Re-Examine Hybrid State-Space and Self-Attention for In-Context Learning

Anonymous Authors[1]

Abstract

Recent research has shown that combining the state-space algorithm-driven Mamba with the self-attention algorithm-driven Transformer outperforms using Mamba or Transformer alone in most language modeling tasks. However, the performance of the mainstream hybrid modeling architecture models in in-context learning tasks is not ideal. We re-examine the advantages and disadvantages of these two algorithms and redesign the structure of this hybrid modeling from the principle. The finally redesigned architecture improves the performance by 1.1% in standard short text tasks, 20.86% in natural long text tasks, and 24.15% in synthetic long text tasks.

1. Introduction

The self-attention algorithm of the Transformers (Wolf et al., 2020) architecture can directly capture the relationship between any two elements in a sequence, effectively handle long-distance dependencies. However, it is limited by quadratic complexity. The state-space algorithm of the Mamba (Gu & Dao, 2023) architecture can achieve linear scaling of sequence length during training and maintain a constant state size during generation, but it leads to bias in capturing long-distance dependencies. Hybrid modeling architecture models, such as Wonderful Matrices (Shi & Wu, 2024), Jamba (Lieber et al., 2024), etc., use state-space and self-attention for hybrid modeling, making the model have efficiency similar to The Mamba and effect similar to The Transformer. However, these models still have a significant gap in performance in in-context learning tasks compared to the original Transformer.

We propose Self-Attention before LM-Head 1, which is a simple change to the existing hybrid stacked architecture models, modifying the state-space and self-attention to use the same positional encoding, and using a Transformer block composed of self-attention and feed-forward networks before the LM-Head predicts the probability distribution. This method allows the model to continue to leverage the advantages of the efficient context summary of the state-space and the effective associative recall of self-attention without bias in the final token prediction.

Our research and evaluation show that Self-Attention before LM-Head can achieve better performance on the in-context learning task benchmark compared to the baseline hybrid model with only a few structural and parameter adjustments. For example, in standard short text tasks, our model improves performance by 1.1%, in natural long text tasks by 20.86%, in synthetic long text tasks by 24.15%, and achieves state-of-the-art performance on the needle in a haystack task.

2. Background

2.1. State-Space

In the field of natural language processing (NLP), SSM is used to model the dynamic characteristics of text sequences. Its advantage is that it can efficiently handle long sequence data and capture the time dependency in the sequence through the update mechanism of the state variables. However, traditional SSM has some limitations, such as the matrix parameters (such as A, B, C) of the linear time-invariant (LTI) system remain fixed throughout the entire sequence generation process, making it difficult for the model to perform content-aware reasoning. For example, when dealing with tasks that require selective attention or ignoring specific inputs, the performance of traditional SSM is not as good as the Transformer model based on the self-attention mechanism. State-space models are mathematical models used to describe the dynamic behavior of systems, and their basic form is:

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t) \\ y(t) &= Ch(t) + Dx(t) \end{aligned} \tag{1}$$

where $h(t)$ is the state vector, $x(t)$ is the input vector, $y(t)$ is the output vector, and A, B, C, D are system matrices.

$$\begin{aligned} h_t &= Ah_{t-1} + Bx_t \\ y_t &= Ch_t \end{aligned} \tag{2}$$

Starting from the state-space model of continuous-time systems, the discrete-time state update formula can be obtained through discretization methods. Equations (2a) and (2b) represent the discrete form of the state-space model. They describe how the state vector and output vector are updated

