Document Segmentation Matters for Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) enhances large language models (LLMs) by integrating external knowledge. A critical yet underexplored challenge in RAG is document segmentation, also known as document chunking. Existing widely-used rule-based chunking methods usually lead to suboptimal splits, where overly large chunks introduce irrelevant information and small chunks lack semantic coherence. Existing semantic-based approaches either require costly LLM calls or fail to adaptively group contextually related sentences. To address these limitations, we propose PIC (**P**seudo-**I**nstruction for document **C**hunking), a simple yet effective method that leverages document summaries as pseudo-instructions to guide chunking. By computing semantic similarity between sentences and the summary, PIC dynamically groups sentences into chunks that align with the document's key themes, ensuring semantic completeness and relevance to potential user instructions. Experiments on multiple open-domain question-answering benchmarks demonstrate that PIC can significantly improve retrieval accuracy (Hits@k) and end-to-end QA performance (Exact Match) without any additional training.

1 Introduction

Retrieval-Augmented Generation (RAG) integrates large language models (LLMs) with large-scale textual knowledge to enhance response quality and mitigate hallucinations in LLMs (Lewis et al., 2020; Karpukhin et al., 2020; Borgeaud et al., 2022; Guu et al., 2020; Edge et al., 2024; Sarthi et al., 2024; Yan et al., 2024). By dynamically retrieving real-time information from databases such as Wikipedia, RAG enables more relevant and accurate responses. RAG has been widely applied across domains such as law (Louis et al., 2023), medicine (Wu et al., 2024), and finance (Zhang et al., 2023), making it an essential component in real-world applications.

A standard RAG pipeline comprises three key stages: (1) Document segmentation into manageable units, named chunks; (2) Knowledge retrieval for input instructions; (3) Response generation based on retrieved chunks. While existing research mainly focuses on retriever improvements (Shi et al., 2024; Rubin et al., 2022) and system architecture design (Edge et al., 2024; Zhang et al., 2024), the foundational challenge of document segmentation remains critically underexplored. Effective document segmentation is crucial, as lengthy source documents in knowledge bases like Wikipedia exceed the processing capacities of both retrievers and LLMs.

The majority of chunking methods employed in RAG systems are rule-based, typically relying on heuristics such as fixed-length segmentation (Lewis et al., 2020; Karpukhin et al., 2020; Borgeaud et al., 2022). However, these methods usually suffer from inherent limitations in selecting appropriate chunk sizes. Larger chunks may include irrelevant information, which can both interfere with retrieval and generation, leading to hallucinations. Conversely, smaller chunks may lack sufficient semantic information, making it challenging for the LLM to generate accurate and coherent responses.

To address the above issues, certain studies have explored semantic-based chunking methods. Some research attempts to rewrite documents. For example, Chen et al. (2024) tries to transform the original text into independent, contextualized, and self-contained segments. This approach can integrate contextual knowledge, but it still relies on rule-based context window segmentation and fails to consider the adaptive combination of sentences. Other studies attempt to identify semantic breakpoints. Greg (2024) seperates adjacent sentences with the least semantic similarity, yet it lacks a global consideration of the entire document's context, leading to suboptimal chunking. Duarte et al. (2024) and Zhao et al. (2024) leverage LLMs' un-

derstanding of semantics for segmentation, but these models require extensive calls to large models, making them costly, time-consuming, and difficult to apply to large-scale text knowledge bases.

We aim to explore a new perspective on what constitutes good chunking. When a user inputs an instruction, our goal is to ensure that sentences relevant to the instruction are grouped within the same chunk, while unrelated sentences are placed in other chunks. This approach improves retrieval accuracy and prevents unrelated information from affecting LLM generation. However, in practical RAG applications, predefining chunks based on specific instructions is infeasible, as the instruction distribution is unknown when constructing chunks. Moreover, dynamically resegmenting a document for each instruction would incur unacceptable computational overhead.

To address the challenge of document segmentation without prior knowledge of real user instruction distributions, we propose PIC (**P**seudo-**I**nstruction for document **C**hunking), which utilizes document summaries as pseudo-instructions to guide segmentation. By generating a summary that preserves key document information (often central to real-world user queries), we compute semantic similarity scores between each sentence and the summary, then group sentences with scores above/below the average similarity threshold into coherent chunks. This approach ensures semantic completeness within chunks while isolating irrelevant content, thereby improving retrieval relevance and reducing hallucinations in LLMs. The method aligns chunk distributions with potential user instructions by prioritizing summary-aligned content aggregation and minimizing unrelated information within individual chunks.

To evaluate the effectiveness of different chunking methods, we process the latest English WikiDump, segmenting each document into chunks to serve as the knowledge base for RAG. We conduct experiments on multiple open-domain question-answering benchmarks, where our proposed method outperforms existing chunking approaches in terms of retrieval performance (Hits@k) and end-to-end QA performance (Exact Match). The results demonstrate that PIC achieves promising performance improvements across various experimental settings, proving to be an effective and generalizable chunking method. Our main contributions are summarized as follows:

- We propose a simple yet effective chunking method, PIC, which leverages an LLM to generate a pseudo-instruction and groups sentences into chunks based on their semantic relationship with the pseudo-instruction.
- We introduce a knowledge corpora PICWiki, derived from Wikipedia. This dataset consists of the original text of each document, the generated pseudo-instruction, and the chunks processed by our method. Serving as a text knowledge base for RAG, this dataset can significantly improve performance on knowledge-intensive tasks without requiring model retraining.
- We conduct extensive experiments on multiple QA benchmarks, demonstrating that our method achieves significant performance improvements in both retrieval accuracy and end-to-end QA system performance.

