IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

Omni Query: A Scalable Privacy-Preserving Offline Rag System for Edge AI

Manya Agrawal[1], Khushi Tyagi[2], Harshvardhan Singh[3], Swati Nagar[4]
B. Tech Students, Computer Science and Engineering[1-3]
Professor, Computer Science and Engineering[4]
Sunder Deep Engineering College, Ghaziabad, India

Abstract: *The Retrieval-Augmented Generation system makes Large Language Models better by using information when it generates responses. But most of these systems need to be on the cloud, which's a problem because it is not private it costs more it is slow and you always need to be connected to the internet. This paper is about Omni Query, a system that works offline and can run on computers. Omni Query is better at finding information because it breaks down documents into parts that make sense and it uses two different ways to search for keywords. Omni Query also uses a process to make sure the results are accurate. The people who made Omni Query also found ways to make the system use memory so it can work well on regular devices. When we tested Omni Query, we found out that it can find the information more than 85 percent of the time and it does this quickly even on devices that are not very powerful. Omni Query can handle a lot of documents than 50,000 and it can still give you answers right away. Because Omni Query works offline it keeps your information private. You do not need to rely on other services. This makes Omni Query a choice for things like healthcare, money and law where you need to be careful with information. Omni Query also lets users interact with it in ways and you can monitor it in real time which makes it more useful, in real life.*

Keywords: Retrieval-Augmented Generation (RAG), Offline AI Systems, Privacy-Preserving Computing, Model Quantization, Hybrid Information Retrieval, Edge AI

I. INTRODUCTION

Large Language Models have made progress in natural language processing. They can do things like reason summarize and have conversations. They still have a problem. They sometimes make up responses that sound good but are not factually correct. This is called hallucination. To make these models more reliable a new approach called retrieval-augmented generation has become popular. It uses knowledge sources to help generate responses. By finding documents and adding them to the model's context this approach helps make sure the responses are factually correct. Most systems that use this approach are designed for cloud-based environments.

This causes problems with data privacy, slow responses. Needing to be connected to the internet all the time. Another problem with systems is that they are not optimized for devices with limited resources. Many solutions need a lot of computing power. Rely on outside APIs to work. This makes them expensive and inefficient. These challenges make it hard to use these systems in real-world situations where resources limited.

We need AI systems that protect privacy and are efficient in areas like healthcare, finance and law. In these areas incorrect responses can have consequences. Also, there are rules about transferring sensitive data to outside cloud services. Offline retrieval-augmented generation systems offer a solution. They can work locally without needing infrastructure. These systems are useful in areas with unreliable internet connectivity. By working on local devices offline systems reduce delays and improve data security. With edge computing and on-device AI being used there is a growing need for systems that can perform well within hardware limits.

Current approaches often focus on either improving retrieval accuracy or optimizing model efficiency. They do not address both aspects at the time. To fill this gap, we propose Omni Query. It is an offline retrieval-augmented

DOI: 10.48175/IJARSCT-34619

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

generation system designed for consumer-grade hardware. The system combines retrieval techniques, efficient document processing and hardware-aware optimizations. This helps achieve a balance between accuracy, efficiency and scalability. The proposed approach shows that high-quality information retrieval and generation can be achieved without services. By integrating keyword-based retrieval with optimized inference mechanisms Omni Query is a practical step toward scalable and privacy-preserving AI systems, for real-world applications.

II. RELATED WORK

A. Retrieval-Augmented Generation

The Large Language Models have a problem with being reliable. So, people are looking at something called Retrieval-Augmented Generation to make them better. This means the models can look at information from outside when they are generating a response. They do not just depend on what they have learned.

Instead, they can find the information they need and use it when they are working. Some time ago people did some studies on this. They found out that when they combined finding information with generating text the models worked better on tasks that need a lot of knowledge. These models can look at a lot of information. Use the best parts. They can even look at information that's new. Now people are trying to make the finding part work better. They are using techniques that help the models understand what the information is about. Sometimes these techniques do not work well when they need to find exact words. This is a problem in some areas. So, people are trying something. They are combining the techniques, with older methods that look for exact words. This way the models can understand what the information is about and also find the words they need.

