Towards Efficient Long-context Modeling in Large Language Models

Chien Van Nguyen,
[1]Department of Computer Science, University of Oregon, Eugene, OR, USA
chienn@uoregon.edu

Abstract

Effectively processing and utilizing information across long text sequences is a fundamental challenge in advancing Natural Language Processing (NLP). Tasks like document-level information extraction inherently require models to understand context that spans beyond single sentences, often throughout an entire document. Our recent work on Document-level Event Argument Extraction (EAE) achieved state-of-the-art performance by leveraging contextualized soft prompts and aggregating relevant document context. However, this method, like many other powerful NLP models, relies on underlying Large Language Models (LLMs) with fixed and relatively short context windows, typically limited to a few thousand tokens. This report presents this state-of-the-art EAE work and highlights how its success, despite these context constraints, reveals the critical need for LLM architectures capable of handling significantly longer contexts efficiently. We then introduce Taipan, a novel hybrid LLM architecture we developed that combines the efficiency of State Space Models (Mamba-2) with the expressive power of Selective Attention Layers. Taipan is designed to efficiently model dependencies and retrieve information across context lengths up to one million tokens. We demonstrate Taipan's superior performance on long-context retrieval and extrapolation tasks, showing its potential to overcome the context bottleneck faced by current SOTA models and enable future advancements in tasks like document-level EAE on unprecedented scales.

1 Introduction

A major research direction in the field of Large Language Models (LLMs) today is Long-context Modeling. This aims to significantly improve LLMs' efficiency and capability in handling extreme long input sequences. The motivation for this is clear: many real-world tasks demand the ability to understand and process documents, conversations, or data streams that easily exceed hundreds of thousands or even a million tokens. Examples span diverse domains, from analyzing scientific literature and legal documents to processing historical archives and engaging in extended dialogues.

However, the prevailing LLM architecture, the Transformer (Vaswani et al., 2017), faces fundamental limitations here. The self-attention mechanism, while powerful for capturing dependencies, has a quadratic computational complexity ($\mathcal{O}(L^2)$) with respect to the sequence length (L). This leads to computational bottlenecks that become prohibitive for long inputs. Furthermore, during inference, the memory usage for storing the Key-Value (KV) cache grows linearly ($\mathcal{O}(L)$) with context length, quickly exceeding available hardware capacity. These limitations make it challenging to scale standard Transformers to the required context lengths, often necessitating truncation or complex chunking strategies that can compromise accuracy by breaking long-range dependencies.

This presents a key challenge in balancing efficiency (reducing memory usage and boosting computation speed) with accuracy (enhancing performance on tasks requiring long contexts).

A significant application area that both benefits from and exemplifies the need for long-context LLMs is **Document-level Information Extraction (IE)**. Unlike traditional IE methods or simpler sentence-level approaches, document-level IE focuses on capturing relationships, events, and context that span an entire document, requiring the integration of information from potentially distant parts of the text. The challenges here are twofold: effectively reasoning across long contexts with distant dependencies, and maintaining computational efficiency when processing large documents or collections of documents.

Despite these challenges, LLMs offer compelling opportunities for Document-level IE. Their pretraining on vast corpora provides a strong foun-

dation for language understanding, which can be further enhanced by incorporating structured documents into training. Crucially, the advent of truly long-context LLMs has the potential to revolutionize this field by enabling models to process full documents without resorting to information-losing truncation.

These observations motivate two core research questions driving our work:

Q1: How can we enable efficient and scalable long-context modeling in LLMs? This question probes the architectural and algorithmic advancements needed to process extremely long context efficiently. What modifications can reduce memory and computational costs? How can we maintain or improve performance on long-range dependencies? Can techniques like KV cache compression or pruning play a role?

Q2: How can we improve document-level information extraction using LLMs? This application-focused question investigates how LLMs can better handle the complexities of document structure and dependencies. How can models effectively capture cross-sentence and cross-page relationships for tasks like event argument extraction or relation extraction?

Addressing Q2 on the scale required by real-world documents (i.e., large documents) fundamentally depends on breakthroughs in Q1. Without efficient long-context architectures, the application of powerful LLMs to large documents for IE tasks remains constrained.

The following sections detail our research contributions towards these questions. We will first present our work on Document-level Event Argument Extraction, which achieved state-of-the-art performance on standard benchmarks and underscored the potential of LLMs for IE tasks, while simultaneously highlighting the limitations imposed by current context windows. We will then introduce our work on Taipan, a novel LLM architecture designed to directly address the efficiency and scalability challenges of Q1, enabling truly long-context modeling and paving the way for future advancements in applications like Document-level IE on unprecedented scales. [1]

[1]The next chapters include material from published papers and preprints. We acknowledge the contributions of co-authors Hieu Man, Thien Huu Nguyen, Huy Huu Nguyen, Thang M. Pham, Ruiyi Zhang, Hanieh Deilamsalehy, Puneet Mathur, Ryan A. Rossi, Trung Bui, Viet Dac Lai, and Franck Dernoncourt.

2 Document-level Event Argument Extraction (EAE): A Case for Long-Context Needs

2.1 Introduction

As an important task in Information Extraction (IE), Event Argument Extraction (EAE) aims to recognize event arguments and roles for given event mentions in text. For example, in the text "*On the morning of 1 March 2019, Taliban gunmen and suicide bombers **attacked** Camp Shorabak.*" with the event trigger "*attacked*" of type *Conflict.Attack*, the goal of EAE systems is to identify "*gunmen*" and "*bombers* as the *Attacker* argument, and "*Camp Shorabak*" as the *Target*. Along with event detection, EAE has important applications for different natural language processing (NLP) tasks.

EAE research progress has been accelerated by deep learning architectures to significantly boost extraction performance. Early deep learning models for EAE have followed the traditional approaches to formulate EAE as classification or sequence labeling problems (Chen et al., 2015; Nguyen et al., 2016; Sha et al., 2018; Liu et al., 2018; Nguyen and Nguyen, 2018; Pouran Ben Veyseh et al., 2022). Recently, there has been a growing interest in solving EAE in the new question answering or text generation frameworks to better exploit task-specific information (e.g., labels/descriptions of argument roles) via prompts for pre-trained language models (PLM). As such, question answering methods for EAE create a question for each argument role to perform span extraction over input context (Du and Cardie, 2020; Liu et al., 2020; Li et al., 2020; Liu et al., 2021a) while text generation models directly consume an input text and argument-specified prompt/template to generate arguments for each event mention (Li et al., 2021; Zeng et al., 2022). However, a common issue in current prompt-based methods for EAE involves the use of discrete and manually-designed prompts to present task information for the models, e.g., event types and argument roles. As such, these prompts often follow some pre-defined templates (Li et al., 2021; Ma et al., 2022) that are applied to extract arguments for all events in text. While convenient for human understanding, discrete and pre-defined prompts might not be ideal for all examples, causing sub-optimal performance (Liu et al., 2021b). The discrete nature also makes it challenging to achieve prompt customization for each example in EAE models. Further, due to the

employment of PLMs, it has been observed that model performance can be very sensitive to specific formulations of discrete prompts (Zhao et al., 2021; Ma et al., 2022), leading to instability and less reliability when adapting to different datasets.

Another issue of hard prompts for EAE models concerns other relevant examples from training data that can provide helpful information to support argument prediction for current input text and event type. As such, a few recent work has retrieved related examples for an input text to combine with hard prompts to improve EAE (Du et al., 2022; Du and Ji, 2022). However, due to the input length limit of PLMs, the number of relevant examples in the prompts is also constrained, thus unable to fully leverage their advantages to boost performance.

