Design and Implementation of a Medical Question Answering System Based on Retrieval-Augmented Generation

Jiacong Nie *, Xianxin Liu, Yixiong Tu, Geng Zhao, Danyang Wu,

Yuhao Tang

North China University of Technology, Beijing 100144, China

* Corresponding Author: Jiacong Nie (Email: sakaski.in@gmail.com)

Abstract. With the rapid growth of medical information demand, providing accurate and reliable medical question-answering services has become a pressing challenge. Traditional generative Q&A models are prone to hallucinations, frequently generating inaccurate medical advice, while simple retrieval systems lack natural fluency. To address this, we propose a Retrieval-Augmented Generation (RAG)-based medical QA system with a five-layer architecture. The data processing layer cleans and formats data from the MedChatZH dataset. The knowledge storage layer encodes documents into 768-dimensional vectors using the BGE-small-zh-v1.5 embedding model and builds an index. The retrieval layer selects relevant documents via cosine similarity. The answer generation layer employs the ChatGLM3-6B language model for context-aware responses. Finally, a user-friendly web interface is implemented using Gradio, offering high-quality medical Q&A services.

Keywords: Retrieval-Augmented Generation; Medical Question Answering; Artificial Intelligence; Natural Language Processing.

1. Introduction

With the rapid advancement of AI, Q&A systems such as ChatGPT [1] and Deepseek [2] have made significant breakthroughs and found applications across domains including manufacturing, healthcare, finance, and education [3]. By offering intuitive dialogue interfaces, these systems expand user accessibility and accelerate the evolution from traditional search engines to AI-integrated systems. However, mainstream Q&A systems trained on static general-purpose corpora face limitations, such as outdated knowledge, reasoning inaccuracies, lack of domain norms, and weak adaptability in specialized fields like medical Q&A [4]. To address these issues, researchers are enhancing large model performance in vertical domains [5]. For example, Microsoft's BioGPT applies Dynamic Knowledge Injection (DKI) [6], improving fluency and domain relevance through continued pre-training on biomedical corpora. In addition, Retrieval-Augmented Generation (RAG) systems enhance factual accuracy and domain adaptability by retrieving specialized content to support response generation. This paper explores the application of RAG combined with the ChatGLM model [7] for Chinese medical Q&A tasks. Given the complexity and specificity of Chinese medical language, traditional models often underperform [8], highlighting the necessity of domain-specific training and datasets tailored for Chinese medicine.

2. RAG Technique and Data Foundation

2.1 Key Technologies

The RAG technique was first proposed by Lewis et al. [9] to address the limited ability of large pre-trained language models to access and precisely manipulate knowledge. As a result, their performance lags behind task-specific frameworks in knowledge-intensive tasks. Traditional pre-trained models store knowledge implicitly within their parameters. Updating their knowledge requires retraining the model, and the overall knowledge capacity is limited by the number of parameters, making it difficult to accommodate the vast amount of real-world knowledge.The traditional pre-training model has the problems of difficult to trace the decision basis and low accuracy of knowledge.

RAG technology effectively addresses these limitations. It takes parametric memory (pre-trained model) + explicit non-parametric memory (document vector index) as the knowledge carrier, and generates answers by dynamically retrieving relevant documents from an external knowledge base and combining them with the parametric model, which significantly reduces the risk of hallucination and improves the factual accuracy of the answers.

2.2 Dataset

The MedChatZH dataset constructed by Tan,Yang et al. collects data from sources such as TCM(Traditional Chinese Medicine) books, the Internet, and hospitals [10], as well as data cleaning and filtering by an AI model. After evaluating the data quality by 7B parameter model, a selected dataset containing 763,629 medical instructions and 1,305,194 general instructions is finally formed. This dataset supports the development of dialog applications in the Chinese medical field by relying on the rich TCM corpus. In the training of the embedding model, the training set and test set are divided in the ratio of 80:20 to satisfy the assumptions of statistical independence and independent homogeneous distribution of the data. The domain specificity significantly optimizes the training effect of the embedding model, which enables the model to acquire semantic similarity features in the medical domain, and improves the semantic accuracy of the embedding vectors with stronger differentiation ability for medical concepts.