This new way is working well. The models are finding information and they are finding the right information. Retrieval-Augmented Generation is getting better because of this. Large Language Models are becoming more reliable because they can use Retrieval-Augmented Generation.

B. Vector Search and Indexing

Similarity search is really important for big RAG systems to work well. Vector databases are a way to handle a lot of embeddings by using a method called approximate nearest neighbor's search. There are tools like FAISS that people use a lot for vector search. These tools can do kinds of indexing like using graphs

or partitions. The graph kind, like HNSW is good because it can search quickly and does not use up many resources, which makes it good for big sets of data. There are libraries, like Annoy and Scan that try to solve the same problems. FAISS is still the one that people like to use because it is flexible can handle a lot of work and works really well on regular computer systems, which is great when you need to use it offline. Vector databases and similarity search are important, for RAG systems and vector databases help with this.

C. Optimization and Quantization

Running language models on small computers needs to be done in a smart way. One good way to do this is to use something called model quantization, which makes the model use precise numbers so it uses less memory and is faster. This means the model can be made smaller which is really helpful. There are also some techniques, like Low-Rank Adaptation that let us make changes to the model without having to start all over again which saves a lot of time. These ways of making the models work better are really important for putting language models on devices, like the ones people have at home, where the computers are not very powerful. Language models need these optimization strategies to work well on these devices.

D. Offline and Privacy-Preserving Systems

The thing about Artificial Intelligence that keeps our information private is getting more important these days. A lot of systems we use work with cloud-based processing, which means our data has to be sent to servers that's not ours. This is a problem because it makes us worry about our data being safe if it is okay, with the rules and if we can trust the

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

people who handle it. To make these problems go away people have thought of a way to make language models work on our own devices. Systems that are made to work on our devices try to get rid of the need to talk to servers, which helps keep our information private and makes things happen faster. A lot of the solutions that do not need the internet have trouble working well when they have to handle a lot of things or they do not have a good way to find the information we need. So we still need systems that can keep our information private find what we need accurately and work well without wasting time or resources. Artificial Intelligence that keeps our information private is still something we need to work on.

E. Research Gap

Although many improvements have been made in systems that use retrieval most current methods focus on one thing, like how they retrieve information or how fast they work. Not many solutions deal with accuracy, scalability and privacy at once especially in situations where resources are limited. To fix this we suggest Omni Query, a system that uses retrieval-augmented generation and works well on regular computers. Our approach combines retrieval methods with efficient processing to balance performance and privacy making Omni Query suitable, for real-world use.

III. SYSTEM ARCHITECTURE

The Omni Query framework has an scalable design. This design helps with retrieval-augmented generation even when its offline.

The system is split into layers. Each layer has a job in the processing pipeline.

• This organization makes it clear what each part is responsible for.

• It also helps to optimize each part

The framework is structured to support processing. The Omni Query framework has functional layers.

Each layer in the Omni Query framework is responsible, for a stage. This helps with the optimization of the Omni Query frame-work components. The Omni Query framework is designed to support offline
retrieval-augmented generation.

A. Overview

The architecture has five parts:

• The Presentation Layer

• The API Layer

• The Services Layer

• The Core Processing Layer

• The Storage and Inference Layer

These parts work together one, after the other to take in what users say and give them answers quickly. The Presentation Layer gets what users say.

The API Layer helps it talk to parts. The Services Layer does some work.

The Core Processing Layer does the main work,

and The Storage and Inference Layer stores and uses data to make decisions.This way they make the system efficient.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

