## СИСТЕМИ ТЕХНІЧНОГО ЗОРУ І ШТУЧНОГО ІНТЕЛЕКТУ З ОБРОБКОЮ ТА РОЗПІЗНАВАННЯМ ЗОБРАЖЕНЬ

**UDC 004.75**

**S.V. KHRUSCHAK, O.M. TKACHENKO, I.S. KOLESNYK**

# RAG EFFICIENCY IMPROVEMENT FOR BUILDING INTELLECTUAL SCIENTIFIC KNOWNLEDGE DATABASES

***Vinnytsia national agrarian university, 21012, Sonyachna, 3, Vinnytsia, Ukraine***
***e-mail: sergey.khruschak@gmail.com***
***Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: iskolesnykcom@gmail.com***

**Анотація.** В статті розглядається розбробка інтелектуальної бази знань на основі наукових статей з використанням великих мовних моделей в режимі генерації доповненої пошуком. Досліджено різні методи підвищення релевантності вибірки цитованих джерел та згенерованих відповідей мовної моделі та вибір підходів до побудови мовних генеративних систем з врахуванням специфіки наукових матеріалів українською та англійською мовами. Також розглянуто використання різних мовних моделей для генерації відповідей. В процесі дослідження обрано набір критеріїв для комплексного оцінювання генеративних систем та надано рекомендації для побудови наукових інтелектуальних баз знань.
Розроблено інтелектуального агента, який дозволяє проводити пошук та аналізувати наукові статті у зручній інтерактивній формі з забезпеченням цитувань оригінальних документів.
**Ключові слова:** штучний інтелект, база знань, великі мовні моделі, LLM, генерація доповнена пошуком, RAG.
**Abstract.** The article describes the development of an intellectual knowledge base based on scientific articles using large language models in the mode of generation by augmented search. Various methods of increasing the relevance of the sample of cited sources and generated answers of the language model and the choice of approaches to building language generative systems taking into account the specifics of scientific materials in Ukrainian and English are investigated. The use of different language models for generating answers is also considered. In the course of the study, a set of criteria for a comprehensive evaluation of generative systems was selected and recommendations for building scientific intellectual knowledge bases were provided.
An intelligent agent has been developed that allows searching and analyzing scientific articles and providing document citations in a convenient interactive form.
**Key words:** artificial intelligence, large language models, LLM, retrieval augmented generation, RAG, knowledge base.

**DOI: 10.31649/1681-7893-2025-49-1-89-97**

## INTRODUCTION

Retrieval-Augmented Generation (RAG) has emerged as a powerful approach, leveraging Large Language Models (LLMs) with external knowledge retrieval to significantly improve AI's ability to handle complex tasks requiring information and reasoning. By integrating external data, RAG enables generative models to produce more relevant and factually grounded responses with provided citations. That feature is particularly valuable in building robust knowledge bases for scientific research or internal corporate data.

However, real-world implementations of the RAG knowledge bases contain lots of ongoing challenges that can impact their practical application. Among them are management of noisy input data, retrieving of a proper context for the question, processing of multimodal information that contains images, formulas and table data. Finding a proper balance between accurate information retrieval and the adaptability of the generative model remains crucial for the overall performance and reliability of these systems [1].

This paper investigates methods for improving the efficiency of RAG frameworks specifically for the task of building and maintaining scientific knowledge bases, mainly focusing on strategies that enhance retrieval quality and improve the integration of retrieved documents within the generation process. Through systematic experimentation and analysis, the study aims to provide insights and practical recommendations for designing RAG-enhanced knowledge management systems that are more scalable, accurate, and efficient.

The purpose of the article is to increase the relevance of the results of large language models when used in the retrieval augmented generation mode for multilanguage scientific articles. To achieve this goal, we need to solve the following tasks:

– to select a set of criteria to be used for evaluating the relevance of the results of large language model output;
– to improve the architecture of an RAG intelligent system adding full content retrieval of the related articles and re-ranking of the content;
– analyze impact of using different language models for the answer generation.