2 Related Works

Retrieval-Augmented Generation. RAG has been proven to be an effective approach for addressing knowledge-intensive NLP tasks (Lewis et al., 2020; Borgeaud et al., 2022; Izacard et al., 2022; Huang et al., 2024; Guu et al., 2020). Existing RAG methods mainly focus on improving the retriever performance and designing the whole RAG pipeline. As for the retriever improvement, Karpukhin et al. (2020) proposes training a dense passage retriever to replace sparse retrieval methods like BM25 (Robertson et al., 1995); Yu et al. (2024) introduce reranking mechanisms after retrieval; Sawarkar et al. (2024) and Juvekar and Purwar (2024) integrate multiple retrieval strategies; and Edge et al. (2024) incorporates knowledge graphs, refining retrieval through graph traversal. On the other hand, some works focus on enhancing the reliability of language model generation. For example, Xu et al. (2023) compresses retrieved documents to extract essential information; Yan et al. (2024) corrects and rewrites retrieved knowledge; Asai et al. (2023) introduces self-evaluation and reflection to selectively apply knowledge during generation; and Zhang et al. (2024) trains the generation model with distractor documents to improve its robustness against misleading information. These approaches effectively integrate up-to-date information and reduce model hallucinations, making them indispensable for modern generative models. However, most of these methods primarily focus on improving the retriever and generation mechanisms within RAG while overlooking the impact

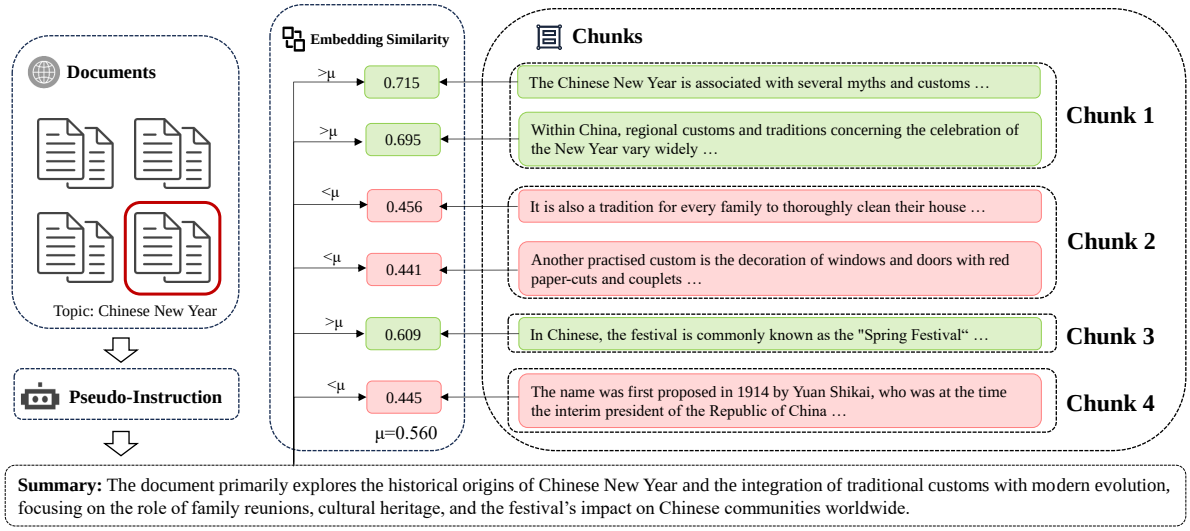

Figure 1: Illustration of our proposed PIC, where the document is dynamically segmented based on the similarity between sentences and the pseudo-instruction.

of chunk quality on downstream task performance.

Retrieval Units in RAG. The RAG process begins by splitting long documents into smaller units, which are stored in a knowledge base. Relevant units are then retrieved based on the user's instructions. This retrieval unit can be defined at various natural text granularities such as sentence, paragraph or document-level (Gao et al., 2023). Some specific rules are also used to manually segment documents into chunks by fixed-size (Borgeaud et al., 2022), punctuation (Liu, 2022), or phrase (Lan et al., 2023). Several advanced chunking approaches further incorporate semantic relationships to enhance retrieval effectiveness. For instance, Greg (2024) suggests computing embedding similarity between adjacent sentences to determine chunking points. Since LLMs capture semantic dependencies better, many model-based methods have been proposed. Chen et al. (2024) uses LLMs to rewrite sentences into contextualized segments with pronouns restored to their corresponding entities. Duarte et al. (2024) leverages LLMs to detect semantic transition points in a document, treating them as chunk boundaries. Zhao et al. (2024) uses perplexity (PPL) to measure semantic coherence, identifying local PPL minima as optimal chunk boundaries.