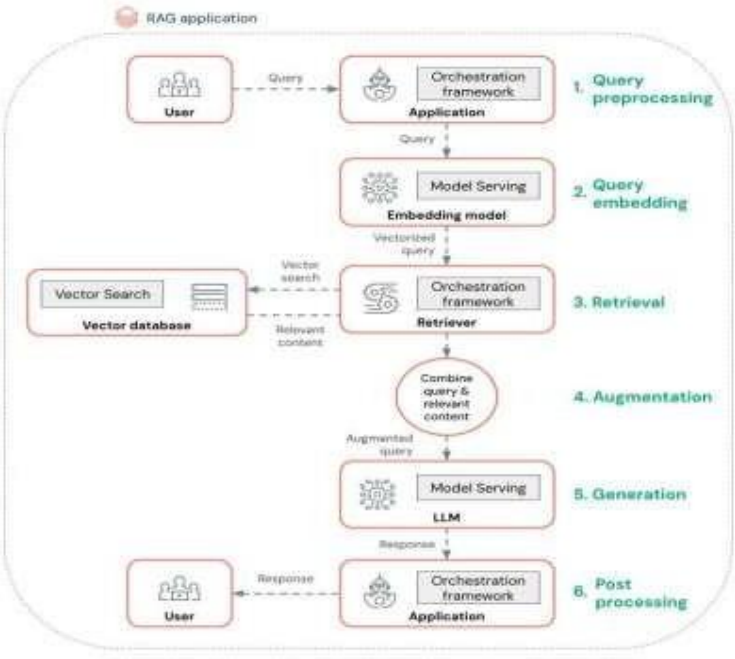

Fig. 1. Omni Query System Architecture

As shown in Fig. 1 Our system has a design. This design makes it easy to add or change parts without affecting the system. It also helps to use computer resources Each layer can be updated on its own. This way we can improve one layer without changing the others. The system stays stable. Works well.

B. Presentation Layer

The presentation layer is, like the face of the system that users interact with. It has both command-line interfaces. This means users can choose how they want to use it.

This layers main job is to get what users want and show them the answers. The presentation layer focuses on being quick and easy to use.

That way it works well on devices that are not very powerful.

The presentation layer helps users get what they need fast. It is designed to be simple. The goal is to make interactions smooth.

C. API Layer

The API layer is like a messenger, between the frontend interface and the backend processing modules. It is built using FastAPI, which helps it to handle requests quickly.

This API layer does things like checking requests, routing and coordinating services. The API layer makes sure that input queries are put together correctly before they are sent to parts of the system for processing. The API layer is really important because it helps the frontend interface and the backend processing modules work together smoothly and the API layer does this by using FastAPI to handle requests.

D. Services Layer

The services layer is where you find the parts of the system that do all the work. This services layer is divided into three services: document ingestion, query processing and response generation.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

The document ingestion service gets the input documents ready and puts them in order so they can be used later. The query processing service is in charge of finding the documents and ranking them. Then the response generation service takes the information it found and creates the output.

Breaking the system into these parts makes it easier to fix things and update the system. Each of the services can be worked on by itself which is really helpful. The services layer and the services in it, like the document ingestion service, the query processing service and the response generation service all work together to make the system work well.

E. Core Processing Layer

The core processing layer handles computer tasks. These tasks include generating embeddings, finding data and reordering results.

We generate text embeddings with Sentence-BERT. This helps us understand how sentences relate to each other. We store these embeddings in a database. This makes it fast to find data.

Our system uses two methods to find data. First it searches for vectors. Second it matches keywords using BM25. This makes our search more accurate. We then use a cross-encoder to reorder the results. This helps us score relevance better.

F. Storage and Inference Layer

The storage part has two things: a vector database called FAISS to store embeddings and a simple database called SQLite to handle metadata.

The inference part uses a language model that has been made smaller which is done using llama.cpp. This process of making it smaller is called quantization. It really helps because it means we need memory to run the model and it works well even on devices that do not have a lot of resources. The language model is a part of this and quantization is what makes it possible to use the language model on these devices.

G. Design Considerations

The system is built with a focus on keeping user data private running efficiently and handling amounts of data. By working completely offline it does not rely on connections, which helps keep data safe.

Also the system uses techniques like data quantization and efficient organization methods, which allow it to run well on regular computers and devices.

The systems modular design also makes it easy to add features, such as support for multiple languages and different types of data without having to make big changes, to how it works.

IV. METHODOLOGY

The proposed system uses a step-by-step method to build a Retrieval-Augmented Generation or RAG pipeline. It is designed to do the following:

- Process documents
- Find information
- Generate responses that are accurate in context

This is done through a series of stages that are fine-tuned for best results.

The RAG pipeline helps to make sure that the information generated is accurate and relevant, to the context.

A. Pipeline Overview

The workflow has steps.

- First we ingest documents.
- Then we preprocess them.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

• Next we do chunking.
• After that we generate embeddings.
• We also do retrieval.
• The results are reranked.
• Finally we generate a response.

