RE^{3}SYN: A Dependency-Based Data Synthesis Framework for Long-Context Post-training

Zhiyang Zhang[*1] **Ziqiang Liu**[*2] **Huiming Wang**[*†3] **Renke Shan**
Li Kuang[†1] **Lu Wang** **De Wen Soh**[3]
[1]Central South University
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]Singapore University of Technology and Design
zzy573678@csu.edu.cn zq.liu4@siat.ac.cn huiming_wang@mymail.sutd.edu.sg
kuangli@csu.edu.cn lwzzfzl@gmail.com dewen_soh@sutd.edu.sg

Abstract

An important trend in the realm of large language models (LLMs) is the development of longer context windows. However, training LLMs with long context windows to acquire the capability of effectively modeling lengthy inputs is often hindered by the scarcity of naturally long-context data. Existing methods for constructing long-context data by concatenating short documents have overlooked a crucial characteristic of long-context data quality, namely semantic dependency. In this paper, we propose a novel framework called **Re**trieval, Dependency **Re**cognition, and **Re**order for data **syn**thesis (**RE^{3}SYN**[1]), which leverages semantic similarity to retrieve relevant documents and form several batches. Within each batch, the framework comprehensively recognizes dependency and utilizes them, along with a reorder algorithm, to organize the short documents into coherent long-context data. Comprehensive experiments on multiple benchmarks indicate that the data generated by the RE^{3}SYN has longer dependencies and significantly enhances the model's long-context capabilities.

1 Introduction

The recent emergence of large language models (LLMs), such as the Flan series (Chung et al., 2022), LLaMA (Touvron et al., 2023a,b; Dubey et al., 2024) and ChatGPT (OpenAI, 2022; Achiam et al., 2023), has brought a paradigm shift in natural language processing (NLP) due to their impressive performance. Their remarkable understanding capabilities prompts more researchers to leverage these models in tackling more complex NLP challenges (Brown, 2020), such as book summarization or learning new tasks on the fly from many examples (Bai et al., 2023; Zhang et al., 2023), which also poses a great challenge to the ability of LLMs to process extremely long inputs.

Due to the fact that positional encodings of LLMs (e.g., Rotary Position Embeddings (RoPE) (Su et al., 2024)) usually exhibit weak generalization to sequences longer than those encountered during training, these LLMs mostly choose to extend the context length by extrapolating positional encodings and fine-tuning with a small amount of long-context data (referred to as Long-Context Post-training) (Chen et al., 2023b; de Vries, 2023).

Despite their exciting performance, natural long documents are often scarce and originate from specific domains (e.g., arXiv, Wikipedia, books, etc.) (Chen et al., 2024), and this uneven domain distribution of such data will lead to a degradation of the general capabilities of LLMs and affect their performance (Fu et al., 2024). On the other hand, studies by (Fu et al., 2023) et al. indicate that the quality of long document data is crucial for unlocking the long-context modeling capabilities of LLMs. (Chen et al., 2024) et al. further reveals that a significant characteristic of high-quality long documents is their stronger forward and backward dependencies. In summary, a promising direction is to leverage short documents to be organized into more coherent and higher-quality long documents.

Several data engineering approaches have been proposed in previous researches to concatenate short documents either randomly or based on similarities (Staniszewski et al., 2023; Shi et al., 2023; Zhao et al., 2024), as shown in Figure 1. However, these methods of concatenating short documents do not reliably generate dependencies, resulting in the creation of numerous pseudo-long documents and raising concerns about data quality. This highlights the need for a more robust framework capable of effectively recognizing dependencies between documents and constructing high-quality long context data for post-training.

[*]Equal contribution.
[†]Corresponding authors: Li Kuang and Huiming Wang.
[1]https://github.com/ZY0025/RE3SYN

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 31468–31480

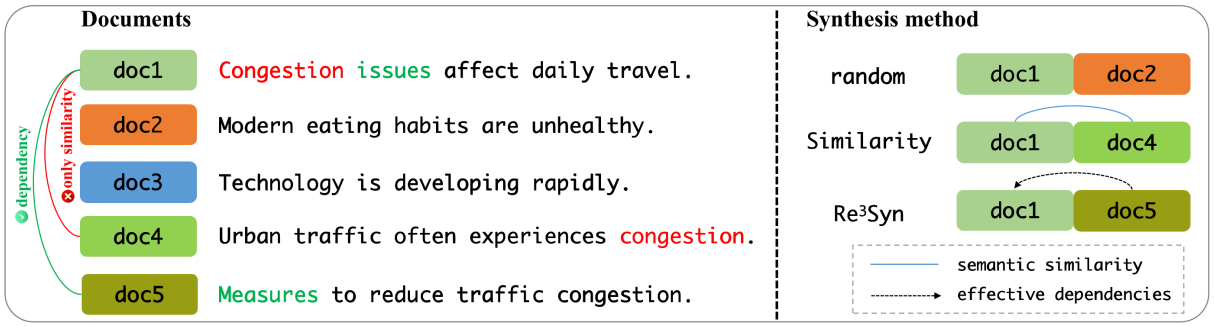

Figure 1: Overview of Different Synthesis Methods. Standard methods randomly mix documents into contexts, while similarity-based methods may group overly similar documents together, which can lead to document redundancy (Gao et al., 2024a) and result in low-quality pseudo-long documents. In contrast, RE^{3}SYN emphasizes the dependencies between documents, thus synthesizing higher-quality long documents with more coherent semantics.

