UDC 004.8

V. Oliinyk, P. Ponochovnyy

EXPERIMENTAL EVALUATION OF RAG COMPONENTS AND THEIR IMPACT ON THE PERFORMANCE OF CUSTOMER SUPPORT CHATBOTS

Abstract: This paper presents an experimental evaluation of Retrieval-Augmented Generation (RAG) components for developing an intelligent support chatbot capable of processing complex documents. The study examines multiple frameworks, vector databases, and text-chunking strategies to determine the optimal configuration ensuring accuracy, contextual completeness, and efficiency. Results show that the LangChain framework achieves higher accuracy and context coverage than LlamaIndex, while FAISS provides superior answer relevancy, faithfulness, and processing speed. A chunk size of 1000 with an overlap of 50 yields the best balance between precision, recall, and response time. The combination of LangChain, FAISS, and the 1000/50 chunking configuration establishes a robust foundation for a high-performance RAG-based chatbot delivering accurate, faithful, and contextually relevant responses.

Keywords: Retrieval-Augmented Generation, LangChain, LlamaIndex, FAISS, chatbot, vector database, context retrieval, chunking strategy, LLM.

Introduction

Modern natural language processing technologies, particularly large language models (LLMs), open new possibilities for developing intelligent support chatbots [1,2]. The Retrieval-Augmented Generation (RAG) approach, which combines text generation with the retrieval of relevant information, enables more accurate and context-aware responses compared to traditional models [3,4]. However, the effectiveness of RAG systems depends on the proper configuration of their components, which directly influences the quality and speed of query processing.

This paper investigates the impact of key components of RAG systems, including frameworks, vector databases, and text chunking parameters, on the performance of support chatbots. Experiments were conducted using domain-specific documents such as insurance contracts, with evaluation metrics designed to assess the relevance, factual accuracy, and contextual completeness of generated responses.

The objective of this study is to identify efficient configurations of RAG systems for developing automated support chatbots capable of handling complex queries in domains such as insurance, healthcare, e-commerce, and technical support. Performance is evaluated based on accuracy, completeness, factual consistency, relevance, and response time–factors that are critical for ensuring high-quality, user-oriented interactions. The results of this study

 ISSN 1560-8956

contribute to the formulation of practical recommendations for improving user service quality by delivering precise and timely responses.

Related research and publications

Research on Retrieval-Augmented Generation (RAG) has confirmed its effectiveness for natural language processing tasks, particularly in support chatbots [1,4]. In [5] the concept of RAG was introduced as an integration of parametric memory (language model) with non-parametric memory (vector database), emphasizing the importance of retrieval part that has significant impact on the accuracy and factual consistency of generated responses. The study in [3] systematizes existing RAG approaches, identifying key components–retrieval, generation, and augmentation–and emphasizing the role of frameworks such as LangChain in simplifying their integration.

The research in [6] highlights the impact of including multi-query in the retrieval phase, while the work in [7] focuses on optimizing text chunking strategies, showing that parameters such as chunk size and chunk overlap affect the balance between retrieval accuracy and contextual completeness.

These studies highlight the necessity of carefully tuning RAG components to ensure high-quality responses in dialogue systems. The decision to investigate frameworks, vector databases, and text chunking parameters (chunk size, chunk overlap) in the present work is based on the findings of these prior studies. Frameworks were selected for analysis due to their ability to optimize the integration of retrieval and generation components, which is critical for support chatbots, as discussed in [3]. Vector databases are examined for their influence on retrieval speed and precision, as noted in [5], particularly when processing complex documents such as insurance contracts. Finally, the parameters chunk size and chunk overlap were chosen for analysis because, as shown in [7, 8], they directly affect contextual quality and completeness – factors essential for ensuring response relevance.

A comprehensive examination of these components enables the formulation of recommendations for designing highly efficient RAG-based systems tailored to the requirements of support chatbots across diverse domains.