Each step is optimized to work on devices with limited resources.

The workflow stages are document ingestion, preprocessing, semantic chunking, embedding generation, retrieval, reranking and response generation.

These stages are optimized for execution, on resource constrained hardware.

We make sure each stage runs smoothly on resources. The goal is to make the workflow efficient. It consists of stages. We optimize every stage.

The stages are designed to work on resource-constrained hardware. We focus on efficiency.

B. Document Processing

The system works with document types like PDF, DOCX, CSV and JSON. It pulls out the content using techniques for each format to get the text data accurately.

Before we do anything we clean up the content by getting rid of weird characters making the formatting the same and ensuring all documents look consistent. This makes the data better, for what comes

Working with many different document types can be tough because they don't always look the same and might be coded differently. We deal with these problems by using strategies that understand each format.

C. Chunking Strategy

Chunking is really important for getting results when searching for things. The system doesn't just break things up into sized pieces it breaks them up in a way that makes sense so we don't lose the meaning of what we're talking about.

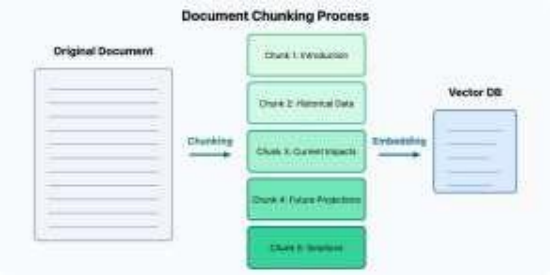

Fig. 2. Document Chunking and Embedding Process

• Documents are broken up into parts based on where it makes sense to break them like at paragraphs or sections.
• We also make some parts overlap so that we don't lose information that's in between.

When it comes to deciding how big each chunk should be there is a compromise. If we make the chunks smaller we get accurate results. If we make them bigger we get information, about what we're talking about. We try to find a balance so that we get results and don't take too long.

As shown in Fig. 2, the document is first segmented into meaningful chunks before being converted into vector representations.

D. Embedding Generation

Each chunk of text gets turned into a dense vector using sentence-BERT. These vectors help understand how sentences are related to each other and make it fast to find text.

To make it work better we create these vectors in groups so we can work on many at the time. This helps speed things upon computers that use a CPU by reducing the time it takes to do the calculations.

DOI: 10.48175/IJARSCT-34619

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

E. Retrieval Mechanism

The retrieval component uses two methods to get information. It uses retrieval and sparse retrieval. Dense retrieval looks for vectors using FAISS. On the hand sparse retrieval uses BM25 to find keywords that are relevant.

The system then combines the results of both retrieval and sparse retrieval methods. This way the system gets the benefits of understanding the meaning of words. Also finding the exact keywords that are needed. The retrieval component really helps the system by using both retrieval and sparse retrieval.

