

A TERM PAPER REPORT
ON
QuIM-RAG: Advancing Retrieval-Augmented
Generation With Inverted Question Matching
for Enhanced QA Performance

Submitted in the partial fulfillment of requirements to

CS- 363 - Term Paper

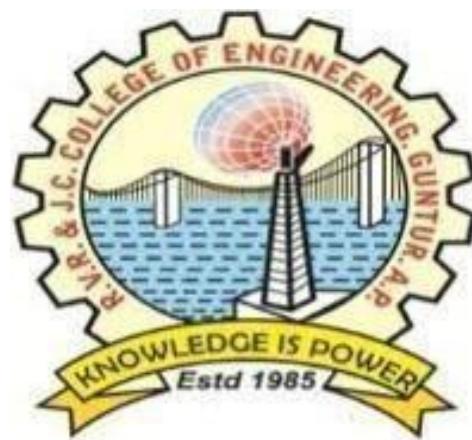
By

Batch No. 07

GALLA DURGA RAMA SATYA PRADEEP KUMAR (Y22CS048)

ENADULA NAGA AVINASH BABU (Y22CS044)

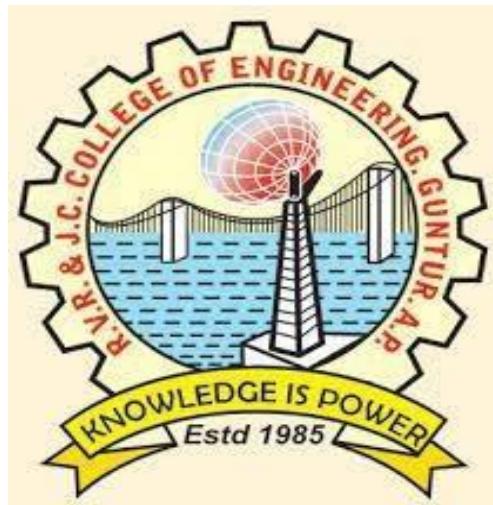
ELIKA POLERU (Y22CS042)



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.

R.V.R & J.C. COLLEGE OF ENGINEERING (Autonomous)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that this Term Paper titled “QuIM-RAG: Advancing Retrieval-Augmented Generation With Inverted Question Matching for Enhanced QA Performance” is the study conducted by **GALLA DURGA RAMA SATYA PRADEEP KUMAR (Y22CS048)**, **ENADULA NAGA AVINASH BABU (Y22CS044)**, **ELIKA POLERU (Y22CS042)** submitted in partial fulfillment of the requirements to CS 363- Term Paper during the Academic Year 2024-2025.

Dr. M. Sreelatha

Term Paper Guide

Smt. N. Zareena

Term Paper In- Charge

Dr. M. Sreelatha

Prof. & Head,CSE

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Term Paper titled “**QuIM-RAG: Advancing Retrieval-Augmented Generation With Inverted Question Matching for Enhanced QA Performance**”.

We are deeply grateful to our guide, **Dr. M. Sreelatha**, for her unwavering guidance, insightful feedback and encouragement. Her expertise and dedication have been instrumental in shaping the direction and quality of this research.

We would like to express our sincere gratitude to **Smt. N. Zareena**, In-Charge for our term paper. Her expertise, guidance and support were instrumental in the successful completion of this research.

We express our sincere thanks to **Dr. M. Sreelatha**, Head of the Department of Computer Science and Engineering for her encouragement, support, commitment to enhance research experience.

We are very much thankful to **Dr. Kolla Srinivas**, Principal, for proving this supportive environment and to engage in research activities.

Finally we submit our heartfelt thanks to lab staff in the Department of Computer Science and Engineering for their cooperation, support for providing administrative support and technical assistance during selection.

GALLA DURGA RAMA SATYA PRADEEP KUMAR (Y22CS048)

ENADULA NAGA AVINASH BABU (Y22CS044)

ELIKA POLERU (Y22CS042)

ABSTRACT

Large Language Models (LLMs) have redefined natural language processing by generating human-like text, yet their application to domain-specific question answering (QA) is limited by issues such as outdated knowledge, contextual misalignment, and hallucination—producing plausible but inaccurate responses. Retrieval-Augmented Generation (RAG) addresses these challenges by integrating external knowledge sources, enhancing LLMs’ ability to provide contextually relevant answers. However, traditional RAG systems falter when handling large datasets, often retrieving irrelevant or diluted information, which compromises answer precision. To overcome these shortcomings, we propose QuIM-RAG (Question-to-Question Inverted Index Matching), a novel RAG framework that enhances QA performance through a meticulously curated domain-specific dataset and an innovative retrieval mechanism. Our approach involves constructing a custom corpus from over 500 pages of North Dakota State University’s (NDSU) Career Advising and Catalog websites, scraped using BeautifulSoup and Scrapy, segmented into 731 chunks, and transformed into 22,609 manually validated question-answer pairs via GPT-3.5-turbo-instruct. This dataset, enriched with source links, ensures high-quality, verifiable content tailored to university-specific queries, mitigating the information dilution prevalent in traditional RAG systems.

QuIM-RAG introduces an inverted question matching strategy, where generated questions from the corpus are embedded using the BAAI/bge-large-en-v1.5 model, quantized into prototypes, and indexed for efficient retrieval. Upon receiving a user query, it is embedded, matched against the index using cosine similarity, and the top three relevant chunks are retrieved as context for the Meta-LLaMA3-8B-instruct model to generate a coherent response. Evaluated against traditional RAG using BERT-Score and RAGAS metrics, QuIM-RAG significantly outperforms on custom datasets in all aspects than traditional RAG on traditional datasets. These gains reflect QuIM-RAG’s ability to align retrieved context precisely with user intent, reducing hallucinations and enhancing trustworthiness through source transparency. Implemented on an open-source platform, QuIM-RAG offers a scalable, reliable solution for domain-specific QA, with future plans to incorporate periodic corpus updates and user studies to further refine its efficiency.

CONTENTS

	Page No.
Title Page	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Contents	v
List of Tables	vi
List of Figures	vii
List of Abbreviations	
1. Introduction	
1.1 Background	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Limitations of existing techniques	5
2. Literature Review	
2.1 Fine-Tuning or Retrieval?	6
2.2 Retrieving Supporting Evidence	8
2.3 Internet-Augmented Language Models	10
2.4 Empower Large Language Model	12
2.5 BERTScore: Evaluating Text Generation with BERT	13
2.6 RAGAS	15
3. Methodologies Used	
3.1 Architecture	17
3.2 Datasets to be used	20
4. Proposed Model	
4.1 QA Problem for a Limited Corpus	22
4.2 Inverted Question Matching for Relevant Chunks	22
4.3 Answer Generation for User Query	30
5. Discussion on proposed output	
5.1 Experiments and Results	32
5.2 Performace Analysis of Results	36
6. Conclusion and Future Work	40
7. References	43

LIST OF TABLES

Table No.	Table Name	Page No.
2.1.2	Performance Comparison of Fine-Tuned Models	7
2.6.1	Comparison of Evaluation Strategies	16
4.2.1	Example Chunks	23
4.2.3	Generated Embeddings	25
4.2.4	Quantized Embeddings	26
4.2.5	Similarity Scores	28
5.1.2	Pipeline Overview	33
5.2.2	Comparison of Performance metrics	38

LIST OF FIGURES

Figure No.	Figure Name	Page No.
2.1.1	A visualization of the knowledge injection framework	6
2.1.3	Relative Accuracy Gain of Knowledge Injection Methods	7
2.2.1	Self-detecting hallucination in LLMs.	9
2.3.1	Schematic representation of Internet-augmented retrieval	11
2.4.1	Interaction FrameWork	12
2.5.1	Computation of the recall metric R_{BERT}	14
3.1.1	Corpus Preparation Architecture	17
3.1.2	Inverted Index Construction Architecture	18
3.1.3	Retreival and Answer Generation Architecture	19
3.2.1	NDSU Career Advising Website	21
3.2.2	NDSU Catalog Website	21
4.2.2	Prompt for Dataset	24
4.2.6	Model Algorithm	29
4.3.1	Prompt for RAG	30
5.1.1	Summary of DataSet	33
5.2.1	WorkFlow of QuIM-RAG and Traditional RAG	36

LIST OF ABBREVIATIONS

LLM	-	Large Language Model
RAG	-	Retrieval-Augmented Generation
QuIM-RAG	-	Question-to-Question Inverted Matching for RAG
QA	-	Question Answering
NLP	-	Natural Language Processing
IQM	-	Inverted Question Matching
BM25	-	Best Matching 25
DPR	-	Dense Passage Retrieval
T5	-	Text-to-Text Transfer Transformer
GPT	-	Generative Pre-trained Transformer
BLEU	-	Bilingual Evaluation Understudy
ROUGE	-	Recall-Oriented Understudy for Gisting Evaluation
RAGAS	-	Retrieval-Augmented Generation Assessment Score
BERT	-	Bidirectional Encoder Representations from keemlknlenleTransformers

1. INTRODUCTION

1.1 Background

In recent years, large language models (LLMs) have revolutionized the field of natural language processing (NLP), enabling advancements in question-answering (QA), text summarization, and conversational AI. Models such as BERT, T5, and GPT have demonstrated exceptional abilities in generating fluent and contextually rich responses. However, despite their progress, these models have a fundamental limitation: they rely on pre-trained knowledge, meaning they cannot access or incorporate new information beyond their training data. As a result, they often generate outdated, incomplete, or factually incorrect responses, particularly in knowledge-intensive applications such as medical diagnosis, legal analysis, and real-time information retrieval.

To address this limitation, Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm that enhances language models with external knowledge sources. Unlike traditional generative models that solely rely on parametric memory (i.e., the information stored within their weights), RAG integrates a retrieval mechanism that fetches relevant documents from external databases, search engines, or knowledge corpora. This retrieved content is then used as additional context for the generation process, improving factual accuracy and reducing hallucinations. RAG models have proven highly effective in open-domain question-answering (QA), customer support automation, and academic research assistance, where access to reliable and up-to-date information is crucial.

Despite its advantages, RAG still faces significant challenges, particularly in the retrieval process. The effectiveness of the entire system depends on the accuracy of the retrieved documents—if the retrieval module selects irrelevant, redundant, or misleading information, the generated response will also be suboptimal.

