Preserving LCO Requirement Relationships through Chunking for RAG on Nuclear Power Plant Technical Specifications

Woo Shin Kim*, Il Hoon Park, Yong Joon Shin
GNP System Co., Ltd., Daejeon, Republic of Korea
**Corresponding author: kws@gnpsys.com*

***Keywords :** RAG, Chunking Strategy, LCO , Technical Specifications, Nuclear Safety Documents

1. Introduction

Large Language Models (LLMs) have recently been applied in the nuclear field to technical document understanding, question answering, and operator support. To process large volumes of technical documents, these systems commonly divide documents into smaller units. However, nuclear technical documents, particularly Limiting Conditions for Operations (LCOs) in Technical Specifications, consist of closely related requirements, and these relationships may be disrupted during document segmentation. Specifically, an LCO is organized around *Condition, Required Action*, and *Completion Time*, and the relationships among these elements determine the actions required when the LCO is not met. At present, determinations regarding applicable LCO requirements are generally made by manually referring to extensive Technical Specifications documents. To support this process, computerized support systems for Technical Specifications surveillance [1] and retrieval systems for similar historical LCO cases [2] have been developed. More recently, RAG and agentic RAG have been applied to question answering and decision support for nuclear technical documents [3].

In these RAG applications, how source documents are divided into retrieval units can affect both retrieval and downstream question answering performance. Irrelevant context can reduce answer accuracy [4], and previous studies have shown that chunking strategy affects RAG performance [5]. Research on question answering in regulatory domains has also shown that methods effective across broad domains do not necessarily provide the same performance in specialized regulatory domains, and that contextualization techniques developed for other domains cannot be directly applied to regulatory documents without adjustment [6]. Recent studies on RAG in the nuclear field have addressed retrieval architectures, document processing, and provenance [3].

However, these studies have not examined how the structure of LCO requirements should be reflected in the construction of retrieval units. Because related LCO requirements often need to be interpreted together, common chunking methods based on text length, overlap, or semantic similarity can separate information that belongs to the same context. As a result, retrieved text may contain only part of the information needed to interpret the applicable LCO state.

To address this issue, we propose condition-aware chunking, which groups the requirements associated with each *Condition* into a single retrieval unit. For comparison, fixed-length, sliding-window, and semantic chunking are used as conventional approaches whose chunk boundaries are not specifically defined by the LCO structure. A smaller retrieval unit is also examined through action-logic chunking, which uses individual Required Actions as retrieval units while preserving their relationships with the parent *Condition* and other actions. The five chunking strategies are evaluated in terms of retrieval effectiveness and downstream answer accuracy.

2. Condition-Aware Chunking for LCO Requirements

2.1 LCO Structure and Text Representation

3.4 REACTOR COOLANT SYSTEM (RCS)

3.4.3 RCS Pressure and Temperature (P/T) Limits

LCO 3.4.3 RCS pressure, RCS temperature, and RCS heatup and cooldown rates shall be maintained within the limits specified in the PTLR.

APPLICABILITY: At all times.

ACTIONS

CONDITION	REQUIRED ACTION		COMPLETION TIME
A. ----------NOTE---------- Required Action A.2 shall be completed whenever this Condition is entered. ---------- Requirements of LCO not met in MODE 1, 2, 3, or 4.	A.1	Restore parameter(s) to within limits.	30 minutes
	AND		
	A.2	Determine RCS is acceptable for continued operation.	72 hours

Fig. 1. Example of an LCO ACTIONS table structure.

The LCO statement specifies the operating requirement for a system or component, while the APPLICABILITY statement identifies the MODE or specified operating condition in which the requirement applies. When the LCO is not met, the *Required Action* for the applicable *Condition* must be performed within the specified *Completion Time*. The ACTIONS table links each *Condition* to its *Required Action* and corresponding *Completion Time*, and a single *Condition* may contain multiple *Required Actions*. Relationships among these actions are represented by the AND and OR logical connectors together with the indentation and grouping of the table. Some actions also include a NOTE or a reference to another provision. Figure 1 shows an example of these elements in an LCO ACTIONS table.

For text processing, the LCO statement and the APPLICABILITY statement were extracted separately from the ACTIONS table. The ACTIONS table was parsed by its *Condition, Required Action,* and *Completion Time* columns. Each *Condition* was associated with the *Required Action* within its table region, and the corresponding *Completion Time*, logical connectors, notes, and references were retained. The grouping represented by the table indentation was recovered from the *Required Action* labels and their logical connectors rather than preserved as visual indentation.