3 Methodology

In the context of RAG, an effective chunking strategy is critical for both retrieval and generation performance. Given a document $D = \{s_1, s_2, ..., s_n\}$ containing n sentences, we aim to segment D into a set of semantically coherent chunks $C = \{c_1, c_2, .., c_m\}$, where each chunk c_i consists of one or more combined sentences, and m represents the total number of chunks.

Our method consists of two key steps: 1) Generating pseudo-instructions. 2) Performing semantic chunking based on pseudo-instructions, as shown in Figure 1. We will provide a detailed explanation of both steps below.

3.1 Pseudo-Instruction Generation

In this paper, we propose using pseudo-instructions to guide chunking without knowing the real distribution of user-instructions. Specifically, we leverage document summaries as pseudo-instructions to construct semantically coherent chunks. Summaries can preserve key document information, allowing us to group adjacent sentences that share semantic alignment with the summary while separating irrelevant content.

Specifically, given a document D, we use a summarizer S to generate a summary s_D for D. The generated summary should be as concise and informative as possible to effectively serve as a pseudo-instruction.

3.2 Chunking Based on Pseudo-Instructions

For a given document D, after obtaining the summary s_D, we perform chunking by computing the semantic similarity between each sentence and the summary. Adjacent sentences with above-average similarity are grouped into coherent chunks, en-

suring semantic completeness within chunks while minimizing unrelated information. Likewise, adjacent sentences with below-average similarity are also grouped together.

Specifically, we first use an embedding model to encode both the summary and sentences. Let ϕ denote the embedding function, then the embeddings for the sentences and the summary are computed as: $e_{s_i} = \phi(s_i), e_{s_D} = \phi(s_D)$. Next, we compute the cosine similarity between each sentence embedding and the summary embedding as follows:

$$r_i = sim(s_i, s_D) = \frac{e_{s_i} \cdot e_{s_D}}{\|e_{s_i}\| \cdot \|e_{s_D}\|}. \quad (1)$$

Based on the similarity between each sentence and the summary, we group adjacent sentences into the same chunk if their similarity scores exceed a predefined threshold. Similarly, sentences with similarity scores below the threshold are also placed in the same chunks. The chunking rules are as follows:

$$C_{rel} = \left\{ [s_i : s_j] \;\middle|\; \begin{array}{l} r_i,\ r_{i+1}, ..., r_j \geq \tau, \\ r_{i-1}, r_{j+1} < \tau \end{array} \right\}, \quad (2)$$

$$C_{irr} = \left\{ [s_i : s_j] \;\middle|\; \begin{array}{l} r_i,\ r_{i+1}, ..., r_j < \tau, \\ r_{i-1}, r_{j+1} \geq \tau \end{array} \right\}, \quad (3)$$

$$C \quad = C_{rel} + C_{irr}. \quad (4)$$

Here, τ is the threshold that controls the chunking condition, $[s_i : s_j]$ represents a chunk consisting of sentences from s_i to s_j $(i \leq j)$, and C_{rel}, C_{irr} represent chunks that are relevant and irrelevant with the pseudo-instruction, respectively. The final result, C, is the segmented chunks for the document D using our method.

For the selection of τ, we use a dynamic threshold, as a fixed one would result in varying similarity levels across documents, making manual selection difficult. Specifically, we set the threshold τ as the mean similarity μ between all sentences in document D and its summary, which is calculated as $\mu = \frac{1}{n}\Sigma_{i=1}^{n} r_i$. We find this mean value performing best among all values through experiments described in Section 6.2.

4 Experimental Settings

To evaluate the performance of different document chunking methods in RAG scenarios, we adopt Wikipedia as the knowledge corpus and conduct experiments on several knowledge-intensive tasks using two different LLMs. In this section, we present the details and results of our extensive experiments.

4.1 Baselines

Following previous work (Gao et al., 2023), we mainly choose the following chunking methods as our baseline methods: 1) Rule-based Chunking. Most existing RAG systems utilize rule-based chunking methods for processing the corpus. In our experiments, we evaluate three widely-used rule-based chunking approaches. Specifically, the documents in the corpus are segmented in three different ways: sentence-by-sentence (**Sentence**), paragraph-by-paragraph (**Paragraph**), and fixed-size windows (**Fixed-size**). In these methods, each chunk corresponds to a single sentence, a single paragraph, or a span of fixed length, respectively. 2) Semantic-based Chunking: These methods segment documents based on the semantic consistency of adjacent sentences. We present the results of the semantic chunking method proposed by Greg, 2024 (**Semantic**), which computes the similarity between adjacent sentence embeddings and segments when the similarity falls below a certain threshold. 3) Generation-based Chunking: These methods attempt to generate text chunks based on given documents. Chen et al. (2024) (**Proposition**) rewrites the sentences into self-contained atomic expressions called propositions, which will be used for further retrieval.

4.2 Knowledge Corpus

We use the latest English Wikipedia dump (2024-12-01), containing over 6 million documents, as our textual knowledge base. We extract the cleaned text using WikiExtractor[1]. Then we apply different processing techniques based on the specific chunking methods: 1) As for the rule-based chunking method, we adopt the NLTK package[2] to split document text to sentences; we split text to paragraphs by the new line character "\n" in the original text; and we split the document into fixed-size chunks of 100 words each, while preserving sentence boundaries. 2) As for the semantic-based chunking method, we first build embeddings for each sentence using bge-large-en-v1.5 (Xiao et al., 2024), and calculate the similarity between every two adjacent sentences. Then, we compute the 20th percentile of all similarity scores and use it as a threshold[3]. Any pair of adjacent sentences

[1] https://github.com/attardi/wikiextractor

[2] https://www.nltk.org/api/nltk.tokenize.sent_tokenize.html

[3] We experimented with thresholds at the 5th, 20th, 35th, and 50th percentiles and selected the 20th percentile, which yielded the best performance, as the baseline for comparison.