The retrieval process plays a critical role in ensuring that generative AI systems produce accurate, informative, and contextually relevant outputs. Improving query-document matching, retrieval efficiency, and semantic understanding is essential for advancing retrieval-augmented generation and making it more reliable for real-world applications.

This research explores how QuIM-RAG improves retrieval in RAG systems, demonstrating its impact on better knowledge grounding, reduced hallucination, and optimized response generation in generative AI applications.

1.2 Problem Statement

Traditional RAG models often suffer from inefficient retrieval mechanisms, which directly impact the accuracy and reliability of generated responses.

One of the main issues with current retrieval methods is that they treat queries and documents as fixed text representations, rather than considering how well a document truly answers the query. BM25 relies on keyword matching, which does not always capture the actual meaning of the question. On the other hand, dense retrieval methods use embeddings, but they often retrieve documents that contain similar words without directly addressing the query. As a result, RAG models may end up using partially related information, making their responses incomplete or misleading. Additionally, retrieval systems often fetch too many irrelevant documents, making it harder for the model to generate a clear and accurate response.

To solve these issues, a better retrieval system is needed—one that selects documents more accurately and ensures they truly match the user's question. QuIM-RAG (Question-to-Question Inverted Matching for Retrieval-Augmented Generation) aims to improve retrieval precision by introducing a Question-to-Question Inverted Matching (QuIM) technique. Instead of directly comparing queries to documents, this approach reformulates the query into different question variations and checks which documents best answer these variations. This method improves query understanding, document selection, and retrieval efficiency, helping AI models generate more accurate, fact-based, and contextually relevant responses.

1.3 Objectives

The main goal of QuIM-RAG (Question-to-Question Inverted Matching for Retrieval-Augmented Generation) is to improve the accuracy and relevance of document retrieval in RAG-based systems. By refining how queries interact with the retrieval process, QuIM-RAG ensures that retrieved documents are highly relevant, contextually aligned, and useful for generating accurate responses. The specific objectives of this research are as follows:

- **Enhance Retrieval Accuracy :** Improve how queries are matched with documents by using Question-to-Question Inverted Matching (QuIM) to ensure that retrieved documents truly answer the query instead of just containing related terms. This approach helps reduce false positives in retrieval, ensuring that only the most relevant sources are used for response generation.
- **Improve Query Understanding :** Reformulate queries into different variations to capture their meaning better, allowing the retrieval system to identify the most informative documents. This dynamic reformulation process ensures that even complex or ambiguous queries can be accurately matched with contextually relevant sources.
- **Reduce Hallucinations in Generated Responses :** Ensure that AI-generated responses are based on reliable, retrieved information, minimizing cases where the model produces misleading or incorrect details. By improving retrieval precision, QuIM-RAG enhances the trustworthiness and accuracy of AI-generated content.
- **Optimize Retrieval Efficiency :** Reduce retrieval noise by filtering out irrelevant or redundant documents, making the system faster and more efficient without sacrificing accuracy. This ensures that the system maintains high accuracy while reducing computational overhead, making it more suitable for real-time applications.
- **Improve Generalization Across Different Domains:** Make the retrieval process adaptable to various topics, datasets, and applications, ensuring that QuIM-RAG performs well across diverse knowledge-intensive tasks. This allows it to be effectively applied in healthcare, legal analysis, academic research, and enterprise AI solutions.
- **Enhance Practical Usability:** Develop a retrieval system that is computationally efficient and can be integrated into real-world AI applications, such as chatbots, research assistants, and enterprise knowledge management systems. By ensuring ease of deployment, QuIM-RAG enables organizations to leverage enhanced retrieval without significant infrastructure modifications.

By achieving these objectives, QuIM-RAG will improve the reliability, accuracy, and efficiency of retrieval-augmented systems, making them more useful for real-world

applications that require factual correctness and context-aware responses.

1.4 Limitations of Existing Systems

Despite the advancements in Retrieval-Augmented Generation (RAG), existing retrieval mechanisms still suffer from several limitations that affect their efficiency, accuracy, and reliability. These shortcomings lead to poor document selection, which in turn results in hallucinations, incomplete responses, or misleading information in AI-generated outputs. The key limitations of current retrieval systems are outlined below:

- **Query-Document Mismatch :** any existing retrieval systems fail to properly align a user's query with the most relevant documents. Sparse retrieval models like BM25 rely on exact keyword matching, often missing semantically similar documents that use different wording. Dense retrieval models, while better at capturing semantics, still struggle with understanding query intent, sometimes retrieving documents that contain similar words but do not actually answer the question.
- **Over-Reliance on Static Matching Techniques :** Current retrieval techniques, whether sparse (BM25) or dense (DPR, ColBERT), treat queries as fixed representations, meaning they do not dynamically adapt to different wording styles, paraphrases, or variations in user intent. This lack of flexibility results in suboptimal document retrieval, reducing the effectiveness of RAG-based models.
- **Retrieval Noise and Redundant Documents :** Many retrieval systems return excessive numbers of documents, many of which are only loosely related to the query. This unnecessary retrieval of redundant or irrelevant information increases the computational cost of the system while forcing the generative model to process unnecessary or misleading content, ultimately degrading response quality.
- **Hallucinations in Generated Responses :** When irrelevant or low-quality documents are retrieved, AI models may fabricate or misinterpret information, leading to hallucinations—false or misleading statements that are not backed by actual sources. Since RAG models rely heavily on retrieved knowledge, poor document selection directly leads to incorrect AI-generated

responses.

- **Computational Inefficiency and Latency Issues :** Many retrieval-based NLP systems depend on multi-stage ranking and reranking pipelines, which introduce high computational overhead and latency. These inefficiencies make it difficult to scale RAG models for real-time applications, such as customer support bots, virtual assistants, or search engines, where fast response times are critical.
- **Poor Generalization Across Domains :** Most retrieval systems are trained on specific datasets and struggle to adapt to different domains. A retrieval model that works well for Wikipedia-based QA might fail when applied to scientific, legal, or medical texts, as domain-specific terminology and retrieval patterns are not well understood by generic models. This lack of adaptability limits the broader applicability of RAG models in specialized fields.

2. Literature Review

2.1 Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs - Oded Ovadia, Menachem Brief, Moshik Mishaeli and Oren Elisha

The paper explores the effectiveness of fine-tuning vs. retrieval-augmented generation (RAG) for injecting knowledge into large language models (LLMs). The authors analyze how LLMs acquire and retain knowledge, comparing unsupervised fine-tuning with retrieval-based approaches. Their experiments show that RAG consistently outperforms fine-tuning in knowledge-intensive tasks, especially for incorporating new or updated information. They also highlight that fine-tuning struggles to teach models new factual knowledge, while RAG provides dynamic and adaptable knowledge retrieval without modifying model weights.

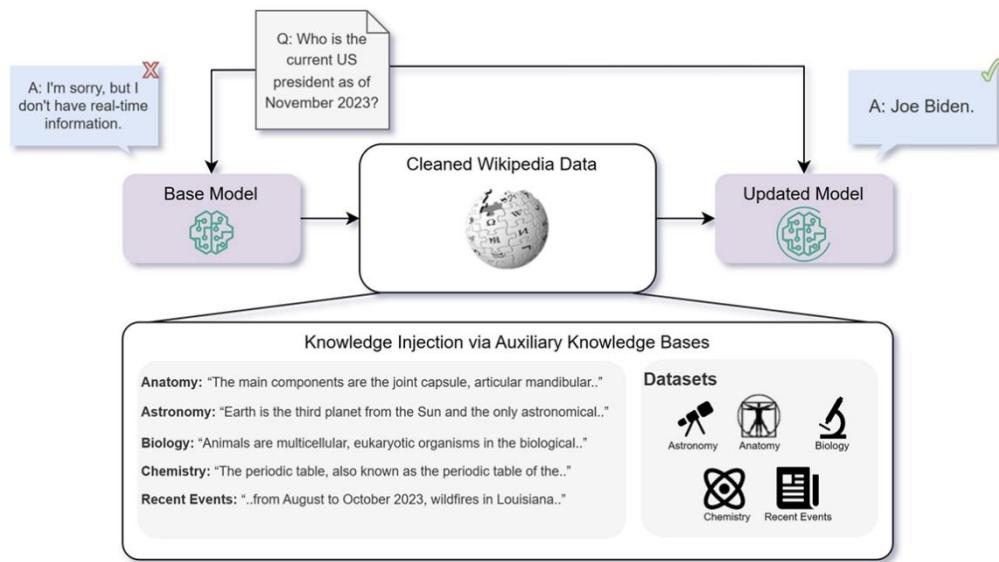


Fig 2.1.1 : A visualization of the knowledge injection framework.

The figure 2.1.1 visually represents the two primary methods for injecting knowledge into Large Language Models (LLMs):

Fine-Tuning : This method involves modifying the model's internal parameters by training it further on new knowledge datasets. The model attempts to store this information within its weights. However, fine-tuning suffers from issues like catastrophic forgetting, requiring repeated exposure to facts to ensure retention.

Retrieval-Augmented Generation (RAG) : Unlike fine-tuning, RAG does not modify the model itself. Instead, it uses an external knowledge base where relevant documents are retrieved during inference and provided as additional context. This approach allows models to generate responses based on real-time, updated knowledge

without changing their internal memory.

The key takeaway from Figure 1 is that RAG allows LLMs to dynamically retrieve and use external knowledge without requiring weight updates, making it more efficient and scalable compared to fine-tuning.

