Chunk, Align, Select: A Simple Long-sequence Processing Method for Transformers

Jiawen Xie[12*], **Pengyu Cheng**[1†], **Xiao Liang**[13], **Yong Dai**[1], **Nan Du**[1]
[1]Tencent AI Lab
[2]Shanghai Jiao Tong University
[3]Tsinghua University

Abstract

Although dominant in natural language processing, transformer-based models still struggle with long-sequence processing, due to the computational costs of their self-attention operations, which increase exponentially as the length of the input sequence grows. To address this challenge, we propose a **Sim**ple framework to enhance the long-content processing of off-the-shelf pre-trained transformers via three steps: **C**hunk, **A**lign, and **S**elect (SimCAS). More specifically, we first divide each long-sequence input into a batch of chunks, then align the inter-chunk information during the encoding steps, and finally, select the most representative hidden states from the encoder for the decoding process. With our SimCAS, the computation and memory costs can be reduced to linear complexity. In experiments, we demonstrate the effectiveness of the proposed method on various real-world long-text summarization and reading comprehension tasks, in which SimCAS significantly outperforms prior long-sequence processing baselines. The code is at https://github.com/xjw-nlp/SimCAS.

1 Introduction

Transformers (Vaswani et al., 2017) have become a fundamental model architecture for sequential data modeling (Sun et al., 2019a; Dosovitskiy et al., 2021; Wysocki et al., 2023), especially in Natural Language Processing (NLP) (Devlin et al., 2018; Brown et al., 2020), where texts are regarded as sequences of tokens. Built with transformer blocks, pre-trained language models (PLMs) have recently shown astonishing empirical performance in various NLP tasks such as question answering (Yang et al., 2019), controllable generation (Byrne et al., 2021; Cheng and Li, 2022), summarization (Rush et al., 2015; Nallapati et al., 2016) and logic reasoning (Wei et al., 2022; Cheng et al., 2024a). However, one fatal weakness, that has hindered transformer-based models from being applied in broader application scenarios, is the quadratically raised computational consumption of self-attention operations when the input length increases. Hence, vanilla transformers have continuously been challenged by long-context tasks, such as machine reading comprehension (Kwiatkowski et al., 2019; Gong et al., 2020; Pang et al., 2022) and long-text summarization (Huang et al., 2021; Ma et al., 2022).

To enhance transformers with more efficient long-sequence processing, prior works focus on two perspectives, efficient attention operations (Beltagy et al., 2020; Zaheer et al., 2020; Choromanski et al., 2020) and sub-sequence processing (Liu et al., 2022). Efficient attention targets on reducing the memory and calculation cost of self-attention operations while preserving transformers' empirical performance on downstream tasks. Unfortunately, most efficient attention methods require customized self-attention implementations, which demand from-scratch training instead of being directly plugged into existing pre-trained models. Moreover, some empirical studies have demonstrated that efficient-attention methods inevitably sacrifice the short-sequence processing performance compared with full-attention models (Phang et al., 2022).

The alternative solution to long-sequence processing is to decompose the long-sequence input into multiple sub-sequences and then feed-forward each separately, known as sub-sequence processing (Moro et al., 2022). Although full-attention blocks are sufficiently utilized in each sub-sequence, these methods have been challenged to capture the semantic information across different sub-contexts. To solve this, some works assign the same fragment to different chunks (Ivgi et al., 2023), which however significantly increases the computational cost of each chunk and runs counter

[*]Work done during an internship at Tencent AI Lab.
[†]Corresponding author.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13500–13519

to the original efficiency goal.

To gather the advantages of the methods above, we introduce a **Sim**ple yet effective learning framework with three typical operations: **C**hunk, **A**lign, and **S**elect (SimCAS). In detail, SimCAS first chunks the input sequence into a batch of subsequences then feeds each subsequence forward the elaborately designed encoding blocks with an inter-chunk alignment mechanism. Finally, the most semantically representative hidden representations are selected via a tailored selection policy to compress the overall sequence length. To align the semantic information across sub-sequence chunks, we introduce a sequential batch alignment operation to calibrate the start and end token embeddings of each sub-sequence in the encoder layers. To learn the selector policy, inspired by the recent success of reinforcement learning in NLP (Ouyang et al., 2022), we adopt the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm with the decoder treated as the environment. Moreover, we use the attention scores and output likelihood to calculate effective rewards for the selector's policy optimization. To verify the effectiveness of SimCA, we conducted comprehensive experiments on seven long-context datasets from the domains of document-level summarization, multi-document summarization, and reading comprehension. The empirical results show that SimCAS outperforms other long-sequence processing baselines with high-level scalability. The main contributions of this paper are:

- We propose a simple yet effective long-sequence processing framework, which noticeably extends the application range of existing full-attention PLMs. Unlike prior works compromising the performance on short sequences, our method maintains satisfying performances processing either short or long contexts.
- We discover that transformers can be conceptualized as simulation environments for policy learning. We leverage transformers' attention scores and output likelihoods to optimize the selector policy to compress the long-sequence information. The optimized selector policy meanwhile facilitates the transformer to concentrate more on hidden states with high semantic importance.
- We conduct comprehensive experiments illustrating that SimCAS consistently surpasses previous baselines. Furthermore, we provide model scaling, efficiency comparisons, and ablation studies to substantiate the superior performance of our proposed method.

2 Background

Language Modeling The training objective for sequence generation consists of a sequence of token decisions made in an auto-regressive manner. This is formulated as a product of decision probabilities corresponding to specific tokens. Given an input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$ and its corresponding output $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_M)$, we model the following conditional probability:

$$p_\phi(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{m=1}^{M} p_\phi(\boldsymbol{y}_m|\boldsymbol{Y}_{<m}, \boldsymbol{X}), \quad (1)$$

where $\boldsymbol{Y}_{<m} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{m-1})$, and ϕ represents the model parameters.

Proximal Policy Optimization In the domain of reinforcement learning (RL), Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a widely used policy gradient method (Kakade, 2001) for its remarkable performance and efficiency in solving complex control and decision-making tasks (Vinyals et al., 2019; Akkaya et al., 2019; Cheng et al., 2024b). The vanilla policy gradient estimator has the form: $\nabla_\theta \mathbb{E}_{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}[A_t^\pi(\boldsymbol{a}_t, \boldsymbol{s}_t)] \approx \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)\hat{A}_t]$, where $\boldsymbol{s}_t \in \mathcal{S}$ is the state at t-step, $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is a stochastic policy acting $\boldsymbol{a}_t$ at $\boldsymbol{s}_t$, $\hat{A}_t$ is the estimated value of the advantage function $A_t^\pi(\boldsymbol{a}_t, \boldsymbol{s}_t)$, and $\hat{\mathbb{E}}_t$ denotes the empirical average over a sample batch. The PPO algorithm improves the training stability of the policy gradient, by optimizing the following objective:

$$L = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)], \quad (2)$$

where $r_t(\theta) = \frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}$ is the probability ratio between new and old policies, and $\varepsilon > 0$ is a hyper-parameter for clipping.

3 Methodology

Given a long-input text $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ with a fairly large input length N, we aim to design a model $p_\phi(\boldsymbol{Y}|\boldsymbol{X})$ to predict a corresponding label sequence $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M)$, where $\boldsymbol{Y}$ can either be classification labels or output sequence tokens. The major difficulty of the task is that the input length N is so large that the original self-attention operations become infeasible with the quadratic complexity $\mathcal{O}(N^2)$.

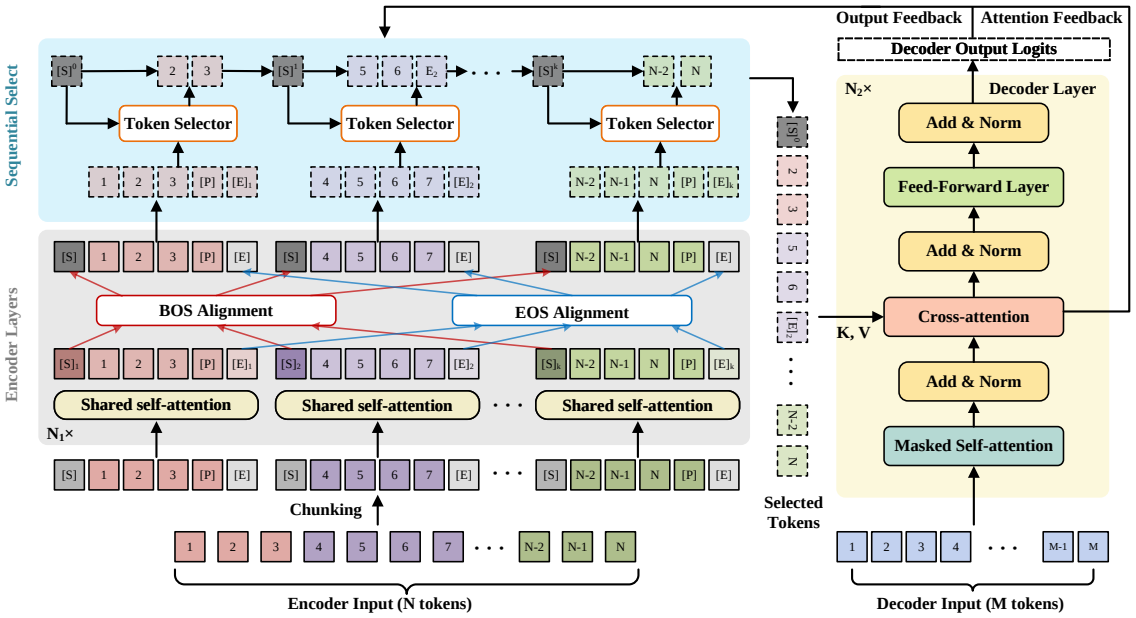

Figure 1: The SimCAS framework: The long inputs are first divided into a batch of chunks, each of which is filled with start token [S], padding token [P] and end token [E]. Then the inter-chunk information can be transferred via the alignment of [S] and [E] representations after each encoder layer. Next, hidden states are selected for decoding steps. The decoder output logits and attention scores are utilized as rewards for updating the token selector.

