Evaluation of Two Leading Polish Language Models in a Real-World RAG Scenario

Szymon Bartanowicz, Krzysztof Jassem
Adam Mickiewicz University, Adam Mickiewicz University
Center for Artificial Intelligence, Center for Artificial Intelligence
szybar4@st.amu.edu.pl, krzysztof.jassem@amu.edu.pl

Abstract

This paper presents a comparative evaluation of two leading Polish instruction-tuned language models, *Bielik-11B-v2.3-Instruct* and *PLLuM-12B-nc-chat*, within a real-world Retrieval-Augmented Generation (RAG) system designed for the technical documentation of a low-code platform. The study aims to identify the optimal configuration of retrieval and generation components for Polish-language applications. The evaluation was conducted in two stages. First, several embedding models and retrieval methods were tested using standard information retrieval metrics, including NDCG. The *OrlikB/KartonBERT-USE-base-v1* model combined with vector-based retrieval achieved the highest performance and was adopted for the second stage. In the generation phase, both models were evaluated using quantitative scoring and pairwise A/B testing with multiple evaluators to ensure robustness. Results show that *Bielik-11B-v2.3-Instruct* consistently outperformed *PLLuM-12B-nc-chat* in producing accurate and contextually relevant answers. The study highlights the importance of constructing a reliable golden set, employing a two-phase evaluation pipeline, and selecting appropriate metrics to ensure objective and reproducible assessment of RAG systems in real-world Polish-language contexts.

Keywords: Retrieval-Augmented Generation (RAG), large language models (LLMs), evaluation methodology, question answering based on technical documents

1. Introduction

One of the main limitations of large language models (LLMs) is their restricted access to up-to-date and domain-specific knowledge. A promising approach to overcoming this issue is the Retrieval-Augmented Generation (RAG) architecture, which combines information retrieval with text generation. In such systems, relevant documents are retrieved from a knowledge base and used to ground the model's responses, improving factual accuracy and enabling adaptation to specialized domains.

This study was conducted as part of a collaborative project between an industrial partner specializing in low-code business software and a university research center focused on artificial intelligence. The company's main platform is used for managing and automating business processes. Its complexity results in an extensive and detailed documentation base, which poses challenges for both developers and end users in locating relevant information efficiently. To address this, a RAG-based assistant was developed, powered by the platform's internal documentation.

The documentation, provided in JSON format, was preprocessed by splitting each webpage into semantically coherent sections and extracting raw text from HTML structures. These text fragments were then embedded into dense vector representations using transformer-based embedding models and stored in a vector database. Several configurations of embedding models and retrieval methods were evaluated against the golden set of questions and answers to determine the optimal setup.

On top of this retrieval layer, two state-of-the-art Polish language models — Bielik-11B-v2.3-Instruct (Ociepa et al., 2025) and PLLuM-12B-instruct (CYFRAGOVPL, 2025) — were evaluated automatically as the generative component.

Our contribution lies in the systematic identification and empirical validation of an optimal retrieval–generation configuration that maximizes factual accuracy and contextual relevance in a real-world Polish RAG system designed for technical-documentation search.

2. Related work

RAG evaluation and benchmarks. Domain- and task-specific benchmarks for English-language RAG systems have recently been introduced. RAG-Bench presents a 100k-example, industry-oriented corpus with explainable annotations (Friel et al., 2024), while MIRAGE focuses on the medical domain (Xiong et al., 2024). These works highlight that robust RAG assessment requires the use of explicit retriever metrics, such as Normalized Discounted Cumulative Gain (NDCG), as well as grounded answer evaluation.

Judgment via LLMs and evaluation bias. The concept of "LLM-as-a-Judge" — large language models serving as evaluators of other models' outputs — was introduced by (Zheng et al., 2023). A 2024 survey further analyzes the limitations and challenges of such model-based evaluation (Zhang et al., 2024), while complementary studies identify

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 2698–2704
11–16 May 2026.

and categorize twelve distinct types of bias affecting LLM judgments (Ye et al., 2025). These findings motivate the use of order-controlled A/B and reversed B/A protocols, as discussed by (Yin et al., 2025), which we also adopt in our study.

Retrievers, embeddings, and hybrid search. It was shown in (Karpukhin et al., 2020) that dense, BERT-based retrievers outperform sparse methods when trained on English QA pairs. In production systems, hybrid retrieval combining semantic vector search with sparse BM25 ranking is increasingly used (Milvus Documentation, 2024). Our results align with this line of work, yet indicate that a strong Polish-specific encoder (KartonBERT-USE) with pure vector search can outperform hybrid variants.