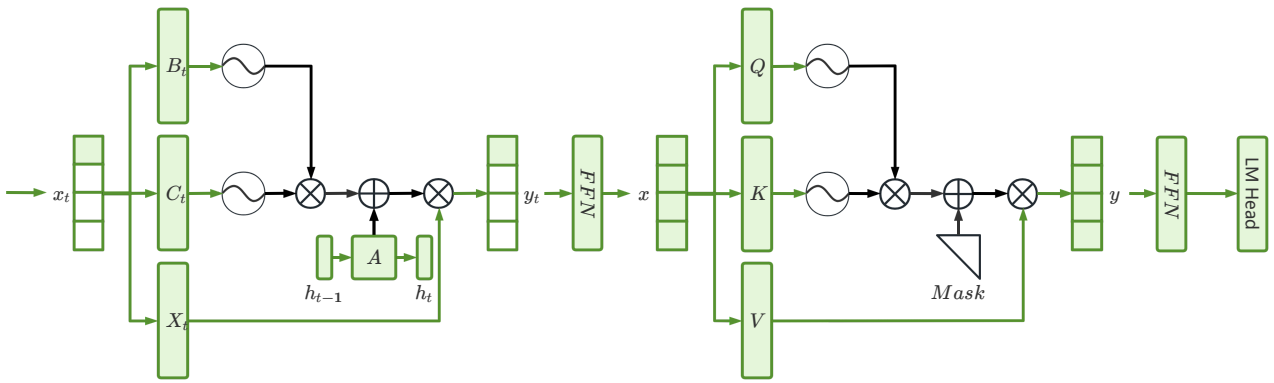

Figure 1. **Self-Attention before LM-Head**. The state-space and self-attention both use the same positional encoding, and a Transformer block composed of self-attention and feed-forward networks is used before the LM-Head, regardless of how the other parts of the model backbone are combined.

based on the previous state and current input at discrete time intervals.

The above is the process of SSM-S4. All structured SSMs are LTI models, and LTI models have basic efficiency limitations. To overcome these limitations, researchers have proposed various improvement methods. For example, the Mamba (Dao & Gu, 2024) model introduces the Selective Scan algorithm and the Hardware-Aware Algorithm, allowing SSM to dynamically adjust parameters to better handle long sequence data and improve computational efficiency. This improved SSM, also known as Selective State Space Model (Selective SSM, referred to as S6), aims to build efficient sequence processing modules like Transformer. However, even so, it is still not as efficient as CNN and Transformer in computation, and the development of modern parallel computing devices makes this computational efficiency gap more obvious, thereby limiting the training of SSM on larger-scale sequence data. Therefore, it is very important to reveal the deeper relationship between Selective SSM and the attention mechanism and propose a new hybrid modeling architecture to overcome these limitations.

2.2. Self Attention

In the Transformer model, the self-attention mechanism determines the importance weights of each position by calculating the similarity between queries, keys, and values. Specifically, the self-attention mechanism first calculates the dot product between the query and the key, then converts these dot products into attention weights through the softmax function, and finally applies these weights to the value vector to generate a weighted context representation.

$$Y = \text{softmax}(QK^\top) \cdot V \tag{3}$$

Through the self-attention mechanism, the Transformer model can directly capture the relationship between any two elements in the sequence, effectively handling long-distance dependency problems. However, the quadratic complexity of the self-attention mechanism limits the computational efficiency of the Transformer model, making it perform poorly when handling long sequence data. To overcome this limitation, researchers have proposed many improvement methods. For example, Lin et al. (Kitaev et al., 2020) proposed a self-attention mechanism based on local sensitive hashing, which reduces the computational cost of dot products between queries and keys, thereby improving the computational efficiency of the Transformer model. Similarly, this improvement method still cannot completely solve the computational efficiency problem of the Transformer model. Therefore, it is very important to reveal the deeper relationship between the self-attention mechanism and the state space and propose a new hybrid modeling architecture to overcome these limitations.

2.3. Hybrid Modeling

Hybrid modeling architecture models are models that combine state-space and self-attention for hybrid modeling, with the goal of making the model have efficiency similar to Mamba and effect similar to Transformer. For example, the Jamba (Lieber et al., 2024) model is a universal language model based on a hybrid modeling architecture, with a basic structure of 8 layers of hybrid cycles, each hybrid cycle consisting of 7 state-space modules, 1 self-attention module, and 8 MLP modules. By combining state-space and self-attention for hybrid modeling, the Jamba model can achieve better performance on most language modeling tasks. Models based on hybrid modeling architecture outperform using Mamba or Transformer alone in most language modeling

tasks, but their performance in in-context learning tasks is not ideal.