The extracted content was then converted into text representations for chunking. For methods operating on continuous text, the LCO content was arranged in source order without adding structural markers. The parsed associations among *Condition, Required Action*, and *Completion Time* were retained separately for methods that use the LCO structure. The construction of retrieval units from these representations is described in Section 2.2.

2.2 Chunking Strategies

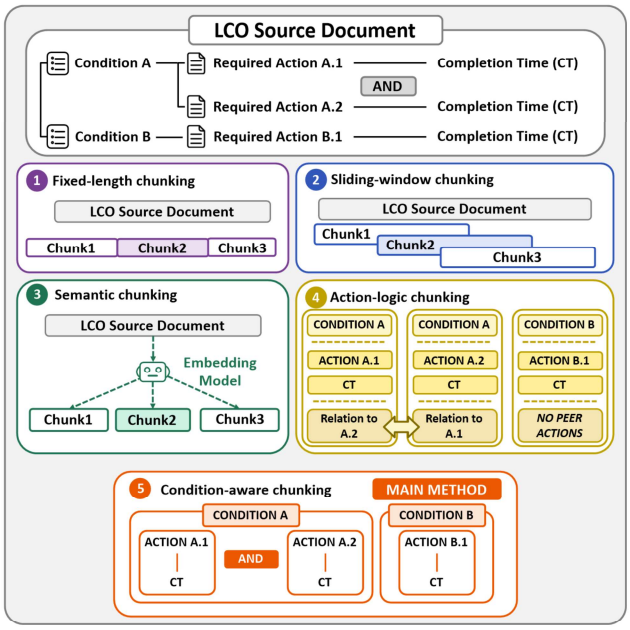

Fig. 2. Comparison of retrieval unit construction across the five chunking strategies.

Condition-aware chunking defines each *Condition* block in the ACTIONS table as a retrieval unit. The text of each *Condition* is combined with all associated *Required Action* and their corresponding *Completion Time*. The action order, grouping, and AND and OR relationships identified during text processing are retained in the linearized chunk. The LCO identifier, title, and applicability are also included as contextual information. This design is intended to keep the requirements associated with a *Condition* within the same retrieval unit.

To evaluate the effect of retrieval unit construction, five chunking strategies were compared using different criteria for defining chunk boundaries. Fixed-length, sliding-window, and semantic chunking were used as conventional approaches based on text length, overlap, or semantic similarity. Because a single *Condition* may contain multiple *Required Action*, action-logic chunking was also considered as a smaller retrieval unit defined at the individual *Required Action* level while retaining its relationship with the parent *Condition* and other actions.

Action-logic chunking defines each individual *Required Action* as a retrieval unit. Each chunk contains the *Required Action*, its parent *Condition*, and its *Completion Time*, together with the LCO identifier, title, and applicability. Separating an individual action from the rest of the *Condition* can remove information needed to interpret its relationship with other actions. This relationship is therefore expressed in a natural language sentence and added to the chunk. For both condition-aware and action-logic chunking, the LCO statement is retained as a separate retrieval unit because it does not correspond to a single *Condition* in the ACTIONS table.

Semantic chunking determines chunk boundaries from the embedding similarity between adjacent text segments. Among the eligible boundary candidates, the point with the lowest semantic similarity is selected. Chunk length is constrained to between 450 and 1200 characters, with a target length of 900 characters.

Sliding-window chunking divides the source text into 900-character windows with a 200-character overlap. Fixed-length chunking divides the same text into sequential 900-character segments with no overlap.

3. Evaluation

3.1 Dataset

The experiments used NUREG-1431 Rev. 5, Vol. 1[8] as the source document. From its Reactor Coolant System chapter, LCO Sections 3.4.1 to 3.4.20 were used to construct the retrieval corpus. Evaluation questions were generated by an LLM using selected LCO sections and then manually reviewed against the source document. The questions were designed to reflect the information needed to interpret and apply LCO requirements, including applicable *Condition*, *Required Action*, *Completion Time*, and the relationships among them. The final evaluation set contained 51 questions derived from LCO 3.4.5, 3.4.6, 3.4.7, and 3.4.12. The review verified the correctness of the questions, reference answers, and supporting evidence.

