

NEAR: Neighborhood-Aware Context Reconstruction for Ultra-Long Document Reading Comprehension of LLMs

Han Li^{1,2,*}, Xuanang Chen^{2,*}, Ben He^{1,2,†}, Le Sun²

¹University of Chinese Academy of Sciences, Beijing, China

²Chinese Information Processing Laboratory, Institute of Software,
Chinese Academy of Sciences, Beijing, China

lihan263@mailsucas.ac.cn, chenxuanang@iscas.ac.cn

benhe@ucas.ac.cn, sunle@iscas.ac.cn

Abstract

Large language models (LLMs) remain challenged by ultra-long document understanding. A common solution is to segment documents into chunks and retrieve those relevant to a query, but this can disrupt semantic and logical coherence by ignoring dependencies between chunks. We propose NEAR (NEighborhood-Aware context Reconstruction), a lightweight, training-free framework that reconstructs coherent context from fragmented segments. NEAR represents each document as a dual-layer neighborhood graph, connecting chunks via topical neighbors to capture long-range semantic relations and physical neighbors to preserve local continuity. An iterative dual-pool exploration traverses chunks from both global and local neighborhoods, and collected chunks are selectively expanded and reordered according to their original positions to form a coherent input context. Extensive experiments on multiple long-document question answering benchmarks demonstrate that NEAR improves comprehension performance while maintaining low latency and memory usage.

1 Introduction

Recent advances in large language models (LLMs) have led to strong performance across a wide range of language understanding and reasoning tasks (DeepSeek-AI, 2025). As context windows expand to hundreds of thousands or even one million tokens in recent models (Yang et al., 2025; xAI, 2026), LLMs are theoretically capable of processing entire books or long reports in a single pass. Reading comprehension over ultra-long documents is becoming increasingly important in real-world applications such as legal analysis (Guha et al., 2023), scientific literature review (Taylor et al., 2022), and financial auditing (Wu et al., 2023).

*Equal contribution.

†Corresponding author.

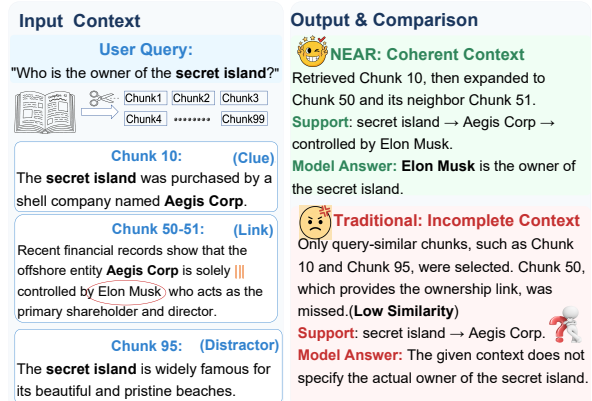


Figure 1: Illustration of information fragmentation in long-document reading comprehension. Chunking and conventional chunk selection often result in the isolation of related information, causing fragmented context. NEAR reconstructs coherent supporting context by leveraging topical neighbors to connect related content and physical neighbors to maintain local continuity.

However, effectively understanding such long documents remains challenging in practice.

Existing approaches for long-document processing generally fall into two categories. One line of research extends Transformer architectures to support longer context windows (Ding et al., 2024). While this increases input capacity, it incurs substantial computational costs due to the quadratic complexity of self-attention. Moreover, such models remain susceptible to issues such as attention dilution and positional bias (Jiang et al., 2024; Liu et al., 2024). In practice, a more economical and widely adopted solution is the chunk-based paradigm (Lewis et al., 2020). Under this paradigm, extremely long documents are segmented into smaller chunks, and only those most relevant to the query are selected for further processing. This strategy significantly reduces computational overhead and scales more naturally to extremely long documents (Sarathi et al., 2024).

However, this chunk-based paradigm inherently

fragments the internal structure of documents. In practice, segmentation and selection are treated as separate stages. Chunking, whether fixed-length or dynamic (Duarte et al., 2024; Sheng et al., 2025), splits continuous context into isolated segments, while selection mechanisms, ranging from lexical retrievers to fine-tuned cross-encoders (Li et al., 2024b), rank chunks independently and do not model their interdependencies (Li et al., 2024c). Consequently, semantic and logical relations across distant parts of a document are weakened, and the resulting input often degenerates into disjoint fragments that lack global coherence, ultimately impairing model performance (Edge et al., 2024).

To address this fragmentation issue, we propose NEAR (NEighborhood-Aware context Reconstruction), a lightweight, training-free framework that dynamically reconstructs coherent contexts from initially fragmented chunks without aggressive information compression. The core principle is that relevant content tends to cluster in semantic space (Hearst and Pedersen, 1996). Although a document is split into discrete chunks, its semantic organization still retains an inherent topology, including both strong local associations and long-range topical dependencies.

NEAR models the document as a dual-layer neighborhood graph that simultaneously captures topical and physical associations. The topical layer connects topically related but spatially distant chunks to establish long-range dependencies, while the physical layer preserves sequential adjacency to maintain local narrative coherence. Based on this representation, NEAR introduces an iterative neighborhood exploration mechanism that combines global semantic exploration with local continuity restoration. This design enables the reconstruction of a semantically coherent context by restoring local continuity and reconnecting related content across chunk boundaries.

Experiments on six long-document question answering benchmarks covering diverse QA settings demonstrate the effectiveness of NEAR, including datasets such as Loogle-SD (Li et al., 2024a) and HotpotWikiQA (Yang et al., 2018; Yuan et al., 2024). We evaluate NEAR with multiple backbone LLMs, including Llama-3 (Llama Team, 2024) and Mistral (Jiang et al., 2023), and compare it against strong baselines spanning extended-context architectures and chunk-based retrieval pipelines. NEAR consistently improves answer accuracy, with particularly strong gains on longer documents,

while introducing only minimal latency and memory overhead. These results indicate that NEAR enhances long-document comprehension while remaining lightweight and easy to integrate into existing long-context pipelines.

The main contributions of this work are three-fold: (1) We propose a dual-layer neighborhood graph that captures both topical relatedness and positional continuity among text chunks, providing a structured foundation for context reconstruction in long documents. (2) We present NEAR¹, a lightweight and training-free framework for reconstructing coherent reading context from fragmented chunks, which can be readily integrated into existing long-context pipelines. (3) Experiments on multiple long-document question answering benchmarks show that NEAR consistently improves answer accuracy while maintaining low latency and memory overhead.

2 Related Work

Context Length Extension Extending context windows to accommodate ultra-long documents is a major direction in LLM research. Peng et al. (2024) introduced YaRN, which utilizes NTK-by-parts interpolation to efficiently extrapolate context length while preserving local semantics, a technique now widely adopted in models such as Qwen2 (Yang et al., 2024). Pushing this direction further, Qwen2.5-1M (Yang et al., 2025) achieves a context window of one million tokens via optimized RoPE base frequencies and sparse attention, maintaining high accuracy in passkey retrieval. However, while these methods increase input capacity, they remain computationally expensive and do not address how to organize relevant content into a coherent reading context for downstream comprehension (Bai et al., 2024).

Chunking Strategies Document chunking provides a practical alternative to LLM’s context length extension for processing ultra-long documents. Existing methods have evolved from fixed-length segmentation to more semantically aware strategies. For example, LumberChunker (Duarte et al., 2024) employs LLMs for long-form document segmentation, while DCS (Sheng et al., 2025) dynamically adjusts chunk boundaries based on semantic similarity. Although these methods improve segmentation over simple rule-based chunk-

¹Our code and data are publicly available at <https://github.com/loglixiaobu/NEAR>.

ing, they incur substantial preprocessing overhead and cannot fully prevent context fragmentation. In contrast, NEAR bypasses reliance on perfect initial segmentation, reconstructing coherent context after retrieval by reconnecting related chunks and preserving local continuity.

Chunk-Level Selection After document chunking, chunk-level selection aims to improve efficiency by identifying the chunks most relevant to a query. Structured methods further model relations among chunks. For example, GraphReader (Li et al., 2024c) performs graph-based traversal with LLM guidance, while GraphRAG (Edge et al., 2024) builds an entity-centric graph to support document-level analysis. However, these methods often introduce substantial preprocessing overhead, and their focus is typically on filtering or restructuring content rather than reconstructing a coherent reading context from retrieved chunks. In contrast, NEAR directly models neighborhood relations among text chunks, operates without LLM-based graph construction, and restores context coherence at low computational cost.

3 Approach

3.1 Overview

Figure 2 illustrates the overall architecture of NEAR. We first organize the input document as a neighborhood graph that jointly captures topical relatedness and physical adjacency between text chunks. Given a user query q , NEAR then performs iterative exploration over this graph. At each step, candidate chunks are selected by alternating between two candidate pools, and the selected chunk is re-scored by a Trust But Verify (TBV) module to guide subsequent score propagation. Finally, during context reconstruction, the retrieved chunks are selectively expanded and reordered according to their original document positions, producing a coherent reading context for the LLM.

