Enhancing Medical Information Retrieval using RAG with Hybrid Vector Search and Query Expansion

Irtiza Hussain [1]

Abstract- **The rise of large language models (LLMs) has improved people's access to medical information. These LLMs, however, lack the domain-specific knowledge essential to answering more specialized questions. To plug this knowledge gap, this paper explores the use of retrieval-augmented generation. A comparative analysis of four different retrieval strategies — dense vector search, dense vector search with query expansion, hybrid vector search, and hybrid vector search with query expansion — was carried out and a combination of hybrid vector search with query expansion was found to be the most effective retrieval strategy. This framework will help efficient and cost effective information retrieval for chatbot used in the medical sector. At the same time, because of its generic nature, this proposed combination can be used for applications in other fields.**

keywords: Retrieval-Augmented Generation, Hybrid Vector Search, Query Expansion, LLMs, Dense Vectors, Sparse Vectors

INTRODUCTION

Efficient retrieval of information about different medical conditions and individual patient's history has become important for personalized treatment plans and effective recovery of patients. The rise of large language models (LLMs) has increased the adoption of generative AI to achieve these goals, especially by building chat bots [1]. These chat bots, with accurate information, can also help patients in countries where the doctor-to-patient ratio is quite low in accessing reliable and correct information about different diseases [2].

While the LLMs used in such chat bots have the capability to answer complex queries, and perform tasks that are related to information already stored in their parameters, they are not capable of answering questions about domain-specific topics like medical records of an individual [3].

In order to use these models for such specialized tasks, it is important to provide them with the necessary context through the prompt, a technique called in-context learning [4]. This is where strategies like Retrieval-Augmented Generation (RAG) come into play [5]. Before any query is passed to the LLM, it is converted to vectors, also called dense vectors, by an embedding model and this vector is passed to a vector database which already carries a list of documents that could be relevant to the user's queries. The necessary contextual information is then retrieved from this database by comparing the query vector with vectors associated with each document of the corpus and ranking it in the order of similarity. This similarity score is generated through metrics like cosine similarity, dot product similarity and Euclidean distance similarity. The retrieved context is then passed to the LLM as part of its prompt using which it would provide an answer to the query.

However, for retrieval from a large corpus of documents, plain dense vector search is often not enough and we need to pair it with more advanced techniques. One such technique is hybrid vector search where, in addition to dense vectors, sparse vectors representing the keywords associated with the corpus are also stored and used for retrieval [6]. The similarity is computed individually for both dense and sparse vectors and then the retrieved documents lists are merged together by assigning weights to both retrieval systems or through advanced techniques like reciprocal rank fusion (RFF) [7] .
Another technique that is used to optimize retrieval is query expansion [8]. A model like BERT, or Word2Vec is trained on the documents and is then used to generate keywords based on the user's query. Queries are also rephrased, as part of this technique, to attach more context and to ensure the embedding model orients the vector in the direction which conveys the highest context.

However, training and hosting a model for query expansion is costly and time consuming. This is where LLMs, with their APIs, provide relief by doing all that we have to do for query expansion without the need to train or host a model, thus eliminating the need for computational capacity and the associated costs. These LLMs can also be used for guiding the retrieval process because, as they are trained on vast amounts of data, they can be used for deciding whether a particular query needs the retrieval process or can be answered without any context. Moreover, an LLM can also generate SQL queries thus, enabling direct data retrieval from databases [9]. In this paper, both hybrid vector search and query expansion using OpenAI's GPT-4o mini were combined to maximize the efficiency of the retrieval process over two medical

NED University of Engineering and
Technolog, Karachi
Country : Pakistan
Email:: irtizahussain221@gmail.com

information datasets, TREC-COVID and NFCorpus, and based on comparison with dense vector search, dense vector search with query expansion and hybrid vector search, it was found that this combination works significantly better than each of the two techniques when used in isolation.

LITERATURE REVIEW

Large language models (LLMs) are getting a lot of attention recently for performing tasks related to report generation, student counselling, content writing, marketing, HR divisions, and code generation [10]. This trend has also resulted in more applications focusing on the uses of natural language processing in the medical sector [1], [2], [11], [12]. While these models can perform some of these tasks very efficiently, they need industry-specific, individual-specific or company-specific training to perform specialized tasks related to these areas.

One way to provide this information is to fine-tune these LLMs as this adjusts the weights of the models as per the additional data and as a result, the model can answer the queries in a more context-aware manner [9].

However, this approach also presents a few problems of its own. For instance, many people might not be completely comfortable with their data being used to train an LLM which is essentially used by everyone [13]. This problem may be solved by deciding to train a completely new model or by using secure fine tuning strategies [13].

Fine-tuning can also increase training and deployment costs but this problem can be mitigated by using solutions like low-rank adaptation (LoRA) where instead of training the whole model, we just freeze the weights of most of the layers and add low-rank matrices carrying fresh weights to be used in a few layers generating better results [14].

The nature of data also has an impact on the success or failure of fine-tuning. For example, if the data from which the context is extracted changes very frequently, the improvements it brings may end up becoming obsolete and ineffective, necessitating a fresh round of data preparation, training and testing.