Motivated by this, in this work, we incorporate dependencies into the synthesis of long context data (Fu et al., 2023; Borgeaud et al., 2022). The underlying principle of dependencies is that the lower the perplexity (ppl) on a pre-trained model for the target corpus, the more fluent the semantics of that corpus are, indicating higher quality. We provide a novel framework for synthesizing long context data, namely **Re**trieval, Dependency **Re**cognition, and **Re**order (**RE^{3}SYN**). RE^{3}SYN effectively recognize dependencies between short documents and concatenates them into long-context data of a specified length based on these dependencies.

The RE^{3}SYN framework consists of two main steps. Firstly, considering that the likelihood of dependencies between documents with low semantic similarity is also low, RE^{3}SYN connects non-redundant short documents based on semantic similarity and divides them into several batches for significant optimization of computational efficiency. Secondly, it evaluates the perplexity scores of different combinations of short documents within each batch using a small model to recognize dependencies, and then reorders the documents according to these dependencies. This ensures that the RE^{3}SYN framework can distribute dependencies throughout the long documents, enhancing the long-range modeling capabilities of LLMs. Further, we adopt various strategies to optimize the computational efficiency, including chunked computation and evaluating perplexity with small models. Through these two steps, we concatenate short documents into long documents with stronger dependencies.

Experimental results on multiple benchmarks and data evaluation frameworks indicate that the training data generated by RE^{3}SYN significantly outperforms current data synthesis method and enhances the LLM's long-context modeling capabilities. Highlights of our contributions are as follows:

• To the best of our knowledge, this is the first study to utilize dependency instead of similarity to synthesize high-quality long-context data.

• We propose a simple yet effective data synthesis framework, **RE^{3}SYN**, which organizes metadata into more meaningful long-context data with clearer preceding and following dependencies through similarity retrieval and document order adjustment.

• We balance the effectiveness and efficiency of the RE^{3}SYN framework, resulting in a data processing step overhead of approximately 10% and 5% throughout the post-training process for the 7B and 13B models, respectively.

• Compared to standard or similarity-based concatenation, the data obtained from RE^{3}SYN framework leads to models that demonstrate superior performance on both short-text and long-context benchmarks.

2 Related Work

2.1 Long-Context LLMs

There are primarily two types of methods for extending the context window of LLMs. The first category involves directly adjusting the LLM's positional encodings or attention matrices (Han et al., 2023; Xiao et al., 2023) to accommodate longer contexts without any additional fine-tuning. The second category builds upon this by post-training (Xiong et al., 2023), which typically results in better performance. Some studies, including Position Interpolation (Chen et al., 2023b), NTK-aware (bloc97, 2023), and YaRN (Peng

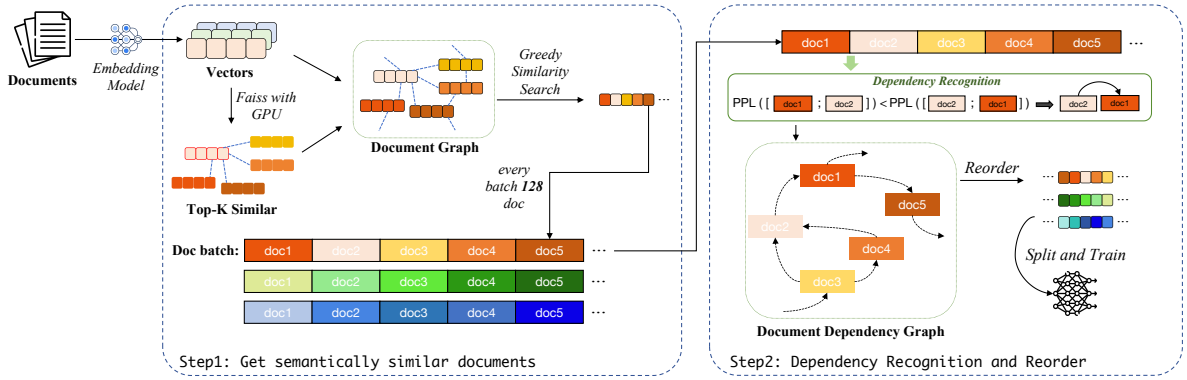

Figure 2: Framework of RE^{3}SYN.

et al., 2023), modify the Rotary Position Encoding (RoPE) and then perform post-training for a certain number of steps to expand the model's context window. MPT-7B-StoryWriter (Team et al., 2023) utilizes the books3 dataset as the training set, fine-tuning MPT-7B (Team et al., 2023) on sequences of length 65K. LongLoRA (Chen et al., 2023c) implements fine-tuning with LoRA (Hu et al., 2021), expanding LLama2 (Touvron et al., 2023b) from a 4k context to 100k.

However, as reported by de Vries (de Vries, 2023), for the post-training phase, the scarcity of long-context data presents the real challenge. As opposed to randomly stitching together unrelated short texts to reach a predetermined length (Chen et al., 2023c; Li et al., 2023; Chen et al., 2023a; Tworkowski et al., 2024), a viable direction is to concatenate metadata into higher-quality long texts.

