Structured Packing in LLM Training Improves Long Context Utilization

Konrad Staniszewski[1, 2], **Szymon Tworkowski**[1, 6], **Sebastian Jaszczur**[1, 2], **Yu Zhao**[3], **Henryk Michalewski**[1, 4], **Łukasz Kuciński**[1, 2, 5], **Piotr Miłoś**[1, 2, 5]

[1]University of Warsaw, Krakowskie Przedmiescie 26/28, 00-927 Warsaw, Poland
[2]IDEAS NCBR, Chmielna 69, 00-801 Warsaw, Poland
[3]University of Edinburg, Old College, South Bridge, Edinburgh EH8 9YL, United Kingdom
[4]Google DeepMind, 5 New Street Square, London, United Kingdom
[5]Institute of Mathematics Polish Academy of Sciences, Jana i Jedrzeja Sniadeckich 8, 00-656, Warsaw, Poland
[6]xAI, San Francisco Bay Area, California, U.S.
ks.staniszewski@uw.edu.pl

Abstract

Recent advancements in long-context language modeling have attracted significant attention, yet their practical applications often suffer from suboptimal context utilization. To efficiently address this issue, we introduce the Structured Packing for Long Context, SPLICE, a method that uses retrieval to collate mutually relevant documents into long training samples. We demonstrate that SPLICE improves performance on long-context tasks, particularly by achieving perfect accuracy on the synthetic Needle in the Haystack benchmark, and effectively mitigating the 'lost-in-the-middle' phenomenon often observed in large language models. Notably, these long-context capabilities also extend to realistic downstream tasks, such as Qasper, across multiple model sizes—3B, 7B, and 13B—and are achieved with only brief fine-tuning on 2-6 billion tokens. We supplement these results with a detailed analysis of SPLICE, examining the impact of hyperparameter choices, the different mixtures and proportions of SPLICE-generated training data, and the choice of the retriever. We also study the transfer of long-context utilization skills between the modalities. An intriguing finding from our analysis is that training on a corpus of code can enhance performance on natural language tasks.

Code — https://github.com/ideas-ncbr/publications_2024
Extended version — https://arxiv.org/abs/2312.17296

1 Introduction

Large language models (LLMs) (Brown et al. 2020; Chowdhery et al. 2022; Lewkowycz et al. 2022; OpenAI 2023a; Bai et al. 2023) have transformed the field of AI. Recently, the field has observed the rise of Long Context Language Models (LCLMs) that promise to unveil novel and powerful capabilities (Anthropic 2023; OpenAI 2023b; Gemini Team 2024). However, their ability to process long contexts is not always as effective as one hopes. Indeed, several studies have highlighted an important limitation: when processing prompts composed of multiple documents, LCLMs frequently encounter difficulties in accurately extracting relevant information (Tworkowski et al. 2023; Liu et al. 2023; Shi et al. 2023; Kamradt 2023). Additionally, they typically find it challenging to utilize information from the middle of their inputs (Liu et al. 2023), even on simple synthetic retrieval tasks (Li et al. 2023a). Understanding these issues is vital for advancements in LCLM technologies and calls for systematic research.

In this work, we take a step towards better context utilization in LCLMs. We focus on training data, keeping other components, such as the architecture and training objectives, unchanged. The broad question is: *Given training data consisting of documents, how should these documents be organized into training samples to enhance long-context capabilities*? While this perspective has received some attention recently (Levine et al. 2022; Chan et al. 2022; Shi et al. 2024), the problem remains unsolved. The central finding of this work is that *structuring training data to increase semantic interdependence is an effective strategy towards better long context utilization*. We achieve this by introducing and evaluating Structured Packing for Long Context (SPLICE), a method for creating training samples by using retrieval (e.g., BM25, Contriever) to collate mutually relevant documents into a single training context.

We empirically validate SPLICE showing that fine-tuning of OpenLLaMA 3Bv2, 7Bv2 (Geng and Liu 2023) and CodeLlama 13B (Rozière et al. 2023) with mere 2B–6B tokens already brings improvements in handling long context information in downstream tasks that require retrieval and in-context learning. These tasks include Qasper (Dasigi et al. 2021) from SCROLLS (Shaham et al. 2022), HotPotQA (Yang et al. 2018), Needle In A Haystack (Kamradt 2023), TREC (Li and Roth 2002; Hovy et al. 2001), and DBpedia (Lehmann et al. 2015). We also show that SPLICE significantly alleviates the 'lost-in-the-middle' phenomenon (Liu et al. 2023) and outperforms standard example packing on the Needle In A Haystack task (Kamradt 2023) (see Figure 1). We perform a comprehensive study of the design choices and properties of SPLICE, showing, in particular, that the acquired long context capabilities transfer between modalities, such as code and text. SPLICE also helps to retain and in some cases even improve performance on short context benchmarks like GSM8K (Cobbe et al. 2021), MMLU (Hendrycks et al. 2021) and TREC (Li and Roth 2002; Hovy et al. 2001).

Our contributions can be summarized as follows:

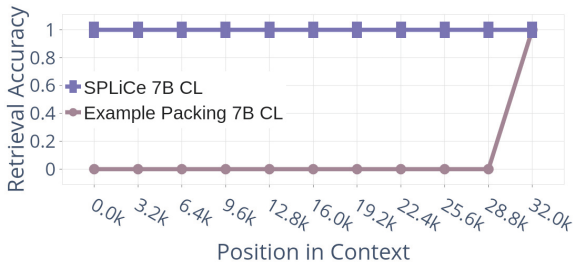

Figure 1: SPLICE vs EXAMPLE PACKING (EP) (BASELINE) on Needle in a Haystack. A model fine-tuned with SPLICE achieves perfect accuracy in retrieving fine-grained information over the whole context, while the baseline can only handle a small final segment (details in Appendix N).