3. System Design and Implementation

The medical Q&A system based on RAG technology designed and implemented in this paper adopts a layered structure design, which mainly contains five core layers: data preprocessing layer, knowledge storage layer, Q&A retrieval layer, answer generation layer and user interaction layer. As shown in Figure 1

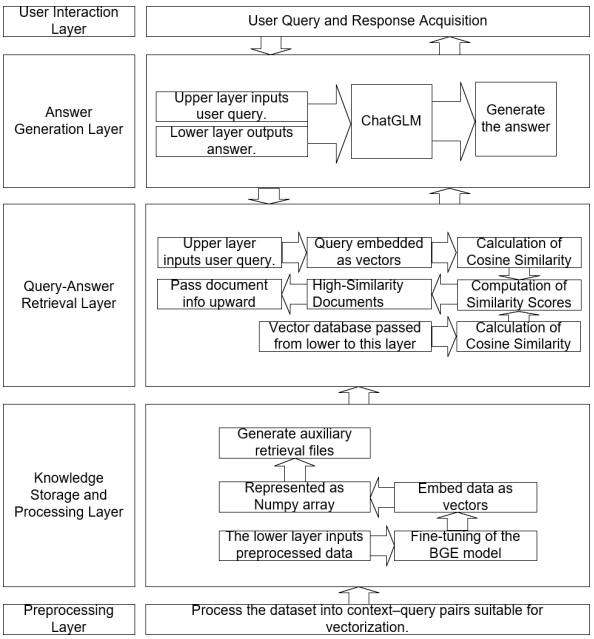

Figure 1. Overall System Architecture

3.1 Preprocessing Layer

The Preprocessing Layer is the data foundation of the whole RAG system, which is mainly responsible for transforming the introduced MedChatZH dataset into a searchable vector database. First, for the characteristics of the MedChatZH dataset, question-answer pairs are accurately extracted by recognizing Q: and A: identifiers, and the multi-line answer texts are automatically merged and transformed into a structured Json format. Each data item contains questions, answers, and metadata

information, enabling the module to process and parse larger-scale medical Q&A data. Through the pandas data processing framework, the nested Json structure is expanded into vectorized context-query pairs, where the context is the answer to the medical question in the dataset and the query corresponds to the question asked. To improve the quality and diversity of the training data, duplicate query-answer pairs in the dataset were removed and the de-duplicated data were saved in CSV format.

3.2 Knowledge Storage and Preprocessing Layer

The knowledge storage layer is the knowledge source of the RAG system. The vectorization processing module is responsible for projecting textual data into high-dimensional semantic vectors [11]. The system is built upon the BGE-small-zh-v1.5 model [12], which has been fine-tuned using data from the MedChatZH dataset. A contrastive learning framework is constructed based on the MultipleNegativesRankingLoss loss function, by semantically aligning the query with its corresponding positive samples, and at the same time learning differentially with the negative samples within the batch. The training process adopts an iterative strategy with 8 epochs, and each training batch is set to 16 samples in conjunction with a gradient accumulation step of 2 and an effective batch size of 32 to balance training stability and computational efficiency. The learning rate is set to 2×10^{-5}, and the cosine annealing scheduling strategy is used to realize the dynamic learning rate decay to ensure the convergence stability of the model in the late stage of training. During processing, the system is optimally deployed based on the hardware configuration of Intel I7-12700H processor and RTX 4070 graphics card. The model automatically detects GPU devices and enables CUDA acceleration. The experimental results show that using GPU acceleration is able to achieve about 94 times speedup for the model fine-tuning process compared to the pure CPU computing mode. The system stores the final vector data in the form of numpy arrays, which contain the document path indexes and the corresponding vector matrices, and outputs an auxiliary retrieval file that contains the original documents to ensure that the correspondence between the vectors and the original text is more accurate.

3.3 Query-Answer Retrieval Layer

The Q&A retrieval layer, as the core component of the RAG system, adopts a dense vector-based semantic matching framework with a dual-encoder architecture, allowing independent encoding of queries and documents. Document vectors are pre-computed and stored, which eliminates redundant online encoding and enhances system efficiency and scalability. The system uses a fine-tuned BGE-small-zh-v1.5 model to encode user queries into 768-dimensional semantic vectors, leveraging the deep semantic capabilities of Transformer architectures for accurate semantic retrieval beyond keyword matching. Encoded queries are converted into PyTorch tensors, and batch cosine similarity is computed between query and document vectors using F.cosine_similarity, exploiting GPU parallelism for speed and memory efficiency. The torch.topk function is employed to retrieve the Top-3 most similar documents, achieving $O(nlogk)$ complexity suitable for large-scale retrieval. The final results include the document paths, similarity scores, vectors, and content, and are passed to the RAG generation module via a standardized interface.