	Base model	Base model + RAG	FT-reg	FT-par	FT-reg + RAG	FT-par + RAG
Mistral 7B	0.481	0.875	0.504	0.588	0.810	0.830
Llama2 7B	0.353	0.585	0.219	0.392	0.326	0.520
Orca2 7B	0.456	0.876	0.511	0.566	0.820	0.826

Fig 2.1.2 : Performance Comparison of Fine-Tuned Models

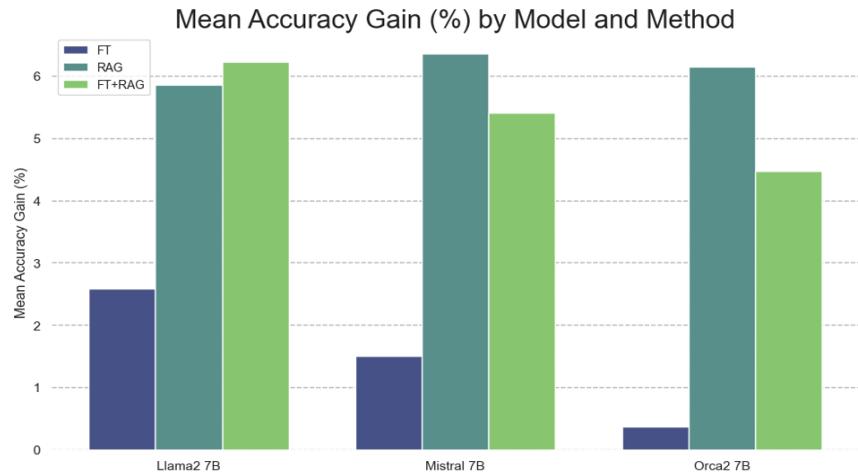


Fig 2.1.3 : Relative Accuracy Gain of Knowledge Injection Methods

This figure compares the performance improvements of different knowledge injection methods across multiple tasks. The y-axis represents relative accuracy gain, while different configurations of fine-tuning and RAG are compared across various LLMs. The key observations include:

RAG Consistently Outperforms Fine-Tuning

- Models using only fine-tuning show small improvements over their base versions.
- RAG alone provides a significantly higher accuracy gain than fine-tuning across all tested models and tasks.
- This suggests that fine-tuning alone is not an effective method for injecting knowledge into LLMs.

Fine-Tuning + RAG Yields Mixed Results

- In some cases, fine-tuned models combined with RAG perform slightly better than RAG alone.
- However, this improvement is not consistent, and in some cases, RAG alone is sufficient for optimal performance.
- This suggests that fine-tuning might help in certain scenarios but does not fundamentally replace the need for effective retrieval mechanisms.

Performance Differences Across LLMs

- The Mistral 7B model benefits the most from RAG, showing the highest relative accuracy improvement.
- The Llama2 7B model struggles with fine-tuning and shows only minor improvements, reinforcing that some models may not effectively retain new knowledge through fine-tuning alone.
- The Orca2 7B model, which is instruction-tuned, shows a smaller but consistent benefit from RAG, indicating that instruction tuning may already improve some retrieval behaviors but still benefits from external knowledge sources.

This study provides strong evidence supporting the importance of retrieval-based enhancements like those proposed in QuIM-RAG (Question-to-Question Inverted Matching for RAG). Since fine-tuning struggles with new knowledge acquisition, an improved query-document matching technique like QuIM-RAG's Q2IM can further refine retrieval accuracy, addressing one of the key limitations identified in the paper.

2.2 Retrieving Supporting Evidence for LLMs Generated Answers - Siqing Huo, Negar Arabzadeh and Charles L. A. Clarke

This paper investigates the challenge of hallucination in Large Language Models (LLMs) and explores methods to verify AI-generated answers against external sources. Since LLMs can produce factually incorrect but confident responses, the authors propose an approach where retrieved evidence is used to validate LLM outputs. The research focuses on leveraging information retrieval techniques to detect inconsistencies, ultimately aiming to improve trustworthiness in AI-generated content.

The study evaluates different retrieval mechanisms (BM25, neural retrieval

models, and hybrid approaches) and tests whether LLMs can self-detect their hallucinations by comparing their generated answers with retrieved supporting evidence. Experimental results indicate that retrieval-based verification improves accuracy, but LLMs still fail in 20–30% of cases, highlighting the need for further improvements in retrieval precision and response validation.

LLMs do not have a built-in fact-checking mechanism and often generate responses that appear confident but may be incorrect. To address this, external retrieval methods are used to verify AI-generated answers. However, even when provided with supporting evidence, LLMs correctly classify hallucinations only 70–80% of the time, meaning they cannot be fully trusted to self-validate. The paper evaluates different retrieval methods, including BM25 (keyword-based retrieval), a neural retrieval stack (SPLADE + ANCE + MonoT5 + DuoT5), and Neural retrieval with a Reader (which extracts precise answers from retrieved documents). While the neural retrieval stack performs best, it still struggles with numerical inconsistencies and partial contradictions in retrieved evidence. Additionally, the study introduces a combined query strategy, where instead of searching only with the original question, the LLM’s generated answer is added to the query. This approach retrieves more contextually relevant passages, improving the verification process and reducing instances where retrieved documents contradict the AI’s response.

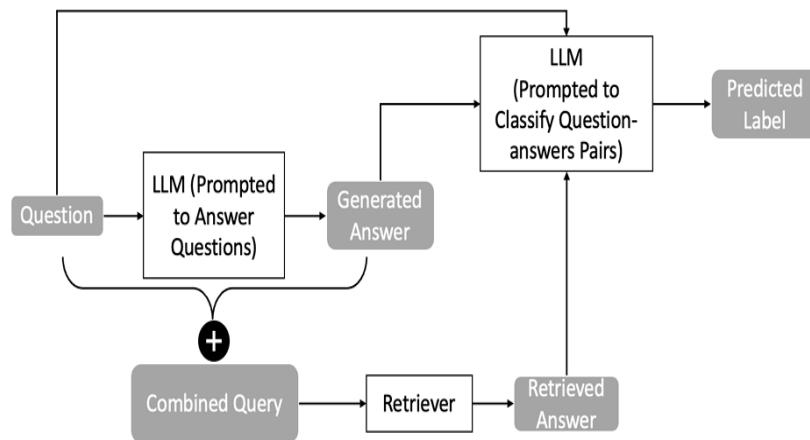


Fig 2.2.1 : Self-detecting hallucination in LLMs.

The diagram outlines the pipeline for verifying AI-generated answers:

- **Step 1:** The LLM generates an answer to a given question.
- **Step 2:** The question and generated answer are combined into a new search query.

- **Step 3:** An external retrieval system searches for relevant supporting documents.
- **Step 4:** The retrieved evidence is compared with the LLM's response to classify it as:
 - **Yes** (answer is supported)
 - **No** (answer contradicts retrieved evidence)
 - **Not Related** (retrieved document is unrelated)

This research confirms that retrieval-based verification is essential for improving AI-generated responses.

The query expansion strategy (combining the question and answer before retrieval) aligns with QuIM-RAG's goal of refining query-document alignment through Question-to-Question Inverted Matching (Q2IM).

The study highlights retrieval noise and inconsistencies in verification, reinforcing the need for improved retrieval mechanisms like QuIM-RAG, which can ensure better document matching and validation.

2.3 Internet-Augmented Language Models Through Few-Shot Prompting for Open-Domain Question Answering - Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, Nikolai Grigorev

The paper explores how few-shot prompting can be used to integrate external knowledge into Large-Scale Language Models (LSLMs) without requiring fine-tuning. The study focuses on retrieval-augmented question-answering (QA) using the Internet (Google Search) as a dynamic knowledge source. Instead of relying solely on a model's pre-trained knowledge, the proposed approach retrieves real-time information from the web and conditions the language model on this external evidence.

The authors find that Internet-augmented language models outperform traditional closed-book models, particularly in knowledge-intensive tasks. The results suggest that instead of focusing solely on increasing model size, improvements in retrieval, prompting techniques, and inference-time compute can lead to better performance and factual accuracy and reliable knowledge integration across domains.

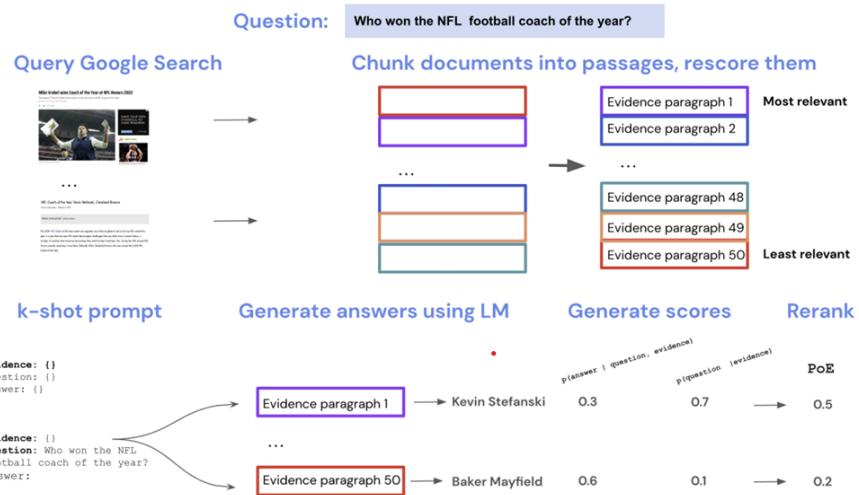


Fig 2.3.1 : schematic representation of an Internet-augmented retrieval process
It outlines how external web search enhances the response quality of an LLM by incorporating retrieved evidence. The process consists of the following key steps:

- **Query Google Search** : The system queries a search engine (e.g., Google) to retrieve relevant documents based on the input question.
- **Chunk and Rescore Documents** : Retrieved documents are broken into smaller passages, which are then scored based on relevance.
- **K-shot Prompting** : Selected evidence paragraphs are included in a prompt given to the LLM to guide answer generation.
- **Generate Answers Using LM** : The LLM generates possible answers using the provided evidence.
- **Generate Scores** : Each generated answer is assigned probability scores based on:
 - $P(\text{answer}|\text{question}, \text{evidence})$: Probability of the answer being correct given the question and supporting evidence.
 - $P(\text{question}|\text{evidence})$:Probability that the evidence actually answers the given question.
- **Rerank Answers** – The final ranking is computed using a PoE model, which balances these probabilities to determine the most reliable answer.

The pipeline ensures that answers are grounded in external knowledge and reduces hallucinations by basing responses on retrieved factual information rather than relying solely on the model's internal memory.

The use of retrieval augmentation with dynamic query expansion aligns with

QuIM-RAG's focus on improving retrieval precision.