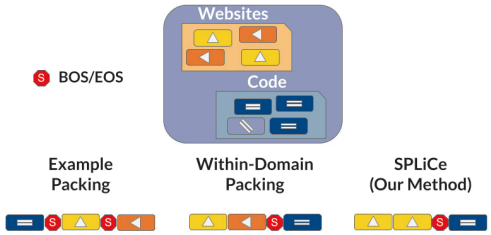

Figure 2: Training samples generated by Example Packing, Within-Domain Example packing, and SPLICE. Similar colors and shapes indicate related documents, which could be found using a retrieval method (e.g., BM25 or Contriever) or metadata (e.g., git repository structure).

- We comprehensively show that structuring training data is a viable way of improving the long context utilization. To this end, we introduce SPLICE, a method for creating training samples by using retrieval to collate mutually relevant documents into a single sample.
- We fine-tune OpenLLaMA 3Bv2, OpenLLaMA 7Bv2 (Geng and Liu 2023) and CodeLlama 13B (Rozière et al. 2023) using SPLICE, showing that it improves long-contex downstream performance.
- We provide a comprehensive analysis of SPLICE's design choices, including retrieval parameters and document concatenation order, and evaluate its robustness and scalability with varying data sources and a parametrizable noisy retriever.

2 Method

SPLICE is a method for constructing training samples that improve the effectiveness of long-context fine-tuning. This leads to improved performance in tasks such as in-context learning, question answering, information retrieval, and long-context language modeling (see Section 3).

Rationale and Intuitions Capturing long-range dependencies is believed to enhance language modeling and retrieval-augmentation (Borgeaud et al. 2022). It is an open question how to achieve such benefits in pre-training or fine-tuning. The primary difficulty comes from long-range dependencies being rare in training data (de Vries 2023) and diminishing with distance. Thus, it is unlikely that a model will learn to utilize long context without more guidance.

Recent studies indicate that structuring data, i.e., going beyond the i.i.d. paradigm, might be beneficial or even necessary to achieve good long-context performance. (Levine et al. 2022) develops a theory showing that the trained model establishes stronger dependencies between text segments in the same training sample. Whereas concurrently to our work (Shi et al. 2024) shows that pre-training on structured data improves performance (see also Section 4 for a more detailed comparison). SPLICE follows these intuitions, and constructs training samples by concatenating mutually relevant documents to increase the dependency density, thus allowing the model to learn to utilize long context.

Structured Packing for Long Context (SPLICE) SPLICE starts by picking a random document from the dataset to create a root of a tree and continues in a breadth-first manner, each time appending top-k similar documents from the corpus. The final sequence is generated by flattening the tree according to a specific traversal strategy; see Algorithm 1. The hyperparameter k introduces flexibility, enabling interpolation between different retrieval modes. Specifically, when $k = 1$, SPLICE simulates a long document by creating a path of related examples. For larger k values, SPLICE generates examples akin to those used in retrieval-augmented models, e.g. (Borgeaud et al. 2022).

SPLICE Retrieval Many possible retrieval methods can be used with SPLICE (RETRIEVE function in Algorithm 1). In our experiments, we test the following:

- SPLICE REPO: based on additional meta-information about the data, that is the repository structure of the code (REPO): we concatenate files using a depth-first search algorithm on the directory structure, that is files from the same directory are grouped together. A similar method has been pioneered by (Wu et al. 2022) and proposed in (Shi et al. 2024) as an interesting future direction.
- SPLICE BM25: based on BM25 (Robertson and Zaragoza 2009; Bassani 2023), a standard retrieval method that uses a bag-of-words approach to rank documents based on their similarity to a query.
- SPLICE CONT: based on Contriever-MSMARCO (CONT) (Izacard et al. 2022), a retrieval method that uses a transformer to rank documents based on their similarity.

SPLICE Computational Efficiency Given the dataset sizes used in training LLMs, computational efficiency plays a crucial role. SPLICE REPO is the fastest and easiest to implement but requires additional directory structure, i.e., it does not apply to general web data. SPLICE BM25 uses a bag

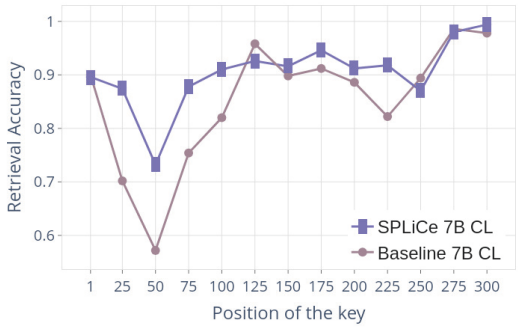

Figure 3: Key-value retrieval performance on a dictionary of 300 key-value pairs ($\approx$ 24K tokens). The 7B CL model trained with SPLICE achieves much higher accuracy on hard-to-retrieve positions in the middle than the Example Packing Baseline. The details about this task can be found in Appendix D. Each position averaged over 500 examples.