3.4 Answer Generation Layer

The answer generation layer, as the system's core output module, integrates retrieved documents and user questions to produce coherent and contextually accurate responses. Upon receiving a query, it invokes the retrieval layer and constructs a structured prompt using a customized RAG template compatible with large language models. To enhance semantic alignment, a similarity threshold (0.3) filters out low-relevance documents prior to generation.The system employs the ChatGLM3-6B model as its generation engine, quantized with FP16 and deployed via the PyTorch framework in combination with the ModelScope library. GPU memory utilization is optimized by allocating 90% of available memory. The text generation process uses nucleus sampling (top-p=0.7), temperature=0.95, and a maximum output length of 2048 tokens. Running on an RTX 4070 GPU

with CUDA acceleration, the system incorporates automatic memory management to maintain high throughput and stability.Through a custom LLM interface, the system integrates with LangChain and exposes services via FastAPI over HTTP [13]. The generation module leverages LLMChain for prompt composition and context retention. The model supports structured inputs, identifies document boundaries, and produces semantically consistent answers. Final outputs are returned in a standardized JSON format.

3.5 User Interaction Layer

The user interaction layer adopts a multilevel interface architecture, providing dual access to both the web interface and API interface . the Web side builds an intuitive dialog interface based on the Gradio framework to support real-time Q&A interactions, history management, and response status display. the API side adopts the FastAPI framework to provide RESTful services and supports structured data exchanges in JSON format. The standardized middleware interface connects to the underlying retrieval and generation modules to achieve front-end and back-end separation of the architectural design. The front-end captures the user's medical consultation questions through the text input component of Gradio, encapsulates the user's input into JSON format, and passes it to the back-end processing module through API calls. After the back-end completes the intelligent reasoning, the generated answers are formatted and quality checked, and finally the processed answers are returned to the front-end in JSON format.

4. System Demonstration and User Interface Demonstration

4.1 Overall Layout of the User Interface

This system is based on the Gradio framework to build the user interface, which realizes the efficient use of space and simplifies the interaction logic through modular layout. The core modules of the interface include dialog display area, user input area and function button area.

The dialogue display area adopts the classic two-column layout of chatbots, presenting user questions on the left and system answers on the right, and enhancing the differentiation of information through differentiated background colors. When the system is initialized, it displays the introductory message “我是医疗问答助手，我将尽力帮您解决问题。” to clarify the service orientation.

The user input area is designed as a label-free multi-line text box with the placeholder “Please enter the content...” as the operation guideline, taking into account the simplicity of the interface. The user input area is designed as a label-free multi-line text box, with the placeholder “Please enter the content...” as the operation guideline, taking into account the simplicity of the interface and the convenience of input.

The function button area contains “Submit” and “Clear” buttons: the eye-catching orange submit button highlights its importance through high-contrast design, which is in line with the user's intuitive expectation of operation.

4.2 System Interaction Flow Design

After the user enters a medical-related question in the input box, clicking the submit button triggers the system to process it. The system uses the Gradio framework's show_progress=True parameter to enable automatic progress indication, and after the user submits the query, the interface will display a loading animation and processing status prompts. Combined with the demo.queue() queue mechanism, the system can provide real-time feedback on the processing progress. After the user successfully submits a query, the user input is automatically cleared At the same time, the answers generated by the system will be displayed in the dialog area in real time, forming a complete Q&A record. The clear button allows users to reset the dialog history and start a new consultation session at any time. For errors such as network delays, API service exceptions or JSON parsing failures, it mainly relies on the default exception handling mechanism of the Gradio framework. Preventive

handling is performed through gr.close_all() to avoid port conflict problems caused by repeated starts. As shown in Figure 2, it demonstrates the complete interaction process of a user consulting about lung infection related issues.

Figure 2. Lung Infection Counseling Cases

5. Experimental Evaluation

5.1 Evaluation Data Preparation

The system reads query-answer pairs from CSV files as test data. Multiple possibly correct related documents are found for each query by semantic similarity to construct a more reasonable test set and avoid the evaluation bias caused by a single answer.

5.2 Evaluation of Systematic Retrieval Accuracy

The system randomly selects 10 test queries and performs vector retrieval for each query. Two specialized doctors are invited to manually score the retrieval results of Top-10 documents (five grades from A-E) to determine whether the retrieval results can answer the user's question well.

