Efficient Long-Text Understanding with Short-Text Models

Maor Ivgi **Uri Shaham** **Jonathan Berant**
The Blavatnik School of Computer Science, Tel-Aviv University, Israel
{maor.ivgi,uri.shaham,joberant}@cs.tau.ac.il

Abstract

Transformer-based pretrained language models (LMs) are ubiquitous across natural language understanding, but cannot be applied to long sequences such as stories, scientific articles, and long documents due to their quadratic complexity. While a myriad of efficient transformer variants have been proposed, they are typically based on custom implementations that require expensive pretraining from scratch. In this work, we propose **SLED**: **SL**iding-**E**ncoder and **D**ecoder, a simple approach for processing long sequences that re-uses and leverages battle-tested short-text pretrained LMs. Specifically, we partition the input into overlapping chunks, encode each with a short-text LM encoder and use the pretrained decoder to fuse information across chunks (*fusion-in-decoder*). We illustrate through controlled experiments that SLED offers a viable strategy for long text understanding and evaluate our approach on SCROLLS, a benchmark with seven datasets across a wide range of language understanding tasks. We find that SLED is competitive with specialized models that are up to 50x larger and require a dedicated and expensive pretraining step.

1 Introduction

Transformer-based pretrained language models (Vaswani et al., 2017; Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020b; Brown et al., 2020) have been widely successful across all areas of natural language understanding. However, applying them over long texts (such as stories, scripts, or scientific articles) is prohibitive due to their quadratic complexity in the input length. To bridge this gap, recent work has developed more efficient transformer variants (Kitaev et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020a; Guo et al., 2022) and applied them over long-range language understanding tasks (Mehta et al., 2022; Shaham et al., 2022).

However, most efficient transformers use specialized architectures with custom implementations that are not guaranteed to scale as well as vanilla transformers (Tay et al., 2022a). Moreover, they require an expensive pretraining step and do not exploit off-the-shelf pretrained LMs that were trained for short texts. To date, their performance on long texts has not matched the success of their short-range counterparts.

In this work, we present **SLED**: **SL**iding-**E**ncoder and **D**ecoder, a simple yet powerful method for applying off-the-shelf pretrained encoder-decoder models on long text problems, with a linear time and space dependency. Specifically (see Figure 2), we partition long documents into overlapping chunks of tokens of constant length and encode each chunk independently with an already-pretrained encoder. Then, a pretrained decoder attends to all contextualized input representations to generate the output. Our main assumption is that input tokens can be contextualized through their local surrounding (using a short-text LM), and any global cross-chunk reasoning can be handled by the decoder, similar to fusion-in-decoder (FiD) (Izacard and Grave, 2021). Our approach can be readily applied to *any* pretrained encoder-decoder LM such as T5 (Raffel et al., 2020b) and BART (Lewis et al., 2020) (but is not applicable to decoder-only [Brown et al., 2020] or encoder-only models [Liu et al., 2019; Conneau et al., 2020]).

We evaluate SLED on a wide range of language understanding tasks. To substantiate SLED's adequacy for text processing, we perform controlled experiments over modified versions of SQuAD 1.1 (Rajpurkar et al., 2016) and HotpotQA (Yang et al., 2018) to show that SLED can (a) find relevant information that is embedded within a long text sequence and (b) fuse information from chunks that were encoded separately.

Our main evaluation is over SCROLLS, a recently-released benchmark that includes 7 long-range tasks across Question Answering (QA),

Transactions of the Association for Computational Linguistics, vol. 11, pp. 284–299, 2023. https://doi.org/10.1162/tacl_a_00547
Action Editor: Mauro Cettolo. Submission batch: 8/2022; Revision batch: 12/2022; Published 3/2023.

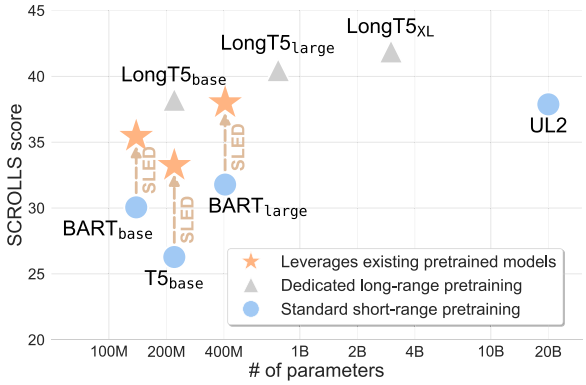

Figure 1: Models' SCROLLS score (Shaham et al., 2022) as a function of parameter count. Plugging existing pretrained LMs into the SLED framework dramatically improves their SCROLLS score (arrows from blue circles to orange stars). Gray triangles indicate models with dedicated pretraining for capturing long-range dependencies. $\text{BART}_{\text{large}}$-SLED is competitive with $\text{LongT5}_{\text{base}}$ (Guo et al., 2022) and UL2 (Tay et al., 2022b) (which has 50x more parameters), and slightly lags behind larger LongT5 models.

Summarization, and Natural Language Inference (NLI). We show (Figure 1) that taking a pre-trained encoder-decoder model, such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020b), and embedding it into SLED's framework results in dramatic improvement in performance (6 points on average across models). Moreover, $\text{BART}_{\text{large}}$-SLED's performance is comparable to $\text{LongT5}_{\text{base}}$ (Guo et al., 2022), a model that was specifically pretrained to handle long-range dependencies, and surpasses UL2 (Tay et al., 2022b), which contains 50x more parameters. Importantly, SLED-based models can use any future pretrained LM out-of-the-box without requiring additional pretraining to further improve performance.

Due to its simplicity, SLED can also be used as a diagnostic tool for analyzing long-range benchmarks. We analyze the seven datasets in SCROLLS through the lens of SLED and show which datasets require the input to be contextualized with remote tokens. Specifically, we find that in QA and NLI tasks, relatively local contextualization is sufficient for high performance.

While SLED is similar to FiD from a technical standpoint, past usage of FiD has centered around open-domain question answering (Izacard and Grave, 2021), where unrelated passages are naturally encoded independently. Here, we test fusion-in-decoder on long documents, where local encoding of chunks is a modeling assumption that needs testing. In recent work, Vig et al. (2022) proposed a similar architecture to tackle long inputs from QMSum (Zhong et al., 2021), but did not systematically analyze it. We standardize this methodology for the first time, and extensively analyze the effectiveness of FiD for encoding long documents across multiple tasks.

To summarize, our main contributions are:

1. We present SLED, a simple and effective approach for processing long texts that leverages off-the-shelf encoder-decoder LMs based on fusion-in-decoder.
2. We demonstrate SLED's efficacy in both controlled experiments, as well as on the SCROLLS benchmark, which leads to competitive results compared to specialized models that include up to 50x more parameters.
3. We use SLED as a diagnostic tool for analyzing the long-range properties of datasets in the SCROLLS benchmark.
4. We provide an open-source implementation of SLED,[1] seamlessly integrated into the Transformers library (Wolf et al., 2020).