Algorithm 1: SPLICE training sample construction

Input:
D: document corpus
k: breadth hyper-parameter
L: maximum length of returned training sample
RETRIEVE: retrieval method to use, e.g., BM25
ORDER: ordering method, e.g., identity, or shuffle
Output: training sample
$d_r \sim D$ {Sample the root document}
$D = D \setminus \{d_r\}$
$C = [d_r]$
$Q =$ `empty queue`
Q.PUSH(d_r)
while $Q \neq \emptyset$ and len(C) $\leq L$ **do**
  $d = Q$.POP()
  $d_1, \ldots, d_k =$ RETRIEVE(d, k)
  {Retrieve top-k most similar documents to d using a selected method, e.g., BM25}
  for each d_i in $d_1, \ldots, d_k$ **do**
    if $d_i \in D$ **then**
      {RETRIEVE uses a precomputed index and may return documents that are already in C}
      $C = C$.APPEND(d_i) {Append d_i to C}
      Q.PUSH(d_i)
      $D = D \setminus \{d_i\}$
    end if
  end for
end while
return CONCAT(TRIM(ORDER(C), L))

of words BM25 method that lacks deeper semantic encoding. However, it was observed to have strong generalization properties (Thakur et al. 2021). SPLICE CONT requires calculating embeddings for each document and retrieval based on the vector inner-product, but can have poorer generalization properties than BM25 (Thakur et al. 2021). The retrieval step can be done efficiently using a fast approximate max IP search, e.g., Faiss (Johnson, Douze, and Jégou 2017). To reduce the number of times the training sample requires just copy-paste abilities and improve training step efficiency, we employ the StarCoder (Li et al. 2023b) dataset, which was deduplicated using the pipeline from (Allal et al. 2023).

3 Experiments

In this section, we show that SPLICE improves the long context performance of large-scale language models. To this end, we use 3B, 7B, and 13B parameter models. First, in Section 3.3, we focus on tasks that test in-context learning, question answering, and in-context information retrieval. Next, we show that SPLICE can improve the core model capabilities by testing its short context performance. Finally, in Section 3.4, we train over 40 medium-size models (270M parameters) using different data mixtures and SPLICE parameters to analyze various design choices, robustness to noise, and scaling properties.

An important finding of our work is that presented improvements occur during a relatively short fine-tuning. To be more precise, 3B models were tuned on 5.4B tokens, whereas 7B and 13B models were tuned on 2B tokens.

3.1 Baselines

We consider two popular baselines used in LLM training pipelines. The first one is Example Packing (Brown et al. 2020), used in the training of GPT-3 models. It constructs training samples by randomly sampling documents from the dataset and separating them with BOS/EOS tokens. The second one, which we call Within-Domain Example Packing takes random documents from the same meta-class (for example, Wikipedia, C source code) and concatenates them to create a training sample (Groeneveld et al. 2024; Zhao et al. 2024). We compare SPLICE against both baselines. As we note no clear benefit of Within-Domain Example Packing over Example Packing in fine-tuning case (see Table 21 in Appendix B.3) in the main body of the paper we compare only against a more established Example Packing. We visualize the differences between baselines in Figure 2.

3.2 Experimental Setup

For 3B model experiments, we fine-tune on a 50/50 mixture of RedPajama, prepared in the standard way, and C prepared using SPLICE BM25. For 7B and 13B ones, we fine-tune on a 50/25/25 mixture of RedPajama (50) prepared in the standard way, StackExchange (25) and C (25) prepared using SPLICE BM25. StackExchange is part of RedPajama (TogetherComputer 2023), and C data come from StarCoder (Li et al. 2023b). Including the standard RedPajama aims to

prevent the model from overfitting to artificially created documents and is inspired by (Ouyang et al. 2022; Rozière et al. 2023). We analyze the impact of data mixture in Section 3.4.

We fine-tune with 32K context length. We employ the Focused Transformer (FoT) (Tworkowski et al. 2023) and CodeLlama (CL) context extension methods (Rozière et al. 2023). We use a batch size of 256K (512K, resp.) tokens per step for 3B and 7B (13B, resp.) models. We set the learning rate of $1.5\text{e}{-5}$ with linear warmup and cosine decay, following (Geng and Liu 2023). In the next section, we test **eight models**:

$$\{\text{3B FoT, 7B FoT, 7B CL, 13B FoT }\} \times \{\text{SPLICE, EP}\},$$

where EP denotes the standard Example Packing (Brown et al. 2020) method (serving as baseline) where context is created by sampling random documents from the corpus and separating them with `BOS`/`EOS` tokens (see Figure 2). We provide results regarding the Within-Domain Example Packing baseline in Appendix C.2. If not stated otherwise, in SPLICE we use $k = 1$ and the identity permutation as `Order` in the Algorithm 1. Hyperparameter details can be found in Appendix A.

3.3 Experimental Results

In-Context Learning In this section, we ask the following research questions: *Does SPLiCe improve in-context learning abilities? If so, with what context length is it the case?* To answer those questions, we evaluate the accuracy of our models on TREC (Li and Roth 2002; Hovy et al. 2001) and DBpedia (Lehmann et al. 2015), which are text classification tasks. For TREC we test $\{2, 16, 32\}$K context lengths, which correspond to $\{90, 780, 1560\}$ in context examples, respectively. For DBpedia, we test $\{16, 32\}$K context lengths, which correspond to $\{190, 380\}$ in-context examples, respectively, and omit the 2K length due to its limited capacity of 20 in-context examples. For each context length, we average the results across several sets of in-context examples and provide average improvement of SPLICE and its 95% bootstrap confidence interval (improvements are calculated per set of in-context examples, see Appendix H). **In both tasks and all considered context lengths, we note that SPLICE significantly improves in-context learning abilities in comparison to both Example Packing and the starting checkpoint.** We hypothesize that by increasing the amount of potentially relevant information in context, SPLICE allows the model to learn longer and better context lookups. We further study this in Section 3.5, where we analyze SPLICE using the framework from (Chan et al. 2022). We present the main results in Tables 1, 2 and additional in Table 35 in Appendix O. Additionally in Appendix H we analyze results per-set of in-context examples and show that SPLICE achieves stochastic domination over Example Packing.

