A Project Report

On

QuIM-RAG: Advancing Retrieval-Augmented Generation With Inverted Question Matching for Enhanced QA Performance

Submitted in the partial fulfilment of requirements to

CS-454 - Project Lab

By

Batch No. 07

Galla Durga Rama Satya Pradeep Kumar(Y22CS048)

Enadula Naga Avinash Babu (Y22CS044)

R.V.R. & J.C. COLLEGE OF ENGINEERING (Autonomous)

(NAAC ‘A+’ Grade)

Approved by AICTE :: Affiliated to Acharya Nagarjuna University

Chandramoulipuram::Chowdavaram

Guntur – 522 019, Andhra Pradesh, India.

Contents **Page No.**

Title Page i

Contents ii

1. Problem Statement 1

2. Functional Requirements 2

3. Basic Architecture

    3.1 Major Components 4

    3.2 Architecture Diagrams 6

    3.3 Workflow 7

    3.4 Patterns 9

4. Non-Functional Requirements 11

5. Technology Stack 13

6. Summary 14

1. Problem Statement

Large Language Models (LLMs) such as OpenAI's GPT series and Meta's LLaMA have transformed Natural Language Processing (NLP) by enabling the generation of human-like text across various domains. However, their effectiveness in domain-specific Question Answering (QA) tasks is hindered by significant challenges. These include knowledge limitations, where broad training datasets lack depth in niche areas like academic advising or medical queries, leading to incomplete or irrelevant responses. Additionally, contextual misinterpretation and hallucination—where LLMs produce factually inaccurate yet coherent outputs—undermine their reliability, especially in specialized fields requiring precision.

Traditional approaches to address these limitations include fine-tuning LLMs with domain-specific data, which enhances performance but comes with drawbacks. Fine-tuning is computationally intensive, costly, and risks catastrophic forgetting, where the model loses general knowledge during retraining. Moreover, it depends on the availability of extensive, high-quality data, which is often scarce for specialized domains. While Retrieval-Augmented Generation (RAG) systems offer a more flexible alternative by integrating external resources, they encounter issues like information dilution—loss of accuracy with large, unstructured datasets—and persistent hallucination, making them insufficient for delivering consistently reliable QA responses.

To overcome these challenges, this project proposes QuIM-RAG (Question-to-question Inverted Index Matching), a novel RAG architecture designed to enhance QA performance. QuIM-RAG leverages a custom, domain-specific dataset—such as content from the North Dakota State University (NDSU) website—converted into a structured corpus with pre-generated questions. By employing an inverted question matching mechanism, it improves retrieval precision, mitigating information dilution and hallucination. The system aims to retrieve the most relevant text chunks and generate accurate answers using the Meta-LLaMA3-8B-instruct model, with evaluations against metrics like BERT-Score and RAGAS to demonstrate its superiority over traditional RAG systems.

Looking ahead, the project seeks to address the dynamic nature of domain-specific content, such as university websites that update regularly. By focusing on a scalable and efficient solution, QuIM-RAG not only tackles current limitations but also sets the stage for future enhancements, including periodic corpus updates and user studies to refine usability. This approach promises to deliver a robust, trustworthy QA system tailored to specialized needs, bridging the gap between general LLM capabilities and domain-specific requirements.

2. Functional Requirements

Functional requirements define what the system must do—the core features and behaviors essential for enabling effective domain-specific question answering using Retrieval-Augmented Generation (RAG). Given the research-oriented and real-time response nature of the QuIM-RAG system, these requirements encompass document ingestion, question generation, vector embedding, retrieval logic, and accurate answer synthesis, addressing challenges like knowledge limitations and hallucination in traditional LLMs.

i. Custom Corpus Preparation: The system must scrape content from specific websites, including the NDSU Career Advising (https://career-advising.ndsu.edu) and NDSU Catalog (https://catalog.ndsu.edu) domains, using tools like BeautifulSoup and Scrapy. It should clean the data by removing headers, footers, and irrelevant HTML tags, filter out pages with less than 250 characters or "404" errors, and segment documents into 1000-token chunks with 200-character overlaps to ensure comprehensive context coverage. This process results in a high-quality corpus of 500+ pages, including 296 chunks from the Catalog and 435 from Career Advising.