3. Method

In the past hybrid modeling architecture models, a one-dimensional convolution in the Mamba (Gu & Dao, 2023) block is used to implicitly perform positional encoding, while the rotational positional encoding used by self-attention is removed. The rotational positional encoding does not directly operate on the attention score matrix. For good extrapolation performance, we can add it to the state space. Let's review the state-space algorithm 2, first, the matrix B controls whether to let the input x_t enter the state h_t, we add rotational positional encoding to the matrix B, the matrix B as an input gate filters x_t after h_t has absolute positional information, then, the matrix A controls whether to let the state h_{t-1} enter the state h_t, that is, it also allows the absolute positional information of the past state to affect the current state, finally, the matrix C controls whether to let the state h_t enter the output y_t, we add rotational positional encoding to the matrix C, the matrix C as an output gate filters h_t after y_t has relative positional information. However, it is worth noting that due to the recursive nature of the state space, that is, only one state cache is retained, the positional encoding will be affected by the superposition state, even the original one-dimensional convolution positional encoding is the same, which is also its difficulty in context learning defect. So using the position information processed by the state space directly as the positional encoding of self-attention will inevitably lead to the confusion of position information. To solve this problem, we should retain the rotational positional encoding of self-attention and provide the original position information for self-attention.

The computational complexity of the state space grows linearly with the sequence length T, and its data-dependent related parameters can enable the model to have the ability to selectively process information. The linear recursion gives the aggregated state that allows us to calculate only the last token. Of course, by approximating the function to generate the optimal solution of the state matrix A to remember history, it can only accurately capture the state change from h_{t-1} to h_t and gradually decay the old state information, which leads to the need for strong replication, context learning, or long-context reasoning capabilities. In tasks that require powerful capabilities, before predicting the token, the information summary of the state space will produce bias. To solve this problem, we should use self-attention and feed-forward networks that do not compress context information before the LM-Head, which can produce context-aware states by weighting elements without distance restrictions, allowing the model to continue to leverage the efficient context summary of the state space and the effective associative recall of self-attention without bias in the final token prediction.

4. Experiments

We used the baseline model Jamba (Lieber et al., 2024) architecture to first construct a 280M small language model and verified our method on semantic similarity, short text classification, natural language reasoning, keyword recognition, and multi-domain multiple-choice tasks. Then, we scaled the model to 1B and verified it on standard short text, natural long text, synthetic long text, and needle in a haystack tasks.

4.1. Datasets and Hyperparameters

The pre-training dataset is a mixed dataset containing open-source datasets such as Book (Ziegler, 2004), Wikipedia (Foundation, 2024), UNCorpus (Nations, 2024), translation2019zh (Brightmart, 2019), WikiMatri (Unknown, 2024), news-commentry (for Machine Translation, 2024a), ParaCrawlv9 (Coordination, 2024), ClueCorpusSmall (Team, 2020), CSL (Li et al., 2022), news-crawl (for Machine Translation, 2024b), etc.

The learning rate scheduler function comes from Attention is All You Need (Wolf et al., 2020).

$$lr = d_{model}^{-0.5} \times \min(steps^{-0.5}, steps \times warmup^{-1.5}) \tag{4}$$

In addition, we did not use linear bias terms, the warm-up steps were 10% of the total steps, $RMSNorm$ was used instead of $LayerNorm$, $AdamW$ hyperparameters were set to $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-6}$, and GLM (Du et al., 2022) was used as the architecture backbone, that is, bidirectional GLM universal language model: MASK across training text sequence spans, with autoregressive filling as the training target, requiring the model to predict the MASKed token. The environment was provided by NVIDIA's open-source PyTorch image (NVIDIA, 2022).

