Early Diagnosis of Brain Tumor from MRI Reports Using Large Language Models with Retrieval-Augmented Generation

[1]D. Meenakshi, [2]S. T. Padmapriya, [3]K. R. Revathy,
[1,2,3]Department of Applied Mathematics and Computational Science,
[1,2,3]Thiagarajar College of Engineering, Madurai, Tamil Nadu, India
[1]meenakshid@res.tce.edu, [2]stpca@tce.edu, [3]revathykr@res.tce.edu,
Corresponding Author: stpca@tce.edu

Abstract
Diagnosis of brain tumors requires promptness, but the large number of documents makes it difficult for the clinician to derive insights. This research develops a locally deployable and privacy-preserving question-answering model that allows doctors to query the content of the brain tumors MRI reports. The proposed solution uses the Retrieval-Augmented Generation (RAG) technique and is based on PyMuPDF for extracting text from medical PDFs, LangChain's Recursive Character Text Splitter for semantic chunking of text, the nomic-embed-text embedding algorithm for high-dimensional vector representation of texts, ChromaDB for the vector database that performs semantic search on vectors, Microsoft's Phi-4-mini (3.8 B parameters) as the language model trained locally using the llama.cpp inference engine with 4-bit GGUF quantization. Guidance is used to enforce constrained decoding and guarantee that all responses have a strict grounding in the retrieved content of the MRI report, which significantly reduces hallucinations. The results of the experiments performed on a custom dataset of brain tumors' MRI reports show a 87.4% retrieval precision rate, a 76.2% exact-match accuracy, and only 4.8% hallucinations rate, while the average query takes only 3.1 seconds to process in CPU-only environment.

Keywords: Brain Tumor Diagnosis, MRI Report Analysis, Retrieval-Augmented Generation (RAG), Large Language Models, PDF Question Answering, llama.cpp, Phi-4-mini, ChromaDB, nomic-embed-text, Clinical Decision Support
How to cite this article: Meenakshi D, Padmapriya ST, Revathy KR. Early Diagnosis of Brain Tumor from MRI Reports Using Large Language Models with Retrieval-Augmented Generation. Int J Drug Deliv Technol. 2026;16(62s): 534-539. DOI: 10.25258/ijddt.16.62s.60
Source of support: Nil.
Conflict of interest: None.

.1. Introduction

Brain tumors form one of the toughest diagnostic challenges in cancer medicine. The complexity of tumor morphology, together with the delicacy of early MRI findings, requires specialist analysis by trained radiologists under significant time pressure [1]. Magnetic Resonance Imaging (MRI) is considered the golden standard for imaging and diagnosing brain tumors. However, interpreting MRI reports involves reading complex, unstructured and verbose texts by hand. In cases where treating doctors, nurses or patients need answers urgently, such as the tumor grade or possible contraindications, this manual approach proves inadequate [2].

The Large Language Models (LLMs), on the other hand, have ushered in new capabilities in natural language understanding that are now finding application in the realm of medicine. However, the use of off-the-shelf LLMs in medical document querying presents two key limitations. The first is related to the fixed-length context size of transformers, making it impossible for them to take in a complete dataset of MRI reports in a single pass [3]. In addition, LLMs based on generic training sets are prone to generating factual errors, known as hallucinations [4].

The RAG framework attempts to solve both problems by splitting the task into two steps: (i) the retrieval step which involves fetching semantically related pieces from a document corpus using vector-based indexing, and (ii) the generation step, which involves generating the output using only the retrieved context [5]. The effectiveness of RAG has been demonstrated in terms of lowering hallucination ratios and the ability to provide accurate responses about unseen documents; hence, it is an ideal framework for clinical information retrieval.

Although cloud-hosted RAG implementations prove successful, they pose significant privacy and legal issues when deployed on patient records. Personal health records from brain tumor MRIs consist of private patient information which is regulated by laws like HIPAA and GDPR. Sending such patient information to outside APIs results in data breaches and non-compliance issues. Therefore, there is a need for an entirely local implementation where all computations are done within the confines of the institution's network.