2.2 Long-Context Data Engineering

Recently, many studies have explored the role of data engineering in LLM training. (Shi et al., 2023) proposed a similar method, ICLM, placing similar documents into the same pre-training window while ensuring that documents do not appear more than once, focusing on training from scratch. (Gao et al., 2024a) synthesizes long-context data by predicting queries for documents, extracting keywords, and concatenating them based on shared keywords.

However, compared to merely "similar" documents, recent studies have delved deeper into the definition of high-quality training data. (Chen et al., 2024) found that the characteristic of high-quality long-context data lies in the existence of stronger dependency, proposing a data filtering method called Prolong based on this insight. Although data quality is ensured, data filtering methods cannot resolve the issue of the shortage of long-context data.

Researchers have also attempted to construct training data with clearer sequential relevance using explicit hyperlinks between documents. (Yasunaga et al., 2022) merged Wikipedia documents containing hyperlinks or citations into the input context and pre-trained a masked language model, but as most data sources do not come with inherent hyperlinks, they are not a universal solution. Instead, we use the relative results calculated from perplexity between documents as an basis to judge the dependencies between documents, which provides an efficient and generalizable solution.

3 RE^{3}SYN Framework

Figure 2 illustrates the overall framework of RE^{3}SYN. For a given set of short documents, considering the low likelihood of dependency among documents with lower similarity, documents can first be clustered to reduce overhead. Specifically, RE^{3}SYN is accomplished through the following two steps: (1) Leveraging an embedding model and dense vector retrieval system, documents are retrieved and grouped into batches based on semantic similarity, with each batch maintaining a fixed size. (2) For each batch, an enhanced topological sorting algorithm is employed to reordering documents by analyzing their inter-dependency.

3.1 Document Clustering

As shown in the left half of Figure 2, to cluster documents based on similarity, we first obtain the similar documents for each document and then construct the document graph based on similarity. In this graph, each node corresponds to a document, and the edges between nodes represent the semantic similarities between the documents. Specifically, for any document d_i in $Docs$, we use the

Algorithm 1 Greedy Similarity Search

Input: Docs Graph $G = (N, E)$
  $N = \{d_i \in Docs\}$
  $Nbr(d_i)$ return the neighbors list of d_i
  $rand(\mathcal{S})$ return a random doc
Output: $\mathcal{L} : \{d_0, d_1, \ldots\}$
$\mathcal{S} \leftarrow$ set(N)
$\mathcal{L} \leftarrow []$
while $\mathcal{S}$ is not empty
  $d \leftarrow rand(\mathcal{S})$
  $\mathcal{L}.append(d)$
  $\mathcal{S}.remove(d)$
  while $Nbr(d) \cap S \neq \emptyset$
    $d \leftarrow Nbr(d)[0]$
    $\mathcal{L}.append(d)$
    $\mathcal{S}.remove(d)$
return $\mathcal{L}$

embedding model to generate the corresponding embedding v_i for each document. Next, we employ the dense vector retrieval tool FAISS (Johnson et al., 2019) to efficiently search in large batches for the top-k documents that have the highest semantic similarity to each document's vector v_i, which we denote as $Nbr(d_i)$. The documents in $Nbr(d_i)$ are arranged in descending order of similarity, ensuring that the most similar documents can be retrieved first during subsequent traversals.

Following this, we treat all documents as nodes $N = \{d_i \text{ in } Docs\}$ and establish the similarity relationships between documents as edges $E = \{(d_i, d_j) \text{ for } d_j \text{ in } Nbr(d_i)\}$ to create the document graph $G = (N, E)$. Previous studies (Shi et al., 2023; Gao et al., 2024a) have shown that identical training data can be detrimental to model performance. Hence, our goal is to traverse the document graph to obtain a sequence of documents that minimizes duplication while keeping similar documents as close as possible (Shi et al., 2023). Specifically, we can use a greedy similarity traversal algorithm to find an approximate solution in this scenario, which can be shown in Algorithm 1.

Initially, an arbitrary node from the current node set is selected to the target document sequence, and removed from the node set. Next, RE^{3}SYN traverses the neighboring nodes of this node until it finds the first neighbor that is not in the node set (i.e., an unvisited node) and continues to traverse. Once all neighboring nodes of the current node are in the node set, a new random node is selected from the set. This process continues until all nodes have been ordered. Through the greedy similarity traversal, we obtain a sequence of documents $\{doc_0, doc_1, ...\}$, in which similar documents are kept close together. This process is similar to the approach by (Shi et al., 2023; Staniszewski et al., 2023) in constructing long-context data based on document similarity, but with a suitably reduced granularity to improve efficiency. Next, we divide the document sequence into several batches, each containing 128 documents, for subsequent dependency recognition and reordering.

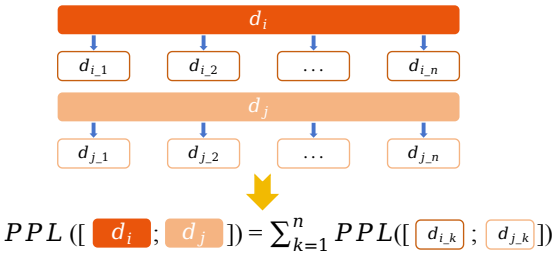

Figure 3: Calculate the ppl between two documents.