Polish LLMs for grounded generation. Polish instruction-tuned LLMs have advanced rapidly. The *Bielik 11B v2 technical report* (Ociepa et al., 2025) describes the *Instruct* line of models also used in our experiments. In parallel, the public-administration-backed PLLuM family (12B scale) (CYFRAGOVPL, 2025) publishes base and instruct checkpoints on Hugging Face, enabling direct comparison in Polish RAG settings. Despite similar parameter scales, our controlled A/B results favor Bielik-11B-v2.3-Instruct, underscoring that model choice should be validated in-domain rather than inferred from general leaderboards.

3. Experiments

3.1. Creation of a golden set

We have obtained a list of 146 reference questions, carefully selected as those most likely to be asked to the system. An initial retrieval module (*OrlikB/KartonBERT-USE-base-v1*) queried a database consisting of approximately 1,200 document chunks for each of these questions, and the ten most relevant documents were saved as lists in separate files. These files, along with the questions, were then passed to company employees who, using their expert knowledge, evaluated how accurately the system retrieved the documents. The evaluation was performed on a three-point scale:

- 0: irrelevant document,
- 1: partially relevant document,
- 2: fully relevant document.

If, according to the annotators, any relevant documents were missing from the list, they were manually added. Documents that received a score of zero were excluded from the set of relevant documents when constructing the reference (golden) set. Importantly, these documents were not removed from the retrieval corpus and could still be returned by retrieval systems during evaluation. In this way, a reference (golden) set was created, which was essential for evaluating the performance of the retrieval module. The distribution of relevant documents per query was skewed towards smaller values: most queries had between 1 and 3 relevant documents. In particular, 37 queries had exactly 1 relevant document, 27 had 2, and 25 had 3. The maximum number of relevant documents per query was 8.

3.2. Two-step experiments

The experimental process was divided into two main stages, each focusing on a different component of the system: the retrieval module and the . The goal of this two-step design was to first optimize the retrieval pipeline and then use the best-performing setup to provide input for the evaluation of the generation component.

Stage 1: Retrieval optimization. The first stage focused on selecting the most suitable embedding model and retrieval method. The *Polish Information Retrieval Benchmark* served as a reference point for model preselection (Dadas et al., 2024). An additional practical constraint was imposed on model size: the embedding model could not exceed 1.5 GB of GPU memory. Taking these requirements into account, four models were selected for comparison, each stored in a separate collection within the *Milvus* vector database:

- *OrlikB/KartonBERT-USE-base-v1*,
- *sdadas/mmlw-roberta-base*,
- *sdadas/mmlw-e5-base*,
- *intfloat/multilingual-e5-base*.

Alongside model selection, three retrieval methods supported by *Milvus* were tested:

- **Vector search** — based on cosine similarity between dense embeddings,
- **Full-text search** — using sparse keyword-based ranking (BM25),
- **Hybrid search** — combining both vector and sparse representations to balance semantic and lexical relevance.

Each configuration (model–method pair) was evaluated against the reference (golden) set using standard information retrieval metrics. The results of this comparative analysis indicated that the configuration using the *OrlikB/KartonBERT-USE-base-v1* model with vector-based search achieved the

highest retrieval accuracy. This setup was therefore selected as the default configuration for the second experimental stage. Detailed results for all tested models and retrieval methods are presented in Section 3.4.

Stage 2: Generation evaluation. In the second stage, the previously optimized retrieval configuration was used to provide the generation module with the top five documents returned by the retrieval system for each query. Using more documents could potentially improve evaluation metrics by increasing the likelihood of including relevant information, but at the same time, the model has a limited context window and we wanted to avoid overwhelming it with excessive input. Selecting five documents was therefore a practical balance between providing sufficient context for accurate generation and staying within the model's context limits.

The purpose of this step was to evaluate the quality of the generated answers while ensuring that retrieval performance did not overly constrain the results. By structuring the experiments into two consecutive phases, it was possible to clearly separate the evaluation of retrieval quality from the assessment of generation accuracy, ensuring that the final analysis reflected the true performance of the generation module given the most relevant documents identified by the retriever.

3.3. Evaluation

The evaluation consisted of two complementary parts, corresponding to the main components of the RAG system: the retrieval module and the generation module. Each was assessed using dedicated metrics and procedures appropriate to its role in the pipeline.