To this end, our work proposes a novel prompt-based method for EAE where learnable soft prompts are explicitly introduced to enable prompt customization for examples, stability improvement, and incorporation of relevant example context. In particular, based on the architecture of generative PLMs, our model directly utilizes input example representations to compute soft prompts for EAE, thus allowing the prompts to be specifically designed for each example for better customization. In addition, soft prompts facilitate the accumulation of representations of relevant examples for an input event type to consume more examples for richer prompts. To exploit this flexibility of soft prompts, our model extensively considers relevant examples as the texts in training data that contain similar event types to an input text, leading to comprehensive external event context to aid EAE. Accordingly, we introduce a graph structure to capture mentioning relations between documents and event types. This graph is then fed into graph neural networks to facilitate representation aggregation of relevant documents with similar events for soft prompt computation. We evaluate the proposed model for EAE on the benchmark datasets RAMS and WIKIEVENTS. The results demonstrate the benefits of the proposed method, leading to state-of-the-art performance for EAE.

2.2 Model

In EAE, given an event trigger/mention in a document, we need to identify argument spans and roles for the event. For convenience, let $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{D}|}\}$ be the set of documents in the training data and $e_k \in D_i$ be the current event trigger in document D_i with event type et for EAE.

Relevant Context Aggregation: Our EAE model follows the prompt-based framework (Ma et al., 2022) where a prompt is created for each event type and fed into the pre-trained language model BART (Lewis et al., 2020) to perform span extraction for the argument roles. As such, in contrast to hard and manually-designed prompts as in previous work, our model introduces soft prompts with example customization and relevant example aggregation to boost the performance. In particular, given the current event trigger $e_k \in D_i$, our model first aims to aggregate context representations from relevant documents in $\mathcal{D}$ for the event type et of e_k to enrich soft prompt computation. Motivated by relevant documents via similar event types, we we first construct an event-type mentioning graph $\mathcal{G}$ between the documents in $\mathcal{D}$ and the event types $\mathcal{T} = \{t_1, \ldots, t_{|T|}\}$ to facilitate representation aggregation. In particular, the node set $\mathcal{V}$ for our graph involves both documents and event types, i.e., $\mathcal{V} = \mathcal{D} \cup \mathcal{T}$. We only connect a document $D_u \in \mathcal{D}$ and an event type t_v in $\mathcal{G}$ if there exists an event mention of type t_v in D_u. In this way, we can link the documents in $\mathcal{D}$ via similar event type mentioning for convenient representation aggregation with graph neural networks.

To obtain representations for each document $D_u \in \mathcal{D}$, we introduce the markers <ET> and </ET> before and after each event trigger word in D_u to generate the marker-augmented document $\hat{D}_u$. The augmented document is then sent into the encoder of BART to produce a representation for each word in $\hat{D}_u$ (using the averages of hidden vectors in the last layer for sub-words). Afterward, the representation $\overline{D}_u^0$ for $D_u \in \mathcal{D}$ is computed by performing mean pooling over the representations for the <ET> markers of event triggers, aiming to retain event-focused context in the representation. For the event types $t_v \in \mathcal{T}$, we initialize their representations $\bar{t}_v^0$ randomly. Afterward, the graph $\mathcal{G}$ and representations $\overline{D}_u^0$ and $\bar{t}_v^0$ for documents and event types are consumed by a graph attention network (Veličković et al., 2017) to aggregate the representations via the connections in $\mathcal{G}$, producing richer representations $\overline{D}_u^L$ and $\bar{t}_v^L$ for D_u and t_v after L layers of transformation. Consequently, we treat the induced representation $\overline{et}$ for the current event type et as the aggregation for context information of relevant documents for prompt computation for $e_k \in D_i$ in the next steps.

Soft Prompt Computation: For convenience, let $\overline{e}_k$ be the representation for the event trigger e_k after the marker-augmented document $\bar{D}_i$ for D_i is encoded by the BART encoder in the previous step. As such, our soft prompt for EAE for trigger $e_k \in D_i$ will be a matrix P_{soft} of size $M_s \times d$ where M_s is a hyper-parameter and d is the dimension of the hidden vectors in our core model BART, thus allowing P_{soft} to be integrated into the computation of BART. To achieve a customized soft prompt P_{soft} for e_k in our model, the contextual representation $\overline{e}_k$ for e_k will be utilized to compute P_{soft}. In addition, as discussed above, P_{soft} will also be conditioned on the aggregation of relevant context representations $\overline{et}$ for the the event type et of e_k to enrich the prompt and facilitate argument extraction. To this end, we utilize a learnable feed-forward network FF to transform the concatenation $\overline{e}_k$ and $\overline{t}_k^L$ into a vector of size $M_s \times d$. This vector will then be reshaped to form our soft prompt $P_{soft} = \text{reshape}(FF([\overline{e}_k; \overline{et}]))$.

Prompt-based EAE: While soft prompts enable example customization and relevant context incorporation for the models, we further inherit the hard prompts to explicitly specify expected argument roles for span extraction. In particular, to achieve a fair comparison, we utilize the same hard prompt for each event type as in previous work (Li et al., 2021; Ma et al., 2022) that connects all argument roles with natural language. For example, for the event type *Life.Consume.Unspecified*, the hard prompt to indicate argument roles is: *<ConsumingEntity> consumed <ConsumedThing> at <Place> place*.

For each event type t, we send its hard prompt into the embedding layer of BART to obtain a representation for each word (i.e., using averages of embeddings for sub-tokens), leading to the hard prompt representation P^t_{hard} of size $M^t_h \times d$ (M^t_h is the length of the hard prompt for t). We then concatenate the soft and hard prompt representations to create a single prompt Pr for EAE with BART, i.e., $Pr^t = [P^t_{hard}, P_{soft}]$ of size $(M_s + M^t_h) \times d$. Next, we follow the prompt-for-extraction framework in (Ma et al., 2022) to use the BART encoder to encode input context for D_i while the prompt P^t will be employed to prompt the BART decoder for span extraction. Given the current trigger $e_k \in D_i$, we first inject the trigger markers $<ET_i>$ and $</ET_i>$ before and after e_k in D_i to create an input context D'_i, which is then encoded by the BART encoder to return a sequence of representations D^{enc}_i for the words in D'_i. In the next step, the BART decoder is employed to learn richer representations for the context and prompt using their interactions via cross-attention in multiple layers, returning the representations $D^{dec}_i = \text{BART-Decoder}(D^{enc}_i; D^{enc}_i)$ and $\overline{Pr}^t = \text{BART-Decoder}(Pr; D^{enc}_i)$ for the context and prompt.

Afterward, for the j-th argument role for event type t, we obtain a role representation ϕ^t_j by mean-pooling its corresponding sub-token representations in the prompt representation $\overline{Pr}^t$. Similar to (Ma et al., 2022), we employ two selection heads s^{start} and s^{end} (of d dimensions) to compute start and end span selectors $\phi^{t,start}_j = \phi^t_j \odot s^{start}$ and $\phi^{t,end}_j = \phi^t_j \odot s^{end}$ ($\odot$ is the element-wise multiplication). Each span selector tuple $\theta^t_j = (\phi^{t,start}_j, \phi^{t,end}_j)$ then aims to select at most one argument span for the j-th role of t. Here, the golden span for this role is denoted by $(a^{t,start}_j, a^{t,end}_j)$. It will be set to $(0, 0)$ if event e_k does not have an argument of this role in D_i. As such, the extractive prompt framework is utilized to estimate distributions over token positions in D for how likely each token in D_i would serve as the start/end position for the argument span of the role: $P^{t,start}_j = \text{softmax}(\phi^{t,start}_j D^{dec}_i)$, $P^{t,end}_j = \text{softmax}(\phi^{t,end}_j D^{dec}_i)$. Finally, to train our model, we optimize the loss: $\mathcal{L} = -\sum_{e_k \in \mathcal{D}} \sum_j (\log(P^{t,start}_j(a^{t,start}_j)) + \log(P^{t,end}_j(a^{t,end}_j)))$ (i.e., over all events in $\mathcal{D}$).