To solve for this, retrieval-augmented generation (RAG) could be used [5]. The data source is split into self-contained documents and each document is then embedded through an embedding model. This model returns a vector for each document, carrying the embeddings and it is then stored in the vector database. Whenever a user enters the query, the same model is used to generate a vector carrying the embeddings for this query and this vector is compared with the vectors associated with each document in our database. Similarity metrics like cosine similarity, Euclidean distance similarity, or inner product similarity [15] are used to quantify the relevance of each document and then a list of most similar documents is generated and returned. This list is then fed to the LLM for context-rich prompting or other relevant tasks.
For ensuring that the vector database contains the most up-to-date data, the data source is indexed periodically. However, this approach can generate inaccurate results if the dataset changes extremely frequently, for example in stock markets, as it becomes difficult to keep the data up-to-date each second. In such cases, the model has to be fed the information from the SQL databases that are used to maintain the most recent data [9].

Another problem with RAG systems could be a lack of relationships between the different documents. GraphRAG is used in such cases as it can create a graph of related documents inside a database like Neo4j with which complex relationships between the related documents can be captured [3], [16]. Multi-modal RAG systems can be used when the retrieval is not limited to text sources and also includes image or video content [17].

In order to maximize the efficiency of our retrieval in RAG, different techniques are used. One such technique is Hypothetical document embeddings (HyDE) [18]. This method is used to generate a hypothetical document matching the response of a user's query. This hypothetical document is then used for the retrieval of similar documents by comparing its embeddings with the embeddings of actual documents in our database.
Another popular optimization technique is query expansion [19], [20]. In this technique, synonyms or related words are attached to a query before the retrieval. This relatively simple technique can nudge the embedding vector in the right direction and increase the efficiency of the system.

For extremely large datasets, we can return a large number of relevant documents during the retrieval stage and then pass these documents to a model called reranker for re-evaluation based on similarity and context of the query. This two-stage retrieval technique is found to be extremely powerful as it provides more accurate retrieval [21].

Another approach to advance the retrieval process is to rely on two kinds of vectors, i). Dense vectors and ii). Sparse vectors. Dense vectors are generated through neural models trained extensively to predict relationships between a word and a given dimension. Sparse vectors are generated through algorithms like BM25 or TF-IDF and represent the keywords or features in the dataset and the query [22], [23]. A separate similarity score is generated using sparse vectors and dense vectors and then the list of retrieved documents through both vectors is merged using methods like convex combination of lexical and semantic scores and reciprocal rank fusion [7]. This approach is called hybrid vector search [6] because of its use of lexical and semantic similarity for retrieval.

A summary of methods for providing context to LLMs is presented in table 1 while table 2 presents performance optimization techniques used in RAG with brief description.

Techniques	Suitability
Custom model training	Suitable when computational costs, and resources are not .an issue and the data is static
Fine-tuning	Suitable for adding context when the data source is static without having to incur the costs associated with custom .model training
RAG	Suitable when the data source is subject to frequent .changes
GraphRAG	Suitable when the documents also need to be linked with each other through relationships
Multi-modal RAG	Suitable when the data source can be other than text (.(image, video, etc

Table 1: A summary of different techniques used for context-rich prompting with LLMs

Techniques	Explanation
Query Expansion	Appends similar words to query before embedding to enhance the context it carries
HyDE	A hypothetical response is .used to enhance retrieval
Hybrid Vector Search	Used when both lexical and semantic similarity is to be paid attention in the retrieval .process

Table 2: Efficiency optimization techniques in a RAG system

Related Works

A summary of related works is shown in table 3.

Paper	Explanation
[١]	Providing personalised treatment recommendations for multiple myeloma using RAG based chatbot
[٢]	RAG based chatbot for helping patients with infectious diseases
[١١]	Enhancing medical information generation by LLMs using Graph RAG
[١٢]	Used Hybrid RAG for securing medical data management through Multi-Modal LLMs
[٢٤]	Enhanced health-information retrieval by using RAG with large language models to improve topical relevance with factual accuracy
[٢٥]	Enhanced electronic medical record search using RAG by adapting to individual medical search semantics
[٢٦] ,[٢٧] ,[٢٨]	Enhanced medical reasoning of LLMs by providing up-to-date data through RAG
[٢٩]	Optimised RAG for medical information retrieval with a need for follow-up queries by using LLMs
[٣٠]	Used sparse encoder and dense vector indexes with hybrid queries for enhancing RAG efficiency
[٣١]	Used RAG for biomedical questions by filtering distractors, using rationale-based queries, and .reducing retriever bias
[٣٢]	Used RAG for query refinement to retrieve medical information

Table 3: A summary of related works

COMPONENTS OF RETRIEVAL-AUGMENTED GENERATION

The essential components of a retrieval-augmented generation system are explained below:

Embedding Models

Embedding models are used to capture the semantic meaning of text documents. Each model carries a given number of dimensions, with each dimension representing a unique feature. Such models generally return a list of numbers with each number corresponding to the similarity between the word and the dimension at the given index [33].