Question Answering and In-Context Retrieval In this section, we ask the following research question: *Does fine-tuning on SPLiCe prepared data result in improved question-answering abilities?* To answer the question, we utilize popular long context benchmarks such as Needle In A Haystack (Kamradt 2023) and lost-in-the-middle key-value retrieval (Liu et al. 2023), along with Qasper (Dasigi et al. 2021) from SCROLLS (Shaham et al. 2022), HotPotQA (Yang et al. 2018) passkey (Mohtashami and Jaggi 2023) and parts of RULER (Hsieh et al. 2024) tasks.

TREC				
Model	Context	EP	SPLICE	Δ [conf interv]
$3B_{FoT}$	32K	73.9	**79.3**	5.4 [4.7, 6.2]
	16K	68.9	**76.9**	8.0 [6.9, 9.3]
$7B_{FoT}$	32K	75.6	**79.4**	3.8 [2.1, 5.1]
	16K	74.0	**79.0**	5.0 [3.4, 6.0]
$7B_{CL}$	32K	75.3	**76.6**	1.3 [0.8, 1.8]
	16K	81.4	**82.5**	1.1 [0.2, 1.6]
$13B_{FoT}$	32K	89.2	**92.4**	3.2 [2.6, 3.8]
	16K	88.2	**91.2**	3.0 [1.9, 3.5]

Table 1: We test the classification accuracy on TREC (Li and Roth 2002; Hovy et al. 2001). We average across 50 sets of in-context examples for 3B models, 10 for 7B models, and 5 for 13B models. Δ [ci] denotes the mean improvement and its 95% bootstrap confidence intervals (see Appendix H).

DBpedia				
Model	Context	EP	SPLICE	Δ [conf interv]
$3B_{FoT}$	32K	82.9	**85.9**	3.0 [2.6, 3.6]
	16K	79.1	**82.0**	2.9 [2.5, 3.4]
$7B_{FoT}$	32K	82.9	**84.9**	2.0 [1.5, 2.4]
	16K	83.6	**85.6**	2.0 [1.5, 2.5]
$7B_{CL}$	32K	95.1	**95.6**	0.5 [0.3, 0.6]
	16K	96.2	**96.4**	0.2 [0.0, 0.3]
$13B_{FoT}$	32K	95.6	**96.0**	0.4 [0.0, 0.8]
	16K	95.8	**96.8**	1.0 [0.6, 1.5]

Table 2: We average results across 40 sets of in-context examples for 3B and 7B models and 5 for 13B. Due to the size of the DBpedia dataset, for each set of in-context examples, we sample a subset of 500 elements of the evaluation set.

On Needle In A Haystack, we observe that the model fine-tuned on data prepared by SPLICE strongly outperforms the model fine-tuned on data prepared by Example Packing. To be more precise **model fine-tuned with SPLICE can answer the question no matter the location of the relevant piece of information.** Whereas the model trained with Example Packing only manages to answer correctly when the information is close to the question (see Figure 1). We also test our models on the lost-in-the-middle key-value retrieval task (Liu et al. 2023), and observe that SPLICE helps on hard-to-retrieve positions (see Figure 3). The main difference between those two tasks is that in the lost-in-the-middle key-value retrieval task, the input is highly structured (dictionary of random 128 bit UUIDs, see Appendix D for de-

Model	MMLU			GSM8K
	STEM	HUM	All	
7B ST CHKP	**33.6**	**42.1**	**40.8**	**8.0**
7B 2K TUNED	**33.7**	40.8	39.4	**8.4**
$7B_{FoT}$ SPLICE	31.0	35.6	36.3	**7.6**
$7B_{FoT}$ EP	30.1	36.8	36.2	6.7
$7B_{CL}$ SPLICE	**32.7**	38.8	36.5	5.9
$7B_{CL}$ EP	**32.7**	37.5	36.1	6.3
13B ST CHKP	36.6	**48.2**	**44.3**	21.4
$13B_{FoT}$ SPLICE	**38.3**	**48.6**	**44.4**	**23.1**
$13B_{FoT}$ EP	**39.0**	47.5	**44.7**	21.9

Table 3: We evaluate our models on MMLU (5-shot), GSM8K (8-shot CoT). We provide an additional comparison with their starting checkpoint. For the 7B case, we additionally compare with a model tuned with 2k context length on the same data. For each task, we highlight the best results up to 1 point. For 3B model results see Appendix I.

tails) and the objective of the model is to retrieve the value assigned to a given key. On the other hand, in the Needle In A Haystack, a piece of information is placed inside a large coherent text, and the model is asked a question related to this information (see Appendix N for details).

We additionally evaluate our models on Qasper (Dasigi et al. 2021), HotPotQA (Yang et al. 2018) passkey retrieval (Mohtashami and Jaggi 2023) and RULER (Hsieh et al. 2024) and observe that SPLICE results in improvements over both the Example Packing and starting checkpoint. We present the results in Appendix K.

Short Context Evaluation One challenge in long-context fine-tuning is the degradation of short-context performance (Dubey, Jauhri, and et al. 2024). This can be overcomed by upsampling the short-context data and more gradual context extension (Dubey, Jauhri, and et al. 2024). We note that those approaches are compatible with SPLICE and instead focus on comparing SPLICE with Example Packing in a single-step context extension setup. We observe that SPLICE seems to be either better or on par with Example Packing (see Table 3). What is intriguing is that for the 13B parameter model, SPLICE even improves the short context performance on GSM8K (Cobbe et al. 2021) by $(+1.7)$ over the starting checkpoint. We hypothesize that GSM8K is a much more attention-demanding task than MMLU (Hendrycks et al. 2021), as it requires extracting relevant pieces of information, composing a chain of thought, and writing the final answer. Whereas MMLU is a well-established collection of tests spanning multiple domains. We hypothesize that the improvement does not occur in smaller models due to their low scores on GSM8K, as we get similar results when evaluating on code in Appendix L.