4.2. Self-Attention before LM-Head is Important

As shown in Table 1, we show the parameter settings of Jamba and Jamba-SABLH. In a Jamba hybrid cycle, we modified the last module from a state-space module to a self-attention module, and the rest of the parameters were the same.

As shown in Table 2, the difference in perplexity performance between Jamba and Jamba-SABLH in the pre-training phase is negligible, indicating that the two learned almost the same knowledge from pre-training. However, in the SFT fine-tuning phase, the model usually needs to

Table 1. **Parameters of Jamba and Jamba-SABLH for Verification**. In the original Jamba paper, every 8 layers are a hybrid cycle, including 7 state-space modules, 1 self-attention module, and 8 MLP modules. We only modified the last module to a self-attention module, and the rest of the parameters are the same as Jamba. We constructed two models of about 280M, Jamba and Jamba-SABLH. In the hybrid structure, S represents the state-space module, A represents the self-attention module, and M represents the MLP module.

MODEL	PARAMS	n_{layers}	d_{model}	d_{ffn}	n_{heads}	d_{state}	HYBRID
Jamba	283M	8	1024	4096	16	16	SMSMSMSMAMSMSMSM
Jamba-SABLH	281M	8	1024	4096	16	16	SMSMSMSMAMSMSMAM

Table 2. **Results of Jamba and Jamba-SABLH for Verification**. The performance gap between Jamba and Jamba-SABLH in pre-training perplexity is not large, but in SFT fine-tuning, the perplexity of Jamba-SABLH is significantly lower than Jamba. We selected the evaluation tasks in CLUEbenchmark (Xu et al., 2020) that are more suitable for the current parameter volume: AFQMC: Semantic similarity, judge whether two sentences have the same meaning or not. TNEWS: Short text classification, from the news section of today's headlines, judge the category of news. IFLYTEK: Long text classification, a long text dataset about software descriptions, judge the application topic of the software description. CMNLI: Language reasoning, judge the logical relationship between two sentences. WSC: Pronoun disambiguation, judge which noun the pronoun in the sentence refers to. CSL: Keyword recognition, taken from the abstract and keywords of the paper, judge whether the keyword is in the abstract. In the validation set inference accuracy of AFQMC, TNEWS, IFLYTEK, WSC, CSL, Jamba-SABLH maintains the lead, slightly behind in CMNLI, and the average accuracy of Jamba-SABLH is 4.25% higher than Jamba.

MODEL	PT PPL ↓	SFT PPL ↓	AFQMC ACC ↑	TNEWS ACC ↑	IFLYTEK ACC ↑	CMNLI ACC ↑	WSC ACC ↑	CSL ACC ↑	AVERAGE ACC ↑
Jamba	12.55	7.24	78.65	63.15	72.40	79.66	76.64	84.83	75.89
Jamba-SABLH	12.30	5.42	79.48	77.98	72.71	79.59	79.93	85.06	79.12

remember specific formats based on the context and reason based on these formats in the subsequent context. This has been proven to be a deficiency in the ability of SSM to aggregate state information recursively. The self-attention part in a single Jamba block is embedded between the state-space layers, resulting in the information state produced by the ability of self-attention to route the knowledge of each answer directly to a single answer token being decayed by the subsequent three SSM layers, ultimately causing bias in knowledge routing. Jamba-SABLH re-weights the state information recursively aggregated by state-space in the last layer with self-attention, which is equivalent to learning and weighting the process of how the information from the first token to the last token is recursively aggregated. Then the two state information is mixed together, effectively avoiding bias in knowledge routing. Therefore, we can now explain why Jamba-SABLH has a significant advantage in most validation tasks.

