UDC 004.8:004.912:004.65 DOI https://doi.org/10.35546/kntu2078-4481.2026.2.49

M. O. FANT
Candidate of Philological Sciences,
Associate Professor at the Department of Software Engineering
Zhytomyr Polytechnic State University
ORCID: 0000-0002-4994-8009

T. A. VAKALIUK
Doctor of Pedagogical Sciences, Professor,
Head of the Department of Software Engineering
Zhytomyr Polytechnic State University
ORCID: 0000-0001-6825-4697

MAX-MIN SEMANTIC CHUNKING FOR TECHNICAL DOCUMENTATION RETRIEVAL: A CASE STUDY WITH HAZELCAST DOCUMENTATION

Effective chunking of source documents is a critical factor in determining retrieval quality in retrieval-augmented generation (RAG) systems. Traditional fixed-size and sentence-based chunking strategies process documents without semantic awareness, frequently breaking cohesive information units at arbitrary boundaries. Max-Min Semantic Chunking uses an embedding-first approach: all sentences are embedded before segmentation, and chunk boundaries are determined by comparing the similarity between a candidate sentence and the existing chunk against a predefined minimum similarity threshold. This paper presents a theoretical case study analysing the suitability of Max-Min Semantic Chunking for large-scale technical documentation, using the Hazelcast documentation as the target corpus.

The analysis identifies and characterises seven content types in the Hazelcast documentation, such as narrative text, API reference entries, code blocks, configuration tables, step-by-step instructions, admonition blocks, and tabbed content panels. For each content type, the expected behaviour of Max-Min chunking is evaluated relative to fixed-size and sentence-based baselines across four dimensions: intra-chunk semantic coherence, retrieval precision, chunk size distribution and variance, and boundary detection quality at content-type transitions. A brief illustrative walkthrough demonstrates the algorithm's operation on a representative Hazelcast page.

The analysis reveals a nuanced picture. Max-Min is predicted to substantially outperform baselines on narrative, step-by-step, and inline admonition content, where embedding-space similarity reliably tracks logical structure. However, it faces structural limitations on code blocks, configuration tables, standalone admonitions, and – most severely – tabbed multi-language content panels, where near-identical embeddings across panels prevent boundary detection. Four concrete adaptation strategies are proposed. These strategies are content-type-aware threshold adjustment, hybrid pre-segmentation of structured elements with per-panel tab extraction, admonition-aware boundary handling, and context prefix injection.

Key words: *Max-Min semantic chunking, retrieval-augmented generation, technical documentation, Hazelcast, vector embeddings.*

М. О. ФАНТ
кандидат філологічних наук,
доцент кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»
ORCID: 0000-0002-4994-8009

Т. А. ВАКАЛЮК
доктор педагогічних наук, професор,
завідувач кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»
ORCID: 0000-0001-6825-4697

МЕТОД MAX-MIN СЕМАНТИЧНОЇ СЕГМЕНТАЦІЇ ТЕКСТУ ДЛЯ ІНФОРМАЦІЙНОГО ПОШУКУ В ТЕХНІЧНІЙ ДОКУМЕНТАЦІЇ: ДОСЛІДЖЕННЯ НА ПРИКЛАДІ ДОКУМЕНТАЦІЇ HAZELCAST

Ефективне розбиття вихідних документів на фрагменти є критичним чинником якості пошуку в системах retrieval-augmented generation (RAG). Традиційні стратегії фрагментації з фіксованим розміром або на основі речень опрацьовують документи без урахування семантики, часто розриваючи цілісні інформаційні одиниці на

 ISSN 2078-4481

довільних межах. Підхід семантичної сегментації тексту Max-Min пропонує модель «embedding-first», за якої всі речення спочатку перетворюються на векторні подання, а межі фрагментів визначаються шляхом порівняння подібності між кандидатом-реченням і поточним фрагментом із наперед заданим мінімальним порогом подібності. У статті подано теоретичне дослідження придатності методу семантичної сегментації Max-Min для великомасштабної технічної документації на прикладі корпусу документації Hazelcast.