2 Background

Recent advances in natural language processing have been by and large fueled by the transformer architecture (Vaswani et al., 2017). A core component of the transformer is the self-attention layer where every input token ''attends'' to every other token to produce its contextualized representation. This results in quadratic time and space dependency w.r.t. the length of the input, limiting the ability of transformers to process long sequences.

This long-text limitation has sparked ample interest in developing efficient transformer variants. One prominent family of methods is based on *sparse attention*, where each token attends to a constant number of other tokens, overcoming the quadratic dependency. Tokens typically attend either to their *local* surrounding (Zaheer et al., 2020a; Beltagy et al., 2020; Ainslie et al., 2020; Gupta and Berant, 2020) or to tokens that are semantically similar (Kitaev et al., 2020; Roy et al., 2021). Moreover, a constant number of *global tokens* that attend to and are attended by

[1] https://github.com/Mivg/SLED.

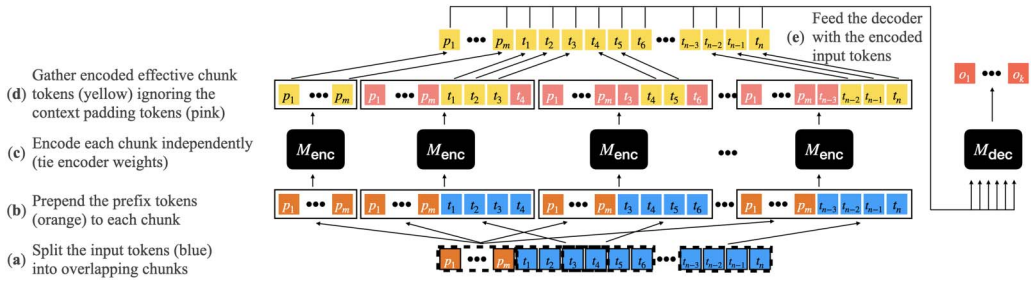

Figure 2: Overview of SLED. (a) Input tokens $(t_1, \ldots, t_n)$ are chunked into C overlapping chunks of length c (here, $c = 4$). Each chunk is made of $P := \frac{\rho \times c}{2}$ *context padding* tokens at the right and left edges of the chunk, and $(1 - \rho) \times c$ *effective chunk* tokens in the middle (here, $\rho = 0.5, P = 1$). (b) We prepend the prefix tokens $(p_1, \ldots, p_m)$ to each chunk ($m \ll n$). (c) Each chunk is encoded independently using the already pretrained backbone encoder M_{enc}. (d) We gather the encoded *effective chunks* tokens (yellow) and discard the *context padding* tokens (pink) (e) We pass the encoded input to the decoder to generate the final output sequence $(o_1, \ldots, o_k)$.

all input tokens are often added to each attention sub-layer. Recent analyses (Xiong et al., 2022a) have shown that sparse transformers with local attention are competitive with other variants on multiple language understanding tasks.

Our method, SLED, falls into the family of local attention variants. However, unlike prior work, SLED re-uses and extends existing short-range encoder-decoder models, and does not require specialized pretraining or dedicated CUDA implementations.

In most local attention variants, for example, LED (Beltagy et al., 2020), attention is local *per-layer*, but the receptive field of tokens grows *across layers*. In SLED, which we describe next, tokens have access to the same number of tokens, independent of a layer's depth, which enables better parallelization. For a survey on the families of efficient transformers, see (Tay et al., 2020). For an in-depth comparison of SLED and LED, we refer to Appendix B.

3 Method

In this work, we propose a simple approach for avoiding transformer's quadratic complexity, motivated by the *Locality of information assumption*:

> In an encoder-decoder architecture, the encoder can effectively contextualize input tokens with local context only, leaving long-range dependencies to be handled by the decoder.

SLED relies on said modeling assumption to encode shorter chunks independently and perform fusion of information in the decoder (Izacard and Grave, 2021). We now describe the SLED model in detail.

Input SLED uses a pretrained encoder-decoder model M as a backbone. SLED receives a tokenized document of length n (blue squares in Figure 2), and an optional short tokenized prefix of length $m \ll n$, typically representing a question about the document, an instruction to perform some generation task, or a hypothesis (orange squares in Figure 2). Unlike static task-specific prefixes (e.g., ''summarize''), SLED supports also sample-specific prefixes that are part of the input (e.g., the question in QA datasets).

Steps SLED follows the following steps:

(a) Document tokens are split into C chunks of length c (In Figure 2, $c = 4$). The *middle* $(1 - \rho) \times c$ tokens in each chunk are contextualized from both the left and right by $P := \frac{\rho \times c}{2}$ tokens, where $\rho \in [0, 0.5]$ ($\rho = 0.5$ in Figure 2). We call these middle tokens *the effective chunk*, since they will constitute the output of the encoder, and term the tokens on each side by *context padding*.

(b) Each chunk is prepended by (optional) prefix tokens (Figure 2(b)).

(c) Each chunk is encoded independently, using the backbone encoder M_{enc} (see Figure 2(c)).

(d) To create a contextualized representation for each token, we keep from each chunk only the tokens from the *effective chunk*, and concatenate them (Figure 2(d)).

(e) To give the decoder access to prefix tokens, we encode the prefix tokens with M_{enc}, and prepend the result to the contextualized representation (leftmost chunk in Figure 2(a)-(d)).

(f) Finally, we generate the output with the backbone decoder, M_{dec}, which uses standard cross-attention over the $m + n$ encoded tokens (Figure 2(e)).

SLED requires handling a few edge cases, namely, dealing with the first and last chunk that do not have bidirectional context. We refer to Appendix A for these details.

SLED's Complexity SLED divides an input of length n to C chunks of size c. Since $\rho \in [0, 0.5]$, it follows that $C \in \left[\frac{n}{c}, \frac{2n}{c}\right]$. While the complexity of encoding each chunk is quadratic in c due to self-attention, $c \ll n$ is constant and thus the memory and compute dependency is linear in n.[2] In particular, the complexity to encode the input with a model of l attention layers is:

$$\mathcal{O}\left(l \times c^2 \times \frac{2n}{c}\right) = \mathcal{O}\left(l \times c \times n\right).$$

Decoding is done as proposed by Vaswani et al. (2017), thus requiring $\mathcal{O}(nk + k^2)$ memory. Assuming a constant output sequence length $k \ll n$, this remains linear in n.