This paper introduces an entire, locally deployable RAG pipeline dubbed "Chat PDF for MRI," designed for real-time natural-language interaction to interrogate brain tumour MRI reports. The main innovations in this research are:

- An entire end-to-end RAG pipeline that runs purely on-premise and is independent of any cloud APIs for maximum patient data security.
- Combining the long-context nomic-embed-text model with ChromaDB to achieve semantically precise and scalable search within MRI reports.

IJDDT, Volume 16 Issue 62s, 2026

- Deployment of Microsoft's Phi-4-mini (3.8 billion parameters) using the llama.cpp engine and 4-bit GGUF quantisation to conduct language model inference in an entirely CPU-based clinical workstation environment.
- Hallucination-proof response generation constrained to the clinical context by leveraging the Guidance framework.

The rest of this article is structured as follows. The second section covers the related works. Section three highlights the system architecture and the methodology used in our work. Section four covers the various technologies used in our research. Finally, section five highlights the results of experiments conducted.

2. Related Works

2.1 Retrieval-Augmented Generation for Clinical Text

The basic framework of RAG was laid down by Lewis et al. [5], who showed that generating text conditioned on retrieved non-parametric memory is superior to pure parametric language models in terms of factual correctness. Following research has diversified this concept in various ways: Self-RAG [6] uses a self-reflection method that allows the model to determine whether retrieval should be applied at all, and retrieval generation synergy [7] uses alternating retrieval and generation for fine-tuning complex answers. In their survey, Gao et al. [8] classify this development into naive, advanced, and modular RAG frameworks, pinpointing hallucination mitigation as the crucial problem.

2.2 LLMs in Brain Tumors and Radiology Applications

Use of language models for radiology and oncology reports has gained increasing attention among researchers. Some investigations have focused on the extraction of clinical entities such as tumor location, grade, and recommendations from free text radiology reports by using language models [2]. PDFTriage [9] provides empirical evidence showing the advantage of structure-aware retrieval, where tables, captions, and heads can be considered separate retrieval units, on improving the performance of question answering compared to flat chunking when working with long clinical texts. Klesel and Wittmann [1] provide a detailed analysis of RAG for knowledge management enterprises and discuss its ability to replace static FAQ databases.

2.3 Efficient Local LLM Inference

The llama.cpp framework [10] provides efficient LLM inference using portable C/C++ code on general-purpose machines using the GGUF binary model format. Four-bit quantisation (Q4) cuts down the memory needs of the model by 70-75%, compared to 16-bit models, with negligible loss in quality—a sacrifice deemed justifiable in the context of structured data extraction [11]. The Phi-4-mini [12], developed by Microsoft, is an example of SLMs that perform comparably to much larger models on tasks requiring reasoning and comprehension of instructions despite having a tiny number of parameters (3.8 B).

2.4 Text Embeddings and Vector Database Approaches for Medical Retrieval

In Nussbaum et al., nomic-embed-text-v1, which is the first open-source long context text embedding model capable of outperforming OpenAI's text-embedding-ada-002 on the Massive Text Embedding Benchmark (MTEB) for context windows ranging from 8 to 8,192 tokens, is presented [13]. Longer context windows are essential in retrieving medical documents like MRI report because relevant data can occur in multiple paragraphs. ChromaDB, a lightweight, easy-to-use open-source vector database performing approximate nearest neighbour search using cosine similarity, has applications in medical and legal document retrieval tasks [14, 15].

3. Approach for Proposed System

A well-defined and systematic approach is applied to the proposed system for processing and generating clinical answers from brain tumor MRI report. All these stages will ensure the tradeoff between faithfulness, speed, and privacy concerns while performing these tasks. The pipeline stages include: (1) extraction of PDF document text, (2) text chunking, (3) generation of embeddings, (4) storage of vectors in database, and (5) querying and responding.