Аналіз ідентифікує та характеризує сім типів контенту в документації Hazelcast – наративний текст, описи API, блоки коду, таблиці конфігурації, покрокові інструкції, блоки застережень та панелі вкладок з мультимовними прикладами. Для кожного типу контенту оцінюється очікувана поведінка методу Max-Min порівняно з базовими методами за чотирма вимірами, такими як семантична зв'язність фрагментів, точність пошуку, розподіл і варіативність розмірів фрагментів та якість виявлення меж на переходах між типами контенту. Аналіз показує, що метод Max-Min суттєво перевершить базові показники для наративного контенту, покрокових інструкцій та вбудованих блоків застережень. Водночас він стикається зі структурними обмеженнями щодо блоків коду, конфігураційних таблиць, автономних блоків застережень і – найсуттєвіше – панелей вкладок із мультимовним контентом, де майже ідентичні вектори між панелями унеможливлюють виявлення меж. Запропоновано чотири стратегії адаптації як напрями подальших емпіричних досліджень.

Ключові слова: *семантична сегментація Max-Min, генерація з доповненим пошуком, технічна документація, Hazelcast, векторні ембедінги.*

Problem statement

Retrieval-augmented generation (RAG) has become the dominant architecture for grounding large language model (LLM) responses in authoritative, up-to-date document collections [1]. The quality of a RAG system depends critically on how source documents are segmented before indexing. The chunking strategy determines what units of information are embedded, stored, and retrieved in response to user queries [2]. A poorly chosen chunking strategy causes two classes of failure. Over-splitting breaks cohesive information units, returning fragments that lack sufficient context to answer a query. Under-splitting returns chunks too large to match queries precisely, polluting retrieval with irrelevant content [3].

Technical documentation presents a particularly demanding chunking context. Unlike general-purpose corpora – news articles, encyclopaedia entries, or academic papers – technical documentation is semantically heterogeneous within individual pages. A single Hazelcast documentation page may contain a conceptual narrative introduction, one or more code examples, a configuration parameter table, and a sequence of procedural steps, each of which constitutes a distinct information type with different semantic density and embedding space behaviour. Standard chunking strategies treat this heterogeneous content uniformly, applying the same splitting logic regardless of content type, and consequently produce chunks of inconsistent quality across the corpus.

The Hazelcast distributed computing platform maintains a documentation portal spanning over ten active product versions for several products and thousands of pages [4]. This corpus provides an ideal test case for evaluating chunking strategies. It is large, multi-versioned, domain-specific, and structurally heterogeneous. The Max-Min Semantic Chunking algorithm proposed by Kiss et al. [5] is a theoretically well-motivated approach that embeds sentences before chunking and uses embedding-space similarities to determine boundaries. However, its behaviour on technical documentation has not been studied. This paper addresses that gap through a systematic theoretical analysis.

Analysis of recent research and publications

Fixed-size chunking – splitting documents into segments of a predetermined token length with optional overlap – remains the most widely deployed strategy due to its simplicity and predictability [2]. Its fundamental limitation is the complete disregard for semantic boundaries. Related sentences are separated, and unrelated sentences are grouped solely based on position within the token count. Recursive character splitting improves upon this by using structural cues such as paragraph breaks and heading markers. However, it remains susceptible to inconsistent document structure and does not guarantee semantic coherence within chunks [6].

Semantic chunking approaches address this by incorporating meaning into the boundary decision. The broad family of semantic chunking methods identifies boundaries where the semantic content of the text shifts significantly, typically indicated by a spike in the cosine distance between adjacent sentence embeddings [3]. Qu et al. [7] conducted a systematic evaluation of semantic chunking across multiple retrieval tasks and found that generic semantic chunking does not consistently outperform fixed-size chunking in terms of retrieval accuracy, while incurring substantially higher computational cost. Their finding motivates the search for more principled semantic chunking algorithms tailored to specific corpus characteristics.

Kiss, Nagy, and Szilágyi [5] introduced Max-Min Semantic Chunking as a constrained sequential clustering approach. The algorithm embeds all sentences in a document prior to segmentation. It makes boundary decisions by comparing two quantities at each step. It measures the maximum cosine similarity between the candidate sentence and any sentence already in the current chunk, and the minimum pairwise cosine similarity among all sentences currently in the chunk. If the candidate sentence is at least as similar to the chunk as the chunk is the least coherent internal pair, it is absorbed; otherwise, a new chunk begins. This formulation provides a principled coherence guarantee that no chunk boundary is

placed unless adding the following sentence would reduce internal coherence below the chunk's existing minimum. Kiss et al. validated this approach on general text corpora and reported superior clustering efficiency (adjusted mutual information) compared to fixed-size baselines. The Milvus vector database platform has highlighted the algorithm as a notable advancement in embedding-first RAG pipelines [8]. However, no study has examined its behaviour on technical documentation corpora, which differ from the general corpora used in the original evaluation in ways that may significantly affect algorithm performance.

