RAG Pipeline Strategies for Ukrainian Multi-Domain Document Understanding Task

Mykola Nosenko[1] **and Pavlo Kilko**[2]
Taras Shevchenko National University of Kyiv
Kyiv, Ukraine
[1]nikolay.nosenko@knu.ua
[2]pavel.kilko@knu.ua

Abstract

In this work, we present top-performing solution to the UNLP 2026 Shared Task on Ukrainian Multi-Domain Document Understanding. This task focuses on answering multiple-choice questions grounded in domain-specific Ukrainian documents, while also requiring systems to identify the source document and page. We developed a modular retrieval-augmented generation (RAG) pipeline and conducted a series of ablation experiments over its individual components to identify the best-performing strategy at each stage. Based on our evaluation results, we propose two final pipeline configurations that differ in their computational cost and retrieval accuracy: a stronger but more compute-intensive document-level augmentation approach and a lighter summary-based augmentation that is suitable for constrained environments. Our submission achieved 3rd place on the private leaderboard. This demonstrates that isolated curation of RAG components can yield strong performance for Ukrainian document grounded question answering without additional language model adaptations.

1 Introduction

Large language models (LLMs) are increasingly used in applied systems, but factual errors and hallucinations still limit their reliable deployment in domains sensitive to information quality and trustworthiness (Ji et al., 2023).

This is especially evident in tasks that require external context rather than relying only on the model's internal knowledge. Such settings fall within document understanding and document question answering, where system quality depends not only on answer correctness, but also on identifying and using relevant supporting evidence.

One of the most common approaches to address this problem is retrieval-augmented generation (RAG), which combines answer generation with the retrieval of relevant information from external sources. Subsequent work has shown that the effectiveness of RAG depends not only on the base model, but also on the configuration of the entire pipeline, including retrieval, context processing, and answer generation. In many applied scenarios, what matters most is not a new architecture, but the careful selection and coordination of existing components.

These challenges are particularly relevant for mid-resource languages. Research on multilingual RAG shows that transferring English-centric solutions is non-trivial and depends on both language-specific and retrieval-related factors (Wu et al., 2024). For Ukrainian, this is further complicated by limited adapted resources, models, and evaluation practices, despite recent progress in Ukrainian-language LLMs and benchmarks (Paniv, 2025).

The UNLP 2026 Shared Task on Ukrainian Multi-Domain Document Understanding[1] requires systems to answer multiple-choice questions grounded in domain-specific Ukrainian PDF documents while also identifying the source document and page. In this work, we describe the pipeline we developed for this task, focusing not on a new architecture or additional model adaptation, but on retrieval pipeline engineering through the selection and coordination of its components. The resulting system placed third on the private competition leaderboard.

2 Related Work

Retrieval-augmented generation has become a practical framework for building systems (Lewis et al., 2021) in which answers are generated from external context rather than solely from the model's parametric knowledge. In document-grounded question answering, this shifts the focus from generation alone to the design of the full pipeline, including

[1]https://unlp.org.ua/shared-task/

Proceedings of the Fifth Ukrainian Natural Language Processing Conference (UNLP 2026), pages 240–248

document representation, retrieval, reranking, and context construction.

One line of techreport (Smith and Troynikov, 2024) studies how documents should be represented for indexing and retrieval, including chunking strategies, fragment size, overlap, and ways of preserving the link between local fragments and broader document context. These choices directly affect both retrieval quality and the accuracy of linking answers to specific source segments.

Another important line of work focuses on improving retrieval at the inference stage. This includes dense retrieval, hybrid search, query expansion, hypothetical document generation, and various reranking approaches applied after the initial retrieval step. Reranking models play a particularly important role in modern pipelines, especially cross-encoder approaches and LLM-based schemes, which allow for more precise estimation of the match between a query and a candidate. At the same time, the effectiveness of such methods depends on the specific task setting: techniques that improve performance in open-domain scenarios with noisy user queries do not necessarily yield the same effect in controlled benchmark-oriented settings.

For document understanding tasks, performance depends not only on retrieving relevant text, but also on how context is constructed for the final model (Reuter et al., 2025). Pipeline designs differ in how they combine retrieved fragments, apply contextual enrichment, and preserve the link between retrieval and final answer selection. As a result, system quality is determined by the coordination of the pipeline as a whole.