2.4 Empower Large Language Model to Perform Better on Industrial Domain-Specific Question Answering - Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang

Large Language Models (LLMs) perform well in general-purpose question answering (QA), but they often struggle with domain-specific tasks due to a lack of specialized knowledge. This paper proposes a model interaction paradigm to improve LLMs for industrial domain-specific QA without requiring extensive fine-tuning. The authors introduce the MSQA dataset, which contains 32,000 QA pairs from Microsoft Q&A forums, focusing on Microsoft products and IT technical problems. The study also highlights limitations in retrieval-based methods, showing that they often fail to capture complex domain-specific information, leading to incorrect or incomplete answers.

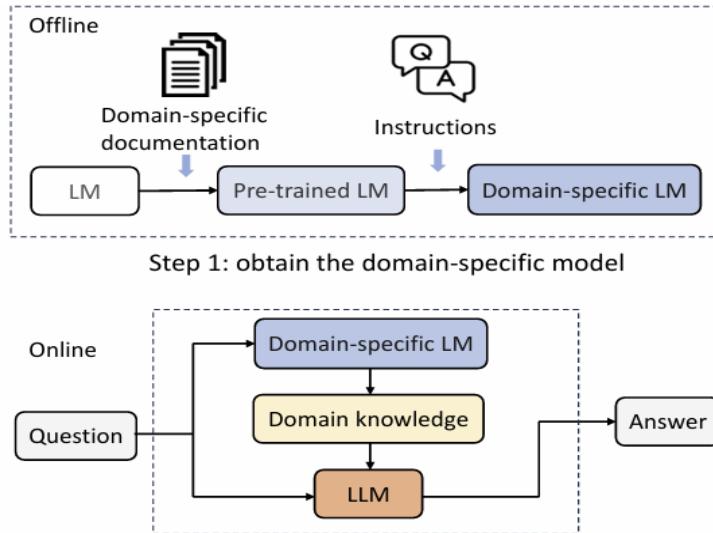


Fig 2.4.1: Interaction FrameWork

Step 1: A small LLM is fine-tuned on domain-specific data (e.g., Azure documentation).

Step 2: This domain-specific model assists a larger LLM at runtime by providing relevant knowledge to generate better answers.

Advantage: The approach enhances LLM performance without expensive full fine-tuning or retrieval-based limitations.

The paper introduces MSQA, a domain-specific question-answering (QA) benchmark that focuses on cloud computing and Microsoft technologies, enabling a more precise evaluation of LLMs in industry-specific tasks. The study highlights the limitations of retrieval-based methods, showing that traditional models like BM25 and DPR struggle with complex queries, often retrieving irrelevant documents that match keywords but lack true contextual relevance. Additionally, input length constraints prevent these models from capturing comprehensive domain knowledge, leading to incomplete answers. To overcome these challenges, the authors propose a Model Interaction Paradigm, where instead of retrieving external data, a smaller domain-specific model is pre-trained on Azure documentation. This expert assistant model enhances the LLM's responses by providing accurate, domain-specific knowledge without requiring extensive fine-tuning. The paper also introduces improved evaluation metrics for long-form QA, incorporating semantic-overlap-based measures (BERT-Score, SIM) and keyword hit rate (KHR) to assess factual correctness more effectively. These contributions collectively aim to improve LLM performance in specialized industrial domains while reducing hallucinations and response inaccuracies.

This study highlights the shortcomings of traditional retrieval-based QA and proposes a domain-specific augmentation method to improve LLM accuracy. QuIM-RAG (Question-to-Question Inverted Matching for RAG) aligns with this by improving retrieval precision, ensuring that retrieved documents are contextually aligned with the query, reducing hallucinations, and enhancing response reliability in domain-specific QA tasks.

2.5 BERTScore: Evaluating Text Generation with BERT - Tianyi Zhang , Varsha Kishore , Felix Wu , Kilian Q. Weinberger , and Yoav Artzi

The study introduces BERTScore, an automatic evaluation metric for natural language generation tasks, such as machine translation and text summarization. Unlike traditional n-gram-based metrics like BLEU and ROUGE, BERTScore leverages contextualized embeddings from BERT to compute similarity between generated text and reference text. Instead of relying on exact word matches, BERTScore measures semantic similarity using cosine similarity between token embeddings. The authors demonstrate that BERTScore correlates more strongly with human judgments

compared to existing evaluation metrics.

Key Contributions

1. Limitations of Traditional Metrics (BLEU, METEOR, ROUGE)
 - These methods focus on exact n-gram overlap, failing to capture paraphrases and semantic similarity.
 - They do not account for word importance differences or long-distance dependencies.
2. BERTScore: A Contextual Embedding-Based Metric
 - Uses pre-trained BERT embeddings to compare each token in the candidate sentence with the reference sentence.
 - Computes similarity using cosine similarity instead of exact word matches.
 - Can be weighted using inverse document frequency (IDF) to prioritize important words.
3. Correlation with Human Evaluations
 - BERTScore shows higher correlation with human judgments in machine translation and image captioning compared to BLEU and ROUGE.
 - The metric is more robust to adversarial paraphrasing compared to traditional methods.
4. Performance in Adversarial Tasks
 - Tested on PAWS (adversarial paraphrase detection dataset) and performed better than BLEU and METEOR.
 - Demonstrates strong performance in sentence-level evaluation and model selection tasks.

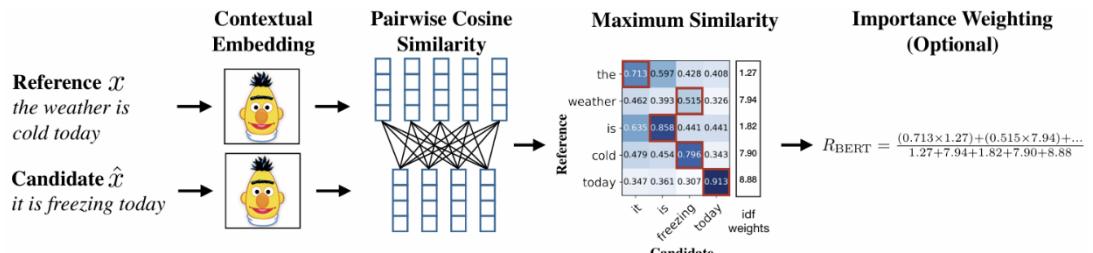


Fig 2.5.1 : Computation of the recall metric RBERT

Computation of BERTScore Recall

- Illustrates how BERTScore measures token similarity between the reference and candidate sentence.
 - Each word in the candidate sentence is mapped to its most similar word in the reference sentence using BERT embeddings.
 - Cosine similarity values between each pair of words are computed and used to derive the final recall score.
 - Optionally, IDF weighting is applied to prioritize important words.
- BERTScore's use of **contextual embeddings** aligns with **QuIM-RAG's focus on improving retrieval accuracy** by enhancing **semantic matching** between queries and documents.
- The findings highlight the **importance of embedding-based similarity** over **exact word-matching**, which is critical for **retrieval-augmented models like QuIM-RAG**.
- The study also reinforces the need for **better evaluation metrics** when assessing **generated responses in NLP systems**.

2.6 RAGAS: Automated Evaluation of Retrieval-Augmented Generation - Shahul Es, Jithin James, Luis Espinosa-Anke, Steven Schockaert

The paper introduces RAGAS (Retrieval-Augmented Generation Assessment), a framework for evaluating Retrieval-Augmented Generation (RAG) pipelines without relying on reference answers. The framework assesses three key aspects of a RAG system: faithfulness, answer relevance, and context relevance. Unlike traditional evaluation metrics that require human-annotated datasets, RAGAS provides fully automated, reference-free evaluation, making it a valuable tool for rapid assessment of RAG architectures.

Key Contributions

1. Challenges in Evaluating RAG Systems
 - Standard metrics like **perplexity or BLEU** do not effectively measure retrieval-based model performance.
 - **Ground-truth reference answers** are often unavailable in real-world

RAG applications.

2. RAGAS: A New Evaluation Framework

- Introduces a **fully automated, reference-free** approach to assess RAG-generated responses.
- Evaluates responses based on **three key dimensions**:
 - **Faithfulness**: Is the generated answer supported by retrieved context?
 - **Answer Relevance**: Does the answer correctly address the question?
 - **Context Relevance**: Is the retrieved document highly relevant to the query?

3. Integration with LLMs and RAG Pipelines

- RAGAS integrates with **LlamaIndex** and **Langchain**, enabling developers to use it for **real-time evaluation of retrieval models**.

4. WikiEval Dataset for RAG Evaluation

- The paper introduces **WikiEval**, a dataset with human-annotated evaluations of **faithfulness, answer relevance, and context relevance**.
- Human evaluations show that RAGAS aligns **closely with human judgments**, outperforming **GPTScore** and **GPTRanking**.

	Faith.	Ans. Rel.	Cont. Rel.
RAGAs	0.95	0.78	0.70
GPT Score	0.72	0.52	0.63
GPT Ranking	0.54	0.40	0.52

Table 2.6.1 : Comparison of Evaluation Strategies

- RAGAS achieves 95% agreement with human annotators for faithfulness.
- GPTScore and GPTRanking perform worse, failing to detect hallucinated or irrelevant responses.
- Human evaluations confirm that RAGAS better detects factual inconsistencies in RAG models compared to standard LLM-based evaluation.