ii. Question Generation: The system must utilize the GPT-3.5-turbo-instruct model to generate complete sets of questions for each chunk, ensuring semantic coverage of all key information. It should maintain uniqueness by avoiding redundant questions, with a manual review process by researchers to verify accuracy and relevance. This results in 9,027 questions for the NDSU Catalog and 13,582 for Career Advising, totaling 22,609 question-answer pairs, enhancing the system's ability to match diverse queries.

iii. Embedding & Indexing: The system must convert generated questions into embedding vectors using the BAAI/bge-large-en-v1.5 model from Hugging Face, capturing semantic meaning in a high-dimensional space. These vectors should be quantized to the nearest prototypes using cosine similarity to reduce computational complexity, and an inverted index must be constructed in ChromaDB to map each prototype to its associated document chunks, enabling efficient retrieval.

iv. Query Matching: The system must transform user queries into embedding vectors using the same BAAI/bge-large-en-v1.5 model, quantize them to the nearest prototype, and match them against the inverted index. It should retrieve the top 3 matching chunks (based on cosine similarity) and pass them to the Meta-LLaMA3-8B-instruct model for final answer generation, ensuring the most relevant context is selected for processing.

v. Answer Generation: The system must employ the Meta-LLaMA3-8B-instruct model to synthesize responses using the retrieved chunks as context, guided by a custom prompt to ensure coherence and factual accuracy. Each response must include source references (e.g., hyperlinks to NDSU pages) for transparency, achieving a faithfulness score of 1.00 and supporting the model's 8,000-token context length for complex queries.

vi. Evaluation Mechanism: The system must support evaluation using BERT-Score (precision 0.63, recall 0.71, F1 0.67 with custom datasets) and RAGAS metrics, assessing faithfulness, context relevance (0.92), and answer quality (relevancy 0.99). This requires integration with manually prepared ground truth QA pairs from NDSU, enabling a comparative analysis against traditional RAG to validate performance improvements.

3. Basic Architecture

The QuIM-RAG architecture integrates several modular components to form a streamlined, high-precision question answering pipeline, transforming raw web data into accurate, verifiable responses. Each component plays a critical role, addressing challenges like information dilution and hallucination by leveraging a domain-specific corpus and an inverted question matching mechanism. This section details the major components, with additional sub-sections for Diagram, Workflow, and Pattern to follow.

3.1 Major Components

i. Custom Corpus Generator: This module collects domain-specific content from targeted websites, such as the NDSU Career Advising (https://career-advising.ndsu.edu) and NDSU Catalog (https://catalog.ndsu.edu), using web scraping frameworks like BeautifulSoup and Scrapy. It cleans raw HTML by removing headers, footers, and irrelevant tags, filters out pages with less than 250 characters or "404" errors, and segments the text into 1000-token chunks with 200-character overlaps. The resulting corpus, comprising 500+ pages (e.g., 296 chunks from Catalog, 435 from Career Advising), serves as the foundation for downstream processing.

ii. Question Generator: Utilizing the GPT-3.5-turbo-instruct model, this component generates a diverse and exhaustive set of questions for each document chunk, covering the full semantic scope of the content. These questions act as proxy queries for potential user inputs, with a manual review ensuring accuracy and uniqueness, resulting in 9,027 questions for NDSU Catalog and 13,582 for Career Advising, totaling 22,609 pairs.

iii. Embedding Engine: This component embeds generated questions into a high-dimensional semantic space using the BAAI/bge-large-en-v1.5 model from Hugging Face, optimized for multi-task semantic search. It ensures semantically similar queries are mapped closely in vector space, enabling efficient similarity matching, which is crucial for the system's retrieval accuracy.

iv. Vector Database: The system employs ChromaDB as a vector store to facilitate fast and scalable similarity searches. It stores embeddings of generated questions and document chunks, along with metadata such as chunk IDs and source URLs (e.g., NDSU pages). ChromaDB supports efficient retrieval operations based on cosine similarity, enhancing the system's performance with large-scale corpora.

