ISSN 2007-9737

Large Language Models for Accelerating Scientific Research Work in Industry: RAG and Agents

Osvaldo Velazquez-Gonzalez*

Quantori, Inc., AI & Machine Learning Team, Massachusetts, USA

osvaldodvego@gmail.com

Abstract. Large Language Models (LLMs) are changing the way scientific work gets planned, carried out, and communicated, especially in fields like life sciences, drug development, and pharmaceuticals. This paper focuses on what is actually working in practice: how teams are using LLMs to navigate scientific literature, pull relevant information from clinical or research text, and support decision-making in drug discovery. We break down the core pieces behind these systems, things like retrieval-augmented generation (RAG), hybrid vector search, agentic LLM orchestration, and discuss how to measure the retrieval process and entire system performance. We also explore some of the newer directions in this space, particularly agentic RAGs, where LLMs do not just answer questions but actively decide what steps to take, having at their disposal tools that they can run when needed. This includes retrieving papers from sources like PubMed, arXiv, bioRxiv, or Google Scholar, querying structured data, and chaining together reasoning steps to dig deeper into a problem.

Keywords. Large Language Models (LLM), Retrieval Augmented Generation (RAG), scientific research, AI assistant research, agentic RAGs, LLM agents.

1 Introduction

Large Language Models (LLMs) have quickly become some of the most useful tools in biomedical and life science research. Trained on enormous amounts of text, these models can generate text content and provide meaningful answers to complex questions, which makes them an interesting tool for tasks like drug discovery, preclinical research, clinical data analysis, and medical decision support. Researchers across drug development and healthcare are already finding practical ways to put them to work: scanning through scientific literature, forming new hypotheses, helping with medical documentation, research, and even helping design or repurpose therapeutic molecules.

This paper looks at how LLMs are actually being used today in life sciences, with particular attention to retrieval augmented generation (RAG) and agentic techniques that enhance model outputs in domain-specific knowledge rather than relying only on the model's knowledge. We walk through common usage patterns and practical lessons learned, highlighting how pairing LLMs with retrieval methods helps tackle real problems such as factual errors and context limitations. The scope runs from computational drug discovery pipelines all the way to clinical workflow support. Along the way, we offer an overview of what LLMs can actually do in scientific settings, explain how RAG frameworks work in practice, and review a set of recent applied studies that put these ideas to the test.

The paper is organized as follows. Section 2 covers related work, including a look at RAG architectures and how LLMs are being used as AI assistants across different domains. Section 3 details how LLMs function as generative models, how they support scientific research, and how they are used in RAG approaches, along with current limitations and ways to improve retrieval quality. Section 4 focuses on how to evaluate RAG systems. Section 5 explores novel agentic RAG approaches. Section 6 summarizes real-world

applications across several industries, and finally, Section 7 wraps things up with conclusions and a discussion of limitations.

2 Related Works

The use of large language models in scientific research has grown quickly, and two directions in particular: systems built around retrieval-augmented generation, and approaches that give models more agency to act on their own.

2.1 LLMs in Biomedical and Life Sciences

One of the clearest lessons from recent years is that training models on data from a specific field pays off. BioBERT [6] showed this well: by continuing to train on biomedical text, it got noticeably better at things like recognizing medical terms and understanding how concepts relate to each other. Beyond specialized models, general-purpose LLMs are also becoming common in research settings, usually connected to some form of retrieval so their answers stay tied to real, up-to-date sources.

2.2 Retrieval-Augmented Generation Systems

RAG is a fairly straightforward idea: instead of relying entirely on what a model has memorized, you give it access to external documents at the time it generates a response [8]. This makes outputs more trustworthy and easier to verify, which matters a lot in healthcare [20]. CLEAR [10] is a good example of this in action, using entity-level retrieval to get more accurate results when extracting clinical information from text.

2.3 LLM Agents and Tool Use

Agentic approaches go one step further. Rather than just retrieving and generating, these systems let models reason through a problem step by step and call external tools along the way. ReAct [21] was one of the first frameworks to show how useful this could be, mixing reasoning and action in a single loop. Since then, tools like LangGraph have made it much easier to build these kinds of workflows without starting from scratch [19, 3].

2.4 AI in Drug Discovery and Development

LLMs are being tested across many parts of the drug discovery process [7], from reading and summarizing scientific literature to helping researchers think through hypotheses and make decisions. Some recent projects have taken a fully agentic approach, where multiple models collaborate on discovery tasks [1]. Others have focused on connecting RAG pipelines to multi-omics data, opening the door to more personalized and data-driven medicine.