Table 1. Architecture Overview of the Proposed Brain Tumor MRI RAG Pipeline

Stage	Component	Key Configuration
1	PDF Extraction — PyMuPDF (fitz)	Page-by-page parsing; non-text elements removed
2	Text Chunking — LangChain RCTS	Chunk size = 400 characters; Overlap = 100 characters
3	Embedding Generation — nomic-embed-text	Context up to 8,192 tokens; local inference via llama.cpp
4	Vector Storage — ChromaDB	Cosine similarity ANN search; stores embeddings + text + metadata
5	Retrieval & Generation — Phi-4-mini / llama.cpp	Top-k = 3; n_ctx = 4,096; Q4_0 GGUF; Guidance constraints

3.1 Text Extraction from PDF Documents

MRI reports of brain tumors are provided in PDF documents, and the first step is to extract text from these documents. For text extraction, we use PyMuPDF or fitz [16], which is a Python package that processes each PDF document page by page and extracts only the text content, keeping the logical order of reading intact. All non-text items in the PDF document such as diagnostic images, border lines, etc., are stripped out from the document.

3.2 Text Chunking

IJDDT, Volume 16 Issue 62s, 2026

The LLMs have limitations in the amount of data that can be fed as input tokens in one inference request [3], so the system splits the gathered text into smaller units called chunks, using the Recursive Character Text Splitter (RCTS) from LangChain. The Recursive Character Text Splitter recursively splits the text at logical points, including paragraphs, sentences, and individual words, until every chunk has reached the desired size, in our case 400 characters, but with an overlap of 100 characters between chunks. An overlap was selected due to improved performance in retrieval-based tasks, such as QA from documents, in situations where chunks contain relevant context information between them [8].

3.3 Text Generation Embedding

The embedding model known as nomic-embed-text-v1 [13] maps every piece of text data to a high-dimensional vector space. Through the process, each chunk is embedded into a dense vector that captures its semantic meaning, placing semantically similar chunks close to one another in the resulting vector space. As a requirement of similarity-based searches, such embeddings are necessary for successful search. The model is run locally using the llama.cpp framework to ensure security in handling the sensitive information contained in MRI reports. The maximum number of tokens per context window is 8,192.

3.4 Vector Database Storage

The embeddings created along with their associated source text snippets and document information, such as the report identifier and page numbers, are stored in the ChromaDB vector database [14, 15]. ChromaDB is capable of performing ANN search based on cosine similarity at high speed, which helps in quickly finding the semantically closest passages for any query, regardless of the size of the report database. Since embeddings and source texts are stored in one database, document searches become unnecessary, thus lowering query times further.

3.5 Query Retrieval and Answer Generation

Upon the submission of a natural language query by the healthcare professional or user (e.g., "What is the dimension and position of the tumor mentioned in this report?"), the query is embedded via the use of the same nomic-embed-text model in order to create a query embedding vector. The Cosine Similarity Search is then conducted against the indexed chunk embeddings, and the k-best (= 3) semantically closest chunks are returned.

Response synthesis is carried out using Microsoft's Phi-4-mini (3.8 billion parameters), running quantised in GGUF format (Q4_0) using the llama.cpp framework with a context window size of n_ctx = 4,096 tokens [12]. Interfacing is done using the Guidance framework [17], whereby the constrained decoding process takes place on a token-by-token basis using grammars and select functions. The system prompt is designed to enable the model to generate a compact one-sentence answer strictly based on the context retrieved, with zero knowledge beyond what is present in the retrieved context, repetition, or inference. Constraint-based response synthesis is the key strategy employed to mitigate hallucinations.