3.2 Neighborhood Graph Construction

Given a document $\mathcal{D}$, we first partition it into N text chunks using a segmentation strategy, denoted as $\mathcal{V} = \{c_1, c_2, \dots, c_N\}$. Existing retrieval-based approaches often treat these chunks as independent units, overlooking two important structural signals in long documents: topical relations between semantically related chunks and positional dependencies induced by their physical locations.

To capture both aspects, we represent the document as a neighborhood graph.

Topical Neighbors To model topical relations, we connect chunks according to their semantic similarity, enabling transitions between segments that may be distant in the document but share related content. Specifically, we define a scoring function $\phi : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$, such as a sparse or dense retriever, that measures the affinity between two chunks. For each chunk c_i , its topical neighborhood $\mathcal{N}_{\text{topic}}^{c_i}$ is defined as the set of its K most relevant chunks:

$$\mathcal{N}_{\text{topic}}^{c_i} = \text{TopK}_{c_j \in \mathcal{V} \setminus \{c_i\}} \phi(c_i, c_j). \quad (1)$$

wherein TopK returns the K chunks with the highest similarity scores.

Physical Neighbors Document chunking may fragment local context and may break discourse dependencies such as coreference. To preserve the sequential structure of the original document, we introduce a physical neighborhood relation that captures positional adjacency between chunks. Formally, the physical neighborhood of chunk c_i is defined as $\mathcal{N}_{\text{phy}}^{c_i} = \{c_{i-1}, c_{i+1}\}$, where out-of-range neighbors are omitted for boundary chunks.

3.3 Neighborhood Graph Exploration

Given a query q , we perform iterative exploration over the neighborhood graph. To combine global retrieval signals with locally propagated relevance, the exploration maintains two candidate pools.

- **Initial Pool ($\mathcal{I}$):** The initial pool provides a global retrieval prior, storing up to N_{max} chunk-score pairs (c, s_{init}^c) , where N_{max} is the maximum number of chunks allowed by the context budget, and s_{init}^c is the initial relevance score between chunk c and query q from a lightweight first-stage retriever such as BM25. These scores remain fixed during exploration, and the chunks are ranked by s_{init}^c to guide the order of graph exploration.
- **Frontier Pool ($\mathcal{F}^{(t)}$):** The frontier pool maintains chunk-score pairs $(c, s_c^{(t)})$, where $s_c^{(t)}$ is the exploration score of chunk c at iteration t . It is initialized as $\mathcal{F}^{(0)} = \emptyset$. During exploration, once a chunk is selected and verified, its topical neighbors are inserted into the frontier pool with propagated scores. If a neighbor is already present in the frontier pool, its score is updated accordingly. In this way, $\mathcal{F}^{(t)}$

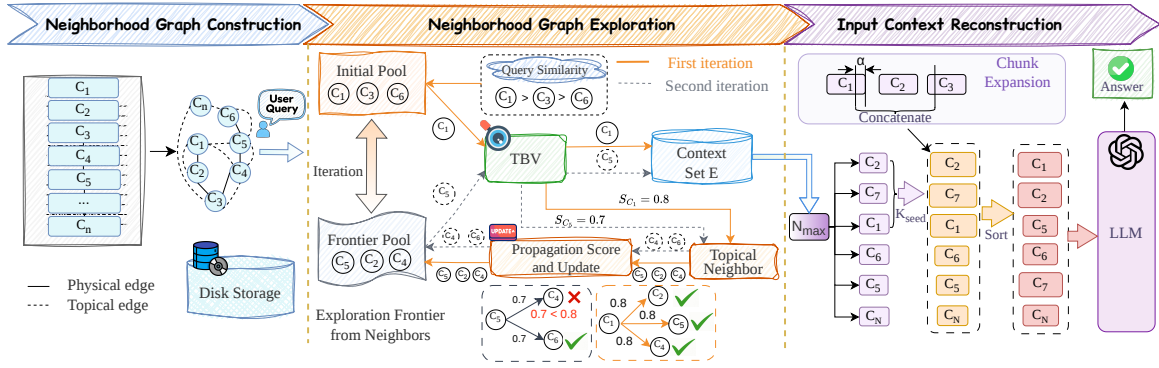


Figure 2: The overall pipeline of NEAR consists of neighborhood graph construction, neighborhood graph exploration, and input reconstruction. We present two iterations: the orange solid line denotes the first iteration, where chunks are selected from the initial pool, and the gray dashed line denotes the second iteration, where chunks are selected from the frontier pool. We also illustrate the score propagation and update process for both iterations.

stores the current exploration frontier induced by previously verified chunks.

Because the two pools rely on different scoring functions, a chunk may maintain separate scores in the initial and frontier pools.

Iterative Chunk Selection The exploration iteratively constructs a context set $\mathcal{E}$, initialized as $\emptyset$, which stores selected chunk-score pairs. At each iteration t , the active candidate pool is defined as:

$$\mathcal{X}^{(t)} = \begin{cases} \mathcal{I}, & \text{if } t \text{ is even} \\ \mathcal{F}^{(t)}, & \text{if } t \text{ is odd} \end{cases} \quad (2)$$

The chunk selected at step t is then given by:

$$c^{(t)} = \arg \max_{c \in \mathcal{X}^{(t)}} s_c \quad (3)$$

The iteration index starts from $t = 0$, with the active pool alternating between the initial and frontier pools. This alternating strategy prevents exploration from becoming trapped in narrow semantic regions (Tong et al., 2006) while balancing global query relevance with locally propagated contextual cues. As a result, NEAR achieves both broad coverage and local contextual depth during exploration.

Trust But Verify (TBV) After selecting chunk $c^{(t)}$, we apply the Trust But Verify (TBV) procedure to maintain the quality of the retrieved context during exploration. While $c^{(t)}$ is *trusted* and retained for consideration, we *verify* its relevance to the query q to prevent error propagation from misleading or only superficially related chunks. Specifically, a stronger verification function $\psi(q, c)$ is

applied to re-evaluate the pertinence of the chunk, producing a verification score $s_{\text{veri}}^{(t)}$:

$$s_{\text{veri}}^{(t)} = \psi(q, c^{(t)}) \quad (4)$$

The verified chunk is then incorporated into the context set $\mathcal{E}$:

$$\mathcal{E} \leftarrow \mathcal{E} \cup \{(c^{(t)}, s_{\text{veri}}^{(t)})\} \quad (5)$$

To maintain efficiency and avoid repeated selection, any chunk added to $\mathcal{E}$ is removed from both candidate pools, ensuring that it will not be reconsidered in subsequent iterations.

Verified Score Propagation Chunks that are relevant to the query often have topical neighbors that are also relevant. We therefore propagate the verified score of the selected chunk $c^{(t)}$ to guide subsequent exploration. For each topical neighbor $c_j \in \mathcal{N}_{\text{topic}}^{c^{(t)}}$ that has not yet been added to the context set $\mathcal{E}$, we update its exploration score as follows:

$$s_{c_j}^{(t+1)} = \begin{cases} s_{\text{veri}}^{(t)}, & \text{if } c_j \notin \mathcal{F}^{(t)} \\ \max(s_{c_j}^{(t)}, s_{\text{veri}}^{(t)}), & \text{if } c_j \in \mathcal{F}^{(t)} \end{cases} \quad (6)$$

The resulting chunk-score pair $(c_j, s_{c_j}^{(t+1)})$ is then inserted into $\mathcal{F}^{(t+1)}$ or updated if already present. This update rule prioritizes neighbors of highly relevant chunks, helping exploration focus on content that is more likely to contribute to the final reconstructed context while reducing the influence of less relevant candidates.

Termination This iterative process of selection, verification, and propagation continuously expands the context set $\mathcal{E}$. The algorithm terminates when a predefined computation budget or context-window budget is exhausted. The stopping criteria are:

$$|\mathcal{E}| \geq N_{\max} \quad \text{or} \quad \sum_{(c,s) \in \mathcal{E}} |c| \geq T_{\max} \quad (7)$$

where $T_{\max}$ is the total token budget, and $|c|$ denotes the token count of chunk c .

3.4 Input Context Reconstruction

Budget-Aware Chunk Expansion After iterative exploration, directly concatenating the retrieved chunks may still yield fragmented context and disrupt the original reading flow. To improve local coherence, we selectively expand high-priority chunks using their physical neighbors. Because expanding every retrieved chunk would quickly exceed the token budget, we first rank the chunks in $\mathcal{E}$ by their verified scores and choose the top K_{seed} chunks as the core set $\mathcal{E}_{\text{core}}$. For each core chunk $c_i \in \mathcal{E}_{\text{core}}$, we include only the boundary portions of its immediate physical neighbors:

$$\Phi(c_i) = \text{Suffix}(c_{i-1}, \alpha) \oplus c_i \oplus \text{Prefix}(c_{i+1}, \alpha) \quad (8)$$

where $\alpha \in (0, 1]$ is the expansion ratio, $\oplus$ denotes concatenation, and Suffix/Prefix return the corresponding fraction of boundary tokens. For boundary chunks, expansion is applied only on the available side. This selective expansion improves local continuity around the most important chunks while remaining within the context budget.