3 System Architecture

Analyzing existing RAG techniques (Gao et al., 2024), we structure our system as a two-stage pipeline, as shown in Figure 1. Our system covers the two main phases of RAG: the indexing pipeline and the question-answering (QA) pipeline. We adopt a modular architecture in which each pipeline component can be replaced independently. This design directly supports our ablation study (§4): by isolating components, we can measure their individual contributions to overall RAG performance while keeping the rest of the pipeline fixed.

The indexing pipeline consists of four main steps. **Document Loading** converts documents into textual representations, since raw artifacts cannot be embedded directly. **Document Splitting** divides each document into chunks for embedding. **Chunk Augmentation** enriches each chunk with broader contextual information from the source document. Finally, **Embedding and Indexing** encodes chunks as dense vectors and stores them in a vector database using an HNSW index with cosine similarity.

The question-answering pipeline consists of four main steps. **Retrieval** selects relevant chunks from the vector store based on the input question. **Reranking** refines the initial results by reordering the retrieved chunks using a more expressive scoring model. **Context Assembly** combines the highest-ranked chunks into a unified context. Finally, **Answer Generation** prompts the LLM with the assembled context and the input question to produce an answer in the required format.

4 Experiments

4.1 Experimental Setup

Dataset. The development dataset for this task, provided by the organizers, consisted of two parts: a document subset and a question-answer subset. The document subset contained 41 files in PDF format. The question-answer subset included 460 questions from the sports and medicine domains. Each question has six answer choices, one correct answer, a source document ID, and a source page number. Additionally, for embedding model evaluation, we constructed our own UNLP-QA dataset[2]. We aligned the organizers' dataset with the RTEB format (Liu et al., 2025) by manually annotating each question with the corresponding gold context and treating the correct page of the source document as the target retrieved chunk. For local RAG pipeline evaluation, we also prepared a separate subset in TXT format of the source documents by converting the original PDFs with Gemini-2.5-Pro (Comanici et al., 2025).

Language Model. The shared task encouraged participants to use LLMs specifically adapted for the Ukrainian language. Following this recommendation, we chose MamayLM (Yukhymenko et al., 2025), a Ukrainian-adapted fine-tune of Gemma 3 12B (Team et al., 2025). It achieves the highest score on the Ukrainian Language Model Leaderboard (Paniv, 2025) and performs well in our inter-

[2] https://github.com/Dialogus-ex-Machina/unlp-2026-qa-rag-pipeline/blob/main/data/dev_questions_with_context.csv

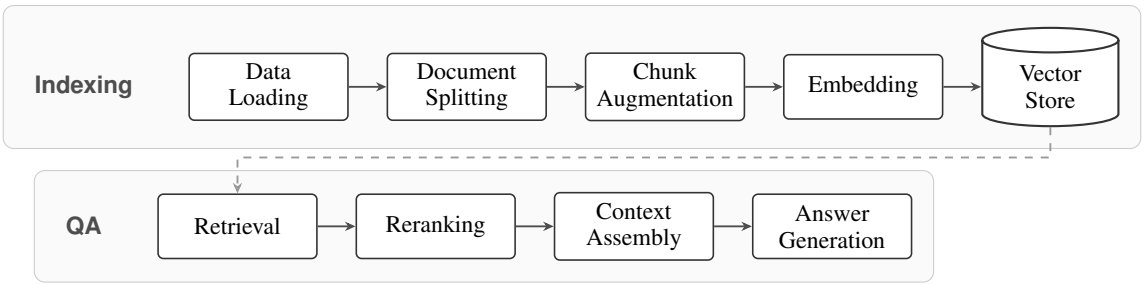

Figure 1: Two-phase RAG pipeline architecture. The dashed arrow indicates the vector store connecting the two phases.

nal evaluations, particularly in question-answering settings when supplied with relevant context. In this work, we do not focus on improving the language model itself through additional fine-tuning or parameter-efficient adaptation methods such as LoRA (Hu et al., 2021). Instead, we keep the LLM fixed and concentrate on improving retrieval and context construction, based on the assumption that modern instruction-tuned models are already sufficiently capable of producing correct answers when provided with well-retrieved and well-structured supporting context.