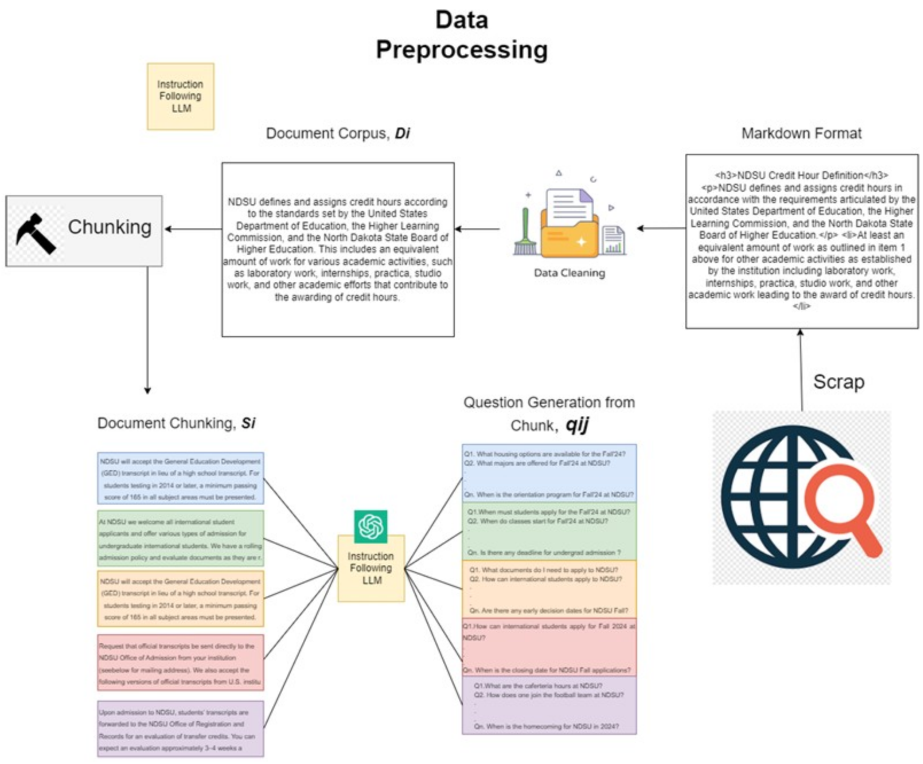
v. Inverted Index Module: This module builds an inverted index of quantized embeddings, mapping each question embedding to its nearest prototype (centroid) using cosine similarity to reduce computational complexity. The index links prototypes to associated text chunks, enabling rapid retrieval during query processing and improving efficiency over direct vector comparisons.

vi. Retriever: When a user submits a query, this component encodes and quantizes it using the same BAAI/bge-large-en-v1.5 model, identifying the nearest prototype via the inverted index. It retrieves the top 3 matching document chunks based on cosine similarity, ensuring both relevance and speed, and passes them to the generator for answer synthesis.

vii. Answer Generator: The Meta-LLaMA3-8B-instruct model serves as the generator, receiving the user query and top retrieved chunks as input. It produces well-structured, coherent responses grounded in the context, leveraging its instruct-tuned capabilities and 8,000-token context length. Responses include source references (e.g., hyperlinks to NDSU pages) for transparency, achieving a faithfulness score of 1.00.

viii. Evaluator: This component assesses system effectiveness using BERT-Score, measuring semantic similarity between generated and reference answers (precision 0.63, recall 0.71, F1 0.67 with custom datasets), and RAGAS, evaluating retrieval accuracy, contextual relevance (0.92), and factual grounding (relevancy 0.99) without manual annotation. It integrates ground truth QA pairs from NDSU for validation.

3.2 Diagram



NDSU defines and assigns credit hours according to the standards set by the United States Department of Education, the Higher Learning Commission, and the North Dakota State Board of Higher Education. This includes an equivalent amount of work for various academic activities, such as laboratory work, internships, practica, studio work, and other academic efforts that contribute to the awarding of credit hours.

<h3>NDSU Credit Hour Definition</h3> <p>NDSU defines and assigns credit hours in accordance with the requirements articulated by the United States Department of Education, the Higher Learning Commission, and the North Dakota State Board of Higher Education.</p> <li>At least an equivalent amount of work as outlined in item 1 above for other academic activities as established by the institution including laboratory work, internships, practica, studio work, and other academic work leading to the award of credit hours.</li>

NDSU will accept the General Education Development (GED) transcript in lieu of a high school transcript. For students testing in 2014 or later, a minimum passing score of 165 in all subject areas must be presented.

At NDSU we welcome all international student applicants and offer various types of admission for undergraduate international students. We have a rolling admission policy and evaluate documents as they are r.