Research objectives

The objective of this paper is to analyse the theoretical suitability and predicted empirical behaviour of Max-Min Semantic Chunking when applied to large-scale technical documentation, using the Hazelcast documentation portal as a concrete corpus. Specific tasks are: (1) characterise the Hazelcast documentation corpus in terms of content-type distribution and its implications for chunking; (2) analyse the expected behaviour of the Max-Min algorithm on each content type relative to fixed-size and sentence-based baselines; (3) illustrate algorithm operation with a worked example drawn from a real Hazelcast documentation page; (4) assess predicted performance across four evaluation dimensions; and (5) identify limitations and propose adaptation strategies as directions for future experimental work.

Main research findings

Corpus characterisation. The Hazelcast documentation portal is built with Antora and AsciiDoc, rendered to static HTML, and organised into version-specific URL namespaces. Analysis of the rendered HTML structure reveals seven primary content types that appear throughout the corpus in varying proportions.

Narrative text – conceptual explanations, introductions, and background sections – constitutes the largest share of the corpus and is characterised by long, grammatically complete sentences with high intra-paragraph semantic coherence. *API reference entries* are short, densely informative, and structurally formulaic. An entry consists of a method signature, a parameter list, a return value, and a brief description. *Code blocks* are syntactically distinct from surrounding prose and embed into a substantially different region of the embedding space than natural language text. *Configuration tables* present parameter names, types, default values, and descriptions in tabular form; rows share a standard structure but vary in semantic content. *Step-by-step instruction sequences* are procedural, with moderate semantic similarity between adjacent steps and apparent topical coherence within a procedure.

Admonition blocks are a structurally distinct element type in Antora-generated documentation, rendered with semantic type labels. The labels are NOTE, TIP, WARNING, CAUTION, and IMPORTANT. Their text is typically one to three sentences and serves a specific rhetorical function: a WARNING flags a known failure mode, a TIP suggests an optimisation, and a NOTE clarifies a common misunderstanding. Admonitions appear both inline within narrative sections (where they modify the surrounding content) and as standalone blocks at the section level (where they are independent information units). This dual character is consequential for chunking. An inline admonition should arguably remain with its surrounding paragraph, while a standalone IMPORTANT block warrants its own chunk. Antora encodes the admonition type as a CSS class, making it detectable without natural language analysis.

Tabbed content blocks represent the most structurally complex element in the Hazelcast documentation. Tabs are used extensively to present the same configuration or procedure in multiple languages (Java, XML, YAML) or across deployment environments (Docker, CLI, Binary). In the rendered HTML, each tab panel is a sibling div sharing a common heading and introductory sentence. This creates a distinct challenge for any chunking algorithm. The tab panels for Java and XML configuration of the same feature are semantically near-identical, sharing vocabulary and parameter names. However, they are technically distinct documents that should be indexed separately to serve language-specific queries. Max-Min will encounter these as a sequence of structurally similar, semantically redundant text blocks with no natural discontinuity in their similarity, meaning it will either merge all tabs into a single large chunk (problematic for context window usage) or split them arbitrarily.

These content types differ in three dimensions that directly affect Max-Min algorithm behavior, such as sentence length distribution (short in API entries and table rows, long in narrative), intra-type semantic similarity (high in narrative and tab panels, moderate in steps, variable in code), and the sharpness of semantic transitions at type boundaries (abrupt between prose and code, gradual within narrative, absent between tab panels). These differences motivate a content-type-aware analysis rather than a single corpus-level assessment.

To ground the analysis, consider a representative page from the Hazelcast Platform 5.6 documentation on Map configuration. The page begins with three narrative sentences explaining what a Hazelcast Map is and its primary use cases. These are followed by a sentence introducing the configuration section, then a code block showing an XML configuration snippet, then a table of configuration parameters (see figure 1). In natural language, this page contains approximately twelve prose sentences and two code blocks.