Method	NQ		TriviaQA		WebQ		SQuAD		EQ		PopQA		Avg.	
	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20
Sentence	33.0	48.4	36.9	44.7	34.4	50.8	39.9	52.6	53.4	65.0	44.5	58.4	40.4	53.3
Paragraph	48.3	62.3	40.3	47.6	43.3	58.9	44.4	57.9	61.5	73.3	50.3	64.8	48.0	60.8
Fixed-size	60.7	71.7	40.7	48.2	49.1	63.7	47.1	63.1	66.0	76.8	63.2	77.3	54.5	66.8
Semantic	59.4	71.2	41.2	48.9	53.5	64.4	46.8	61.2	69.9	79.2	65.1	79.4	56.0	67.4
Proposition	60.0	71.5	40.8	48.1	52.5	65.3	**56.0**	**68.5**	69.9	78.8	67.1	81.3	57.7	68.9
PIC	**61.0**	**71.6**	**42.3**	**49.2**	**54.5**	**66.9**	53.0	67.0	**71.1**	**80.3**	**68.7**	**81.9**	**58.4**	**69.5**

Table 1: Retrieval performance (Hits@k) across different methods. The proposed PIC method achieves the highest average performance across all datasets, outperforming baseline methods such as semantic and proposition-based chunking in both Top-5 and Top-20 retrieval accuracy.

with a similarity score below this threshold is split, which is defined as the chunk boundaries. 3) As for the proposition method, we use the open-source Propositionizer model[4], first segmenting text into 100-word passages and then apply the model to rewrite passages into propositions.

Notably, we utilize PIC to process the same Wikipedia corpus, generating the resulting datasets named PICWiki. The dataset can be seamlessly integrated to enhance the performance of RAG systems in a plug-and-play manner. We will release PICWiki to facilitate the progress of RAG.

4.3 Evaluation Benchmarks

We evaluate the chunking methods on the following six OpenQA benchmarks, including Natural Questions (NQ, Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (WebQ, Bordes et al., 2014), SQuAD (Rajpurkar et al., 2016), EntityQuestions (EQ, Sciavolino et al., 2021) and PopQA (Mallen et al., 2023), which focus on reading comprehension and answering natural language queries using Wikipedia or structured knowledge bases.

To fairly compare different chunking methods on these benchmarks, we mainly evaluate the performance of the methods from these two aspects: 1) **Retrieval Performance**: We evaluate the retrieval performance using Hits@k, measuring whether the top-k retrieved chunks contain the correct answer. 2) **End-to-end QA Performance**: To demonstrate the impact of different chunking methods on the question-answering abilities of the RAG system, we also evaluate the end-to-end QA performance by Exact Match (EM), which is calculated as the percentage of questions for which the predicted answer exactly matches the ground truth.

[4]https://huggingface.co/chentong00/propositionizer-wiki-flan-t5-large

4.4 Implementation Details

For the model implementation, we use GPT-4o-mini (OpenAI, 2024) as the summarizer to generate summaries for all documents with the prompt provided in Appendix C. We use bge-en-large-v1.5 as both the embedding model for chunking and the retriever model for RAG, and evaluate the end-to-end QA performance with two LLMs: Qwen2.5-7B-Instruct (Team, 2024) and Meta-Llama-3-8B-Instruct (Grattafiori et al., 2024). We retrieve the top 5 and top 20 chunks separately and evaluate both their retrieval performance and end-to-end QA performance. The retrieved chunks are then concatenated with the input instructions using the prompt template provided in Appendix D. We use the greedy sampling strategy for answer generation and the maximum length is set as 512.

5 Main Results

5.1 Retrieval Performance

Table 1 presents the retrieval performance of PIC compared to other baseline chunking methods. PIC achieves the highest average retrieval performance across six datasets, with the highest 2-point increase on WebQ. This suggests that our PIC method captures semantic information more effectively, leading to superior retrieval outcomes. We also conduct a case study in Appendix B.

Despite its effectiveness on most datasets, PIC exhibits suboptimal performance on SQuAD. One potential reason for this gap is that SQuAD is more akin to a reading comprehension dataset, with 100k questions derived from only 536 documents. This makes it difficult for a retriever trained on general data to capture the relationship between instructions and chunks. In contrast, Proposition eliminates this bias by rewriting the original text, leading to improved performance. Similar phenomenon is also observed by Lee et al., 2019.