For tasks like AFQMC that judge whether sentences have the same meaning, CSL that judge whether keywords are in the abstract rather than extracting keywords from the abstract, and CMNLI that judge whether two sentences imply, are neutral, or are contradictory, they mainly rely on selectively extracting key information on the sequence and recursively aggregating state information. Therefore, the performance gap between Jamba and Jamba-SABLH on these tasks is not significant, and the slight lead of Jamba-SABLH on AFQMC and CSL may be due to it correcting some knowledge routing bias. In tasks like TNEWS that predict the category of short sentences, since the sentences do not contain or contain very little context information, they rely heavily on the quality of pre-training and fine-tuning. This requires that the knowledge learned by the model in the pre-training phase can be correctly and accurately guided in the fine-tuning phase, resulting in Jamba's performance on TNEWS being far worse than Jamba-SABLH. In contrast to TNEWS, the text in the IFLYTEK task is all long context, and the vocabulary of the application topic appears multiple times. This constitutes a relatively simple single-query associative recall task, where the selectively stored single topic vocabulary that appears multiple times is allowed to be stored in the state by the selective state-space model and becomes the main factor affecting the output. Therefore, the performance gap between the two on IFLYTEK is not significant. In tasks like WSC that require pronoun disambiguation, the model needs to be able to trace back to the noun that appeared before the pronoun after the pronoun appears and point the pronoun to the correct noun. This belongs to the additional requirement of understanding the current language in the single-query associative recall task. Therefore, the reason for the performance gap between the two on WSC is obvious in these tasks, the two main fac-

Table 3. **Parameters of Jamba and Jamba-SABLH for Associative Recall**. The hybrid cycle of Jamba is equivalent to the hybrid configuration in Table 1 three times, and Jamba-SABLH is the same as Jamba except for the last layer of the self-attention module. In addition, we use mixture of experts as the feed-forward network to reduce training costs. We carefully adjusted d_{ffn} to keep the parameter volume at 1B.

MODEL	PARAMS	n_{layers}	d_{model}	n_{heads}	d_{state}	$n_{experts}$	$top_{experts}$
Jamba	1B	24	1024	16	64	4	1
Jamba-SABLH	1B	24	1024	16	64	4	1

Table 4. **Results of Jamba and Jamba-SABLH for Associative Recall**. We selected associative recall validation tasks including: standard short context task: PIQA (Bisk et al., 2020), natural long context task: NarrativeQA (Kovcisky et al., 2018), synthetic long context task: HotpotQA (Yang et al., 2018). We evaluated the performance of Jamba and Jamba-SABLH on these three tasks. Jamba-SABLH improved by 1.1% on the standard short context task, 20.86% on the natural long context task, 24.15% on the synthetic long context task, and an average improvement of 10.24%.

MODEL	PIQA ACC ↑	NARRATIVEQA ACC ↑	HOTPOTQA ACC ↑	AVERAGE ACC ↑
Jamba	78.65	23.15	34.40	45.40
Jamba-SABLH	79.48	27.98	42.71	50.05

tors are the difference in the quality of fine-tuning and the re-attention of the recursively aggregated information that gradually decays the long-term strong dependency relationship before prediction by Jamba-SABLH.

4.3. Associative Recall

We trained 1B parameter Jamba and Jamba-SABLH models on 80% of the pre-training dataset with a sequence length of 2048, then packaged the remaining 20% of the pre-training dataset into a dataset with a context length of 8192 to continue pre-training the model. Then the fine-tuning dataset was also packaged into a dataset with a context length of 8192 to fine-tune the model. The two models were extended to a context length of 8192 for evaluation, and the parameter settings of the two models are shown in Table 3.