Metrics. The organizers scoring metric combines answer correctness (50%), document source identification (25%), and page proximity (25%). Since our ablation focuses on retrieval, we isolate the latter two components into separate metrics.

Document source accuracy ($Accuracy_{doc}$) measures correct document identification:

$$Accuracy_{doc} = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{1}$$

where $d_i = \mathbf{1}(Doc_i^{\text{pred}} = Doc_i^{\text{true}})$.

Page accuracy ($Accuracy_{page}$) measures page-level proximity within correctly identified documents:

$$Accuracy_{page} = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{2}$$

$$p_i = \begin{cases} 1 - \dfrac{|Page_i^{\text{pred}} - Page_i^{\text{true}}|}{n\_pages_i} & \text{if } d_i = 1, \\ 0 & \text{if } d_i = 0. \end{cases} \tag{3}$$

In addition, we report **page recall** ($Recall@k$) for correct page retrieval.

Hardware. All experiments run on a single NVIDIA RTX 5060 Ti with 16 GB of VRAM.

4.2 Embedding Model Selection

The embedding model is a critical component of the indexing pipeline because it determines how effectively relevant documents are retrieved during the initial search stage.

With a focus on computationally constrained environments, we selected the top embedding models with fewer than 2B parameters from the Multilingual RTEB leaderboard[3] and reevaluated them for Ukrainian on three retrieval tasks (Table 1) using our UNLP-QA dataset, along with Ukrainian subsets of the multilingual BelebeleRetrieval (Bandarkar et al., 2024) and WebFAQRetrieval datasets.

As a result, Snowflake Arctic Embed L v2.0[4] (568M parameters, 1024 dimensions) achieves the best balance across all three tasks.

4.3 Index Pipeline Optimization

Document Splitting. Drawing on prior work on document splitting strategies (Smith and Troynikov, 2024), we compared four main approaches: RecursiveCharacterTextSplitter (LangChain's default splitter) with varying chunk sizes and overlaps, SentenceSplitter (LlamaIndex's default splitter) at multiple chunk sizes, ClusterSemanticChunker (ChromaDB's proposed splitter) and full-page splitting.

In the results (Table 3 in the Appendix B), full-page splitting achieves the highest $Accuracy_{doc}$ and $Accuracy_{page}$ metrics, outperforming all sub-page strategies.

We attribute these findings to several factors.

First, in the organizer's dataset, each question's answer appears on a single page, so page-level chunks naturally preserve the source metadata needed for the evaluation metric, whereas sub-page

[3] http://mteb-leaderboard.hf.space/?benchmark_name=RTEB(beta)

[4] https://huggingface.co/Snowflake/snowflake-arctic-embed-l-v2.0

Model	BelebeleRetrieval				WebFAQRetrieval				UNLP-QA			
	$R@1$	$R@3$	$R@5$	$R@20$	$R@1$	$R@3$	$R@5$	$R@20$	$R@1$	$R@3$	$R@5$	$R@20$
Qwen3-Embedding-0.6B	82.67	92.11	93.22	97.44	61.19	73.80	77.80	87.21	53.44	72.27	79.96	94.94
Octen-Embedding-0.6B	82.44	92.33	94.00	97.11	62.17	74.34	78.25	87.74	54.25	74.09	80.97	95.75
Jina-Embed-v5-small	88.78	95.56	97.44	99.00	68.62	81.00	85.16	92.64	59.11	78.14	85.22	**97.17**
Arctic-Embed-L-v2.0	87.33	94.89	96.78	98.56	**71.71**	**83.73**	**87.30**	**93.99**	**65.18**	**80.57**	**86.44**	94.74
BGE-M3	88.00	95.67	97.56	99.00	69.26	81.39	84.92	93.01	61.34	78.34	85.43	94.74
Multilingual-E5-Large	**90.44**	**97.44**	**98.11**	**99.44**	71.04	82.67	86.29	93.59	61.34	78.75	85.43	96.15

Table 1: Embedding model comparison on three RTEB Ukrainian tasks. $R@k$ = $Recall@k$ (%).

splitting introduces ambiguity in page assignment when chunks span page boundaries.

Second, modern embedding models may handle longer contexts more effectively, making full-page chunks increasingly practical for such tasks.