To address the challenge, we propose a novel method that intuitively splits the long inputs into chunks with feasible lengths, then selects the most representative tokens for decoding steps. To guarantee inter-chunk semantic information extracted during encoding, we design an aligning scheme in the encoder blocks. In the following, we will describe the chunking scheme, the aligning strategy, and the selector design in detail.

3.1 Chunking Scheme

Assume the maximum input sequence length of a pre-trained transformer model is S. We first split the long input sequence into $B = \lceil \frac{N}{S} \rceil$ chunks with each chunk length equal to S. To simplify the expression, we assume that a sentence is in only one chunk. In the practical experiment, we use the sentence-level segmentation (Moro and Ragazzi, 2022) to acquire a sequence of sentences, and then employ the greedy method to allocate sequentially these sentences to the chunks. If there is not enough space in the chunk for the current sentence, we will use the padding tokens to fill the chunk. Mathematically, these divided segments can be represented as:

$$\left\{(\boldsymbol{c}_1^k, \boldsymbol{c}_2^k, \ldots, \boldsymbol{c}_S^k)\right\}_{k=1}^{B}, \tag{3}$$

where $\boldsymbol{c}_i^k = \boldsymbol{x}_{(k-1)S+i}$, and $\lceil \cdot \rceil$ is the ceiling function. Since the chunk might not have S tokens from $\boldsymbol{X}$, we append special padding tokens at the end (see Figure 1). After chunking, we add the start token ([S]) and end token ([E]) to each chunk, and treat the chunks as a normal transformer input batch $\mathcal{C}$ with batch size B:

$$\mathcal{C} = \left\{([\texttt{S}], \boldsymbol{c}_1^k, \boldsymbol{c}_2^k \ldots, \boldsymbol{c}_S^k, [\texttt{E}])\right\}_{k=1}^{B}. \tag{4}$$

3.2 Sequential Batch Alignment

After chunking the input text into a standard token batch, we can encode it with the transformer encoder layers. As in Figure 1, we assume the encoder has N_1 layers. Denote the hidden representations of k-th chunk in $\mathcal{C}$ (in equation 4) at l-th encoder layer as $\boldsymbol{H}^{k,l} = (\boldsymbol{h}_0^{k,l}, \boldsymbol{h}_1^{k,l}, \ldots, \boldsymbol{h}_S^{k,l}, \boldsymbol{h}_{S+1}^{k,l})$, where $\boldsymbol{h}_0^{k,l}$ and $\boldsymbol{h}_{S+1}^{k,l}$ are hidden representation of [S] and [E] tokens respectively, and $\boldsymbol{h}_i^{k,l}$ is the embedding for $\boldsymbol{c}_i^k$ with $1 \le i \le S$.

As mentioned in Section 1, chunking methods make it difficult to capture the inter-chunk semantic information. Inspired by recent work using special tokens for full sequence information (Mohtashami and Jaggi, 2023), we align the information of [S] and [E] tokens at each encoding block. More specifically, at l-th layer, our batch alignment average the hidden states of [S] and [E] of all chunks:

$$\bar{\boldsymbol{h}}_{\text{BOS}}^l = \frac{1}{B}\sum_{k=1}^{B} \boldsymbol{h}_0^{k,l}, \quad \bar{\boldsymbol{h}}_{\text{EOS}}^l = \frac{1}{B}\sum_{k=1}^{B} \boldsymbol{h}_{S+1}^{k,l}. \tag{5}$$

Then we replace $\boldsymbol{h}_0^{k,l}$ and $\boldsymbol{h}_{S+1}^{k,l}$ with the aligned $\bar{\boldsymbol{h}}_{\text{BOS}}^l$ and $\bar{\boldsymbol{h}}_{\text{EOS}}^l$ into the hidden states for the next-layer encoding block, as shown in Figure 1.

3.3 Token Selector

After being encoded into the last hidden space with our sequential batch aligning scheme, the chunks should be reformed back to a sequence for the next decoding steps. Directly tiling the chunks' representations back into a sequence is still infeasible because the overlong sequence makes it difficult to fuse information at the decoding stage. Therefore, we propose a token selector to select the most representative hidden representations for decoding steps. Inspired by Ramamurthy et al. (2023), we design the selector from the perspective of reinforcement learning (RL).

Selection Module Design Formally, the token selector takes the last hidden states $\boldsymbol{H}^L = \{\boldsymbol{h}_0^{k,L}, \boldsymbol{h}_1^{k,L}, \ldots, \boldsymbol{h}_S^{k,L}, \boldsymbol{h}_{S+1}^{k,L}\}_{k=1}^B$ and selection actions $\boldsymbol{A}_t = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{t-1})$ as inputs and predicts the next selection action $\boldsymbol{a}_t$, where each action $\boldsymbol{a}_t$ has two values *"select"* and *"skip"* for operating the t-th token $\boldsymbol{x}_t$ in $\boldsymbol{X}$. We set the state of RL as $\boldsymbol{s}_t = (\boldsymbol{H}^L, \boldsymbol{A}_t)$, then the selector is a policy $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$ to predict next action $\boldsymbol{a}_t$.

We implement the selector with the actor-critic style (Konda and Tsitsiklis, 1999). Both actor and critic are simple feed-forward networks, but the actor outputs a probability distribution over the action space and the critic outputs a single scalar value. At state $\boldsymbol{s}_t$, to consider the sequential effect of previous action $\boldsymbol{A}_t$, we first take the average of all selected hidden states as

$$\bar{\boldsymbol{h}}_t = \frac{\sum_{k,i} \mathbb{I}_{\{\boldsymbol{a}_j = \textit{"select"}, j<t\}} \boldsymbol{h}_i^{k,L}}{\sum_{k,i} \mathbb{I}_{\{\boldsymbol{a}_j = \textit{"select"}, j<t\}}}, \tag{6}$$

where $j = (k-1)S + i$ maps chunk indices back to the input sequence, and $\mathbb{I}_{\{\cdot\}}$ is the indicator function. Then we concatenate the current selector state $\bar{\boldsymbol{h}}_t$ and token information $\boldsymbol{h}_i^{k,L}$, $t = (k-1)S + i$, to predict next action $\boldsymbol{a}_t$ via the actor:

$$\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) = \text{actor}(\bar{\boldsymbol{h}}_t, \boldsymbol{h}_i^{k,L}). \tag{7}$$

Reward Design To train the selector within an RL scheme, we treat the transformer as an environment and design action rewards. Inspired by Yuan et al. (2021), we can directly utilize the language modeling likelihood as the generation quality reward for selection actions:

$$R_{\text{LM}} = \xi \exp\{\frac{1}{M} \log p_\phi(\boldsymbol{Y}|\boldsymbol{X})\}, \tag{8}$$

where ξ is a coefficient that magnifies the value of the reward for easier optimization of the selector. However, R_{LM} is only a scalar value, which cannot provide fine-grained guidance to the selector. Therefore, we use the input-output cross-attention scores to calibrate R_{LM}. More specifically, we denote the cross-attention matrix of the q-th attention head in the l-th layer of the decoder as $\boldsymbol{A}_q^l \in \mathbb{R}^{M \times N}$, and the overall cross-attention

$$\bar{\boldsymbol{A}} = \frac{1}{N_2 \cdot Q} \sum_{l=1}^{N_2} \sum_{q=1}^{Q} \boldsymbol{A}_q^l, \tag{9}$$

where N_2 is the number of decoder layers, and Q is the number of cross-attention heads. With the overall feedback R_{LM} and cross-attention $\bar{\boldsymbol{A}} = (\bar{a}_{ij})_{M \times N}$, we adjust the reward R_j^+ to each selected tokens. For *"skip"* action, we intend to limit the selected sequence length. Assume the number of overall input tokens and selected token representations is L_{all} and L_{select} respectively. We set a hyper-parameter L_{hyper} to control the size of L_{select} with the skipping reward R^-:

$$\begin{aligned} R_j^+ &= \frac{\bar{a}_j}{1 - \bar{a}_0} R_{\text{LM}}, \bar{a}_j = \frac{1}{M} \sum_{i=1}^{M} \bar{a}_{ij}, \\ R^- &= \begin{cases} \frac{R_{\text{LM}}}{L_{\text{all}}} & \text{if } L_{\text{select}} < L_{\text{hyper}} \\ \frac{R_{\text{LM}}}{L_{\text{select}}} & \text{otherwise.} \end{cases} \end{aligned} \tag{10}$$

With the selector and rewards designed above, we can optimize the selector with the PPO algorithm described in Section 2. Note that in our setups, the environment (the transformer) changes during the training steps. Therefore, we alternatively update the selector and transformer: in each interaction, we first fix the transformer and use the reward R_{LM} and cross-attention scores to update the selector, then fix the selector and update the transformer with language modeling loss. In addition, the selector is similar to the chunk-wise RNN, so the time overhead of the selection process is low.