3 LLMs and RAGs

This section summarizes the core RAG pipeline used in scientific research work: pre-processing, retrieval, and generation.

3.1 Pre-Processing

The main goal of pre-processing is to take scientific content that comes in all shapes and sizes and turn it into a consistent enough content to retrieve and work with reliably. In practice this means handling whatever format the source material comes in: PDFs, Word documents, PowerPoint files, spreadsheets, Google Docs, plain text exports, and more. Everything gets funneled into a single text-based pipeline.

Once the content is in plain text, the next step is to split it into smaller pieces called chunks. These chunks are kept small enough to be focused but overlap slightly with their neighbors, usually around 10 to 20 percent, so that ideas stretching across two chunks do not get cut off and lost.

This approach has been a standard practice in question-answering systems for a while [5], and it also helps work around one of the known weaknesses of LLMs: their tendency to lose track of information buried in the middle of long inputs [9].

Each chunk then gets passed through an embedding model, which converts the text into a numerical vector that captures its meaning. These vectors are stored in a vector database alongside metadata like the title, authors, section, and date. Popular open-source options for generating these

ISSN 2007-9737

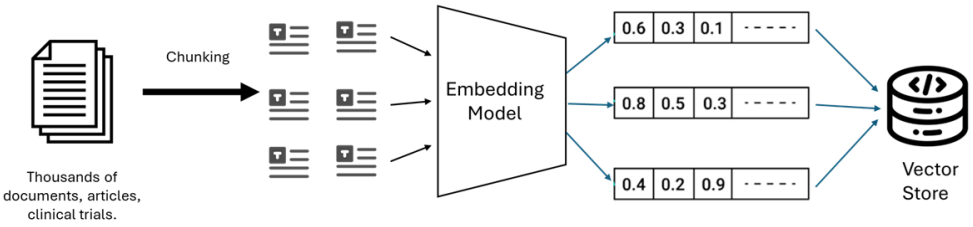

Fig. 1. Pre-processing pipeline for RAG systems. Documents are chunked, embedded, and stored in a vector database for efficient retrieval

embeddings include Sentence-BERT [17], while commercial alternatives from OpenAI are also widely used [14].

On top of this, the same chunks are also indexed using a keyword-based method like BM25 [18], which works by matching exact terms rather than meaning. The reason for doing both is that dense and sparse retrieval tend to complement each other, and combining them often leads to better results overall [11]. In life sciences and drug development especially, this matters a lot because a model needs to handle both the meaning of a passage and precise terminology, things like drug names, gene symbols, or clinical trial identifiers.

3.2 Retrieval

This step happens after a user submits a query into the system; the system does not just take it at raw text. If there is a previous conversation history, the query gets rewritten first to reflect that context, and then it goes through the same embedding model that was used to create the vector store in the pre-processing stage. This produces a vector that captures what the user is actually looking for.

From there, the retriever compares that query vector against every vector stored in the vector store and selects the top-k chunks that are closest in meaning (Fig. 2).

The standard way to measure that closeness is cosine similarity, which looks at the angle between two vectors rather than their raw distance, making it a reliable way to compare meaning regardless of how long the text is.

At the same time, a keyword-based retriever runs in parallel, scoring the same chunks using a method like BM25. A practical way to combine

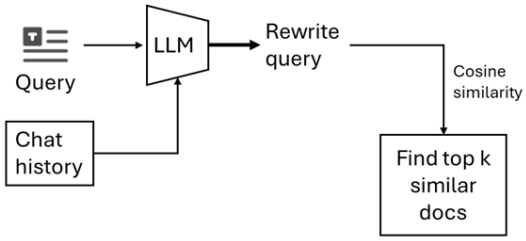

Fig. 2. Retrieval phase of RAG. Queries are embedded (and optionally rewritten), then matched against stored embeddings using cosine similarity to select the top-k relevant chunks

dense and sparse retrieval is a weighted fusion score:

$$\text{similarity}(q,d) = \frac{q \cdot d}{\|q\|\|d\|}, \tag{1}$$

$$\begin{aligned}\text{score}(q,d) = \alpha\, &\text{similarity}(q,d) \\ &+ (1-\alpha)\,\text{BM25}(q,d),\end{aligned} \tag{2}$$

where $\alpha \in [0,1]$.