Inference: At inference time, given an input text, event type, and argument role, we consider all possible argument spans for the role, ensuring that the start indexes are smaller than the end indexes (including $(0, 0)$) and their lengths do not exceed a maximum value computed over training data. A score for each span is obtained using the probability $\log(P^{t,start}_j(a^{t,start}_j)) + \log(P^{t,end}_j(a^{t,end}_j))$. The span with the highest score will be chosen for prediction. Finally, the aggregations $\overline{et}$ of relevant context representations for event types, which are learned during training, are used in test time.

2.3 Experiments

Datasets and Hyper-parameters: Following previous work (Li et al., 2021; Ma et al., 2022), we employ two latest datasets for EAE to evaluate our model, i.e., RAMS (Ebner et al., 2020) and WIKIEVENTS (Li et al., 2021). Both datasets involve multiple events in a document where argu-

ments can distribute over different sentences from the event triggers. We utilize the same train/dev/test splits, data pre-processing, and evaluation metrics for the datasets as in previous work (Ma et al., 2022) for fair comparison. In particular, our metrics include: Argument Identification F1 score (Arg-I) and Argument Classification F1 score (Arg-C) scores. For WIKIEVENTS, we also use Argument Head F1 score (Head-C) to only consider headword matching for arguments. Finally, we fine-tune the hyper-parameters for our model on the development data.

Comparison: We compare our method (called **SPEAE** for soft prompts for EAE) with the state-of-the-art models for EAE. In particular, we consider two groups of baselines: (i) text generation-based models: BART-Gen (Li et al., 2021), and (ii) question answering-based models: FEAE (Wei et al., 2021), DocMRC (Liu et al., 2021a), EEQA (Du and Cardie, 2020), EEQA-BART (Du and Cardie, 2020), EA2E (Zeng et al., 2022), and PAIE (Ma et al., 2022). The performance of EA2E is obtained by running the provided code over our preprocessed data using the same evaluation metrics for a fair comparison. The performance for other baselines is inherited from (Ma et al., 2022), which presents the model PAIE with current best-reported results for our datasets.

Model	PLM	RAMS		WIKIEVENTS		
		Arg-I	Arg-C	Arg-I	Arg-C	Head-C
BART-Gen	BART-b	50.9	44.9	47.5	41.7	44.2
	BART-l	51.2	47.1	66.8	62.4	65.4
FEAE	BERT-b	53.5	47.4	-	-	-
DocMRC	BERT-b	-	45.7	-	43.3	-
EEQA	BERT-b	46.4	44	54.3	53.2	56.9
	BERT-l	48.7	46.7	56.9	54.5	59.3
EEQA-BART	BART-b	49.4	46.3	60.3	57.1	61.4
	BART-l	51.7	48.7	61.6	57.4	61.3
EA2E	BART-b	-	-	64.5	58.6	61.7
	BART-l	-	-	70.8	65.7	67.8
PAIE	BART-b	54.7	49.5	68.9	63.4	66.5
	BART-l	56.8	52.2	70.5	65.3	68.4
SPEAE (ours)	BART-b	**56.0**	**51.1**	**70.6**	**66.2**	**69.6**
	BART-l	**58.0**	**53.3**	**71.9**	**66.1**	**68.8**

Table 1: Model performance on test data.

Table 1 shows the performance of the methods over the test datasets along with their corresponding PLM versions. The most important observation from the table is that the proposed method SPEAE significantly outperforms the baseline methods (with $p < 0.01$) for both base and larger versions of the PLM models (i.e., BERT and BART). It just clearly demonstrates the benefits of the proposed method for EAE with contextualized soft prompts for instances and relevant context.

Model	**Arg-I**	**Arg-C**	**Head-C**
SPEAE (full)	70.6	66.2	69.6
No example context $\overline{e}_k$ for P_{soft}	69.7	65.5	68.4
No relevant context $\overline{et}$ for P_{soft}	70.1	65.8	68.9
No soft prompt P_{soft}	69.4	64.7	67.5
No graph for $\overline{et}$	69.1	65.3	68.1

Table 2: Ablation study on test data.

Ablation Study: To reveal the contribution of the designed components in SPEAE, we perform an ablation study over the WIKIEVENT test data. Table 2 presents the performance of the ablated models. In particular, for soft prompts, we first exclude either the example-specific context representation $\overline{e}_k$ or the relevant context aggregation $\overline{et}$ from the computation of the soft prompt P_{soft}. As the performance of the resulting models is significantly worse than SPEAE, it clearly testifies to the importance of such components for prompt-based models for EAE. The performance is furthered degraded when the soft prompt P_{soft} is completely eliminated from the prompt, thus suggesting the effectiveness of soft prompts for EAE. Additional, instead of computing the relevant context aggregations for event types $\overline{et}$ with a graph neural network, we explore a variant to directly obtain $\overline{et}$ from the average representation of the documents in $\mathcal{D}$ that contain an event mention of type et. The worse performance of the ablated model clearly confirms the benefits of the graph neural network for representation aggregation of relevant documents/event types for soft prompt computation for EAE.

Low-resource Learning: To better understand the operation of the proposed model SPEAE under low-resource training settings, we perform an evaluation when different ratios of training data are employed to train the models. In particular, we compare SPEAE with the previous state-of-the-art models, i.e., EEQA (Du and Cardie, 2020), EEQA-BART (Du and Cardie, 2020), and PAIE (Ma et al., 2022) in this low-resource learning experiment. Table 3 demonstrates the performance of the models (based on Arg-C) on the development data of WIKIEVENTS. As can be seen from the table, the proposed model SPEAE is significantly better than the baseline methods over different ratios of training data, ranging from 1% to 50%. It just clearly highlights the advantages of our proposed method for low-resource learning settings. We attribute these advantages to the introduction of context information from current example and

relevant documents to enrich soft prompts, allowing SPEAE to better utilize available training data to boost performance.

Model	Training Data Ratio					
	1%	2%	5%	10%	20%	50%
EEQA	15.0	18.1	35.7	43.2	45.5	49.6
EEQA-BART	21.2	18.3	42.9	44.3	54.1	56.8
PAIE	31.3	40	52.1	51.4	54.9	59.8
SPEAE (ours)	**35.0**	**43.8**	**52.3**	**56.7**	**58.7**	**64.7**

Table 3: Low-resource learning performance (Arg-C) of the models on the development data of WIKIEVENTS. Models are trained on different ratios of training data.

Stability Study: One of the major issues with the discrete prompts in previous EAE models is that model performance can be sensitive to specific formats of the hand-designed prompts (Zhao et al., 2021; Ma et al., 2022). This raises an important concern for the applications of EAE models to different datasets and problems as optimal formats of the prompts might be unclear for new datasets, necessitating further laborious efforts for prompt development and selection. To understand the sensitivity/stability of EAE models over different formats of discrete prompts, this experiment explores three variants of discrete prompts for EAE as discussed in (Ma et al., 2022), i.e., Manual Template, Uncontextualized Soft Prompt, and Concatenate Template. In particular, Manual Template (MA) involves the discrete prompts we utilize in our work, (Li et al., 2021), and (Ma et al., 2022). It concatenates all argument roles for an event type using natural language. For Uncontextualized Soft Prompt (USP), the prompts link argument roles with role-specific special tokens (Qin and Eisner, 2021; Liu et al., 2021b). These tokens are associated with learnable embedding vectors to help transform discrete prompts into representation vectors for further computation. Here, a key difference between these embeddings for argument role-specific tokens and our soft prompts is that our soft prompts are contextualized over current example context and relevant documents. In contrast, the learnable embeddings for role-specific tokens in USP are only initialized randomly, thus unable to contextualize over example context for better customization and richer prompts as in our soft prompts. Finally, in Concatenate Template (CA), all argument role names for an event type are simply concatenated to form prompts (Ma et al., 2022).

Using three variants of discrete prompts, we compare the performance (based Arg-C) of our proposed model SPEAE and the current state-of-the-art discrete-prompt model PAIE for EAE. Table 4 presents model performance on the RAMS development data. It is clear from the table that the proposed model SPEAE performs significantly better than PAIE over different variants of discrete prompts, thus further demonstrating the benefits of SPEAE. Importantly, while PAIE exhibits diverse performance gaps across different discrete prompts, SPEAE maintains more stable performance. This suggests an important advantage of SPEAE that is less sensitive to specific discrete prompt formats to enable convenient extension to new applications with less development efforts for prompt design.