4 Related Work

Efficient Transformers The attention mechanism in transformers requires quadratically increased computational complexity with respect to the input sequence length, which limits the application scenarios, especially for long-text processing. To address this issue, various previous works have been proposed for designing more efficient attention operations (Tay et al., 2023; Fournier et al.,

2023). Longformer (Beltagy et al., 2020), BIGBIRD (Zaheer et al., 2020), GMAT (Gupta and Berant, 2020), and ETC (Ainslie et al., 2020) reduce the memory consumption of dense attentions by elaborate combinations of global attention and local attention mechanisms. LongT5 (Guo et al., 2022) is based on the original T5 with global-local attention sparsity patterns to handle long input and has an additional pre-training phase on C4 dataset. CoLT5 (Ainslie et al., 2023) is an improved version of LongT5, with the same parameter size and pre-training dataset as LongT5, and it optimizes the computation of the attention module and the feed-forward module and proposes a new pre-training objective. Additionally, Reformer (Kitaev et al., 2020) leverages a locality-sensitive hashing to the attention mechanism, changing its complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$, where n is the input text sequence length. Routing Transformer (Roy et al., 2021) applies a sparse routing module based on online k-means to self-attention while reducing the overall complexity of attention. Approximation-based methods, such as Performers (Choromanski et al., 2020) and RFA (Peng et al., 2021), use linear space and time complexity to estimate the attention matrix based on random features. Luna (Ma et al., 2021) attends only to a fixed number of hidden vectors. Linformer (Wang et al., 2020) calculates self-attention by a low-rank matrix. However, the vast majority of these methods are difficult to apply to existing PLMs. Moreover, Xiong et al. (2022) proposes that many efficient-attention transformers do not even perform as well as simple local-attention models on downstream language tasks.

Chunking Methods for Long Sequence Another solution for long-sequence processing is to perform sequence chunking and then process them respectively (Zhong et al., 2022; Wang et al., 2023). Among chunking methods, SLED (Ivgi et al., 2023) splits the long sequence into overlapping chunks and processes each chunk with the encoder, then fuses cross-chunk information with the decoder. PageSum (Liu et al., 2022) separates the long sequence into non-overlapping chunks and effectively tackles them by the principle of locality (Denning, 2005). Unlimiformer (Bertsch et al., 2023) encodes long inputs in chunks and utilizes only the top-k input tokens for every attention head.

Length Extrapolation Length extrapolation in transformers refers to their ability to handle input sequences of varying lengths during both training and inference (Press et al., 2022; Sun et al., 2023). Transformers use a self-attention mechanism to capture dependencies across positions in a sequence, allowing them to generalize well to sequences of different lengths. This flexibility is essential for tasks like NLP and time series analysis, where input lengths can vary.

Sequence Length Reduction Reducing the length of hidden states (Guan et al., 2022; Kim et al., 2022) is the method of model compression from the width perspective, which is promising since some studies showed that there is redundant encoded information in token representations (Ethayarajh, 2019; Klafka and Ettinger, 2020). Among the redundancy, some tokens carry more task-specific information than others, suggesting that these tokens are more salient and imperative to be selected to feed into subsequent operations. Compared with model compression via layer-wise pruning, token-level pruning does not come at the expense of model performance in complex reasoning (Sanh et al., 2019; Sun et al., 2019b).

5 Experiments

To evaluate the performance of SimCAS, we conduct experiments on the long-text abstractive summarization and machine reading comprehension tasks. In the following, we introduce the information about the datasets, baselines, model implementations, and evaluation results of our experiments.

5.1 Datasets

We conduct experiments on two types of NLP tasks: long-text summarization and machine reading comprehension. For long-text summarization, we use four single-document summarization datasets: arXiv, PubMed (Cohan et al., 2018), GovReport (Huang et al., 2021), SummScreen (Chen et al., 2022), and two multi-document summarization datasets: Multi-News (Fabbri et al., 2019b), WCEP (Gholipour Ghalandari et al., 2020). For the reading comprehension task, we test on the NarrativeQA (Kočiský et al., 2018) dataset. We list the introduction about these datasets below. More dataset details can be found in Appendix § A.

arXiv & PubMed[1] are two long-document summarization datasets in the scientific research domain. Each document is a scientific paper whose summary is the corresponding abstract.

[1] https://github.com/armancohan/long-summarization

Base Model	Method	arXiv				PubMed			
		R-1	R-2	R-L	BS	R-1	R-2	R-L	BS
LED_{large}	Standard	46.63	19.62	41.83	-	-	-	-	-
LED_{large}	PRIMERA	47.60	**20.80**	42.60	-	-	-	-	-
$PEGASUS_{large}$	Standard	44.21	16.95	38.83	-	45.97	20.15	41.34	-
$PEGASUS_{large}$	BIGBIRD	46.63	19.02	41.77	-	46.32	20.65	42.33	-
$BART_{large}$	HEPOS	47.87	20.00	41.50	-	47.93	20.74	42.58	-
$BART_{base}$	Standard	40.36	13.78	36.11	59.44	40.36	13.29	35.02	61.77
$BART_{large}$	Standard	42.97	15.54	37.02	61.18	42.87	15.44	36.93	63.08
$BART_{base}$	SimCAS	47.22	19.35	42.25	63.51	48.17	21.11	43.90	66.33
$BART_{large}$	SimCAS	**48.14**	19.77	**42.93**	**63.78**	**48.65**	**21.40**	**44.14**	**66.52**

Table 1: Average results on arXiv and PubMed test sets. R-1/2/L is the ROUGE-1/2/L F1 score. BS refers to the neural model-based metrics BERTScore. Bold (underline) are used to denote the best (second-best) metric.

GovReport[2] is a long-document summarization dataset based on reports published by the U.S. Government Accountability Office and Congressional Research Service.

SummScreen[3], which includes TV series transcripts, often indirectly presents plot details through character dialogues scattered throughout the transcript. These details need to be consolidated to create concise plot descriptions.

Multi-News[4] is a large-scale multi-document summarization dataset. It consists of news articles and human-written summaries of these articles. Each summary is professionally written by editors and with links to the original articles cited.

WCEP[5] is a dataset for multi-document summarization (MDS). It contains short, human-written summaries about news events, obtained from the Wikipedia Current Events Portal (WCEP).

NarrativeQA[6] is a reading comprehension dataset over entire books from Project Gutenberg and movie scripts from different websites.

5.2 Baselines

There are several baselines for comparison: among them, those based on the full-attention mechanism are HiMAP (Fabbri et al., 2019a), BERTREG (Gholipour Ghalandari et al., 2020), Submodular+Abs (Gholipour Ghalandari et al., 2020), BART (Lewis et al., 2020), PEGASUS (Zhang et al., 2020), DynE (Hokamp et al., 2020), GraphSum (Li et al., 2020), BART-Long-Graph (Pasunuru et al., 2021), SLED (Ivgi et al., 2023), Memorizing Transformers (Wu et al., 2022), Unlimiformer (Bertsch et al., 2023).

Moreover, several baselines introduce the tailored sparse attention to address longer sequences such as LED (Longformer Encoder-Decoder) (Beltagy et al., 2020), BIGBIRD (Zaheer et al., 2020), PRIMERA (Xiao et al., 2022), HEPOS (Huang et al., 2021) and LED+RELAX (Parnell et al., 2022). More details of baselines can be found in Appendix §C.

	R-1	R-2	R-L	BS
R-1	1.000	0.872	0.913	0.810
R-2	0.872	1.000	0.898	0.794
R-L	0.913	0.898	1.000	0.825
BS	0.810	0.794	0.825	1.000

Table 2: **Pearson correlation coefficient** between the four automatic evaluation metrics (R-1, R-2, R-L, BS) used for a base BART with beam search on **GovReport**.

5.3 Implementation Details

Our implementation is based on *PyTorch* (Paszke et al., 2019) and *Transformers* (Wolf et al., 2020) libraries. We train our model by using 8 NVIDIA V100 32G GPUs.

During the training phase, the maximum input lengths for $BART_{large}$ and $BART_{base}$ are set to 8192 and 16384, respectively, unless otherwise specified. To ensure efficient training, we update the parameters of the original backbone and the selector alternately. The reward estimation for each action is computed based on decoder cross-attention and the output feedback of the generative model. This estimation process is detached from the computation graph and does not participate in backpropagation.

