IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

Retrieval-Augmented Document Querying and Context-Aware Answer Generation Using Vector Indexing and Large Language Models

M. Ayyappa Chakravarthi[1], Yayavaram Raja Sri[2], Moparthi Asha[3], Shaik Samirin Kousar[4], Tammuluri Reena Prashanthi[5]

Associate Professor, Dept. of CSE–Data Science, KKR & KSR Institute of Technology and Sciences, Guntur, Andhra Pradesh [1]

B.Tech student, Dept. of CSE–Data Science, KKR & KSR Institute of Technology and Sciences, Guntur, Andhra Pradesh[2-5]

Abstract: The rapid growth of digital documents has been increasing exponentially in various domains and the effective retrieval of documents has become a challenging task. Traditional keyword search could not understand the real meaning of words. It only looks for matching words, so many times the results are not correct. It also misses the context of the question. Because of this, the answers are not accurate. Now, new technologies like Generative Artificial Intelligence and Natural Language Processing make this work easier and better.

The objective of our project is to develop an intelligent document querying system that enables efficient question-answering by integrating document retrieval methods with Large Language Models through a Retrieval Augmented Generation (RAG) framework. This system focuses on context-aware answer generation by including semantic search techniques rather than typical keyword-based retrieval.

Our proposed system supports multiple document formats ingestion like PDFs, Text files or Docs. These documents are divided into multiple segments known as embeddings and are stored in a Vector Database which enables faster retrieval. When a user submits a query, relevant segments of the content are forwarded to the Language Model for accurate answer generation.

The system is implemented using Python for backend development, vector indexing techniques for semantic retrieval, pretrained embedding models for representation learning, and Large Language Models for answer generation. The modular architecture ensures scalability and allows the system to be adapted to different document domains with minimal modification.

The expected outcome of this project is a reliable question answering system, improving reliability, reducing ambiguity and minimizing hallucinated outputs usually associated with regular Language Models. The proposed system can be effectively deployed in educational institutions, research environments, legal documentation systems, and enterprise knowledge management platforms to support intelligent, data-driven decision making.

Index Terms: Retrieval-Augmented Generation, Generative Artificial Intelligence, Natural Language Processing, Semantic Search, Vector Database, Document Question Answering, Large Language Models

I. INTRODUCTION

In this digital era, the amount of information generated and stored has been increasing widely and the traditional methods of retrieving documents is not meeting the needs anymore resulting in the need of emergence of new technologies. Generally, most of these documents will be unstructured making it difficult for the users to work on. Conventional methods of search and retrieval often fail in understanding the semantic meaning of user queries and the contextual relevance of the content leading to inaccurate outcomes.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

Recent improvements in Natural Language Processing and Generative AI have brought strong language models. These models can understand text and also create text like humans. Large Language Models work very well in many tasks such as answering questions, summarizing text, and chatting with users [1]. However, the standalone LLMs rely on pretrained knowledge which sometimes is not grounded on documents resulting in hallucinated or outdated responses.

To solve these problems, Retrieval-Augmented Generation (RAG) is used [1]. RAG mixes document searching with text generation models. In this system, useful information is taken from outside documents and given to the language model as extra context. This helps the model give correct answers that match the question. The answers come from the documents, not just from guessing by the model.

Vector indexing is very important for this process. Instead of searching only exact words, the text is changed into numbers called embeddings [3]. These embeddings understand the meaning of words and sentences [3]. Because of this, the system can find related parts of documents using similarity, even if the words are different. This makes the answers more accurate and based on real content.

The main goal of this system is to make documents easy to use, reduce too much information, and give clear and trustworthy answers in a simple way.