Semantic Chunking

Since embedding models have a limitation on the number of words or sub words that could be passed to them in one iteration and the given text can have more words than this limit, we need to either truncate each document at the prescribed word limit or split the text into individual text chunks with each chunk respecting the limit of the model. This process is called chunking.

Semantic chunking divides the text by paying attention to the semantic meaning of each portion of the text so that the meaning of the whole document is preserved [34].

Dense Vectors

The vectorized representation of a given text that captures its semantic meaning through an embedding model is called a dense vector. Embedding models usually return a separate vector for each word in the given chunk based on the context of that word. These vectors are combined together by taking the average of the predicted relationship of each dimension. This technique is called mean pooling [35].

The mathematical formula for mean pooling is given below:

Where,
n = total number of vectors
v = vector

Sparse Vectors

Sparse vectors are high dimensional vectors in which most of the values are zeros. The non-zero values reflect the most repeated keywords or features of a given text. BM25 and TF-IDF are some of the most widely used methods for generating this kind of vector [22], [23].

L2 Normalization

L2 normalization is the technique in which we divide each element of a given embedding vector with the magnitude of the entire vector to ensure that the resultant vector has magnitude of 1. It is achieved by the following formula:

where,

$$v_{norm} = \frac{v}{||v||}$$

Vector databases

Vector databases store the vectorized representation of a text source. These stored vectors are later used for retrieval purposes. Common examples include Pinecone, Weaviate, Milvus, etc. [36].

Query Expansion

Query expansion is the process of adding similar or related keywords to a query before using it for retrieval. This method improves the retrieval quality by nudging the query vector in the direction of most similar documents [19].

Cosine Similarity

Cosine similarity is used to determine the similarity between a query and the text documents. It captures the similarity in the orientations of two vectors without considering their magnitudes [15]. Mathematically, cosine similarity between two vectors, A and B, can be computed as:

$$Cosine\ Similarity = \frac{A.B}{||A||\,||B||}$$

$$||A|| = \sqrt{\sum_{i=1}^{n} A_i^2}$$

$$||B|| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

$$A.B = \sum_{i=1}^{n} a_i b_i$$

where,

n = number of dimensions of vectors

a_i = element of vector A at index i of the vector

b_i = element of vector B at index i of the vector

Dot Product Similarity

Dot product is computed by taking into account both the magnitude as well as the direction of the vectors [15]. For two vectors A and B, it can be computed as:

$$Dot\ Product = A.B = \sum_{i=1}^{n} a_i b_i$$

Where,

Hybrid Vector Search

Hybrid vector search is the technique used to enhance the quality of retrieval by using both dense vector and sparse vector search. Dense vector and sparse vector similarities are computed separately using metrics like cosine similarity or dot product similarity. The list of retrieved documents from both lists is then combined to form a final list of retrieved documents [6].

Reciprocal Rank Fusion

This technique generates a unified ranking by using the reciprocal rank of each document in the individual retrieval system and adding them together [7]. A small constant, called bias, is also added to the rank of each individual retrieval system before combining them to prevent the chance of division by zero and to minimize the influence of lower ranked documents. For a given document d, it is computed by the following mathematical equation:

$$RRF(d) = \sum_{i=1}^{k} \frac{1}{(r_{i,d} + bias)}$$

where,

k = number of retrieval systems

$r_{i,d}$ = rank of document d in ith retrieval system

bias = small constant value added to prevent division by zero and to prioritize high ranking documents

DATA PREPARATION AND METHODOLOGY

Datasets Used

Two medical datasets, TREC-COVID [37] and NFCorpus [38] were used for this study. Both the datasets contain queries which are associated with text articles present in the dataset. Relationship between the queries and corpus is established by classifying the articles as:
a). Highly relevant
b). Partially relevant
c). not relevant at all [39]
For this study, all 50 queries from NFCorpus and 50 queries from TREC-COVID were selected. The query sample from TREC-COVID was taken randomly.

A total of 1047 documents were indexed from NFCorpus and 17,537 documents were indexed from TREC-COVID. Each of the documents was relevant to at least one of the queries.
For this paper, the documents were classified as only:
a). Relevant to a query
b). Not relevant to a query

These datasets were used because:
a). Both of them are large datasets carrying information related to the medical sector with queries already related and marked as relevant or not relevant against the documents in the corpus.
b). TREC-COVID is mostly related to information about Covid-19 virus and is therefore more domain specific while NFCorpus is related to general information about a wide range of health related topics. This ensured that we could get performance insights for both general-purpose and domain focused topics ensuring a more nuanced understanding of the approach.
c). This setup allows us to evaluate our retrieval system's performance across datasets of different size as large datasets tend to have lower values of recall because of the presence of high number of relevant documents per query while smaller datasets may show good recall because of the presence of a relatively small number of topics, or documents. The dual evaluation helps in accounting for scale-related biases and reaching generalizable results.
Both the datasets were downloaded using BEIR's python wrapper.