Position Reordering and Concatenation To form the final prompt context, we replace each core chunk in $\mathcal{E}$ with its expanded version $\Phi(c_i)$ and keep the remaining chunks in $\mathcal{E} \setminus \mathcal{E}_{\text{core}}$ unchanged. We then merge these text segments, removing any overlapping or repeated spans. To preserve the original document flow, all segments in the updated $\mathcal{E}$ are sorted by their absolute positions in the source document $\mathcal{D}$ (Laitenberger et al., 2025). The final context sequence $\mathcal{T}_{\text{context}}$ is obtained by concatenating the sorted segments:

$$\mathcal{T}_{\text{context}} = \text{Concat}_{c \in \text{sorted}(\mathcal{E})}(c) \quad (9)$$

Finally, $\mathcal{T}_{\text{context}}$ and the user query q are incorporated into a prompt template, assembling a coherent and contextually enriched input for the model to facilitate accurate long-document comprehension.

4 Experimental Setup

4.1 Benchmarks

Datasets To comprehensively evaluate our method, we conduct experiments on six datasets covering both single-hop and multi-hop question answering (QA) scenarios. Specifically, the single-hop long-context QA benchmark includes Multi-FieldQA_en (Yuan et al., 2024), Loogle-SD (Li et al., 2024a), and Factrecall (Yuan et al., 2024), while the multi-hop long-context QA benchmark includes Loogle-MIR (Li et al., 2024a), HotpotWikiQA (Yang et al., 2018; Yuan et al., 2024), and Loogle-CR (Li et al., 2024a). Furthermore, following the construction protocol of LVEval (Yuan et al., 2024), we select text subsets with lengths ranging from 16k to 256k tokens to evaluate the robustness of NEAR on ultra-long documents.

Metrics To ensure a fair and comprehensive comparison, we apply post-processing to model outputs and adopt two evaluation metrics. The first is the standard F1 score provided by LVEval (Yuan et al., 2024). The second is LLM-based evaluation score, where we use a stronger model, Qwen3-32B (Qwen Team, 2025), as the judge to score predicted answers against the ground truth. The prompt templates and detailed post-processing procedures are provided in the Appendix H and B.

4.2 Baselines

We use Llama-3-8B-Instruct (Llama Team, 2024) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) as backbone LLMs. Their maximum context lengths are 8k and 32k, respectively. We compare our method against representative baselines from three technical paradigms: (i) architecture-modifying methods for long context modeling, including LM-Infinite (Han et al., 2024) and InfLLM (Xiao et al., 2024), for which we follow the original inference procedures; (ii) chunking and candidate selection methods, including Dynamic Chunking and Selection (DCS) (Sheng et al., 2025), retrieval-based methods that use BM25 and MiniLM-L12-v2 (Wang et al., 2020) for TopK candidate retrieval; and (iii) graph-structured approaches such as GraphReader (Li et al., 2024c), which constructs document-level graphs to facilitate long-form reading comprehension.

4.3 Implementation Details

For $\phi(c_i, c_j)$, we use BM25 for scoring, while for $\psi(q, c)$, we adopt the sentence-transformer model

Method	Factrecall		MultifieldQA		Loogle_SD		Loogle_CR		Loogle_MIR		HotpotWikiQA		Total AVG	
	F1	LLM	F1	LLM	F1	LLM	F1	LLM	F1	LLM	F1	LLM	F1	LLM
Llama-3-8B-Instruct	43.56	39.50	18.30	27.12	27.49	38.12	16.44	22.73	12.68	24.32	14.22	18.68	22.11	28.41
w/ InfLLM	49.49	48.75	18.34	27.43	21.48	30.44	15.77	23.52	11.76	22.72	9.60	20.47	21.07	28.88
w/ LM-Infinite	35.67	33.60	14.24	20.32	21.41	26.50	13.99	19.59	11.87	18.69	13.45	17.42	18.44	22.68
w/ BM25	54.50	50.55	28.14	41.97	54.21	67.31	23.28	32.38	23.13	38.24	23.82	32.74	34.51	43.86
w/ MiniLM-L12-v2	72.70	72.50	27.96	40.75	51.59	65.37	23.19	32.23	23.09	39.71	23.77	29.95	37.05	46.75
w/ DCS	88.40	84.95	24.79	41.88	34.92	46.79	19.53	27.68	18.77	30.93	24.09	29.47	35.08	43.61
w/ GraphReader	85.00	86.00	35.26	42.47	50.57	63.44	20.71	29.85	13.96	26.91	20.13	27.17	37.61	45.97
w/ NEAR (Ours)	93.76	90.50	31.42	44.65	58.19	71.69	25.56	34.13	25.07	41.27	25.76	30.91	43.29	52.19

Table 1: Performance comparison of various methods across different datasets with a context length of 16k tokens. We employ Llama-3 as the base reader and utilize recursive chunking for text segmentation. Both F1 score and LLM-based evaluation metrics are reported. The best results are highlighted in blue. Results based on Mistral, as well as those utilizing fixed-length chunking, are detailed in the Appendix C.

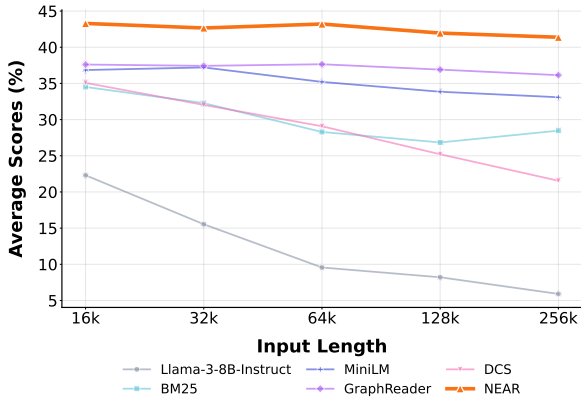


Figure 3: Average F1 scores on six datasets across input lengths from 16k to 256k tokens using Llama-3-8B-Instruct. The results illustrate the exceptional stability of NEAR on ultra-long documents.

MiniLM-L12-v2. We consider two text chunking strategies: fixed-token chunking and recursive chunking. In recursive chunking, every three consecutive sentences are first grouped into a preliminary unit, which is then packed to approach the token-length limit. For our method with recursive chunking, we set the number of topical neighbors per node K to 8, the expansion ratio α to 30%, and the number of top seed chunks K_{seed} to 18. All chunk-based methods are evaluated under the same configuration: we cap the input context at 7.5k tokens, use a maximum chunk size of 128 tokens, and set N_{max} to 56. Additional implementation details are provided in the Appendix E.

5 Results and Analysis

5.1 Main Results

Table 1 compares NEAR with several competitive baseline models across six long-document QA datasets. Overall, NEAR achieves the best aver-

age performance, recording the highest F1 score (43.29) and LLM-based evaluation score (52.19). Notably, NEAR substantially improves the comprehension accuracy of the base Llama-3-8B-Instruct model. Furthermore, when compared to advanced methods, NEAR’s advantages are equally prominent. For instance, although dynamic chunking and selection methods like DCS effectively filter out irrelevant noise, they ultimately provide the LLM with disjointed text snippets. By reconstructing these fragmented chunks into a more coherent and query-relevant reading context, NEAR outperforms DCS by a relative margin of 23.40% in average F1. Additionally, against GraphReader, a strong baseline that relies on LLM-based atomic information extraction, NEAR achieves a 15.10% relative improvement. We attribute this to the fact that atomic extraction often disrupts the original narrative flow. In contrast, NEAR operates directly on original text segments via its dual-layer neighborhood graph, better preserving document structure and continuity. In conclusion, these results demonstrate that NEAR effectively reconstructs a coherent and query-relevant context from fragmented retrieved chunks. By providing the model with a more continuous and information-rich input sequence, NEAR significantly enhances final answer accuracy.