4 Efficacy of Fusion in Decoder

As mentioned (§3), SLED relies on the assumption that chunks can be encoded independently and fusion across them can be delegated to the decoder (*Locality of information assumption*). This is similar to the Fusion-in-Decoder approach, introduced by Izacard and Grave (2021) for open-domain question answering (ODQA). However, there, the encoder-decoder receives a set of independent passages and needs to generate an answer that can typically be extracted from a single passage. Here, we extend the scope of FiD by applying it over a single, long, and coherent input that potentially requires global contextualization.

To demonstrate the viability of FiD for long text language tasks, we design two controlled experiments that quantify the extent to which FiD can perform two operations at the heart of long-text processing. First, can FiD find a ''needle-in-a-haystack'', that is, locate a piece of short information embedded in long text, disregarding irrelevant information. Second, can FiD ''piece the puzzle'' and fuse two pieces of information that are encoded independently when generating an output.

4.1 Needle in a Haystack

To check if SLED can ignore irrelevant text and locate a single piece of information, we cast SQuAD 1.1 (Rajpurkar et al., 2016) as a sequence-to-sequence task with long input. SQuAD is a question answering dataset, where given a question-paragraph pair the goal is to generate the answer (which lies within the paragraph). For each question-paragraph pair, we randomly sample 9 other paragraphs from the the dataset and concatenate them to form a long document.[3] We then finetune and evaluate our models in two settings: a) *Ordered Distractors*: the gold paragraph is the first one, and all other distractors are concatenated after it. b) *Shuffled Distractors*: we randomly shuffle the order of all paragraphs so the answer can be anywhere in the input document. Since this is a QA task, the prefix is the question.

We use $\text{BART}_{\texttt{base}}$ (Lewis et al., 2020) as our backbone model, M, throughout §4, and compare SLED to an *oracle* $\text{BART}_{\texttt{base}}$ that is given the gold paragraph only with no distractor paragraphs. this is an oracle setup since $\text{BART}_{\texttt{base}}$ can take 1,024 tokens as input and all gold paragraphs are shorter. If SLED can match the oracle performance, we can infer that indeed the decoder can find a needle in a haystack. In addition, we compare SLED to $\text{BART}_{\texttt{base}}$, which is given only the first 1K tokens, and to LED (Beltagy et al., 2020), which uses local sparse attention, similar to SLED (LED has the same backbone $\text{BART}_{\texttt{base}}$). However, as explained in §2, the receptive field of LED layers linearly grows with the number of layers, and thus information can be fused in the encoder, unlike SLED where cross-chunk fusion must be delegated to the decoder. Last, for QA tasks, LED defines the question tokens as *global tokens*, and as an additional sanity test we evaluate $\text{LED}^{\mathcal{L}}$, that is, a local LED model where no global tokens are used. For both LED and SLED we use a chunk size $c = 256$.

[2] We assume the prefix length (m) is negligible and thus its effect on asymptotic complexity is negligible.

[3] We only consider paragraphs that are not within the gold document and do not contain the gold answer.

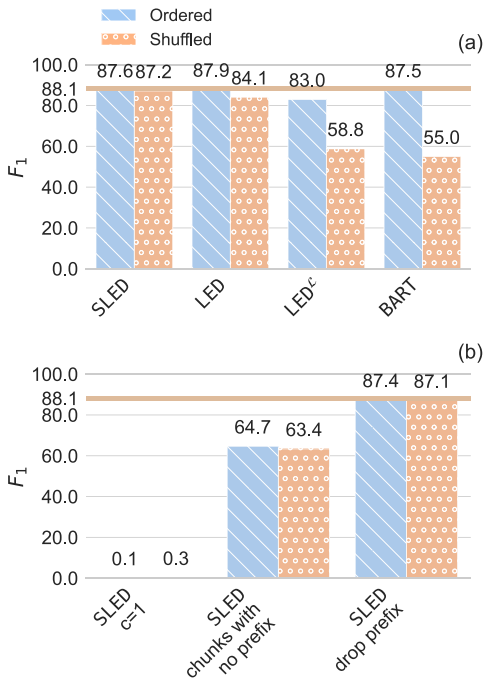

Figure 3: F_1 results on our modified SQuAD 1.1's (Rajpurkar et al., 2016) development set evaluation: (a) the horizontal line gives the performance of an oracle $\text{BART}_{\texttt{base}}$ given the gold paragraph only. SLED matches oracle performance in both the ordered and shuffled setting (see text). LED slightly underperforms SLED in the shuffled setup. Both BART (given only the first 1K tokens) and LED with no global tokens (LED$^{\mathcal{L}}$) perform poorly in the shuffled setup. (b) Ablations on SLED's architecture, see §4.3 for details.

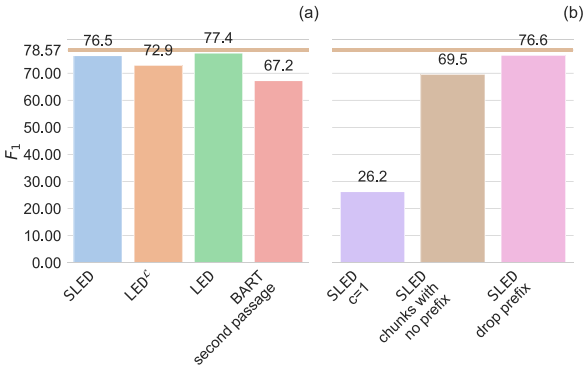

Figure 4: F_1 results on our HotpotQA's development set (Yang et al., 2018). (a) SLED reaches an F_1 that is close to the oracle $\text{BART}_{\texttt{base}}$ (horizontal line), outperforming a model with access to the paragraph that contains the answer (''second paragraph''). This shows that SLED effectively fuses information from two chunks. See text for further explanation on each model. (b) Ablations on SLED's architecture, see §4.3 for details.

Results Figure 3(a) shows the results of our evaluation on the development set. SLED almost matches the performance of an oracle $\text{BART}_{\texttt{base}}$ that is not given any distractor paragraphs, reaching an F_1 score of 87.6 compared to the oracle F_1 of 88.1 (horizontal line in the figure). LED also achieves high performance (but lower than SLED in the shuffled setup), showing both models learn to ignore distracting information and find a needle in a haystack. As expected, both LED$^{\mathcal{L}}$ and BART suffer a significant drop in performance when the passages are shuffled, as the gold paragraph is not contextualized with the question.

4.2 Piecing a Puzzle

We now verify that SLED can fuse pieces of information from different chunks. To this end, we modify HotpotQA (Yang et al., 2018), a multi-hop question answering dataset, in which every question relies on two pieces of information (located in different paragraphs). While in the original setting, each input in HotpotQA has two gold paragraphs and 8 distractor paragraphs, we include only the two gold paragraphs in our experiments. To ensure that SLED and LED encode the relevant two pieces of information in *separate* chunks, we set the chunk size to $c = 128$.

