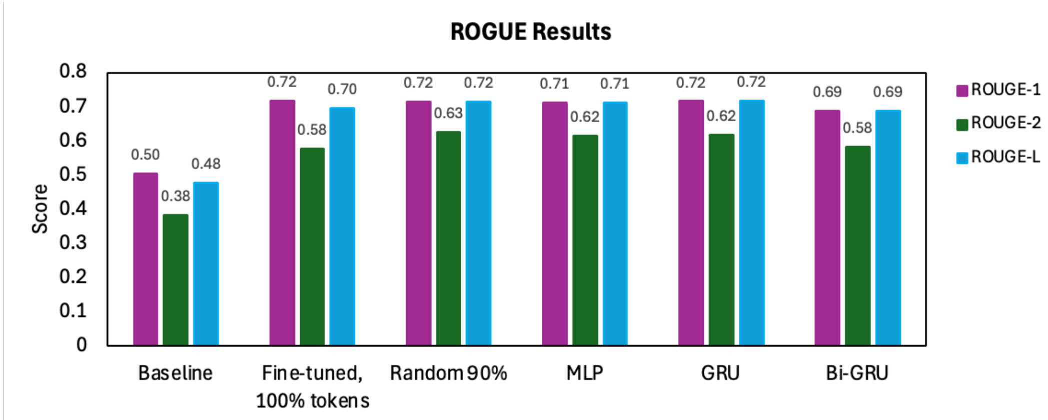

Figure 4: ROGUE1,2,L scores for model variations

learns to prioritize tokens that are crucial for maintaining the model's performance, even with fewer input tokens. This result underscores the potential of learnable masking strategies, particularly bidirectional architectures, to achieve performance comparable to full-token models with reduced input complexity.

7 Token Masking Analysis

7.1 Example Prompt Masking

For the prompt below we can see which tokens the LLM has chosen to forget along with the mask tensor that indicates which tokens are forgotten (0) / retained (1).

Prompt: ' Homarus gammarus , known as the European lobster or common lobster , is a species of clawed lobster from the eastern Atlantic Ocean , Mediterranean Sea and parts of the Black Sea . It is closely related to the American lobster , H. americanus . It may grow to a length of 60 cm (24 in) and a mass of 6 kilograms (13 lb) , and bears a conspicuous pair of claws . In life , the lobsters are blue , only becoming " lobster red " on cooking . Mating occurs in the summer , producing eggs which are carried by the females for up to a year before hatching into planktonic larvae . Homarus gammarus is a highly esteemed food , and is widely caught using lobster pots , mostly around the British Isles . \n'

Forgotten tokens: ['<|begin_of_text|>', ' Hom', 'arus', ' gam', 'mar', 'us', ',', ' known', ' as', ' the', ' European', ' lobster', ' common', ' British', ' Isles', '.', ' \n']

Mask Tensor: mask tensor([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0.]]

7.2 Evalutation

We conducted an initial token masking analysis to understand the patterns in which tokens the Bi-GRU LLM chose to mask. Specifically, using the validation data from the wikitext 103-raw-v1 dataset, we examined 20 prompts across 4 wikipedia topics. We selected the first 5 prompts from each topic, excluding topic headers. The temperature remained constant at 0.1 throughout the evaluation.

We then examined trends in which tokens were masked in the 20 prompts. In all prompts, tokens located at the beginning of the prompt were masked. Additionally, in general, it was observed that the LLM masked a larger number of tokens at the beginning of the prompt as compared to tokens at

the end of the prompt. In 70% (14/20) of the prompts, the LLM chose to also mask tokens near the end of the prompt.

Furthermore, in 2 of the 14 prompts where ending tokens were masked, the only tokens masked were punctuation, special characters, or a single character. Finally, we observed that the LLM only masked tokens in the middle of the prompts for 10% (2/20) of prompts.

Thus, for our prompts, the model generally chose to retain tokens in the middle of the prompts. The model instead masked tokens at the beginning of prompts and oftentimes ending tokens as well. As all of our prompts ranged from 222 - 1183 characters, future work would include testing if the token masking behavior changes for longer prompts. Published research has documented LLM "forgetfulness" on information in the middle of verbose prompts [2].

One limitation of our work is that it is difficult to qualitatively assess whether the masked tokens are "optimal" using the wikitext dataset. This is because within the wikitext dataset we could not easily identify a defined relationship between one sentence to the next. In comparison, using a dataset such as MIT-Bench, where the prompts of the model are user queries, would have likely allowed for a more defined analysis [1].

8 Discussion

This experiment successfully reduced computational complexity by masking 10% of input tokens, while maintaining accuracy levels comparable to models that used all tokens. The use of reinforcement learning for optimizing learned masks proved to be an effective method for identifying which tokens could be discarded without significantly affecting performance. This approach highlights the potential for more efficient resource use in large language models by selectively retaining only the most relevant tokens.

Token masking analysis revealed that the location of the token affected if the model chose to mask it. This suggests a potential avenue for future research: dynamically adjusting the token retention rate based on the structure of the sentence or the requirements of the task. Such an adaptive approach could optimize token retention strategies further, particularly for datasets with varied contexts or outputs.[10].

An additional optimization strategy involves placing a forgetting layer before every transformer block in the network. By progressively masking tokens at each stage of processing, computational costs could be reduced even further, as subsequent layers would process increasingly shorter sequences. This hierarchical token pruning approach could be especially beneficial in very large models, where computation costs scale quadratically with sequence length.

Furthermore, testing on more complex datasets, such as MT-Bench or other benchmarks involving question-answering or summarization, could provide further insights. Reduced token inputs might enhance model focus and improve accuracy in tasks that require precise attention to critical information. This adaptive masking strategy holds promise for improving the scalability of LLMs while maintaining or even enhancing their performance across diverse tasks.

9 Contributions

- **Daniel**: Model design.
- **Aileen**: Training and evaluation scripts.
- **Yogesh**: Model training and background research.
- **Clarise**: Training infrastructure and data preparation.
- **Caroline**: Token masking analysis and background research.

References

[1] Bai, G., Liu, J., Bu, X., He, Y., Liu, J., Zhou, Z., Lin, Z., Su, W., Ge, T., Zheng, B., Ouyang, W. (2024). MT-Bench-101: A Fine-Grained Benchmark for Evaluating Large Language Models in Multi-Turn Dialogues. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 7421–7454. https://doi.org/10.18653/v1/2024.acl-long.401

[2] Liu, Nelson F., et al. “Lost in the Middle: How Language Models Use Long Contexts.” Transactions of the Association for Computational Linguistics, vol. 12, 2024, pp. 157–73. Crossref, https://doi.org/10.1162/tacl_a_00638.

[3] Shelf.io next-generation knowledge management for accurate genai answers. Shelf. (2024). https://shelf.io

[4] Chen, Yihong, et al. "Improving language plasticity via pretraining with active forgetting." Advances in Neural Information Processing Systems 36 (2023): 31543-31557.

[5] Tay, Yi, et al. "Sparse Sinkhorn Attention." arXiv preprint, 2020, https://doi.org/10.48550/arXiv.2002.11296.

[6] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[7] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324. https://doi.org/10.1109/5.726791.

[8] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013). https://doi.org/10.48550/arXiv.1312.5602.

[9] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017). https://doi.org/10.48550/arXiv.1707.06347.

[10] Bengio, Yoshua, et al. "Learning deep architectures for AI." Foundations and trends in Machine Learning 2.1 (2009): 1-127. https://doi.org/10.1561/2200000006.

[11] Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems 30 (2017): 5998-6008. https://doi.org/10.48550/arXiv.1706.03762.