3.2 Dependency Recognition

As shown in the right half of Figure 2, for any two documents d_i and d_j within a batch, the dependency between d_i and d_j can be assessed using perplexity under different concatenation orders.

Specifically, if there is a dependency from d_j to d_i (i.e., it is more coherent for d_i to precede d_j), the $ppl([d_i; d_j])$ should be lower than $ppl([d_j; d_i])$, where $[d_i; d_j]$ indicates that the document d_i is concatenated directly before the document d_j. Therefore, we can select a smaller-sized language model to compute $ppl([d_i; d_j])$ and $ppl([d_j; d_i])$. If $ppl([d_i; d_j])$ is lower, we connect an edge from d_j to d_i to indicate a dependency between the two documents, and vice versa.

As shown in Figure 3, to mitigate the computation overhead associated with perplexity calculations, we employed a sampling and summation strategy. Specifically, we sampled n fixed-length, non-overlapping chunks from both d_i and d_j, then calculated the perplexity for n pairs of chunks in the specified order and summed them up to obtain $ppl([d_i; d_j])$ for that order.

3.3 Dependency-based Reorder

Through dependency recognition, we can determine the dependency relationships between any two documents within a batch, allowing us to create a document dependency graph $G_d = (N_d, E_d)$.

Algorithm 2 Dependency-based Reorder

Input: Document Dependency Graph $G_d = (N_d, E_d)$
  $N = \{d_i \in Docs\_batch\}$
  $Nbr(d_i)$ return the list of d_j if $(d_j, d_i) \in E_d$
  $otd(d_i)$ return the out-degree of d_i
  $cand\_nodes$ return nodes with out-degree 0
  $rm\_cycle(G)$ remove cycles from the graph
Output: $\mathcal{L} : \{d_0, d_1, \ldots\}$
$\mathcal{L} \leftarrow []$
$orig\_otd(d_i) \leftarrow otd(d_i)$
$rm\_cycle(G_d)$
while $|N_d| > 0$ **do**
  $d_i \leftarrow argmin_{d \in cand\_nodes} orig\_otd(d_i)$
  $\mathcal{L}.append(d_i)$
  for each $d_j \in Nbr(d_i)$ **do**
    $otd[d_j] \leftarrow otd[d_j] - 1$
  end for
end while
return $\mathcal{L}$

Where the nodes represent the documents in the current batch $N_d = \{d_i \text{ in } Docs\_batch\}$, and the dependency relationships between the documents serve as edges $E_d = \{(d_i, d_j, w_{ij}) \mid d_i, d_j \in N_b \wedge ppl([d_i; d_j]) < ppl([d_j; d_i])\}$, where $w_{ij} = \frac{\text{ppl}([\text{doc}_i;\text{doc}_j])}{\text{ppl}([\text{doc}_j;\text{doc}_i])}$ represents the relative size of the dependency between the two documents.

Our goal is to ensure that the concatenated documents retain only unidirectional dependency relationships after reordering, which can transform to a topological sorting problem (Kahn, 1962). The objective is to traverse the graph and ensure that for each directed edge (j, i), node i appears before node j in the ordering. We design an enhanced algorithm based on topological sorting, called Dependency-based Reorder tailored to our specific problem scenario, which is illustrated in Algorithm 2.

First, we traverse the graph, upon encountering cycles, we remove the edge with the smallest weight w_{ij} within the cycle (indicating the weakest dependency between the two documents) to eliminate the cycle, thus transforming it into an acyclic graph. Next, we obtain all nodes with an out-degree of zero and select the node with the highest original out-degree among them to add to the ordered sequence, while updating the out-degrees of its neighboring nodes. This process continues until all nodes are ordered. The motivation for selecting the node with the highest original out-degree is to prioritize those nodes that directly impact a greater number of other nodes, thereby facilitating a tighter arrangement of interdependent nodes.

Finally, we use the sequences obtained after dependency-based reorder to concatenate all documents and split them into fixed-size contexts for training.

4 Experiment Setup

4.1 Training Details

We conduct post-training on LLama2 (Touvron et al., 2023b) models with 7B and 13B. We applied an NTK-aware (bloc97, 2023) method with a base of 160,000 to adjust position indices, extending the model's capability to support a context window of 32K tokens. Both model variants were trained with the objective of next token prediction, with the learning rate set to 2e-5, no weight decay, and a single epoch, while the global batch size was set to 128. Additionally, we utilized over 20 steps of linear learning rate warm-up, along with the AdamW optimizer, where β_1=0.9 and β_2=0.95.

To enhance the reliability of our methods, multiple trials are necessary. Given the substantial computational resource cost of a round of experiments, we conducted a total of three rounds of experiments. The results reported in the paper are the averages of these rounds.

4.2 Training Data

We use the open-source dataset FineWeb (Penedo et al., 2024) for training, which contains cleaned and deduplicated English web data from CommonCrawl (Wenzek et al., 2019). It is worth noting that CommonCrawl accounts for more than 80% of LLaMA's training data (de Vries, 2023), which ensures that the data distribution in the pre-training stage and the post-training stage is basically consistent. We randomly sampled 40B tokens in total from the SAMPLE-350BT (Penedo et al., 2024) version of FineWeb. We used the RE^{3}SYN framework to construct our training data, which has a length of 32K. The training dataset remains completely consistent across all our experiments.