Similar to §4.1, we compare SLED to an oracle $\text{BART}_{\texttt{base}}$ with full attention over 1,024 tokens,[4] to LED, and to LED$^{\mathcal{L}}$. Finally, past work has shown that many examples in HotpotQA can be answered with access to the ''second'' gold paragraph only, which contains the answer (Jiang and Bansal, 2019). Thus, we also evaluate a BART model that is given the second passage only.

Results Figure 4(a) shows that indeed, SLED's decoder can effectively fuse information from two separately encoded chunks, reaching an F_1 of 76.5, slightly lower than the oracle F_1 of 78.6. Notably, SLED substantially outperforms a BART model with access to the entire second paragraph, showing that information is fused by the decoder. LED slightly outperforms SLED, but when denied access to global tokens (LED$^{\mathcal{L}}$) its performance drops sharply. This shows that the large receptive field of deep LED layers does not suffice for information fusion and interaction between the question and text is crucial for the decoder.

To summarize, our two controlled experiments show that SLED can perform the operations of retrieving and fusing information, which are fundamental for long text language tasks.

[4] All examples have ≤1,024 tokens, including the prefix.

4.3 Ablations of Design Choices

We leverage our controlled experimental setup to further investigate the components of SLED.

Efficacy of the Encoder While §4.2 shows that SLED can fuse separate pieces of information in the decoder, it is not clear to what extent local contextualization is necessary. To check whether it is possible for all fusion to occur in the decoder, we finetune SLED with a chunk size of $c = 1$, such that input tokens do not observe any context in the encoder. As can be seen in the leftmost bar(s) in Figure 3(b) and Figure 4(b), removing local contextualization results in poor performance, illustrating the importance of local contextualization.

Contextualizing Chunks with a Prefix As explained, SLED does not use global tokens, but instead contextualizes each chunk with a prepended prefix. To verify its necessity, we finetune a SLED model that treats the prefix as another chunk and does not prepend it to document chunks.[5] The second bar(s) in Figure 3(b) and Figure 4(b) shows a significant drop in performance for all settings, suggesting the prefix is needed during encoding.

As expected, there is practically no difference between the *Ordered* and *Shuffled* settings in Figure 3(b). In contrast, LED$^{\mathcal{L}}$, which is similar in concept (due to the lack of global tokens), shows a significant drop when paragraphs are shuffled. This shows the possible effectiveness of the increased receptive field in LED, but only when the gold paragraph is relatively close to the prefix.

Encoding the Prefix After showing that the prefix is crucial for the encoder, we ask whether the decoder needs direct access to the prefix or whether relevant information from the prefix can be infused into the chunk representations. To test that, we finetune SLED as usual, but remove the prefix tokens from the final representation given to the decoder. The rightmost bar(s) in Figure 3(b) and Figure 4(b) shows that providing the decoder with prefix representations makes little difference if any at all, suggesting that indeed the encoder can infuse the important information from the prefix into the encoded document tokens.

[5]We add masked padding after the prefix to ensure chunking of the document remains identical.

5 Experiments

We evaluate SLED on SCROLLS (Shaham et al., 2022), a recently proposed benchmark for evaluating long text understanding. SCROLLS contains seven datasets that span three different language understanding tasks:

1. Summariazation: *GovReport* (Huang et al., 2021) is a summarization task over reports from the Congressional Research Service; *SummScreenFD* (Chen et al., 2022) is a summarization dataset over TV scripts; *QMSum* (Zhong et al., 2021) is a query-based summarization dataset over meeting transcripts from various domains. While GovReport and SummScreenFD do not contain a prefix, for QMSum we consider the query as the prefix.
2. Question answering (QA): *Qasper* (Dasigi et al., 2021) is a QA benchmark that contains questions over NLP papers; *NarrativeQA* (Kočiský et al., 2018) contains questions over entire books and movie scripts; *QuALITY* (Pang et al., 2022) is a multiple-choice QA dataset over books and articles. For all QA datasets, we set the question as the prefix. For QuALITY, we consider the four answer options part of the question.
3. Natural language inference: *ContractNLI* (Koreeda and Manning, 2021) contains short legal hypotheses (set as the prefix) and legal documents as the premise. Models are tasked to predict whether the premise entails, contradicts or is neutral w.r.t. to the hypothesis.

For each task, we use the official evaluation metrics defined in SCROLLS, which are based on the metrics from the original datasets.

5.1 Settings

We evaluate SLED with both BART (Lewis et al., 2020) and T5 (Raffel et al., 2020b) as backbone models. For each backbone model, we compare performance with SLED, which can consume long sequences, vs. the backbone models alone that are fed with the first 1,024 tokens. For comparison, we also finetune LED$_{\texttt{base}}$. In all SLED and LED experiments, we use a maximal sequence length of 16K tokens and chunk size of 256 to allow for a fair evaluation.

For each model-dataset pair, we run hyperparameter tuning (detailed in Appendix C) based