Chunk Augmentation. Anthropic, in its engineering blog on contextual retrieval (Anthropic, 2024), showed that enriching chunks with document-level identifying context can improve their relevance during retrieval by making chunks easier to distinguish from similar ones.

In their approach, each chunk is enriched with additional contextual information generated from both the chunk itself and the broader source document. This method can significantly improve retrieval quality, although it is computationally expensive because it requires an LLM call for every chunk.

To make this approach suitable for computational-constrained environments, we adopted document-level summary augmentation, which has previously been used in RAG pipelines for legal-domain applications (Reuter et al., 2025). Instead of producing unique context for each chunk, we force the LLM to produce a short summary once per document, which is then prepended to all chunks derived from that document. This summary-based augmentation offers a favorable trade-off between cost and retrieval quality: it requires only a single LLM call per document, rather than one per chunk, while maintaining a distinction between similar chunks from different documents by grounding each chunk in its document-level context. Details are provided in Appendix A.

4.4 Question Answering Pipeline Optimization

Retrieval Strategies. To optimize the pipeline's retrieval step, various strategies can be incorporated, ranging from different search methods to advanced query transformation techniques. These methods generally aim to address the semantic differences between user queries and encoded document chunks.

In the strategy evaluation, we compared default dense retrieval, hybrid search (dense + BM25 sparse), Hypothetical Document Embeddings (HyDE) (Gao et al., 2023), multi-query expansion (Li et al., 2025), and their combinations.

We define the retrieval limit to the top 20 chunks as a pragmatic heuristic that provides comprehensive coverage of the relevant chunk space, and additional value beyond this threshold tends to degrade overall pipeline performance.

Default dense retrieval achieves the best results (Table 4 in Appendix B). Advanced query transformation and search methods do not improve performance on this dataset. We assume this is related to the nature of the questions: they are carefully hand-crafted with clear intent, unlike ambiguous real-user queries where such techniques typically excel. The evaluated techniques add noise rather than coverage for well-specified questions.

Reranking. For reranking optimization, we tested two main approaches: dedicated reranking models and LLM-based reranking such as listwise scoring (Ma et al., 2023) and pointwise logprob reranking.

The pointwise logprob method is based on adapting the monoT5 approach (Nogueira et al., 2020): for each document, we prompt the language model with the question-document pair, asking whether the document answers the question and requesting a "*Yes*" or "*No*" response. We extract token-level log-probabilities and compute relevance as:

$$s = \sigma\big(\log p(\text{Yes}) - \log p(\text{No})\big) \tag{4}$$

where σ is the sigmoid function. This yields a normalized score in $[0, 1]$ reflecting the model's confidence that the document is relevant.

The results (Table 5 in the Appendix B) did not indicate a consistently better-performing approach.

Without context augmentation, pointwise logprob reranking achieves the best $Recall@k$ results. With context augmentation, BGE Reranker v2-m3 outperforms all other approaches, achieving the highest $Accuracy_{doc}$, $Accuracy_{page}$, and $Recall@k$, though logprob reranking still delivers competitive results.

These findings suggest that a language model fine-tuned for Ukrainian can serve as an effective reranker, approaching the performance of dedicated cross-encoder models. This is particularly relevant in resource-constrained environments where deploying a separate reranking model is not feasible. In such cases, the same LLM can be used for both answer generation and reranking. With targeted fine-tuning on reranking objectives, these models could potentially be used without dedicated rerankers.

Context Assembly. After the reranking step, the top-ranked chunks are concatenated in order to form the final context passed to the language model for answer generation. For the competition submission, we used the top 3 chunks, which in our evaluations provided a balance between relevance page recall, input context compactness, and output generation time.

5 Results

Taking into account the results of all standalone RAG component optimization experiments, we selected two final configurations:

- **Approach A:** document-level contextual augmentation, full-page chunking, Arctic-Embed-L-v2.0 as the embedding model, and BGE Reranker v2-m3 as the reranker.
- **Approach B:** summary-based augmentation, full-page chunking, Arctic-Embed-L-v2.0 as the embedding model, and BGE Reranker v2-m3 as the reranker.

According to the evaluation metrics (Table 2), Approach A achieved the strongest overall performance. However, due to the environmental constraints of the UNLP 2026 Shared Task competition, we were unable to run Approach A in the final submission. Instead, we submitted Approach B, which ranked third on the private leaderboard.