3.2 Retrieval Evaluation

The retrieval evaluation was designed to compare how effectively the five chunking strategies retrieve the evidence required to answer the same set of questions. All strategies were evaluated using the same 51 questions, the BAAI/bge-small-en-v1.5 [7] embedding model, and dense retrieval. Because the strategies produce retrieval units of different lengths, retrieval was evalu-

ated both by rank and under equal cumulative context token budgets.

Retrieval relevance was determined using the evidence keyword groups assigned to each question. Each group represents one evidence unit and was considered matched only when all keywords in the group appeared in a retrieved chunk. This criterion allows the evaluation to distinguish between retrieving at least one relevant evidence unit and retrieving all evidence required for a question.

Dense retrieval returned up to five ranked chunks for each query. The retrieval run used for the rank-based evaluation was subject to a cumulative context limit of 1024 tokens. Retrieval was also evaluated under cumulative context token budgets of 256 and 512 tokens by applying the corresponding limits to the ranked results. For the effectively unbounded setting, retrieval was performed with the token limit removed while retaining the same maximum retrieval depth of five. This allowed all five ranked chunks to be included even when their cumulative length exceeded 1024 tokens.

Retrieval performance was measured using MRR, Hit@k, Recall@k, and SetRecall@k for k = 1, 3, and 5. MRR and Hit@k characterize the ranking of relevant evidence, whereas Recall@k and SetRecall@k characterize evidence coverage. Hit@k indicates whether at least one evidence unit appears within the top k retrieved chunks. MRR, Recall@k, and SetRecall@k were calculated as follows. MRR measures the rank of the first retrieved chunk that matches at least one evidence unit:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N} \text{RR}_i \tag{1}$$

where $\text{RR}_i = 1/r_i$, and r_i is the rank of the first matched chunk for question i. If no evidence unit is matched, $\text{RR}_i = 0$. Recall@k measures the proportion of required evidence units covered within the top k retrieved chunks:

$$\text{Recall@}k = \frac{1}{\text{N}}\sum_{i=1}^{N} \frac{C_i(k)}{E_i} \tag{2}$$

SetRecall@k measures the proportion of questions for which all required evidence units are covered:

$$\text{SetRecall@}k = \frac{1}{N}\sum_{i=1}^{N} 1[C_i(k) = E_i] \tag{3}$$

Here, N is the number of evaluation questions, E_i is the number of required evidence units for question i. $C_i(k)$ is the number of those units covered within the top k retrieved chunks.

3.3 Downstream Evaluation

To examine how differences in retrieval translated into downstream answer accuracy, answers were generated for each chunking strategy using the same context token budgets as in the retrieval evaluation. The retrieved chunks that fit within each budget were provided to the same generator using identical prompts and generation settings. An LLM-only baseline without retrieved context was also included for comparison. All answers were generated using gemma4:12b [9].

The generated answers were evaluated using Claude Haiku as an LLM judge [10]. The judge received the question, the manually verified reference answer, and the generated answer, and assigned a binary verdict of correct or incorrect. The evaluation considered whether the answer identified the applicable *Condition*, included the *Required Action* and corresponding *Completion Time* when relevant, correctly interpreted AND and OR relationships, and avoided incorrect or contradictory claims. Answers that omitted required information or provided only a partial response were judged incorrect. Because the retrieved context was not provided to the judge, the evaluation assessed answer correctness only and did not assess whether the answer was supported by the retrieved context. Answer accuracy was then calculated for each chunking strategy.

4. Results

Condition-aware chunking showed the highest overall evidence coverage in the retrieval evaluation. The differences among the strategies were largest at smaller token budgets and narrowed as more context became available. In the downstream evaluation, condition-aware chunking also achieved the highest answer accuracy at every evaluated context budget.

4.1 Retrieval Results

Table I. Retrieval ranking performance

Strategy	MRR	Hit@1	Hit@3	Hit@5
Fixed-length	.506	.333	.686	.804
Sliding-window	.526	.373	.667	.765
Semantic	.571	.392	**.745**	**.824**
Action-logic	.599	**.471**	.725	.784
Condition-aware	**.605**	**.471**	**.745**	**.824**

As shown in Table I, condition-aware chunking recorded the highest MRR of 0.605 and tied with action-logic chunking for the highest Hit@1. It also matched semantic chunking at Hit@3 and Hit@5. Semantic chunking reached the same Hit@3 and Hit@5 despite its lower MRR, suggesting that relevant evidence tended to appear later in its ranked results. These

results show that condition-aware chunking tended to rank relevant evidence near the top while sustaining high Hit rates as retrieval depth increased.