5.2 Robustness Across Extended Contexts

To evaluate performance on ultra-long documents, we compare NEAR with competitive baselines across input lengths ranging from 16k to 256k tokens. As shown in Figure 3, NEAR consistently maintains the highest average scores across all evaluated scales. As document length increases, baselines suffer severe performance degradation. This trend suggests that conventional long-document

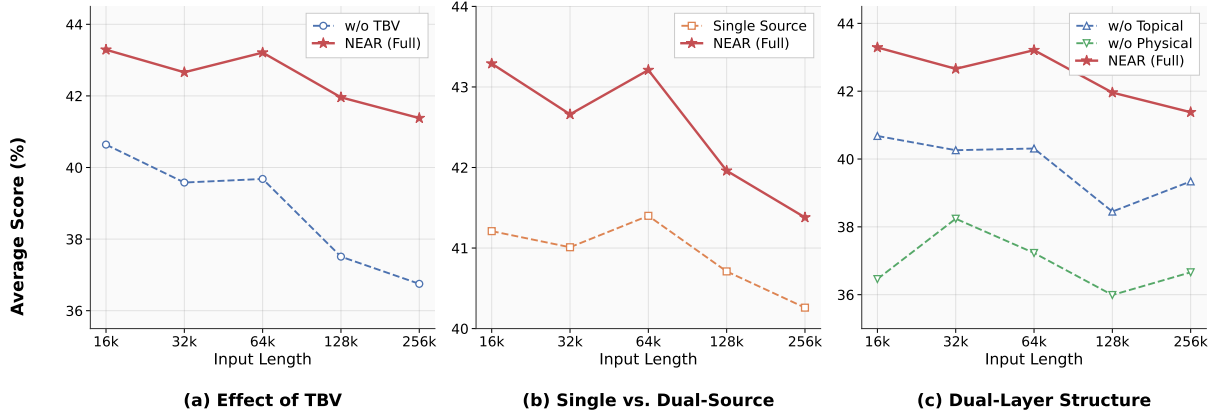


Figure 4: Ablation studies on NEAR. (a) Effect of TBV. (b) Comparison between single source and dual-source configurations. (c) Contribution of physical and topical neighbor layers. Performance is measured as average F1 score across six datasets with varying input lengths (16k–256k).

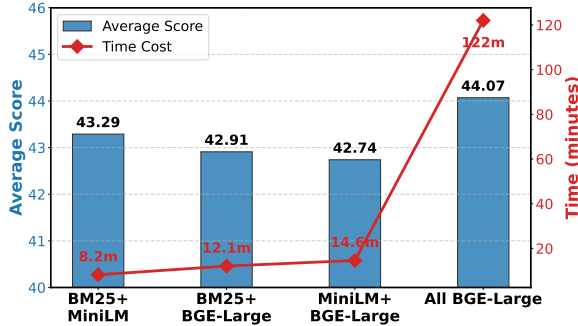


Figure 5: Trade-off between accuracy and computational overhead for different scoring methods at a 16k token input length. The x-axis represents the combined strategies, where the first line corresponds to graph construction and the second line to TBV scoring. Evaluated across six datasets via average F1 scores and time costs (minutes).

pipelines become increasingly brittle when relevant information is distributed more sparsely and surrounded by more distracting content. In stark contrast, NEAR demonstrates exceptional stability, varying by less than 2 absolute percentage points across the entire spectrum. At the extreme 256k context length, NEAR establishes a substantial lead, outperforming DCS and BM25 by relative margins of 92.0% and 45.3%, respectively. This robustness likely stems from NEAR’s ability to reconstruct coherent context by combining long-range topical relatedness with local continuity, rather than relying on isolated retrieved chunks alone.

5.3 Ablation

We conduct three sets of ablation experiments to validate the design choices of NEAR, with detailed results reported in Figure 4.

TBV Mechanism Removing the Trust But Verify module leads to consistent performance drops across all document lengths, and the gap widens as input length increases. Specifically, the drop grows from about 3 percentage points at 16k tokens to about 6 percentage points at 256k tokens. This result highlights the importance of explicit relevance verification in ultra-long contexts, where iterative expansion may otherwise propagate noisy or weakly relevant chunks.

Dual-Source Exploration Replacing the alternating dual-source strategy with single-source exploration (i.e., relying solely on the Frontier Pool $\mathcal{F}$) leads to a consistent and noticeable drop in performance; for example, the performance decreases by 2.71% at 256k tokens. This result further validates that balancing global query similarity with local frontier expansion is important for preventing premature convergence to semantically narrow or overly concentrated regions.

Dual-Layer Graph We further study the contribution of the dual-layer neighborhood graph. Using only the physical graph or only the topical graph degrades performance consistently across all input lengths. This suggests that topical relations alone may introduce noise and coreference ambiguity, while physical relations alone are insufficient to capture important long-distance dependencies. Combining both graph layers yields the best performance across all evaluation scales.

5.4 Discussion

Scoring Mechanism Selection We evaluate several scoring methods for neighborhood graph con-

Method	Factrecall	HotpotWikiQA	Loogle_CR	Loogle_MIR	Loogle_SD	MultifieldQA	Average
TT	90.50	19.35	19.19	22.30	49.38	23.76	39.45
TF	3.00	5.65	4.04	11.51	6.88	4.95	5.81
FT	4.00	7.26	8.08	12.23	11.25	7.92	8.18
FF	2.50	67.74	68.69	53.96	32.50	63.37	46.56

Table 2: Quantitative comparison of graph construction correctness between BM25 and BGE-reranker-large. TT: both methods correct; TF: only BM25 correct; FT: only BGE-reranker correct; FF: both methods incorrect. All values are percentages (%).

struction and TBV verification: sparse retrieval (BM25), dense retrieval (MiniLM-L12-v2), and cross-encoder reranking namely BGE-reranker-large (Xiao et al., 2023). As shown in Figure 5, the first two provide the optimal balance of accuracy and efficiency. Therefore, we adopt BM25 for initial graph construction to enable fast lexical matching in large spaces, and use MiniLM for TBV verification to achieve strong semantic matching at moderate cost. Although the cross-encoder yields higher accuracy, its substantial computational overhead limits its practicality.

To further quantify the limitations of BM25’s lexical matching in graph construction, we conduct comparative experiments on six benchmark datasets, building graphs with BM25 and BGE-reranker-large, respectively. A relation is considered correctly constructed if its semantic similarity to the ground-truth relation exceeds 70%. As shown in Table 2, the overall metric difference $FT - TF = 2.37\%$ indicates that BM25-based graph construction only loses a small number of relations that are semantically recoverable. Combined with the substantial computational overhead of BGE-reranker-large for graph construction, this result validates the rationale of adopting BM25 as the initial graph builder. Benefiting from NEAR’s dual-source exploration mechanism and TBV mechanism, the risk of error propagation caused by locally incorrect relations can be effectively mitigated. Therefore, this limited structural loss constitutes an acceptable performance-efficiency trade-off.

Hyperparameters We conduct systematic hyperparameter optimization experiments to determine the optimal configuration, with detailed results shown in Table 3. Taking Llama-3 as an example, the optimal chunk size is 128 tokens under the recursive chunking strategy. If the granularity is too small, chunks may lack sufficient information to construct meaningful topical edges in the

Hyperparameter	Settings & F1 Score				
(a) <i>Neighbors (K)</i>	1	2	4	8	16
	40.73	41.05	42.53	43.29	42.21
(b) <i>Chunk Size</i>	64	128	256	512	1024
	43.04	43.29	41.12	39.44	39.33
(c) <i>Ratio α (%)</i>	10	30	50	75	100
	42.98	43.29	42.06	42.13	41.17
(d) <i>K_{seed}</i>	1	6	12	18	24
	42.42	42.71	42.93	43.29	43.02

Table 3: Hyperparameter sensitivity analysis. All experiments are conducted under the 16k context window constraint, with average F1 scores across six datasets reported. The optimal configurations are highlighted.

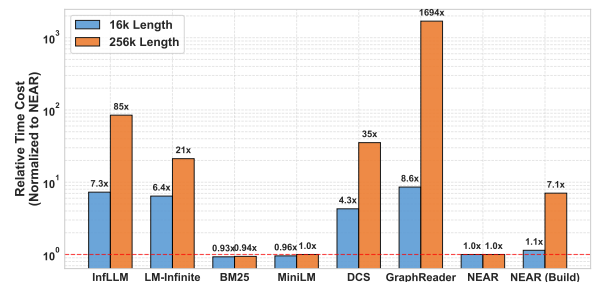


Figure 6: Relative time cost comparison of various methods normalized to NEAR. We report the average processing time evaluated on six datasets with input lengths of 16k and 256k tokens, where "NEAR (Build)" includes the neighborhood graph construction overhead.

graph. If they are too large, chunk representations become less specific, which can reduce retrieval precision and introduce irrelevant content into the reconstructed context. For the adaptive expansion ratio α , the optimal value is 50% for fixed-size chunking and 30% for recursive chunking. This is consistent with the intuition that recursive chunking preserves natural sentence boundaries more effectively, thus requiring less external context to maintain coherence. Notably, NEAR exhibits only small performance fluctuations, indicating strong robustness to hyperparameter choices.

Method	Chunk Size (tokens)				
	64	128	256	512	1024
BM25	31.99	34.51	35.53	35.07	33.93
MiniLM-L12-v2	34.73	37.05	36.09	33.43	29.18
NEAR	43.04	43.29	41.12	39.44	39.33
$\Delta(\text{NEAR} - \text{Best})$	8.31	6.24	5.03	4.37	5.40

Table 4: Result over different chunk sizes. All experiments are performed within a 16k-token context window, and we report the average F1 score over six datasets. Δ denotes the performance gain of NEAR over the best baseline under each chunk size.