Metric	Approach A	Approach B
$Accuracy_{doc}$	**99.78**	**99.78**
$Accuracy_{page}$	**95.35**	94.70
$Recall@1$	**78.96**	78.52
$Recall@3$	**93.49**	91.54
$Recall@5$	**95.23**	94.14
$Recall@10$	96.53	**96.75**
$Recall@20$	**97.61**	96.96

Table 2: Final system results on the development dataset. Approach A uses document-level contextual augmentation; Approach B uses summary-based augmentation.

6 Conclusion

We presented a modular RAG pipeline for the UNLP 2026 Shared Task on Ukrainian Multi-Domain Document Understanding, achieving 3rd place on the private leaderboard.

Our ablation study highlights several practical findings. For document splitting, full-page chunks consistently outperform sub-page strategies in both document-source accuracy and page-level recall because they naturally preserve the page-level metadata required by the evaluation metric. Chunk augmentation and its variations substantially improve retrieval quality across metrics by enriching chunks with additional document context, making it easier to distinguish them from similar ones. For retrieval, default dense search is most effective on this dataset, since the well-specified competition questions do not benefit from query transformation techniques such as HyDE or multi-query expansion. Among reranking approaches, dedicated cross-encoder models allow us to achieve the strongest overall performance, while LLM-based pointwise logprob reranking is competitive, suggesting that a Ukrainian-adapted language model can serve as an effective reranker in resource-constrained settings and may benefit from further fine-tuning for reranking objectives.

These results show that strong performance on the Ukrainian document-grounded question answering can be achieved through systematic isolated curation of RAG components, without additional language model adaptations. In future work, we plan to explore fine-tuning embedding and reranking models on Ukrainian data and to investigate approaches for visual document understanding.

Limitations

Despite these results, our work has several limitations.

First, we focused on the most impactful parts of the RAG pipeline while leaving aside more elaborate ETL and raw document loading techniques. This choice was partly motivated by the development dataset provided by the organizers, which does not contain corrupted or noisy data that would substantially affect evaluation results.

Second, the QA dataset does not include unanswerable questions. As a result, we cannot fully assess the distinction between cases where the LLM actually knows the answer and cases where it merely predicts one.

Third, the shared task evaluation does not distinguish between answers grounded in retrieved documents and answers generated from the LLM's internal knowledge. As a result, our system always produces an answer, even when the retrieved chunks are not sufficiently connected to the source document. This may lead to confidently incorrect outputs when retrieval fails.

Fourth, although we did not explicitly tune the system for the specific domains represented in the dataset, the final component selection may still have been influenced by the nature of the task and development datasets.

Finally, we do not claim that the proposed system will generalize reliably beyond the shared task environment. In real question-answering scenarios, user queries may be substantially more ambiguous and structurally complex than the benchmark questions used in this work. In real question answering systems, retrieval performance would likely require not only the indexing strategies explored in this work, but also additional query transformation techniques and hierarchical index architectures that align retrieval with domain-level understanding rather than tying it to specific pages or documents.

References

Anthropic. 2024. Introducing contextual retrieval. https://www.anthropic.com/engineering/contextual-retrieval. Anthropic Engineering Blog, accessed 2026-04-08.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. 2024. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 749–775, Bangkok, Thailand. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Minghan Li, Xinxuan Lv, Junjie Zou, Tongna Chen, Chao Zhang, Suchao An, Ercong Nie, and Guodong Zhou. 2025. Query expansion in the age of pre-trained and large language models: A comprehensive survey. *Preprint*, arXiv:2509.07794.

Frank Liu, Kenneth Enevoldsen, Roman Solomatin, Isaac Chung, Tom Aarsen, and Zoltán Fődi. 2025. Introducing rteb: A new standard for retrieval evaluation. Hugging Face Blog. Accessed: 2026-04-23.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *Preprint*, arXiv:2305.02156.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics:*

EMNLP 2020, pages 708–718, Online. Association for Computational Linguistics.

Yurii Paniv. 2025. Isolating LLM performance gains in pre-training versus instruction-tuning for mid-resource languages: The Ukrainian benchmark study. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI Era*, pages 876–883, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Markus Reuter, Tobias Lingenberg, Rūta Liepiņa, Francesca Lagioia, Marco Lippi, Giovanni Sartor, Andrea Passerini, and Burcu Sayin. 2025. Towards reliable retrieval in rag systems for large legal datasets. *Preprint*, arXiv:2510.06999.