Model	(Chunk/Input)	#Params	Avg	GovRep ROUGE-1/2/L	SumScr ROUGE-1/2/L	QMSum ROUGE-1/2/L	Qspr F1	Nrtv F1	QALT EM-T/H	CNLI EM
Development Scores										
LED_{base}	$(256/16K)$	162M	–	**57.3/27.9/30.0**	30.7/6.3/17.9	32.5/9.0/21.1	30.4	20.2	30.9	82.3
T5_{base}	$(1K/1K)$	220M	–	32.8/11.7/20.2	22.2/3.7/15.3	26.1/6.6/19.8	13.2	14.9	35.1	76.8
T5_{base}-SLED	$(256/16K)$	220M	–	47.0/20.2/25.2	25.3/5.0/16.6	29.9/8.7/21.4	38.2	18.2	34.6	82.4
$\text{BART}_{\text{base}}$	$(1K/1K)$	139M	–	47.7/18.5/22.3	30.1/7.0/18.3	32.2/9.3/21.1	23.3	15.9	33.8	78.4
$\text{BART}_{\text{base}}$-SLED	$(256/16K)$	139M	–	55.7/24.8/25.8	33.6/8.5/19.2	34.4/11.5/22.7	35.8	21.3	33.7	**85.3**
$\text{BART}_{\text{large}}$	$(1K/1K)$	406M	–	50.6/19.8/23.5	32.1/7.4/18.7	33.3/9.4/21.6	24.5	17.9	36.1	79.3
$\text{BART}_{\text{large}}$-SLED	$(256/16K)$	406M	–	**57.4**/26.3/27.5	**35.3/8.8/19.5**	**36.3/12.2/23.3**	**42.5**	**23.6**	**37.2**	**85.3**
Test Scores										
LED_{base}	$(256/16K)$	162M	33.6	56.8/**27.3/29.2**	30.0/6.0/17.5	31.3/8.6/20.5	34.8	21.0	28.5/28.3	82.9
T5_{base}	$(1K/1K)$	220M	26.3	33.2/12.1/20.4	21.4/3.6/15.0	24.2/5.9/18.6	16.3	15.0	31.9/28.6	76.3
T5_{base}-SLED	$(256/16K)$	220M	33.3	46.6/20.1/25.1	24.5/4.6/16.5	28.4/8.7/20.5	43.0	18.9	31.2/29.4	81.4
$\text{BART}_{\text{base}}$	$(1K/1K)$	139M	30.6	48.0/19.1/22.7	30.1/6.6/18.1	31.2/9.1/20.3	27.6	16.0	32.5/31.6	77.1
$\text{BART}_{\text{base}}$-SLED	$(256/16K)$	139M	35.4	54.7/24.4/25.4	32.7/7.9/19.1	33.8/**11.7/22.6**	41.1	21.5	29.7/30.4	85.6
$\text{BART}_{\text{large}}$	$(1K/1K)$	406M	32.1	50.7/20.1/23.5	31.6/6.8/18.5	32.0/9.1/20.8	29.2	18.3	**34.8**/33.9	79.7
$\text{BART}_{\text{large}}$-SLED	$(256/16K)$	406M	**38.0**	**57.5**/26.3/27.4	**35.2/8.7/19.4**	**34.2**/11.0/22.0	**46.9**	**24.1**	**34.8/34.8**	**87.3**
$\text{LED}^{\text{SCROLLS}\dagger}_{\text{base}}$	$(1K/16K)$	162M	29.2	**56.2**/26.6/28.8	24.2/4.5/15.4	25.1/6.7/18.8	26.6	18.5	25.8/25.4	71.5
$\text{LongT5}^{\dagger}_{\text{base}}$	$(255/16K)$	220M	38.2	53.5/27.3/29.3	34.8/**9.6**/21.1	33.9/11.0/22.8	46.6	23.0	37.9/36.6	85.6
$\text{LongT5}^{\dagger}_{\text{large}}$	$(255/16K)$	770M	40.5	54.2/27.8/29.8	35.6/9.2/**21.2**	35.1/**12.0**/23.3	52.3	27.2	40.6/38.6	87.3
$\text{LongT5}^{\dagger}_{\text{XL}}$	$(255/16K)$	3B	**41.9**	54.7/**28.2/30.2**	**35.8/9.6**/21.1	**34.9**/11.8/**23.5**	**53.1**	**29.3**	**46.0/42.1**	88.2
UL2†	$(2K/2K)$	20B	37.9	53.6/26.1/28.8	32.9/7.8/19.4	31.1/8.5/20.4	37.6	24.2	45.8/40.7	**88.7**

Table 1: Main results on the SCROLLS benchmark. Chunk/Input refers to the chunk size used (c) and to the maximal input length (n). Avg is the average SCROLLS score as described in Shaham et al. (2022). Development scores for QuALITY are only for the full set (T). † indicates reported results from SCROLLS public leaderboard.[6] $\text{LED}^{\text{SCROLLS}}_{\text{base}}$ scores were reported by Shaham et al. (2022) and are lower than our LED_{base} implementation, presumably since our implementation uses all question tokens for global attention rather than just the first one. The results for LongT5 and UL2 were submitted to the SCROLLS leaderboard by their authors.

on the development set. Additionally, we submit generated predictions over the test set to SCROLLS leaderboard,[6] and compare to the reported performance of other models at the time of submission.

5.2 Results

Table 1 reports results over SCROLLS development and test sets. Taking short-range pretrained LMs like BART and T5 and casting them into SLED's framework allows them to process long documents effectively, improving the average SCROLLS score by 4.8-7 points. Examining $\text{BART}_{\text{base}}$-SLED, we see a large improvement compared to LED_{base} (33.6→35.4), and competitive performance on multiple tasks compared to $\text{LongT5}_{\text{base}}$ and UL2. Moreover, adding SLED to $\text{BART}_{\text{large}}$ results in a high-performing model with results that are comparable to $\text{LongT5}_{\text{base}}$ and outperforming UL2, despite UL2's large parameter count (50x larger), and with no need for expensive pretraining geared towards long-range tasks. $\text{BART}_{\text{large}}$-SLED's performance is moderately lower than the larger LongT5 models.

Barring QuALITY, SLED significantly improves performance across all tasks compared to the corresponding backbone models. All summarization datasets (GovReport, SummScreenFD and QMSum) show impressive gains of up to 35% compared to their baseline scores, across all metrics (Rouge-1/Rouge-2/Rouge-L [Lin, 2004]) and for all three backbone models. Similarly, on ContractNLI (Koreeda and Manning, 2021) we see large relative improvements. As the performance of the baseline models was already high, this boost in performance is even more significant. Finally, the QA datasets Qasper and NarrativeQA show the largest gains, improving by an average of 60%.

QuALITY In stark contrast to other datasets lies the multi-choice QA dataset QuALITY (Pang et al., 2022). While the performance of $\text{BART}_{\text{large}}$-SLED is above chance, it barely improves the performance of its backbone model ($\text{BART}_{\text{large}}$), which observes only the first 1K

[6] https://www.scrolls-benchmark.com/leaderboard.

tokens, with a similar trend in other backbone models. Analyzing test scores in Table 1, we see that increasing model size consistently improves performance (up to 46% exact match), but increasing input length has a negligible effect. Since reported human accuracy on QuALITY is high (93.5%), this hints that QuALITY might require commonsense reasoning and knowledge that are absent from models with a lower parameter count.

Summary We have shown that taking off-the-shelf pretrained LMs and embedding them into SLED leads to competitive performance on SCROLLS. Importantly, any future pretrained LM can be easily plugged into SLED, without the need for an expensive pretraining step.

5.3 Dataset Analysis

SLED's simplicity and modularity allow it to be used as a useful tool for dataset analyses. Specifically, we can vary the chunk size, c, and the number of tokens, n, across datasets to analyze a) how local are individual pieces of relevant information, and b) how far into the document they are located.

Locality of Information SLED relies on an assumption that information can be contextualized locally at encoding time. To analyze locality, we vary the chunk size, c, which defines the attention window, and measure the effect on SCROLLS datasets with input length 16K. Figure 5 shows the results of this experiment, where the y-axis shows the relative improvement compared to $\text{BART}_{\texttt{base}}$ on a target metric as a function of the chunk size c for all datasets. We observe that in all datasets the best performing chunk size is relatively small (up to 256), and further increasing c even hurts the performance in some cases. However, the summarization datasets show a much larger gain in performance when increasing c up to that threshold. This coincides with a common hypothesis that QA and NLI require relatively local context, and thus increasing c can add noise and hurt optimization, while summarization may require a more high-level view of information.