This explicit fusion rule lets the system match on meaning and on exact terms, which is especially useful when queries contain specific names, identifiers, or technical vocabulary.

To make cosine similarity a bit more concrete, each piece of text is represented as an arrow pointing in some direction in a high-dimensional space. Fig. 3 reduces this to two dimensions so it is easier to see. The query is one arrow, and each candidate document chunk is another. The retriever simply looks at how closely aligned those arrows are. The smaller the angle between the query and a chunk, the more similar they are in meaning, and the higher that chunk gets ranked.

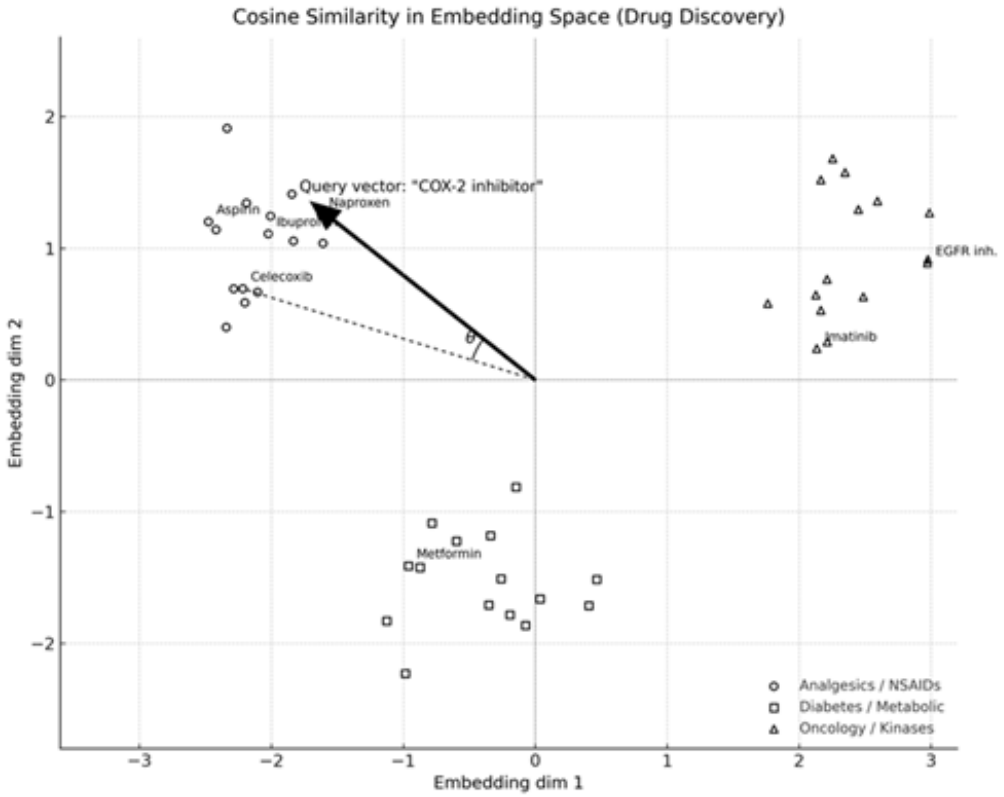

Fig. 3. Cosine similarity in embedding space (illustrative example). The retriever scores candidates by the normalized dot product with the query vector, which corresponds to the angle between vectors

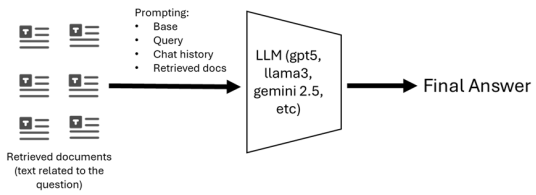

Fig. 4. Generation phase of RAG. Retrieved passages are incorporated into the prompt to produce grounded outputs

3.3 Generation

Once the relevant chunks have been retrieved, they are merged with a base prompt that sets the tone and rules, the original user query, any relevant conversation history, and the retrieved documents (Fig. 4). The LLM then uses all of that context to generate a meaningful final response.

4 RAG Evaluation

Evaluating RAG system performance properly means looking at two things. How well it retrieves relevant information, and how well it turns that information into a good answer. RAGAs [4] addresses this by automating the evaluation process using LLM-based metrics, which reduces the need for large amounts of manually labeled data.