Brandon Smith and Anton Troynikov. 2024. Evaluating chunking strategies for retrieval. Technical report, Chroma.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Suhang Wu, Jialong Tang, Baosong Yang, Hongcheng Guo, Ruifeng Wen, Zhefeng Wang, Baotian Hu, and Min Zhang. 2024. Not all languages are equal: Insights into multilingual retrieval-augmented generation. *Preprint*, arXiv:2410.21970.

Hanna Yukhymenko, Anton Alexandrov, and Martin Vechev. 2025. Mamaylm v1.0: An efficient state-of-the-art multimodal ukrainian llm.

A Chunk Augmentation Details

This appendix describes the implementation of the two chunk augmentation strategies evaluated in Section 4. Both operate on chunks produced by the splitter and enrich each chunk with additional contextual information before it is passed to the embedding model. During the evaluation, we used Ukrainian versions of the prompts. In this paper, the prompts are presented as direct English translations.

A.1 Summary-based Augmentation

In this strategy, we prompt the language model once per document to produce a short summary of the document, based on two variables: `{char_length}`, a target upper bound on the summary length in characters, and `{document_content}`, a text representation of the document. The resulting summary is prepended to each chunk of that document.

`{document_content}` is built from a fixed window of chunks rather than the full document. Given the window size w, if the document has at most w chunks, all chunks are used. Otherwise, the first $\lfloor w/2 \rfloor$ and last $\lceil w/2 \rceil$ chunks are concatenated, with an explicit placeholder inserted between the head and tail to indicate how many middle chunks were omitted. It bounds the prompt length even for long documents while informing the model that part of the document is missing from the input.

In our final configuration, we use $w = 10$, `{char_length}` = 250, and align each chunk with a single document page.

Summary-based Prompt

You are an expert at document summarization.
Provide a brief summary of the given document text, no longer than `{char_length}` characters. Focus on extracting the most important entities, the main purpose, and key topics. The summary should be concise and optimized to provide context for smaller text fragments. Output only the summary text.
Document:
`{document_content}`

The prompt below illustrates the value of `{document_content}` variable, for the case when a 20-page document is rendered with windows size $w = 10$: the first 5 and last 5 pages are kept, and the omission placeholder is inserted between them.

Example value of `{document_content}` variable for a 20-page document with windows size $w = 10$

[Page 1]
<*text of page 1*>
[Page 2]
<*text of page 2*>
⋮
[Page 5]
<*text of page 5*>
[... PART OF THE DOCUMENT OMITTED ...]
(10 more pages exist between the start and end. When summarizing, take into account that the middle of the document is missing from the input.)
[Page 16]
<*text of page 16*>
⋮
[Page 20]
<*text of page 20*>

A.2 Document-Level Contextual Augmentation

In this strategy we prompt the language model once per chunk to produce a short piece of context that situates the chunk within its document, based on

two variables: `{chunk_content}`, the chunk being contextualized, and `{doc_content}`, the document text surrounding that chunk. The generated context is appended to the chunk's content.

Given the window size w, `{doc_content}` is built by using $\lfloor w/2 \rfloor$ chunks before contextualized chunk and $\lceil w/2 \rceil$ chunks after it. Near document boundaries, the window is shifted rather than shrunk, so that every prompt contains the same number of neighboring chunks. This avoids a size mismatch between the prompts at the start/end of a document and those in the middle.

In our final configuration, we use the window size $w = 4$. Compared to Summary Augmentation, this strategy is substantially more compute-intensive because the number of LLM calls scales with chunks rather than documents.

Document-Level Contextual Augmentation Prompt

We want to situate a chunk in the context of the entire document.
Document:
`{doc_content}`
Chunk:
`{chunk_content}`
Please give a short succinct context to situate this chunk within the entire document for the purposes of improving search retrieval of the chunk. Answer only with the succinct context and nothing else.