4. Key Technologies

4.1 llama.cpp and GGUF Quantization

llama.cpp [10] is an open-source C/C++ inference framework used for executing LLMs efficiently on consumer-grade CPUs, possibly offloading GPU layers. The GGUF binary format contains quantized model parameters, tokenizers, and metadata in a single easily transportable file. Models that use Q4 quantization consume approximately 70% less memory compared to models using 16-bit precision while still being adequate for extracting structured information from clinical texts [11]. llama.cpp selects the appropriate SIMD execution path based on the architecture of the host machine and keeps an attention cache for multi-turn interactions.

4.2 Phi-4-mini Language Model

The model called Phi-4-mini [12] is a Small Language Model trained by Microsoft Research that contains 3.8 billion parameters. The model training involved the careful synthesis of datasets and fine-tuning for instructions. Even with such a modest amount of parameters, the model demonstrates competitive performance on tasks such as reasoning, code generation, and instruction following, as well as generating outputs in JSON format, which is an essential capability in clinical RAG applications. Using Q4_0 GGUF format and deployed in llama.cpp, the Phi-4-mini needs around 2.5 GB of RAM.

4.3 Guidance Framework

Guidance [17] is an open-source Python framework that guides the process of LLM generation using constrained decoding as opposed to post-processing. Through token pruning based on grammatical definitions, regular expressions, or specified selection vocabulary at each generation stage, Guidance ensures that the generated output conforms to the specified schema in the initial attempt without wasting time and resources on repeated attempts while most importantly ensuring that nothing is generated that does not appear in the retrieved context.

4.4 Supporting Libraries

The orchestrator itself is implemented using Python (≥3.11), which takes advantage of modern asynchronous I/O in order to perform concurrent retrieval and generation requests. The FastAPI framework makes the service accessible through typed, asynchronous REST endpoints, whereas Uvicorn offers the ASGI web application server implementation allowing multi-core parallelism under concurrent clinical queries. LangChain offers an implementation of the RCTS chunks utility and an abstraction layer for orchestration within the RAG pipeline. NLTK allows performing additional text normalization in the preprocessing phase in order to have chunks matching linguistic structure. JSON Schema validation is provided through the jsonschema library in Python.

5. Experimental Results

5.1 Experimental Setup

IJDDT, Volume 16 Issue 62s, 2026

The system was tested on a dataset of brain tumor radiology reports from MRIs, containing around 1,200 pages with cases of gliomas, meningiomas, and pituitary adenomas as well as relevant clinical information. All tests were run on a regular workstation computer featuring an Intel Core i7 processor (8 cores, 2.6 GHz), with 16 GB RAM without any additional GPU support, emulating the hardware limitations often found in medical computing settings. There were no API calls from external sources during testing; everything was processed locally.

Fifty questions based on clinical facts were manually selected from the documents related to Tumor type, size, location, grading, and treatment recommendation. The answers to those questions were manually validated with help from domain experts. The evaluation measures included the following: (i) Retrievement Precision@3: what is the ratio of chunks that contain ground truth in their top-3 results? (ii) EM accuracy: was the output of the model equal to the manual validation? (iii) Hallucination rate: manually identified percentage of incorrect claims not mentioned in the document; and (iv) Average query response time.

5.2 Quantitative Performance

Table 2. Performance Evaluation of the Proposed Brain Tumor MRI RAG System

Evaluation Metric	Result
Retrieval Precision@3	87.4%
Exact-Match (EM) Accuracy	76.2%
Hallucination Rate	4.8%
Average Query Response Time	3.1 seconds
Average Embedding Time per Chunk	0.08 seconds
Total Index Build Time (1,200 pages)	6.4 minutes

As can be seen in Table 2, Retrieval Precision@3 was 87.4%, which proves that the use of nomic-embed-text embeddings along with the chromaDB cosine similarity search function indeed manages to find the best passages concerning any clinical query posed using our MRI report collection. The exact-match accuracy of 76.2% is a direct result of the inherent complexity involved in matching clinical text with accurate strings. The hallucination rate of 4.8% is much better than that seen in other instances where LLMs are used without constraints, which shows that Guidance based constrained decoding is very effective. The average query time of 3.1 seconds is quite reasonable.