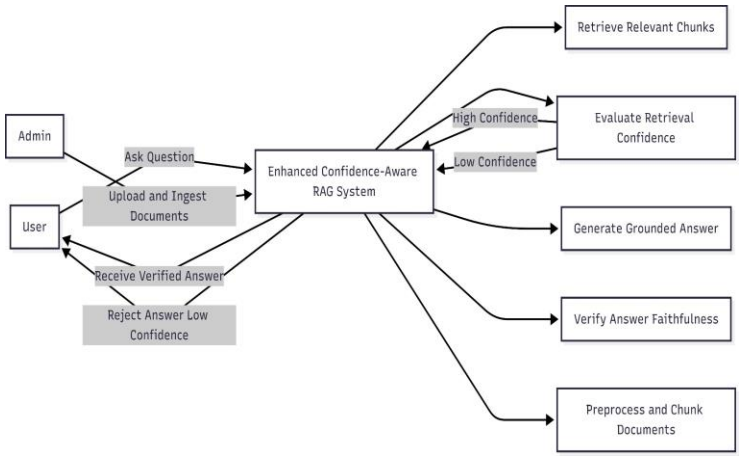

Fig. 1. Use Case Diagram of the Proposed RAG System

II. PROBLEM STATEMENT

There is a rapid observable growth of digital documents and retrieval of accurate and meaningful information has become difficult using traditional keyword-based search methods. Since these methods fails at understanding semantic meaning of user queries resulting in irrelevant responses. Large Language Models can generate responses betterly. But still there is a chance of producing incorrect or hallucinated answers when no sufficient documents are sourced. Therefore, a system that uses both semantic retrieval and context-aware answer generation. This provides accurate, reliability, and document-based responses.

III. OBJECTIVES

- To design and develop a Retrieval-Augmented Document Querying system that combines semantic retrieval with Large Language Models for accurate, document-based question answering.
- To implement vector indexing and similarity-based retrieval techniques for efficiently identifying relevant document segments from large collections.
- To generate context-aware and reliable answers by grounding language model responses strictly on retrieved document content.
- To reduce ambiguity and hallucinated outputs commonly produced by standalone language models through confidence-aware retrieval and context-constrained generation.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

IV. LITERATURE REVIEW

Natural Language Processing has gotten a lot better lately. This means we have good language models now. These language models are great at understanding what people write. They can even write things that sound like they were written by a person.

Natural Language Processing and Generative AI are used to make Language Models. Large Language Models are very good at doing things like question-answering, summarizing etc. The original work done by Lewis and his team introduced the RAG framework [1]. This framework combines searching documents with a language model that generates text one word at a time. The RAG framework is generally used for tasks which needs a lot of knowledge. Lewis and his team found out that when a computer generates responses, it is more accurate if it is based on information from documents that the computer has retrieved. This is better when the computer generates a response on its own.

However, RAG has some problems. It mostly uses the fixed document searching ways. It cannot work well when the context is broken or the retrieved information is not relevant. This RAG framework cannot handle these situations. Karpukhin and other people proposed something called Dense Passage Retrieval [2]. This DPR uses a kind of setup to find the meaning of things in large documents. It does this by making a list of numbers representing what the documents are about. It is better than looking for keywords. Dense Passage Retrieval is really good at finding the documents. It uses a lot of computer power. Sometimes Dense Passage Retrieval just sends the documents it finds to the step without checking if they are really relevant or not. To make it easier to build RAG pipelines we have tools like LangChain and LlamaIndex [4], [5]. These tools give us parts that we can use to add documents, break them down into pieces, make embeddings and get the information we need. They make it simple for people to build and test systems that use retrieval to help with answers.

The problem is that these tools usually use the same way of breaking down text and finding the top results. This results in picking up the text that is not really connected to the content. It also breaks the content into pieces that do not make any meaning. It creates a problem of understanding also. We need to make sure that these models work well. so, some studies use hybrid retrieval methods. These hybrid retrieval methods combine keyword-based search like BM25 with vector-based search to find information. Hybrid retrieval methods are useful for finding related information about the topic you are searching for. Hybrid retrieval methods can make the system more complex and slower. Even with these improvements many problems still exist in RAG systems. It is really tough to keep the context connected in RAG systems. RAG systems also have a time handling results that are not very confident. A lot of RAG systems focus on making search or making text generation better but they do not properly check the text they get before they give answers from RAG systems. Based on earlier studies, this work focuses on improving important parts of the RAG process instead of creating a new structure. By using better document chunking, relevance-based vector indexing, and confidence-limited answer generation, the system tries to give clearer and more reliable answers. Making sure the answers from RAG systems are trustworthy is another challenge.