During the document clustering, we use the embedding model `all-MiniLM-L6-v2` (Reimers and Gurevych, 2019) to generate corresponding embeddings for each document, with a maximum retrieval number k = 10. To improve computational efficiency in the dependency recognition, we choose OPT-350m (Zhang et al., 2022) as the smaller lan-

Method	Test set	Avg.	Evaluation Context Window Size				
			2048	4096	8192	16384	32768
Standard	proofpile	3.47	3.92	3.54	3.38	3.29	3.25
ICLM	proofpile	3.16	3.81	3.35	3.05	2.85	2.74
RE^{3}SYN	proofpile	**3.11**	**3.74**	**3.30**	**3.01**	**2.82**	**2.71**
Standard	PG19	10.11	**10.86**	10.27	9.95	9.77	9.70
ICLM	PG19	10.17	11.69	10.66	9.93	9.41	9.18
RE^{3}SYN	PG19	**9.87**	11.08	**10.25**	**9.68**	**9.28**	**9.07**

Table 1: Sliding window perplexity (stride S=1024) of 32k-length documents truncated to evaluation context window size on different data systhesis method base on LLaMA2 7B.

guage model to calculate perplexity, which has been shown to maintain high accuracy with faster speed (Chen et al., 2024). The chunk sampling number and size are set to 4 and 128, respectively.

4.3 Baseline

We compared the RE^{3}SYN framework with two baseline methods: (1) The **Standard** randomly concat documents in the corpus until a preset context window is reached, after which it is truncated (Roziere et al., 2023; Chen et al., 2023c; Tworkowski et al., 2024; Li et al., 2023). (2) **ICLM** (Shi et al., 2023) concatenates the current document with the most similar documents in the corpus, using the traveling salesman algorithm to avoid duplicates. We use this open-sourced method to represent similarity-based data concatenation solutions of the same type. All methods are processed and synthesised on the same data, going through the same number of post-training steps to ensure consistent computational costs.

4.4 Task Setting

We comprehensively evaluate the model's performance under different data processing methods, which includes the following three tasks:

Language Modeling Task Language modeling task measures the perplexity of the LLM on the test set corpus, serving as one of the key indicators for assessing LLM performance. We follow (Chen et al., 2023c, 2024) to randomly sampled 128 documents from the total proof-pile (Azerbayev et al., 2022) and PG19 test split, and all perplexity calculations used a sliding window method with a stride of s = 1024.

Long-Context Task We used LongBench (Bai et al., 2023) to evaluate the LLM's performance on real-world long-context tasks, including single/multi-document question answering, summarization, code completion, etc. The average length of the tasks ranges from 5K to 15K, which poses a significant challenge for the LLM's understanding capabilities in ultra-long contexts. In addition, **Needle in a HayStack Task** is also an important benchmark for evaluating the retrieval ability of LLMs. The evaluation results of Needle in a HayStack Task can be found in AppendixA.2.

Standard Benchmark We assessed the LLM against a selection of standard short benchmarks from the HuggingFace Open LLM harness (Gao et al., 2024b), thereby validating its foundational comprehension and modeling skills.

5 Experiment Result

5.1 Language Modeling Task

We evaluated the language modeling perplexity of LLMs trained on concatenated data using the RE^{3}SYN and baseline methods. Results are shown in Table 1. Firstly, the perplexity of the RE^{3}SYN method is significantly better than that of the similarity concatenation method, indicating that the data constructed using the RE^{3}SYN method better stimulates the language modeling capabilities of LLMs. Additionally, we observed that on PG19, the performance of ICLM is slightly inferior to that of the Standard method. We speculate this is due to the concatenation of overly similar documents when the context length increases to 32k, leading to document redundancy and thus causing a decline in performance, and the dependency recognition and reorder of RE^{3}SYN avoid excessive redundancy and enhance the inter-document coherence.

5.2 Performance on LongBench

We use a real-world benchmark LongBench to assess models trained with different data synthesis

Method	Model size	Avg.	Sgl.	Multi.	Sum.	Few.	Syn.	Code.
Standard	7B	23.94	17.32	17.76	13.95	60.67	2.5	31.45
ICLM	7B	29.01	24.09	23.12	22.13	**66.71**	2.38	35.62
RE^{3}SYN	7B	**30.31**	**24.43**	**24.84**	**23.61**	66.27	**3.28**	**39.43**
Standard	13B	21.28	14.72	19.93	12.59	61.00	3.14	16.31
ICLM	13B	27.95	**22.28**	17.23	17.49	**66.29**	3.0	**41.46**
RE^{3}SYN	13B	**29.13**	21.49	**23.78**	**18.55**	66.27	**3.75**	40.95

Table 2: Experimental results of models with 32k context length on LongBench.