Computational Efficiency We analyze the computational overhead of NEAR against representative baselines, with results shown in Figure 6. A key practical advantage of NEAR is the separation between offline graph construction and online inference, which allows preprocessing cost to be amortized across multiple queries. During online inference, NEAR performs only lightweight graph exploration and context reconstruction. As a result, its latency ($1.0\times$) remains close to that of direct retrieval baselines such as BM25. NEAR also avoids the additional cost associated with dynamic chunking methods. Approaches such as DCS incur substantial overhead, reaching $35\times$ at 256k tokens, because they rely on semantically adaptive segmentation before retrieval. In contrast, NEAR works effectively with simple chunk partitions and restores missing coherence through a lightweight post-retrieval reconstruction step. The efficiency of NEAR is also evident in the offline stage. Methods such as GraphReader become increasingly expensive as document length grows because graph construction depends on repeated LLM calls. Its overhead rises from $8.6\times$ at 16k tokens to $1694\times$ at 256k tokens. By comparison, NEAR constructs its neighborhood graph using lightweight scoring functions, and its full build cost remains limited to $7.1\times$ even at 256k tokens. Overall, NEAR retains the structural benefits of graph-based context reconstruction while keeping preprocessing cost and inference latency at a practical level.

Impact of Information Fragmentation Information fragmentation is a central challenge in long-context retrieval. In this section, we treat chunk size as a controllable proxy for fragmentation severity and investigate how varying degrees of fragmentation affect retrieval performance. As shown in Table 4, both excessive fragmentation

and overly coarse granularity degrade performance. Very small chunks lose necessary surrounding context, while very large chunks, although preserving more context, introduce substantial noise that renders retrieval unreliable. More importantly, as fragmentation intensifies, the performance gap between NEAR and strong baselines consistently widens, growing from 5.40 at 1024 tokens to 8.31 at 64 tokens. This trend directly validates our core claim: when information is highly fragmented and individual chunks lack sufficient local context, NEAR can reconstruct coherent context and effectively mitigate the adverse effects of fragmentation.

6 Conclusion

We present NEAR, a lightweight, training-free framework designed to reconstruct coherent context from fragmented chunks. Rather than treating retrieved chunks in isolation or relying on expensive LLM-based graph construction, NEAR models inherent structural relations in text and reorganizes the original content into a logically coherent, query-relevant sequence, without requiring costly chunk optimization. Extensive experiments on long-document question answering benchmarks show that NEAR consistently improves downstream accuracy and remains highly robust as document length increases substantially. NEAR incurs minimal online inference overhead while supporting offline precomputation, making it well suited for scalable, low-latency real-world deployment. We hope this work will inspire future research on context reconstruction and more efficient methods for long-document reading comprehension.

Limitations

While NEAR effectively balances efficiency and performance, its design leaves room for improvement in two aspects. First, to maintain a lightweight graph construction process, NEAR relies primarily on BM25 for the initial graph construction. Although subsequent vector-based verification helps refine the graph, these relatively coarse matching signals may still fail to capture implicit coreference relationships across chunks. Future work could incorporate richer structural representations, such as AMR-based representations (Banarescu et al., 2013), to better identify such latent dependencies. Second, in favor of efficiency, several components of NEAR employ relatively simple heuristics, without being jointly optimized un-

der a unified objective or explicitly accounting for factors such as distance-aware decay and varying edge confidence. More adaptive routing and propagation mechanisms, guided by expected information gain or uncertainty reduction, could therefore further improve the robustness, interpretability, and generalizability of the framework.

Acknowledgments

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This work was supported by the Natural Science Foundation of China (No. 62506354, 62572456, 62272439), the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20251041.

References

- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. [Longbench: A bilingual, multi-task benchmark for long context understanding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 3119–3137. Association for Computational Linguistics.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. [Abstract Meaning Representation for sembanking](#). In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Yiran Ding, Li Lina Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. [Longrope: Extending LLM context window beyond 2 million tokens](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 11091–11104. PMLR / OpenReview.net.
- André V. Duarte, João DS Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L. Oliveira. 2024. [LumberChunker: Long-form narrative document segmentation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486, Miami, Florida, USA. Association for Computational Linguistics.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. [From local to global: A graph RAG approach to query-focused summarization](#). *CoRR*, abs/2404.16130.
- Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. [Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model](#). In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1074–1084. Association for Computational Linguistics.
- Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, K. Aditya, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. [Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. [Lm-infinite: Zero-shot extreme length generalization for large language models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, NAACL 2024, Mexico City, Mexico, June 16-21, 2024, pages 3991–4008. Association for Computational Linguistics.
- Marti A. Hearst and Jan O. Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '96*, New York, NY, USA. ACM.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *CoRR*, abs/2310.06825.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. [Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 1658–1677. Association for Computational Linguistics.

- M. G. KENDALL. 1938. [A new measure of rank correlation](#). *Biometrika*, 30(1-2):81–93.
- Alex Laitenberger, Christopher D. Manning, and Nelson F. Liu. 2025. [Stronger baselines for retrieval-augmented generation with long-context language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 32559–32569. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024a. [Loogle: Can long-context language models understand long contexts?](#) In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 16304–16333. Association for Computational Linguistics.
- Minghan Li, Éric Gaussier, Juntao Li, and Guodong Zhou. 2024b. [Keyb2: Selecting key blocks is also important for long document ranking with large language models](#). *CoRR*, abs/2411.06254.
- Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. 2024c. [Graphreader: Building graph-based agent to enhance long-context abilities of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pages 12758–12786. Association for Computational Linguistics.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Trans. Assoc. Comput. Linguistics*, 12:157–173.
- Llama Team. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. [Yarn: Efficient context window extension of large language models](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Qwen Team. 2025. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. [RAPTOR: recursive abstractive processing for tree-organized retrieval](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Boheng Sheng, Jiacheng Yao, Meicong Zhang, and Guoxiu He. 2025. [Dynamic chunking and selection for reading comprehension of ultra-long context in large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31857–31876, Vienna, Austria. Association for Computational Linguistics.
- C. Spearman. 1904. [The proof and measurement of association between two things](#). *The American Journal of Psychology*, 15(1):72–101.
- Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. [Galactica: A large language model for science](#). *CoRR*, abs/2211.09085.
- Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. [Fast random walk with restart and its applications](#). In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 613–622. IEEE Computer Society.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. [Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhajan Kambadur, David S. Rosenberg, and Gideon Mann. 2023. [Bloomberggpt: A large language model for finance](#). *CoRR*, abs/2303.17564.
- xAI. 2026. Grok 4.3 api documentation. <https://x.ai/api>. Accessed: 2026-05-10.
- Chaojun Xiao, Pengl Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. [Inflm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory](#). *CoRR*, abs/2402.04617.
- Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. [C-pack: Packaged resources to advance general chinese embedding](#). *CoRR*, abs/2309.07597.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian

Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. [Qwen2 technical report](#). *CoRR*, abs/2407.10671.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025. [Qwen2.5-1m technical report](#). *CoRR*, abs/2501.15383.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [Hotpotqa: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. 2024. [Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k](#). *CoRR*, abs/2402.05136.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2025. [Judgelm: Fine-tuned large language models are scalable judges](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

A LVEval Benchmark

We evaluate our method using LVEval (Yuan et al., 2024), a comprehensive benchmark designed to assess the long-context understanding of LLMs. LVEval challenges models across five length tiers (16k, 32k, 64k, 128k, and 256k tokens) on both single-hop and multi-hop question answering (QA) tasks. To ensure rigorous and objective evaluation, the benchmark incorporates three key mechanisms: (1) deliberately inserting confusing facts to test distractor identification; (2) applying keyword substitution to prevent models from relying on superficial pattern matching; and (3) utilizing keyword-recall-based metrics to mitigate the biases of traditional evaluation scoring. In this study, we conduct experiments on the six English datasets from LVEval. Detailed length statistics for each dataset are provided in Table 5.

B Post-processing

To ensure precise evaluation and mitigate the impact of verbose or redundant model outputs on metric calculation, we implemented a rigorous post processing pipeline for the model predictions. The

Dataset	16k	32k	64k	128k	256k
Factrecall	13,920	27,670	54,744	108,311	213,750
HotpotWikiQA	20,286	36,414	68,005	132,253	260,506
Loogle_CR	25,093	40,928	77,021	141,723	270,157
Loogle_MIR	26,245	40,555	78,006	142,490	271,622
Loogle_SD	19,893	40,183	73,787	138,642	265,863
MultiFieldQA	18,854	35,212	67,333	130,623	258,984
Average	20,715	36,827	69,816	132,340	256,814

Table 5: Detailed average token lengths for the six selected LVEval datasets across different context length settings.

post-process strategy consists of four key components: Prefix and Artifact Removal: We strip specific instructional prefixes (e.g., "Answer:", "Assistant:") and model specific termination tokens to isolate the core response. Stop Phrase Truncation: To prevent hallucinated continuations where the model generates subsequent questions or repeats input context, we define a set of stop markers (including "Question:", "Article:", and "Note:"). The output is truncated immediately upon the detection of any such marker. Single Line Enforcement: Given the concise nature of the QA tasks in LVEval, we retain only the first line of the generated text, discarding any subsequent elaboration or reasoning that often introduces noise. Parenthetical Filtering: We verify parenthetical content to remove "self correction" or meta commentary (e.g., phrases indicating "incorrect" or "typo"), ensuring that only the primary candidate answer remains for evaluation. We provide examples in Figure 7.