B Detailed Experimental Results

Splitting Strategy	$\boldsymbol{Accuracy_{doc}}$	$\boldsymbol{Accuracy_{page}}$	$R@1$	$R@3$	$R@5$	$R@20$
Recursive (600, 0)	89.70	80.47	53.94	69.70	76.77	87.68
Recursive (800, 0)	89.90	80.30	51.92	69.50	76.97	88.28
Recursive (1200, 0)	92.12	81.45	52.32	69.70	74.75	87.48
Recursive (1200, 300)	91.52	81.95	53.94	70.91	77.17	87.88
Recursive (800, 300)	89.70	80.24	53.94	70.91	**77.98**	**88.49**
Full Page	**95.56**	**84.64**	**54.55**	**71.11**	75.76	87.07
Sentence (600, 0)	92.73	83.22	53.33	67.48	74.34	86.87
Sentence (800, 0)	94.14	83.35	49.09	65.66	71.72	83.43
Sentence (1200, 0)	94.34	83.77	48.89	67.07	71.72	84.04
ClusterSemantic (1600, 500)	93.94	82.42	48.49	63.64	68.08	79.80
ClusterSemantic (1200, 300)	93.33	81.46	45.46	63.43	70.10	82.02

Table 3: Document splitting strategy comparison. Numbers in parentheses: (chunk size, overlap) for Recursive and Sentence; (max size, min size) for ClusterSemantic. $R@k$ = $Recall@k$ (%).

Retrieval Strategy	$Accuracy_{doc}$	$Accuracy_{page}$	$R@1$	$R@3$	$R@5$	$R@20$
Without chunk augmentation						
Dense (default)	**95.56**	**84.64**	**54.55**	**71.11**	75.76	87.07
Hybrid (dense + sparse)	86.67	77.22	50.30	71.11	75.56	88.08
HyDE	91.31	81.44	54.34	70.91	**76.97**	**88.28**
HyDE + Hybrid	77.37	68.65	45.45	68.69	76.36	88.08
Multi-Query	94.34	82.64	47.88	67.27	72.73	85.86
Multi-Query + Hybrid	81.41	71.62	43.84	60.81	70.10	88.89
With chunk augmentation (document-level contextual augmentation)						
Dense (default)	**98.18**	88.68	**63.03**	**85.25**	**89.29**	96.16
Hybrid (dense + sparse)	92.53	84.42	60.00	82.83	88.08	**96.36**
HyDE	97.98	**89.15**	62.83	85.05	88.89	96.16
HyDE + Hybrid	90.10	80.26	54.95	79.39	85.05	95.15
Multi-Query	97.98	88.33	58.99	80.20	88.08	95.96
Multi-Query + Hybrid	86.87	77.29	50.51	73.33	82.22	97.17

Table 4: Retrieval strategy comparison with and without chunk augmentation. $R@k$ = $Recall@k$ (%).

Reranking Model	$Accuracy_{doc}$	$Accuracy_{page}$	$R@1$	$R@3$	$R@5$	$R@10$
Without chunk augmentation						
Without reranking	**95.56**	84.64	54.55	71.11	75.76	81.21
Logprob Reranker	88.48	83.38	**67.07**	**79.60**	**83.23**	**86.46**
Qwen3-Reranker-0.6B	88.08	80.68	57.78	74.14	78.18	83.84
Jina Reranker v3	89.49	81.94	60.61	74.75	78.99	83.23
LLM Reranker (listwise)	93.87	84.65	57.55	71.99	78.12	82.28
ContextualAI-Rerank-v2-1B	91.92	83.82	60.81	75.56	78.99	85.05
BGE Reranker v2-m3	93.13	**85.94**	64.24	77.58	80.81	85.05
With chunk augmentation (document-level contextual augmentation)						
Without reranking	98.18	88.68	63.03	85.25	89.29	93.74
Logprob Reranker	98.92	94.92	**80.26**	91.76	94.36	96.31
Qwen3-Reranker-0.6B	95.76	87.89	65.05	86.87	91.11	94.95
Jina Reranker v3	96.97	90.04	68.69	85.45	89.70	94.14
LLM Reranker (listwise)	98.73	90.48	68.53	83.25	88.83	91.88
ContextualAI-Rerank-v2-1B	97.78	92.42	72.93	89.49	91.92	94.95
BGE Reranker v2-m3	**99.78**	**95.35**	78.96	**93.49**	**95.23**	**96.53**

Table 5: Reranking model comparison with and without chunk augmentation. $R@k$ = $Recall@k$ (%).