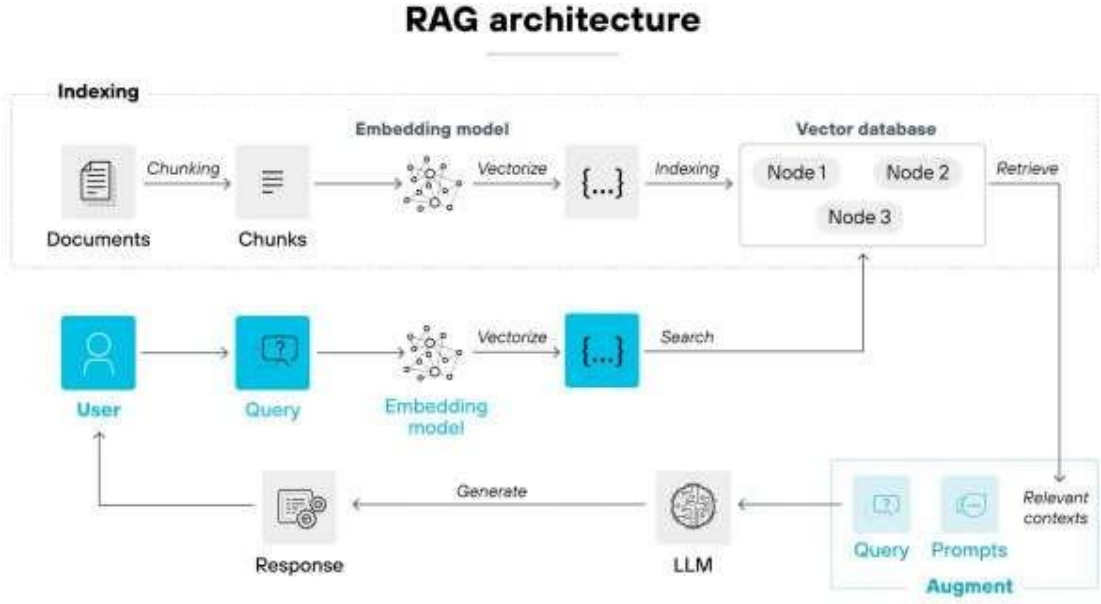

Fig. 3. Hybrid Retrieval-Augmented Generation Pipeline

Fig. 3 illustrates the hybrid retrieval workflow, highlighting how semantic and lexical retrieval are integrated to improve overall performance.

F. Reranking

To make the search results better a special model called a encoder reranking model is used on the best documents that the system finds. This model looks at the search query and the documents together. Gives them new scores that show how relevant they are.

Using this reranking model makes the search results more accurate. However it also takes time and computer power to do this. So the system only uses this model on the documents that's most likely to be what the user is looking for. This way the system can find a balance, between giving good results and not taking too long to do it. The cross-encoder reranking model is only applied to the -encoder reranking models most relevant candidates.

G. Response Generation

The final answer is created using a kind of language model. This model looks at what the user asked for. The information it found that is related to the question.

The model uses a plan to manage all the information it has. It makes sure to use the important information and leaves out the things that are not as important when it has to. The model does this so it can work within the limits it has.

H. Hardware Optimization

To make work on devices that do not have many resources we use several optimization techniques. One of them is model quantization. It helps to use memory and makes the model work faster.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

We also use ways to index things, like HNSW. This helps to get answers. Furthermore, we process things in batches. This way we can make use of the hardware and get more work done.

I. Design Summary

The method they are suggesting uses a lot of techniques to get a good balance, between being accurate working quickly and keeping things private. This system works well and can even work without being connected to the internet because it uses things like breaking down information into smaller pieces getting information in a special way and making good guesses with the information it has. The system is reliable. It does all of this without needing to be online.

J. System Features and Capabilities

Multi-Modal and Multilingual Interaction: Beyond text-
based queries our system also supports other ways to interact such as using images and speech.

• We use Optical Character Recognition (OCR) to get the text from images. This helps a lot because people can just take a picture and get the text from it.

• We also have a feature that converts speech to text. This means you can talk to the system and it will understand you. The system works in languages, which makes it useful for people all, over the world. It is easy to use for people who speak languages.

Context-Aware Interaction: The system remembers what we talked about before. This helps the system give answers that make sense when we have a long conversation. The system uses a way to keep track of what we say during a conversation.

This means the system can give answers that'r more relevant to what we are talking about. The system is good at having conversations that go back and forth because it knows what the conversation is, about.

User-Controlled Inference: You can use the control panel to change things like the temperature and how tokens you can use. It also lets you decide how you want to search for things. This means you have a lot of control, over how the system works so you can make it do what you need it to do.

System Monitoring and Optimization: A real-time monitoring interface shows how much of the CPU and memory are being used.

This helps manage resources while the system is running.

The interface provides insights, into CPU usage and memory usage. It enables management of CPU and memory during runtime.

Data Management: The system has things that help us look at documents before we use them keep our storage neat and make sure our data is good. This helps the retrieval pipeline get the information from the document system. The document system is very important, for getting results.

Export and Reproducibility: People can take out the answers that are made and the things that were talked about before so they can do the thing again and use it with other things they do outside of this.

Scalability: With, over 25 features built-in this system can easily handle real-world situations. It is very scalable and adaptable. The system shows it can handle real-world deployments. It has scalability and adaptability. These features make the system more, than a basic RAG implementation. They make the system a practical offline Artificial Intelligence platform. The Artificial Intelligence platform is really useful because it can be used offline. The features of the system and the Artificial Intelligence platform are very important. The system is a tool because it has these features and it is an offline Artificial Intelligence platform.