As shown in Table 4, we show the performance of Jamba and Jamba-SABLH on associative recall tasks, including standard short context tasks, natural long context tasks, and synthetic long context tasks. In the standard short context task, it is obvious that the model does not need to have the ability to handle long-distance dependencies. Our goal is to evaluate whether the accuracy of short-distance associative recall of Jamba-SABLH will be affected when the Jamba-SABLH model is extended to handle long-context environments in common tasks. The experimental results show that Jamba-SABLH outperforms Jamba in the standard short context task. Therefore, Jamba-SABLH does not sacrifice accuracy in common tasks while adapting to long-context environments. In the natural long context task, Jamba-SABLH achieved a performance improvement of up to 20.86% compared to Jamba. We believe that the main reason for this improvement is that the state-space has a learnable matrix (A, B, C) in the training phase. However, in the inference phase, although the values of these matrices can be dynamically calculated according to different input data, the parameters used for these calculations (i.e., the function or mapping that determines how to calculate these matrices and the step size) are fixed, and these parameters are reused in the inference phase after being determined in the training phase. This means that the state of the state-space may still be influenced by irrelevant documents during inference. In contrast, the last layer of Jamba consists of state-space and MLP. In inference, not only the output state may be confused when approaching the output, but the output state may also be unstable. These factors together result in Jamba's performance in handling long-context tasks being inferior to Jamba-SABLH. It is well known that Transformers models tend to copy the answers in the context examples rather than predict the actual answers to the questions. In contrast, SSMs models tend to predict answers and are difficult to copy answers directly from the context. In the synthetic long context task, the model needs to retrieve information from a large amount of irrelevant text and demonstrate the ability to track and aggregate information across contexts. Jamba-SABLH outperforms Jamba by 24.15% in the synthetic long context task, and the reasons behind it are similar to the attention decay and knowledge routing bias discussed in the previous section 4.2. Jamba-SABLH corrects the knowledge routing bias by re-attending to the recursively aggregated information that gradually decays the long-term dependency relationship and learns how

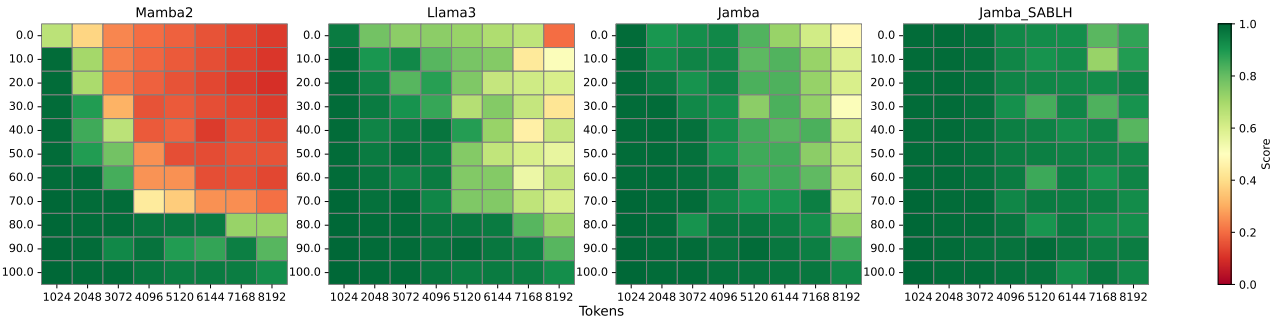

Figure 2. **Needle in a Haystack**. Performance of models with different architectures on the Needle in a Haystack task. We evaluated Mamba2 (Dao & Gu, 2024) using state-space alone, Llama3 (Grattafiori et al., 2024) using self-attention alone, Jamba using a hybrid algorithm of state-space and self-attention, and our method Jamba-SABLH on different context lengths. Our method shows the best performance.

SSMs recursively aggregate information. This mechanism balances the ability to copy answers from context examples and predict actual answers to questions, enabling Jamba-SABLH to demonstrate superior performance in handling synthetic long context tasks, surpassing Jamba.

4.4. Needle in a Haystack

We further compare the performance of Jamba-SABLH, Jamba, Llama3, and Mamba2 in the synthetic retrieval task Needle in a Haystack. A random and informative sentence (i.e., needle) is inserted into a long document (i.e., haystack), and the model is required to retrieve the needle from the haystack to answer the question. All models are of size 1B and trained with the same settings: 80% of the pre-training dataset uses a sequence length of 2048, the remaining 20% of the dataset uses a sequence length of 8192 for pre-training, and the fine-tuning dataset also uses a sequence length of 8192 for fine-tuning. As shown in Figure 2, the Jamba-SABLH model significantly outperforms Mamba2, Llama3, and Jamba.