Proposed solution

For the development of the support chatbot system, the RAG model shown in Figure 1 was defined. The proposed model includes two main components, described below.

1. Data Preparation

- *Raw Data Sources*: Data is obtained from text files or other document formats.
- *Information Extraction*: Relevant information is extracted for further processing.
- *Chunking*: Data is divided into smaller, manageable parts suitable for analysis.

– *Embedding*: Logical data segments are encoded into vector representations for storage in a vector database. This enables the system to identify semantic similarities between user queries and stored data.

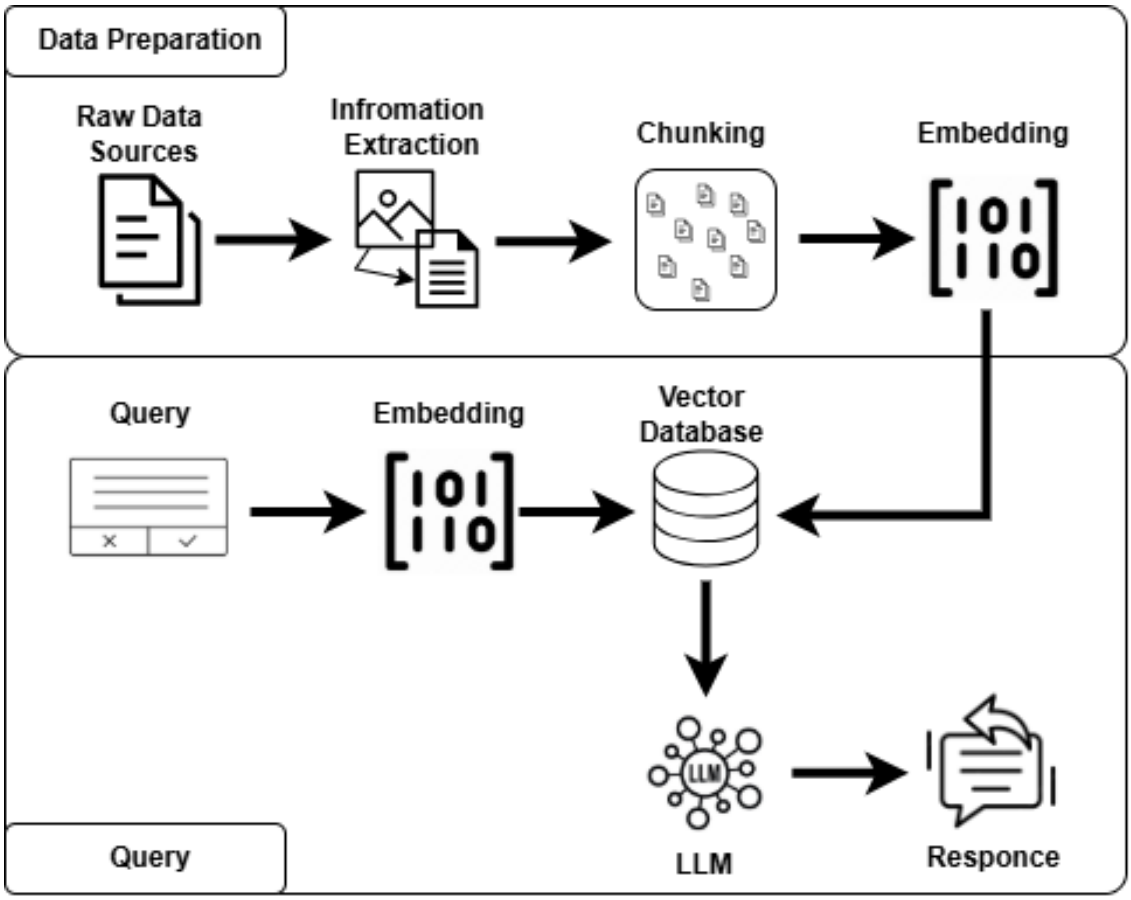

Figure 1. Architecture and Workflow of the RAG System

2. Query Processing

– *Query*: The user submits a request to the system.
– *Embedding*: The query is converted into a vector representation.
– *Vector Database*: The vector database retrieves the most relevant data segments based on the query.
– *Relevant Data*: Retrieved data fragments are passed to the generation component.
– *Large Language Model (LLM)*: The LLM processes the user's query together with the retrieved data to generate a personalized and contextually relevant response.
– *Response*: The generated response is returned to the user.