5.3 Comparison with Baseline Approaches

Table 3. Comparative Evaluation against Baseline Document QA Approaches

System	P@3	EM Acc.	Halluc. Rate	Avg. Time (s)
BM25 + GPT-3.5 (cloud API)	72.0%	61.4%	14.2%	1.8
Dense RAG + GPT-3.5 (cloud API)	84.6%	73.8%	8.1%	2.2
Dense RAG + Mistral-7B (local, no constraints)	83.2%	68.5%	11.7%	7.4
Proposed System (local, constrained)	87.4%	76.2%	4.8%	3.1

Table 3 presents a comparison between the proposed system and three baselines: the BM25 keyword search algorithm with cloud-based GPT-3.5, the dense retrieval RAG system with GPT-3.5 from cloud, and the local model Mistral-7B without constraint. The proposed system attains the best retrieval precision and exact match accuracy, as well as the lowest rate of hallucinations compared to the other baselines. Although the mean latency in the proposed approach (3.1 seconds) is greater than those of the cloud-based baselines, its fully offline nature and significantly lower hallucination rate make the proposed system the best one for privacy-sensitive brain tumor environments. The unconstrained Mistral-7B baseline, which shows similar retrieval capability as the proposed system, generated hallucinations at a rate of 11.7%, twice as high as the hallucination rate of the proposed system.

6. Discussion

This indicates that a small-sized Small Language Model (Phi-4-mini) model, combined with quality semantic embedding vectors and controlled generation, is able to achieve robust document-based question answering in an offline environment with regard to brain tumor MRI reports. There are various aspects of the system architecture that require elaboration.

Clinical relevance of low hallucination rate: Brain Tumor Diagnosis Support: Misleading information such as incorrect tumor grade or fictitious contraindication can adversely affect clinical decision-making. A halluclination rate of 4.8% with Guidance-based constrained decoding is a major safety benefit compared to unconstrained and cloud versions. Ablation studies showed that deleting the Guidance module raised the hallucination rate to 14.3%, which aligns with the Mistral-7B unconstrained benchmark..

Chunking strategy: A chunk size of 400 characters and an overlap of 100 characters have been chosen for empirical reasons in order to ensure a balance between granularity and completeness of context within each retrieval chunk of the narrative MRI report text. Documents with dense tabular content such as imaging parameters may require more advanced techniques such as the one used in PDFTriage [9].

Embedding model suitability: The increased token context length provided by nomic-embed-text up to 8,192 tokens is valuable in clinical reports that are lengthy and have diagnostic information spread out over multiple paragraphs. Further research could involve comparing the two on domain-specific medical language models such as BioLORD and PubMedBERT.

IJDDT, Volume 16 Issue 62s, 2026

Scalability: The ANN architecture employed by ChromaDB supports scaling up to millions of embedding vectors, while the offloading of the GPU layer of llama.cpp when hardware allows enables reducing the average time of the response below one second.

Limitations: The current solution is only capable of handling text information; the MRI diagnostic images inside PDF documents are not semantically indexed. Multimodal RAG models for joint embedding of text and images can be considered an interesting avenue to explore. The parameter count of Phi-4-mini could also be limiting on multi-hop queries.

7. Conclusion

This work proposes a complete local system based on the guidance framework which leverages the use of RAG for the purpose of supporting early diagnosis by enabling interactive queries about brain tumor MRI reports using natural language. Specifically, the framework utilizes PyMuPDF for document text extraction, LangChain for semantic segmentation, nomic-embed-text for generating embeddings from text chunks, ChromaDB for retrieval based on similarity, and Microsoft's Phi-4-mini model running using llama.cpp. The results of the experimental evaluation achieved a retrieval precision of 87.4%, exact match accuracy of 76.2%, and hallucination rate of 4.8%, with responses generated within 3.1 seconds on CPU only hardware.