At the inference stage, compared to the original generation process, our framework only adds a chunk-wise selection procedure between the en-

[2] https://github.com/luyang-huang96/LongDocSum
[3] https://github.com/mingdachen/SummScreen
[4] https://github.com/Alex-Fabbri/Multi-News
[5] https://github.com/allenai/PRIMER
[6] https://github.com/deepmind/narrativeqa

Base Model	Method	GovReport				SummScreen			
		R-1	R-2	R-L	BS	R-1	R-2	R-L	BS
$BART_{base}$	SLED	54.70	24.40	25.40	-	32.70	7.90	19.10	-
$BART_{large}$	SLED	57.50	26.30	27.40	-	35.20	8.70	19.40	-
LED_{large}	PRIMERA	55.10	23.90	25.90	67.00	32.30	7.10	18.30	57.10
$BART_{base}$	Memorizing	55.20	25.10	26.40	67.50	32.70	7.40	19.20	57.40
$BART_{base}$	Unlimiformer	56.60	26.30	27.60	68.20	34.70	8.50	**19.90**	58.50
PRIMERA	Unlimiformer	57.40	26.20	**28.00**	68.10	33.30	7.60	18.90	57.70
$BART_{base}$	Standard	51.72	19.37	23.11	64.12	29.73	5.23	15.65	54.30
$BART_{large}$	Standard	52.94	19.78	23.71	64.44	29.89	5.32	15.71	54.43
$BART_{base}$	SimCAS	59.30	25.95	27.07	68.17	43.45	12.74	18.38	62.46
$BART_{large}$	SimCAS	**60.29**	**26.68**	27.97	**68.64**	**44.15**	**13.42**	18.50	**62.82**

Table 3: Average results on GovReport and SummScreen test sets. R-1/2/L is the ROUGE-1/2/L F1 score. BS refers to the neural model-based metrics BERTScore. The best and second-best results are bolded and underlined respectively.

System	R-1	R-2	R-L	BS
$BART_{large}$	42.04	14.88	23.34	-
HiMAP	44.17	16.05	21.38	-
GraphSum	45.87	17.56	23.39	-
BART-Long-Graph	49.24	18.99	23.97	-
LED+RELAX	47.23	18.86	25.03	-
PRIMERA	**49.94**	**21.05**	25.85	-
$BART_{base}$	40.54	12.18	22.39	58.75
$BART_{large}$	42.16	14.69	23.51	60.70
$BART_{base}$+SimCAS	48.88	20.01	25.32	65.05
$BART_{large}$+SimCAS	49.40	20.47	**25.96**	**65.40**

System	R-1	R-2	R-L	BS
BERTREG	35.00	13.50	25.50	-
SUBMODULAR+ABS	34.40	13.10	25.00	-
DynE	35.40	15.10	25.60	-
LED	39.79	18.94	32.10	-
LED+RELAX	41.11	19.46	33.13	-
PRIMERA	46.08	**25.21**	37.86	-
$BART_{base}$	36.12	13.77	29.98	60.84
$BART_{large}$	37.66	15.98	31.01	62.32
$BART_{base}$+SimCAS	45.68	22.80	37.71	70.59
$BART_{large}$+SimCAS	**46.29**	24.45	**38.61**	**71.38**

Table 4: Average results on Multi-News (left) and WCEP (right) test sets. R-1/2/L is the ROUGE-1/2/L F1 score. BS refers to the model-based metrics BERTScore. Bold (underline) are used to denote the best (second-best) metric.

Datasets	w/o Chunk	w/o Align	w/o Select	SimCAS	Δ_{Chunk}	Δ_{Align}	Δ_{Select}
arXiv	30.10 ± 0.73	36.12 ± 0.47	32.70 ± 0.62	**36.46 ± 0.56**	↑ 21.13%	↑ 0.94%	↑ 11.50%
PubMed	29.63 ± 0.51	37.20 ± 0.64	34.32 ± 0.59	**37.77 ± 0.51**	↑ 27.47%	↑ 1.53%	↑ 10.05%
GovReport	31.23 ± 0.50	37.12 ± 0.66	32.91 ± 0.81	**37.54 ± 0.48**	↑ 20.20%	↑ 1.13%	↑ 14.07%
SummScreen	19.71 ± 0.49	20.69 ± 0.45	19.63 ± 0.62	**25.15 ± 0.42**	↑ 27.60%	↑ 2.56%	↑ 08.10%
Multi-News	25.34 ± 0.38	31.45 ± 0.55	27.98 ± 0.35	**31.36 ± 0.51**	↑ 23.76%	↓ 0.29%	↑ 12.08%
WCEP	28.42 ± 0.72	35.01 ± 0.64	30.33 ± 0.76	**35.31 ± 0.59**	↑ 24.24%	↑ 0.86%	↑ 16.42%
NarrativeQA	21.70 ± 0.55	31.52 ± 0.46	23.20 ± 0.23	**31.76 ± 0.44**	↑ 46.36%	↑ 0.76%	↑ 36.90%

Table 5: Ablation study results on the development sets of all datasets. Performance changes compared with the full model ($BART_{base}$+SimCAS) are reported. The metrics of summarization and reading comprehension datasets are the average of ROUGE-1/2/L and F1 respectively.

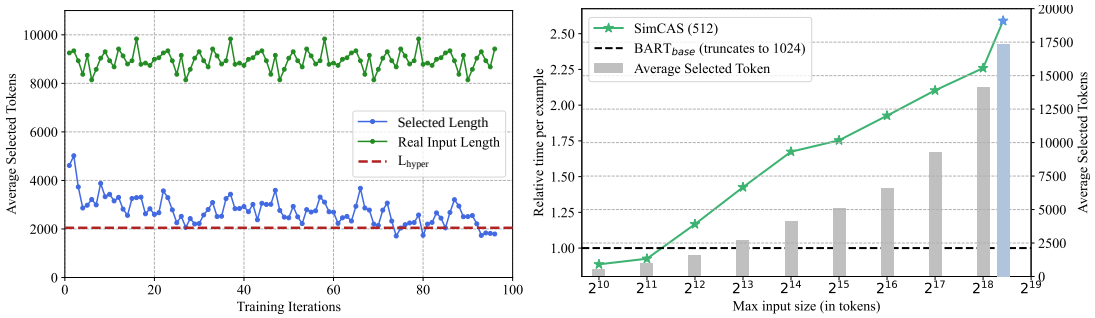

Figure 2: The left-hand-side plot shows the change of the actual number of tokens entered into the model and the number of tokens selected by our selector during training. The red dashed line represents the conditional boundary for the skipping reward. The right-hand-side plot shows the effect of increasing the number of input tokens in the inference phase on the time latency and the number of selected tokens. The area marked in blue on the right represents the limit of the number of tokens that the V100 can handle (350K tokens).

coder and the decoder, which takes very little time. At the decoding stage, the target sequence is generated with beam search in an auto-regressive manner (Wiseman and Rush, 2016).

5.4 Evaluations

Like most previous works, for abstractive summarization tasks, we measure the quality of generated summaries using the popular metric ROUGE (Lin, 2004). On the test set of arXiv, PubMed, GovReport, SummScreen, Multi-News, and WCEP, we report full-length F1-based ROUGE-1, ROUGE-2, and ROUGE-L scores computed with the standard ROUGE Perl package. Furthermore, we also use a popular model-based semantic metric BERTScore[7] (Zhang* et al., 2020) to demonstrate the superiority of our approaches comprehensively. As shown in Table 2, ROUGE-1 and ROUGE-L are strongly correlated (Pearson correlation score of 0.913). In particular, the relatively low Pearson correlation coefficient (< 0.8) between ROUGE-2 and BERTScore indicates a significant difference in preferences.

As for the reading comprehension task, we use the F1 and exact match (EM) metrics defined in SCROLLS (Shaham et al., 2022) to evaluate the model performance on the NarrativeQA dataset. Both metrics normalize the reference and system output strings via lowercasing, removing punctuation and stopwords, and normalizing whitespace.

6 Discussion

Main Result Anaylses Table 1 and 3 report results over four long-document test sets. We note that casting the backbone $\text{BART}_{\text{base}}$ into our SimCAS can significantly improve the model performance, and increasing the size of the model can further improve performance. Our approach outperforms baseline on several metrics and achieves new state-of-the-art performance on the PubMed test set. In Table 4, the results of multi-document summarization tasks have a similar trend. Apart from PRIMERA, which customizes a pre-training objective for multi-document summarization, our method significantly outperforms previous results. See Figure 3, we observe a substantial improvement in the performance of our model on the NarrativeQA test set compared to previous works. Given the fact that this dataset has an extremely long average input length ($\geq$100K) and short average output length ($\leq$10), we attribute this significant performance enhancement to our method's proficiency in filtering out an immense amount of task-irrelevant information. This, in turn, enables the decoding blocks to fuse information more efficiently.