Experimental Setup

Following steps were used to perform the experiment for each dataset:
a). Split the dataset into chunks using semantic chunking through Python's semchunk package.
b). Embedded each chunk of document using bidirectional-encoder representations from transformers (BERT)'s bert-base-uncased-model with mean pooling.
c). Each documents' chunks were then combined into one final vector by averaging the vectors corresponding to each chunk.
d). BM25 encoder was fitted on each dataset's text corpus and stored in a separate file using Python's pickle library. Pinecone's BM25 wrapper was used for this purpose.
e). Sparse vectors for each document were generated using the fitted encoder.
f). Two local instances of the milvus database were created for each dataset. One carried only dense vectors for each selected document in the dataset and one stored both dense and sparse vectors.
g). Against each query, five keywords were obtained using OpenAI's GPT-4o mini model. The following prompt was used for generating the keywords:
"Give me 5 comma separated keywords for this query. Return nothing else".
h). For dense vector retrieval, cosine similarity was used and for sparse vector retrieval, dot product similarity was used.
i). For hybrid vector retrieval, the retrieved lists for both dense and sparse vectors were combined using reciprocal rank fusion in milvus. A bias value of 100 was chosen as it falls in the recommended range of 10 to 100 [40] and being on the higher side, ensures that both dense and sparse retrieval systems contribute meaningfully to the final rankings without one dominating the other.
j). Four kinds of retrievals were performed for each query: i). Dense vector retrieval, ii). Dense vector retrieval with keywords, iii). Hybrid vector retrieval and iv). Hybrid vector retrieval with keywords.
k) For dense vector and hybrid vector retrieval, the queries were vectorized directly and then used for similarity search.
l). For dense vector search with keywords and hybrid vector search with keywords, the queries were vectorized in the following format and then used for similarity search:
"{{Query}}
Keywords: {{comma separated keywords}}"
The architecture of each retrieval strategy is also demonstrated in Figure 1, Figure 2, Figure 3, and Figure 4.

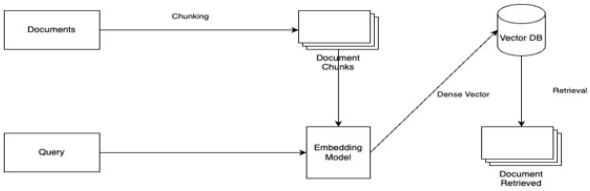

Figure 1: Dense vector retrieval

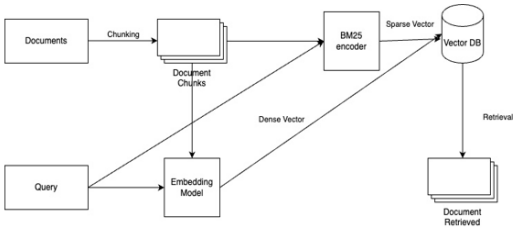

Figure 2: Dense vector retrieval with query expansion

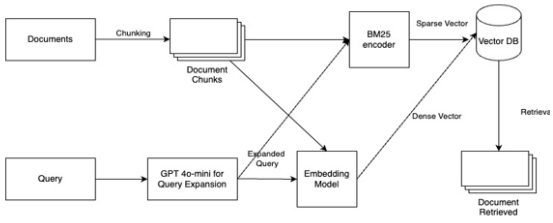

Figure 3: Hybrid vector retrieval

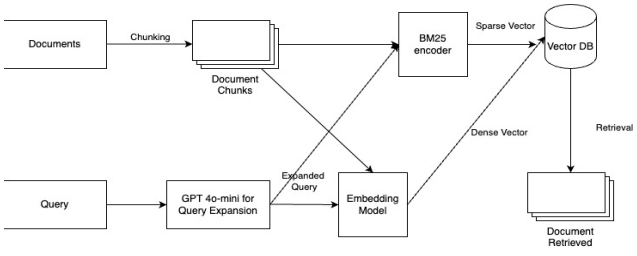

Figure 4: Hybrid vector retrieval with query expansion

Evaluation Metrics

a). Mean Reciprocal Rank at k (MRR@k)

Mean reciprocal rank at k is used to determine the rank of the first relevant document in the list of retrieved documents [41]. Mathematically, it can be computed as:

$$MRR@k = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{Rank_i}\ if\ Rank_i\ \leq k, else\ 0$$

Where,

n = number of queries

Ranki = Rank of the most relevant document in the first k number of retrieved documents

Precision at k (P@k)

It gives us the proportion of relevant documents out of top k retrieved documents. It is calculated by the following formula [42]:

$$P@k = \frac{Total\ relevant\ documents\ retrieved\ in\ the\ top\ k}{k}$$

Recall at k (R@k)

This metric tells us the proportion of relevant documents that were retrieved out of the total number of relevant documents [42].

$$R@k = \frac{Total\ relevant\ documents\ retrieved\ in\ the\ top\ k}{Total\ number\ of\ relevant\ documents}$$

However, in many cases, there could be a large number of relevant documents per query which would result in extremely low values of recall, thus making it a little ineffective indicator of performance. For such cases, it is better to calculate the F1 as discussed below.