The program performs batch retrieval for all test queries and calculates the metrics including accuracy, recall, and F1 score when the K value is 10. The results are shown in Table 1

Table 1. Retrieval Accuracy Results

Top-10 Precision	Top-10 Recall	F1 Score
95.8%	99.6%	0.976

5.3 User Evaluation and Feedback Analysis

In order to evaluate the performance of this system and native ChatGLM in medical Q&A, 100 professional medical questions were selected to be asked and replies were recorded, and 20 medical professionals were invited to score based on a standardized questionnaire. The participants included 5 doctors, 5 nursing staff, and 10 medical students, and the results are shown in Table 2.

Table 2. User Evaluation Table

Model	Response Accuracy	Customer Satisfaction
Our system	99.2%	99.8%
Vanilla ChatGLM3-6B	95.4%	93.7%

6. Conclusion

Aiming at the shortcomings of traditional generative Q&A models, this paper implements a medical Q&A system based on RAG technology, which effectively solves the “hallucination”

problem and the lack of timeliness of knowledge in traditional medical Q&A systems by integrating dense vector retrieval and large language model generation capabilities. However, there is some room for improvement in this paper, for example, when dealing with multifactorial diagnosis of complex diseases and personalized treatment advice for a certain patient, the depth of reasoning and the degree of personalization of the system need to be strengthened. Looking ahead, RAG-based medical Q&A systems have a broad development prospect. With the continuous enhancement of the large language model and the continuous enrichment of medical data resources, the intelligent medical Q&A system will have an increasingly important position in the fields of medical consulting and medical education, and will also provide a basis for our further optimization.

Acknowledgments

This work was supported in part by the 2025 Beijing College Students' Innovation Training Program under the project titled “Intelligent AI Medical Consultation System Based on Large Models.”

References

[1] WU T, HE S, LIU J, et al. A brief overview of ChatGPT: the history, status quo and potential future development[J]. IEEE /CAA Journal of Automatica Sinica,2023,10(5):1122-1136.

[2] Zhang L, Wang Y, Li S, et al. An Evaluation of DeepSeek Models in Biomedical Natural Language Processing [EB/OL]. arXiv:2503.00624, 2025. Available: https://arxiv.org/abs/2503.00624.

[3] Chen Yongwei. Beyond ChatGPT: Opportunities, Risks, and Challenges of Generative AI. Journal of Shandong University (Philosophy and Social Sciences Edition), 2023, (3): 127–143.

[4] Yan Jianzhi, He Yuxin, Luo Ziye, et al. Typical Applications and Challenges of Generative Large Language Models in the Medical Field. Journal of Medical Informatics, 2023, 44(09): 23–31.

[5] Hu Zhensheng, Yang Rui, Zhu Jiahao, et al. Research and Development of Large Language Models in the Medical Domain. Artificial Intelligence, 2023, (04): 10–19. DOI:10.16453/j.2096-5036.2023.04.002.

[6] Luo R, Sun L, Xia Y, et al. BioGPT: generative pre-trained transformer for biomedical text generation and mining[J]. Briefings in bioinformatics, 2022, 23(6): bbac409.

[7] GLM T, Zeng A, Xu B, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools[J]. arXiv preprint arXiv:2406.12793, 2024.

[8] Wang Yang. Application and Prospects of Traditional Chinese Medicine Diagnosis and Treatment Technologies under the Background of Artificial Intelligence. Information & Computer (Theoretical Edition), 2019, (11): 135–136.

[9] Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks[J]. Advances in neural information processing systems, 2020, 33: 9459-9474.

[10] Tan Y, Li M, Huang Z, et al. Medchatzh: a better medical adviser learns from better instructions[J]. arXiv preprint arXiv:2309.01114, 2023.

[11] Reimers N, Gurevych I. Sentence-bert: Sentence embeddings using siamese bert-networks[J]. arXiv preprint arXiv:1908.10084, 2019.

[12] Xiao S, Liu Z, Zhang P, et al. C-pack: Packed resources for general chinese embeddings[C]//Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval. 2024: 641-649.

[13] Dou Fengqi, Hu Shan, Li Jialong, et al. Design and Implementation of a RAG-based QA System Using LangChain: A Case Study of C Programming Course. Information & Computer (Theoretical Edition), 2024, 36(06): 101–103.