3. Methodologies Used

3.1 Architecture

The QuIM-RAG (Question-to-Question Inverted Matching for Retrieval-Augmented Generation) architecture is designed to enhance the retrieval phase of RAG-based systems by improving query-document alignment through inverted question matching. The architecture consists of the following key components:

1. Corpus Preparation Architecture

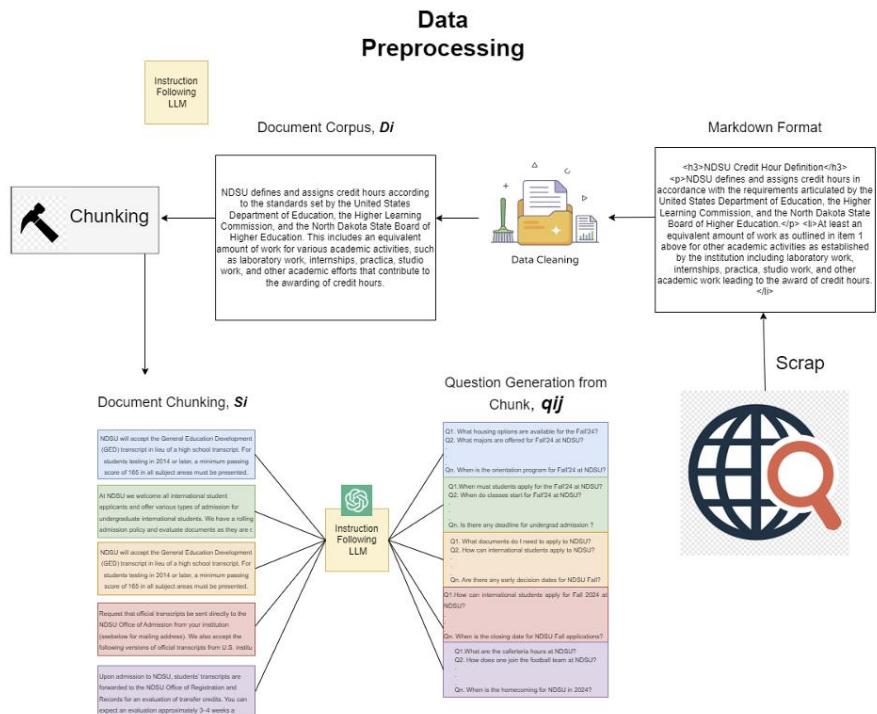


Fig 3.1.1 : Corpus Preparation Architecture

The Corpus Preparation Architecture of QuIM-RAG is designed to enhance retrieval efficiency by transforming raw document text into structured, retrievable question-answer pairs. This preparation ensures that the retrieval system can precisely match user queries with the most relevant knowledge, leading to better response generation. The key steps in this process are as follows:

- **Data Collection:** Structured web scraping is performed on relevant sources.
- **Chunking and Preprocessing:** The scraped content is segmented into 1000-token chunks with overlapping windows for context preservation.

- **Question Generation:** Instead of directly embedding chunks, multiple question-answer pairs are generated for each chunk using GPT-3.5-turbo-instruct.
- **Embedding and Indexing:** Generated questions are vectorized and stored in an efficient embedding index.

2. Retrieval and Matching Architecture

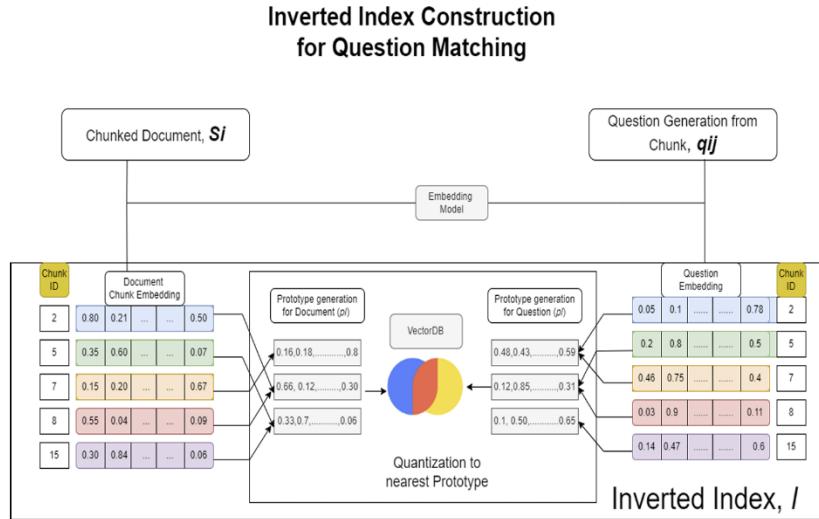


Fig 3.1.2: Inverted Index Construction Architecture

- **Embedding Questions & Chunks::**
 - Generated questions are converted into vector embeddings using BAAI/bge-large-en-v1.5 from Hugging Face.
 - Each chunk also has an associated embedding.
- **Quantization & Inverted Indexing:**
 - Embedding vectors are quantized to prototype vectors, reducing storage and improving retrieval speed.
 - The system creates an inverted index, mapping each prototype vector to its corresponding questions and chunks.

3. Retrieval and Answer Generation Architecture

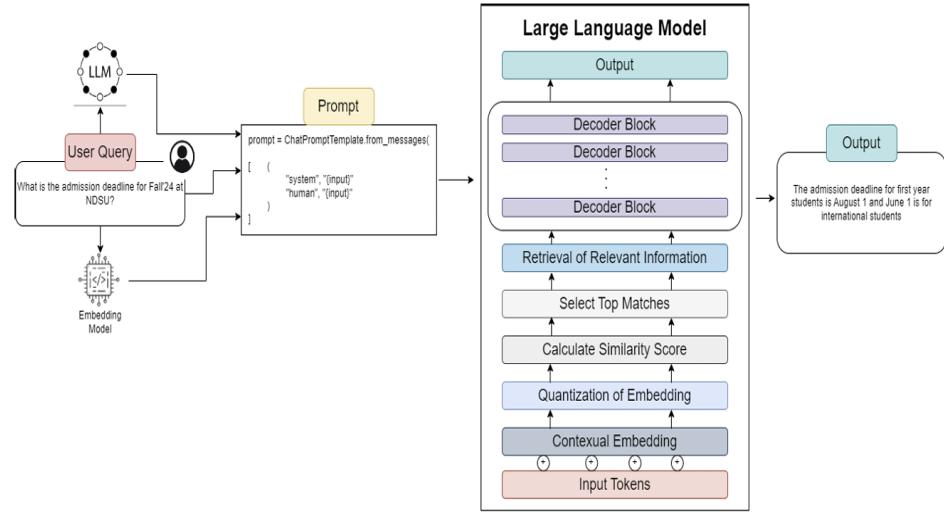


Fig 3.1.3: Retrieval and Answer Generation Architecture

➤ **Query Processing:**

- The user submits a question.
- It is embedded and quantized using the same vectorization method as the stored data.

➤ **Question Matching & Retrieval:**

- The system finds the top k matching questions in the inverted index.
- Corresponding document chunks are retrieved.

➤ **Answer Generation with Llama3-8B-instruct::**

- Retrieved chunks are fed into Meta-LLMA3-8B-instruct.
- The model generates a contextually accurate response using a custom prompt.
- The user receives a well-structured, contextually relevant answer

3.2 Datasets Used

The study uses a custom dataset extracted from North Dakota State University's (NDSU) career-advising and catalog resources. The dataset preparation involved web scraping, chunking, question generation, and manual verification to ensure high accuracy and relevance.

3.2.1 NDSU Career Advising Website

The NDSU Career Advising Website (career-advising.ndsu.edu) is an official resource provided by North Dakota State University (NDSU) to assist students in the following aspects:

- Career Planning Resources
 - Provides guidance on choosing a major, exploring career paths, and planning for graduate school.
 - Offers self-assessment tools to help students identify their skills and interests.
- ❖ Job Search and Internship Support
 - Lists job postings, internship opportunities, and career fairs for students and graduates.
 - Provides resume and cover letter templates with best practices for job applications.
- ❖ Interview Preparation and Skill Development
 - Offers mock interview sessions and tips on negotiating salaries.
 - Hosts workshops and webinars on professional skills, networking, and LinkedIn optimization.
- ❖ Employer Connections and Networking
 - Facilitates campus recruitment events where employers connect with students
 - Offers one-on-one career advising appointments to help students with job market readiness.
- ❖ Graduate Outcomes and Alumni Network
 - Provides data on graduate employment rates and career trajectories.
 - Features success stories from NDSU alumni to inspire current students.

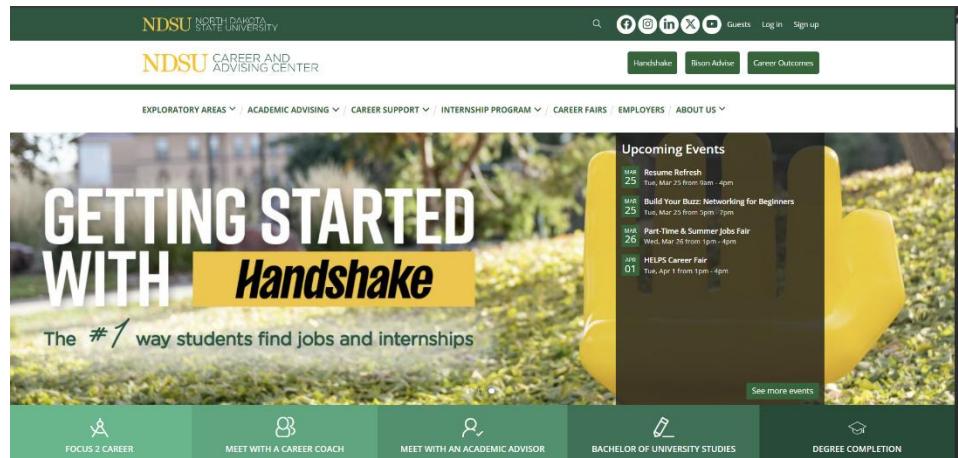


Fig 3.2.1 : NDSU Career Advising Website

3.2.2 NDSU Catalog Website

The NDSU Catalog Website (catalog.ndsu.edu) is the official academic catalog of North Dakota State University (NDSU). It provides detailed and structured information about the university's degree programs, course offerings, academic policies, and graduation requirements. The catalog is regularly updated to reflect changes in curricula, institutional policies, and course availability. It serves as a primary reference for students, faculty, and academic advisors.



Fig 3.2.2 : NDSU Catalog Website

Dataset plays a crucial role in academic advising, course planning, and AI-driven educational support systems, ensuring that students and faculty can access accurate and relevant information quickly.

4. Proposed Model

The QuIM-RAG (Question-to-Question Inverted Matching for Retrieval-Augmented Generation) model is designed to enhance retrieval precision and answer accuracy by transforming document chunks into retrievable question-answer pairs. The model incorporates inverted question matching (IQM), embedding quantization, and an optimized retrieval mechanism to improve traditional retrieval-augmented generation (RAG) systems.

4.1 QA Problem for a Limited Corpus

In the QuIM-RAG framework, the question-answering (QA) task is designed to extract the most relevant answer (A) from a finite set of documents (C) in response to a natural language query (Q). The corpus consists of a structured collection of documents $\{D_1, D_2, \dots, D_n\}$, where each document D_i is composed of smaller chunks $(S_{i1}, S_{i2}, \dots, S_{im})$ representing sentences or paragraph segments.