## ANALYSIS OF RAG-BASED INTELLIGENT SYSTEMS

Large Language Models (LLMs) are a class of advanced neural network architectures, typically characterized by their substantial depth and extensive parameter counts and usually trained on a vast corpora of unlabeled textual data. This large-scale pre-training enables prominent LLMs, including GPT-series, PaLM, LLaMA, and Claude, among others, to develop a sophisticated understanding of linguistic patterns, semantic relationships, and contextual nuances of human texts. Core functional applications of LLMs encompass text and source code generation, the engineering of intelligent conversational agents, the description of visual information, and the detection of anomalies within complex datasets. The versatility of these models has fostered their widespread integration across various applied domains, including the development of adaptive educational platforms, the creation of sophisticated financial instruments, and their utility as analytical assistants in scientific research, particularly for tasks like literature review and data interpretation [2].

Despite their significant potential and demonstrated efficiency, LLMs possess inherent limitations that necessitate careful considerations of their usage. A primary constraint is the substantial computational expenditure required for their training and inference. Furthermore, practical implementations often impose restrictions on the input query length (token limits), which can curtail the complexity of prompts or the volume of contextual information provided [3]. A notable challenge arises when LLMs encounter queries related to highly specialized or niche domains underrepresented in their initial training corpora. This can lead to a substantial degradation in output accuracy and relevance. Models may also exhibit a tendency towards overgeneralization, resulting in responses that, while fluent and looks plausible, lack precision or contain many terminological inaccuracies. Another aspect is that the knowledge encoded within LLMs is inherently static and reflects the temporal horizon of their last training dataset, leading to potential obsolescence as new information emerges, while the cost and time intensity of retraining impede frequent updates. A critical and widely discussed issue is the phenomenon of "hallucinations," wherein the model generates plausible sounding, but totally incorrect or unsubstantiated information, likely stemming from the model's internal mechanisms for pattern completion rather than genuine comprehension or data retrieval [4]. Additionally, LLMs typically lack the capability to provide accurate citations or attribute information to specific sources, which makes them harder to use in scientific and academic contexts where verifiable provenance is paramount.

To address these limitations and enhance the reliability and accuracy of LLM outputs, various techniques are actively used. Those include sophisticated prompt engineering strategies designed to guide the model's focus and constrain its output space, targeted fine-tuning of pre-trained models on domain-specific datasets to saturate model with specialized knowledge, and the integration of LLMs with external knowledge bases or live data retrieval systems. Among these, architectural approaches centered on retrieval augmented generation (RAG) have emerged as particularly effective. This method combines the information retrieval capabilities of dedicated search systems with the generative capabilities of LLMs. By grounding the generation process in relevant, retrieved documents, this method aims to substantially reduce the tendency for hallucinations, improve the factual accuracy of responses, and enhance the overall trustworthiness of LLM-generated content by providing information sources, thereby mitigating some of the most pressing challenges associated with these powerful models [5].

The basic idea of RAG is that before synthesizing a response it involves a preparatory information retrieval step from an external knowledge repository (a vector database or indexed document store). The task of the RAG system is to identify and retrieve information chunks that are most relevant to the input query and the current context. This retrieved contextual data is subsequently provided as an input to the LLM, which then instructed to formulate an answer explicitly grounded in this supplementary information. This methodology

significantly mitigates the occurrence of "hallucinations" or the generation of unsubstantiated content, thereby enhancing the factual accuracy and depth of the output. Moreover, it inherently supports the capacity to provide citations and references to the source documents, thereby enabling verification of the generated results [6].

There are various implementations of RAG systems, but most usually they consist of following elements: a storage preparation and vectorization unit, a retrieval unit that uses vector similarity search or hybrid methods that combine semantic and keyword search to find relevant information to a question in the store, and a context generation unit where the received pieces of information are selected, prepared and transferred to a language model for generating an answer.