The process itself is straightforward, as illustrated in Fig. 5. Questions are generated directly from the vector store to form a synthetic test dataset that reflects the content the system is actually working with. Those questions go through the full RAG pipeline and the outputs get scored. The scores give you a concrete sense of whether retrieval is doing its job and whether the

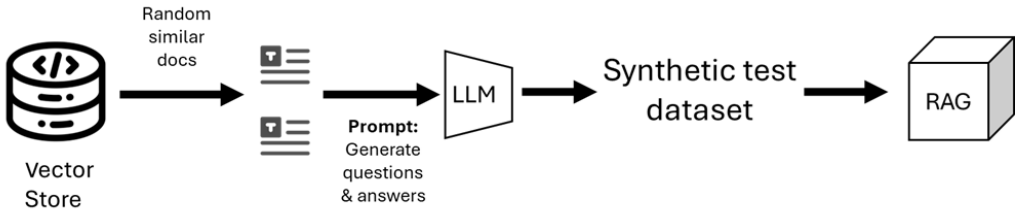

Fig. 5. RAGAs evaluation framework. Synthetic datasets can be generated from the vector store, run through the RAG system, and scored with LLM-based metrics

generated answers are genuinely backed by the retrieved content.

4.1 Core Metrics

The RAGAs framework [4] defines four core metrics for evaluating RAG systems.

Before getting into the metrics, it helps to define the key pieces. K represents how many chunks were retrieved, and for each chunk at rank k, a binary indicator is used to record whether it was actually relevant $R_k \in \{0, 1\}$.

On the generation side, the claims in the reference answer C_{ref} and the claims in the model's response C_{resp} are treated as two separate sets. The analysis then checks which claims are supported by the retrieved content; that supported-claims set is denoted by S. For measuring answer relevancy, the embedding of the original question $\mathbf{q}^{\text{orig}}$ is compared against the embeddings of N synthetic questions $\{\mathbf{q}_i^{\text{synth}}\}_{i=1}^{N}$ generated by the model, and that comparison is used to assess how well the answer stays on topic:

$$\text{ContextPrecision@}K = \frac{\sum_{k=1}^{K} p_k \, R_k}{\max\left(1, \sum_{k=1}^{K} R_k\right)}, \quad (3)$$

$$\text{ContextRecall} = \frac{|C_{\text{ref}} \cap S|}{|C_{\text{ref}}|}, \quad (4)$$

$$\text{Faithfulness} = \frac{|C_{\text{resp}} \cap S|}{|C_{\text{resp}}|}, \quad (5)$$

$$\text{AnswerRelevancy} = \frac{1}{N} \sum_{i=1}^{N} \cos(\mathbf{q}_i^{\text{synth}}, \mathbf{q}^{\text{orig}}), \quad (6)$$

where $p_k = \text{Precision@}k = |\{j : j \leq k \wedge R_j = 1\}|/k$ is the standard precision at rank k.

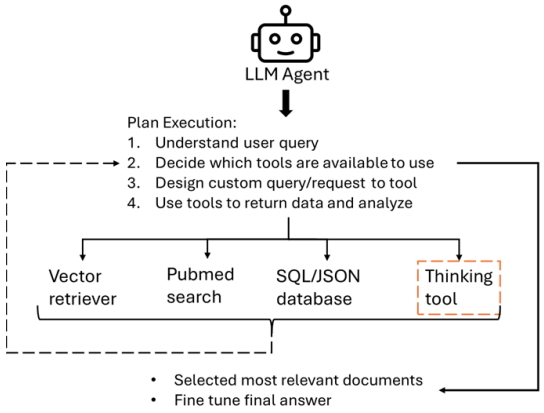

Fig. 6. Agentic RAG with tool integration. The agent selects tools (e.g., vector retrieval, PubMed search, SQL/JSON access) and synthesizes results into a final answer

5 Agentic RAGs

Agentic RAG extends the retrieve-then-generate process by letting an LLM plan, call tools, and iterate over intermediate results in order to provide more detailed information in the final answer.