3.4 Detailed Study with Medium Models

In Table 4, Table 5, and Table 8, we present a comprehensive examination of the impact of document packing on long-context performance using 270M parameter models, showing that SPLICE brings consistent improvement in long context language modeling. In Table 6, we scale context to 64K and observe even greater benefits over the Example Packing. We also expand our results to 131K and 160K context length in Tables 19 and 20 in Appendix. In Table 7, we show that SPLICE is quite robust to the non-accurate retriever. What is more results in Table 7 clearly show that SPLICE is an improvement over Within-Domain Example Packing. In particular, we note that the more noise we add, the closer SPLICE is to Within-Domain Example Packing (semantically), and that with 100% noise SPLICE turns into Within-Domain Example Packing (this is because we use SPLICE to prepare data coming from a single domain).

Training and Evaluation Initially, we train with the 2K context length on 6.3B tokens from RedPajama (TogetherComputer 2023). Subsequently, we fine-tune using 1B tokens with the context extended to 32K on a mixture of the original RedPajama data (TogetherComputer 2023) and long context data created using SPLICE/EP. We employ the Focused Transformer (FoT) (Tworkowski et al. 2023) for context extension (unless stated otherwise). We measure perplexity on held-out portions of the arXiv (Azerbayev, Piotrowski, and Avigad 2022) and StarCoder (Li et al. 2023b) datasets. The selection of these datasets is motivated by the fact that they can benefit from long-context information as demonstrated in (Chen et al. 2023; Li et al. 2023b). We provide additional details in Appendix M.

3.5 Properties of SPLICE Generated Data

SPLICE conjecturally falls into the framework presented in (Chan et al. 2022), which shows that the distributional properties of the training data affect the in-context capabilities of transformer models. In particular, it indicates the importance of "burstiness", i.e., a flatter frequency distribution with a relatively higher mass on the rare, long-tail tokens appearing in a sequence. In Table 9, we show that SPLICE increases the burstiness of the training data (measured in terms of Zipf's coefficient of token frequency) in comparison to the Example Packing.

4 Related Work

There is an increasing number of works aiming to study the role of data in LLM training in detail. For instance, (Levine et al. 2022) developed a theory and demonstrated empirically that incorporating non-adjacent but semantically related sentences in training samples leads to better sentence embeddings and improves open-domain question-answering performance. Another study by (Gu et al. 2023) introduced a pretraining framework grounded on the idea that text documents often include intrinsic tasks. They showed that this approach substantially boosts in-context learning. Additionally, there is existing work on training long-context language models using repository-level code data, such as (Wu et al. 2022). Work of (Chan et al. 2022) identifies the training data's distributional properties that affect transformer models' in-context capabilities. Similarly, (Han et al. 2023) constructs small-scale data using an iterative gradient approach and shows that such data improve in-context performance.

Altered Data	Method	arXiv	Code				Code & arXiv
			Haskell	Python	CUDA	All	
C#	SPLICE BM25	**5.52** (.13)	**3.33** (.25)	**2.90** (.17)	**2.46** (.19)	**3.11** (.20)	**3.26** (.20)
	SPLICE CONT	5.53 (.12)	3.35 (.23)	2.91 (.16)	2.48 (.17)	3.12 (.19)	3.27 (.19)
	SPLICE REPO	5.53 (.12)	3.35 (.23)	2.91 (.16)	2.49 (.16)	3.12 (.19)	3.27 (.19)
	EP	5.65	3.58	3.07	2.65	3.31	3.46
Python	SPLICE BM25	**5.47** (.10)	**3.25** (.21)	**2.53** (.09)	**2.41** (.15)	**3.02** (.15)	**3.17** (.15)
	SPLICE CONT	5.49 (.08)	3.28 (.18)	**2.53** (.09)	2.43 (.13)	3.03 (.14)	3.19 (.13)
	SPLICE REPO	5.48 (.09)	3.27 (.19)	2.54 (.08)	2.44 (.12)	3.03 (.14)	3.18 (.14)
	EP	5.57	3.46	2.62	2.56	3.17	3.32
Wikipedia	SPLICE BM25	**5.64** (.09)	**3.82** (.15)	**3.26** (.11)	**2.87** (.13)	**3.55** (.13)	**3.68** (.13)
	SPLICE CONT	5.65 (.08)	3.87 (.10)	3.30 (.07)	2.92 (.08)	3.59 (.09)	3.72 (.09)
	EP	5.73	3.97	3.37	3.00	3.68	3.81
StackEX	SPLICE BM25	**5.07** (.07)	**3.88** (.06)	**3.32** (.04)	**2.89** (.05)	**3.60** (.05)	**3.69** (.05)
	SPLICE CONT	5.09 (.05)	3.91 (.03)	3.35 (.01)	2.93 (.01)	3.63 (.02)	3.73 (.01)
	EP	5.14	3.94	3.36	2.94	3.65	3.74

Table 4: Perplexity with an improvement over EP highlighted in the subscript:$_{\text{(imp over EP)}}$. We fine-tune a 270M parameter model with 32K context on a 50/50 mixture of RedPajama (organized in a standard way) and long-context data C#, Python, Wikipedia, StackExchange prepared using a method of choice (SPLICE BM25, SPLICE CONT, SPLICE REPO, EP). EP denotes organizing long-context data in the same way as RedPajama. SPLICE beats the EP often by a large margin. The variants of SPLICE perform similarly, with SPLICE BM25 being slightly better. For detailed results, see Appendix B.3.