Strengths of Max-Min on technical documentation. For narrative text and step-by-step instruction content – which together constitute the majority of a typical Hazelcast documentation page – Max-Min chunking is predicted to outperform both baselines substantially. The algorithm's coherence guarantee ensures that narrative paragraphs on a single topic are grouped rather than split mid-explanation, a primary failure mode of fixed-size chunking. In step-by-step procedures,

the moderate semantic similarity between adjacent steps (they share vocabulary and refer to the same process) keeps procedurally related steps together. In contrast, a shift in action or subject between independent procedures signals a boundary. Both are cases where embedding-space similarity reliably tracks the logical structure of the content.

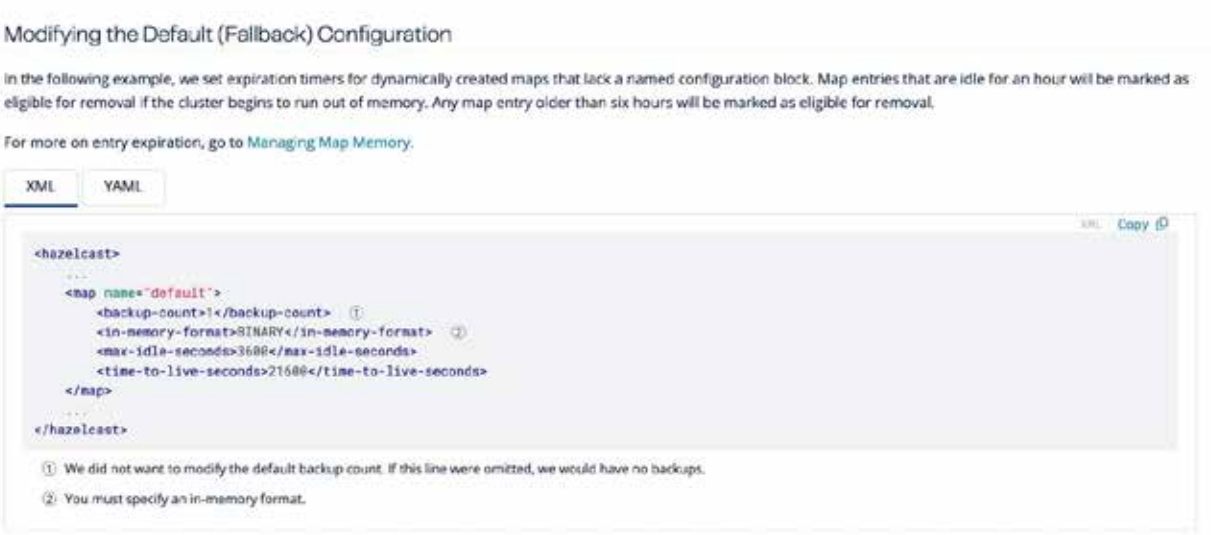

Fig. 1. Representative page from Hazelcast documentation

For inline admonitions – NOTE or TIP blocks appearing mid-section – Max-Min is also predicted to behave well. Because the admonition text is topically related to the surrounding narrative (a NOTE clarifying the preceding statement will share vocabulary with it), its embedding will score above the chunk minimum similarity threshold and be absorbed into the surrounding narrative chunk. This is the desired outcome, because the admonition and its surrounding context form a coherent information unit for retrieval. A user querying about the behaviour described in the surrounding paragraph should also see the associated NOTE.

Chunk size variance – often cited as a disadvantage of variable-length chunking – is in fact a feature rather than a bug in the technical documentation context. Fixed-size chunking enforces uniform chunk sizes that are inevitably a poor fit for the distribution of actual information units. A 512-token chunk is too large for a short API entry and too small for a detailed configuration explanation. Max-Min produces chunks that are as large as needed to contain a coherent topic and no larger, which aligns naturally with the variable-length distribution of real information units in technical documentation.

Weaknesses and failure modes. The algorithm's most significant weakness on this corpus concerns code blocks and configuration tables. Code blocks generate embeddings in a region of the vector space that is distant from natural language prose – not because the code is semantically unrelated to the surrounding text, but because code and prose use fundamentally different token distributions and syntactic structures. The result is that Max-Min correctly identifies the prose-to-code boundary (an apparent strength) but treats the code block as a self-contained semantic unit isolated from its explanatory context. A retrieved code chunk without its surrounding explanation is frequently insufficient to answer a user query, just as the explanation without the code is incomplete.