V. PROPOSED SYSTEM

Our proposed system accepts digital documents as inputs and preprocesses them to meaningful content. These documents are divided into smaller chunks and converted into vector embeddings using pretrained models. These embed- dings are stored in a vector database. This enables efficient similarity-based retrieval. When a user submits a query, the system retrieves the most relevant document chunks. This is achieved by semantic search. The retrieved context is then passed to a Large Language Model to generate accurate, context-aware answers while minimizing unsupported or hallucinated responses.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

A. System Architecture

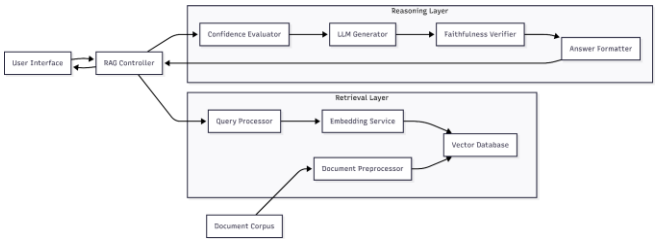

Fig. 2. Workflow of the Enhanced Confidence-Aware RAG System

VI. METHODOLOGY

The new way of doing things uses a method called Retrieval Augmented Document Querying and Context-Aware Answer Generation. This method is supposed to give us reliable answers. It is different from language models because it uses a few different techniques like semantic retrieval and vector indexing to make sure the answers are good and trustworthy. The whole process has a lot of steps. First we take in the documents. Then we clean them up. After that we make embeddings and index the vectors. When we get a question we process it.

The whole process has a lot of steps. First we take in the documents. Then we clean them up. After that we make embeddings and index the vectors. When we get a question we process it. Use the context to generate an answer. We also have some mechanisms to make our system better, than the standard RAG systems. The Retrieval Augmented Document Querying and Context-Aware Answer Generation method is what makes our system special.

A. Dataset

The data used in this project is a collection of documents that we got from many different places. These documents are all different when it comes to how long they are, what they look like, what they say and how they show the real world. The data is stored and retained safely. With this, we do not need to upload it again and again. This helps us to check and use the data for a long time. We use LangChain Document Loaders which supports multiple document formats like PDFs and text documents.

B. Document Preprocessing

When we get the documents we do some work on them before we put them in the system. We do this so that everything is consistent. We can find what we need easily. The work we do on the documents includes getting the text out of PDF files taking out information that we do not need to make sure the spaces between words are right and getting rid of symbols that do not mean anything. We do this to the documents so that the documents are ready, for use and we can find what we need in the documents.

These documents are broken down into pieces that make sense on their own. This way the documents still make sense when we break them up. We do not lose the meaning at the edges. Each piece of the document is treated separately when we try to find and organize the documents.

C. Embedding Generation

Each part of the text is changed into numbers that are close together and we call these numbers embeddings. We use tools that have already been trained to make these embeddings. These embeddings show how the parts of the text are related to each other so we can find parts. The embeddings are made using pretrained text embedding models and the pretrained text embedding models help us find the relationship between the chunks of text, which is really important, for similarity-based retrieval of the text segments. We are using Hugging Face Embeddings for converting document chunks and user queries into numbers.

The user query and the content are put into the format using the same embedding model, which makes them line up. This format lets us look for things even when the content does not have the exact same words, as the user query. The user query

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

and the content can be compared using these embeddings.

D. Vector indexing and Storage

The embeddings that are made are stored in a database that helps us find things that are similar. This database is good at finding things even when we have a lot of document chunks stored in it. We use something called Approximate Nearest Neighbor indexing techniques to make sure we can find what we need quickly even when we have a lot of document chunks. This way the Approximate Nearest Neighbor indexing techniques help us to find what we are looking for in the database of embeddings. We use ChromaDB for this purpose. We also store some metadata information with each vector like the name of the document it comes from, where it is in that document and where the information originally came from. This extra information helps us keep track of things, understand how we got our answers and say where our information comes from when we generate answers with the help of vectors and metadata, like document identifiers and source references.