C More Results

C.1 Results on Fixed Chunking

To further verify the robustness of the NEAR method, we evaluated its performance under a fixed chunking strategy. Table 6 presents a performance comparison between traditional retrieval augmented methods and NEAR across six QA datasets under the fixed 128 token chunking setting. As can be seen from the results, under the fixed chunking constraint, NEAR achieves significant accuracy improvements on all evaluated datasets: on the Factrecall_en dataset, the accuracy of NEAR is improved by approximately 23.45% compared to the best performing baseline model, and the improvement over the baseline model Llama-3-8B-Instruct exceeds 100%; on the comprehensive metric Total AVG, the performance of NEAR is also improved by approximately 12.39% compared to the best performing baseline method. This indi-

Category	Raw Model Output	Post processed Output
Self-Correction / Note	Ludwig Beethoven (Note: he is not a real physicist, but a composer...)	Ludwig Beethoven
Stop Marker (Truncation)	Ludwig Beethoven\nQuestion: What is the name of the scientist...?	Ludwig Beethoven
Prefix Removal	Answer: Ludwig Beethoven	Ludwig Beethoven
Parenthetical Self-Correction	Ludwig Beethoven (This is incorrect, the correct answer is Albert Einstein)	Ludwig Beethoven
Instructional Text Removal	Ludwig Beethoven\nPlease answer the following question...	Ludwig Beethoven

Figure 7: Example of the post-processing pipeline on the Factrecall_en dataset.

Model	Factrecall	MultiFieldQA	Loogle_SD	Loogle_CR	Loogle_MIR	HotpotWikiQA	AVG
Llama-3-8B-Instruct	43.56	18.30	27.49	16.44	12.68	14.22	22.11
BM25	59.00	27.67	50.20	21.48	20.27	26.34	34.16
MiniLM-L12-v2	72.50	28.02	48.65	22.45	21.84	24.37	36.31
NEAR (Ours)	89.50	28.64	54.22	23.06	22.50	26.91	40.81

Table 6: Performance comparison on different datasets. We use a fixed chunk size of 128 tokens for text segmentation. The best results in each column are highlighted in **bold** with a blue background.

cates that NEAR can not only leverage its strengths in recursive chunking scenarios but also maintain excellent performance under fixed chunking strategies, fully demonstrating its good adaptability and robustness to different chunking methods.

C.2 Results on Mistral

To further validate the generalization capability of NEAR across diverse foundation models, we carry out supplementary experiments on Mistral-7B-Instruct-v0.3 over context lengths ranging from 16k to 256k tokens, as visualized in Figure 8. While vanilla Mistral and most baselines degrade substantially with extended context, NEAR sustains high average F1 above 40 with negligible performance drop even at 256k tokens. Impressively, as shown in Figure 9, our approach using only 7,500 tokens outperforms vanilla Mistral with a 30,000-token budget, showing great efficiency for long-context tasks. The dataset-level results at 16k context are summarized in Table 7. NEAR achieves the best overall average and obtains top performance on five of six datasets, with strong results across most benchmarks. Jointly, these results demonstrate consistent performance gains on Mistral-7B-Instruct-v0.3 and confirm our method’s strong generalization across different base models.

C.3 Results on More Document-Centric Tasks

To demonstrate the generalization of NEAR, we conduct supplementary experiments on additional datasets covering different document-centric tasks.

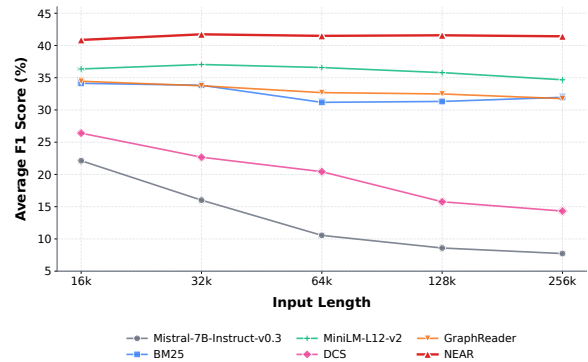


Figure 8: Average F1 scores over six datasets with input lengths ranging from 16k to 256k tokens on Mistral-7B-Instruct-v0.3. NEAR maintains stable high performance, while vanilla Mistral and most baseline methods suffer obvious performance degradation as the input context expands.

We test multi-document QA and multi-hop reasoning on HotpotQA (Yang et al., 2018), evidence extraction and information retrieval on Passage Retrieval En (Bai et al., 2024), and multi-document summarization on Multi-News (Fabbri et al., 2019). Table 8 presents the results based on Llama-3-8-Instruct. NEAR obtains the best average performance over the three tasks and outperforms other compared methods across these diverse task settings. These results show that NEAR can work well beyond question answering and has good generalization for various document-centric applications.

Method	Factrecall	HotpotWikiQA	Loogle_CR	Loogle_MIR	Loogle_SD	MultifieldQA	AVG
Mistral-7B-Instruct-v0.3	37.42	10.78	18.53	16.93	28.47	20.60	22.13
With InfLLM	50.60	9.31	15.77	12.02	19.48	18.36	20.92
With LM-Infinite	57.96	8.72	16.89	11.97	15.84	16.97	21.39
With BM25	55.50	20.24	23.65	21.65	54.79	28.93	34.13
With MiniLM-L12-v2	71.50	19.89	22.78	20.35	54.68	28.96	36.36
With DCS	34.35	20.52	21.85	18.17	35.67	27.82	26.40
With GraphReader	74.93	22.05	20.53	18.76	38.87	31.58	34.45
With NEAR (Ours)	86.54	22.20	25.26	22.83	57.69	30.62	40.86

Table 7: Performance comparison of various methods across different datasets with a context length of 16k tokens. We employ Mistral-7B-Instruct-v0.3 as the base reader and utilize recursive chunking for text segmentation. Only F1 scores are reported. The best results are highlighted in blue .

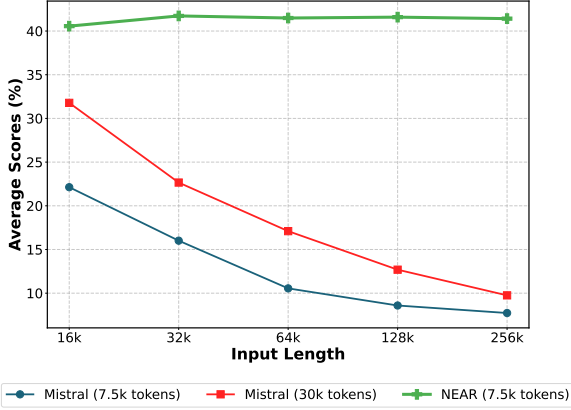


Figure 9: Experimental results on six datasets (Loogle-SD, MultiFieldQA, Factrecall, Loogle-MIR, HotpotWikiQA, Loogle-CR) based on the Mistral-7B model. The x-axis denotes the range of input context lengths (from 16k to 256k tokens), while the y-axis reports the average F1 scores of the evaluated methods.

D Algorithm Description

Algorithm 1 presents the detailed procedure of NEAR, which consists of three phases: neighborhood graph construction, neighborhood graph exploration, and input context reconstruction.

D.1 Neighborhood Graph Construction

NEAR first chunks the document D into $\mathcal{V} = \{c_1, \dots, c_N\}$ and builds a sparse neighborhood graph. For each chunk c_i , it retrieves a topical neighbor set $\mathcal{N}_{\text{topic}}^{c_i}$ as the Top- K most similar chunks under $\phi(c_i, c_j)$, and defines a physical neighbor set $\mathcal{N}_{\text{phy}}^{c_i} = \{c_{i-1}, c_{i+1}\}$ to capture local order. This graph is constructed offline and can be cached and reused across queries. The offline cost is $O(N^2)$ with exhaustive similarity, or about $O(N \log N)$ when Top- K retrieval is supported by an index; physical neighbors take $O(N)$.