NDSU will accept the General Education Development (GED) transcript in lieu of a high school transcript. For students testing in 2014 or later, a minimum passing score of 165 in all subject areas must be presented.

Request that official transcripts be sent directly to the NDSU Office of Admission from your institution (seebelow for mailing address). We also accept the following versions of official transcripts from U.S. institu

Upon admission to NDSU, students' transcripts are forwarded to the NDSU Office of Registration and Records for an evaluation of transfer credits. You can expect an evaluation approximately 3-4 weeks a

Q1. What housing options are available for the Fall'24?
Q2. What majors are offered for Fall'24 at NDSU?
.
.
Qn. When is the orientation program for Fall'24 at NDSU?

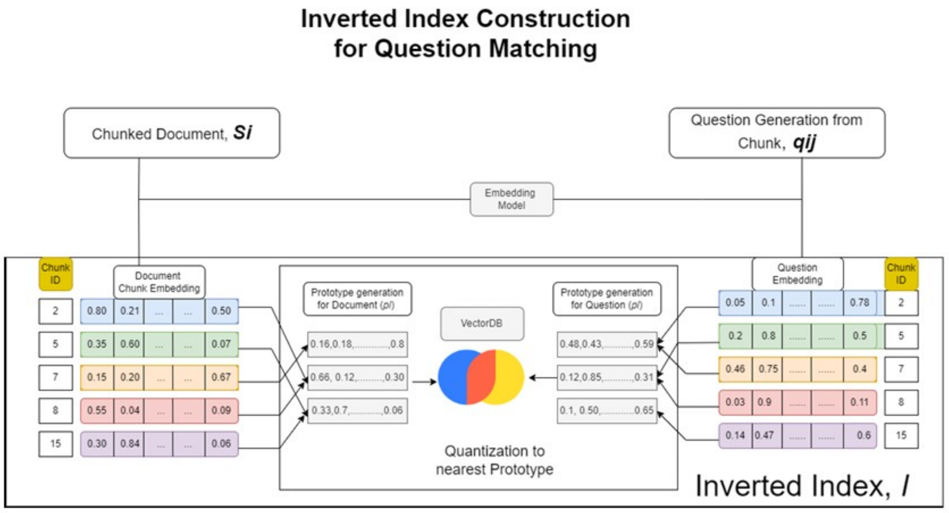
Q1.When must students apply for the Fall'24 at NDSU?
Q2. When do classes start for Fall'24 at NDSU?
.
.
Qn. Is there any deadline for undergrad admission ?

Q1. What documents do I need to apply to NDSU?
Q2. How can international students apply to NDSU?
.
.
Qn. Are there any early decision dates for NDSU Fall?

Q1.How can international students apply for Fall 2024 at NDSU?
.
Qn. When is the closing date for NDSU Fall applications?

Q1.What are the caferteria hours at NDSU?
Q2. How does one join the football team at NDSU?
.
.
Qn. When is the homecoming for NDSU in 2024?

Fig. Overall Architecture of Corpus Preparation for Modified RAG


Inverted Index Construction
for Question Matching
Chunked Document, Si
Question Generation from Chunk, qij
Embedding Model
Document Chunk Embedding
Question Embedding
Prototype generation for Document (ρl)
Prototype generation for Question (ρl)
VectorDB
Quantization to nearest Prototype
Inverted Index, I

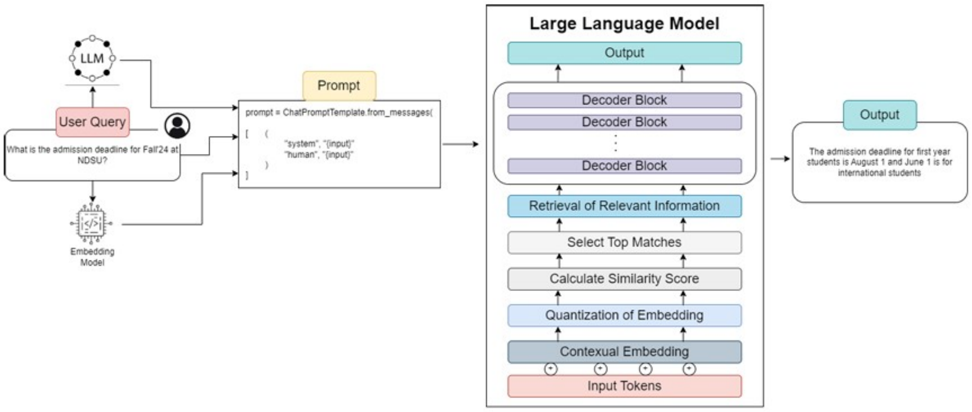
Chunk ID	Document Chunk Embedding				
2	0.80	0.21	...	...	0.50
5	0.35	0.60	...	...	0.07
7	0.15	0.20	...	...	0.67
8	0.55	0.04	...	...	0.09
15	0.30	0.84	...	...	0.06