Model	MA	USP	CA
PAIE	48.8	47.4	45.2
SPEAE (ours)	**51.5**	**52.2**	**51.1**

Table 4: Model performance over the development data of RAMS using different variants of discrete prompts: MA (Manual Template), USP (Uncontextualized Soft Prompts), and CA (Concatenate Template). Performance for PAIE is obtained by running the provided code from the original paper to achieve fair comparison.

3 Taipan: An Architecture for Efficient Long-Context Modeling

3.1 Introduction

Transformer-based architectures (Vaswani et al., 2017; Brown, 2020) have revolutionized Natural Language Processing (NLP), delivering exceptional performance across diverse language modeling tasks (Touvron et al., 2023). This success stems from their ability to capture complex word dependencies using the self-attention mechanism. In addition, Transformers are highly scalable and well-suited for parallel training on large datasets. However, despite their success, they still face notable challenges when handling long-context sequences. Specifically, the self-attention mechanism suffers from quadratic computational complexity, and the memory requirement grows linearly with context length during inference, as the model must store key-value vectors for the entire context. These factors impose practical constraints on sequence length due to the high computational and memory costs.

To this end, recent advancements in recurrent-based architectures, particularly State Space Models (SSMs) (Gu et al., 2021b,a), have emerged as

promising alternatives for efficient language modeling (Gu and Dao, 2023; Dao and Gu, 2024). SSMs offer constant memory usage during inference, and architectures like Mamba-2 (Dao and Gu, 2024), a variant of SSMs, have demonstrated performance comparable to Transformers in certain language tasks (Waleffe et al., 2024). Some studies even suggest that SSMs can outperform Transformers in areas like state tracking (Merrill et al., 2024) due to their Markovian nature. However, despite these advancements, SSM-based models still fall short in scenarios requiring in-context retrieval or handling complex long-range dependencies (Arora et al., 2024; Waleffe et al., 2024).

To address these challenges, we introduce Taipan, a hybrid architecture that combines the efficiency of Mamba with enhanced long-range dependency handling through Selective Attention Layers (SALs). While Mamba is highly efficient, it relies on the Markov assumption—where predictions are based solely on the last hidden state—which can lead to information loss for tokens that need interactions with distant tokens. To mitigate this, Taipan incorporates SALs that strategically select key tokens in the input sequence requiring long-range dependencies. These selected tokens first undergo feature refinement to remove unimportant information, and then are passed through an attention module to capture long-range dependencies.

Less critical tokens bypass the attention step, as we hypothesize that their Markovian representations from Mamba contain sufficient information for accurate prediction, obviating the need for additional attention-based augmentation. This selective approach enables Taipan to balance Mamba's computational efficiency with enhanced long-range modeling capabilities.

SALs play a crucial role in Taipan's design, both in enhancing performance and ensuring computational efficiency. By focusing the attention mechanism on a subset of important tokens, SALs reduce the computational costs that come from attention modules. This targeted approach enables Taipan to excel in memory-intensive tasks while maintaining efficiency during both training and inference. Importantly, Taipan retains the linear memory usage characteristic of SSMs, offering a significant advantage over traditional Transformer models in handling extremely long sequences.

We scale Taipan to 190M, 450M, and 1.3B parameters, pre-training on 100B tokens. Experimental results demonstrate Taipan's superior performance across a wide range of tasks. In zero-shot language modeling evaluations, Taipan consistently outperforms both Transformer and Mamba baselines, showcasing its strong general language understanding capabilities. More notably, in memory-intensive tasks such as long-context retrieval and structured information extraction, Taipan exhibits significant improvements over Mamba-2, addressing a key limitation of existing recurrent-based models. Furthermore, Taipan demonstrates remarkable extrapolation capabilities, maintaining high performance on sequences up to 1 million tokens in context-length - while preserving efficient generation capabilities. This combination of broad task proficiency, superior performance in memory-intensive scenarios, and exceptional long-context modeling positions Taipan as a versatile and powerful architecture for advanced language processing tasks.

3.2 Background

This section briefly overviews the foundational architectures relevant to our work. We first review Causal Self-Attention (Vaswani et al., 2017), the core mechanism of Transformer models. We then discuss Linear Attention (Katharopoulos et al., 2020), an efficient variant that achieves linear complexity. Finally, we examine Mamba-2, a recent architecture that generalizes Linear Attention using structured state-space models (SSMs) (Dao and Gu, 2024). We emphasize how each model balances computational efficiency and recall accuracy, particularly in memory-intensive tasks ().

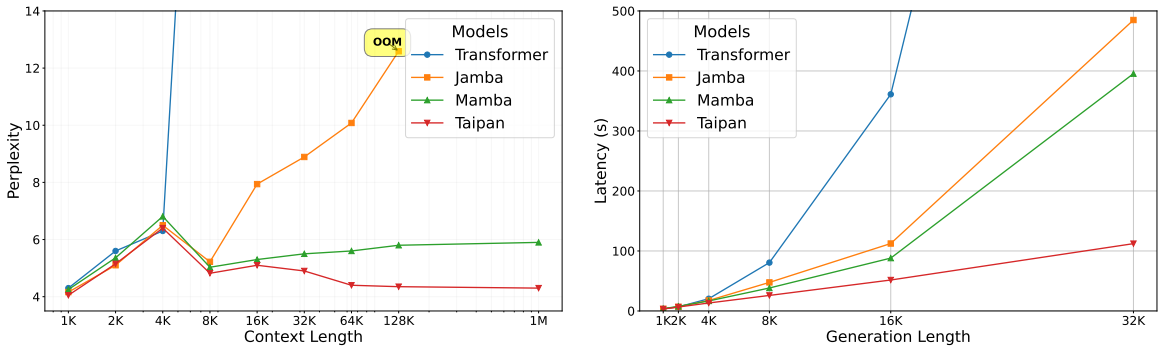
3.2.1 Causal Self-Attention

Causal Self-Attention is the key component in Transformer architectures that allows each token in a sequence to attend to all other previous tokens (Vaswani et al., 2017). Given an input sequence $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L] \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the embedding dimension, self-attention firsts computes the query, key, and value vectors for each token via linear projections:

$$\mathbf{q}_i = \mathbf{W}_Q \mathbf{x}_i, \quad \mathbf{k}_i = \mathbf{W}_K \mathbf{x}_i, \quad \mathbf{v}_i = \mathbf{W}_V \mathbf{x}_i$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are learnable weight matrices.

Then, the attention output $\mathbf{o}_i$ for each token $\mathbf{x}_i$ will be calculated as a weighted sum of the value vectors over the distribution of similarity matrix



a) Perplexity Across Context Lengths

b) Latency Across Generation Lengths

Figure 1: **Model Performance Comparison**. a) Perplexity across different context lengths. Lower perplexity indicates better performance. b) Latency comparison of models at various generation lengths. Taipan exhibits significantly lower latency and superior scaling compared to other strong baselines for longer sequences.

between its query vector and previous key vectors:

$$\mathbf{o}_i = \sum_{t=1}^{i} \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_t/\sqrt{d})}{\sum_{j=1}^{i} \exp(\mathbf{q}_i^\top \mathbf{k}_j/\sqrt{d})} \mathbf{v}_t$$

The non-linear softmax distribution allows the models to capture intricate relationships between tokens, and concentrate on salient features (Qin et al., 2022; Zhao et al., 2019). As such, self-attention can encode complex language patterns and long-range dependencies that are crucial for complex language understanding and generation tasks.