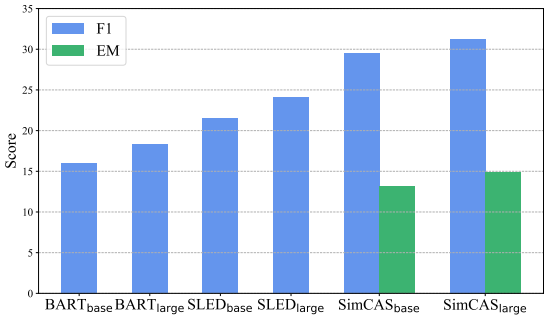

Figure 3: System performance comparison on NarrativeQA test set.

Ablation Study SimCAS is composed of three components: `Chunk`, `Align`, and `Select`. To investigate the contributions of each component, we independently removed each one. Notably, after removing `Chunk`, the maximum input length was restricted to 1024, causing `Align` to fail. As demonstrated in Table 5, the performance experienced a significant decline when either `Chunk` or `Select` was removed, underscoring their effectiveness. The `Align` component improved performance on all datasets, with the exception of the Multi-News dataset. We hypothesize that more sophisticated alignment methods could potentially enhance performance further.

Effectiveness of Conditional Skipping Reward To prevent our selector from choosing an excessive number of token embeddings, we implement a conditional reward mechanism for skipping actions, which imposes a soft constraint on the number of selected token embeddings. Figure 2 (left) demonstrates the effectiveness of this reward mechanism. As training progresses, the number of selected embeddings gradually converges to a threshold of 2048. This indicates that our selection process is effective in identifying essential token embeddings. Additionally, as the input sequence length increases, the ratio of selected token embeddings to the total length decreases. This is due to the length penalty in equation 10, which discourages the selection of an excessive number of token embeddings.

Increasing Maximum Input Length We explore the performance changes of our model on

[7] In order to make a fair comparison with the previous work, we also use checkpoint "facebook/bart-large-mnli" for BERTScore.

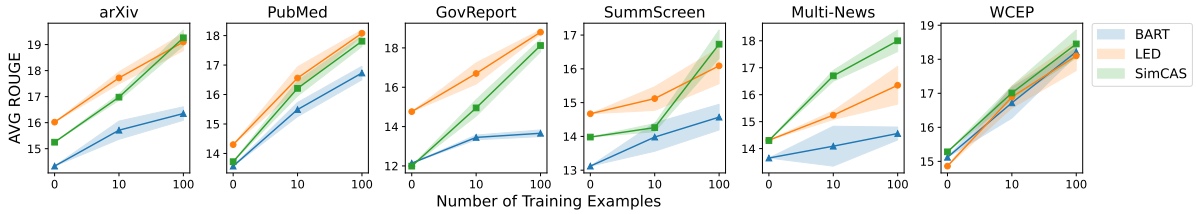

Figure 4: The AVG ROUGE scores (R-1, R-2, and R-L) of the pre-trained models ($\text{BART}_{\text{base}}$, LED_{base}, $\text{SimCAS}_{\text{base}}$) with 0, 10, and 100 training examples with variance. All results are obtained by the average of 5 random runs with different seeds.

a single 32G V100 GPU as the input sequence length increases during inference. Initially, we assess the efficiency of our framework in processing long sequences using $\text{BART}_{\text{base}}$ as the backbone. We depict the fluctuations in the ratio of the average time required by SimCAS to the time taken by BART (1024) in Figure 2 (right). Specifically, a line graph represents the relative time per sample, while a bar graph displays the number of token embeddings selected by SimCAS. When the input length is short ($< 2^{14}$), the time cost exhibits near-linear growth. We attribute this phenomenon to our method's significant reduction in computational complexity and the redundant computation resource for short sequences. However, with a dramatic increase in input length ($> 2^{17}$), the time cost rises sharply. As indicated in the blue area on the far right of the graph, our method can handle input sequences of up to 350k in length on a V100. Moreover, even as the number of input tokens increases significantly, the selected hidden states remain at a reasonable size, demonstrating our selector's capability to retrieve high-contribution information.

Low-Resource Evaluation In real-world applications, the quantity of training samples available for downstream tasks is frequently quite constrained. Consequently, the performance of the model under low-resource conditions is of paramount importance. In our setup, we randomly sample a few (10 and 100) training examples from specific datasets to adapt these PLMs such as $\text{BART}_{\text{base}}$ (1024), LED_{base} (4096), and $\text{SimCAS}_{\text{base}}$ (4096), for corresponding data distributions. The results in Figure 4 indicate that our model demonstrates superior sample efficiency in low-resource scenarios compared to previous robust baselines.

7 Conclusion and Future Work

In this paper, we introduced a simple method for long-text processing via chunking, aligning, and selecting, called SimCAS. We divided the input long sequence into chunks, and encoded them with sequential batch alignment to capture the inter-chunk semantics. To select the important token representations in the encoded output of the encoder, we introduced a reinforcement learning-based token selector with the PPO method. We leverage the transformer as an environment and design a reward scheme for the corresponding actions based on the output logits and decoder cross-attention feedback to optimize the hidden token selector. Substantial experiment results and analyses demonstrate the satisfying effectiveness of SimCAS. Besides, our method does not depend on any particular tasks or models, which have good generalization ability for various application scenarios.

For future work, our method can be naturally adopted into non-language long-sequence processing tasks, such as molecular structure analysis. Also, SimCAS has the potential to enhance the long text pre-training to transformers.

8 Limitations

Even though our framework significantly reduces the overhead of processing long sequences, due to memory constraints, the maximum input length on a single V100 32G GPU during training is 16k, and there is an extremely large consumption of GPU memory even if the batch size is small, which might suppress the potential of our method. In addition, we use small-scale PLMs with good performance as the backbone for downstream tasks. The effectiveness of our framework on large language models needs to be investigated further.

9 Ethical Considerations

Similar to existing methods, there is no guarantee that the generated content is factually consistent and free from hallucination (Maynez et al., 2020; Kang and Hashimoto, 2020). Therefore caution is imperative when applying our method to scenarios with high demand of generation accuracy.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. CoLT5: Faster long-range transformers with conditional computation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5100, Singapore. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. 2019. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Kale. 2021. TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems. 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 671–680, Online. Association for Computational Linguistics.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics.

Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Yong Dai, Lei Han, and Nan Du. 2024a. Self-playing adversarial language game enhances llm reasoning. *arXiv preprint arXiv:2404.10642*.

Pengyu Cheng and Ruineng Li. 2022. Replacing language model for style transfer. *arXiv preprint arXiv:2211.07343*.

Pengyu Cheng, Yifan Yang, Jian Li, Yong Dai, Tianhao Hu, Peixin Cao, Nan Du, and Xiaolong Li. 2024b. Adversarial preference optimization: Enhancing your alignment via rm-llm game. In *Findings of the Association for Computational Linguistics*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Peter J. Denning. 2005. The locality principle. *Commun. ACM*, 48(7):19–24.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019a. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019b. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2023. A practical survey on faster and lighter transformers. *ACM Comput. Surv.* Just Accepted.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6751–6761, Online. Association for Computational Linguistics.

Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. 2022. Transkimmer: Transformer learns to layer-wise skim. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7275–7286, Dublin, Ireland. Association for Computational Linguistics.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Ankit Gupta and Jonathan Berant. 2020. Gmat: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274*.

Chris Hokamp, Demian Gholipour Ghalandari, Nghia The Pham, and John Glover. 2020. Dyne: Dynamic ensemble decoding for multi-document summarization.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Sham M Kakade. 2001. A natural policy gradient. *Advances in neural information processing systems*, 14.

Daniel Kang and Tatsunori B. Hashimoto. 2020. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 718–731, Online. Association for Computational Linguistics.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Josef Klafka and Allyson Ettinger. 2020. Spying on your neighbors: Fine-grained probing of contextual embeddings for information about surrounding words. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4801–4811, Online. Association for Computational Linguistics.

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph

to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yixin Liu, Ansong Ni, Linyong Nan, Budhaditya Deb, Chenguang Zhu, Ahmed Hassan Awadallah, and Dragomir Radev. 2022. Leveraging locality in abstractive text summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6081–6093, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Congbo Ma, Wei Emma Zhang, Mingyu Guo, Hu Wang, and Quan Z. Sheng. 2022. Multi-document summarization via deep learning techniques: A survey. *ACM Comput. Surv.*, 55(5).

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. In *Advances in Neural Information Processing Systems*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

Gianluca Moro and Luca Ragazzi. 2022. Semantic self-segmentation for abstractive summarization of long documents in low-resource regimes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11085–11093.

Gianluca Moro, Luca Ragazzi, Lorenzo Valgimigli, and Davide Freddi. 2022. Discriminative marginalized probabilistic neural method for multi-document summarization of medical literature. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 180–189, Dublin, Ireland. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States. Association for Computational Linguistics.

Jacob Parnell, Inigo Jauregi Unanue, and Massimo Piccardi. 2022. A multi-document coverage reward for relaxed multi-document summarization.

Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. 2021. Efficiently summarizing text and graph encodings of multi-document clusters. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *International Conference on Learning Representations*.