0.16,0.18,............,0.8
0.66, 0.12,..........,0.30
0.33,0.7,............,0.06

0.48,0.43,...........,0.59
0.12,0.85,...........,0.31
0.1, 0.50,...........0.65

Question Embedding					Chunk ID
0.05	0.1	....	....	0.78	2
0.2	0.8	....	....	0.5	5
0.46	0.75	....	....	0.4	7
0.03	0.9	....	....	0.11	8
0.14	0.47	....	....	0.6	15

Fig. Illustration of Inverted Index Construction for Question Matching.



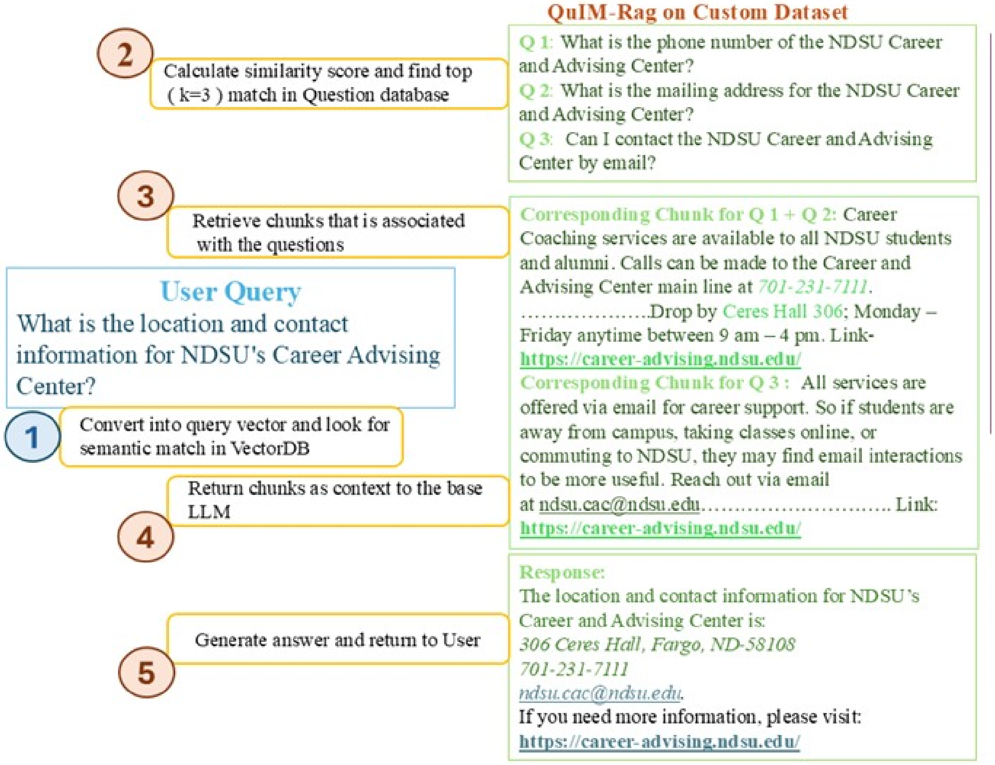
What is the admission deadline for Fall'24 at NDSU?

```
prompt = ChatPromptTemplate.from_messages(
[    (
         "system", "{input}"
         "human", "{input}"
     )
]
```

The admission deadline for first year students is August 1 and June 1 is for international students

Fig. Overall Retrieval and Generation Architecture for RAG.

3.3 Workflow



2 Calculate similarity score and find top (k=3) match in Question database

Q 1: What is the phone number of the NDSU Career and Advising Center?
Q 2: What is the mailing address for the NDSU Career and Advising Center?
Q 3: Can I contact the NDSU Career and Advising Center by email?