This RAG system workflow is used as the foundation for the conducted experiments.

Description of Components for Investigation

1. **Framework Selection**. Developing a RAG-based support chatbot requires selecting an appropriate framework optimized for retrieval and language model integration. Two widely used frameworks were considered: LangChain and LlamaIndex, both designed to facilitate interaction with large language models and retrieval pipelines.

 ISSN 1560-8956

Table 1.

Parameter Combinations of *Chunk Size* and *Chunk Overlap* Used in the Experiment

Characteristic	**InMemory VectorStore**	**Chroma**	**FAISS**	**LanceDB**
Storage Type	In-memory	Local storage	Local (with optional disk integration)	On-disk with scalability support
Access Speed	Very high	High	High (especially with GPU acceleration)	High
Scalability	Limited	Limited	High	High
Multimodality Support	No	Yes	Partial	Yes

LangChain is a framework for building systems powered by language models. Its main advantage is the ability to structure data processing into chains, which ensures context preservation throughout the dialogue. LangChain supports integration with various data sources, including knowledge bases and APIs, making it highly effective for RAG applications where response accuracy and relevance are crucial.

LlamaIndex is a data indexing and retrieval tool that leverages vector representations for efficient information access. Its primary strength lies in optimizing the process of retrieving relevant data and passing it to the language model. This enhances response precision in RAG systems, particularly when working with large volumes of textual information.

2. Vector Database Selection. The choice of vector database significantly affects the quality of chatbot responses. Therefore, several popular databases with distinct characteristics and scalability options were selected for comparison. The comparative characteristics of the chosen databases are summarized in Tab. 1.

3. **Chunking Parameter Configuration.** Four configurations of the parameters chunk size (maximum length of a text fragment) and chunk overlap (the degree of overlap between adjacent chunks to preserve textual coherence) were selected for testing. These configurations, presented in Tab. 2, enable the evaluation of how different parameter combinations affect the quality and consistency of generated responses.

Experimental results

Gemma 3 models, which operate solely on text input (no direct visual features), achieved higher top-1 scores – up to 71.25% (27B) – and up to 95% top-3 accuracy, although at the cost of significantly higher inference times (up to 6.9 seconds).

Table 2.

Parameter Combinations of *Chunk Size* and *Chunk Overlap* Used in the Experiment

No	Chunk Size	Chunk Overlap	Description
1	500	50	Small chunks with minimal overlap – used to assess the risk of context loss caused by excessively short text fragments.
2	1000	50	Medium chunk size with minimal overlap – used to examine whether limited overlap provides sufficient contextual continuity.
3	1000	200	Medium chunk size with moderate overlap – used to evaluate the balance between contextual completeness and processing efficiency.
4	1500	500	Large chunks with high overlap – used to analyze performance for long document fragments where context preservation is critical.

The study and selection of the RAG system components described above aim to identify the most effective solutions for implementing a support chatbot.

Evaluation Metrics

RAGAs is a framework designed to provide a comprehensive evaluation of RAG systems, encompassing both individual components and the overall process. Its key advantage lies in reference-free evaluation, which leverages large language models (LLMs) instead of manually annotated datasets, thereby making the assessment process faster and more cost-efficient.

RAGAs requires several inputs for operation: a question, answer, contexts, ground-truth responses, and file search results. This enables efficient evaluation of response relevance and accuracy. Ground-truth responses in this study were obtained using the OpenAI Assistants API, which integrates seamlessly with applications and supports tools for file retrieval, code interpretation, and query handling.