Method	NQ		TriviaQA		WebQ		SQuAD		EQ		PopQA		Avg.	
	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20
Qwen2.5-7B-Instruct														
Sentence	29.8	35.0	55.4	60.2	30.0	34.3	34.8	39.6	42.6	46.4	36.9	41.2	38.2	42.8
Paragraph	38.6	40.9	60.9	64.1	34.3	37.9	35.7	39.6	49.0	52.4	42.4	46.2	43.5	46.8
Fixed-size	42.9	44.8	61.2	64.8	35.9	40.3	33.8	38.3	51.8	55.0	47.5	49.7	45.5	48.8
Semantic	42.5	44.5	61.7	63.5	36.8	40.4	28.3	32.2	53.7	55.8	47.8	48.7	45.1	47.5
Proposition	42.5	44.3	61.5	64.6	36.9	41.0	**42.7**	**45.4**	52.7	54.6	51.2	51.1	47.9	50.2
PIC	**45.7**	**48.0**	**63.3**	**66.1**	**40.1**	**42.7**	39.4	43.0	**58.7**	**61.5**	**52.2**	**53.8**	**49.9**	**52.5**
Meta-Llama-3-8B-Instruct														
Sentence	32.8	37.0	59.1	63.0	30.6	35.5	35.3	40.5	42.9	47.5	38.8	43.9	39.9	44.6
Paragraph	40.1	42.4	63.0	65.9	36.4	38.9	36.8	41.3	50.3	53.8	44.5	49.0	45.2	48.5
Fixed-size	44.6	46.1	63.2	67.0	36.0	39.2	34.6	39.8	52.4	55.4	50.7	52.2	46.9	50.0
Semantic	43.9	46.0	63.3	65.4	37.1	40.4	29.2	33.6	54.7	56.1	50.9	51.8	46.5	48.9
Proposition	44.4	46.7	63.9	67.1	38.0	40.7	**44.8**	**47.8**	55.1	56.3	54.5	55.1	50.1	52.3
PIC	**45.6**	**48.3**	**64.6**	**67.2**	**39.4**	**41.5**	40.5	44.2	**58.8**	**61.5**	**56.9**	**58.5**	**51.0**	**53.5**

Table 2: End-to-end question-answering performance across different document segmentation methods. We present results with two different LLMs as the generation models, evaluated by Exact Match (EM).

5.2 End-to-end QA Performance

Similar conclusions can be drawn regarding QA performance. As shown in Table 2, experiments with two different LLMs demonstrate that PIC consistently achieves the best results. When using Qwen as the LLM, PIC shows an average improvement of 2 points across six datasets, with a remarkable 5.7-point increase on the EQ dataset. When using Llama-3, PIC also demonstrates an average improvement of nearly 1 point.

It can be observed that the accuracy of the QA task is significantly lower than that of retrieval, indicating that sometimes, even when the correct relevant text is retrieved, the model still fails to provide the correct answer. This may be due to the retrieved text containing too much irrelevant information, making it difficult for the LLM to effectively capture the answer. However, among all methods, our approach exhibits the smallest performance drop. Taking the results on Qwen as an example, under the top-5 setting, our method decreases by 8.5 points (58.4 $\rightarrow$ 49.9), while the proposition method decreases by 9.8 points (57.7 $\rightarrow$ 47.9) and the Semantic method by 10.9 points (56.0 $\rightarrow$ 45.1). We attribute this to the reduced irrelevant information in the chunks, making the LLM less prone to hallucinations.

6 Further Analysis

6.1 Impacts of the Pseudo-Instruction

The core idea of our method is to construct pseudo-instruction (PI) to guide chunking, ensuring that the segmented chunks better align with the possible real instruction distribution. In our implementation, we choose summary as the PI because it encapsulates the main information of the document, allowing the document to be segmented based on its relevance to the central theme. To better understand the importance of PI, we conduct an ablation study by modifying its implementation. Specifically, we implement the following variations of PI: 1) Random: For each document, we use the summary of a randomly selected different document as its PI instead of its own summary. 2) Sent: For each document, we use a randomly chosen sentence from the document as its PI. 3) Mean: For each document, we compute the mean value of all the sentence embeddings and use it as the PI embedding for chunking.

We follow the experimental setup described in Section 4 and evaluate the retrieval and QA performance under different PI implementations. The results are presented in Table 3. It can be seen that using summaries as the PI achieves the best performance. PIC-Random performs the worst, as the random-selected summary does not contain any information from the document itself, resulting in chunks with little semantic coherence. PIC-Sent contains document-specific information, but its information capacity is too limited, and the selection of a single sentence is highly random, leading to suboptimal chunking. PIC-Mean performs relatively well, likely because it integrates information from all sentences in the document. However, av-

Method	NQ		TriviaQA		WebQ		SQuAD		EQ		PopQA		Avg.	
	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20	Top5	Top20
Retrieval Performance (Hits@k)														
PIC	**61.0**	**71.6**	**42.3**	**49.2**	**54.5**	**66.9**	**53.0**	**67.0**	**71.1**	**80.3**	**68.7**	**81.9**	**58.4**	**69.5**
Random	56.6	68.8	38.7	45.5	50.2	62.5	49.1	62.4	67.5	76.6	65.4	78.6	54.6	65.7
Sent	56.8	69.1	38.7	45.2	50.8	63.4	49.2	62.2	67.9	76.8	66.3	78.7	54.9	65.9
Mean	57.7	69.4	39.6	46.5	51.7	63.9	50.3	64.3	68.4	78.0	67.3	80.0	55.8	67.0
Question-Answering Performance (EM)														
PIC	**45.7**	**48.0**	**63.3**	**66.1**	**40.1**	**42.7**	**39.4**	**43.0**	**58.7**	**61.5**	**52.2**	**53.8**	**49.9**	**52.5**
Random	39.6	43.1	57.3	60.5	35.1	38.0	34.6	38.4	53.6	55.6	48.8	50.6	44.8	47.7
Sent	40.6	43.0	58.3	60.9	34.7	38.4	35.4	38.4	54.0	57.4	49.4	51.4	45.4	48.3
Mean	40.8	43.2	59.5	62.4	35.9	40.8	36.3	40.1	54.8	58.5	50.7	51.8	46.3	49.5