Table II. Recall and Set Recall at different retrieval depths

Strategy	Recall / SetRecall		
	@1	@3	@5
Fixed-length	.310/.294	.602/.529	.720/.647
Sliding-window	.294/.235	.578/.510	.682/.628
Semantic	.333/.255	.637/.490	.700/.549
Action-logic	.396/.314	.649/.529	.711/.608
Condition-aware	**.400/.353**	**.679/.608**	**.761/.667**

Table II compares Recall and SetRecall at retrieval depths of 1, 3, and 5. Condition-aware chunking showed the highest values for both metrics at all three retrieval depths. This indicates that its advantage in evidence coverage held across retrieval depths, across different values of k.

Table III. Recall and SetRecall under different token budgets.

Strategy	Recall / SetRecall			
	@256	@512	@1024	Unbounded
Fixed-length	.359/.333	.626/.549	.720/.647	.729/**.667**
Sliding-window	.306/.235	.564/.490	.682/.627	.682/.627
Semantic	.258/.176	.592/.451	.700/.549	.708/.549
Action-logic	.492/.392	.672/.569	.711/.608	.711/.608
Condition-aware	**.498/.431**	**.742/.647**	**.761/.667**	**.761/.667**

Table III compares evidence coverage under equal context token budgets. The differences among the strategies were largest at the smaller budgets, particularly at 256 tokens, and became smaller as more context was allowed. Condition-aware chunking had the highest Recall at every evaluated budget and the highest SetRecall up to 1024 tokens. Under the unbounded setting, its SetRecall was tied with fixed-length chunking at 0.667. These results indicate that the evidence coverage advantage of condition-aware chunking persisted under constrained context budgets, suggesting that the advantage was not simply due to receiving more context.

Table IV. Average retrieved context tokens under different token budgets.

Strategy	256	512	1024	Unbounded
Fixed-length	186.5	434.0	756.2	802.3
Sliding-window	172.5	441.5	790.5	803.3
Semantic	136.6	396.3	835.9	945.1
Action-logic	180.7	422.1	582.4	582.4
Condition-aware	179.9	420.4	579.9	579.9

Table IV shows the average amount of retrieved context under each token budget. At the 1024 token and unbounded settings, condition-aware chunking used approximately 580 tokens on average, substantially less than fixed-length, sliding-window, and semantic chunking strategies. This further shows that its higher evidence coverage in Table III was not associated with receiving more retrieved context.

Overall, condition-aware chunking showed the highest retrieval performance across the evaluated metrics, while action-logic chunking generally ranked next. The performance of action-logic suggests that preserving the relationships among LCO requirements is beneficial, whereas the further advantage of condition-aware chunking indicates that keeping related requirements within the same *Condition* may be more effective.

4.2 Downstream Results

Table V. Downstream answer accuracy under different context token budgets. (%)

Strategy	256	512	1024	Unbounded
Fixed-length	17.6	35.3	37.3	37.3
Sliding-window	13.7	29.4	39.2	41.2
Semantic	19.6	33.3	39.2	41.2
Action-logic	29.4	49.0	52.9	52.9
Condition-aware	**35.3**	**54.9**	**58.8**	**58.8**
LLM-only (no context)	0.0			

Table V shows the downstream answer accuracy for each chunking strategy. Condition-aware chunking had the highest accuracy at every evaluated context budget, with action-logic chunking ranking second. The LLM-only baseline produced no correct answers across the 51 evaluation questions. The relationship between these accuracy results and the actual amount of retrieved context is examined in Figure 3.

Increasing the context budget from 256 to 512 tokens improved answer accuracy for all five chunking strategies. Accuracy also increased from 512 to 1024 tokens, although the increase was smaller for most strategies. Moving from the 1024 token to the unbounded condition produced little or no further improvement.