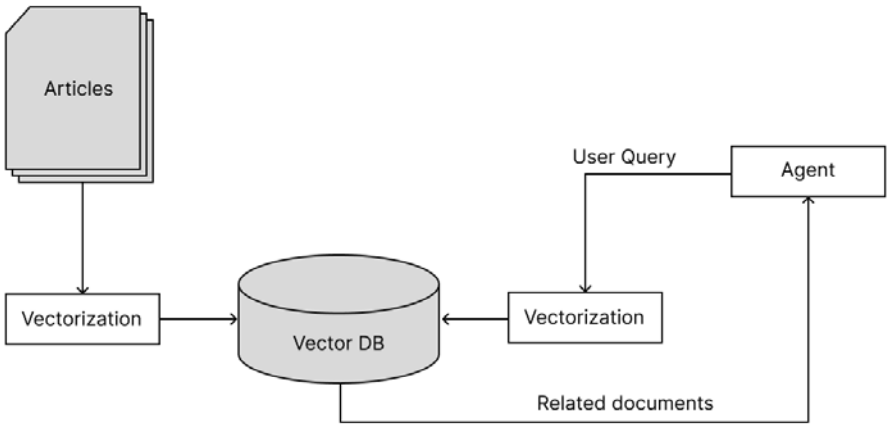


Figure 1 – Basic RAG system diagram

For storing the vectorized documents usually any of specialized vector databases like Chroma, QDrant or Pinecore are used [7]. Also many modern relational and NoSQL databases actively adding vector operations and storage support, among them PostgreSQL, MongoDB and others. These databases are used to store not only the primary data, but also metadata and document's corresponding vector representations, commonly referred to as embeddings. These embeddings, generated by distinct machine learning models, encapsulate the semantic essence of the source data. The utilization of such vector representations facilitates efficient similarity searches, enabling the rapid identification of records that are semantically close to a given query. A common preprocessing step involves the segmentation of text documents into smaller, often overlapping, chunks prior to their ingestion into the vector database. This chunking strategy aims to enhance the granularity of the retrieval process, thereby increasing the likelihood of surfacing the most relevant information segments in response to a query. Subsequently, during the retrieval phase, when records semantically relative to the user's query are identified, a standard practice is to select the top-k results. These selected segments are then integrated into the context window of a large language model to narrow its response generation.

However, despite the considerable promise of Retrieval Augmented Generation (RAG) systems to enhance the reliability and contextual relevance of generated text, their practical implementation is frequently accompanied by several challenges. Among these, the retrieval process often finds non-relevant or sub-optimal text fragments from the vector stores. Also, many conventional RAG architectures unable to process non-textual information, such as charts, diagrams, and figures, which are often crucial for comprehensive understanding [6]. These limitations are especially acute when working with scientific information, which usually have a complex structure, rich terminology and frequently rely on visual elements.

This research addresses the development and systematic evaluation of an intelligent knowledge base, based on a RAG framework, specifically considering interactions with the academical data. The input dataset for this study comprises 105 scientific technical articles published. The articles come in multiple languages, rich with visual information and math. The proposed system incorporates several enhancements to improve retrieval and generation phases of the system, including the integration of computer vision techniques for the extraction and interpretation of graphical materials, a preliminary document categorization and re-ranking. These integrated approaches are designed to improve the relevance and correctness of responses generated by large language models when answering field related questions. The efficacy of the developed system was assessed at each stage of its development and implementation.

## EVALUATION OF THE DEVELOPED SYSTEM

Before doing any changes to the system, it is important part to choose the criteria and methods used to evaluate them. The main criterion in this paper is the relevance and completeness of the model's answers to questions related to scientific articles in the database.