E. Query Processing and Semantic Retrieval

When a user submits a natural language query the system uses the method to understand it as it does for document indexing. This natural language query is then compared to the content to find how similar they are. The system uses a technique like Cosine Similarity to find this similarity, between the natural language query and the content.

The system gets the relevant parts of the document, which are the top parts that are similar to what we are looking for.

We want to make sure we get the parts so we use a special number to decide if a part is good enough. This number is called a threshold. If a part of the document is similar enough and its score is higher than this threshold then we think it is a piece of evidence to help us generate an answer. The system only uses document parts that are similar enough, that is, the parts that exceed this threshold to help it come up with an answer.

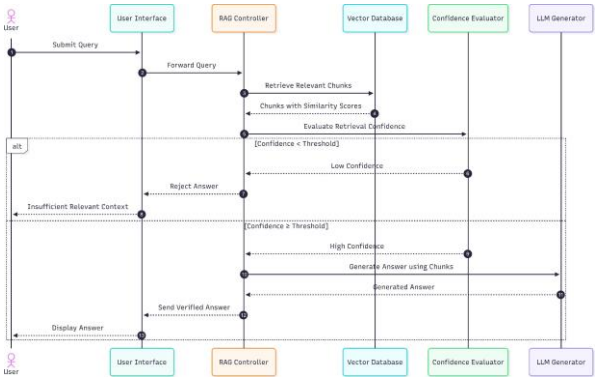

Fig. 3. Sequence Diagram of Query Processing and Answer Generation

F. Context-Aware Answer Generation

The retrieved chunks are forwarded to the Language model and is instructed to strictly generate only the grounded answers reducing unsupported responses. The model stops generating responses when the context is insufficient or does not meet the confidence threshold and requests for clarification or reports the missing of relevant content. This model focuses on explainability rather than relying completely on pretrained knowledge.

VII. PROPOSED ENHANCEMENTS OVER EXISTING RAG SYSTEMS

To differentiate our system, the following enhancements are employed:

A. Confidence-Aware Retrieval

Unlike naive RAG systems which retrieve a fixed number of document chunks regardless of relevance, our proposed system employs a similarity-based confidence threshold. Answer generation is performed only when the retrieved content meets a predefined confidence level, reducing the risk of hallucinated outputs.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

B. Optimized Document Chunking

Overlapping chunks are used to maintain the meaning and relation between different parts of the document. This helps understand the context in a better way and improves accuracy even for large documents.

C. Context-Constrained Generation

The language model is restricted to generate responses only from retrieved document context, therefore minimizing unsupported information.

VIII. IMPLEMENTATION

Our proposed system was implemented using python as the primary programming language. Our model uses a modular architecture. This system supports multiple document formats like PDF, text files and documents allowing users to query information from different sources. During implementation, uploaded documents are processed to extract raw text. This extracted text is divided into overlapping chunks to preserve contextual meaning. These chunks are converted into dense vector representations using embedding models. These embeddings are stored in vector databases and retrieved using distance/similarity techniques. When a user submits a query, the same process is applied to that query and converted into an embedding. Confidence threshold is used to find the reliability of chunks which helps reducing hallucinated outputs. The Large Language Model is constrained strictly to generate grounded responses.

IX. TECHNOLOGY STACK

Programming Language: Python
Utilities: Numpy, Pandas
Document Loaders: Langchain Document Loaders
Embedding models: Sentence Transformers, Huggingface Embeddings
Vector Database: Chromadb
LLM: Huggingface

X. RESULTS

Our proposed system, Retrieval-Augmented Document Querying and Context-Aware Answer Generation was tested to analyze its performance.

The testing of Retrieval-Augmented Document Querying and Context-Aware Answer Generation helped us figure out if it is really doing what it is supposed to do.

Figure 4 shows how the system gives answers by under- standing what the user is actually asking. The system does not just look at the words that the user types. It actually tries to understand what the question means. Then the system finds information from the documents that it has. The system does this so that the answers it gives to the user make sense and are more related to the question that the user asked about the system.