Distance from Start of Document We now analyze whether the entire document is indeed required for tasks in SCROLLS by varying the maximum document length, n. Figure 6 shows the results of this experiment, where the y-axis shows relative improvement of $\text{BART}_{\texttt{base}}$-SLED

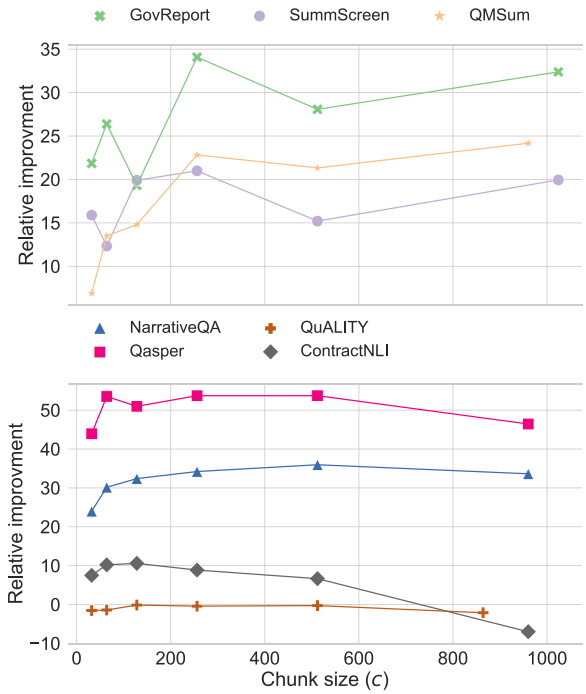

Figure 5: $\text{BART}_{\texttt{base}}$-SLED relative improvement compared to $\text{BART}_{\texttt{base}}$ results, when varying the SLED's chunk size (i.e., c), fixing the maximum input length to 16K. **Top**: Summarization datasets. The y-axis measures relative improvement of Rouge-2. **Bottom**: QA and NLI datasets. The y-axis measures relative improvement of exact match for QuALITY and ContractNLI and F_1 for NarrativeQA and Qasper.

compared to $\text{BART}_{\texttt{base}}$ as a function of the first n tokens from the document (chunk size $c = 256$). As expected, all datasets (except QuALITY) show a roughly monotonic improvement in performance with n. This shows that (a) SLED is able to effectively use all of the information in a long sequence (up to 16K tokens),[7] and that (b) observing the entire inputs from SCROLLS improves performance.

5.4 Effect of Context Padding

In all experiments thus far, we used a conservative padding value $\rho = 0.5$, resulting in effective chunk size of $\frac{c}{2}$ and $\frac{c}{4}$ context padding tokens on each side. Since both memory and, more importantly, the number of forwards passes through the encoder are linear in the number of chunks, a natural question is how much padding and overlap are necessary to achieve satisfactory results.

To explore this, we finetune $\text{BART}_{\texttt{base}}$-SLED on all six datasets where SLED showed gains

[7]For ContractNLI, the length of over 95% of the tokenized examples is less than 8K.

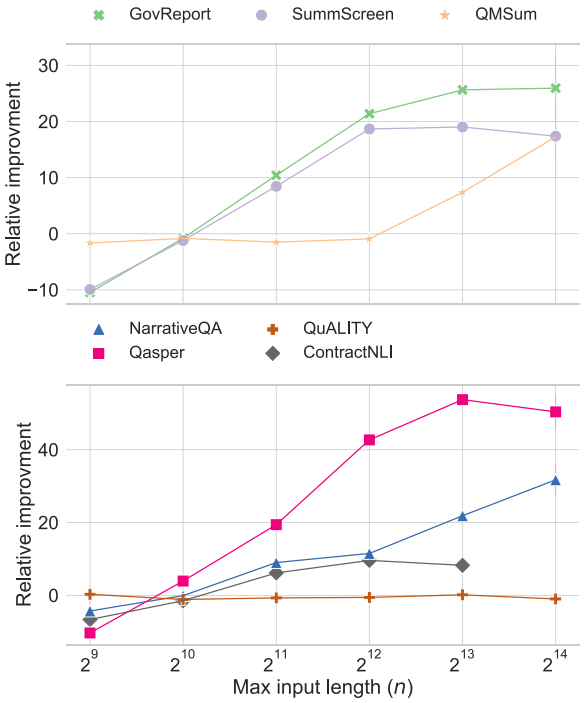

Figure 6: $\text{BART}_{\texttt{base}}$-SLED relative improvement compared to $\text{BART}_{\texttt{base}}$ results, when varying the input length fed to SLED, fixing $c = 256$. **Top**: Summarization datasets compared w.r.t. Rouge-2. **Bottom**: QA and NLI datasets. Relative improvment is measured w.r.t. exact match for QuALITY and ContractNLI and F_1 for NarrativeQA and Qasper.

over its baseline model (i.e., all datasets except for QuALITY), varying the value of ρ, and fixing $c = 256$. Table 2 shows the results of this experiment, where we compare relative gain compared to $\text{BART}_{\texttt{base}}$ across different ρ values.

As expected, decreasing the padding factor and consequently the number of chunks reduces training time. When $\rho = 0.05$ training can be faster by up to 2x compared to $\rho = 0.5$ as the number of chunks drops to almost half. Moreover, relative gain (i.e., improvement relative to the baseline) is often close to or even higher with less padding (perhaps due to better encoding or more stable optimization). Nevertheless, there is no single ρ value that consistently beats the conservative choice of $\rho = 0.5$. In particular, in all six datasets, setting $\rho = 0.5$ results in a top-2 performance, often by a large margin and never considerably worse then the best result. Thus, we conclude that one may improve the efficiency and performance of SLED by tuning the hyperparameter ρ for optimal behavior w.r.t. a specific task, and we fix $\rho = 0.5$ in our experiments.

ρ	Relative Gain					
	GovRep	SumScr	QMSum	Qspr	Nrtv	CNLI
50%	**34.1%**	**21.0%**	22.8%	53.7%	**34.2%**	8.9%
25%	28.5%	19.0%	17.9%	**54.7%**	29.4%	**10.1%**
5%	18.7%	15.9%	**23.5%**	52.0%	31.9%	7.4%
0%	27.1%	9.5%	11.5%	46.1%	29.2%	6.9%

Table 2: $\text{BART}_{\texttt{base}}$-SLED relative improvement compared to $\text{BART}_{\texttt{base}}$ when varying the padding percentage (ρ). In all cases the maximum input length is 16K and $c = 256$. Relative gain is measured w.r.t. Rouge-2 for GovReport, SummScreenFD, and QMSum, F_1 for Qasper and NarrativeQA and exact match for ContractNLI. In each column, **boldface** marks the top performing value and underline the second-best.