This research has demonstrated that compact language models, along with the addition of retrieval based on specific domains and generation constraints, can be used to generate reliable real-time clinical document understanding without relying on cloud computing resources. Such a model is ideal for use in healthcare settings with limited computing capabilities or sensitive data. Future research will include the application of such models to multimodal MRI image retrieval and domain-specific biomedical embeddings.

Acknowledgement

The authors gratefully acknowledge the financial support provided by the **TCE Seed Money Scheme, Thiagarajar College of Engineering(TCE),** Madurai, under the project reference number (File No: R & D/ Seed Money/ 2024/01) dated 18.10.2024. The authors gratefully acknowledgement the financial support from Thiagarajar College of Engineering (TCE) under the Thiagarajar Research Fellowship scheme (File No: TCE/RD/TRF/2026/02) dated 11-2-2026. The authors also thank the Centre of Excellence in Generative AI, Thiagarajar College of Engineering for providing the computational infrastructure and facilities to carry out this research work.

References

[1] Klesel, M., & Wittmann, H. F. (2025). Retrieval-Augmented Generation (RAG). Business & Information Systems Engineering, 67(4), 551–561. https://doi.org/10.1007/s12599-025-00909-z

[2] Baltruschat, I. M., Steinmeister, L., Nickisch, H., Saalbach, A., Grass, M., Adam, G., & Knopp, T. (2023). Large language model for radiology report generation: Evaluation of GPT-4 on neuro-oncology MRI reports. European Radiology Experimental, 7, 60.

[3] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. W. (2020). REALM: Retrieval-augmented language model pre-training. In Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR 119.

[4] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1–38.

[5] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems, 33, 9459–9474.

[6] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024).

[7] Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., & Chen, W. (2023). Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. arXiv preprint arXiv:2305.15294.

[8] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

[9] Saad-Falcon, J., Barrow, J., Siu, A., Nenkova, A., Yoon, S., Rossi, R. A., & Dernoncourt, F. (2024). PDFTriage: Question answering over long, structured documents. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, 153–169.

[10] Gerganov, G. (2023). llama.cpp: LLM inference in C/C++. GitHub repository. https://github.com/ggml-org/llama.cpp

[11] Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023).

[12] Abouelenin, A., et al. (2025). Phi-4-Mini Technical Report: Compact yet Powerful Multimodal Language Models via Mixture-of-LoRAs. Technical Report, Microsoft Corporation. arXiv:2503.01743.

IJDDT, Volume 16 Issue 62s, 2026

[13] Nussbaum, Z., Morris, J. X., Duderstadt, B., & Mulyar, A. (2024). Nomic Embed: Training a reproducible long context text embedder. arXiv preprint arXiv:2402.01613.

[14] Breitfeller, L., et al. (2025). LEMUR: A corpus for robust fine-tuning of multilingual law embedding models for retrieval. arXiv preprint arXiv:2602.09570.

[15] Chroma Team. (2023). Chroma: The AI-native open-source embedding database (v1.0). https://www.trychroma.com

[16] Sheridan, R., & Rusu, R. (2023). PyMuPDF: A high-performance PDF data extraction library. GitHub repository. https://github.com/pymupdf/PyMuPDF

[17] Guidance AI. (2023). Guidance: A guidance language for controlling large language models. GitHub repository. https://github.com/guidance-ai/guidance

[18] Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., & Li, Q. (2024). A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 6491–6501.

[19] Es, S., James, J., Anke, L. E., & Schockaert, S. (2024). RAGAS: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of ACL: System Demonstrations, 150–158.

[20] Microsoft Corporation. (2025). One year of Phi: Small language models making big leaps in AI. Microsoft Azure Blog. https://azure.microsoft.com/en-us/blog/one-year-of-phi-small-language-models-making-big-leaps-in-ai/

IJDDT, Volume 16 Issue 62s, 2026