3.2.2 Linear Attention

To address the quadratic complexity, recent work has shown that it is possible to achieve linear complexity with the attention mechanism by replacing the softmax attention with dot-product attention (Shen et al., 2021; Katharopoulos et al., 2020). Given a feature transformation $\phi(\mathbf{x})$, causal self-attention can be rewritten as:

$$\mathbf{o}_i = \sum_{t=1}^{i} \frac{\phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_t)}{\sum_{j=1}^{i} \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)} \mathbf{v}_t$$

Then, using the associate property of matrix multiplication, this can be reformulated as:

$$\mathbf{o}_i = \frac{\phi(\mathbf{q}_i)^\top \sum_{t=1}^{i} \phi(\mathbf{k}_t)\mathbf{v}_t^\top}{\phi(\mathbf{q}_i)^\top \sum_{t=1}^{i} \phi(\mathbf{k}_t)}$$

Let $\mathbf{S}_i = \sum_{t=1}^{i} \phi(\mathbf{k}_t)\mathbf{v}_t^\top$ and $\mathbf{z}_i = \sum_{t=1}^{i} \phi(\mathbf{k}_t)$. We can then rewrite the equation in a recurrent form:

$$\mathbf{S}_i = \mathbf{S}_{i-1} + \phi(\mathbf{k}_i)\mathbf{v}_i^\top$$

$$\mathbf{o}_i = \frac{\mathbf{S}_i \phi(\mathbf{q}_i)}{\mathbf{z}_i^\top \phi(\mathbf{q}_i)} \approx \mathbf{S}_i \phi(\mathbf{q}_i)$$

This formulation allows for efficient training and inference. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$ be the query, key, and value matrices of the sequence input $\mathbf{X}$. During training, we can use the matrix multiplication form: $\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}_L)\mathbf{V}$, where $\mathbf{M}_L$ is a causal mask. At inference time, we can use the recurrent form for efficient sequential processing.

However, despite its computational efficiency, linear attention has notable limitations compared to softmax attention. The dot-product approximation in linear attention lacks the nonlinear normalization of softmax, often resulting in a more uniform distribution of attention weights (Han et al., 2023). This uniformity can impair the model's ability to focus sharply on specific and relevant tokens. Consequently, linear attention models may underperform in tasks requiring precise in-context retrieval or focused attention on particular input segments (Han et al., 2023).

3.2.3 Mamba-2

3.3 Model

Mamba (Gu and Dao, 2023) is a variant of structured state space models (SSMs) that uses the selective data-dependent mechanism. Mamba-2 (Dao and Gu, 2024) builds on this foundation, revealing deep connections between SSMs and linear attention (Katharopoulos et al., 2020) through the framework of structured state-space duality (SSD).

The core of Mamba-2 can be defined by using the recurrent form:

$$\mathbf{h}_t = \mathbf{A}_t \mathbf{h}_{t-1} + \mathbf{B}_t \mathbf{x}_t$$

$$\mathbf{o}_t = \mathbf{C}_t \mathbf{h}_t$$

where $\mathbf{A}_t$ is further simplified to a scalar multiplied by the identity matrix. This formulation allows

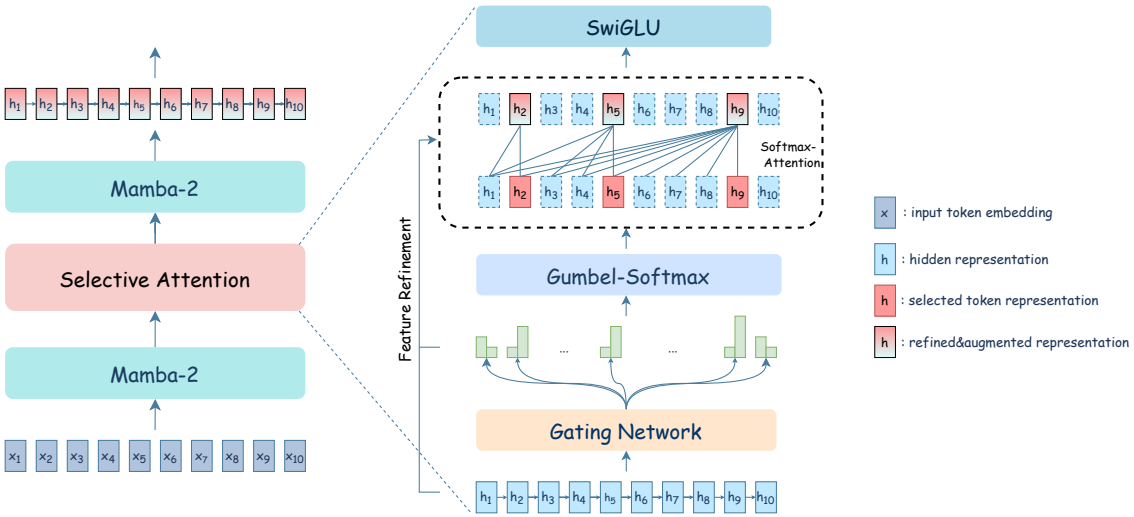


Figure 2: An overview of the Taipan architecture.

Mamba-2 to be interpreted as a generalization of linear attention.

The key insight of Mamba-2 is that this recurrence can be equivalently expressed as a matrix multiplication:

$$\mathbf{O}_t = (\mathbf{L}_t \odot \mathbf{C}_t\mathbf{B}_t^\top)\mathbf{X}_t$$

where $\mathbf{L}$ is a 1-semiseparable matrix. This matrix form reveals the duality between the recurrent (linear-time) and attention-like (quadratic-time) computations. Also, the 1-semiseparable matrix $\mathbf{L}$ encodes the temporal dependencies, while $\mathbf{CB}^\top$ represents content-based interactions similar to attention. This formulation generalizes linear attention, which can be seen as a special case where $\mathbf{L}$ is the all-ones lower triangular matrix.

While Mamba-2 is efficient, it shares the same limitations as Linear Attention in terms of precise memory recall (Arora et al., 2024; Wen et al., 2024), leading to reduced performance in tasks that demand accurate retrieval of specific sections in the input sequence.

To address the limited modeling capabilities of Mamba-2 and Linear Attention while preserving their computational efficiency, we introduce Taipan, a new architecture for sequence encoding in language modeling. In Taipan, we strategically incorporate Selective Attention Layers (SALs) within the Mamba framework, as shown in Figure 2. SALs are inserted after every K Mamba-2 blocks, creating a hybrid structure that combines Mamba-2's efficiency with Transformer-style attention for effective sequence representation.

The core of SALs is a gating network that identifies important tokens for enhanced representation modeling. These tokens undergo two phases: (1) feature refinement to filter out irrelevant information and (2) representation augmentation via softmax attention. This allows Taipan to capture complex, non-Markovian dependencies when necessary.

Taipan processes input through Mamba-2 blocks, with SALs periodically refining key token representations. These enhanced representations are then passed into the subsequent Mamba-2 layers, influencing further processing. This hybrid structure balances Mamba-2's efficiency with the expressive power of SALs, enabling Taipan to excel in tasks requiring both speed and accurate information retrieval. The following sections detail each component's structure and function.

3.3.1 Selective Attention Layers

Selective Attention Layers (SALs) are the key innovation in Taipan, designed to enhance the model's ability to focus on critical tokens while maintaining overall efficiency. These layers employ a lightweight gating network G_θ to dynamically determine which tokens should undergo softmax attention processing.