Moreover, Table 2 demonstrates the importance of having chunks at least partially overlapping. In all six dataset, using non-overlapping chunks ($\rho = 0$) results in a drop of at least 10% gain compared to the best setting, where in some cases this gap grows to over 50%. This supports our hypothesis that chunking inputs with no overlap may lead to crucial loss of information.

6 Related Work

Efficient Transformers Many efficient attention variants were proposed in recent years, to alleviate the quadratic complexity of dense attention (Tay et al., 2020; Fournier et al., 2021). Among those are clustering vectors to distinct buckets, calculating attention only within each one (Kitaev et al., 2020), attending only to a fixed number of hidden vectors (Ma et al., 2021), using random features to approximate the attention matrix (Choromanski et al., 2021; Peng et al., 2021), and using low-rank factorizations (Wang et al., 2020). Despite achieving respectable performance when finetuning these models on the Long Range Arena benchmark (Tay et al., 2021), many of them were not yet proven to work well as a backbone for pretrained language models. In fact, recent work (Xiong et al., 2022b) on encoder-only models found many do *not* outperform a simple local attention sliding window on downstream language tasks. We discuss such methods next.

Sparse Attention Variants A popular and simple solution for allowing attention-based models to process long sequences is to use local attention,

where each token attend to a local window around it. Longformer (Beltagy et al., 2020), GMAT (Gupta and Berant, 2020), and ETC (Ainslie et al., 2020) use short windows of full attention, combined with full attention to a small number of predefined global input tokens. BigBird (Zaheer et al., 2020b) shares the local and global features, and, additionally, randomly samples tokens to attend to. Finally, the recently proposed LongT5 (Guo et al., 2022) extends T5 (Raffel et al., 2020a) with local and global attention components based on ETC, relieving the need to manually specify global tokens. In this work, we demonstrate that a simple sliding window with off-the-shelf models without any modifications is a strong alternative for multiple generative tasks that require processing long documents.

Beyond Transformers As an alternative to transformers for processing long sequences, Gu et al. (2021) proposed the Structured State Space (S4) architecture showing dramatic gains over transformers on the LRA benchmark (Tay et al., 2021). State space models are now an active research field (Gupta, 2022; Mehta et al., 2022), but their efficacy on long-range language understanding tasks has not been tested yet.

Fusion-in-Decoder Izacard and Grave (2021) proposed to encode multiple independent passages separately, and concatenate the encodings prior to the decoding phase. Despite encouraging empirical evidence (Amouyal et al., 2022; Yavuz et al., 2022), we are the first (to our knowledge) to analyze FiD's feasibility and limitations in a controlled setting. Importantly, we test FiD on long-range tasks over a single long document, rather than a collection of independent passages.

Pretrained Models with Sliding Windows Wrapping a BERT encoder within a sliding window was proposed by Cui and Hu (2021) in the context of a specialized architecture for summarization. Wang et al. (2019) showed that sliding BERT across text improves performance on several QA datasets. In this work, we propose a sliding window approach that can be easily plugged into any existing encoder-decoder model without additional parameters or task-specific training, and show its efficacy for long-range text understanding. Most similar to SLED, is the SEGENC approach proposed by Vig et al. (2022). By dividing inputs from QMSum into overlapping chunks, encoding them separately, and then performing FiD (using two representations for every input token), the authors were able to achieve state-of-the-art results. However, Vig et al. (2022) were focused on summarization and did not perform a systematic analysis of this type of architecture.

7 Limitations

We present SLED as a simple and effective method to extend the capabilities of pretrained short-text models to long-text tasks. Despite its impressive empirical performance on SCROLLS, SLED suffers from two disadvantages which may limit its applicability to some long-range tasks.

Long Output To obtain linear complexity, SLED assumes the output length k is constant. This is because the decoder uses quadratic self-attention over the output, on top of $\mathcal{O}(nk)$ cross-attention between the output and input. While most current long-text tasks follow this assumption, future tasks, such as academic reports or script writing, may require long text generation. This limitation is not unique to SLED and affects other long-range transformers including LongT5 and LED. Aside from finetuning, this also affects pretraining models on long inputs with self-supervised losses such as span-corruption (Raffel et al., 2020b) or denoising (Lewis et al., 2020), which require the decoder to process an output that is linear in the length of the input.

Co-reference Resolution and Fact Retention An assumption at the heart of SLED is the *Locality of information assumption*. When the input text is long, this assumption may break if distant entity resolution or factual knowledge are required. For example, a chapter in a book may mention *''they were walking into the room''* when knowledge of what room or who walked is located a few chapters back. In such cases, the encoder used by SLED will not be able to access this information, moving more responsibility to the decoder and reducing the effectiveness of the contextual encoding. Similarly, in multi-hop questions (Yang et al., 2018), attending to one part of the context is necessary in order to fully understand the question and encode a second piece of information correctly. As the encoder will not have access to the first context that leads to better question understanding, here as well more responsibility is delegated to the decoder.

8 Conclusions

In this work we present SLED, a simple approach for modeling long texts that slides a pretrained short-range encoder over a long input document and then generates an output by attending to the encoded tokens. We show SLED can perform core operations that are important for long text understanding, such as finding relevant pieces of information and fusing them at decoding time, and demonstrate competitive performance on the SCROLLS benchmark compared to larger models and models that employ a dedicated and expensive pretraining step.

One of SLED's most attractive features is that it can be readily used with any short-range pretrained LM. Thus, any future encoder-decoder model can be flexibly plugged into it to achieve further gains in performance on SCROLLS, some of its tasks, or any other long-range task.

We open source SLED and hope it encourages the research community to easily extend to longer inputs and push the borders of natural language understanding models' applicability in real-world use-cases.

Acknowledgments

This research was partially supported by The Yandex Initiative for Machine Learning, the Shashua Fellowship, the Len Blavatnik and the Blavatnik Family Foundation, and the European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800). We would also like to thank our action editor and the anonymous reviewers for their insightful suggestions and feedback. This work was completed in partial fulfillment for the Ph.D. degree of the first author.

A SLED Implementation Details

While §3 details SLED's method it leaves out dealing with the edge tokens for brevity. Encoding the first and last $\frac{\rho \times c}{2}$ input tokens requires special attention, as they lack bidirectional context. To preserve as much commonality between chunks, all first $\frac{(2-\rho) \times c}{2}$ tokens are considered the *effective chunk* tokens in the first chunk. To account for the final tokens, the last chunk will always start at token t_{n-c+1} so it would contain exactly c tokens, and its *effective chunk* tokens will be defined as all tokens that were not part of any previous *effective chunk*.

B Chunking vs. Local-attention

Both LED and SLED are long-range models built on top of the same short-text model (BART), and employ local attention. However, SLED relies on chunking, while LED uses per-layer local attention. In this section, we now discuss in more detail the relation between the two approaches.