First let's consider the retrieval step, the primary objective of it is to ensure that the documents or fragments selected are semantically aligned with the query intent. To quantify retrieval performance, the following similarity metrics were employed: for queries with known relevant documents (established via ground-truth datasets), we calculated the recall at top-k retrieved results, the metric then assessed how often the correct or most relevant documents were successfully retrieved within the top k candidates. This metric is called Mean Reciprocal Rank (MRR) and usually is used to evaluate the ranking quality by considering the position of the first relevant retrieved item. A higher MRR score suggested that relevant documents appeared earlier in the retrieval list, minimizing the search burden on the generation model. Also to ensure that categorization enhanced retrieval precision, we measured the consistency between the predicted category of the query and the categories of retrieved documents. High categorization consistency correlated with improved retrieval quality and generation relevance [8].

For automated performance evaluation of the language model answer generation, two classes of metrics are usually used: metrics based on ground truth, which evaluate the similarity of the generated system response to the reference answer, and metrics without ground truth (zero-reference), when there is no reference answer and the relevance, logic, or factuality of the response is checked.

One of the popular metrics that is used to compare the semantic similarity of the model's response with the ground truth is embedding based metric. Embedding models are trained on a very large corpus of text itself, so their similarity results quite well alight with human perception. In our case a cosine distance between the calculated embedding vectors, obtained from OpenAI service was used. The formula for the reference ($V_e$) and the generated ($V_g$) answer vectors in this case is:

$$Sim(V_e, V_g) = \frac{\overrightarrow{V_e} \cdot \overrightarrow{V_g}}{\|V_e\| \cdot \|V_g\|} \quad (1)$$

Additionally, we used LLM evaluation metrics, in which the model receives both texts as input and is tasked with evaluating their similarity based on various criteria: relevance, truthfulness, style and completeness. To cover evaluation methods without a basic truth, the language model perplexity scores (MP-PPL) were used. This value indicates of how confidently the model is able to predict the sequence of words, the higher the confusion, the less confident the model is in predicting the observed sequence. It is computed by the formula:

$$Ppl = \exp\left(-\sum_{i=1}^{t} \log\left(P_\theta\left(x_i | x_j \neq x_i\right)\right)\right), \quad (2)$$

where $P_\theta$ is the logarithm of the probability of the *i*-th output token of the model, provided that other tokens appear in the sentence [8]. And the last approach that was used for model evaluation is to use another LLM to evaluate the completeness and relevance of the answer given only question and the answer [9].

To obtain a ground truth answers for evaluation a random set of articles was picked and 60 questions were manually generated on the material, both specific to a single article and covering information from several articles and graphical information. Since the volume of articles and the number of meta-parameters and components are quite significant, it was important to automate the testing process so it can be easily run on every iteration. Therefore, the proposed evaluation methods were implemented in code and integrated with the LangGraph service, that provides tracing and evaluation capabilities.

## DEVELOPMENT OF A MULTIMODAL RAG SYSTEM WITH RE-RANKING

The intelligent system can be divided into two main, relatively independent modules: uploading and processing. The module for parsing and uploading articles to the vector storage is triggered only when documents are initially uploaded or updated. For improving that part it is proposed to add images and diagrams recognition to the ingestion step with additional pre-processing needed for later steps: extracting keywords, short description, and category of the article. The processing phase in turn can be split into two major phases: context retrieval and answer generation, it is possible to improve and measure them separately as well by adding categorization and re-ranking. The proposed updated architecture is presented on the Figure 2.

Let us consider the main elements of the subsystem for processing and uploading documents to the system. Before being uploaded to the vector store, documents undergo preliminary processing, where parts that are not directly related to the article are removed: title pages, information about the journal, table of contents, etc. After that, they are divided into separate categories; if there are keywords present in the article they are extracted, otherwise the category is determined using the GPT-4o mini model with instructions for determining categories based on the text of the article. The resulting categories, keywords, and links to document related to the same article are added as metadata to the corresponding records in the vector repository and are further used to improve search relevance.