V. RESULTS AND ANALYSIS

This part looks at how the Omni Query system works in different areas, such as how good it is at finding what you need the quality of the answers it gives and how well it can handle a lot of work.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

We compare the results of the Omni Query system with systems to show what makes the Omni Query system better, than the others.

A. Experimental Setup

All my experiments were run on a computer, with an Intel i5 processor and 16GB of RAM. This computer works completely offline. Doesn't use any external APIs.

The evaluation was done on known benchmark datasets like:

- Natural Questions
- MS MARCO
- TREC-COVID

I measured how well the system retrieves information using common metrics like:

- NDCG@10
- Mean Reciprocal Rank (MRR)
- Recall@10

B. Retrieval Performance

The results show that Omni Query does better than the baseline system in all areas. Omni Query gets a score of 0.85 for NDCG@10, which's much better than PrivateGPT.

The reason Omni Query works well is because it uses a mix of retrieval methods. One method, retrieval helps find documents that are related to the query. Another method, BM25 helps find documents that have the keywords. Using both methods together makes it better at finding what you need and ranking the results.

Also, Omni Query is much faster, than the baseline system. This is because it has indexing and is optimized for the hardware, which makes it process queries quickly.

System	NDCG@10	MRR	Recall@10	Latency
PrivateGPT	0.71	0.65	0.62	850 ms
Omni Query	0.85	0.82	0.82	380 ms

TABLE I: RETRIEVAL PERFORMANCE COMPARISON

C. Ablation Study

The study looks at what each part of the system does. If we take out the part that helps find meanings the system does not work as well. This shows that this part is important for finding things that mean the thing.

If we do not use BM25 the system is not as good at finding things when we use keywords. The system works best when we use both the part that helps find meanings and the part that looks at keywords. This shows that using both is an idea. When we add a tool that helps rank the results the system gets even better. We get accurate results with a score of 0.87. But this also means the system has to work so it is a trade- off between getting good results and using a lot of computer power. The study, with the system and the special tool shows that using the system with the tool is a good way to get good results but it also uses more computer power.

TABLE II ABLATION STUDY

Configuration	NDCG@10
Dense Only	0.75
BM25 Only	0.70
Hybrid	0.85
Hybrid + Rerank	0.87

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

D. Generation Quality

System	ROUGE-L	BLEU-4	Human Score
LLM Only	0.23	0.12	2.1
PrivateGPT	0.36	0.21	3.4
Omni Query	0.43	0.28	4.2

TABLE III GENERATION PERFORMANCE

I looked at the responses that Omni Query generated. It does a job than other systems. The responses from Omni Query make sense and are more accurate. When Omni Query uses information it has found it makes a difference. This helps stop the system from making things up and makes the responses easier to understand.

People also looked at the responses. They agree. Omni Query got a score of 4.2 out of 5. This means that Omni Query produces good responses. Omni Query is good, at giving quality responses.

E. Scalability Analysis

The system handles datasets really well.

• It gets better as the dataset size increases.

The delay in response time does not increase quickly.

This is because it uses indexing methods like HNSW to manage data efficiently. Latency grows slowly as dataset size increases.

The system demonstrates scalability characteristics, with efficient indexing mechanisms.

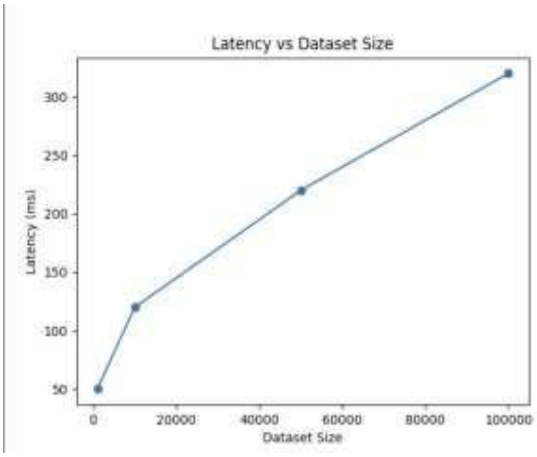

Fig. 4. Latency vs Dataset Size

If you look at Figure 1 you can see that the system works well even when we add documents. The system still works well even when we have a lot of documents like 100,000 documents.