Method	Model size	Avg.	MMLU	Winogrande	Hellaswag	ARC_E	ARC_C
Standard	7B	0.622	0.411	0.652	0.774	0.773	0.501
ICLM	7B	0.627	0.411	0.671	**0.785**	0.771	0.499
RE^{3}SYN	7B	**0.634**	**0.422**	**0.672**	0.782	**0.783**	**0.512**
Standard	13B	0.670	0.500	**0.677**	0.805	0.810	0.558
ICLM	13B	0.671	0.501	0.664	**0.812**	**0.820**	0.565
RE^{3}SYN	13B	**0.674**	**0.502**	0.673	0.810	0.815	**0.569**

Table 3: Performance comparison of different models on different short text benchmarks.

methods, the average results of the six categories of tasks in LongBench are shown in Table 2. Detailed results can be found in Appendix A.1. From this, we can see that our method significantly outperforms other baseline methods overall. Specifically, on the 7B model, RE^{3}SYN improves by an average of 6.37% compared to standard method, and by an average of 1.3% compared to ICLM. On the 13B model, RE^{3}SYN improves by an average of 7.85% compared to standard method, and by an average of 1.18% compared to ICLM. These results demonstrate the effectiveness of using the RE^{3}SYN framework to concatenate metadata into longer dependency texts for training, suggesting that from a data perspective, it can further address real-world long-context tasks.

5.3 Standard Benchmark

We assessed LLMs corresponding to different data concatenation methods on several standard benchmarks from the Hugging Face Open LLM leaderboard, including MMLU (Hendrycks et al., 2020), Winogrande (Sakaguchi et al., 2021), Hellaswag (Zellers et al., 2019),ARC_Easy and ARC_Challenge (Clark et al., 2018). As shown in Table 3, we can see that LLMs obtained with RE^{3}SYN also maintain stronger performance on short benchmarks, outperforming baseline methods or being roughly on par. This is crucial because it indicates that the LLMs trained on concatenated texts with longer dependencies have not lost their short-text capabilities while improving long-context abilities, further validating the robustness of our method.

6 Analysis

6.1 Balance between Effectiveness and Efficiency

Considering that in practical applications, data processing steps should not incur excessive overhead, we provide some more efficient techniques, such as clustering in Section 3.1 and chunked computation in dependency recognition in Section 3.2, which help us complete data preparation more quickly.

Taking 40B training tokens and a 7B model as an example, we present the overhead (GPU·hours) and proportions of the three stages in the entire process in Table 4. Specifically, the three stages are as follows: **Clustering**: get semantically similar documents and greedy similarity search; **Reorder**: dependency recognition and reorder; **Training**: We perform post-training on the LLama2 7B using A800 GPUs. It can be seen that the costs of Clustering and Reorder are roughly equal. For the 7B model, these two steps account for 4.6% and 5.6% of the total cost, respectively, while the entire data processing constitutes about 10% of the whole process. For the 13B model, the costs of these two

Stage	GPU·Hour	Proportion
Clustering	312	4.6%
Reorder	384	5.6%
Training	6144	89.8%

Table 4: The proportion of each process overhead in the entire training phase.

Method	Prolong score
Standard	19.432
ICLM	64.741
RE^{3}SYN	**94.587**
RE^{3}SYN$_{knn}$	72.476
RE^{3}SYN$_{64doc}$	83.130

Table 5: The scores given by prolong under different data synthesis methods for the same original data.

stages will be reduced to 2.4% and 2.9%, respectively. As the model size increases, this proportion will decrease further, which is within an acceptable range. On the other hand, the synthesized data using our method can be reused repeatedly and even applied to tasks such as dialogue data construction, further ensuring the versatility of our approach.

6.2 Document Redundancy and Long-Dependency

Research has shown that concatenating documents based on similarity can lead to excessively similar content in long-context data, resulting in document redundancy, which can decrease model performance (Gao et al., 2024a). On the other hand, long dependencies have been proven to be an important indicator of the quality of long-context data (Chen et al., 2024). One important goal in the design of the RE^{3}SYN is to reduce redundancy and enhance the dependency relationships between documents. Specifically, the documents in the dataset we used have an average length of about 1.6k, and to concatenate 32k of data, approximately 20 documents are needed. We perform dependency recognition and reordering in batches of 128 documents, allowing the documents to recognize dependency relationships with other documents that are further apart (i.e., less similar), while effectively resolving overly redundancy arising from similarity searches.

To evaluate whether our method can ultimately lead to longer dependencies, we used the Prolong (Chen et al., 2024) framework to compute scores based on all data concatenated through standard, ICLM, RE^{3}SYN and its two variants: RE^{3}SYN$_{knn}$ (use knn (Guu et al., 2020; Levine et al., 2021) for document clustering) and RE^{3}SYN$_{64doc}$ (reorder using batches of 64 documents), taking the average. The results are shown in Table 5. We can see that the scores obtained using our method are significantly higher than both baseline methods, nearly five times the random concatenation score, and an improvement of 46% over similarity concatenation. This indicates that RE^{3}SYN surpasses current state-of-the-art methods in synthesizing long-range dependencies in long-texts data. Using knn for document clustering and reordering smaller batches both lead to a reduction in the final data dependencies. We believe this is because the one-to-many similarity search in knn increases document redundancy. This result also underscores the importance of maintaining larger batches to uncover more potential dependencies.

It is worth noting that, RE^{3}SYN do not set longer dependencies as an explicit synthesis goal, such as separating two documents with dependencies as much as possible. We believe that methods which select the next document based on the similarity of remaining candidate documents to the previous one, lead to associations existing among only nearby documents (Shi et al., 2023). In contrast, RE^{3}SYN calculates the dependencies between all documents to be connected and applies an improved topological sort to ensure that the overall sequence of documents remains more coherent, which is the source of the long dependencies in the data produced by RE^{3}SYN.