RAGAs also supports annotated metrics and automatic generation of test data, making it an effective framework for assessing the response quality of RAG-based systems.

The following metrics were selected for system evaluation:

1. *Context Precision* – measures the ratio of useful information to noise within retrieved contexts.
2. *Context Recall* – assesses whether the retrieved contexts contain all necessary information.
3. *Faithfulness* – evaluates the factual consistency of the response with the retrieved contexts.
4. *Answer Relevancy* – measures how well the generated answer addresses the user's query in terms of completeness and accuracy.

 ISSN 1560-8956

5. *Answer Semantic Simila*rity – quantifies the semantic similarity between the generated response and the ground-truth answer.
6. *Response Time* – measures the latency of answer generation.

All metrics are normalized within the range [0, 1], where higher values indicate better performance.

Experimental Setup

To conduct the experiments, evaluation functions for the respective RAG system components were implemented. An insurance contract from an open public dataset was chosen as a test document.

Experiment No. 1

Two basic question–answer (Q&A) systems were implemented using the LangChain and LlamaIndex frameworks.

Both systems were tested with identical queries based on the selected document, and their responses were evaluated using the predefined metrics: Context Precision, Context Recall, Faithfulness, Answer Relevancy, Answer Semantic Similarity, and Response Time.

For convenience of comparison, the average values of all metrics were calculated for each system. The summarized results are presented in Tab. 3. Subsequent experiments report only the mean values of metrics.

Table 3.

Average performance metric values for LangChain and LlamaIndex frameworks

Framework	**Relevancy**	**Faithfulness**	**Context recall**	**Context precision**	**Semantic Similarity**	**Elapsed time**
LangChain	0.96362	0.808404	0.843333	0.788889	0.890264	3.60444
LlamaIndex	0.69379	0.829048	0.711429	0.75	0.877473	1.51861

Based on the obtained evaluation results, LangChain is the most suitable choice for tasks where accuracy, relevance, and context completeness are of primary importance. Its answer relevancy score reached 0.96362, which is significantly higher compare to LlamaIndex (0.69379). Furthermore, the context recall metric for LangChain was noticeably superior, indicating that LangChain provides broader coverage of the relevant information. These metrics collectively demonstrate that LangChain is better suited for applications where high-quality, detailed responses are essential.

Conversely, LlamaIndex showed a considerable advantage in terms of response time, achieving an average of 1.51861 seconds, compared to 3.60444 seconds for LangChain. This higher speed makes it more effective for interactive or time-critical scenarios, where rapid response generation is a key requirement. Additionally, the faithfulness metric for LlamaIndex was slightly higher (0.829048 versus 0.808404 for LangChain), suggesting a lower risk of factual inaccuracies in the generated responses.

Overall, for support chatbots that process contractual documents and must ensure both accuracy and contextual completeness, LangChain represents better framework due to its strong performance across the most critical quality metrics. However, in scenarios where processing speed is the dominant factor, LlamaIndex may be a more appropriate choice, particularly for simpler queries or high-throughput environments.

Experiment No. 2

Based on the findings of the first experiment, LangChain was identified as the most optimal framework. Consequently, all subsequent Q&A system implementations were developed using this framework as the foundation.

In the second study, the vector database was selected as the variable parameter. Responses were generated using each of the previously described vector databases, and the resulting outputs were evaluated. Average metric values were calculated and summarized for analysis.

For comparative purposes, additional results obtained using the InMemoryVectorStore were also included. The summarized outcomes of this evaluation are presented in Tab. 4.

Table 4.