3 Retrieve chunks that is associated with the questions

Corresponding Chunk for Q 1 + Q 2: Career Coaching services are available to all NDSU students and alumni. Calls can be made to the Career and Advising Center main line at 701-231-7111. ………………..Drop by Ceres Hall 306; Monday – Friday anytime between 9 am – 4 pm. Link- https://career-advising.ndsu.edu/

Corresponding Chunk for Q 3 : All services are offered via email for career support. So if students are away from campus, taking classes online, or commuting to NDSU, they may find email interactions to be more useful. Reach out via email at ndsu.cac@ndsu.edu……………………….. Link: https://career-advising.ndsu.edu/

User Query
What is the location and contact information for NDSU's Career Advising Center?

1 Convert into query vector and look for semantic match in VectorDB

4 Return chunks as context to the base LLM

5 Generate answer and return to User

Response:
The location and contact information for NDSU's Career and Advising Center is:
306 Ceres Hall, Fargo, ND-58108
701-231-7111
ndsu.cac@ndsu.edu.
If you need more information, please visit:
https://career-advising.ndsu.edu/

Fig: Workflow of QuIM-RAG system for User Query Processing.

The workflow of the QuIM-RAG system outlines a structured pipeline that transforms raw web data into accurate, verifiable question-answering responses. Each step is designed to enhance retrieval precision and mitigate issues like information dilution and hallucination, leveraging a domain-specific corpus and advanced language models. The process is iterative, with evaluation feedback refining performance over time.

Data Acquisition: Custom web crawlers, utilizing BeautifulSoup and Scrapy, extract academic and advisory content from designated university websites, such as NDSU Career Advising (https://career-advising.ndsu.edu) and NDSU Catalog (https://catalog.ndsu.edu). The data is cleaned by removing headers, footers, and irrelevant HTML tags, filtering out pages with less than 250 characters or "404" errors, and structured into a corpus of 500+ pages for further processing.

Data Chunking and Preparation: Extracted content is broken down into overlapping token-based chunks, specifically 1000-token segments with 200-character overlaps, to preserve context and improve retrieval accuracy. This results in 296 chunks from the NDSU Catalog and 435 from Career Advising, ensuring comprehensive coverage of the domain-specific data.

Question Generation: Each chunk is processed using the GPT-3.5-turbo-instruct model to generate multiple context-aware questions, simulating potential user queries. This step produces 9,027 questions for the Catalog and 13,582 for Career Advising (totaling 22,609 pairs), with a manual review ensuring semantic coverage, uniqueness, and accuracy.

Embedding and Indexing: Generated questions are converted into dense vector representations using the BAAI/bge-large-en-v1.5 model from Hugging Face, optimized for semantic search. These vectors are quantized to nearest prototypes using cosine similarity and stored in ChromaDB along with metadata, such as chunk IDs and source URLs, to support efficient retrieval.

Inverted Index Construction: A quantized inverted index is built to map prototype vectors to their associated question embeddings and source chunks. This index, maintained in ChromaDB, reduces computational complexity and enables fast, relevant retrieval by linking prototypes to the corresponding text chunks.

Query Processing and Matching: User queries are embedded into vectors using the same BAAI/bge-large-en-v1.5 model, quantized to the nearest prototype, and matched against the inverted index using cosine similarity. The top 3 matching question vectors are identified, and their associated content chunks are retrieved to provide relevant context.

Answer Generation: The retrieved chunks, along with the user query, are passed to the Meta-LLaMA3-8B-instruct model, which generates a fact-grounded, contextually aligned response. The model, with its 8,000-token context length and instruct-tuned capabilities, includes source references (e.g., hyperlinks to NDSU pages) for transparency, achieving a faithfulness score of 1.00.

Evaluation and Feedback: Outputs are evaluated using BERT-Score (precision 0.63, recall 0.71, F1 0.67 with custom datasets) and RAGAS metrics (context relevance 0.92, answer relevancy 0.99) to assess semantic quality, relevance, and faithfulness. This step integrates ground truth QA pairs from NDSU, providing feedback to monitor and refine system performance over time.