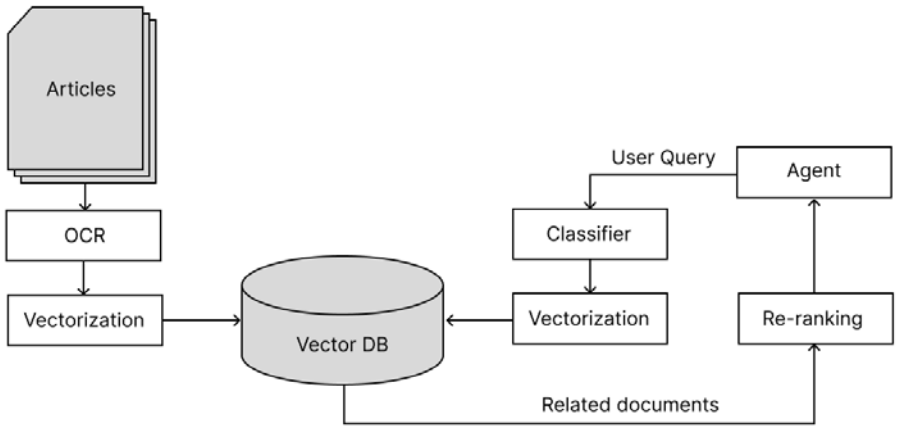


Figure 2 – Proposed RAG system architecture diagram

Categorization plays a central role in improving retrieval efficiency and generation relevance as it enables segmented retrieval, where queries are first routed to the most semantically appropriate category before similarity search is performed. By narrowing the search space, categorization can significantly reduce retrieval noise and computational cost, while simultaneously improving the precision of the retrieved contexts.

After categorized retrieval of the related documents it is proposed to re-rank them based on keywords, which are usually provided in the articles or can be obtained using a separate LLM pre-processing step. For implementing re-ranking step a BM25 method is used [10]. As an input it gets documents and keywords retrieved from the vector storage along with other parts related to the same articles and evaluates the retrieved passages based on term frequency and document length normalization to assign more accurate relevance scores. Additionally, it is proposed to store generalized descriptions generated by the LLM during articles loading step, with links to original documents separately, which allows better contextualization of answers to general questions on the topic, that needs to fit multiple articles into the LLM context window. The resulting summarized information about all articles is added separately to the vector storage and linked with other information from the document using metadata. As an optimization, the summarization, categorization and keywords retrieval blocks are combined into a single LLM processing step, which allows getting all values in one pass of the model.

As the primary focus of the developed system is the processing of scientific texts, the proposed architecture adds image and diagram recognition and their conversion into textual descriptions [11]. This can be done only once per document in the pre-processing step using any language model with image recognition support, our tests shown that gpt-4o multimodal model gives the best results for the task. The resulting textual description of the image data then added to the main text of the article. The last step before uploading documents is to split the texts of articles into separate overlapping blocks, convert them to vector representation, and upload them to a vector database. At the same time, all previously obtained information is added to the metadata of the documents, which allows the system to select other parts of the document based on a single found record.

During the main operation of the system, the incoming question goes through a chain of processors that fill it with additional information. First, the previously stored history of interactions with the agent is retrieved, which improves understanding of the question context by the system. Since the context of language models is limited, and older history is less important than relevant documents, the received history is truncated to 1000 tokens. Further, the question and interaction history is classified using the list of categories determined during the upload phase, that allows to narrow down the search and thus increase the relevance of the results found. If the categories cannot be determined, the search will be performed on the entire articles dataset. The question

category is determined using the gpt-40-mini model, even though Gemini model was tested as well, receiving similar results. The next step is to search for relevant documents in the vector repository. The search for documents is performed by finding the documents that have the smallest cosine distance between the embedding vectors to the question embedding.

For testing the proposed RAG-system a modular prototype using widely adopted tools and frameworks was implemented: a set of Jupyter notebooks were used for development and experimentation, LangChain framework for pipeline orchestration, FAISS was used as the local vector storage for context retrieval, and LangSmith for detailed tracing and observability. A total of 107 articles in PDF format were downloaded from the journals of Vinnytsia National Technical University and processed by the system. Many articles contained additional information about the journal, authors positions, etc. so they were passed through the pre-processing step to clean up all the irrelevant information. The LangChain library provides a set of tools, like PyPDFLoader, which were used to process and vectorize the PDF documents page by page, or LLMImageBlobParser that allowed to plug in different multimodal LLM models for image recognition during pre-processing step. Since language models produce data in an arbitrary format, to ensure a predictable structured result, OpenAI's integration with Pydantic and LangChain tools was used, which allowed to parse the model's output and return it in the form of a Python object of the required structure. When processing queries, the MemorySaver class, provided by the LangChain library, was used to save and load the message history.