In practice, the agent does not rely on a single source of information. It has a set of tools it can call on depending on what the question actually needs, as shown in Fig. 6. A vector retriever handles the semantic side, pulling passages from the indexed document store that are closest in meaning to the query including hybrid search approaches. When the question calls for biomedical literature that goes

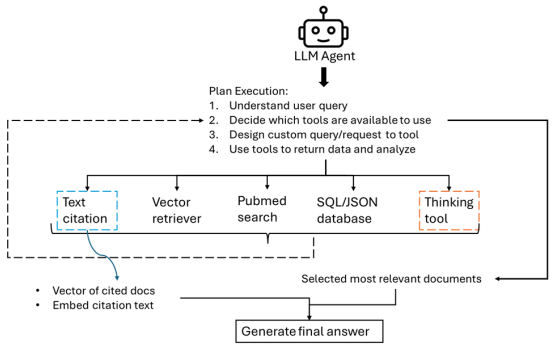

Fig. 7. Agentic RAG execution loop (illustrative). The agent iterates over planning, tool calls, and observation updates before producing a final grounded answer

beyond what is stored locally in the vector store, the agent can reach out to PubMed directly. And when the answer requires structured data, SQL and JSON database tools let it query clinical tables, generating the SQL query statement to retrieve the data, knowledge bases, or experiment records in a more precise way.

There is also a dedicated reasoning step, sometimes called a thinking tool, that gives the agent space to work through a problem more carefully before committing to an answer. This tends to improve response quality, and the outputs from this step can be evaluated using standard RAGAs metrics described in Section 4 (and the comparison in this section). On top of all that, agents can include citation tools that automatically link the final answer back to the specific documents that were retrieved, making the response easier to verify and trust.

Most agentic pipelines follow a repeating loop: the model thinks about what to do, takes an action, observes the result, and then decides what to do next. ReAct [21] formalized this pattern, and it has become a common foundation for how these systems are built.

LangGraph [19, 3] takes a similar idea and organizes it as a graph, making it easier to design and manage more complex workflows with multiple steps and branching paths.

More recently, dedicated thinking tools [16] have been introduced to give models a structured space to reason through harder problems before acting, which tends to help in situations where several steps need to chain together.

Table 1. Illustrative performance comparison of agentic RAG systems with and without thinking capabilities (values reported in the presentation)

Metric	Non-thinking	Thinking
Faithfulness	0.9569	0.9552
Answer relevancy	0.8033	0.8184
Context precision	0.8276	0.8548
Context recall	0.8361	0.9265
Semantic similarity	0.9240	0.9262
Answer correctness	0.6000	0.6014

5.1 Thinking vs. Non-Thinking Results

To illustrate how the thinking step may affect system performance, the same agentic RAG system was run twice, once with the thinking tool enabled and once without it. The table below reports the observed metric values for that comparison.

In this comparison, adding the thinking step was associated with higher answer relevancy, context precision, context recall, and semantic similarity. Faithfulness changed minimally, and answer correctness remained nearly unchanged. Because this section reports one comparison from presentation results, these differences should be interpreted as indicative rather than conclusive; multiple runs with variance estimates are needed to confirm the effect.

6 Applications

In life sciences and drug development, RAG systems are most useful when they are connected to the right sources, research papers, clinical text, internal databases, and other domain-specific material. Pairing that with agentic tool use makes the whole setup considerably more capable.

The practical benefit is that these systems cut down the time researchers spend on the slower parts of scientific work. Instead of manually combing through literature, gathering evidence piece by piece, or waiting on data synthesis, teams can run those steps faster and interactively. Sources get retrieved and

checked quickly, hypotheses can be refined earlier in the process, and the gap between asking a question and having something testable gets noticeably smaller.

This kind of setup is already being explored in real workflows. Collaborative multi-agent designs have been applied to drug discovery [7], and agentic RAG pipelines have been used to work through multi-omics data in precision medicine contexts [1]. In the same broader direction, recent work on summarization, information retrieval, and natural-language medical interfaces provides enabling components that align with this pipeline-oriented view of enhancing scientific reporting approaches [12, 15, 13, 2].

7 Conclusions and Limitations

This paper summarizes RAG and agentic RAG approaches used to accelerate scientific work in industry environments, with a focus on drug development use cases.

Current limitations include retrieval bottlenecks (missing or low-quality context), hallucination risk in high-stakes scientific settings, and increased latency/cost for iterative or thinking-enabled agents. Future work is likely to focus on improved domain embeddings, stronger provenance/citation tracking, and tighter integration with scientific data sources and automation.