3.3.1. Retrieval module

Retrieval quality was measured using four metrics: *Accuracy*, *Recall*, *F1-score*, and *NDCG*. All were computed for varying values of *top-k*, where $k \in \{1, \ldots, 8\}$, meaning that for each query, up to k highest-ranked documents were considered. The evaluation was carried out against the reference (golden) set. Among the tested metrics, *NDCG* (Normalized Discounted Cumulative Gain) was selected as the primary criterion, as it accounts not only for the presence of relevant documents but also for their ranking order, expected in real-world conditions.

3.3.2. Generation module

The evaluation of the generation module followed the "LLM-as-a-Judge" approach (Zheng et al., 2023). To ensure robustness and reduce bias, three independent evaluator models were used: *gpt-oss-20b*, *Mistral-Small-3.2-24B-Instruct-2506*, and *Qwen3-30B-A3B-Instruct-2507*. These models originate from different providers, are open-weight, and instruction-tuned, which allows for a diverse and balanced evaluation of the generated answers. Each generated answer was rated on a five-point scale, where 5 indicated a fully correct and comprehensive answer and 1 an incorrect or irrelevant one. The scoring model received structured prompts containing the question, the generated answer, and the context used during generation. The structure of this instruction is shown below:

```
prompt = f"""
Question: {question}
Answer: {model_answer}
Context: {full_context}

Evaluate the answer (Answer) to the question (Question)
based on the provided context (Context).

Scoring instructions:
Rate the answer on a scale of 1-5, where:
{scoring_instructions}
"""
```

Code listing 1: Prompt structure used for computing the average score

This allowed for a fine-grained evaluation of the generated answers. In addition to the scoring-based evaluation, an A/B testing procedure was applied to compare pairs of answers directly. The goal of this approach was to determine which of the two responses was more accurate and complete given the same context. The model received both answers together with the original question and supporting documents and was asked to choose the better one.

However, language models are known to exhibit a tendency to prefer responses based on their order in the input prompt, a phenomenon known as *order bias* (Yin et al., 2025). To mitigate this effect, both A/B and reversed B/A tests were performed—i.e., the same pair of answers was presented in swapped order.

The instruction used for these comparisons had the following structure:

```
prompt = f"""
Compare the two answers (answer_a and answer_b) and
choose the one that is more complete and accurate
based on the provided context.

Question: {question}
Answer A: {answer_a}
Answer B: {answer_b}
Context: {context}
"""
```

Code listing 2: Prompt structure used in A/B testing

By combining quantitative metrics with pairwise preference testing, the evaluation provided both absolute and comparative insights into the performance of the generation models.

3.4. Results

3.4.1. Retrieval module

Table 1 and Table 2 present the averaged results for all evaluated configurations. Due to the large amount of data, only results for $k \in \{3, 5, 7\}$ are shown. The first table includes results for the *full text* retrieval method, where the embedding model did not affect performance, while the second table reports results for the *vector* and *hybrid* retrieval methods across all four embedding models.

As expected, increasing k generally leads to higher metric values, as more relevant documents are likely to appear among the top-ranked results. However, setting k too high may distort the evaluation, as it inflates recall at the cost of precision. After analyzing the rate of metric growth between consecutive k values, $k = 5$ was determined to be the optimal cutoff point for evaluating retrieval performance.

The results show that the *OrlikB/KartonBERT-USE-base-v1* model combined with the *vector-based* retrieval method achieved the best overall performance across all metrics: *Accuracy*, *Recall*, and particularly *NDCG*, which was the primary optimization criterion. The second-best performance was observed for the same model with the *hybrid* retrieval method. The remaining models, such as *mmlw-roberta-base* and *multilingual-e5-base*, achieved moderate results, while *mmlw-e5-base* performed notably worse, especially in vector mode, suggesting limited suitability for domain-specific Polish text.

3.4.2. Generation module

Table 3 presents the averaged scoring results across all evaluation models. The results indicate that *Bielik-11B-v2.3-Instruct* consistently achieves higher scores than *PLLuM-12B-nc-chat*, regardless of the evaluation model used.

Tables 4 and 5 show the averaged results of the A/B tests conducted on the same set of 146 reference questions as in the scoring-based evaluation. Each table reports how many answers from a given model were judged to be better, with percentages indicating their share of the total question set.