Configuration tables present a different failure mode. Table rows are structurally similar – each consists of a parameter name, type, default value, and short description – which means their embeddings cluster tightly regardless of whether the parameters are logically related. The minimum intra-chunk similarity will be high for a chunk of table rows, meaning new rows will be absorbed until a size limit is reached or the document ends. The result is large table chunks that merge logically unrelated parameters simply because their surface forms are similar.

Standalone admonitions – IMPORTANT or WARNING blocks that appear at section boundaries rather than inline – present a subtler failure mode. Such blocks are typically short (one or two sentences) and use vocabulary that may or may not overlap with the immediately surrounding text. If the WARNING appears after a code block, its embedding sits between the code-space region and the prose-space region, and its similarity to either the preceding code chunk or the following narrative chunk is unpredictable. The result is that standalone admonitions may be incorrectly merged with code blocks or become isolated single-sentence chunks, both of which degrade retrieval quality.

Tabbed content blocks represent the most severe failure mode in the Hazelcast corpus. Tabs presenting the same configuration in Java, XML, and YAML generate embeddings that are near-identical in the semantic space because the underlying concepts – the same parameters with the same values – are the same. The minimum intra-chunk similarity of a growing tab-panel chunk will therefore remain high as additional panels are absorbed, and no boundary will be triggered

between them. The practical result is a single, very large chunk containing three or more complete code examples that would, individually, exceed the optimal retrieval context size. When a user queries specifically for a Java or YAML example, this merged chunk either satisfies both queries (redundantly) or neither (if the context window forces truncation of the later panels). Critically, this failure is invisible to the algorithm since, from the embedding-space perspective, everything is working as intended.

Table 1 summarises the comparative performance of the three chunking strategies across the four evaluation dimensions and seven content types, based on the theoretical analysis. Where "F" means Fixed-size, "S" Sentence-based, "M" Max-Min semantic, "✓" advantage, "~" moderate, "✗" disadvantage, "–" not applicable.

Table 1

Predicted comparative performance of chunking strategies on Hazelcast documentation

Content type	Semantic coherence			Retrieval precision			Size variance			Boundary quality		
	F	S	M	F	S	M	F	S	M	F	S	M
Narrative	✗	~	✓	~	~	✓	–	~	✓	✗	~	✓
API ref.	~	✓	✓	✗	~	~	–	~	~	✗	~	~
Code blocks	~	✗	~	✗	✗	~	–	~	~	✗	✗	~
Config tables	✗	✗	✗	✗	✗	✗	–	~	✗	✗	✗	✗
Step-by-step	✗	~	✓	✗	~	✓	–	~	✓	✗	~	✓
Admonitions	~	~	~	✗	~	~	–	~	~	✗	~	~
Tabbed content	✗	✗	✗	✗	✗	✗	–	–	✗	✗	✗	✗

Proposed adaptation strategies. Four adaptation strategies are proposed to address the identified failure modes, as candidates for empirical evaluation in future work.

The first strategy is *content-type-aware threshold adjustment*. The minimum similarity threshold for deciding whether a candidate sentence joins a chunk should vary across content types. Narrative text benefits from a high threshold ensuring tight coherence, while step-by-step content benefits from a somewhat lower threshold that accommodates minor topic shifts within a procedure. This requires a lightweight content-type classifier as a pre-processing step, implemented deterministically using the HTML structure of Antora-generated pages, requiring no machine learning.

Second strategy is *hybrid pre-segmentation for structured elements*. According to this strategy, before applying Max-Min, identify code blocks, configuration tables, and tabbed content panels using CSS class detection (.listingblock, table,. tabbed-content,.tab-pane), and treat them as atomic, predefined chunks that bypass the Max-Min boundary decision entirely. For tabbed content specifically, each tab panel should be extracted as a separate predefined chunk, tagged with its language or platform metadata (Java, XML, YAML, Docker, CLI) derived from the tab label. This directly addresses the tab-panel merging failure mode. Max-Min then operates only on the prose segments between predefined chunks, where its coherence guarantee is most valuable.

The third strategy is *admonition-aware boundary handling*. Standalone admonitions appearing at section boundaries should be pre-classified by their position in the heading hierarchy. An admonition that is the sole content under a heading should be treated as an independent chunk with a context prefix derived from the heading and its parent. An admonition appearing between prose paragraphs under the same heading should be merged with the surrounding prose chunk regardless of similarity score, using a forced-merge rule.