The goal of the system is to retrieve and generate a text snippet response (A) that directly addresses the question Q while ensuring that the answer is derived entirely from the corpus. Since A is a phrase generated by combining relevant pieces of information, the system must determine:

$$A = \arg \max_{A' \in C} \text{Relevance}(A', Q)$$

Where $\text{Relevance}(A', Q)$ is a scoring function that measures how well a potential answer A' satisfies the query Q using the context from the corpus C .

4.2 Inverted Question Matching To Find Relevant Chunks For Answer

Relevant chunks from multiple documents in the corpus can be combined to create a comprehensive informational context for answering the user's question. To efficiently retrieve these chunks, QuIM-RAG employs an inverted index scheme, where document chunks are stored in embedding space and matched based on semantic similarity. This approach ensures that the most contextually relevant chunks are

retrieved and combined, improving the accuracy and completeness of the generated answer.

1. Corpus and Document Structure

The corpus (C) is a collection of documents $\{D_1, D_2, \dots, D_n\}$, where each document D_i is divided into smaller chunks or sentences $\{S_1, S_2, \dots, S_m\}$. These chunks serve as fundamental units of information, ensuring that retrieval is performed at a granular level rather than on entire documents. By structuring the corpus in this way, QuIM-RAG enables the generation of semantically meaningful questions for each chunk, allowing for precise query-document matching and improved retrieval accuracy.

For example let the Chunks be :

ID	Extracted Text from Documents
S ₁	"Students can find internship opportunities through CareerLink or external job portals. Applications must be submitted before the deadline, and some internships require departmental approval."
S ₂	"Applicants must prepare a resume, cover letter, and sometimes a recommendation letter. Some companies may also require a transcript or writing sample."
S ₃	"Internship applications are submitted via CareerLink. Students must verify eligibility based on their major before submission. Departmental approval is sometimes required."
S ₄	"The Career Center offers resume reviews, mock interviews, and one-on-one coaching to help students improve their applications."

Table 4.2.1: Chunks

2. Question Generation from Chunks

For each chunk (S_j) within a document (D_i), an instruction-following large language model (LLM) is used to generate a set of questions $\{q_{ij1}, q_{ij2}, \dots, q_{ijk}\}$. These generated questions encapsulate the key information and concepts present in the chunk, effectively transforming raw text into structured queries for improved retrieval. This process ensures that each chunk is linked to

multiple semantically relevant questions, allowing the system to retrieve the most contextually appropriate information when answering user queries. Intuitively, these generated questions function as "frequently asked questions" (FAQs) for each chunk, making document retrieval more precise and effective.

A pre-trained instruction-following LLM (GPT-3.5-turbo-instruct) is used to generate questions from each chunk.

The model is prompted with an **instruction template** like:

Prompt for Dataset	Your task is to generate a diverse set of questions that comprehensively cover all key information and concepts within the chunk. Each question should be unique, contextually relevant , and designed to encapsulate different aspects of the chunk to ensure full coverage without redundancy. Ensure the questions are direct, clear, and closely aligned with the content of the chunk . They should accurately reflect the main themes and details to enhance the retrieval performance in the RAG system.
---------------------------	---

Fig 4.2.2: Prompt for Dataset

The LLM processes the chunk and outputs multiple questions that encapsulate the key information within the text.

Generated Questions for each Chunk:

For Chunk S₁ (Internship Process & Approval):

"Where can NDSU students find internship opportunities?"

"What are the steps to apply for an internship at NDSU?"

"Are internships at NDSU approved by departments?"

"What is the deadline for internship applications?"

For Chunk S₂ (Required Documents for Application)

"What documents are required for an internship application?"

"Do I need a cover letter for an internship?"

"Is a recommendation letter mandatory for internship applications?"

"Do some companies require transcripts for internship applications?"

For Chunk S₃ (Submission Process & Eligibility)

"Where should internship applications be submitted?"

"How do students verify their eligibility for an internship?"

"Is departmental approval required for internship applications?"

"What happens if a student is not eligible for an internship?"

For Chunk S₄ (Career Center Assistance)

"How can the Career Center help with internship applications?"

"Does the Career Center offer resume reviews?"

"Can students get mock interview practice at the Career Center?"

"Does the Career Center provide one-on-one coaching for job applications?"

3. Embedding of Generated Questions

Once the questions $\{q_{ij1}, q_{ij2}, \dots, q_{jk}\}$ are generated for each document chunk, they are converted into embedding vectors to enable efficient semantic retrieval. Each question q_{ij1} is transformed into a high-dimensional embedding vector $v_{ij1} \in R_d$ using a pre-trained encoding model such as **BAAI/bge-large-en-v1.5**. These embeddings capture the semantic meaning of the questions, ensuring that query-question matching is based on meaning rather than exact word matching. By storing these embeddings in a vector database, the system can efficiently compare and retrieve the most relevant questions when a user submits a query.

These embeddings allow **semantic similarity search**, making retrieval more **accurate** than traditional keyword matching.

Generated Questions & Their Embeddings:

Generated Question	Embedding Vector (Example Values)
"Where can NDSU students find internship opportunities?"	(0.12, -0.45, 0.78, ..., 0.98)
"What are the steps to apply for an internship at NDSU?"	(0.34, -0.67, 0.45, ..., 0.91)

Table 4.2.3: Generated Embeddings

4. Quantization of Embeddings

Since embedding vectors exist in a high-dimensional space, performing direct pairwise comparisons between all stored question embeddings and incoming queries would be computationally expensive. To reduce this complexity, QuIM-RAG applies embedding quantization.

- Each **embedding vector** v_{ijl} is quantized to the **nearest prototype** (p_l) from a fixed set of prototypes $\{p_1, p_2, \dots, p_k\}$.
- The quantization process **minimizes the cosine similarity distance** between the embedding and the prototype:

$$p_l = \arg \min_p \text{CosineSimilarity}(v_{ijl}, p)$$

- Instead of storing and searching across the entire **high-dimensional vector space**, the system **groups embeddings into quantized clusters**.
- During retrieval, the user's **query embedding is also quantized** and matched to the **nearest prototype**, allowing for **fast and efficient lookup** of relevant questions and document chunks.

Prototype Vector (Quantized Embedding)	Mapped Question Embeddings	Linked Chunks
P1 (0.34, -0.67, 0.45, ...)	"What are the steps to apply for an internship at NDSU?" "How do I submit an internship application?"	S1, S3
P2 (0.55, -0.23, 0.68, ...)	"What documents are required for an internship application?" "Is a resume necessary for an internship?" "Do internships require cover letters?"	S2, S3

Table 4.2.4: Quantized Embeddings

5. Construction of the Inverted Index

To facilitate efficient retrieval, QuIM-RAG constructs an inverted index (I) that maps each prototype (p_l) to a set of quantized embedding vectors (v_{ijl}) and their corresponding text chunks (S_j). This indexing structure allows the system to quickly retrieve relevant text chunks based on the quantized embeddings. The index is constructed as follows:

$$I(p_l) = \{(v_{ijl}, S_j) \text{ for all } v_{ijl} \text{ quantized to } p_l\}$$

Instead of performing an **exhaustive search across all embeddings**, the inverted index enables **fast and scalable retrieval** by grouping similar embeddings under shared prototypes.

6. Query Processing and Matching

When a **user submits a query (Q)**, it undergoes a **multi-step retrieval process** to find the most relevant document chunk:

1. Query Encoding:

- The query **Q** is converted into an **embedding vector (V)** using the same pre-trained encoding model used for the generated questions.

2. Query Quantization:

- The query embedding is **mapped to the nearest prototype (pl)** in the embedding space using **cosine similarity minimization**:

$$p_l = \arg \min_p \text{CosineSimilarity}(V, p)$$

➤ User Query:

- "How do I apply for an internship and what documents are needed?"

➤ Query Matching Process

- Convert user query into an embedding vector using BAAI/bge-large-en-v1.5.
- Query Embedding Vector: (0.30, -0.50, 0.75, ..., 0.93)
- Find the nearest prototype vector in the inverted index using cosine similarity.

Top Matches (Cosine Similarity Scores):

User Query Embedding	Matched Prototype Vector	Similarity Score	Retrieved Chunks
(0.30, -0.50, 0.75, ...)	P2	0.98	S1, S3
(0.30, -0.50, 0.75, ...)	P3	0.92	S2, S3

Table 4.2.5: Similarity Score

Since these prototype vectors map to multiple questions and chunks, the system retrieves:

- S1 (General steps for applying)
- S2 (Required documents: resume, cover letter, transcripts, recommendations, etc.)
- S3 (Submission process via CareerLink & approval requirements)

7. Retrieval of Relevant Text Chunks

Once the retrieved embedding vectors (v_{ij1}) are identified through the inverted index, the system decodes the corresponding original questions (q_{ij1}) and maps them to their associated text chunks (S_j). These retrieved chunks represent the most contextually relevant sections of the document corpus that can effectively answer the user's query (Q).

By retrieving only the most semantically aligned chunks, QuIM-RAG ensures that the generated response is factually grounded and contextually accurate, improving the quality of answers in RAG-based systems.