After that, the texts are classified and divided into smaller fragments of 1000 tokens with an overlap of 200 tokens between fragments. Each text fragment is then converted into a vector in a multidimensional space using the OpenAIEmbeddings embedding model. The dimension of vectors for this model is 1536. These vectors describe the content of the fragments and allowed to calculate the similarity between the user's query and the parts of the corpus. After that, the obtained vector representations of articles and summarized information along with metadata were added to the FAISS index. The article metadata included file names for providing links to sources, defined article categories, keywords and neighbouring text fragments. The query category is determined based on both the current question and the historical context, for which the gpt-4o model with a set of instructions for determining the category is used. All new information received is added to the main question at each stage of its passage through the processing chain. To facilitate system testing, an interactive agent built into Jupyter was created using the Gradio component library, which allows testing the system with different handler settings and display intermediate search results. The use of interactive chat allowed customizing the meta-parameters of document sampling and classification and test the system's performance on the go.

As a result, the developed system allows analyzing uploaded scientific articles and formulating generalized conclusions about individual articles or topics in general, providing citations for generated answers. The architecture is designed in such a way that most components can be replaced or excluded from the system, which allows for a better assessment of the impact of each change on the overall system performance.

The LangSmith framework was used as a framework for testing and monitoring experiments. This framework allows to obtain detailed execution traces, capturing every step of the retrieval and generation process from query classification to retrieval selection and final generation, as well as each pipeline component's inputs and outputs. The framework also allows to capture retrieval times, generation delays, and categorization accuracy enabling more precise tuning of the specific parts of the system. The ground truth questions were uploaded as a dataset to LangSmith and used to first evaluate the retrieval phase of the system using MRR metrics, the results of the evaluation are shown in Figure 3.

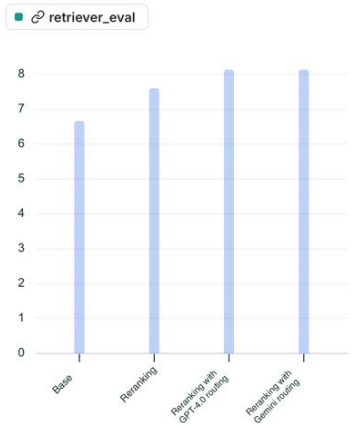


Figure 3 – Retrieval step evaluation results using MRR

MRR evaluation results show how good on average system in finding the correct chunk of text related to the ground truth questions. Each test was repeated at least 3 times and results represents the average value. For the base case of retrieval that obtains top 3 related documents from the vector storage the result is 6.64 points on average. Adding retrieval of the other parts of the same article with re-ranking step allows to improve the result to 7.60 and addition of the classification block with routing allowed to improve the result further to 8.13. The classification step is performed by the LLM, but the task is very simple, so using both GPT4.0 and Gemini for the classification shown similar results on average.

The next step was evaluating the final results of the LLM answers generation on different improvement stages and using different LLMs. The evaluation results are presented on the Fig. 4. The paper considers only the best average results after hyperparameters and prompts tuning using two different LLMs: GPT4.0 and Gemini. All of these metrics operate on the principle that the higher the value, the better the response quality.