For each token hidden representation $\mathbf{h}_i$ in the input sequence, the gating network G computes a score vector:

$$\mathbf{s}_i = G_\theta(\mathbf{h}_i) \tag{1}$$

where $G_\theta : \mathbb{R}^d \to \mathbb{R}^2$ is parameterized by θ. This score vector $\mathbf{s}_i = [s_{i,0}, s_{i,1}]$ serves two purposes:

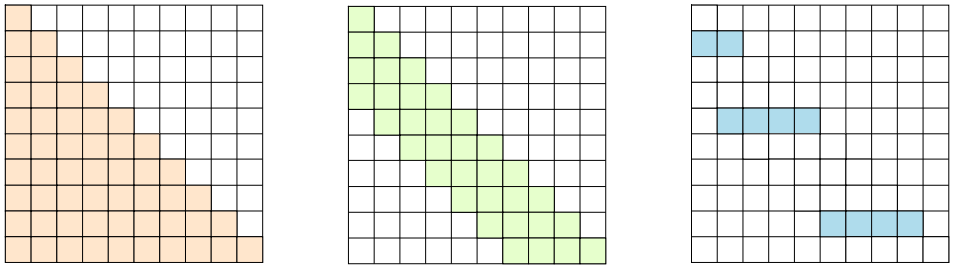


Figure 3: Attention mechanisms in Taipan's Selective Attention Layers. White areas indicate no attention. (a) Full Causal Attention (b) Sliding Window Attention ($w = 4$) (c) Selective Attention ($C = 0.3$, $w = 5$)

1) it is used to generate a binary mask m_i for token selection, and 2) it guides feature refinement.

To maintain differentiability while allowing for discrete token selection, we employ the Straight-Through Gumbel-Softmax trick (Jang et al., 2017). A binary mask m_i is generated from $\mathbf{s}_i$ to select tokens during the forward pass of the network:

$$m_i = \text{argmax}(\text{GumbelSoftmax}(\mathbf{s}_i, \tau)) \quad (2)$$

where τ is the temperature parameter. $\mathbf{h}_i$ will only be selected for attention processing if $m_i = 1$.

For the backward pass, we instead use continuous Gumbel-Softmax approximation of m_i to achieve computation differentiability for the network:

$$z_0 = (\mathbf{s}_{i,0} + g_0)/\tau \quad (3)$$

$$z_1 = (\mathbf{s}_{i,1} + g_1)/\tau \quad (4)$$

$$\tilde{m}_i = \frac{\mathbb{I}[m_i = 0]\exp(z_0) + \mathbb{I}[m_i = 1]\exp(z_1)}{\exp(z_0) + \exp(z_1)} \quad (5)$$

where $\mathbb{I}[]$ is the indicator function, and g_0 and g_1 are i.i.d samples from the Gumbel$(0, 1)$ distribution. In this way, we are able to train our entire model, including the gating network, in an end-to-end fashion for language modeling.

3.3.2 Sliding Window Attention

To maintain linear time complexity while leveraging the benefits of attention, Taipan employs Sliding Window Attention (SWA) (Beltagy et al., 2020). SWA's computational complexity scales linearly with sequence length, allowing Taipan to handle theoretically unlimited context lengths during inference. Importantly, the combination of Selective Attention and Sliding Window Attention in Taipan leads to a significantly sparser attention weight map compared to full attention or standard windowed attention (Figure 3), thus enhancing the computational efficiency of Selective Attention for processing long sequences for our model. In addition, the sparser attention map allows us to afford a longer sliding window (i.e., $w = 2048$ in our work) to effectively capture longer-range dependencies for input sequences. In this way, our designed Taipan architecture offers a mechanism to balance the efficient processing of long sequences with the ability to capture important long-range dependencies, thereby addressing a key limitation of existing efficient attention mechanisms. Finally, removing positional embeddings from the Attention Module improves extrapolation capabilities, suggesting that the model can better generalize temporal relationships. We explore this impact of positional embeddings in more detail in Section 3.5.2.

For the selected tokens (those with a mask value m_i of 1), we compute their attention-based representations:

$$\mathbf{o}_i = \text{Attention}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) \quad (6)$$

where $\mathbf{q}_i$ is the query vector for the i-th selected token (denoted $\mathbf{h}_i^s$), and $\mathbf{K}$ and $\mathbf{V}$ are the key and value matrices for previous tokens.

In our model, the score vector $\mathbf{s}_i$ is also used to refine the representations of selected tokens. We employ the softmax of $\mathbf{s}_i$ to compute the mixing weights: $[1 - \alpha_i, \alpha_i] = \text{softmax}(\mathbf{s}_i)$. The final output for a selected token $\mathbf{h}_i^s$ is a weighted combination:

$$\mathbf{h}_i^s = (1 - \alpha_i)\mathbf{h}_i^s + \alpha_i \mathbf{o}_i \quad (7)$$

As such, Taipan can adaptively preserve key information in $\mathbf{h}_i^s$ while enriching the representation with the attention output $\mathbf{o}_i$. In other words, α_i acts as the *data-dependent factor*, filtering out unimportant features from the original representation while integrating richer information from the attention

outputs. Here, it is important to note that unselected tokens (i.e., $m_i = 0$) skip the attention module and retain their original representations from Mamba-2.

Finally, all token representations are passed through a residual SwiGLU (Shazeer, 2020) layer:

$$\mathbf{h} = \mathbf{h} + \text{SwiGLU}(\mathbf{h}) \tag{8}$$

This final transformation ensures that all token representations undergo consistent non-linear processing before being passed to the next layer in the network, enhancing the model's ability to capture complex dependencies.

3.3.3 Training and Inference

To better balance efficiency and expressiveness, we introduce an attention budget constraint. Given a predefined budget C, representing the desired fraction of tokens to receive attention, we incorporate a constraint loss into our training objective:

$$\mathcal{L}_{\text{constraint}} = \sum_{n=1}^{N} \left\| C - \frac{\sum_{i=1}^{L} m_i}{L} \right\|_2^2 \tag{9}$$

Here, N is the number of SALs, L is the sequence length, and $\sum_{i=1}^{L} m_i$ represents the number of tokens selected for attention processing. During training, we employ the Straight Through Gumbel Softmax estimator for $\tilde{m}_i$ in the backward pass (Jang et al., 2017; Bengio et al., 2013), ensuring differentiability while maintaining discrete token selection in the forward pass, thereby enabling end-to-end training of the entire model. As such, our overall training objective includes a standard cross-entropy loss $\mathcal{L}_{\text{CE}}$ for language modeling and the budget constraint term: $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{constraint}}$, where λ is a hyperparameter.

During inference, Taipan processes input tokens sequentially through Mamba-2 blocks. At each Selective Attention Layer, the gating network G_θ computes a score vector $\mathbf{s}_i = G_\theta(\mathbf{h}_i)$ for each token representation $\mathbf{h}_i$. This score computes a binary mask m_i to determine if $\mathbf{h}_i$ should be used for attention processing. Consequently, our selective attention approach maintains Mamba-2's efficiency for most tokens while applying targeted attention to critical elements, enabling effective long-range dependency modeling with minimal computational overhead.

3.4 Experiments

We conducted extensive experiments to evaluate Taipan's performance across various scales and tasks. Our evaluation strategy focuses on three main areas: (1) zero-shot evaluation on diverse benchmarks to demonstrate Taipan's general language understanding capabilities (Section 3.4.2), (2) in-context retrieval tasks to assess Taipan's ability to retrieve information from historical contexts (Section 3.4.3), and (3) extrapolation ability in long-context scenarios to evaluate performance on extremely long sequences (Section 3.4.4).

3.4.1 Experimental Setup

We evaluate Taipan across three model sizes: 190M, 450M, and 1.3B parameters. To ensure a comprehensive and fair comparison, we benchmark Taipan against three strong baselines:

- **Transformer++** (Touvron et al., 2023): An enhanced version of the LLaMA architecture (Touvron et al., 2023), incorporating Rotary Positional Embeddings (Su et al., 2024), SwiGLU (Shazeer, 2020), and RMSNorm (Zhang and Sennrich, 2019).
- **Mamba-2** (Dao and Gu, 2024): A state-of-the-art linear RNN model based on State Space Models (SSMs). Each Mamba-2 block consists of a depthwise convolutional layer (Poli et al., 2023; Gu and Dao, 2023), an SSM layer (Dao and Gu, 2024), and MLP layers.
- **Jamba** (Lieber et al., 2024): A hybrid model combining full Causal Self-Attention layers (with Rotary Position Embedding (Su et al., 2024)) and Mamba-2 layers. Unlike Taipan, Jamba uses full Causal self-attention instead of selective attention, retains positional embeddings, and lacks a feature refinement mechanism.