F1 Score at k (F1@k)

This metric is used to combine both P@k and R@k into a single metric by balancing the weight assigned to each of them through harmonic mean [42]. It is given by the following formula:

$$F1@k = 2 \times \frac{P@k \times R@k}{P@k + R@k}$$

For this paper, the value of k was taken to be 15.

Baseline for Evaluation

Simple dense vector search results were used as a baseline to measure the improvement brought by each retrieval system as measured by the evaluation metrics discussed above.

RESULTS AND DISCUSSIONS

The results of each retrieval mechanism for both datasets, as measured by mean reciprocal rank at k, precision at k, recall at k, and F1 score at k, with k=15, are presented in Figure 5, Figure 6, Figure 7, and Figure 8 respectively. Percentage improvements in each metric, when compared with simple dense vector search, is presented in table 4 and table 5 for TREC-COVID and NFCorpus respectively.

The heat maps for reciprocal ranks and F1 scores are presented in Figure 9, Figure 10, Figure 11 and Figure 12. These heat maps show that hybrid vectors with query expansion give the most accurate retrieval results. Moreover, the queries with few words show the most pronounced improvement when this combination is used. For example, for the following query in NFCorpus dataset:

“alternative medicine”,

The reciprocal rank improved from 0.2 to 1 and F1 score improved from 0.037 to 0.074 when hybrid vector search was used in combination with query expansion.

When considered against the baseline of simple dense vector search, these improvements can be attributed to the combined effect of following factors:

i). Query expansion improves the performance by adding more context through keywords and removing ambiguity from queries before they are vectorized. The chances for adding noise through keywords is minimal because modern LLMs are fully capable of understanding complex context and generating keywords based on that.

ii). Sparse vector retrieval system accounts for repeated keywords in the corpus and queries which enhances the retrieval quality.

Across the two datasets, TREC-COVID shows higher values of MRR and precision while NFCorpus shows higher values of recall and F1 scores. This can be attributed to the following factors:

i). TREC-COVID is a domain-specific dataset and contains consistent phrasing and jargon while NFCorpus contains diverse topics and inconsistent jargon which can confuse the retriever.

ii). TREC-COVID had more documents in our database (17,537) compared to NFCorpus (1,047) and therefore, carries more relevant documents per query, making it difficult to retrieve all relevant documents and therefore, resulting in lower values of recall while simultaneously increasing the value of precision.

iii). F1 Scores provide a harmonic mean between both precision and recall offering a balanced retrieval performance. The percentage improvement in F1 scores (155.56% for TREC-COVID vs 146.30% for NFCorpus), when compared with the baseline of simple dense vector search, is more pronounced in TREC-COVID because of its domain-specificity (See table 4 and table 5).

iv). Smaller value for F1 score (0.046 for TREC-COVID vs 0.133 for NFCorpus) for TREC-COVID can be explained by the low recall values because of the presence of a high number of relevant documents per query.

Overall, the results suggest that hybrid vector search with query expansion improves the performance across both small, large, general, and domain-specific datasets as the proposed system improved MRR, precision, recall and F1 scores for both datasets when compared with the benchmark of simple dense vector search.

	Dense vector with keywords	Hybrid vectors	Hybrid vectors with keywords
MRR@k	٨,٩٩	٦٥,٧٩	٦٥,٣٥
P@k	٤٧,٨٣	١٠٦,١٦	١٢٧,٥٤
R@k	٥٥,٥٦	١٤٤,٤٤	١٦٦,٦٧
k@F١	٥٠	١٢٧,٧٨	١٥٥,٥٦

Table 4: Percentage improvement through each retrieval system in TREC-COVID

	Dense vector with keywords	Hybrid vectors	Hybrid vectors with keywords
MRR@k	٩٧,١٨	١٧٢,٣٢	٢٣٦,١٦
P@k	٣٤,٧٨	١٢٢,٨٣	١٥١,٠٩
Rl@k	٣٦,٣٦	١٥٦,٠٦	٢١٣,٦٤
k@F١	٤٠,٧٤	١٠٧,٤١	١٤٦,٣٠