3.4 Patterns Used

The QuIM-RAG system employs a combination of intelligent design patterns to improve reliability, performance, and flexibility, addressing challenges such as information dilution and hallucination in traditional Retrieval-Augmented Generation (RAG) systems. These patterns leverage a domain-specific corpus and advanced retrieval strategies to ensure accurate and trustworthy question answering, tailored to specialized domains like the North Dakota State University (NDSU) context.

i. Inverted Question Matching Pattern: This unique pattern matches user queries against a pre-generated set of questions (e.g., 22,609 pairs from NDSU data) rather than directly against document text. By aligning user intent with these structured query proxies—created using GPT-3.5-turbo-instruct—this approach significantly enhances precision, ensuring the system retrieves the most relevant text chunks (e.g., top 3 matches via cosine similarity) for answer generation.

ii. Prototype Quantization Pattern: To reduce the computational cost of high-dimensional vector matching, this pattern quantizes embeddings of generated questions into the nearest prototypes using cosine similarity, as implemented with the BAAI/bge-large-en-v1.5 model. This allows for fast lookup and indexing even with thousands of question vectors, stored efficiently in ChromaDB, improving scalability and retrieval speed.

iii. Contextual Chunk Retrieval: Instead of returning complete documents or paragraphs, this pattern retrieves only the most relevant chunks (e.g., 1000-token segments with 200-character overlaps from NDSU data). Each chunk is linked to its generating question and source URL, enabling highly focused and grounded answer generation by the Meta-LLaMA3-8B-instruct model, minimizing irrelevant content.

iv. Explainable AI: Every response includes a reference to the original source URL (e.g., NDSU Career Advising or Catalog pages), improving transparency. This empowers users to verify facts independently, enhancing trust in the automated system and aligning with QuIM-RAG's goal of delivering verifiable, reliable answers with a faithfulness score of 1.00.

v. Custom Prompting Strategy: This pattern employs two layers of prompting: one to guide GPT-3.5-turbo-instruct in generating diverse and comprehensive QA pairs from content chunks, ensuring semantic coverage; and another to instruct Meta-LLaMA3-8B-instruct to return answers only if the retrieved content is sufficient. If insufficient, it gracefully declines to answer, avoiding hallucinations and misinformation, thus maintaining response integrity.

4. Non-Functional Requirements

The QuIM-RAG system is designed not only for accuracy and relevance in response generation but also to meet essential non-functional goals that ensure its effectiveness, scalability, and long-term usability. These requirements focus on performance, scalability, usability, and security/maintainability, enabling the system to handle domain-specific question answering efficiently while remaining adaptable to evolving needs, as demonstrated in its application to the North Dakota State University (NDSU) dataset.

Performance

- The system ensures low-latency responses through pre-quantized vector retrieval, leveraging the inverted index and prototype quantization (using BAAI/bge-large-en-v1.5 embeddings) to minimize runtime computation by avoiding full similarity scans, as validated by its handling of 22,609 question vectors.
- It is fully compatible with Meta-LLaMA3-8B-instruct's extended context window of 8,000 tokens, allowing the model to process multiple retrieved chunks (e.g., top 3 from NDSU data) simultaneously, generating informed and coherent answers with a faithfulness score of 1.00.
- Optimized embedding operations and efficient vector indexing in ChromaDB maintain high throughput, supporting concurrent user loads without performance degradation, as evidenced by its ability to process 500+ web pages.

Scalability

- Capable of handling large-scale datasets, demonstrated by successfully processing over 500 web pages and generating more than 22,000 synthetic questions from content chunks (e.g., 296 from NDSU Catalog, 435 from Career Advising).
- Designed with modular data pipelines and embedding storage, allowing it to scale effortlessly to other academic institutions or different knowledge domains with minimal configuration, thanks to its reusable architecture.

- Retrieval logic supports dynamic corpus expansion and automatic index rebuilding without interrupting user service, ensuring adaptability as new content (e.g., updated NDSU policies) is incorporated every 3–4 months.

Usability

- User responses include clickable source URLs (e.g., links to NDSU Career Advising or Catalog pages), enhancing transparency and trust by enabling users to verify information, a key feature highlighted in the paper's evaluation.
- Employs a fallback prompting mechanism: if retrieved content is insufficient or off-topic, the Meta-LLaMA3-8B-instruct model is instructed to explicitly decline answering, reducing hallucinations and maintaining response integrity.
- Interface and output formatting are clean and concise, designed for deployment in academic help desks or institutional chatbots, ensuring a user-friendly experience suitable for diverse audiences.