The fourth strategy is *context prefix injection*. Regardless of which algorithm produces the final chunk, each chunk should be enriched with a short context prefix derived from its parent heading hierarchy before embedding generation. This partially addresses the isolated code chunk and standalone admonition problems by ensuring the chunk's embedding incorporates the topic context of the surrounding section, improving retrieval relevance for language-specific and concept-specific queries alike.

Implications for chunk size distribution. The predicted chunk-size variance for Max-Min on the Hazelcast corpus has practical implications for the vector index. A corpus of variable-length chunks requires more careful management of context window usage at retrieval time. A top-k retrieval that returns k chunks of average length 800 tokens consumes more context than the same k from a 256-token fixed-size corpus. The retrieval strategy should therefore use a token budget rather than a fixed k. This is a downstream architectural consideration that an experimental study should address alongside the chunking comparison itself.

Conclusions

This paper presented a case study analysing the application of Max-Min Semantic Chunking to large-scale technical documentation, using the Hazelcast documentation portal as the target corpus. The analysis yields a nuanced and practically functional assessment. Seven content types were identified and characterised, with lists explicitly argued as a sub-category of narrative and procedural content rather than a separate class – a methodological decision that itself clarifies the taxonomy. Max-Min is predicted to offer meaningful advantages over fixed-size and sentence-based baselines for

narrative text, step-by-step instructions, and inline admonitions, where embedding-space similarity reliably tracks logical structure, and the algorithm's coherence guarantee prevents arbitrary mid-topic splits.

However, the algorithm faces structural limitations specific to the properties of technical documentation rather than deficiencies in the algorithm itself. Code blocks and configuration tables exhibit embedding-space characteristics that the Max-Min decision rule cannot distinguish from those of high-coherence content. Tabbed multi-language panels – a pervasive pattern in the Hazelcast documentation – represent the most severe failure mode. Near-identical embeddings across panels mean no similarity discontinuity is detected, and all panels are merged into a single oversized chunk. Standalone admonitions introduce boundary ambiguity that the algorithm resolves inconsistently.

Four concrete adaptation strategies are proposed. Those strategies are content-type-aware threshold adjustment, hybrid pre-segmentation of structured elements with per-panel tab extraction, admonition-aware boundary handling, and context prefix injection. These strategies can be directly implemented using the deterministic HTML structure of Antora-generated pages and require no additional machine learning components. The Hazelcast documentation corpus – public, large, multi-version, and structurally well-characterised – provides an ideal environment for the planned experimental study that will empirically confirm or refine these theoretical predictions and report results across all four evaluation dimensions.

References

1. Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 9459–9474. https://dl.acm.org/doi/abs/10.5555/3495724.3496517

2. Gao, Y., Xiong, Y., Gao, X., et al. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

https://doi.org/10.48550/arXiv.2312.10997

3. Qu, R., Tu, R., & Bao, F. S. (2025). Is Semantic Chunking Worth the Computational Cost? Findings of the Association for Computational Linguistics: NAACL 2025, 2155–2177. https://doi.org/10.18653/v1/2025.findings-naacl.114

4. Hazelcast. (2026). Hazelcast Documentation. Retrieved from https://docs.hazelcast.com

5. Kiss, C., Nagy, M., & Szilágyi, P. (2025). Max-Min semantic chunking of documents for RAG application. Discover Computing, 28, 117. https://doi.org/10.1007/s10791-025-09638-7

6. Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., & Yih, W. (2023). REPLUG: Retrieval-Augmented Black-Box Language Models. North American Chapter of the Association for Computational Linguistics. https://doi.org/10.48550/arXiv.2301.12652

7. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 3982–3992. https://doi.org/10.18653/v1/D19-1410

8. Milvus. (2025). Embedding First, Then Chunking: Smarter RAG Retrieval with Max-Min Semantic Chunking. Retrieved from https://milvus.io/blog/embedding-first-chunking-second-smarter-rag-retrieval-with-max-min-semantic-chunking.md

Дата першого надходження статті до видання: 13.02.2026
Дата прийняття статті до друку після рецензування: 20.03.2026
Дата публікації (оприлюднення) статті: 07.05.2026