Table 5: Percentage improvement through each retrieval system in NFCorpus

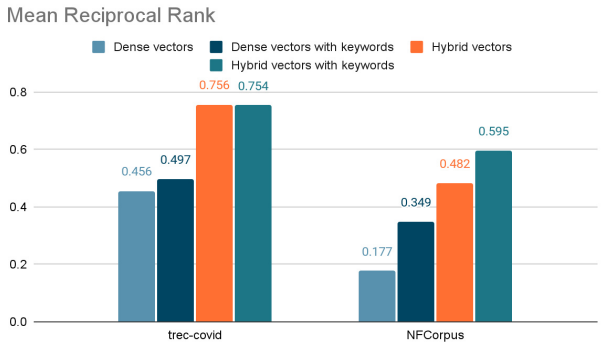

Figure 5: MRR@k with different retrieval strategies

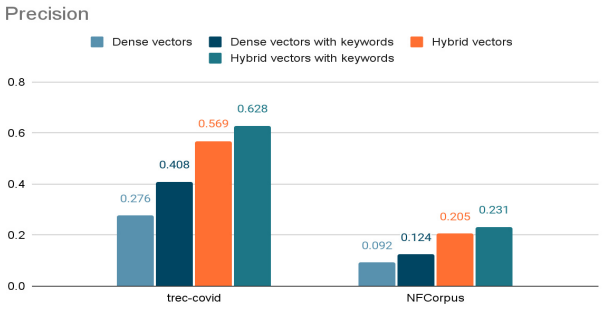

Figure 6: P@k with different retrieval strategies

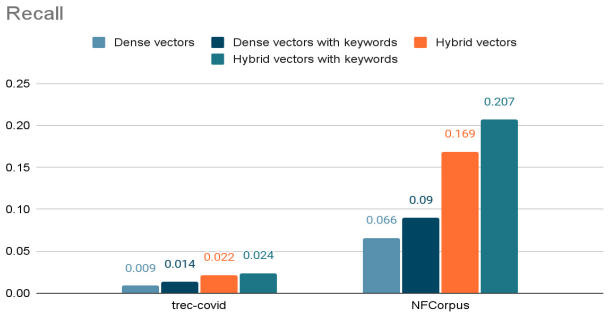

Figure 7: R@k with different retrieval strategies

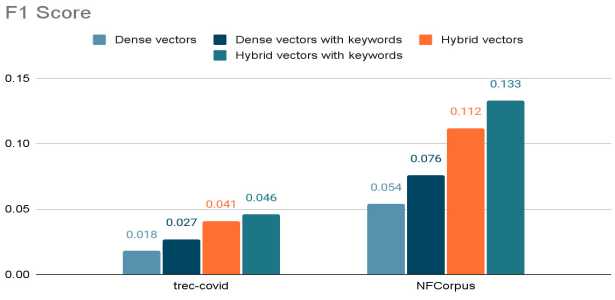

Figure 8: F1@k with different retrieval strategies

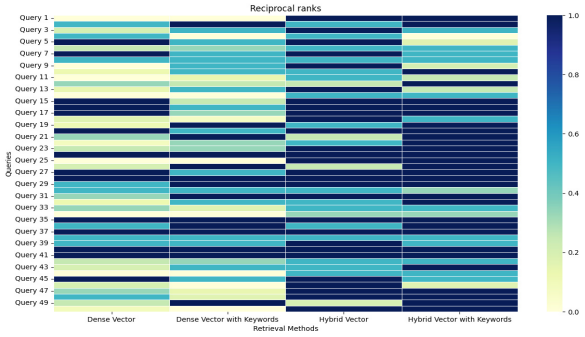

Figure 9: Heat map for reciprocal ranks @k for each query of TREC-COVID

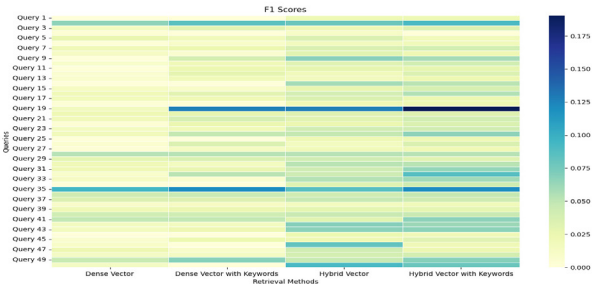

Figure 10: F1@k Scores heat map for each query of TREC-COVID

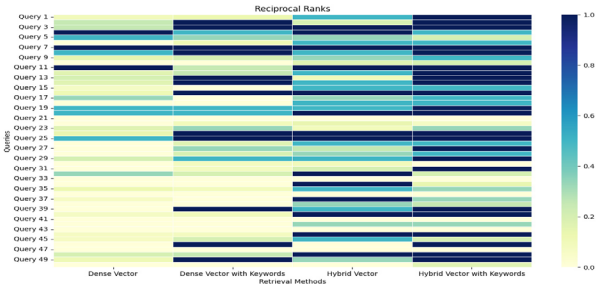

Figure 11: Heat map for reciprocal ranks @k for each query of NFCorpus

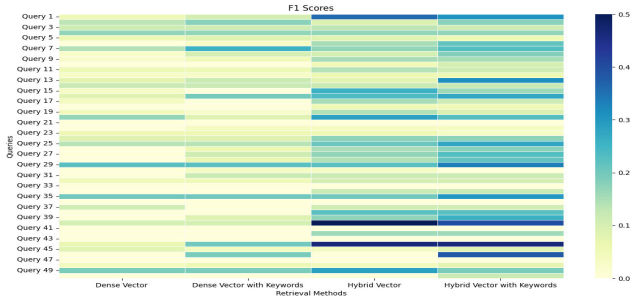

Figure 12: F1@k heat map for each query of NFCorpus

CONCLUSIONS

In the light of this study, it is found that hybrid vector search with query expansion using an LLM like OpenAI's GPT-4o mini can be used to improve the medical information retrieval process significantly and cost-effectively. It is also found that, when compared with plain dense vector search, dense vector search with query expansion, and hybrid vector search, this approach is the most efficient.

This study is significant because the proposed framework can be applied to other practical applications easily.

In the future, this study can be extended by applying this method using knowledge graphs as many real-world applications involve medical documents with complex relationships to each other.