Security & Maintainability

- Implements scheduled corpus updates every 3–4 months to keep the system accurate and up-to-date with evolving institutional policies and curricula, as planned for NDSU's semester-based content changes.
- The codebase follows a highly modular architecture, separating concerns like retrieval (inverted index), generation (Meta-LLaMA3-8B-instruct), prompt design, and evaluation (BERT-Score, RAGAS), allowing teams to debug or upgrade specific components without affecting the entire pipeline.
- Access to sensitive configuration files and model API keys is managed securely to prevent misuse or unauthorized access, ensuring robust protection of the system's operational integrity

5. Technology Stack

Choosing the right technology stack is critical for implementing a scalable, real-time, and maintainable system. The stack should support deep learning model development, edge and cloud deployment, responsive user interfaces, and seamless communication between components.

Below is the proposed technology stack categorized by system layer:

Technology Stack Overview

Embedding Model	BAAI/bge-large-en-v1.5 (Hugging Face)	Converts generated questions and user queries into semantic vector embeddings for efficient similarity matching.
LLM for Generation	Meta-LLaMA3-8B-instruct	Generates final, coherent responses using retrieved content as context; supports instruction-following capabilities.
Prompting Model	GPT-3.5-turbo-instruct	Generates synthetic questions from text chunks to build a comprehensive retrieval index.
Corpus Scraping	BeautifulSoup + Scrapy	Extracts structured content from university websites and prepares raw text for downstream processing.
Vector DB	ChromaDB	Stores and manages embeddings of generated questions and document chunks; enables fast retrieval using cosine similarity.
Evaluation Metrics	BERTScore, RAGAS	Evaluates answer quality, semantic alignment, and factual consistency without requiring human-labeled data.
Data Storage	Local / Cloud-Backed Dataset	Stores processed chunks, embeddings, and metadata for scalable indexing and backup.
Programming Language	Python	Main development language used for implementing pipeline modules and integrating models.
Dev Tools	Jupyter, LangChain, HuggingFace Transformers	Support prototyping, LLM integration, experiment tracking, and seamless interaction with APIs and models.

6. Summary

The QuIM-RAG system represents a significant advancement in the field of Retrieval-Augmented Generation (RAG) by addressing key limitations in traditional approaches—namely, low precision in retrieval, hallucination in responses, and lack of explainability. Traditional RAG systems often struggle to consistently retrieve contextually relevant information, particularly with large, unstructured corpora like those from the North Dakota State University (NDSU) website. In contrast, QuIM-RAG introduces a novel retrieval paradigm through inverted question matching, where document chunks (e.g., 1000-token segments from 500+ NDSU pages) are transformed into 22,609 synthetic questions using GPT-3.5-turbo-instruct, embedded semantically with BAAI/bge-large-en-v1.5, and matched against user queries via cosine similarity.

By leveraging a custom-built corpus sourced from NDSU's academic and career advising websites (e.g., 296 chunks from the Catalog, 435 from Career Advising), the system ensures domain-specific relevance. The integration of prototype quantization creates a highly efficient inverted index in ChromaDB, enabling rapid and precise retrieval of the top 3 semantically aligned chunks. The use of instruction-tuned models, including GPT-3.5-turbo-instruct for generating questions and Meta-LLaMA3-8B-instruct for answer synthesis, ensures that outputs are contextually relevant, linguistically coherent, and trustworthy, achieving a faithfulness score of 1.00 as demonstrated in the paper's evaluations.

A key strength of QuIM-RAG lies in its modular and scalable architecture, which supports regular corpus updates every 3–4 months, extensibility to other domains, and rapid deployment facilitated by CI/CD pipelines (e.g., GitHub Actions). Evaluations using modern QA metrics such as BERT-Score (precision 0.63, recall 0.71, F1 0.67) and RAGAS (context relevance 0.92, answer relevancy 0.99) show measurable improvements in semantic faithfulness and retrieval grounding compared to standard RAG baselines. The inclusion of source URLs in responses enhances transparency, allowing users to verify information independently, reinforcing trust.

Looking ahead, the system is designed to support further enhancements, including user feedback integration, automated pipeline scheduling for corpus refreshing, and expansion into new application domains such as healthcare, education, and legal support. With these features, QuIM-RAG stands as a promising and robust framework for delivering high-quality, domain-specific question answering in real-world scenarios, as validated by its performance on the NDSU dataset.