References

1. **Arowolo, M. O., Abdulsalam, S. O., Isaaka, R. M., Igulu, K. T., Balogun, B. F., Popescu, M., Xu, D. (2025).** Agentic RAG-driven multi-omics analysis for PI3K/AKT pathway deregulation in precision medicine. Algorithms, Vol. 18, No. 9, pp. 545. DOI: 10.3390/a18090545.
2. **Beladam, N., Ghomari, A. (2025).** Formal modeling and verification of healthcare systems requirements based on simulation and Hoare logic. Computación y Sistemas, Vol. 29, No. 1, pp. 423–440. DOI: 10.13053/CyS-29-1-5140.
3. **Duan, Z., Wang, J. (2024).** Exploration of LLM multi-agent application implementation based on LangGraph+CrewAI. arXiv preprint arXiv:2411.18241.
4. **Es, S., James, J., Espinosa-Anke, L., Schockaert, S. (2024).** RAGAs: Automated evaluation of retrieval augmented generation. Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, pp. 150–158. DOI: 10.18653/v1/2024.eacl-demo.16.
5. **Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.-t. (2020).** Dense passage retrieval for open-domain question answering. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550.
6. **Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., Kang, J. (2020).** BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, Vol. 36, No. 4, pp. 1234–1240.
7. **Lee, N., De Brouwer, E., Hajiramezanali, E., Biancalani, T., Park, C., Scalia, G. (2025).** RAG-enhanced collaborative LLM agents for drug discovery. arXiv preprint arXiv:2502.17506.
8. **Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D. (2020).** Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, Vol. 33, pp. 9459–9474.
9. **Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P. (2024).** Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, Vol. 12, pp. 157–173. DOI: 10.1162/tacl_a_00638.

10. **Lopez, I., Swaminathan, A., Vedula, K., Narayanan, S., Nateghi Haredasht, F., Ma, S. P., Liang, A. S., Tate, S., Maddali, M., Gallo, R. J., Shah, N. H., Chen, J. H. (2025).** Clinical entity augmented retrieval for clinical information extraction. npj Digital Medicine, Vol. 8, No. 1, pp. 45. DOI: 10.1038/s41746-024-01377-1.

11. **Mao, Y., He, P., Liu, X., Shen, Y., Gao, J., Han, J., Chen, W. (2021).** Generation-augmented retrieval for open-domain question answering. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, pp. 4089–4100. DOI: 10.18653/v1/2021.acl-long.316.

12. **Neri-Mendoza, V., Ledeneva, Y., García-Hernández, R. A., Hernández-Castañeda, Á. (2024).** Multi-document text summarization through features relevance calculation. Computación y Sistemas, Vol. 28, No. 3, pp. 1417–1432. DOI: 10.13053/CyS-28-3-5201.

13. **Olivares-Rojas, J. C., González-Serna, J. G., Ramos-Díaz, J. G., Castro-Sánchez, N. A., González-Murueta, J. W. (2025).** From GUI to VUI: A natural language approach to multimodal medical system. Computación y Sistemas, Vol. 29, No. 1, pp. 295–313. DOI: 10.13053/CyS-29-1-5507.

14. **OpenAI (2024).** Embeddings. OpenAI API Documentation.

15. **Padilla-Luis, E. A., Pinto, D., Cerino-Jiménez, R., López-Cortés, F. J., Reyes-Peralta, A. E., Beltrán, B. (2024).** An information retrieval system for fast identification of objects in long duration surveillance videos. Computación y Sistemas, Vol. 28, No. 4, pp. 2353–2368. DOI: 10.13053/CyS-28-4-5283.

16. **Rawat, M., Gupta, A., Goomer, R., Di Bari, A., Gupta, N., Pieraccini, R. (2025).** Pre-act: Multi-step planning and reasoning improves acting in LLM agents. arXiv preprint arXiv:2505.09970.

17. **Reimers, N., Gurevych, I. (2019).** Sentence-bert: Sentence embeddings using siamese bert-networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

18. **Robertson, S., Zaragoza, H. (2009).** The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval, Vol. 3, No. 4, pp. 333–389. DOI: 10.1561/1500000019.

19. **Wang, J., Duan, Z. (2024).** Agent AI with LangGraph: A modular framework for enhancing machine translation using large language models. arXiv preprint arXiv:2412.03801.

20. **Yang, R., Ning, Y., Keppo, E., Liu, M., Hong, C., Bitterman, D. S., Ong, J. C. L., Ting, D. S. W., Liu, N. (2025).** Retrieval-augmented generation for generative artificial intelligence in health care. npj Health Systems, Vol. 2, pp. 4. DOI: 10.1038/s44401-024-00004-1.

21. **Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y. (2023).** ReAct: Synergizing reasoning and acting in language models. International Conference on Learning Representations.

Article received on 12/02/2026; accepted on 02/04/2026.
**Corresponding author is Osvaldo Velazquez-Gonzalez.*