DECLARATION OF COMPETING INTERESTS

The author declares no competing interests associated with this work.

REFERENCES

[1] M. A. Quidwai and A. Lagana, "A RAG chatbot for precision medicine of multiple myeloma," medRxiv, Mar. 2024, doi: 10.1101/2024.03.14.24304293.

[2] S. Kirubakaran S, J. W. K. G, G. M. K. E, M. R. J, R. G. Singh A, and Y. E, "A RAG-based Medical Assistant Especially for Infectious Diseases," in 2024 International Conference on Inventive Computation Technologies (ICICT), Apr. 2024, pp. 1128–1133, doi: 10.1109/ICICT60155.2024.10544639.

[3] T. T. Procko and O. Ochoa, "Graph Retrieval-Augmented Generation for Large Language Models: A Survey," in 2024 Conference on AI, Science, Engineering, and Technology (AIxSET), Sep. 2024, pp. 166–169, doi: 10.1109/AIxSET62544.2024.00030.

[4] Q. Dong et al., "A Survey for In-context Learning," arXiv, 2023, doi: 10.48550/arxiv.2301.00234.

[5] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," arXiv, 2020, doi: 10.48550/arxiv.2005.11401.

[6] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli, "Hybrid search: effectively combining keywords and semantic searches," in The semantic web: research and applications, S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 554–568.

[7] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, Jul. 2009, pp. 758–759, doi: 10.1145/1571941.1572114.

[8] Z. Bai et al., "GQE: Generalized Query Expansion for Enhanced Text-Video Retrieval," arXiv, 2024, doi: 10.48550/arxiv.2408.07249.

[9] "How Q4 Inc. used Amazon Bedrock, RAG, and SQLDatabaseChain to address numerical and structured dataset challenges building their Q&A chatbot | AWS Machine Learning Blog." https://aws.amazon.com/blogs/machine-learning/how-q4-inc-used-amazon-bedrock-rag-and-sqldatabasechain-to-address-numerical-and-structured-dataset-challenges-building-their-qa-chatbot/ (accessed Jan. 19, 2025).

[10] AI at Wharton and GBK Collective, "Growing Up: Navigating Generative AI's Early Years ," AI at Wharton, Oct. 2024.

[11] J. Wu, J. Zhu, and Y. Qi, "Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation," arXiv, 2024, doi: 10.48550/arxiv.2408.04187.

[12] C. Su et al., "Hybrid RAG-empowered Multi-modal LLM for Secure Data Management in Internet of Medical Things: A Diffusion-based Contract Approach," arXiv, 2024, doi: 10.48550/arxiv.2407.00978.

[13] R. Behnia, M. R. Ebrahimi, J. Pacheco, and B. Padmanabhan, "EW-Tune: A Framework for Privately Fine-Tuning Large Language Models with Differential Privacy," in 2022 IEEE International Conference on Data Mining Workshops (ICDMW), Nov. 2022, pp. 560–566, doi: 10.1109/ICDMW58026.2022.00078.

[14] J. Lu, "Low-Rank Approximation, Adaptation, and Other Tales," arXiv, 2024, doi: 10.48550/arxiv.2408.05883.

[15] C. D. Manning, P. Raghavan, and H. Schutze, Introduction to information retrieval. Cambridge: Cambridge University Press, 2008.

[16] J. J. Miller, "Graph database applications and concepts with Neo4j," in Proceedings of the Southern Association for Information Systems Conference, 2013, vol. 2324, pp. 141--147.

[17] W. Lin, J. Chen, J. Mei, A. Coca, and B. Byrne, "Fine-grained late-interaction multi-modal retrieval for retrieval augmented visual question answering," presented at the 37th International Conference on Neural Information Processing Systems (NeurIPS 2024), Red Hook, NY, USA, 2024.

[18] M. Jostmann, "Evaluation of Hypothetical Document and Query Embeddings for Information Retrieval Enhancements in the Context of Diverse User Queries," University of Muenster, Institutional Repository MIAMI, 2024, doi: 10.17879/25968508220.

[19] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey," Information Processing & Management, vol. 56, no. 5, pp. 1698–1735, Sep. 2019, doi: 10.1016/j.ipm.2019.05.009.

[20] K. Ueki, Y. Suzuki, H. Takushima, and T. Hori, "Improving Video Retrieval Performance with Query Expansion Using ChatGPT," in Proceedings of the 2024 7th International Conference on Image and Graphics Processing, New York, NY, USA, Jan. 2024, pp. 431–436, doi: 10.1145/3647649.3647716.

[21] M. Glass, G. Rossiello, M. F. M. Chowdhury, A. Naik, P. Cai, and A. Gliozzo, "Re2g: retrieve, rerank, generate," in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Stroudsburg, PA, USA, 2022, pp. 2701–2715, doi: 10.18653/v1/2022.naacl-main.194.

[22] S. Robertson, "The probabilistic relevance framework: BM25 and beyond," FNT in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2010, doi: 10.1561/1500000019.