Average performance metric values for different vector databases

Database	**Relevancy**	**Faithfulness**	**Context recall**	**Context precision**	**Semantic Similarity**	**Elapsed time**
InMemoryVectorStore	0.96362	0.808404	0.843333	0.788889	0.890264	3.60444
Chroma	0.750281	0.858274	0.7	0.755556	0.876092	3.11018
FAISS	0.965183	0.866667	0.9	0.8	0.890555	2.76104
LanceDB	0.933294	0.79	0.9	0.816667	0.893194	3.00584

Among the evaluated vector databases, FAISS demonstrated the best overall performance. It achieved the highest scores in answer relevancy (0.965183), faithfulness (0.866667), and context recall (0.9), while also providing the fastest response time (2.76104 seconds). These results make FAISS the optimal choice for tasks that require a balance of accuracy, reliability, and computational efficiency.

The InMemoryVectorStore exhibited strong results in answer relevancy (0.96362) and semantic similarity (0.890264), but was less competitive in terms of faithfulness (0.808404) and response time (3.60444 seconds). This makes it less suitable for interactive or real-time applications, though it remains a viable option for small-scale or prototype systems.

LanceDB showed relatively high semantic similarity (0.893194) and context recall (0.9), yet its faithfulness score (0.79) and slightly longer response time (3.00584 seconds) limit its advantage compared to FAISS. Nevertheless, it may be beneficial in use cases where semantic coherence is prioritized over factual accuracy.

In contrast, Chroma produced the lowest results across several metrics, including answer relevancy (0.750281) and context recall (0.7). Despite a moderate response time

 ISSN 1560-8956

(3.11018 seconds), its reduced answer quality and incomplete contextual retrieval make it less suitable for high-precision applications.

In summary, FAISS represents the most effective choice for implementing RAG-based systems due to its superior performance in accuracy, faithfulness, and speed. InMemoryVectorStore may be used for smaller-scale or experimental tasks, while LanceDB is appropriate when semantic similarity is a key factor. Chroma, on the other hand, is better suited for simple or non-critical applications with lower quality requirements.

Experiment No. 3

Based on the findings of the previous experiments, the LangChain framework and the FAISS vector database were identified as the optimal combination for building RAG-based Q&A systems.

In this final experiment, a new Q&A system was implemented using these components, while varying two parameters: chunk size and chunk overlap. Results were obtained for all parameter combinations listed in Table 1, and the average metric values are presented in Tab. 5.

Table 5.

Average performance metric values for different chunk size and overlap combinations

No	Chunk Size	Chunk Overlap	Relevancy	Faithfulness	Context recall	Context precision	Semantic Similarity	Elapsed time
1	500	50	0.951175	0.818182	0.8	0.816667	0.892624	3.59871
2	1000	50	0.950767	0.819972	0.9	0.816667	0.891994	2.96236
3	1000	200	0.950937	0.790417	0.9	0.780556	0.891495	2.95429
4	1500	500	0.951006	0.848333	0.8	0.816667	0.892098	3.8678

Based on the obtained results, the following observations can be made for each configuration:

Chunk size 500 with overlap 50 demonstrates the highest answer relevancy (0.951175) and high context precision (0.816667), but has an average context recall (0.8) and a longer execution time (3.59871 seconds).

Chunk size 1000 with overlap 50 provides the most balanced performance, with the maximum context recall (0.9), consistently high context precision (0.816667) and faithfulness (0.819972), and a moderate execution time (2.96236 seconds).

Chunk size 1000 with overlap 50 achieves the fastest execution time (2.95429 seconds) and the maximum context recall (0.9), but its context precision (0.780556) and faithfulness (0.790417) are lower, which may affect the overall response quality.

Chunk size 1500 with overlap 500 exhibits the highest faithfulness (0.848333) and strong answer relevancy (0.951006), but shows lower context recall (0.8) and the longest execution time (3.8678 seconds).

Among the tests, 1000/50 configuration demonstrates the most optimal trade-off between quality and efficiency. It combines the maximum context recall (0.9), high context

precision (0.816667), and balanced execution time (2.96236 seconds), making it well-suited for complex customer support scenarios.

The 500/50 configuration is preferable when high contextual accuracy and answer relevancy are prioritized over processing speed.

The 1000/200 setup is appropriate for real-time applications requiring high recall and rapid response generation.