Implementation Details We train all models from scratch in three configurations: 190M, 450M, and 1.3B parameters. The training process is consistent across configurations with the following hyperparameters: a batch size of 0.5M tokens per step, a cosine learning rate schedule with 2000 warm-up steps, and AdamW (Loshchilov, 2017) optimization with a peak learning rate of $5e-4$ decaying to a final rate of $1e-5$. We apply a weight decay of 0.01 and use gradient clipping with a maximum value of 1.0. All models are trained with a fixed context length of 4096 tokens.

The training data size varies by model scale: the 190M model is trained on 27 billion tokens,

while the 450M and 1.3B models are trained on 100 billion tokens.

For Taipan-specific implementation, we use a hybrid ratio of $6:1$, inserting a Selective Attention Layer (SAL) after every 6 Mamba-2 Blocks. The Mamba-2 blocks are kept identical to the original work (Dao and Gu, 2024). We set the attention capacity $C = 0.15$. The sliding window attention mechanism uses a window size (w) of 2048 tokens.

3.4.2 Language Modeling Performance

We report the zero-shot performance of Taipan and baseline models on a diverse set of commonsense reasoning and question-answering tasks. These include Winograd (Wino.) (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), HellaSwag (Hella.) (Zellers et al., 2019), ARC-easy and ARC-challenge (ARCe & ARCc) (Clark et al., 2018), OpenbookQA (OB.) (Mihaylov et al., 2018), TruthfulQA (Truth.) (Lin et al., 2021), RACE (Lai et al., 2017), and BoolQ (Clark et al., 2019). It is worth noting that these tasks are brief and do not involve in-context learning, thus inadequately demonstrating long-context modeling or in-context learning retrieval abilities.

Table 5 presents the zero-shot results for models of three sizes: 190M, 450M, and 1.3B parameters. The results are evaluated using the lm-evaluation-harnesshttps://github.com/EleutherAI/lm-evaluation-harness (Gao et al., 2024) framework.

As can be seen, Taipan consistently outperforms the baseline models across most tasks for all model sizes. Notably, the performance gap widens as the model size increases, with the 1.3B Taipan model showing significant improvements over other baselines. This suggests that Taipan's architecture effectively captures and utilizes linguistic patterns, even in tasks that do not fully showcase its long-context modeling capabilities.

3.4.3 In-Context Recall-Intensive Performance

To evaluate Taipan's proficiency in precise in-context retrieval, we assessed all models on a set of recall-intensive tasks (Arora et al., 2024). These tasks are designed to test a model's ability to extract and utilize information from longer contexts, a capability particularly relevant to Taipan's architecture. Our evaluation suite includes two types of tasks: structured information extraction and question answering. For structured information extraction, we used the SWDE and FDA tasks (Arora et al., 2024), which involve extracting structured data from HTML and PDF documents, respectively. To assess question-answering capabilities, we employed SQuAD (Rajpurkar et al., 2018), which requires models to ground their answers in provided documents.

Table 6 demonstrates Taipan's significant performance advantages over both Mamba and Jamba in in-context retrieval tasks. Notably, Taipan achieves this superiority while consuming fewer computational resources than Jamba, which utilizes full attention mechanisms. This efficiency is attributed to Taipan's architecture, which combines Mamba-like elements with selective attention mechanisms, allowing it to filter out less important features. We also notice that Transformers excel at memory-intensive tasks in this experiment; however, they are constrained by linear memory scaling with sequence length, limiting their effectiveness and applicability for very long sequences. In contrast, Taipan maintains constant memory usage, offering a more efficient solution for processing long documents.

3.4.4 Long-Context Extrapolation

Figure 1 illustrates Taipan's superior performance in handling extended sequences compared to Transformer, Jamba, and Mamba models. In perplexity evaluations across context lengths from 1K to 1M tokens (Figure 1a), Taipan yields the lowest perplexity, particularly excelling beyond the training context length.

This performance contrasts sharply with other models: Transformers struggle with longer contexts due to quadratic computational complexity and linear memory scaling with sequence length, often leading to out-of-memory errors. Jamba, despite its hybrid nature, faces similar challenges due to its use of full attention mechanisms. Both Transformer and Jamba models exhibit limited extrapolation ability beyond their training context lengths. Mamba, while more efficient than Transformers and Jamba, still shows performance degradation for very long sequences.

Latency comparisons (Figure 1b) further highlight Taipan's exceptional efficiency. It demonstrates the lowest latency among all models, with linear scaling across sequence lengths. This contrasts with the quadratic scaling of Transformers and higher latency growth of Jamba. Notably, Taipan consistently outperforms Mamba-2, primarily due to its selective attention mechanism.

Params & Data	Model	Wino.	PIQA	Hella.	ARC$_E$	ARC$_C$	OB.	Truth.	RACE	BoolQ	Avg.
190M 27B	Transformer++	47.1	60.9	27.9	42.2	20.5	18.9	42.9	25.4	57.2	38.1
	Mamba	49.6	60.7	29.3	45.3	**21.8**	20.6	40.8	27.2	**59.3**	39.4
	Jamba	49.9	60.3	29.2	46.3	21.4	18.5	39.8	**27.4**	58.6	39.1
	Taipan	**51.0**	**62.6**	**29.4**	**46.7**	20.7	**21.8**	**41.1**	26.6	58.7	**39.9**
450M 100B	Transformer++	51.5	67.6	42.3	60.8	27.7	33.4	**39.2**	30.5	54.7	45.3
	Mamba	52.7	68.9	42.7	61.4	27.1	34.0	38.5	29.3	53.2	45.3
	Jamba	**53.1**	69.3	44.3	62.6	28.7	34.4	37.5	31.3	55.7	46.3
	Taipan	53.0	**69.6**	**46.6**	**65.6**	**32.9**	**36.6**	38.6	**30.7**	**60.4**	**48.2**
1.3B 100B	Transformer++	53.8	71.6	53.8	63.2	36.3	36.4	**44.0**	31.2	59.4	49.9
	Mamba	55.2	73.0	55.6	70.7	38.0	39.0	39.9	32.0	**61.8**	51.7
	Jamba	54.7	73.8	55.8	69.7	37.6	41.8	40.4	32.8	59.2	51.8
	Taipan	**57.0**	**74.9**	**57.9**	**71.2**	**39.3**	**40.4**	43.0	**34.4**	61.5	**53.3**

Table 5: Zero shot results of Taipan against baseline models.

Params & Data	Model	SWDE	FDA	SQuAD	Avg.
450M	Transformer++	**43.0**	**48.7**	**18.1**	**36.6**
	Mamba	27.9	9.8	12.5	16.7
	Jamba	35.4	36.6	16.3	29.4
	Taipan	*41.4*	*39.6*	*17.8*	*32.9*
1.3B	Transformer++	**64.2**	**64.5**	**41.2**	**56.6**
	Mamba	48.6	32.3	31.2	37.4
	Jamba	56.4	49.7	33.4	46.5
	Taipan	*61.5*	*59.7*	*36.9*	*52.7*

Table 6: Performance on in-context retrieval tasks.

3.5 Ablation Study

We conducted a comprehensive ablation study to investigate the effect of the two key components in Taipan's architecture, i.e., the attention budget capacity C and the inclusion of Positional Embeddings in the SALs, on its performance and efficacy.

3.5.1 Effect of Attention Budget Capacity

Our first experiment aimed to determine the optimal value of Capacity C that would maintain computational efficiency while maximizing performance on downstream tasks. We trained multiple variants of Taipan, each with 1.3B parameters, using different Capacity C values: $[0.10, 0.15, 0.20, 0.25]$. Each variant was trained for $24,000$ steps, allowing us to observe both the immediate impact of different C values and their effect on model performance over time.

We evaluated the performance of each variant at regular intervals on two representative tasks: SWDE (Arora et al., 2024) (for structured information extraction) and HellaSwag (Zellers et al., 2019) (for commonsense reasoning). These tasks were chosen to assess both the model's ability to handle long-context retrieval and its general language understanding capabilities.