Long Context Data	Method	arXiv	Code		Code & arXiv
			Python	All	
C	SPLICE BM25	**5.463** $\pm$ .002	**2.810** $\pm$ .002	**2.942** $\pm$ .005	**3.100** $\pm$ .004
	SPLICE CONT	5.477 $\pm$ .005	2.824 $\pm$.001	2.957 $\pm$ .006	3.115 $\pm$ .006
	SPLICE REPO	5.474 $\pm$ .007	2.827 $\pm$ .006	2.958 $\pm$ .009	3.115 $\pm$ .009
	EP	5.550 $\pm$ .002	2.931 $\pm$ .008	3.073 $\pm$ .006	3.228 $\pm$ .005

Table 5: Perplexity fine-tune on a $50/50$ data mixture of RedPajama and C code. We report the mean and standard deviation. Interestingly, training on the code data with SPLICE improves general long-context performance on arXiv.

Altered Data	Method	arXiv	Code				Code & arXiv
			Haskell	Python	CUDA	All	
C#	SPLICE BM25	**4.86** (.15)	**2.60** (.19)	**2.66** (.16)	**2.32** (.19)	**2.75** (.19)	**2.88** (.19)
	SPLICE CONT	4.88 (.13)	2.62 (.17)	2.67 (.15)	2.34 (.17)	2.77 (.17)	2.90 (.17)
	SPLICE REPO	4.88 (.13)	2.62 (.17)	2.68 (.14)	2.35 (.16)	2.77 (.17)	2.90 (.17)
	EP	5.01	2.79	2.82	2.51	2.94	3.07

Table 6: Perplexity $_{\text{(imp over EP)}}$ for training on a $50/50$ data mixture of RedPajama and C# code with longer 64K context.

Method		SPLICE						EP
Eval Data/Noise		**0%**	**10%**	**25%**	**50%**	**75%**	**90%**	**-**
arXiv		**5.46**	5.47	5.48	5.50	5.53	5.55	5.55
Code	Python	**2.81**	2.82	2.83	2.86	2.89	2.92	2.93
	All	**2.94**	2.95	2.97	3.01	3.04	3.06	3.07
Code & arXiv		**3.10**	3.11	3.13	3.16	3.19	3.22	3.23

Table 7: We test the robustness of SPLICE to noisy retriever. We achieve this by preparing data using BM25 retriever that with probability p returns a random document instead of the most related one. We note that SPLICE is quite robust and only with $p = 0.9$ approaches Example Packing.

Altered Data	Method	arXiv	Code		Code arXiv
			Python	All	
25%	SPLICE	**5.43**	**2.91**	**3.08**	**3.22**
25%	EP	5.49	3.00	3.19	3.34
50%	SPLICE	**5.46**	**2.81**	**2.94**	**3.10**
50%	EP	5.55	2.93	3.07	3.23
75%	SPLICE	**5.61**	**2.80**	**2.86**	**3.05**
75%	EP	5.73	2.94	3.04	3.23

Table 8: We note that SPLICE beats the EP perplexity when trained with various proportions of SPLICE BM25/EP prepared C data (the remaining data is unaltered RedPajama).

Training Data	Method	Zipf's Coefficient
C	SPLICE BM25	$1.512_{(0.055)}$
	EP	$1.593_{(0.025)}$
StackEx	SPLICE BM25	$1.643_{(0.026)}$
	EP	$1.664_{(0.013)}$

Table 9: Zipf's coefficient of token frequency on EP and SPLICE along with standard deviation. A lower Zipf's coefficient represents a more significant burstiness property.

Our methodology diverges from these works in several key ways. First, while prior studies have focused on sentence-level (Levine et al. 2022) or paragraph-level (Gu et al. 2023) granularity, we emphasize document-level context during training, specifically targeting long-context performance. We validate our approach in large-scale language modeling, using models such as OpenLLaMA 3B, 7B, and CodeLlama 13B. Second, we construct a tree structure of related documents using BM25/Contriever-MSMARCO retrieval, which we then linearize to form long-context samples. This approach allows for greater control over the coherence of samples, compared to relying solely on natural data structures like repository-level code. While the gradient-based method in (Han et al. 2023) shares similarities with our retrieval-based approach, our method scales to larger datasets and operates at a different granularity.

Concurrently with our research, (Shi et al. 2024) introduced a method for preparing training data that shares similarities with SPLICE, particularly in its default settings ($k = 1$ with identity as `Order`). However, while their approach focuses on training models from scratch, our work demonstrates that long-context capabilities can be effectively achieved through short and cost-efficient fine-tuning. In addition to this distinction, we employ significantly longer context lengths, extending above 64K tokens compared to the 8K tokens used in (Shi et al. 2024), which allows for more comprehensive context handling. Furthermore, we provide an in-depth analysis of our design choices, such as the advantages of data reordering (detailed in Appendix J) and the impact of varying k values (Appendix C.1). These analyses underline the effectiveness and flexibility of our approach. Our findings are especially pertinent in the context of recent research on the "Physics of Language Models" (Allen-Zhu and Li 2024), which discusses the limitations of fine-tuning. Despite these limitations, we show that SPLICE offers substantial and quantifiable improvements even with relatively brief fine-tuning, providing a practical advantage in enhancing long-context capabilities.

5 Limitations and Future Work

We show that structuring the training data is a viable way of improving the model's long-context performance. The presented method, SPLICE, can be viewed as a general framework for organizing the documents into training samples. This opens multiple further research avenues.

Retrieval Granularity Another avenue for future work is to study the granularity of the pieces from which the training samples are constructed. In this work, we focus on the document-level granularity. However, it is possible to construct training samples from smaller pieces.