On average, *Bielik-11B-v2.3-Instruct* responses were preferred in 81.5% of A/B cases (where Bielik appeared second), compared to 18.5% for *PLLuM-12B-nc-chat*. In the reversed B/A setup, Bielik's preference rate dropped to 54.3%, again outperforming PLLuM's 45.7%, although the advantage decreased substantially. The average counts across all evaluators—119.0 vs. 27.0 in A/B and 79.3 vs. 66.7 in B/A—further reinforce this consistent advantage.

The findings suggest that *Bielik-11B-v2.3-Instruct* consistently produces more accurate and complete answers than *PLLuM-12B-nc-chat*, irrespective of the evaluation model. Only in the reversed B/A test with the *Qwen3-30B-A3B-Instruct-2507* evaluator *PLLuM-12B-nc-chat* obtained a higher score. These outcomes also confirm the presence of the well-documented *order bias* phenomenon (Yin et al., 2025).

The most stable results were obtained with the *gpt-oss-20b* evaluator, where the decrease in Bielik's preference rate between the A/B and B/A settings was minimal (from 77.4% to 70.5%). This consistency suggests that *gpt-oss-20b* is less affected by positional bias than other evaluation models.

Overall, both the quantitative and comparative analyses demonstrate that *Bielik-11B-v2.3-Instruct* is the superior Polish-language generative model in this real-world RAG setup, achieving higher factual consistency and completeness in its answers while maintaining stable performance across evaluation methods.

Table 1: Average results for $k \in \{3, 5, 7\}$ across all questions for full text search (identical across all embedding models)

k	Accuracy@k	Recall@k	F1-score@k	NDCG@k
3	0.597	0.334	0.292	0.364
5	0.705	0.433	0.280	0.393
7	0.752	0.489	0.253	0.411

Table 2: Average results for $k \in \{3, 5, 7\}$ across all questions for hybrid and vector search

Model	Method	k	Accuracy@k	Recall@k	F1-score@k	NDCG@k
KartonBERT-USE-base-v1	vector	3	0.868	0.626	**0.529**	0.656
		5	0.891	0.744	0.485	0.682
		7	**0.899**	**0.796**	0.424	**0.701**
	hybrid	3	0.853	0.590	0.494	0.628
		5	0.891	0.660	0.435	0.638
		7	0.891	0.676	0.401	0.642
multilingual-e5-base	vector	3	0.775	0.496	0.414	0.541
		5	0.806	0.587	0.380	0.562
		7	0.860	0.649	0.339	0.584
	hybrid	3	0.767	0.496	0.415	0.535
		5	0.791	0.576	0.386	0.549
		7	0.845	0.622	0.365	0.564
mmlw-roberta-base	vector	3	0.729	0.448	0.374	0.460
		5	0.783	0.535	0.346	0.476
		7	0.814	0.609	0.322	0.504
	hybrid	3	0.744	0.471	0.393	0.485
		5	0.806	0.568	0.374	0.508
		7	0.845	0.621	0.361	0.531
mmlw-e5-base	vector	3	0.411	0.179	0.156	0.200
		5	0.496	0.232	0.154	0.211
		7	0.589	0.292	0.153	0.238
	hybrid	3	0.426	0.192	0.169	0.212
		5	0.620	0.332	0.218	0.266
		7	0.721	0.477	0.262	0.325

Table 3: Average evaluation scores on a 1–5 scale

	PLLuM-12B-nc-chat	Bielik-11B-v2.3-Instruct
gpt-oss-20b	3.466	**4.192**
Mistral-Small-3.2-24B-Instruct-2506	4.219	**4.548**
Qwen3-30B-A3B-Instruct-2507	4.390	**4.822**
Average	4.025	**4.521**

Table 4: A/B test results (A before B)

Model	PLLuM-12B-nc-chat (A)	Bielik-11B-v2.3-Instruct (B)
gpt-oss-20b	33 (22.6%)	**113** (77.4%)
Mistral-Small-3.2-24B-Instruct-2506	34 (23.3%)	**112** (76.7%)
Qwen3-30B-A3B-Instruct-2507	14 (9.6%)	**132** (90.4%)
Average	27.0 (18.5%)	**119.0** (81.5%)

Table 5: A/B test results (B before A)