Jason Phang, Yao Zhao, and Peter J Liu. 2022. Investigating efficiently extending transformers for long input summarization. *arXiv preprint arXiv:2208.04347*.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian

Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *The Eleventh International Conference on Learning Representations*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019a. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7464–7473.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019b. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2023. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14590–14604, Toronto, Canada. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Q. Tran, Dani Yogatama, and Donald Metzler. 2023. Scaling laws vs model architectures: How does inductive bias influence scaling?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5.

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

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transform-

ers. In *International Conference on Learning Representations*.

Oskar Wysocki, Zili Zhou, Paul O'Regan, Deborah Ferreira, Magdalena Wysocka, Dónal Landers, and André Freitas. 2023. Transformers and the Representation of Biomedical Background Knowledge. *Computational Linguistics*, 49(1):73–115.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland. Association for Computational Linguistics.

Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. 2022. Simple local attentions remain competitive for long-context tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1975–1986, Seattle, United States. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

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

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training language models with memory augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5657–5673, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Dataset Statistics

Dataset	Train	Valid	Test	Avg. Input Tokens
arXiv	203.0K	6.4K	6.4K	6.0K
PubMed	119.9K	6.6K	6.7K	3.0K
GovReport	17.5K	1.0K	1.0K	9.6K
SummScreen	22.6K	2.1K	2.1K	6.6K
Multi-News	44.9K	5.6K	5.6K	1.8K
WCEP	8.1K	1.0K	1.0K	3.9K
NarrativeQA	32.7K	3.5K	10.6K	121.7K

Table 6: Statistics of used datasets.

B Hyper-parameters & Packages

Table 7 and Table 8 delineate the hyper-parameter settings at the training and inference phases of the experiment, respectively. For evaluation metrics, we used the following packages:

- For the uniform data processing, before the evaluation, all the reference texts and model-generated texts are converted to the lowercase and tokenized using the PTB tokenizer: https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/PTBTokenizer.html.
- For ROUGE metrics, we used the public rouge-score Perl package provided by the authors: https://github.com/summanlp/evaluation/tree/master/ROUGE-RELEASE-1.5.5.
- For BERTScore (Zhang* et al., 2020), we used the public bert-score package shared by the authors: https://github.com/Tiiiger/bert_score.

C Introduction for Baselines

We use the competitive baselines that demonstrate the downstream task results for comparison. Among them, **BART** (Lewis et al., 2020) is a standard full-attention PLM for sequence generation. Compared with BART, **PEGASUS** (Zhang et al., 2020) has a tailored pre-training objective for abstractive text summarization. **LED** (Longformer Encoder-Decoder) (Beltagy et al., 2020) uses the sparse attention-based encoder and full-attention decoder. Before pre-training, its parameters are initialized from BART. **BIGBIRD** (Zaheer et al., 2020), for an encoder-decoder setup, also introduces their specified sparse attention mechanism only at the encoder side. **PRIMERA** based on LED introduces a task-specific per-training objective for multi-document summarization. **SLED** (Ivgi et al., 2023) processes long sequences via short-context PLMs. The origin long sequence is partitioned into overlapping chunks. **HEPOS** (Huang et al., 2021) proposes head-wise positional strides to effectively pinpoint salient information from the source. **Memorizing Transformers** (Wu et al., 2022) employs a trainable attention gate to moderate between the standard cross-attention and attention over retrieved keys from a datastore. **Unlimiformer** (Bertsch et al., 2023) uses the k-nearest-neighbor KNN index to choose full-length input to reduce computation overhead. **HiMAP** (Fabbri et al., 2019a) expands the existing pointer-generator network model into a hierarchical network. **GraphSum** (Li et al., 2020) employs well-known graph representations of documents to effectively process multiple input documents for abstractive summarization. **BART-Long-Graph** (Pasunuru et al., 2021) is fine-tuned based on LED and additionally injects discourse graph. **LED+RELAX** (Parnell et al., 2022) introduces a RELAX gradient estimator with multi-document coverage reward. **BERTREG** (Gholipour Ghalandari et al., 2020) is a regression-based sentence ranking system with BERT embedding, which is used as an extractive summarization method. **Submodular+Abs** (Gholipour Ghalandari et al., 2020) consists of a submodular-based extractive summarizer and a bottom-up abstractive summarizer. **DynE** (Hokamp et al., 2020) ensembles multiple-document for abstractive summarization by single document summarization models.

D Perplexity on Test Sets

To more comprehensively demonstrate the effectiveness of our approach, we additionally use perplexity as an evaluation metric for comparative experiments. The experimental results in Table 9 indicate that our system (SimCAS) significantly enhances model performance compared to the baselines (BART).

E Time Latency at Inference Stage

At the inference stage, our framework additionally introduces a selection process compared with the standard transformer. Therefore, We study the proportion of our selector in the average inference time cost per sample. The results in Table 10 indicate that our selector only adds a small amount

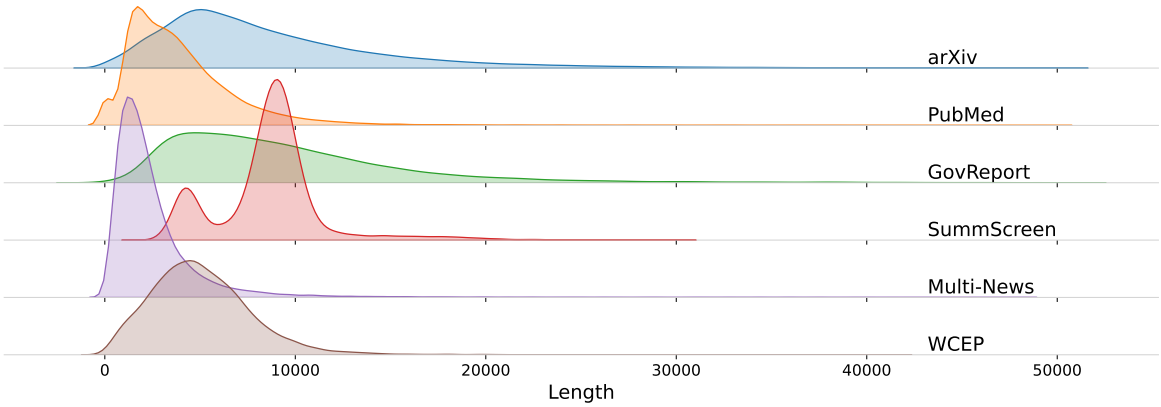

Figure 5: Input text length distributions on the six summarization datasets

Dataset	Model	Batch Size	Max Epoch	Warmup Steps
arXiv	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	10	6400,10000
PubMed	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	10	6400, 10000
GovReport	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	20	6400, 10000
SummScreen	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	20	2500, 6400
Multi-News	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	20	6400, 10000
WCEP	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	100	1600, 2500
NarrativeQA	$\text{BART}_{\text{base}}$, $\text{BART}_{\text{large}}$	1	10	6400, 10000

Table 7: Hyper-parameter grid for downstream task fine-tuning. We use Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-6$) for all datasets.

Dataset	Generation Parameters
arXiv	beam: 4, max_len: 300, min_len: 50, no_repeat_ngrams: 3, length_penalty: 5.0
PubMed	beam: 4, max_len: 400, min_len: 40, no_repeat_ngrams: 3, length_penalty: 4.0
GovReport	beam: 4, max_len: 740, min_len: 50, no_repeat_ngrams: 3, length_penalty: 4.0
SummScreen	beam: 4, max_len: 750, min_len: 50, no_repeat_ngrams: 3, length_penalty: 4.0
Multi-News	beam: 4, max_len: 400, min_len: 150, no_repeat_ngrams: 3, length_penalty: 2.0
WCEP	beam: 4, max_len: 40, min_len: 15, no_repeat_ngrams: 3, length_penalty: 2.0
NarrativeQA	beam: 4, max_len: 20, no_repeat_ngrams: 3, length_penalty: 1.0

Table 8: Hyper-parameter settings during inference for each dataset.

Method	arXiv	PubMed	GovReport	SummScreen	Multi-News	WCEP	NarrativeQA
BART	31.82	34.81	42.10	48.91	49.40	50.91	23.34
SimCAS	6.89	5.58	6.23	10.80	8.25	8.85	4.71

Table 9: Perplexity on all seven test sets including arXiv, PubMed, GovReport, SummScreen, Multi-News, WCEP, and NarrativeQA.

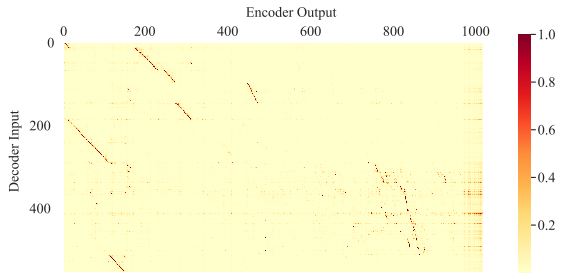

Figure 6: Visualization of the average cross-attention of all attention heads at all layers in the decoder (excluding start position). This example is generated by the $\text{BART}_{\text{base}}$+SimCAS trained on GovReport.