Other Data Sources One of the approaches to training long-context language models is to use conversational data (Li et al. 2023a), which is complementary to our method. SPLICE can utilize data that already exists in vast quantities and can be easily applied to different types of text (like code, Wikipedia articles, or StackExchange) to further increase the number of long-context samples. We leave researching how SPLICE integrates with other methods for preparing the long-context data as future work.

Data Curation Using highly correlated samples has the potential to result in training instability. However, we noted no performance degradation during our experiments. We leave the study of how SPLICE integrates with different data types for the future. In particular, in our studies, the datasets used were reasonably deduplicated.

Neural Retriever In our work, we have utilized Contriever (Izacard et al. 2022) in a zero-shot setup, using the first 512 tokens to generate the embedding. On the other hand, BM25 had access to all the document content. Further study is required to determine whether SPLICE can additionally significantly benefit from properly tuned neural retrievers. In particular, in our case, Contriever tended to produce samples consisting of fewer repositories than BM25. We leave this for future work.

6 Conclusions

In this work, we present SPLICE, a method of constructing training samples for long-context language models. It utilizes BM25/Contriever-MSMARCO to find relevant documents and feed them to the model in a structured manner. We show that SPLICE improves performance on downstream tasks and the language modeling abilities of LLMs. We further show that SPLICE can be used to improve long-context utilization of large-scale models using only short fine-tuning. We believe that our work indicates multiple interesting research directions for improving the performance of long-context language models with structured data.

Ethical Statement

Our work develops a generic technique that allows for improving context utilization in language models via low-cost fine-tuning. However, we note that it does not create any new threats, but only exacerbates existing ones. Therefore, we refer to the existing literature on the broader impact of language models, such as (Borgeaud et al. 2022).

Acknowledgments

We are thankful for the TPU Research Cloud program, which was instrumental to our research by providing significant computational resources. Parts of the project were realized using the resources of IDEAS NCBR.

References

Allal, L. B.; Li, R.; Kocetkov, D.; Mou, C.; Akiki, C.; Ferrandis, C. M.; Muennighoff, N.; Mishra, M.; Gu, A.; Dey, M.; Umapathi, L. K.; Anderson, C. J.; Zi, Y.; Poirier, J. L.; Schoelkopf, H.; Troshin, S.; Abulkhanov, D.; Romero, M.; Lappert, M.; Toni, F. D.; del Río, B. G.; Liu, Q.; Bose, S.; Bhattacharyya, U.; Zhuo, T. Y.; Yu, I.; Villegas, P.; Zocca, M.; Mangrulkar, S.; Lansky, D.; Nguyen, H.; Contractor, D.; Villa, L.; Li, J.; Bahdanau, D.; Jernite, Y.; Hughes, S.; Fried, D.; Guha, A.; de Vries, H.; and von Werra, L. 2023. SantaCoder: don't reach for the stars! arXiv:2301.03988.

Allen-Zhu, Z.; and Li, Y. 2024. Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. arXiv:2309.14316.

Anthropic. 2023. Model Card and Evaluations for Claude Models. Technical report, Anthropic.

Azerbayev, Z.; Piotrowski, B.; and Avigad, J. 2022. ProofNet: A Benchmark for Autoformalizing and Formally Proving Undergraduate-Level Mathematics Problems. In *Advances in Neural Information Processing Systems 35, 2nd MATH-AI Workshop at NeurIPS'22*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; Hui, B.; Ji, L.; Li, M.; Lin, J.; Lin, R.; Liu, D.; Liu, G.; Lu, C.; Lu, K.; Ma, J.; Men, R.; Ren, X.; Ren, X.; Tan, C.; Tan, S.; Tu, J.; Wang, P.; Wang, S.; Wang, W.; Wu, S.; Xu, B.; Xu, J.; Yang, A.; Yang, H.; Yang, J.; Yang, S.; Yao, Y.; Yu, B.; Yuan, H.; Yuan, Z.; Zhang, J.; Zhang, X.; Zhang, Y.; Zhang, Z.; Zhou, C.; Zhou, J.; Zhou, X.; and Zhu, T. 2023. Qwen Technical Report. arXiv:2309.16609.

Bassani, E. 2023. retriv: A Python Search Engine for the Common Man.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; van den Driessche, G.; Lespiau, J.; Damoc, B.; Clark, A.; de Las Casas, D.; Guy, A.; Menick, J.; Ring, R.; Hennigan, T.; Huang, S.; Maggiore, L.; Jones, C.; Cassirer, A.; Brock, A.; Paganini, M.; Irving, G.; Vinyals, O.; Osindero, S.; Simonyan, K.; Rae, J. W.; Elsen, E.; and Sifre, L. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 2206–2240. PMLR.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Chan, S.; Santoro, A.; Lampinen, A. K.; Wang, J.; Singh, A.; Richemond, P. H.; McClelland, J. L.; and Hill, F. 2022. Data Distributional Properties Drive Emergent In-Context Learning in Transformers. In *NeurIPS*.

Chen, S.; Wong, S.; Chen, L.; and Tian, Y. 2023. Extending Context Window of Large Language Models via Positional Interpolation. arXiv:2306.15595.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; and et al. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Dasigi, P.; Lo, K.; Beltagy, I.; Cohan, A.; Smith, N. A.; and Gardner, M. 2021. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. arXiv:2105.03011.

de Vries, H. 2023. In the long (context) run. Accessed: 2023-09-28.

Dubey, A.; Jauhri, A.; and et al., A. P. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Gemini Team, G. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530.

Geng, X.; and Liu, H. 2023. OpenLLaMA: An Open Reproduction of LLaMA.