Implementation One of SLED's biggest advantages is that it is *agnostic* to the backbone encoder-decoder model, and can extend any existing model without additional implementation overhead. In contrast, The attention mechanism in Longformer, and subsequently LED, was implemented by Beltagy et al. (2020) with a specialized CUDA kernel that is coupled to the architecture and implementation of BART. This makes LED more efficient, but extending it to new architectures incurs significant engineering overhead. This is since LED uses a ''diagonal'' local-window attention across layers, for which a naïve implementation is inefficient. Conversely, SLED uses chunking, which allows to simply wrap an existing encoder-decoder model.

Contextualization The most significant difference between LED and SLED from a conceptual point of view is their contextualization mechanism. While SLED splits the input into (overlapping) chunks and encodes each of them independently, LED performs local attention *per-layer*. This results in an effective receptive field that grows linearly with the encoder depth, potentially allowing it to perform more ''global'' contextualization. Our results in §4 suggest that such global contextualization is beneficial, and a similar conclusion can be reached when observing that $\text{LED}_{\texttt{base}}$, which uses all prefix tokens as global tokens, outperforms $\text{LED}^{\texttt{SCROLLS}}_{\texttt{base}}$, which uses only a single token for global contextualization.

Positional Information SLED's chunking mechanism means that it utilizes the positional encoding of the underlying model independently in each chunk, and is thus agnostic to the positional embedding technique used by the backbone model. Moreover, it potentially allows SLED to generalize to arbitrary input lengths. In contrast, LED utilizes BART's absolute embeddings, duplicating them 16 times to support 16K-long sequences. This limits its ability to generalize

to longer inputs, and potentially induces a requirement for significant amounts of long-input samples to properly tune those new parameters (Shaham et al., 2022). This is evident in Table 1 when comparing tests scores of $\text{LED}_{\texttt{base}}$ against $\text{BART}_{\texttt{base}}$-SLED and considering the number for training samples. In NarrativeQA and GovReport, which contain ∼71K and ∼19K samples respectively, LED is comparable to SLED and even slightly outperforms it on some metrics. In ContractNLI (∼10K examples), it does slightly worse. In all other datasets, where the training data is small, LED is significantly worse than SLED.

Complexity We analyzed the complexity analysis of SLED's encoder (§3), which is $\mathcal{O}\left(l \times c \times n\right)$. A similar analysis of LED yields that in each layer, LED considers $\mathcal{O}(n)$ windows of length c, where in each window only the middle token attends to its local neighborhood, resulting in $\mathcal{O}\left(l \times c \times n\right)$ memory complexity as well. However, due to SLED's use of overlap and full self-attention within each chunk, SLED's encoding may require up 2x more memory compared to LED when $\rho = 0.5$.

C Experimental Details

Our experimental setup is based on the SCROLLS official repository.[8] The dataset inputs and splits remained as suggested by the authors of SCROLLS as well as the suggested number of epochs per dataset. To perform model selection, for each model-dataset pair we finetuned 9 models with LINEAR learning rate scheduling, AdamW optimizer with the default settings, and setting the learning rate to one of $\{2\text{e}{-5}, 5\text{e}{-5}, 1\text{e}{-4}\}$ and the effective batch size to one of $\{8, 16, 32\}$. Warmup was fixed at 10% and weight decay at 0.01. All code, data, Python environment requirements, hyperparameters, and scripts required to reproduce our results are available at `https://github.com/Mivg/SLED`.

References

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. https://doi.org/10.18653/v1/2020.emnlp-main.19

Samuel Amouyal, Ohad Rubin, Ori Yoran, Tomer Wolfson, Jonathan Herzig, and Jonathan Berant. 2022. Qampari: An open-domain question answering benchmark for questions with many answers from multiple paragraphs. *ArXiv*, abs/2205.12665.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

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. https://doi.org/10.18653/v1/2022.acl-long.589

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR*

[8] `https://github.com/tau-nlp/scrolls`.

2021, Virtual Event, Austria, May 3–7, 2021. OpenReview.net.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Peng Cui and Le Hu. 2021. Sliding selector network with dynamic memory for extractive summarization of long documents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5881–5891, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.470

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.365

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Minneapolis, Minnesota. Association for Computational Linguistics.

Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2021. A practical survey on faster and lighter transformers.

Albert Gu, Karan Goel, and Christopher R'e. 2021. Efficiently modeling long sequences with structured state spaces. *ArXiv*, abs/2111.00396.

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. https://doi.org/10.18653/v1/2022.findings-naacl.55

Ankit Gupta. 2022. Diagonal state spaces are as effective as structured state spaces. *ArXiv*, abs/2203.14343.

Ankit Gupta and Jonathan Berant. 2020. GMAT: Global memory augmentation for transformers. *ArXiv preprint*, abs/2006.03274.

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. https://doi.org/10.18653/v1/2021.naacl-main.112

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880. Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.74

Yichen Jiang and Mohit Bansal. 2019. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop QA. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2726–2736, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1262

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.

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.

Yuta Koreeda and Christopher Manning. 2021. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computaional Linguistics: EMNLP 2021*, pages 1907–1919, Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-emnlp.164

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. https://doi.org/10.18653/v1/2020.acl-main.703

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692.

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*.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2022. Long range language modeling via gated state spaces. *ArXiv*, abs/2206.13947.

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. https://doi.org/10.18653/v1/2022.naacl-main.391

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021*. OpenReview.net.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1264

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. https://doi.org/10.1162/tacl_a_00353

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. *ArXiv*, abs/2201.03533.

Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao,

Sharan Narang, Vinh Quang Tran, Dani Yogatama, and Donald Metzler. 2022a. Scaling laws vs model architectures: How does inductive bias influence scaling? *ArXiv*, abs/2207.10551.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena: A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021*. OpenReview.net.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey.

Yi Tay, Mostafa Dehghani, Vinh Quang Tran, Xavier García, Dara Bahri, Tal Schuster, Huaixiu Zheng, Neil Houlsby, and Donald Metzler. 2022b. Unifying language learning paradigms. *ArXiv*, abs/2205.05131.

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 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jesse Vig, Alexander Fabbri, Wojciech Kryscinski, Chien-Sheng Wu, and Wenhao Liu. 2022. Exploring neural models for query-focused summarization. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1455–1468, Seattle, United States. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-naacl.109

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

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5878–5882, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1599

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. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. 2022a. 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. https://doi.org/10.18653/v1/2022.naacl-main.144

Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. 2022b. 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. https://doi.org/10.18653/v1/2022.naacl-main.144

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1259

Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, Nitish Shirish Keskar, and Caiming Xiong. 2022. Modeling multi-hop question answering as single sequence prediction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 974–990, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.69

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020a. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020b. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.472