The Retrieved Chunks are: S1, S2, S3

Algorithm 1 Inverted Index in Embedding Space With Question Generation and Matching

```
1: Input: Corpus  $C = \{D_1, D_2, \dots, D_n\}$ , Prototypes  $\{p_1, p_2, \dots, p_k\}$ 
2: Output: Relevant text chunks for user query  $Q$ 
3: function GenerateQuestions( $S_i$ )
4:     Use the Instruction-following LLM to generate questions  $\{q_{i1}, q_{i2}, \dots, q_{ik}\}$  from chunk  $S_i$ 
5:     return Questions  $\{q_{i1}, q_{i2}, \dots, q_{ik}\}$ 
6: end function
7: function EmbedQuestions( $\{q_{i1}, \dots, q_{ik}\}$ )
8:     for each question  $q_{ij}$  do
9:         Compute embedding  $v_{ij} \leftarrow \text{Encode}(q_{ij})$ 
10:        Append  $v_{ij}$  to the list of embeddings
11:    end for
12:    return Embeddings  $\{v_{i1}, v_{i2}, \dots, v_{ik}\}$ 
13: end function
14: function QuantizeEmbedding( $v_{ij}, \{p_1, \dots, p_k\}$ )
15:     return Prototype  $p_l \leftarrow \arg \min_p \text{CosineSimilarity}(v_{ij}, p)$ 
16: end function
17: function BuildInvertedIndex( $C, \{p_1, \dots, p_k\}$ )
18:     Initialize an empty inverted index  $I$ 
19:     for each document  $D_i$  in  $C$  do
20:         for each chunk  $S_j$  in  $D_i$  do
21:             Questions  $\leftarrow$  GenerateQuestions( $S_j$ )
22:             Embeddings  $\leftarrow$  EmbedQuestions(Questions)
23:             for each embedding  $v_{ij}$  in Embeddings do
24:                 Prototype  $p_l \leftarrow \text{QuantizeEmbedding}(v_{ij}, \{p_1, \dots, p_k\})$ 
25:                 Update  $I(p_l) \leftarrow I(p_l) \cup \{(v_{ij}, S_j)\}$ 
26:             end for
27:         end for
28:     end for
29:     return  $I$ 
30: end function
31: function MatchQuery( $Q, \{p_1, \dots, p_k\}, I$ )
32:     Compute query embedding  $V \leftarrow \text{Encode}(Q)$ 
33:     Find the nearest prototype  $p_l \leftarrow \arg \min_p \text{CosineSimilarity}(V, p)$ 
34:     Retrieve relevant embeddings  $E \leftarrow I(p_l)$ 
35:     for each  $(v_{ij}, S_j)$  in  $E$  do
36:         Decode  $v_{ij}$  to original question  $q_{ij}$ 
37:         Append chunk  $S_j$  to the list of relevant texts
38:     end for
39:     return Relevant text chunks
40: end function
```

Fig 4.2.6: Model Algorithm

4.3 Answer Generation for User Query

Once the most relevant text chunks (S_j) are retrieved, they are passed to the LLaMA3-8B-Instruct model to generate a coherent and factually grounded response. Unlike traditional RAG models that rely **on** pre-trained knowledge, QuIM-RAG strictly ensures that the LLM only uses retrieved chunks as context, effectively minimizing hallucinations and improving response accuracy.

The final response is generated using **LlaMA3-8B-Instruct**, an instruct-tuned model from Hugging Face, chosen for its advanced conversational capabilities and extended context length of 8,000 tokens. This enables the model to process longer input passages, ensuring better context retention and accurate response synthesis. The retrieval system first extracts the most relevant information, which is then fed into LLaMA3 using a custom prompting strategy. This prompting method ensures that the response remains factually accurate and contextually aligned with the retrieved content, maintaining consistency, correctness, and readability in the final answer.

Prompt for RAG	You are a dedicated assistant tasked with responding to inquiries specifically related to North Dakota State University (NDSU). Only give precise answers to the queries, nothing else. Include all relevant information from the provided context. Add the source url in your response. Upon receiving a question, you will be given both the query and the relevant context . If the provided context does not align with the query or if the information required to answer is beyond the scope of your knowledge base, you should respond with, "I cannot find an answer to this question based on the available information." If the query is unrelated to NDSU , clearly state, " This question is not related to NDSU ".
-----------------------	--

Fig 4.3.1: Prompt for RAG

Final LLM input:

User Query: “How do I apply for an internship and what documents are needed?”

Relevant Information:

1. Steps for applying: Students can find internship opportunities through CareerLink or external job portals. Applications must be submitted before the deadline.
2. Required documents: A resume, cover letter, and sometimes a recommendation letter or transcript may be needed.
3. Submission process: Applications are submitted via CareerLink, and some require departmental approval.

Generate a concise and accurate answer based on the above information.

Final Response Generated by Meta-LlaMA3-8B-instruct:

To apply for an internship at NDSU, students should explore opportunities on CareerLink or external job portals. They must prepare a resume, cover letter, and sometimes additional documents like a transcript or recommendation letter. Applications should be submitted via CareerLink, and students should verify their eligibility before applying. Departmental approval may be required for certain internships.

5. Discussions on the Proposed Output

5.1 Experiments and Results

5.1.1 Data Collection and Preprocessing

To build an effective retrieval-augmented generation (RAG) system, a high-quality dataset was curated by scraping, preprocessing, and structuring text from authoritative sources. The dataset preparation involved web scraping, cleaning, chunking, and question-answer pair generation to improve retrieval accuracy and response quality.

Data Sources

The dataset was constructed using publicly available data from two primary sources:

- **NDSU Catalog Dataset** – Contained information on academic programs, course descriptions, enrollment policies, and university regulations.
- **NDSU Career Advising Dataset** – Provided data on career planning, job search, and interview preparation.

Preprocessing and Cleaning

Since raw scraped data contains irrelevant elements, the following steps were applied to clean and structure the corpus:

- **Removal of Irrelevant Content:** Eliminated headers, footers, ads, navigation menus, and duplicate text.
- **Filtering of Low-Quality Pages:** Removed pages with <250 characters or 404 errors.
- **Text Normalization:** Lowercased text, removed special characters and redundant whitespace.

Chunking Strategy

To ensure effective retrieval, the dataset was divided into manageable chunks:

- **Chunk Size:** 1,000 tokens (with a 200-token overlap to retain context).
- **Total Number of Chunks Created:** 296 (NDSU Catalog), 435 (NDSU Career Advising)

TABLE 1. Summary of the NDSU catalog and career advising datasets.

Metric	NDSU Catalog Dataset	NDSU Career Advising
No. of Links Scraped	288	399
No. of Questions	9,027	13,582
No. of chunks	296	435

Table 5.1.1: Summary of DataSet

Question-Answer Pair Generation

To enhance retrieval effectiveness, each chunk was converted into multiple question-answer pairs using GPT-3.5-turbo-instruct.

- **Total Number of Questions Generated:** 9,027 (NDSU Catalog), 13,582 (NDSU Career Advising)
- **Average Questions per Chunk:** ≈ 30.5 (NDSU Catalog), ≈ 31.2 (NDSU Career Advising)
- **Question Filtering:** Removed redundant and vague questions to retain high-quality, contextually relevant Q&A pairs.

Total Data Pipeline Overview:

Step	Process	Tools Used
1. Data Collection	Scraping university websites	BeautifulSoup, Scrapy
2. Preprocessing	Cleaning, text normalization	NLTK, spaCy
3. Chunking	Fixed-length & semantic segmentation	NLTK, Sentence Transformers

4.Embedding Generation	Convert text into vector representations	BAAI/bge-large-en-v1.5
5. Indexing	Store embeddings in an inverted index	FAISS, ChromaDB
6. Retrieval	Query matching using cosine similarity	FAISS, ChromaDB
7.Answer Generation	LLM generates response	Meta-LLaMA3-8B-instruct

Table 5.1.1 : Pipeline Overview

5.1.2 Evaluation Metrics Used

To evaluate the performance of our QuIM-RAG model, we employ two evaluation frameworks: BERTScore and RAGAS. These frameworks provide a detailed quantitative analysis focused on semantic accuracy, retrieval precision, and response relevance.

Traditional metrics such as BLEU, METEOR, and ROUGE are not well-suited for evaluating retrieval-augmented generation (RAG) systems, as they primarily assess machine translation and text summarization. Instead, BERTScore and RAGAS are used because they are tailored to question-answering models that rely on retrieval mechanisms.

A. BERTScore

BERTScore measures the semantic quality of generated text by evaluating the overlap between model-generated responses and reference texts. Unlike traditional n-gram-based methods, BERTScore leverages contextual embeddings from transformer models (e.g., BERT, RoBERTa) to capture deeper semantic relationships.

Computation of BERTScore

- Convert both reference and generated responses into contextual embeddings:

$$\text{BERT}((x_1, x_2, \dots, x_n)) = (X_1, X_2, \dots, X_n)$$

$$\text{BERT}((\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)) = (\hat{X}_1, \hat{X}_2, \dots, \hat{X}_m)$$

- Compute the **cosine similarity** between token embeddings in reference and generated texts.

$$\text{similarity}(X_i, \hat{X}_j) = \frac{X_i^T \hat{X}_j}{\|X_i\| \|\hat{X}_j\|}$$

- Aggregate similarity scores into three distinct metrics:

- **BERT Precision** (P_{BERT}): Measures how accurately the generated answer matches the reference answer by computing token-level similarity.

$$P_{\text{BERT}} = \frac{1}{|\hat{\mathbf{X}}|} \sum_{\hat{x}_j \in \hat{\mathbf{X}}} \max_{x_i \in \mathbf{X}} \left\{ \text{cosine similarity}(\hat{X}_i^T, \hat{X}_j) \right\}$$

- **BERT Recall** (R_{BERT}): Evaluates how well the generated answer covers all key information present in the reference answer.

$$R_{\text{BERT}} = \frac{1}{|\mathbf{X}|} \sum_{x_i \in \mathbf{X}} \max_{\hat{x}_j \in \hat{\mathbf{X}}} \left\{ \text{cosine similarity}(X_i^T, \hat{X}_j) \right\}$$

- **BERT F1 Score** (F_{BERT}): Combines precision and recall to give an overall measure of answer quality and completeness.

$$F_{\text{BERT}} = 2 \times \frac{P_{\text{BERT}} \times R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}$$

B. Retrieval-Augmented Generation Assessment (RAGAS)

RAGAS is an evaluation framework specifically designed for RAG-based systems, which combine retrieval mechanisms with large language models (LLMs) to generate responses. Unlike traditional evaluation techniques, RAGAS does not require human annotations, making it more scalable and adaptable.