D.2 Neighborhood Graph Exploration

Given query q , NEAR maintains an initial pool $\mathcal{I}$ (global prior from BM25) and a frontier pool $\mathcal{F}^{(t)}$ (candidates reached from verified chunks). At iteration t , it alternates between $\mathcal{I}$ and $\mathcal{F}^{(t)}$ to select the next chunk, which helps balance global coverage and local refinement. After selecting $c^{(t)}$, NEAR applies Trust But Verify by re-scoring with $\psi(q, c^{(t)})$ and propagates the verified score to topical neighbors to update the frontier. Let L be the number of iterations. By the stopping criteria, $L \leq N_{\max}$ (and is also bounded by $T_{\max}$). With priority queues, each iteration performs one pop in $O(\log N)$, one verification with cost T_{ψ} , and at most K frontier updates in $O(K \log N)$. Thus the online exploration cost is $O(L \cdot (K \cdot \log N + T_{\psi}))$.

D.3 Input Context Reconstruction

To form a coherent prompt, NEAR expands only the top K_{seed} chunks (by verified scores) using boundary tokens from their physical neighbors with ratio α , then merges overlaps and reorders all segments by their positions in D before concatenation. This step costs $O(N_{\max} \log N_{\max})$ for sorting plus linear time in the final context length.

E More Analysis and Details

To situate our approach within the current landscape of long-context modeling, we categorize existing baselines into three primary streams and analyze their respective behaviors:

(i) **Architecture-modifying methods.** Methods like LM-Infinite (Han et al., 2024) and InfLLM (Xiao et al., 2024) optimize attention mechanisms (e.g., local attention masks or external memory lookups) to extend the effective context window. While these designs maintain fluency and reduce perplexity, they often prioritize length extrapolation

Method	HotpotQA	Passage Retrieval En	Multi-News	AVG
Llama-3-8B-Instruct	46.74	66.5	27.65	46.96
w/ InfLLM	44.15	74.00	27.59	48.58
w/ LM-Infinite	47.52	47.50	27.66	40.89
w/ MiniLM-L12-v2	51.24	97.50	27.58	58.77
w/ BM25	50.99	97.50	27.46	58.65
w/ DCS	48.85	90.00	27.52	55.46
w/ GraphReader	46.67	62.02	27.11	45.26
w/ NEAR	52.26	99.00	28.13	59.80

Table 8: Experimental results across different document-centric tasks based on Llama-3-8B-Instruct.

over retrieval precision. By compressing or ignoring the middle context, these models frequently miss critical information required for precise QA tasks.

(ii) Chunking and selection strategies. Approaches such as DCS (Sheng et al., 2025) and traditional retrieval pipelines reduce input noise by selecting only the most query-relevant segments. However, a critical drawback of purely selection-based methods is that they treat text chunks as independent entities, ignoring the intrinsic logical connections and narrative flow of the original document. Consequently, the provided context is logically fragmented. This severely limits the model’s ability to answer complex questions that depend on integrating information from spatially distant but implicitly related chunks.

(iii) Graph-structured approaches. To address structural fragmentation, methods like GraphReader (Li et al., 2024c) explicitly model segment relationships by constructing document-level graphs. While this structural awareness improves comprehension stability, relying on repeated LLM calls for graph construction and traversal introduces prohibitive computational overhead and latency. NEAR mitigates this cost by adopting lightweight scoring functions (e.g., BM25) to efficiently construct the graph topology, striking a practical balance between deep structural awareness and inference speed.

(iv) Fair comparison and implementation details. To ensure a rigorous and fair evaluation, we evaluate all methods under identical hardware environments (NVIDIA RTX 3090 GPUs) and introduce strict control variables to isolate the impact of context modeling. While we maintain the core mechanisms of each baseline, necessary unifications are applied. For architecture-modifying methods, we strictly follow their native inference

	F1	Qwen-32B	JudgeLM	Llama3
Spearman’s ρ				
F1	1.0000	0.9286	0.8333	0.9286
Qwen-32B	0.9286	1.0000	0.9048	1.0000
JudgeLM	0.8333	0.9048	1.0000	0.9048
Llama3	0.9286	1.0000	0.9048	1.0000
Kendall’s τ				
F1	1.0000	0.7857	0.7143	0.7857
Qwen-32B	0.7857	1.0000	0.7857	1.0000
JudgeLM	0.7143	0.7857	1.0000	0.7857
Llama3	0.7857	1.0000	0.7857	1.0000

Table 9: Correlation coefficient matrix across different judge models.

procedures. For GraphReader (Li et al., 2024c), we replace its backbone with Llama-3-8B-Instruct across all graph operations to rule out performance differences caused by underlying LLM capabilities. For DCS (Sheng et al., 2025), we unify its chunk size to 128 tokens to match our segmentation granularity, while retaining its original query-aware selection pipeline. This controlled protocol ensures that the observed performance gains are directly attributable to the context reconstruction methodologies themselves.

F Time and Memory Overhead

To assess practical deployment feasibility, we analyze the computational overhead of NEAR. During inference, the peak GPU memory usage is approximately 17.58 GB (18,000 MB). This footprint is almost entirely dominated by the Llama-3-8B-Instruct backbone. The proposed dual-layer graph construction and iterative traversal mechanisms introduce negligible additional memory, enabling the entire framework to operate comfortably on a single 24GB consumer-grade GPU. Regarding inference latency, NEAR demonstrates exceptional efficiency. At the 16k-token scale, direct LLM generation takes 2.68 seconds per sample, and a standard BM25 pipeline takes 2.75 seconds. NEAR requires

	Origin	Lenient	Strict
Spearman’s ρ			
Origin	1.0000	0.9762	0.9524
Lenient	0.9762	1.0000	0.9762
Strict	0.9524	0.9762	1.0000
Kendall’s τ			
Origin	1.0000	0.9286	0.8571
Lenient	0.9286	1.0000	0.9286
Strict	0.8571	0.9286	1.0000

Table 10: Correlation coefficient matrix across different prompt styles.

3.19 seconds, representing a marginal increase of only 0.44 seconds over static retrieval. Crucially, this efficiency is maintained even on ultra-long documents: at the extreme 256k-token scale, BM25 takes 7.03 seconds while NEAR takes 7.49 seconds, adding merely 0.46 seconds of overhead. Because NEAR allows graph structures to be computed offline and cached locally, online inference avoids prohibitive computational bottlenecks, confirming that the framework possesses strong scalability and high real-world applicability.

G Consistency of LLM Evaluation

LLM-as-judge automatic evaluation is widely adopted in LLM benchmarking. Nevertheless, its outputs are susceptible to evaluator model characteristics, prompt styles and subjective grading preferences, which may introduce result fluctuations. Its alignment with conventional objective metrics such as F1-score also requires further validation. To demonstrate the robustness of our evaluation system, we conduct consistency-check experiments along two dimensions: judge-model variation and grading-prompt variation.

Judge model variation To mitigate bias from a single judge model and verify cross-model ranking stability, we employ the popular open source evaluator Llama-3-8B-Instruct and the evaluation-specialized JudgeLM-7B (Zhu et al., 2025). We fix test data, evaluation pipeline and grading criteria, and only substitute the judge model. This setup guarantees that ranking shifts stem purely from evaluator differences, enabling valid assessment of the generalizability and stability of our evaluation results.

Prompt style variation Wording and strictness of grading prompts directly affect scoring by LLMs. To quantify the prompt sensitivity of our evaluation

system, we design two additional prompt variants (lenient and strict) based on the standard prompt. The lenient prompt prioritizes overall semantics and task completion while tolerating minor imperfections in outputs. The strict prompt imposes rigorous requirements for factual accuracy, logical soundness and format compliance. Full contents of all prompts are illustrated in Figure 10.

Results We adopt Spearman’s ρ (Spearman, 1904) and Kendall’s τ (KENDALL, 1938) to measure ranking consistency, and follow common evaluation practice to set stability thresholds: $\rho \geq 0.8$ and $\tau \geq 0.7$. Correlation matrices for cross-judge-model and cross-prompt-style experiments are presented in Table 9 and Table 10, respectively. Experimental results exhibit high evaluation consistency. For varying judge models, the average correlation values are $\rho = 0.9365$ and $\tau = 0.8571$. For varying prompt styles, the average correlations reach $\rho = 0.9863$ and $\tau = 0.9048$. All pairwise correlation coefficients exceed the predefined stability thresholds. These observations strongly suggest that our LLM-as-judge evaluation is barely disturbed by judge-model and prompt-style changes, confirming the robustness and reliability of our evaluation conclusions.

H Prompt

This section details the prompt templates adopted for constructing our evaluation datasets, covering the standardized input prompts for different task scenarios. The specific template contents for each setting are visualized in Figures 11, 12, and 13, respectively. Furthermore, the dedicated grading prompt template utilized for LLM-as-judge scoring is provided in Figure 11.

I Notation

For better readability and clarity, we summarize the frequently used mathematical notations and key symbols throughout this paper in Table 11.

Original Grader Prompt:

(System Prompt) “You are a strict grader. Score the factual correctness of the prediction by comparing it with the reference answers. Only focus on factual alignment and ignore formatting and stylistic differences. Return only a single floating-point number between 0 and 1 (inclusive) as the first token.”