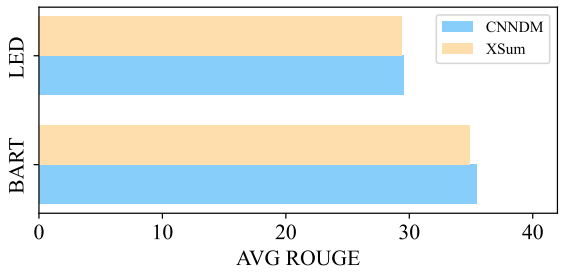

Figure 7: Comparison of the performance of full-attention PLM BART and sparse-attention PLM LED on short-text summarization datasets CNNDM (Nallapati et al., 2016) and XSum (Narayan et al., 2018). AVG ROUGE denotes the average of ROUGE-1/2/L F1 scores.

of time cost while significantly improving model performance, demonstrating the effectiveness of our approach.

F Potential to select representation

Because in the decoder cross-attention module, the encoded output from the encoder is used as "Key" and "Value" to participate in the calculation, its attention score can lead to the contribution of the corresponding token representation from encoded output to the current token decision at the decoding step. Figure6 demonstrates how the cross-attention scores change during the decoding of the reference output with one example in GovReport. We can observe that 1) most of the token decisions in the decoding phase focus only on a small set of encoded representations; 2) For each token decision, the contributions of different encoded token representations vary greatly. These phenomena suggest that there is still the feasibility of further filtering out low-contribution encoded representations.

G Compatible with short-input tasks

While the sparse-attention transformers have been proven effective on a wide range of long-sequence datasets, as shown in Figure 7, these methods tend to underperform traditional full-attention transformers on the more common short-sequence tasks. However, in real scenarios, short-sequence inputs and long-sequence inputs are often mixed together, and the former occupies the vast majority. This limits the application scope of sparse-attention transformer architecture. In contrast, applying our framework SimCAS to existing full-attention transformers has strong flexibility. Specifically, given the full-attention model under SimCAS, if the input sequence exceeds the maximum length of a single chunk, we will perform chunking and selecting, otherwise, we can naturally switch to a standard short-text encoding form by skipping chunking procedures.

H Efficacy of SBA

As our framework introduces SBA to align inter-chunk information during the encoding steps, we explore the change of hidden state in the forward propagation process under the influence of this component. Figure 8 demonstrates the visualized similarity matrix between each chunk's start hidden state after the encoding block. It can be seen from an array of cosine similarity matrices in the top half that, through the alternation of SBA and encoding blocks, the cosine similarities between the start hidden states from different chunks are all close to 1, which indicates that their directions in high-dimensional space are almost the same.

Considering that cosine similarity can only reflect the closeness between vector directions, we design a similarity calculation method to add the measure of Euclidean distance. Given a pair of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^d$, the custom similarity **Sim** between them is formulated as follows:

$$\textbf{Sim} = \frac{(\boldsymbol{v}_1, \boldsymbol{v}_2)}{\|\boldsymbol{v}_1\| * \|\boldsymbol{v}_2\| * (1 + \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|)},$$

by which, we scale the cosine similarity to observe its change. As can be seen from the lower part of Figure 8, after each encoder layer, the start hidden states will vary in scale due to different contexts.

I More Experimental Details

Since we aim to minimize dataset-specific modeling, we unify the processing of the used long-

Method	arXiv	PubMed	GovReport	SummScreen	Multi-News	WCEP	NarrativeQA
Selection (ms)	157.5	70.4	293.9	103.4	74.2	9.8	112.4
All (ms)	9661.5	13799.1	104953.7	30424.0	17664.2	1356.2	689.6
Ratio (%)	1.63	0.51	0.28	0.34	0.42	0.72	16.30

Table 10: The average time overhead of the selection process (Select) and the whole process (All), and their ratio (Ratio) for a single sample.

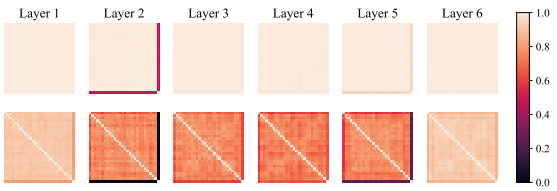

Figure 8: Evolution of similarities between start hidden states of each chunk after each encoding layer during one forward propagation. Only cosine similarity is used in the top half of the figure. The lower part takes into account both cosine similarity and Euclidean distance.

Algorithm 1 Token embedding selection with PPO

Initialize agent and hyper-parameters
Initialize advantages, rewards, and returns
Compute Rewards and Advantages
Initialize next_value $\leftarrow 0$, last_gae_lam $\leftarrow 0$
valid_reward_num $\leftarrow$ size of select_rewards
for $t = 0$ to num_steps-1 **do**
    Compute action sum for each step and update rewards
end for
cur_num $\leftarrow 0$
for $t = 0$ to num_steps-1 **do**
    Update rewards and actions based on select_rewards
end for
for $t =$ num_steps-1 to 0 **do**
    Compute delta, advantages, and returns
end for
Reverse advantages and returns
Policy and Value Function Update
for $epoch = 0$ to update_epochs-1 **do**
    Shuffle batch indices
    for $batch\_st = 0$ to seq_len by mini_batch **do**
        Compute policy loss, value loss, and entropy loss
        Perform gradient ascent
    end for
    if approx_kl $>$ threshold **then**
        Break
    end if
end for

document summarization, multi-document summarization, and machine reading comprehension datasets. In the NarrativeQA dataset for machine reading comprehension, we concatenate together the question and reference content, each of which prepends a prefix to indicate the type of text. For a fair comparison, we uniformly use the full-attention PLM BART[8] on the seven public datasets above. Built on the BART, our framework introduces an additional parameterized selector to focus on more task-specific token representations. The selector follows the actor-critic style (Konda and Tsitsiklis, 1999) and contains around 8M parameters. There are two Adam optimizers with $\beta_1 = 0.9$, $\beta_2 = 0.999$ for BART and selector respectively.

Additionally, to handle longer sequences during the training phase, we set the chunk size to 512 instead of 1024 in all experiments (considering the start token and end token). To maintain chunk-wise alignment, we pad the chunks to uniform the size of each chunk. During the forward propagation, the embedding layer embeds position representations for each chunk independently.[9]

[8]The checkpoints of BART are "facebook/bart-base" and "facebook/bart-large" containing around 139 M and 406 M parameters respectively, whose maximum encoding length is 1024.

[9]Considering the context of each chunk is different, although our design may lead to repeated position representation, we argue that after the encoding stage, the combination of the same token and position information would still produce various representations (Kazemnejad et al., 2023)

Maximum Input Length	R-1	R-2	R-L	BS
4096	44.96	22.60	37.17	69.74
8192	45.41	**22.79**	37.40	70.51
16384*	45.22	22.48	37.34	70.46
32768	45.45	22.56	37.45	70.52
$+\infty$	**45.68**	22.80	**37.71**	**70.59**

Table 11: Performance of $\text{BART}_{\text{base}}$-SimCAS on WCEP with different maximum input lengths during inference. R-1/2/L is the ROUGE-1/2/L F1 score. BS refers to the model-based metric BERTScore. $\star$: the maximum input length in the training phase. $+\infty$: the input sequence to the model is complete. The best results are bolded.

Training Details In this paper, for the training of $\text{BART}_{\text{base}}$ on various datasets we uniformly employ the Adam optimizer with the following dynamic learning rate:

$$lr = \gamma \min(step^{-0.5}, step \times warmup^{-1.5}),$$

where γ affects the maximum learning rate and *warmup* indicates the warmup steps, *step* is the number of updating steps, and lr is the learning rate. In addition, there is another separate Adam optimizer for our reinforcement learning-based parameterized selector, in which the learning rate is fixed to 1×10^{-4}. The optimization of the selector and transformer is performed alternately, with one model trained while the other model is fixed.

Inference Details In practice, We also investigate how the model performs when the maximum effective input length is increased during inference. In Table 11, we can observe that although the maximum input length is set to 16384 during the training phase due to the memory limitation, increasing this maximum length during inference still improves the model performance.

The Details of Selector Our token embedding selection process based on PPO is shown in Algorithm 1. In our setting, the selector consists of an actor component and a critic component with PPO optimization, both of which are the feed-forward network, except for the final layers. In order to enable the selector to choose more diverse token representations instead of becoming homogeneous during the chunk-wise selection process, the state space consists of the current token representation and the selector's hidden state that is affected by previous actions. At the beginning of the selection procedure, the initial selector hidden state is the average of the start hidden state of all chunk representations. At each time step, we input the current selector hidden state and a chunk of token representations. Then the actor of the selector outputs a probability distribution over action space, and the critic of the selector outputs a single estimated scalar value for each token based on the hidden state selector and the corresponding token representation. For the state transition after executing current actions, we update the hidden state of the selector using the selected token representation from the current chunk. Note that in order to avoid the extreme case where the selector skips all tokens, in each chunk decision, if all actions are "*skipping*", they are all switched to "*selecting*", and the corresponding action probability and value are re-obtained. Additionally, to increase the training stability, the advantage in the PPO algorithm is approximated using the Generalized Advantage Estimation (GAE) (Schulman et al., 2016).