In contrast, 1500/500 configuration is recommended for tasks emphasizing response faithfulness, where execution speed and context coverage are less critical.

Conclusion

The study investigated the components of Retrieval-Augmented Generation (RAG) systems for developing an efficient support chatbot capable of processing complex documents. The experimental evaluation revealed the configuration that provides the best trade-off between accuracy, context completeness, and computational efficiency.

We found that the LangChain framework achieves the highest accuracy and context recall, making it well suited for generating detailed and reliable answers. The FAISS library demonstrated superior performance among all tested vector databases in terms of answer relevancy, faithfulness, and context recall, while maintaining the fastest response time. The text-splitting configuration with chunk size 1000 and chunk overlap 50 showed the most balanced results, providing high context precision, maximal recall, and reasonable processing time.

Therefore, the combination of LangChain, FAISS, and the 1000/50 chunking configuration represents the optimal solution for implementing a high-performance RAG-based support chatbot in our setting. This configuration ensures accurate, faithful, and contextually relevant responses, contributing to improved reliability and service quality in real-world applications and can be recommended as a base configuration for support chatbots in different domain.

Future work will investigate the performance of RAG systems in low-resource and multilingual settings [9], with particular attention to the influence of embedding quality, cross-lingual alignment, and retrieval strategies on overall system effectiveness.

REFERENCES

1. *Aslanova V., Oliinyk V.* Psychological support assistant based on fine-tuned LLaMA 3 model // The International Conference on Security, Fault Tolerance, Intelligence ICSFTI2024 (June 07, 2024, Kyiv, Ukraine), page 1-13. URL: https://icsfti-proc.kpi.ua/article/view/309532 (application date: 09.06.2025)

2. *Oliinyk V.* A comparative study of task formulations for detecting propaganda using Large Language Models/ Oliinyk V., Zakharchyn N. // Адаптивні системи автоматичного управління: міжвідомчий науково-технічний збірник. – 2025. – № 2 (47).

3. *Gao Y., Xiong Y., Gao X., et al.* Retrieval-Augmented Generation for Large Language Models: A Survey // arXiv:2312.10997, 2023. DOI: https://doi.org/10.48550/arXiv.2312.10997

4. *Поночовний П.С.* Аналітичний огляд способів застосування великих мовних моделей (LLM) для вирішення прикладних задач / Поночовний П.С., Олійник В.В.// Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2023): матеріали V Міжнародної науково-практичної конференції молодих вчених та студентів, 19-21 грудня 2023 року, м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2023. С. 272-276. URL: https://drive.google.com/file/d/1racc22TBKkFFNzBRSzOrePKpFfbnGDJ1/view (дата звернення: 25.10.2025).

5. *Lewis P., Perez E., Piktus A., et al.* Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // arXiv:2005.11401, 2020. DOI:
https://doi.org/10.48550/arXiv.2005.11401

6. *Afzal A.* Towards Optimizing a Retrieval Augmented Generation using Large Language Model on Academic Data / A. Afzal, J. Vladika, G. Fazlija, A. Staradubets, F. Matthes //. In Proceedings of the 2024 8th International Conference on Natural Language Processing and Information Retrieval (NLPIR '24). – Association for Computing Machinery, New York, NY, USA, 2025. – P. 250–257.

7. *Hladěna J. et al.* The Effect of Chunk Size on the RAG //Software Engineering: Emerging Trends and Practices in System Development: Proceedings of 14th Computer Science Online Conference 2025, Volume 2. – Springer Nature, 2025. – P. 317.

8. *Stäbler M. et al.* The impact of chunking strategies on domain-specific information retrieval in rag systems //2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS). – IEEE, 2025. – P. 1-6.

9. *Oliinyk V.* Low-resource text classification using cross-lingual models for bullying detection in the Ukrainian language / Oliinyk V., Matviichuk I. // Adaptive systems of automatic control, 2023. Vol. 1, №42. – P. 87-100.