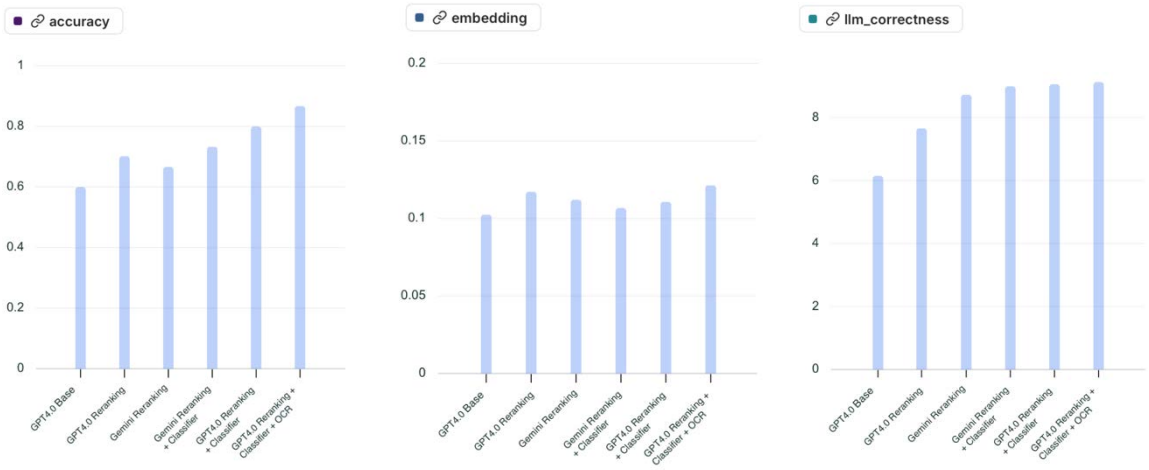


Figure 4 – System evaluation results using different metrics

The major variants that were considered on the graph are: base model – the basic RAG architecture based on GPT-4.0 model, model with added articles re-ranking, model with re-ranking and categorization and the final variant that adds images OCR data to the vector storage. The metrics that are displayed are: accuracy – comparison of the reference and the generated answer using another LLM, embeddings distance – the average value of the cosine distance between the embedding vectors for the model response and the reference response, and an assessment of the answer by the other LLM model (GPT 4.0), with instructions provided to check the relevance and accuracy of the answer without specifying a reference answer. The results are also presented in the Table 1.

**Table 1**

**Model evaluation results**

| Metrics | GPT4.0 base model | GPT4.0 with re-ranking | Gemini with re-ranking | GPT4.0 with re-ranking and classifier | Gemini with re-ranking and classifier | Final GPT4.0 model with added image recognition |
|---|---|---|---|---|---|---|
| Accuracy | 0,6 | 0,70 | 0,67 | 0,73 | 0,80 | 0,87 |
| Embedding | 0,105 | 0,134 | 0,124 | 0,113 | 0,120 | 0,143 |
| LLM correctness | 6,15 | 7,67 | 8,53 | 8,87 | 9,00 | 9,11 |

As can be seen from the testing results, each architecture improvement has a positive impact on system accuracy, with re-ranking being the most impactful. On the other hand, image recognition feature adds quite little on top of the existing changes, which also can be caused by the limited number of ground truth questions that rely solely on the data from images. As for the models GPT4.0 shows slightly better results in most metrics,

compared to Gemini, so it was chosen for the final system variant. At the same time, the request processing time increased from 3,9 for the base system to 7,4 seconds, which is acceptable for systems of this type. Thus, the architecture is effective from a practical point of view and demonstrates the relevance of responses and stability in various usage scenarios.

## CONCLUSIONS

In this article, an improved architecture of an intelligent system based on retrieval augmented generation, focused on working with scientific information in both Ukrainian and English, was developed and implemented. The system integrates the capabilities of large language models with mechanisms for categorization, generalization and processing of graphic information of loaded documents. A set of metrics was analyzed and selected, by which the system was evaluated, and a series of experiments were conducted for a comprehensive assessment of each introduced architectural change. The results demonstrate an increase in the relevance of responses compared to basic generation approach. At the same time, each new element of the system provides computational complexity and time for processing system requests.

Therefore, the proposed system can be used as a basis for creating intelligent knowledge bases in education institutions, as well as for the further development of language generative systems adapted to the specifics of scientific texts in Ukrainian and English.