J Case Study on GovReport Dataset

In addition to using regular automatic evaluation metrics to measure the effect of model generation, we also present some actual output to support the results. Figure 9 displays several examples of summaries generated by the fine-tuned base model $\text{BART}_{\text{large}}$ and our $\text{BART}_{\text{base}}$-SimCAS. We can observe that the system output of our model has fewer grammatical errors and higher coherence compared with the base model. Furthermore, since our model is able to perceive longer sequences, our output is more informative and better aligned with the reference text.

System	Summary
Reference	in may 2018 , gao reported that the trust fund , which pays disability benefits to certain coal miners , faced financial challenges . the trust fund has borrowed from the u.s. treasury 's general fund almost every year since 1979 to make needed expenditures . gao 's june 2019 testimony included preliminary observations that coal operator bankruptcies were further straining trust fund finances because , in some cases , benefit responsibility was transferred to the trust fund . this testimony is based on gao 's report being released today , and describes (1) how coal mine operator bankruptcies have affected the trust fund , and (2) how dol managed coal mine operator insurance to limit financial risk to the trust fund . in producing this report , gao identified coal operators that filed for bankruptcy from 2014 through 2016 . gao analyzed information on commercially - insured and self - insured coal operators , and examined workers ' compensation insurance practices in four of the nation 's top five coal producing states . gao also interviewed dol officials , coal mine operators , and insurance company representatives , among others . coal mine operator bankruptcies have led to the transfer of about $ 865 million in estimated benefit responsibility to the federal government 's black lung disability trust fund (trust fund) , according to dol estimates . the trust fund pays benefits when no responsible operator is identified , or when the liable operator does not pay . gao previously testified in june 2019 that it had identified three bankrupt , self - insured operators for which benefit responsibility was transferred to the trust fund . since that time , dol 's estimate of the transferred benefit responsibility has grown — from a prior range of $ 313 million to $ 325 million to the more recent $ 865 million estimate provided to gao in january 2020 . according to dol , this escalation was due , in part , to recent increases in black lung benefit award rates and higher medical treatment costs , and to an underestimate of one company 's (patriot coal) future benefit claims . trust fund , filed from 2014 through 2016 dol 's limited oversight of coal mine operator insurance has exposed the trust fund to financial risk , though recent changes , if implemented effectively , can help address these risks . in overseeing self - insurance in the past , dol did not : estimate future benefit liability when setting the amount of collateral required to self - insure ; regularly review operators to assess whether the required amount of collateral should change ; or always take action to protect the trust fund by revoking an operators ' ability to self - insure as appropriate . in july 2019 , dol began implementing a new self - insurance process that could help address past deficiencies in estimating collateral and regularly reviewing self - insured operators . however , dol 's new process still lacks procedures for its planned annual renewal of self - insured operators and for resolving coal operator appeals should operators dispute dol collateral requirements . this could hinder dol from revoking operators ' ability to self - insure should they not comply with dol requirements . further , for those operators that do not self - insure , dol does not monitor them to ensure they maintain adequate and continuous commercial coverage as appropriate . as a result , the trust fund may in some instances assume responsibility for paying benefits that otherwise would have been paid by insurers . gao made three recommendations to dol to establish procedures for self - insurance renewals and coal operator appeals , and to develop a process to monitor whether commercially - insured operators maintain adequate and continuous coverage . dol agreed with these recommendations .
BART (1K)	of the eight coal mine operator bankruptcies gao identified from 2014 through 2016 , three resulted in a transfer of estimated benefit liability from the coal operator to the trust fund and five did not , according to dol . dol estimates for how these bankruptcies will affect the trust fund have considerably increased from what dol had previously reported . in june 2019 , gao reported that dol estimated that between $ 313 million to $ 325 million in benefit liabilities would transfer to thetrust fund as a result of the bankruptcies . in january 2020 , however , dol provided updated estimates stating that $ 865 million in benefits would be transferred to thetrust fund.gao also reported in june 2019 that the federal government does not have a reliable estimate of the amount of collateral dol required from coal mine operators to self-insure , and that benefit liabilities in excess of the collateral can be transferred from the bankrupt operator.what gao foundthe department of labor 's (dol) estimates for the extent of black lung benefit liabilities transferred from operators to the u.s. trust fund are considerably higher than dol 's estimates , and dol officials said that their estimates increased , among other reasons , to account for higher black lung award rates that occurred from fiscal years 2016 through 2019 , and higher medical treatment cost inflation in recent years.dol 's prior estimate for the patriot coal (patriot) bankruptcy did not fully cover the estimated benefit liabilities . when this occurs , benefit liabilities that in excess or in excess to the collateral could be transferred into the trust trust fund . for example , the collateral required from alpha natural resources (alpha) was about $ 12 million and approximately $ 494 million of estimated black lung liability transferred to a trust fund in 2016.since 2016 , several other self-insured operators have also filed for bankruptcy , including cambrian coal , cloud peak energy , murray energy , and westmoreland coal . dol does not estimate future benefit liability when setting collateral or regularly review operators to monitor their changing financial conditions . in the past , agency procedures require that collateral be obtained from operators in an amount deemed necessary and sufficient to secure the payment of the operators ' liability . to determine collateral amounts under the former process , the agency procedures stated that an operator 's net worth was to be equal to 3 , 5 , or 10 years of a operator 's annual black lung benefit payments made at the time of a coal operator 's self-insurance application depending on its net worth . specifically , if net worth were $ 1 billion or greater , then dol procedures set collateral equal to three years of benefit payments . if net worth is $ 1.1 billion , then net worth would equal to 5 , 10 , or 20 years of the operator 's monthly benefit payments , depending on the net worth of the underlying collateral . the amount of the required collateral from these operators would not have to be transferred in the future , and therefore would not be transferred.the federal government has not developed reliable estimates of future black lung benefits liability for the coal operators gao selected for review because dol has not established a time frame for doing so . in december 2017 , d dol announced that it was developing a new estimate of future benefit liabilities for coal operators , but it has not yet determined the extent to which the new estimate will address the limitations of the current estimate . without reliable estimates , it will be difficult for dol to determine whether future benefits liability will be transferred or whether additional actions may be needed to address these limitations .
SimCAS (16K)	coal mine operator bankruptcies resulted in a transfer of estimated benefit liability from the coal operator to the trust fund and five did not , according to the department of labor (dol) . dol estimates that between $ 313 million to $ 325 million in benefit liabilities would transfer to the trust fund as a result of these bankruptcies . gao was asked to review dol 's oversight of coal mine operators ' self - insurance . this report examines (1) the extent to which dol has taken steps to protect the financial interests of the trust fund and (2) the actions dol took to mitigate the financial losses of the bankrupt operators . to conduct this work , gao reviewed dol documents and interviewed dol officials . from 2014 through 2016 , three self - insured coal mine operations resulted in the transfer of $ 865 million of estimated black lung benefit liability to the u.s. trust fund (trust fund) . of the eight bankruptcies gao identified , three resulted in an transfer of the estimated benefit liabilities from the coal mine operators (coal operators) : three (energy future holdings , peabody energy , and walter energy) did not affect the trust fund . the amount of collateral dol required from these operators to self - insure did not fully cover their estimated benefits liabilities . when this occurs , benefit liabilities in excess of the collateral can be transferred to thetrust fund . for example , the collateral dl required from energy future holdings was about $ 12 million and approximately $ 494 million of expected benefit liability transferred to the trust fund . dol also did not routinely consider potential future claims for which an operator could be responsible . in reviewing the most recent reauthorization memos for each of the self - insured operators , we found that these operators were not reauthorized since 1988 . in january 2020 , dol provided updated estimates stating that $ 865 million in benefits would be transferred from these two bankruptcies as a consequence of the three other bankruptcies , but dol does not have procedures that specify the duration of an operator 's selfinsurance authority or the conditions under which that authority would not be renewed . in the absence of a process to monitor operator compliance with program requirements , the agency risks not identifying or cancelling operator coverage . in addition , the new procedures do not specify , among other things , how long an operator will be authorized to selfinsure ; when an operator is authorized to do so ; when the operator is not renewing the operator 's authority ; and what conditions are under which this authority would be not renewed . the new requirements for setting collateral and for the more frequent review of selfinsured operators are key elements of internal controls , which call for agency management to implement control activities through policy . however , dl 's new self - insurance procedures do not specify , for example , how much collateral the agency will require an operator to secure and how much time dol appeals decisions should be made . in particular , dls does not specify a goal for the amount of time that it will take for the operator to apply for and receive a renewal application . without such a goal , it is difficult for dol to ensure that the application is valid and that the operator has a valid application for and receives an extension of the application . in commenting on a draft of this report , dod concurred with gao 's findings and recommendations .

Figure 9: Example summary generated by BART and SimCAS trained on GovReport dataset. The maximum input length of standard BART and SimCAS is 1024 and 16384 respectively. The sentence in green is included in the SimCAS summary, while the one in red is discarded.