Lenient Grader Prompt:

(System Prompt) “You are a lenient grader. Evaluate the semantic match between the prediction and the reference answer. If the core meaning is substantially correct, assign a high score. Minor factual discrepancies or missing secondary details should not lead to significant score deductions. Only deduct points for major factual errors or completely incorrect responses. Ignore formatting and stylistic differences. Return only a single floating-point number between 0 and 1 (inclusive) as the first token.”

Strict Grader Prompt:

(System Prompt) “You are an extremely strict grader. Evaluate the prediction against the reference answer with absolute precision. Only factual correctness is taken into account, while formatting and stylistic differences are ignored. Scoring rules: Assign a score of 1 only if the prediction is factually identical to the reference; assign a score of 0 for any factual error, information omission or redundant addition. No intermediate scores are allowed. Return only one number, either 0 or 1, as the first token.”

Figure 10: Full system prompts for the original, lenient, and strict grading variants.

Multifieldqa_en_mixup: Please answer the following question based on the given passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Article: {context}

Please answer the following question based on the above passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Question: {input} **Answer:**

Factrecall_en: Please answer the following questions based on the given article.

Article: {context}

Please answer the following questions based on the above article.

Question: {input} **Answer:**

LLM_Evaluator (Metric):

(System Prompt) "You are a strict grader. Score how correct the prediction is compared to the reference answers. Only care about factual alignment, ignore formatting or style. Return only one floating number between 0 and 1 (inclusive) as the first token."

(User Prompt) **Model answer:** {pred}

Reference answers: {answers}

Return only the numeric score between 0 and 1.

Figure 11: Prompt templates used for MultifieldQA and FactRecall datasets, and LLM scoring metric.

Loogle_CR_mixup: Please answer the following question based on the given passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Article: {context}

Please answer the following question based on the above passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Question: {input} **Answer:**

Loogle_MIR_mixup: Please answer the following question based on the given passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Article: {context}

Please answer the following question based on the above passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Question: {input} **Answer:**

Figure 12: Prompt templates used for Loogle-CR and Loogle-MIR datasets.

HotpotWikiQA_mixup: Answer the question based on the given passages. Questions and answers are only relevant to some passages. Only give me the answer and do not output any other explanation and evidence.

Article: {context}

Please answer the following question based on the above passages. Questions and answers are only relevant to some passages. Only give me the answer and do not output any other explanation and evidence.

Question: {input} **Answer:**

Loogle_SD_mixup: Please answer the following question based on the given passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Article: {context}

Please answer the following question based on the above passages. Questions and answers are only relevant to one passage. Only give me the answer and do not output any other explanation and evidence.

Question: {input} **Answer:**

Figure 13: Prompt templates used for HotpotWikiQA and Loogle-SD datasets.

Symbol	Description
$\mathcal{D}, q$	Input document and user query
$N, \mathcal{V}$	Total number of textual chunks, and the set of all chunks $\{c_1, c_2, \dots, c_N\}$
$c_i, c_i $	The i -th textual chunk and its token count
$\phi(c_i, c_j)$	Scoring function measuring topical affinity/similarity between two chunks
$\mathcal{N}_{\text{topic}}^{c_i}$	Topical neighborhood of chunk c_i based on semantic similarity
$\mathcal{N}_{\text{phy}}^{c_i}$	Physical neighborhood of chunk c_i preserving positional adjacency
K	Number of the most relevant chunks in the topical neighborhood
$\mathcal{I}$	Initial pool providing a global retrieval prior
$\mathcal{F}^{(t)}$	Frontier pool at iteration step t
$s_{\text{init}}^c, s_c^{(t)}$	Initial relevance score of chunk c , and its exploration score at iteration t
$\mathcal{X}^{(t)}$	Active candidate pool alternating between $\mathcal{I}$ and $\mathcal{F}^{(t)}$ at step t
$c^{(t)}$	The chunk selected from the active pool at iteration step t
$\psi(q, c)$	Verification function recomputing query-chunk relevance
s_{veri}^c	Verified relevance score of the selected chunk
$\mathcal{E}$	Context set storing selected and verified chunk-score pairs
$N_{\text{max}}, T_{\text{max}}$	Maximum chunk capacity limit and total token budget for termination
K_{seed}	Number of top verified chunks selected for boundary expansion
$\mathcal{E}_{\text{core}}$	Core subset consisting of the top K_{seed} chunks
α	Expansion ratio controlling the fraction of boundary tokens used
$\Phi(c_i)$	Expanded version of core chunk c_i including its physical neighbors
$\mathcal{T}_{\text{context}}$	Final reconstructed and positionally reordered context sequence

Table 11: Notation and symbols used in our proposed method.

Algorithm 1 NEAR: Neighborhood-Aware Context Reconstruction

Require: Document D , query q , functions $\phi(\cdot, \cdot)$, $\psi(\cdot, \cdot)$, neighbor size K , max chunks $N_{\max}$, token budget $T_{\max}$, seed size K_{seed} , expansion ratio α

Ensure: Final context $\mathcal{T}_{\text{context}}$

```
1: procedure NEAR( $D, q, \phi, \psi, K, N_{\max}, T_{\max}, K_{\text{seed}}, \alpha$ )
2: // Phase 1: Neighborhood Graph Construction
3: Partition  $D$  into  $\mathcal{V} = \{c_1, c_2, \dots, c_N\}$ 
4: for each  $c_i \in \mathcal{V}$  do
5:    $\mathcal{N}_{\text{topic}}^{c_i} \leftarrow \text{TopK}_{c_j \in \mathcal{V} \setminus \{c_i\}} \phi(c_i, c_j)$   $\triangleright$  Topical neighbors via semantic similarity
6:    $\mathcal{N}_{\text{phy}}^{c_i} \leftarrow \{c_{i-1}, c_{i+1}\}$   $\triangleright$  Physical neighbors via positional adjacency
7: end for
8: // Phase 2: Neighborhood Graph Exploration
9:  $\mathcal{I} \leftarrow \text{Top-}N_{\max}\{(c, \text{BM25}(q, c))\}_{c \in \mathcal{V}}, \mathcal{F}^{(0)} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset, t \leftarrow 0$ 
10: while  $|\mathcal{E}| < N_{\max}$  and  $\sum_{(c,s) \in \mathcal{E}} |c| < T_{\max}$  do
11:    $\mathcal{X}^{(t)} \leftarrow \mathcal{I}$  if  $t$  is even else  $\mathcal{F}^{(t)}$   $\triangleright$  Alternate between initial and frontier pool
12:   if  $\mathcal{X}^{(t)} = \emptyset$  then
13:     break  $\triangleright$  No candidates remain, terminate early
14:   end if
15:    $c^{(t)} \leftarrow \arg \max_{c \in \mathcal{X}^{(t)}} s_c$   $\triangleright$  Select the highest-scored chunk
16:    $s_{\text{veri}}^{c^{(t)}} \leftarrow \psi(q, c^{(t)})$   $\triangleright$  Trust But Verify: re-score with stronger verifier
17:    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(c^{(t)}, s_{\text{veri}}^{c^{(t)}})\}$ ; Remove  $c^{(t)}$  from  $\mathcal{I}$  and  $\mathcal{F}^{(t)}$ 
18:   for each  $c_j \in \mathcal{N}_{\text{topic}}^{c^{(t)}}$  where  $c_j \notin \mathcal{E}$  do  $\triangleright$  Verified score propagation to topical neighbors
19:      $s_{c_j}^{(t+1)} \leftarrow s_{\text{veri}}^{c^{(t)}}$  if  $c_j \notin \mathcal{F}^{(t)}$  else  $\max(s_{c_j}^{(t)}, s_{\text{veri}}^{c^{(t)}})$ 
20:     Insert or update  $(c_j, s_{c_j}^{(t+1)})$  in  $\mathcal{F}^{(t+1)}$ 
21:   end for
22:    $t \leftarrow t + 1$ 
23: end while
24: // Phase 3: Input Context Reconstruction
25:  $\mathcal{E}_{\text{core}} \leftarrow \text{Top-}K_{\text{seed}}$  chunks in  $\mathcal{E}$  ranked by  $s_{\text{veri}}$   $\triangleright$  Select high-priority seed chunks for expansion
26: for each  $c_i \in \mathcal{E}_{\text{core}}$  do
27:    $\Phi(c_i) \leftarrow \text{Suffix}(c_{i-1}, \alpha) \oplus c_i \oplus \text{Prefix}(c_{i+1}, \alpha)$   $\triangleright$  Expand with boundary tokens of physical neighbors
28: end for
29: Replace  $c_i \in \mathcal{E}_{\text{core}}$  with  $\Phi(c_i)$ ; merge overlaps; sort  $\mathcal{E}$  by position in  $D$ 
30:  $\mathcal{T}_{\text{context}} \leftarrow \text{Concat}_{c \in \text{sorted}(\mathcal{E})}(c)$ 
31: return  $\mathcal{T}_{\text{context}}$ 
```