One of the directions of further research can be the use of special models for working with scientific articles, such as, for example, MathCoder, instead of general-purpose language models. Using external re-reanking services that allows to further improve the context used for answer generation. As well as using the MCP protocol to integrate external knowledge systems and utilities.

## REFERENCES

1. Andriopoulos, K. and Johan, P. (2023). Augmenting LLMs with knowledge: a survey on hallucination prevention. arXiv. URL: https://doi.org/10.48550/arXiv.2309.16459.
2. Wang, Chenguang, Mu Li, and Alexander J. Smola. Language models with transformers. arXiv preprint arXiv:1904.09408 (2019). URL: https://doi.org/10.48550/arXiv.1904.09408.
3. Jeong, Cheonsu (2023). A study on the implementation of generative AI services using an enterprise data-based LLM application architecture. arXiv. URL: https://doi.org/10.48550/arXiv.2309.
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P. and Neelakantan, A. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems 33, 1877–1901. DOI: https://doi.org/10.48550/arXiv.2005.14165.
5. Cui, J., Li, Z., Yan, Y., Chen, B. and Yuan, L. (2023). ChatLaw: open-source legal large language model with integrated external knowledge bases. DOI: https://doi.org/10.48550/arXiv.2306.16092.
6. Tolga Şakar, Hakan Emekci, Maximizing RAG efficiency: A comparative analysis of RAG methods / Natural Language Processing, Cambridge, Vol. 31, Issue 1, 2025, pp. 1-25.
7. Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges / Cognitive Systems Research, Vol. 85, 2024. DOI: https://doi.org/10.1016/j.cogsys.2024.101216.
8. Roucher A. RAG Evaluation. URL: https://huggingface.co/learn/cookbook/rag_evaluation.
9. Herreros Q., Veasey T., Papaoikonomou T. RAG evaluation metrics: A journey through metrics. URL: https://www.elastic.co/search-labs/blog/evaluating-rag-metrics.
10. Trotman A., Puurula A., Burgess B.. Improvements to BM25 and Language Models Examined. In Proc., 19th Australasian Document Computing Symp., ADCS ’14, 58–65. New York: Association for Computing Machinery. https://doi.org/10.1145/2682862.2682863.
11. Romanyuk, O., Zavalniuk, Y., Pavlov, **S.,** etc. New surface reflectance model with the combination of two cubic functions usage, *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska*, 2023, 13(3), pp. 101–106
12. Chen, W., Chen, J., Zou, F., Li, Y.-F., Lu, P. and Zhao, W. (2019). RobustiQ: a robust ANN search method for billion-scale similarity search on GPUs. Proceedings of the 2019 International Conference On Multimedia Retrieval (pp. 132–140). URL: https://doi.org/10.1145/3323873.3325018.

*Надійшла до редакції 24.03.2025 р.*

**KHRUSCHAK SERGII** – Ph. D., Senior Teacher of Computer Sciences and Digital Economics Department, Vinnytsia national agrarian university, Vinnytsia, Ukraine, ***e-mail: serqey.khruschak@qmail.com***

**TKACHENKO OLEXANDR** – Ph. D., assistant professor of Software Department, Vinnytsia National Technical University, Vinnytsia, Ukraine, ***e-mail: alextk1960@gmail.com***

**KOLESNYK IRYNA** – Ph. D., assistant professor of Computing Technology Department, Vinnytsia National Technical University, Vinnytsia, Ukraine, ***e-mail: iskolesnykcom@gmail.com***

С.В. ХРУЩАК, О.М. ТКАЧЕНКО, І.С. КОЛЕСНИК

**ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ RAG ДЛЯ ПОБУДОВИ НАУКОВИХ ІНТЕЛЕКТУАЛЬНИХ БАЗ ЗНАНЬ**

Вінницький національний аграрний університет, Сонячна 3, м. Вінниця, Україна
Вінницький національний технічний університет, Україна