Model	Bielik-11B-v2.3-Instruct (B)	PLLuM-12B-nc-chat (A)
gpt-oss-20b	**103** (70.5%)	43 (29.5%)
Mistral-Small-3.2-24B-Instruct-2506	**82** (56.2%)	64 (43.8%)
Qwen3-30B-A3B-Instruct-2507	53 (36.3%)	**93** (63.7%)
Average	**79.3** (54.3%)	66.7 (45.7%)

4. Conclusions

In this study, we evaluated two state-of-the-art Polish language models, *Bielik-11B-v2.3-Instruct* and *PLLuM-12B-nc-chat*, within a real-world Retrieval-Augmented Generation (RAG) scenario applied to the technical documentation of the real-world platform. The experiments were conducted in two stages: first, optimizing the retrieval module, and then assessing the generative performance of the language models using a combination of scoring-based metrics and A/B preference testing.

The results demonstrate that the *OrlikB/KartonBERT-USE-base-v1* embedding model with vector-based search achieved the best retrieval performance, providing a reliable and semantically rich set of documents for downstream generation. Among the generative models, *Bielik-11B-v2.3-Instruct* consistently outperformed *PLLuM-12B-nc-chat*, achieving higher average scores in both quantitative evaluation and pairwise A/B testing. The analysis also confirmed the presence of positional bias in preference testing, which was mitigated through reversed-order evaluations.

The experiments further highlighted that implementing a RAG system requires a tailored approach: strong performance in one configuration does not guarantee universal applicability. Critical factors such as the completeness and quality of the knowledge base, the choice of evaluation metrics, and continuous interaction with end-users were shown to strongly influence system effectiveness. Regular evaluation and iterative refinement remain essential to ensure that the system continues to meet practical requirements.

Overall, our findings indicate that combining a high-quality, Polish-specific embedding model with a strong instruction-tuned language model yields an effective RAG system for domain-specific technical documentation. Beyond the quantitative outcomes, this study underscores the importance of careful system design, rigorous evaluation, and user-centric development practices, offering actionable guidance for future implementations of Polish-language RAG systems.

5. Limitations and future work

Future research should address several limitations identified during the evaluation process. First, the list of suggested documents presented to annotators was generated using the *KartonBERT-USE-base-v1* model, which also achieved the best performance overall. This may have introduced bias, as employees evaluated only the results from this single model without a comparative perspective. A more balanced approach would involve presenting evaluators with document sets generated by multiple models to ensure a more independent assessment.

Moreover, while vector representations were constructed from document fragments, the evaluators received links to entire sections. This mismatch may have led to inconsistent evaluations—relevant fragments could have been penalized due to the broader context of the section, or irrelevant fragments could have been rated higher if the section happened to contain relevant information elsewhere. Future evaluations should therefore be conducted at the fragment level to achieve finer-grained and more reliable results.

Finally, although automatic evaluation techniques enable scalable and repeatable assessments, they inherently lack expert-level judgment. Incorporating human, domain-specific evaluation would significantly enhance the validity and interpretability of the results, providing a more comprehensive understanding of both the retrieval and generation modules.

References

CYFRAGOVPL. 2025. Pllum-12b-instruct: Model card. https://huggingface.co/CYFRAGOVPL/PLLuM-12B-instruct.

Sławomir Dadas, Michał Perełkiewicz, and Rafał Poświata. 2024. Pirb: A comprehensive benchmark of polish dense and hybrid text retrieval methods.

Robert Friel et al. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation. In *Proceedings of a 2024 venue (preprint: arXiv:2407.11005)*.

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 2020)*, pages 6769–6781. Introduces DPR, the standard English dense retriever trained monolingually on QA pairs.

Milvus Documentation. 2024. Multi-vector hybrid search. https://milvus.io/docs/multi-vector-search.md.

Krzysztof Ociepa, Łukasz Flis, Krzysztof Wróbel, Adrian Gwoździej, and Remigiusz Kinas. 2025. Bielik 11b v2 technical report. Technical Report arXiv:2505.02410, arXiv.

Guang Xiong et al. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of ACL 2024*.

Jiawei Ye et al. 2025. Justice or prejudice? quantifying biases in llm-as-a-judge. In *OpenReview (2024/2025)*.

Haonan Yin, Shai Vardi, and Vidyanand Choudhary. 2025. Fragile preferences: A deep dive into order effects in large language models. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL 2025)*.

Yuwei Zhang et al. 2024. Llms-as-judges: A comprehensive survey on llm-based evaluation. *arXiv preprint arXiv:2412.05579*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.