Figure 5 shows what happens to the system when we use a way of breaking things into chunks. When we break things into chunks in a way the system can find more correct information. The system becomes better at giving the answers so the answers are more accurate. The system can also find useful information so we get more of what we need. Because of this the F1-score of the system gets better too. This really shows that breaking things into chunks in a way makes the system work a lot better. The system works better because of chunking and this is what Figure 5 is showing us about the performance of the system, with good chunking.

Figure 6 shows how long the system takes to answer a question. Figure 6 compares the proposed system with keyword-based search and dense retrieval methods. The RAG-based

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

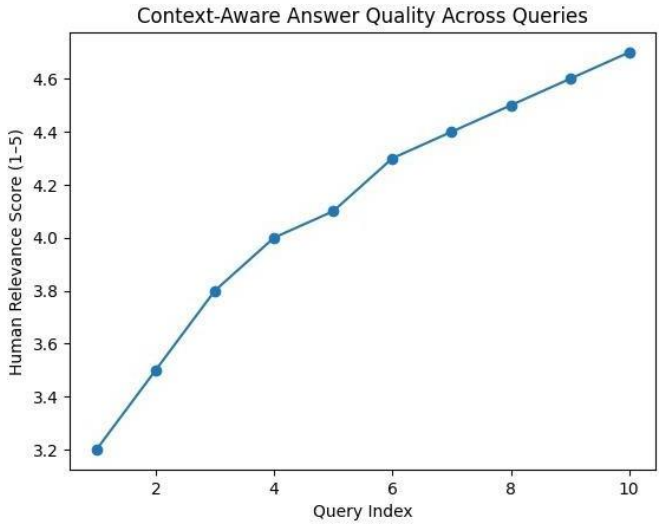

Fig. 4. Context-Aware Answer Quality Across User Queries

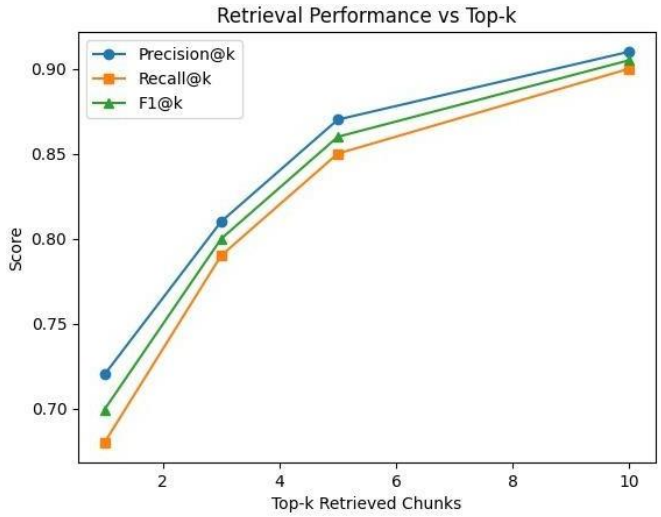

Fig. 5. Retrieval Performance in Terms of Precision, Recall, and F1-Score at Different Top-k Values

system takes a little time because the RAG-based system needs to create embeddings and use a language model to answer the question. The RAG-based system does this work but the response time of the RAG-based system is still reasonable and the RAG-based system does not feel slow when it answers a question.

The results show that the system is able to find the parts of documents and give answers that are clear and meaningful. The RAG system takes a little time to do this but it gives better answers. This is because the RAG system is able to find the parts of documents and give answers that are more accurate and reliable which is what the RAG system is supposed to do. Overall, the results confirm that the proposed system per- forms better than traditional retrieval methods. It gives answers that understand context, are based on documents, and can be trusted. This shows that the system is useful for document- based question answering where correct information is very important.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 2, February 2026