[23] A. Aizawa, "An information-theoretic perspective of tf–idf measures," Information Processing & Management, vol. 39, no. 1, pp. 45–65, Jan. 2003, doi: 10.1016/S0306-4573(02)00021-3.

[24] R. Upadhyay and M. Viviani, "Enhancing Health Information Retrieval with RAG by prioritizing topical relevance and factual accuracy," Discov Computing, vol. 28, no. 1, p. 27, Apr. 2025, doi: 10.1007/s10791-025-09505-5.

[25] C. Ye, "Exploring a learning-to-rank approach to enhance the Retrieval Augmented Generation (RAG)-based electronic medical records search engines," Informatics and Health, vol. 1, no. 2, pp. 93–99, Sep. 2024, doi: 10.1016/j.infoh.2024.07.001.

[26] Z. Hammane, F.-E. Ben-Bouazza, and A. Fennan, "SelfRewardRAG: Enhancing Medical Reasoning with Retrieval-Augmented Generation and Self-Evaluation in Large Language Models," in 2024 International Conference on Intelligent Systems and Computer Vision (ISCV), May 2024, pp. 1–8, doi: 10.1109/ISCV60512.2024.10620139.

[27] K. K. Y. Ng, I. Matsuba, and P. C. Zhang, "RAG in Health Care: A Novel Framework for Improving Communication and Decision-Making by Addressing LLM Limitations," NEJM AI, vol. 2, no. 1, Jan. 2025, doi: 10.1056/AIra2400380.

[28] K. Kharitonova, D. Pérez-Fernández, J. Gutiérrez-Hernando, A. Gutiérrez-Fandiño, Z. Callejas, and D. Griol, "Leveraging Retrieval-Augmented Generation for Reliable Medical Question Answering Using Large Language Models," in Hybrid artificial intelligent systems: 19th international conference, HAIS 2024, salamanca, spain, october 9–11, 2024, proceedings, part II, vol. 14858, H. Quintián, E. Corchado, A. Troncoso Lora, H. Pérez García, E. Jove Pérez, J. L. Calvo Rolle, F. J. Martínez de Pisón, P. García Bringas, F. Martínez Álvarez, Á. Herrero, and P. Fosci, Eds. Cham: Springer Nature Switzerland, 2025, pp. 141–153.

[29] G. Xiong, Q. Jin, X. Wang, M. Zhang, Z. Lu, and A. Zhang, "Improving Retrieval-Augmented Generation in Medicine with Iterative Follow-up Questions.," Pac. Symp. Biocomput., vol. 30, pp. 199–214, 2025, doi: 10.1142/9789819807024_0015.

[30] K. Sawarkar, A. Mangal, and S. R. Solanki, "Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers," in 2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR), Aug. 2024, pp. 155–161, doi: 10.1109/MIPR62202.2024.00031.

[31] J. Sohn et al., "Rationale-Guided Retrieval Augmented Generation for Medical Question Answering," arXiv, 2024, doi: 10.48550/arxiv.2411.00300.

[32] A. Tommasel and I. Assent, "Semantic grounding of LLMs using knowledge graphs for query reformulation in medical information retrieval," in 2024 IEEE International Conference on Big Data (BigData), Dec. 2024, pp. 4048–4057, doi: 10.1109/BigData62323.2024.10826117.

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv, 2013, doi: 10.48550/arxiv.1301.3781.

[34] R. Qu, R. Tu, and F. Bao, "Is Semantic Chunking Worth the Computational Cost?," arXiv, 2024, doi: 10.48550/arxiv.2410.13070.

[35] Q. Chen, Z.-H. Ling, and X. Zhu, "Enhancing Sentence Embedding with Generalized Pooling," arXiv, 2018, doi: 10.48550/arxiv.1806.09828.

[36] J. J. Pan, J. Wang, and G. Li, "Survey of vector database management systems," The VLDB Journal, vol. 33, no. 5, pp. 1591–1615, Sep. 2024, doi: 10.1007/s00778-024-00864-x.

[37] E. Voorhees et al., "TREC-COVID: constructing a pandemic information retrieval test collection," SIGIR Forum, vol. 54, no. 1, pp. 1–12, Jun. 2020, doi: 10.1145/3451964.3451965.

[38] V. Boteva, D. Gholipour, A. Sokolov, and S. Riezler, "A Full-Text Learning to Rank Dataset for Medical Information Retrieval," in Advances in information retrieval, vol. 9626, N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, Eds. Cham: Springer International Publishing, 2016, pp. 716–722.

[39] National Institute of Standards and Technology, U.S. Department of Commerce, "TREC-COVID Round 2 Task Guidelines." https://ir.nist.gov/trec-covid/round2.html?utm_source (accessed Jan. 19, 2025).

[40] "Reranking | Milvus Documentation." https://milvus.io/docs/reranking.md (accessed Jun. 18, 2025).

[41] D. R. Radev, H. Qi, H. Wu, and W. Fan, "Evaluating Web-based Question Answering Systems," in Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02), Las Palmas, Canary Islands - Spain, May 2002.

[42] D. Powers and Ailab, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," Mach. Learn. Technol, vol. 2, pp. 2229–3981, Jan. 2011.