Table 3: Performance comparison of different pseudo-instruction implementations. For question-answering performance evaluation, we adopt Qwen2.5-7B-Instruct as the backbone model.

eraging embeddings weakens its ability to capture semantic structures, making it less effective than the summary-based approach.

6.2 Impacts of Chunking Threshold

In Section 3, our PIC algorithm selects the **mean** similarity between sentences and the summary as a threshold for chunking. However, this threshold requires further consideration. Our method aims to distinguish between sentences that are relevant to the pseudo-instruction and those that are not. A higher threshold results in shorter and fewer relevant chunks, and vice versa.

To explore this, we select five different thresholds ranging from $\mu - 2\sigma$ to $\mu + 2\sigma$ and conduct experiments under the same setting as Section 4. The results are shown in the figure 2. Here, μ represents the mean, and σ represents the standard deviation.

The results indicate that setting the threshold at μ yields the best retrieval and QA performance. As the threshold deviates in either direction (becoming too high or too low), performance gradually declines. We attribute this to the increasing mixture of relevant and irrelevant sentences within chunks as the threshold shifts. This reduces the distinction between chunks and, in extreme cases (when the threshold approaches 0 or 1), degrades the process into no chunking at all, ultimately leading to performance deterioration.

6.3 Real-Instruction for Document Chunking

When constructing chunks, we typically lack access to the actual user instruction distribution. Therefore, we propose the PIC method, using pseudo-instructions as an approximation of real user instructions. But if the actual instructions are known in advance and used for chunking, it may yield optimal results. To validate this hypothesis, we conducted experiments on the NaturalQuestions (Kwiatkowski et al., 2019) and Qasper (Dasigi et al., 2021) datasets. The NaturalQuestions (NQ) dataset comprises real anonymized queries issued to the Google search engine, paired with corresponding Wikipedia pages. The Qasper dataset consists of human-written QA pairs based on NLP papers. For each dataset, we use the documents it provides as the corpus, instead of Wikipedia. For the Real-Instruction Chunking (RIC), we use the query of each document to calculate similarities with the sentences. The results are shown in Table 4. It can be observed that chunking with real instructions yields the optimal results, which aligns with the viewpoint we presented in Section 1.

Method	NQ	Qasper
Semantic	37.8	14.9
Proposition	38.3	14.8
PIC	38.8	15.6
RIC	**39.2**	**16.2**

Table 4: Performance comparison of Real-Instruction Chunking with other methods. We retrieve the top-5 chunks, and evaluate the QA performance using Qwen-2.5-7B-Instruct. We use Exatch Match for evaluation on NQ, and F1 for Qasper.

6.4 Chunk Analysis

Distribution of Chunk Lengths. Different chunking methods yield varying length distributions, which may influence chunking effectiveness. To explore this, we visualized the length distribu-

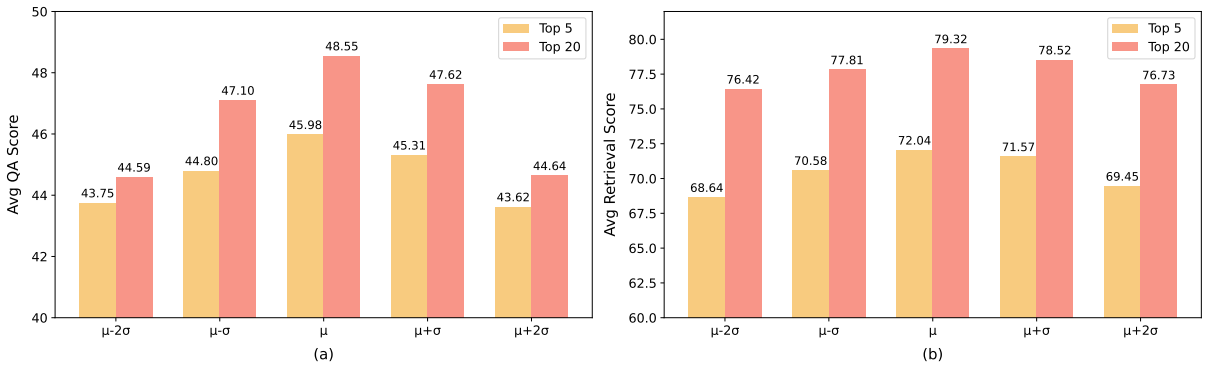

Figure 2: The influence of different chunking thresholds. Figure (a) represents the average end-to-end QA performance, and figure (b) represents the average retrieval performance.