As illustrated in Figure 4, Taipan achieves optimal performance with a Capacity $C = 0.15$. We observed that increasing C beyond 0.15 does not lead to significant improvements in results while increasing computational costs. Conversely, reducing C below 0.15 resulted in a noticeable drop in performance on tasks requiring precise in-context retrieval or complex long-range dependencies. These findings support our hypothesis that computational demands vary across tokens, with many adequately represented by Mamba's Markovian structure without requiring attention mechanisms.

By selectively applying attention only to tokens that benefit from it, Taipan optimizes resource allocation, enabling high performance while improving computational efficiency.

3.5.2 Impact of Positional Embeddings

Our second experiment investigated the impact of Positional Embeddings in Taipan's Attention mechanism, focusing on the model's ability to handle and generalize to various context lengths. We

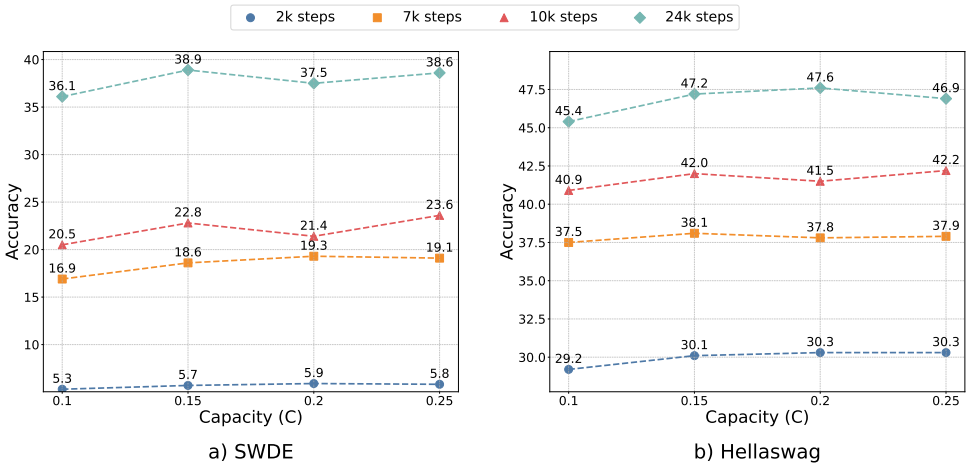
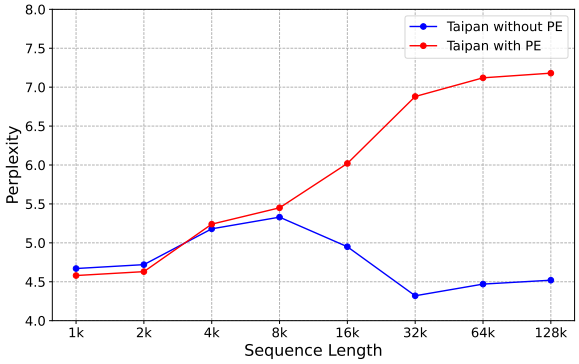


Figure 4: Effect of Attention Budget Capacity C on Taipan's Performance

trained two variants of the 1.3B parameter Taipan model for $24,000$ steps with a fixed context length of 4096 tokens. One variant incorporates Rotary Positional Embeddings (Su et al., 2024) in the Selective Attention layers, while the other excludes them. Figure 3.5.2 illustrates the performance of both variants in terms of perplexity across different context lengths.

The results reveal that Taipan without Positional Embeddings performs superiorly in generalizing context lengths beyond the training context. Both variants show comparable performance for sequences similar to or shorter than the training context length. However, as the sequence length increases, the performance gap between the two variants widens, with Taipan without Positional Embeddings maintaining lower perplexity scores. This suggests that the absence of Positional Embeddings enables more robust scaling to longer sequences. We attribute this improved generalization to the model's increased reliance on attention representation rather than positional biases.



4 Conclusion

The challenge of efficiently processing and understanding truly long textual contexts is a major hurdle for LLMs. While our state-of-the-art method for Document-level Event Argument Extraction (EAE) achieved strong performance on standard document benchmarks, it highlighted the fundamental limitation imposed by the fixed context windows of current LLM architectures. This bottleneck prevents scaling document analysis to very large documents, motivating our focus on core architectural improvements.

To address this, we developed **Taipan**, a novel hybrid LLM architecture for efficient and expressive long-context modeling. Taipan combines the linear efficiency of Mamba-2 with Selective Attention Layers that dynamically apply attention only to critical tokens. This design effectively balances computational cost with the ability to capture long-range dependencies.

Our experiments demonstrate Taipan's superior performance on long-context retrieval and extrapolation tasks, successfully enabling efficient modeling and retrieval across context lengths up to one million tokens. By achieving this scale, Taipan represents a significant step towards answering Q1 and simultaneously paves the way for tackling tasks like Document-level IE (Q2) on unprecedentedly large documents, overcoming the context limitations faced by previous methods.

5 Future Directions

This work has addressed the challenges of long-context modeling and document-level information extraction by proposing new architectural innova-

tions and demonstrating their utility in key applications such as event argument extraction. However, several important research directions remain open and will form the foundation of future work.

First, while Taipan shows that efficient long-context modeling is possible through selective attention and hybrid state-space mechanisms, further exploration is needed to generalize these capabilities to even more complex reasoning settings. One direction involves incorporating structured memory into the model, allowing it to maintain persistent, queryable state across documents or sessions. Such mechanisms could enable models to scale beyond single document understanding and support cross-document synthesis.

Second, future research will continue to refine document-level information extraction, extending beyond event argument extraction to more comprehensive document understanding tasks. These include event coreference resolution, temporal relation modeling, discourse-aware summarization, and narrative structure prediction. Achieving high performance in these tasks requires not only improvements in underlying model capacity but also new training paradigms that emphasize document structure and generalization to low-resource domains. In particular, exploring how LLMs can learn to reason over full-document and multi-document inputs without relying on heuristic chunking will be a central challenge.

Finally, a broader goal of this research is to build models that are not only efficient and accurate but also interpretable and adaptable to real-world applications. Future work will also explore human-in-the-loop training, enabling expert users to guide model attention or memory retention.

References

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. 2024. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*.

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

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Xinya Du and Heng Ji. 2022. Retrieval-augmented generative question answering for event argument extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xinya Du, Sha Li, and Heng Ji. 2022. Dynamic global memory for document-level argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5264–5275, Dublin, Ireland. Association for Computational Linguistics.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa,

Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

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

Albert Gu, Karan Goel, and Christopher Ré. 2021a. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021b. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. 2023. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5961–5971.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Jian Liu, Yufeng Chen, and Jinan Xu. 2021a. Machine reading comprehension as data augmentation: A case study on implicit event argument extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2725, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *ArXiv*, abs/2103.10385.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6759–6774, Dublin, Ireland. Association for Computational Linguistics.

William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 300–309, San Diego, California. Association for Computational Linguistics.

Trung Minh Nguyen and Thien Huu Nguyen. 2018. One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR.

Amir Pouran Ben Veyseh, Javid Ebrahimi, Franck Dernoncourt, and Thien Nguyen. 2022. MEE: A novel multilingual event extraction dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9603–9613, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*.

Pranav Rajpurkar, Jian Zhang, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL 2018*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurNIPS*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *6th International Conference on Learning Representations*.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. 2024. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*.

Kaiwen Wei, Xian Sun, Zequn Zhang, Jingyuan Zhang, Guo Zhi, and Li Jin. 2021. Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4672–4682, Online. Association for Computational Linguistics.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. 2024. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. *arXiv preprint arXiv:2402.18510*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qi Zeng, Qiusi Zhan, and Heng Ji. 2022. EA^{2}E: Improving consistency with event awareness for document-level argument extraction. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2649–2655, Seattle, United States. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. 2019. Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the International Conference on Machine Learning (ICML)*.