7 Conclusion

In this paper, we introduce a simple and effective data synthesis framework RE^{3}SYN aimed at establishing stronger dependencies, which recognize dependencies between discrete documents by recognizing dependencies on similar documents, resulting in more coherent long-context data through reordering. Extensive experiments on multiple benchmarks demonstrate that RE^{3}SYN can further enhance the long context modeling capability of LLMs. We believe this research can provide new insights for future researchers in the direction of high-quality data synthesis.

8 Limitations

The following are some limitations that the RE^{3}SYN may face: (1) We implemented document clustering for RE^{3}SYN using a simple and fast method, which could be improved by currently available more precise similar document retrieval methods. (2) Due to resource constraints, we validated the effectiveness of RE^{3}SYN on the LLama2-7B and LLama2-13B models, it is excepted to obtain better performance with more base-models. We will explore the above listed limitations further in future studies.

9 Ethical Statements

Our study do not carry any ethical concerns. Specifically, Our training data are publicly available and designated for research purposes only. We inspect our dataset to ensure it does not contain any unethical content, private information and offensive topics. Moreover, the base models we used are also publicly available for research purpose.

Acknowledgments

This work has been supported by the National Key R&D Program of China under grant No. 2022YFF0902500, the National Natural Science Foundation of China under grant No.62472447, the Science and Technology Innovation Program of Hunan Province under grant No.2023RC1023, Fujian Provincial Natural Science Foundation of China (No. 2024J08371), and Hunan Provincial Natural Science Foundation of China under grant No.2024JK2006.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. 2022. Proof-pile.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

bloc97. 2023. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

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

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023a. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*.

Longze Chen, Ziqiang Liu, Wanwei He, Yunshui Li, Run Luo, and Min Yang. 2024. Long context is not long at all: A prospector of long-dependency data for large language models. *arXiv preprint arXiv:2405.17915*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023c. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

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

Harm de Vries. 2023. In the long (context) run.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Fu, Xinyao Niu, Xiang Yue, Rameswar Panda, Kim, Yoon, and Hao Peng. 2023. Understanding data influence on context scaling.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

Chaochen Gao, Xing Wu, Qi Fu, and Songlin Hu. 2024a. Quest: Query-centric data synthesis approach for long-context scaling of large language model. *arXiv preprint arXiv:2405.19846*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024b. A framework for few-shot language model evaluation.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Arthur B Kahn. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Yoav Levine, Noam Wies, Daniel Jannai, Dan Navon, Yedid Hoshen, and Amnon Shashua. 2021. The inductive bias of in-context learning: Rethinking pretraining example design. *arXiv preprint arXiv:2110.04541*.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2023. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*.

OpenAI. 2022. Introducing chatgpt.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

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

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Victoria Lin, Noah A Smith, Luke Zettlemoyer, Scott Yih, and Mike Lewis. 2023. In-context pretraining: Language modeling beyond document boundaries. *arXiv preprint arXiv:2310.10638*.

Konrad Staniszewski, Szymon Tworkowski, Sebastian Jaszczur, Henryk Michalewski, Łukasz Kuciński, and Piotr Miłoś. 2023. Structured packing in llm training improves long context utilization. *arXiv preprint arXiv:2312.17296*.

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

MosaicML NLP Team et al. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Linkbert: Pretraining language models with document links. *arXiv preprint arXiv:2203.15827*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Lei Zhang, Yunshui Li, Ziqiang Liu, Junhao Liu, Min Yang, et al. 2023. Marathon: A race through the realm of long context with large language models. *arXiv preprint arXiv:2312.09542*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. 2024. Analysing the impact of sequence composition on language model pretraining. *arXiv preprint arXiv:2402.13991*.

A Experimental Details

A.1 32K Longbench Results

Table 6 and Table 7 report the detailed performance of different LLM on Longbench subtasks.

Method	Model size	Single-Doc QA			Multi-Doc QA			Code Completion	
		NarrativeQA	Qasper	MultiFieldQA	HotpotQA	2WikiMultihopQA	MuSiQue	LCC	RepoBench-P
Standard	7B	16.48	15.64	19.85	24.06	20.46	8.78	34.35	28.55
ICLM	7B	19.53	19.97	32.79	32.71	21.27	15.39	38.41	32.84
RE^{3}SYN	7B	19.95	19.82	33.51	34.88	26.8	12.84	42.82	36.03
Standard	13B	16.03	15.96	12.18	30.06	20.28	9.47	16.72	15.89
ICLM	13B	15.02	24.89	26.93	29.02	14.49	8.19	41.15	41.77
RE^{3}SYN	13B	12.06	23.44	28.97	36.31	21.92	13.1	42.16	39.73

Table 6: Evaluation Results on LongBench subtasks.