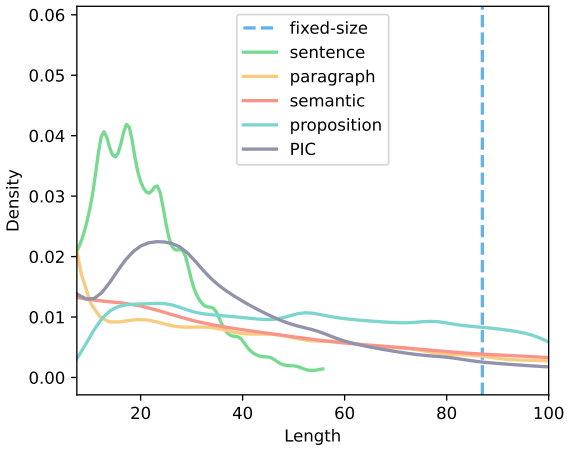

Figure 3: Length Distribution of Different Chunking Methods.

Dataset	Top-5	Top-20	Golden
NQ	70.1	65.7	66.7
TriviaQA	73.1	67.7	68.4
WebQ	77.0	72.7	66.7
SQuAD	60.7	57.9	65.8
EQ	67.2	63.7	100.0
PopQA	64.3	61.5	91.0

Table 5: Proportion of relevant chunks in retrieved PIC Chunks. Top-5 and Top-20 refer to the top 5 and top 20 retrieved chunks, respectively, and Golden represents the retrieved chunks that contain the correct answer.

tion for each method, as shown in Figure 3, with detailed statistics provided in Appendix A. The overall distribution patterns suggest that chunk length itself does not directly correlate with retrieval or QA performance in RAG. Instead, the primary factor affecting downstream performance is how the text is structured within each chunk. Notably, our proposed PIC method maintains a relatively shorter average chunk length, which helps mitigate the input length burden on the LLM.

Distribution of Retrieved PIC Chunks. The relevant chunks (C_{rel}) in PIC are expected to contain more key information from the document, making them easier to be retrieved. To evaluate this, we calculate the proportion of relevant chunks retrieved by the PIC method across different datasets, as shown in Table 5. The results indicate that in all cases, relevant chunks account for more than 50% of the retrieved chunks, demonstrating the effectiveness of the PIC method in categorizing chunks and making topic-related chunks more retrievable. Moreover, on the EQ and PopQA datasets, nearly all retrieved golden chunks are relevant chunks. We attribute this to the entity-centric nature of the datasets, which focus on less popular, long-tail entities. In this scenario, the PIC-relevant chunks overlap significantly with user instructions, which further reinforces our conclusion.

7 Conclusions

In this paper, we identify document segmentation as a critical yet underexplored challenge in retrieval-augmented generation (RAG) systems. We propose PIC, a novel document chunking method that leverages document summaries as pseudo-instructions to guide semantic chunking. By dynamically grouping sentences based on their relevance to document topic, PIC creates chunks that better align with potential user instructions while maintaining semantic coherence. Extensive experiments across six QA benchmarks demonstrate that our method achieves significant improvements in both retrieval accuracy and end-to-end QA performance compared to existing approaches, without requiring additional model retraining.

Limitations

Our work focuses on creating chunks that better align with potential user instructions while maintaining semantic coherence. One limitation of our approach is the computational overhead introduced during summary generation. While this overhead is significantly lower than methods requiring multiple LLM calls per document, future work should explore more efficient strategies for generating high-quality pseudo-instructions. Additionally, our current evaluation is limited to general-domain tasks, leaving the exploration of specialized scenarios requiring cross-document reasoning for future investigation.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics.

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

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.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey.

Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Kamradt Greg. 2024. semantic chunking.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Yufei Huang, Xu Han, and Maosong Sun. 2024. Fast-FiD: Improve inference efficiency of open domain question answering via sentence selection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6262–6276, Bangkok, Thailand. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models. *Preprint*, arXiv:2208.03299.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Kush Juvekar and Anupam Purwar. 2024. Cos-mix: Cosine similarity and distance fusion for improved information retrieval. *Preprint*, arXiv:2406.00638.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural*

Language Processing (EMNLP), pages 6769–6781, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is all you need. *Preprint*, arXiv:2307.06962.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. 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*, volume 33, pages 9459–9474. Curran Associates, Inc.

Jerry Liu. 2022. LlamaIndex.

Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. 2023. Interpretable long-form legal question answering with retrieval-augmented large language models. *Preprint*, arXiv:2309.17050.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2024. GPT-4o mini: Advancing Cost-Efficient Intelligence. *OpenAI*.

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

Stephen E. Robertson, Susan Walker, Steve Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at trec-3. *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 155–161.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box 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)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. 2024. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. *Preprint*, arXiv:2309.07597.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Preprint*, arXiv:2407.02485.

Boyu Zhang, Hongyang Yang, Tianyu Zhou, Muhammad Ali Babar, and Xiao-Yang Liu. 2023. Enhancing financial sentiment analysis via retrieval augmented large language models. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 349–356, New York, NY, USA. Association for Computing Machinery.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *Preprint*, arXiv:2403.10131.

Jihao Zhao, Zhiyuan Ji, Yuchen Feng, Pengnian Qi, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2024. Meta-chunking: Learning efficient text segmentation via logical perception.