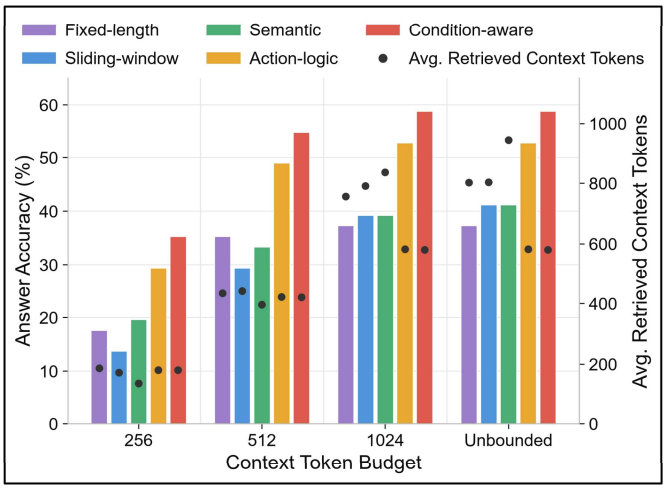

Fig. 3. Retrieved context size and downstream answer accuracy under different token budgets

Figure 3 compares the downstream accuracy reported in Table V with the average amount of retrieved context shown in Table IV. Purple, blue, green, orange, and red bars represent fixed-length, sliding-window, semantic, action-logic, and condition-aware chunking, respectively, while the black markers indicate the average number of retrieved context tokens.

At the smaller budgets, the amount of retrieved context was relatively similar across the strategies, while answer accuracy varied considerably. At the larger budgets, the differences in retrieved context became more pronounced, but strategies retrieving more context did not necessarily show higher answer accuracy. Thus, although increasing the context budget generally improved accuracy within each strategy, differences in downstream accuracy among the strategies were not explained simply by the amount of retrieved context.

Condition-aware chunking consistently showed the highest evidence coverage and downstream accuracy, suggesting that keeping the requirements associated with a *Condition* within the same retrieval unit is effective for LCO retrieval and downstream evaluation. Action-logic chunking, which preserves these relationships while separating individual *Required Action* into different retrieval units, also performed well but remained below condition-aware chunking. This comparison suggests that preserving the relationships among LCO requirements is beneficial, while keeping all requirements associated with the same *Condition* within a single retrieval unit may provide an additional advantage.

5. Conclusion

In this study, we evaluated how the construction of retrieval units affects retrieval and downstream QA performance for LCOs. We proposed condition-aware chunking, which groups each *Condition* with its associated *Required Action* and *Completion Time*, and compared it with fixed-length, sliding-window, semantic, and action-logic chunking. Condition-aware chunking achieved the highest overall evidence coverage in the retrieval evaluation and the highest downstream answer accuracy across all evaluated context budgets. These results suggest that preserving the relationships among *Condition, Required Action*, and *Completion Time* is important when constructing retrieval units for LCOs. They also suggest that the *Condition* can serve as an effective unit for grouping related requirements.

These findings should, however, be interpreted within the scope of the present evaluation, which was based on 51 questions drawn from four selected LCOs and used a single embedding model, generator, and judge model. A broader evaluation across LCO requirements, questions, and models would be needed to determine how well the observed results generalize.

REFERENCES

[1] Lee, S. B., and J. H. Kim. "A Computerized Operator Support System to Monitor the Technical Specifications." European Safety and Reliablility Conference (ESREL), Hannover, Germany. 2019.
[2] Seong, No Kyu, et al. "Retrieval methodology for similar NPP LCO cases based on domain specific NLP." Nuclear Engineering and Technology 55.2 (2023): 421-431.
[3] Ndum, Zavier Ndum, et al. "RADIANT-LLM: an Agentic Retrieval Augmented Generation Framework for Reliable Decision Support in Safety-Critical Nuclear Engineering." arXiv preprint arXiv:2604.22755 (2026).
[4] Shi, Freda, et al. "Large language models can be easily distracted by irrelevant context." International conference on machine learning. PMLR, 2023.
[5] Duarte, André V., et al. "Lumberchunker: Long-form narrative document segmentation." Findings of the Association for Computational Linguistics: EMNLP 2024. 2024.
[6] Aushev, Islam, et al. "RAGulator: Effective RAG for regulatory question answering." Proceedings of the 1st Regulatory NLP Workshop (RegNLP 2025). 2025.
[7] Beijing Academy of Artificial Intelligence (BAAI), "BAAI/bge-small-en-v1.5," Hugging Face Model Hub. [Online]. Available: https://huggingface.co/BAAI/bge-small-en-v1.5
[8] U.S. Nuclear Regulatory Commission, Standard Technical Specifications – Westinghouse Plants: Specifications, NUREG-1431, Revision 5, Volume 1, September 2021.
[9] Team, Gemma, et al. "Gemma 4 technical report." arXiv preprint arXiv:2607.02770 (2026).
[10] Anthropic, Claude Haiku 4.5 System Card, October 2025.