Groeneveld, D.; Beltagy, I.; Walsh, P.; Bhagia, A.; Kinney, R.; Tafjord, O.; Jha, A. H.; Ivison, H.; Magnusson, I.; Wang, Y.; Arora, S.; Atkinson, D.; Authur, R.; Chandu, K. R.; Cohan, A.; Dumas, J.; Elazar, Y.; Gu, Y.; Hessel, J.; Khot, T.; Merrill, W.; Morrison, J.; Muennighoff, N.; Naik, A.; Nam, C.; Peters, M. E.; Pyatkin, V.; Ravichander, A.; Schwenk, D.; Shah, S.; Smith, W.; Strubell, E.; Subramani, N.; Wortsman, M.; Dasigi, P.; Lambert, N.; Richardson, K.; Zettlemoyer, L.; Dodge, J.; Lo, K.; Soldaini, L.; Smith, N. A.; and Hajishirzi, H. 2024. OLMo: Accelerating the Science of Language Models. arXiv:2402.00838.

Gu, Y.; Dong, L.; Wei, F.; and Huang, M. 2023. Pre-Training to Learn in Context. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 4849–4870. Toronto, Canada: Association for Computational Linguistics.

Han, X.; Simig, D.; Mihaylov, T.; Tsvetkov, Y.; Celikyilmaz, A.; and Wang, T. 2023. Understanding In-Context Learning via Supportive Pretraining Data. arXiv:2306.15091.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. arXiv:2009.03300.

Hovy, E.; Gerber, L.; Hermjakob, U.; Lin, C.-Y.; and Ravichandran, D. 2001. Toward Semantics-Based Answer Pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Hsieh, C.-P.; Sun, S.; Kriman, S.; Acharya, S.; Rekesh, D.; Jia, F.; Zhang, Y.; and Ginsburg, B. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models? arXiv:2404.06654.

Izacard, G.; Caron, M.; Hosseini, L.; Riedel, S.; Bojanowski, P.; Joulin, A.; and Grave, E. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Trans. Mach. Learn. Res.*, 2022.

Johnson, J.; Douze, M.; and Jégou, H. 2017. Billion-scale similarity search with GPUs. arXiv:1702.08734.

Kamradt, G. 2023. Needle In A Haystack - Pressure Testing LLMs.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; and Bizer, C. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2): 167–195.

Levine, Y.; Wies, N.; Jannai, D.; Navon, D.; Hoshen, Y.; and Shashua, A. 2022. The Inductive Bias of In-Context Learning: Rethinking Pretraining Example Design. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Lewkowycz, A.; Andreassen, A. J.; Dohan, D.; Dyer, E.; Michalewski, H.; Ramasesh, V. V.; Slone, A.; Anil, C.; Schlag, I.; Gutman-Solo, T.; Wu, Y.; Neyshabur, B.; Gur-Ari, G.; and Misra, V. 2022. Solving Quantitative Reasoning Problems with Language Models. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Li, D.; Shao, R.; Xie, A.; Sheng, Y.; Zheng, L.; Gonzalez, J. E.; Stoica, I.; Ma, X.; and Zhang, H. 2023a. How Long Can Open-Source LLMs Truly Promise on Context Length?

Li, R.; Allal, L. B.; Zi, Y.; and et al. 2023b. StarCoder: may the source be with you! *CoRR*, abs/2305.06161.

Li, X.; and Roth, D. 2002. Learning Question Classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2023. Lost in the Middle: How Language Models Use Long Contexts. *CoRR*, abs/2307.03172.

Mohtashami, A.; and Jaggi, M. 2023. Landmark Attention: Random-Access Infinite Context Length for Transformers. *CoRR*, abs/2305.16300.

OpenAI. 2023a. GPT-4 Technical Report. arXiv:2303.08774.

OpenAI. 2023b. New models and developer products. OpenAI Blog.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Robertson, S. E.; and Zaragoza, H. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4): 333–389.

Rozière, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Remez, T.; Rapin, J.; Kozhevnikov, A.; Evtimov, I.; Bitton, J.; Bhatt, M.; Ferrer, C. C.; Grattafiori, A.; Xiong, W.; Défossez, A.; Copet, J.; Azhar, F.; Touvron, H.; Martin, L.; Usunier, N.; Scialom, T.; and Synnaeve, G. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950.

Shaham, U.; Segal, E.; Ivgi, M.; Efrat, A.; Yoran, O.; Haviv, A.; Gupta, A.; Xiong, W.; Geva, M.; Berant, J.; and Levy, O. 2022. SCROLLS: Standardized CompaRison Over Long Language Sequences. arXiv:2201.03533.

Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E. H.; Schärli, N.; and Zhou, D. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 31210–31227. PMLR.

Shi, W.; Min, S.; Lomeli, M.; Zhou, C.; Li, M.; Lin, X. V.; Smith, N. A.; Zettlemoyer, L.; tau Yih, W.; and Lewis, M. 2024. In-Context Pretraining: Language Modeling Beyond Document Boundaries. In *The Twelfth International Conference on Learning Representations*.

Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; and Gurevych, I. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663.

TogetherComputer. 2023. RedPajama: An Open Source Recipe to Reproduce LLaMA training dataset.

Tworkowski, S.; Staniszewski, K.; Pacek, M.; Wu, Y.; Michalewski, H.; and Milos, P. 2023. Focused Transformer: Contrastive Training for Context Scaling. *NeurIPS 2023*.

Wu, Y.; Rabe, M. N.; Hutchins, D.; and Szegedy, C. 2022. Memorizing Transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 2369–2380. Association for Computational Linguistics.

Zhao, Y.; Qu, Y.; Staniszewski, K.; Tworkowski, S.; Liu, W.; Miłoś, P.; Wu, Y.; and Minervini, P. 2024. Analysing The Impact of Sequence Composition on Language Model Pre-Training. arXiv:2402.13991.