This shows that the system is good, for real world applications that need to be able to handle a lot of information.

F. Key Observations

The experimental results provide several important insights:

• Hybrid retrieval is really good because it helps with finding the things and getting the right results. It does this by using two kinds of matching: matching and lexical matching. Hybrid retrieval improves recall, which's about finding all the things that are relevant and precision which is, about finding only the things that are really relevant. By combining lexical matching, hybrid retrieval gets better at both recall and precision.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

• Model quantization helps to deploy models efficiently without a big drop in performance. It allows for faster and more cost-effective model execution. This approach reduces the model's memory and computational requirements.

As a result, model quantization is a useful technique for making AI models more practical in real-world applications. It achieves this by reducing the precision of the model's weights and activations

Typically, this means going from 32-bit floating-point numbers to 8-bit integers.

By doing so, it minimizes the performance loss while still achieving significant improvements in efficiency.

Model quantization is an important technique for deploying AI models in environments with limited computational resources.

It allows complex models to run efficiently on edge devices that would otherwise not support high-performance inference.

By reducing the precision of model parameters, quantization decreases memory usage and computational requirements without causing a significant drop in performance. This makes it especially useful for applications where speed and efficiency are critical.

As a result, developers can build and deploy scalable AI solutions across a wide range of platforms, including smartphones, embedded systems, and other resource-constrained devices. It also helps in reducing operational costs and improving overall system efficiency.

Furthermore, quantization enables faster inference and better utilization of hardware resources, making real-time AI applications more practical and accessible.

Overall, model quantization plays a key role in making AI systems more efficient, scalable, and suitable for real-world deployment.

Model quantization plays an important role in improving system efficiency by reducing computational requirements while maintaining acceptable performance.

Reranking helps improve the accuracy of results, although it adds some additional computational cost.

The system performs effectively even with large datasets and is capable of handling increasing amounts of data efficiently. It demonstrates strong scalability in real-world scenarios.

Overall, the results show that the proposed approach successfully balances accuracy, efficiency, and scalability. The system delivers reliable performance while remaining resource-efficient and capable of managing large-scale data.

VI. CONCLUSION

In this work we introduced Omni Query, a system that is supposed to make Retrieval-Augmented Generation work better in situations where we do not have a lot of resources. The system we are proposing combines a things like understanding the meaning of things finding the right information putting things in order and using simple language models to get a good balance between being accurate working fast and keeping things private.

When we tried out Omni Query, we found out that it works well with accuracy of more than 0.85 and it does this without taking too long or using too much memory. The way we find information, which's by combining understanding what things mean with looking for specific words is really important for getting good results.

What is also great about Omni Query is that it works well on computers, which means it can be used in places where we do not have a lot of powerful computers. Because it can work without being connected to the internet it keeps our data safe which is really important for things like healthcare, money and legal stuff.

One other good thing about this work is that it shows we can use AI systems without needing big servers. This makes it cheaper to use. Helps people who do not have very good internet connections.

In the future we can make Omni Query work with languages and it can look at pictures and other things, not just text. We can also try to make it use energy. And we can make it smarter about how it finds information so it can balance being accurate and being fast.

Overall Omni Query is a step, towards making AI systems that's good work fast and keep our information safe which is what we need for real life.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

REFERENCES

[1] T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.
[2] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proceedings of NeurIPS, 2020.
[3] V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," in Proceedings of EMNLP, 2020.
[4] X. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023.
[5] J. Johnson et al., "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.
[6] E. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," in Proceedings of ICLR, 2022.
[7] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in Proceedings of CVPR, 2018.
[8] T. Kwiatkowski et al., "Natural Questions: A Benchmark for Question Answering Research," Transactions of the Association for Computational Linguistics, 2019.
[9] P. Bajaj et al., "MS MARCO: A Human Generated Machine Reading Comprehension Dataset," arXiv preprint arXiv:1611.09268, 2016.
[10] E. Voorhees et al., "TREC-COVID: Relevance Judgments and Bench-marking," in Proceedings of the SIGIR Workshop, 2021