DOI: 10.17148/IJARCCE.2026.15235

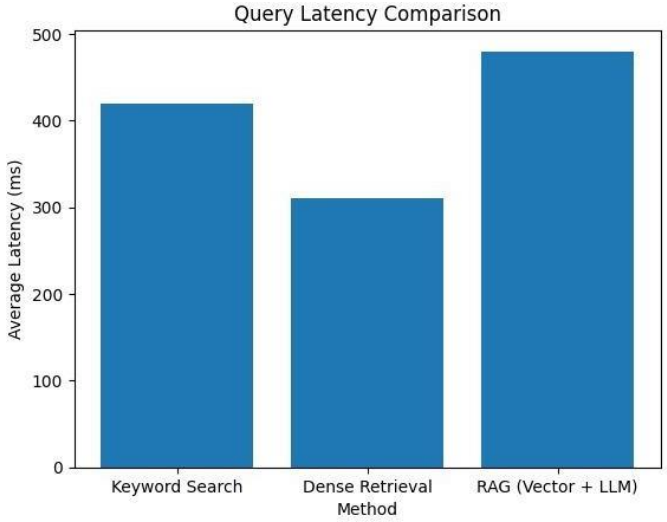

Fig. 6. Comparison of Average Query Latency Across Retrieval Methods

TABLE I
PERFORMANCE COMPARISON ACROSS RETRIEVAL METHODS

Method	Answer Quality	Avg Latency
Keyword Search	Low	Fast
Dense Retrieval	Medium	Moderate
Proposed RAG System	High	Slightly Higher

XI. DISCUSSION

This project is about a system that helps answer questions by looking at documents. A lot of documents are messy and not easy to understand. It is difficult to find the information in them. The system makes it easier by going through the documents and picking out the parts. The system reads the documents, chooses the information that is useful for answering questions from the documents.

When you use search tools they only look for the words you type, they do not really understand what you mean. Some artificial intelligence tools will give you an answer. They do not show you where they got that information from. The search system I am talking about works in a safer way. It looks at the document first. Then it gives you an answer based on what's actually in the document.

The System uses language tool that looks for things. The system stays stable when you have a lot of files. The language tool understands what the user wants to know about the documents. The system is good at understanding what the user actually want. The process is really easy to follow. It has few steps followed internally. First we collect the documents, then clean these documents, converts the text into numbers. These numbers helps us store the meaning of documents. We also do extra checks to avoid errors. We first take the text, split into small parts. Each of these parts is tested to see if it is useful. This is how we reduce the number of replies that the text generator gives for the source text. The system finds the parts we need from files. It does not try to guess what we are looking for. Each part of the system has a specific job to do. All the parts of the system work together well. The system is easy to modify for things because the parts of the system work together so smoothly. This is what makes the system so great for use. The system is very good at finding sections from files.

Each answer shows where it came from. This means that users can see the source of the information. The information came from this source so users can check it. This helps to build trust in the information and users feel more confident in what

DOI: 10.17148/IJARCCE.2026.15235

they are reading about the information and its source. In the end, the system gives clear and correct answers. It depends on real document data. It can help in schools, offices, research, and record work.

XII. CONCLUSION

This system helps find answers from many documents that are messy or not organized. This system works better than normal searches or just using a regular language model. It has two things, one to search documents and the other for answer generation. This helps it handle many files and give clear answers. It only gives one answer at a time. This approach stops wrong or made-up answers. It makes sure answers relate to the documents. The system can find exact parts of documents and give answers that make sense. It looks at the context to understand questions. In short, the system gives a simple and reliable way to answer questions from documents. It uses careful search and checks answers with sources. It can be helpful in schools, research, legal work, and companies.

REFERENCES

[1]. Lewis, P., Perez, E., Piktus, A., et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Facebook AI Research, 2020.

[2]. Karpukhin, V., Oguz, B., Min, S., et al., "Dense Passage Retrieval for Open-Domain Question Answering," Proceedings of the EMNLP Conference, 2020.

[3]. Reimers, N., and Gurevych, I., "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proceedings of the EMNLP Confer- ence, 2019.

[4]. LangChain Documentation, "Building Retrieval-Augmented Generation Pipelines," LangChain Official Documentation.

[5]. LlamaIndex Documentation, "Data Indexing and Retrieval for Large Language Model Applications," LlamaIndex Official Documentation.

[6]. Johnson, J., Douze, M., and Je´gou, H., "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, 2019.