Method	Model size	Few-shot			Summarization			Synthetic Tasks	
		TriviaQA	SAMSum	TREC	GovReport	QMSum	MultiNews	PassageRetrieval	PassageCount
Standard	7B	79.54	38.47	64.0	11.66	20.5	9.7	4.5	0.5
ICLM	7B	86.22	42.93	71.0	28.98	21.11	16.31	4.25	0.5
RE^{3}SYN	7B	86.47	42.71	69.5	29.02	21.44	20.36	5.0	1.55
Standard	13B	81.87	37.65	63.5	10.72	19.25	7.8	5.27	1.0
ICLM	13B	83.73	41.65	73.5	25.25	23.77	3.45	5.0	1.0
RE^{3}SYN	13B	82.92	42.38	73.5	26.02	23.78	5.84	6.0	1.5

Table 7: Evaluation Results on LongBench subtasks.

Furthermore, table 8 and table 9 report the evaluation results for LongBench-E. LongBench-E is designed with uniform length-based sampling, ensuring a comparable amount of test data across the 0-4k, 4-8k, and 8k+ length intervals. This makes it more suitable for evaluating a model's ability to handle variations in input lengths.

Method	Model size	Avg.	passage_count	samsum	lcc	trec	2wikimqa	multi_news
ICLM	7B	27.60	1.00	39.99	43.25	67.33	9.62	13.20
RE^{3}SYN	7B	**28.51**	2.98	40.35	47.61	67.33	10.03	17.42
ICLM	13B	28.49	3.05	40.93	43.26	73.67	10.22	7.33
RE^{3}SYN	13B	**29.46**	4.26	40.08	50.35	70.33	11.84	8.28

Table 8: Evaluation Results on LongBench-E.

A.2 Needle in a HayStack Task

Needle in a Haystack Task This task places a random statement in the middle of a long context window, requiring the model to retrieve and output this text, evaluating the LLM's retrieval ability in long contexts. We set the maximum context length to 32K. The evaluation results for the LLMs corresponding to the three methods are shown in Figure 4. RE^{3}SYNobtains 94% needle in a haystack accuracy across all tested depths and context length, representing a significant improvement over the 86% accuracy of the ICLM method, and far surpassing the random synthesis method. This further highlights that using our method to synthesis long-dependency data for LLM training can enhance the retrieval capabilities of LLMs.

A.3 32K RULER Result

This task(Hsieh et al., 2024) offers a more comprehensive evaluation of long-context language models than simple retrieval-based benchmarks. It comprises a diverse set of tasks designed to probe a model's true long-context understanding and reasoning capabilities, including multihoptracing, aggregation, and

Method	Model size	triviaqa	multifieldqa	hotpotqa	gov_report	repobench	passage_retrieval	qasper
ICLM	7B	85.32	16.67	10.68	29.37	27.85	6.50	8.30
RE^{3}SYN	7B	82.21	16.71	11.37	28.57	31.34	6.11	8.63
ICLM	13B	84.45	14.84	10.62	25.77	39.85	8.11	8.26
RE^{3}SYN	13B	84.42	16.03	11.57	26.92	37.50	13.39	8.02

Table 9: Evaluation Results on LongBench-E.

Method	Model size	Avg.	4K	8K	16K	32K
Standard	7B	66.7	82.1	71.4	62.4	50.7
ICLM	7B	71.4	85.5	**76.6**	65.4	57.9
RE^{3}SYN	7B	**73.8**	**87.0**	**76.2**	**69.7**	**62.1**
Standard	13B	67.7	82.7	72.3	63.8	51.8
ICLM	13B	72.6	86.9	76.7	68.5	58.2
RE^{3}SYN	13B	**74.7**	**88.2**	**77.8**	**69.5**	**63.3**

Table 10: Experimental results of models with 32k context length on RULER.

complex question answering, rather than merely assessing its ability to locate specific information. We evaluated our methods in 32K RULER, and the detailed results are presented in Table 10. Our proposed RE^{3}SYNconsistently outperforms both the standard and ICLM methods in all context lengths tested (4K, 8K, 16K, and 32K).

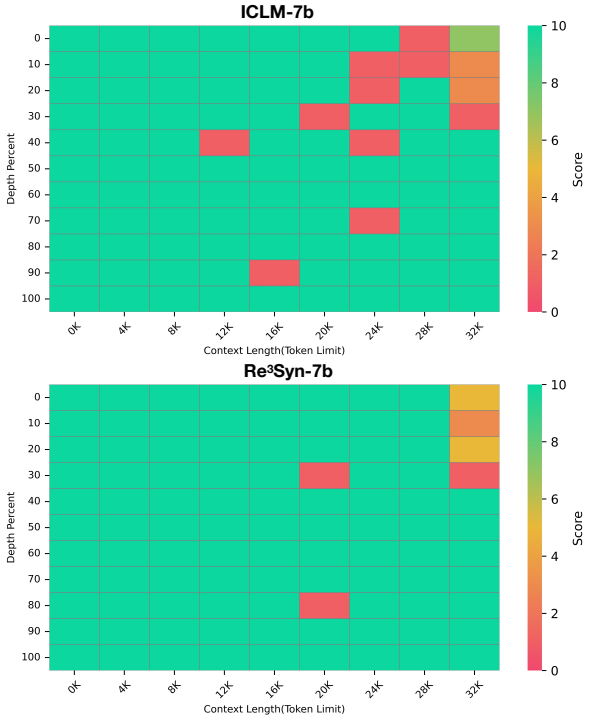

Figure 4: Performance of ICLM-7B and RE^{3}SYN-7B in Needle In A Haystack.