Appendix

A Statistics of Chunk Lengths

	Mean	Std
Sentence	18.8	12.7
Paragraph	42.2	48.0
Fixed-size	87.8	26.6
Semantic	78.6	88.9
Proposition	58.1	32.0
PIC	42.8	38.3

Table 6: Length Statistics of Different Chunking Methods, counted by words.

The detailed statistics of the length of the chunks under different aggregation methods are shown in Table 6.

B Case Study

We demonstrate the difference of our chunking method with semantic chunking in Table 9. We select a query "where was the statue of liberty originally built" from the NQ dataset with its answer as "France". To retrieve all the relevant information, the retrieved chunk must both include "built in France" and "the Statue of Liberty", in sentence 3 and 5, respectively. However, as we can see from the table, the semantic chunking method fails to group sentence 3 and sentence 5 in one chunk. Instead, it splits sentence 3 and 4 due to the relatively low similarity between them (In the table, sim_{next} represents the similarity of one sentence with the next sentence). This results in a low similarity ($sim_{red-chunk-with-query}$ in the table) between the query and the chunk of sentence 3, where the answer "in France" lies, and leads to a retrieval failure. The PIC method treats sentence 3 to 5 as relevant sentences, and groups them into one chunk based on sim_{pi}, which represents the similarity between the pseudo-instruction (PI) and the sentence. This chunk, with both "built in France" and "the Statue of Liberty" in it, has a higher similarity score with the query ($r_{green-chunk-with-query}$), and can be easily retrieved by the retriever.

C Prompt For Summary

The prompt we use for writing summary for each document is shown in Table 7.

Summarizer Prompt
Read this passage: {context} Write a summary for this passage. Your summary should be concise and informative. Give the summary directly, without any other text.

Table 7: Prompt used when generating summaries for wiki documents.

RAG Prompt
Refer to the passages below and answer the following question with just a few words. {chunk 1} {chunk 2} ... Refer to the context above and answer the following question with just a few words. Question: {Question} The answer is

Table 8: Prompt used when concatenating queries with retrieved chunks.

D Prompt For Question Answering

The prompt we use for generating answers in QA tasks is shown in Table 8.

Query: where was the statue of liberty originally built?
Answer: France.

Semantic:

...

Fundraising proved difficult, especially for the Americans, and by 1885 work on the pedestal was threatened by lack of funds. ($sim_{next} = 0.566$)
Publisher Joseph Pulitzer, of the "New York World", started a drive for donations to finish the project and attracted more than 120,000 contributors, most of whom gave less than a dollar. ($sim_{next} = 0.4314$, cut!)
The statue was built in France, shipped overseas in crates, and assembled on the completed pedestal on what was then called Bedloe's Island. ($sim_{next} = 0.4958$, cut!)
The statue's completion was marked by New York's first ticker-tape parade and a dedication ceremony presided over by President Grover Cleveland. ($sim_{next} = 0.538$)
The statue was administered by the United States Lighthouse Board until 1901 and then by the Department of War; since 1933, it has been maintained by the National Park Service as part of the Statue of Liberty National Monument, and is a major tourist attraction.

...

$r_{red-chunk-with-query}$=0.5713

PIC:

...

Fundraising proved difficult, especially for the Americans, and by 1885 work on the pedestal was threatened by lack of funds. ($sim_{pi} = 0.493$, irrelevant)
Publisher Joseph Pulitzer, of the "New York World", started a drive for donations to finish the project and attracted more than 120,000 contributors, most of whom gave less than a dollar. ($sim_{pi} = 0.445$, irrelevant)
The statue was built in France, shipped overseas in crates, and assembled on the completed pedestal on what was then called Bedloe's Island. ($sim_{pi} = 0.572$, relevant)
The statue's completion was marked by New York's first ticker-tape parade and a dedication ceremony presided over by President Grover Cleveland. ($sim_{pi} = 0.557$, relevant)
The statue was administered by the United States Lighthouse Board until 1901 and then by the Department of War; since 1933, it has been maintained by the National Park Service as part of the Statue of Liberty National Monument, and is a major tourist attraction. ($sim_{pi} = 0.703$, relevant)

...

$r_{green-chunk-with-query}$=0.73

Pseudo-Instruction: The Statue of Liberty, a colossal neoclassical sculpture designed by Frric Auguste Bartholdi with a metal framework by Gustave Eiffel, was dedicated in New York Harbor on October 28, 1886, as a gift from France to the United States. Inspired by the Roman goddess Libertas, the statue symbolizes freedom and welcome to immigrants. Conceived in 1865 by French historian douard de Laboulaye, its construction was delayed by the Franco-Prussian War until fundraising efforts resumed in the 1870s. Joseph Pulitzer's 1885 fundraising campaign in the U.S. attracted over 120,000 donors, enabling the pedestal's completion. Assembled on Liberty Island, the statue oxidized to its green color within twenty years and became an iconic landmark associated with immigration, especially linked to nearby Ellis Island. Managed initially by the Lighthouse Board and the Department of War, it has been maintained by the National Park Service since 1933. The statue has undergone several restorations, notably for its centennial in 1986, and has been closed and reopened multiple times due to events like the September 11 attacks and Hurricane Sandy.· · ·

Table 9: Case Study