5. conclusion

This paper re-examines the performance of the hybrid modeling architecture combining state-space algorithms and self-attention algorithms in context learning tasks. Through detailed analysis and extensive experiments, we reveal the limitations of existing hybrid models in context learning tasks, although they perform better than using state-space or self-attention algorithms alone in most language modeling tasks. Our proposed "Self-Attention before LM-Head" method has made significant progress in addressing these limitations. By modifying the hybrid architecture to make the state-space and self-attention use the same encoding and adding a Transformer module composed of self-attention and feed-forward networks before the language model head (LM-Head), we effectively alleviate the bias in knowledge routing and enhance the model's ability to leverage the advantages of state-space and self-attention mechanisms. This improvement has achieved significant performance improvements in various tasks, including standard short text tasks, natural long text tasks, and synthetic long text tasks, and has reached the state-of-the-art performance level in some tasks. The experimental results show that it is crucial to correctly integrate these two algorithms to fully exploit their complementary advantages. The redesigned Jamba-SABLH not only improves the accuracy of the model in common tasks of short-distance associative recall but also significantly enhances the model's ability to handle long text tasks, demonstrating its robustness and adaptability. In addition, our method performs well in the "Needle in a Haystack" task, demonstrating its significant advantage in balancing the ability to copy answers from context and predict actual answers to questions. Our work provides new ideas for improving the hybrid modeling architecture and provides valuable insights for further improving the performance of the model in context learning tasks.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, 2020.

Brightmart. translation2019zh dataset, 2019. A large-scale

Chinese-English parallel corpus with 5.2 million sentence pairs.

Coordination, E. L. R. Paracrawl dataset, 2024. Accessed: 2025-01-28.

Dao, T. and Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 320–335, 2022.

for Machine Translation, E. A. News commentary dataset, 2024a. Accessed: 2025-01-28.

for Machine Translation, E. A. News crawl dataset, 2024b. Accessed: 2025-01-28.

Foundation, W. Wikipedia corpus, 2024. Accessed: 2025-01-28.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L.,

Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

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

Kovcisky, T., McCann, B., Bradbury, J., Xiong, C., and Socher, R. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Li, Y., Zhang, Y., Zhao, Z., Shen, L., Liu, W., Mao, W., and Zhang, H. CSL: A large-scale Chinese scientific literature dataset. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 3917–3923, Gyeongju, Republic of Korea, 2022. International Committee on Computational Linguistics.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Nations, U. Uncorpus dataset, 2024. Accessed: 2025-01-28.

NVIDIA, M. Pytorch container image. https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch, 2022.

Shi, J. and Wu, B. Wonderful matrices: Combining for a more efficient and effective foundation model architecture, 2024. URL https://arxiv.org/abs/2412.11834.

Team, C. Cluecorpussmall dataset, 2020. Accessed: 2025-01-28.

Unknown. Wikimatri dataset, 2024. Accessed: 2025-01-28.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Xu, L., Hu, H., Zhang, X., Li, L., Cao, C., Li, Y., Xu, Y., Sun, K., Yu, D., Yu, C., Tian, Y., Dong, Q., Liu, W., Shi, B., Cui, Y., Li, J., Zeng, J., Wang, R., Xie, W., Li, Y., Patterson, Y., Tian, Z., Zhang, Y., Zhou, H., Liu, S., Zhao, Z., Zhao, Q., Yue, C., Zhang, X., Yang, Z., Richardson, K., and Lan, Z. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 4762–4772, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.419. URL https://aclanthology.org/2020.coling-main.419.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Ziegler, C.-N. Book-crossing dataset, 2004. Accessed: 2025-01-28.