RAGAS evaluates three core **performance aspects** of a RAG system:

- **Faithfulness** : Measures whether the generated response is factually supported by the retrieved document chunks, avoiding hallucinations.

$$\text{Faithfulness} = \frac{\text{Number of verified claims}}{\text{Total claims made}}$$

- **Answer Relevance** : Assesses the semantic alignment between the generated answer and the user's original question using cosine similarity.

$$AR = \frac{1}{n} \sum_{i=1}^n \text{sim}(q, q_i)$$

- **Context Relevance** : Evaluates whether the retrieved document chunks contain information that is directly useful for answering the query.

$$CR = \frac{\text{Number of relevant sentences extracted}}{\text{Total sentences in the context}}$$

5.2 Performance Analysis of Results

5.2.1 Work Flow Analysis (QuIM-RAG vs Traditional RAG)

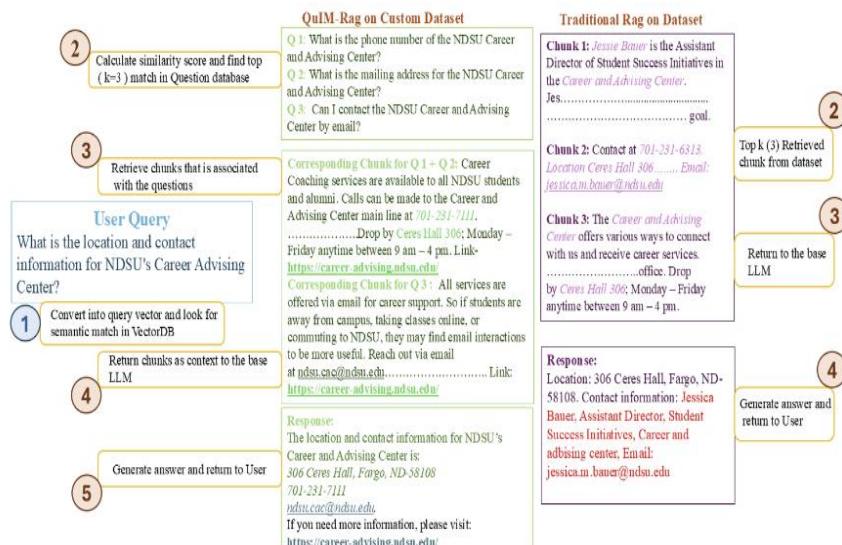


Fig 5.2.1 : WorkFlow of QuIM-RAG and Traditional RAG

1. User Query Processing:

A user asks:

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

2. Query Transformation & Retrieval:

- **QuIM-RAG:** Converts the query into a vector representation and looks for a semantic match in a VectorDB.
- **Traditional RAG:** Retrieves the top-k matching chunks from the dataset using similarity search.

3. Context Chunk Selection

- **QuIM-RAG:** Retrieves specific semantically relevant chunks related to the user query.
- **Traditional RAG:** Selects chunks based on keyword matches rather than semantic understanding.

4. Answer Generation

- **QuIM-RAG:** Returns only the most relevant chunks to the base LLM, ensuring more accurate and contextual responses.
- **Traditional RAG:** Passes retrieved text chunks directly to the LLM, often leading to less coherent or incomplete responses.

5. Final Answer Output

- **QuIM-RAG:** Generates a well-structured answer, providing a concise and direct response with verified contact details, links, and references.
- **Traditional RAG:** May return less precise or redundant information, sometimes including unnecessary details (e.g., personal details of a staff member).

- QuIM-RAG provides a more precise and relevant answer by leveraging semantic search and chunk retrieval.
- Traditional RAG retrieves information based on textual similarity, sometimes leading to irrelevant or incomplete results.

5.2.2 Comparison of performance metrics

Evaluation Matrix	Traditional RAG		QuIM-RAG	
	Traditional	Custom	Traditional	Custom
Faithfulness	0.69	0.72	0.91	1.00
Answer Relevancy	0.79	0.82	0.93	0.99
Context Precision	0.45	0.69	0.82	0.92
Context Recall	0.39	0.45	0.60	0.74
Harmfulness	0	0	0	0
BERTScore P.	0.32	0.37	0.55	0.63
BERTScore R.	0.29	0.35	0.63	0.71
F1 Score	0.31	0.36	0.59	0.67

Table 5.2.2: Comparison of Performance metrics

Key Observations:

❖ Faithfulness

- Faithfulness measures how factually correct the generated responses are.
- QuIM-RAG (0.91 Traditional / 1.00 Custom) significantly outperforms Traditional RAG (0.69 Traditional / 0.72 Custom).
- This indicates that QuIM-RAG maintains high factual accuracy, particularly when using a custom dataset.

❖ Answer Relevancy

- Measures how well the generated answer aligns with the user's query.
- Traditional RAG shows moderate performance (0.79 Traditional / 0.82 Custom).
- QuIM-RAG achieves near-perfect relevancy (0.93 Traditional / 0.99 Custom), meaning it provides highly useful responses.

❖ Context Precision & Recall

- Precision evaluates whether retrieved chunks are relevant, while recall assesses whether all necessary information is included.
- Traditional RAG has low precision (0.45) and recall (0.39), leading to incomplete or imprecise answers.
- QuIM-RAG dramatically improves these values (Precision: 0.82 Traditional / 0.92 Custom, Recall: 0.60 Traditional / 0.74 Custom), retrieving more useful and complete information.

❖ **Harmfulness**

- All methods maintain a score of 0, ensuring no harmful or biased content is generated.

❖ **BERTScore Precision & Recall**

- These metrics assess how semantically close the generated answers are to a reference ground truth.
- QuIM-RAG achieves higher precision (0.55 Traditional / 0.63 Custom) and recall (0.63 Traditional / 0.71 Custom), reinforcing its stronger contextual understanding.

❖ **F1 Score**

- Combines precision and recall into a single performance measure.
- Traditional RAG has lower F1 scores (0.31 Traditional / 0.36 Custom).
- QuIM-RAG shows almost double the effectiveness (0.59 Traditional / 0.67 Custom).

Overall Performance

- QuIM-RAG outperforms Traditional RAG across all metrics, especially when using a Custom dataset.
- The improvements in faithfulness, precision, and recall ensure higher-quality and contextually relevant responses.
- These results demonstrate that QuIM-RAG is a superior approach for retrieval-augmented generation (RAG) systems, making it more effective for real-world NLP applications.

6. Conclusion and Future Work

6.1. Conclusion

In this study, we proposed QuIM-RAG (Question-to-Question Inverted Matching for Retrieval-Augmented Generation), a novel framework designed to enhance retrieval accuracy and response generation in RAG-based systems. Unlike traditional RAG approaches that rely on direct document chunk retrieval, our method pre-generates and indexes semantically meaningful questions, improving the alignment between user queries and retrieved content.

One of the key innovations of QuIM-RAG is its custom dataset construction, specifically designed to optimize retrieval efficiency and question-answer alignment. Traditional RAG models, when applied to large, unstructured datasets, often suffer from retrieval noise, pulling in irrelevant or loosely related content. By integrating a structured dataset and LLaMA-3-8B-Instruct, our approach significantly improves faithfulness, answer relevancy, and overall response quality compared to baseline models using raw web data. The model’s ability to provide source citations further enhances user trust, allowing individuals to verify the provided information and seek additional details if needed.

Looking forward, our next steps involve conducting a comprehensive user study to evaluate user satisfaction, response relevance, and system usability. The insights gained from this study will help refine QuIM-RAG’s retrieval pipeline, response structuring, and user interaction capabilities. By incorporating real-world feedback, we aim to create a more adaptable, responsive, and efficient QA system that continues to meet user needs effectively while remaining scalable and reliable in dynamically evolving knowledge environments.

The results of this research demonstrate that QuIM-RAG represents a major step forward in the evolution of retrieval-augmented generation systems, providing a high-precision, context-aware, and reliable framework for domain-specific question-answering. As AI-driven knowledge systems continue to expand, our approach provides a scalable blueprint for developing intelligent, fact-grounded, and user-centric retrieval-based applications.

6.2. Future Work

While QuIM-RAG has demonstrated significant improvements in retrieval precision and response generation, there are several areas for further research and enhancement.

1. Continuous Corpus Updates for Dynamic Content

- Since university websites and knowledge repositories frequently update their content, ensuring that retrieved information remains current is a critical challenge.
- We plan to implement an automated corpus update mechanism that refreshes the dataset every four months, ensuring new academic policies, course structures, and career resources are reflected in retrieval outputs.

2. Expanding to Multi-Domain Question Answering

- Currently, QuIM-RAG is optimized for domain-specific knowledge, particularly in academic and advisory systems.
- Future iterations will expand the model's adaptability to handle multi-domain QA tasks, ensuring it generalizes well across different knowledge-intensive applications (e.g., healthcare, finance, technical documentation).

3. Enhancing Question Reformulation Strategies

- QuIM-RAG's inverted matching mechanism relies on a robust question generation process.
- Future enhancements will focus on more advanced query reformulation strategies that dynamically rewrite complex queries into simpler, semantically aligned formats for better retrieval accuracy.

4. Improving Computational Efficiency and Scalability

- The current retrieval and quantization process has significantly optimized search speed.
- However, further research into low-latency embedding search techniques (e.g., approximate nearest neighbor indexing, hybrid sparse-dense retrieval) will help make the system faster and more scalable.

5. Conducting Large-Scale User Studies

- A comprehensive user study will be conducted to assess system usability, response reliability, and overall user satisfaction.
- Insights from these studies will help fine-tune retrieval mechanisms, improve prompt optimization, and ensure better alignment with user expectations.

6. Addressing Out-of-Domain Queries

- Current RAG systems struggle with queries that extend beyond the model's knowledge base.
- We aim to refine QuIM-RAG's response strategies by integrating uncertainty-aware mechanisms that explicitly inform users when relevant information is unavailable, reducing AI hallucinations.

References

- [1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [2] Z. Wang, F. Yang, P. Zhao, L. Wang, J. Zhang, M. Garg, Q. Lin, and D. Zhang, “Empower large language model to perform better on industrial domain-specific question answering,” *arXiv preprint arXiv:2305.11541*, 2023.
- [3] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, “Large language models struggle to learn long-tail knowledge,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 696–15 707.
- [4] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, “Fine-tuning or retrieval? comparing knowledge injection in llms,” *arXiv preprint arXiv:2312.05934*, 2023.
- [5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [6] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An empirical investigation of catastrophic forgetting in gradient-based neural networks,” *arXiv preprint arXiv:1312.6211*, 2013.
- [7] S. Chen, Y. Hou, Y. Cui, W. Che, T. Liu, and X. Yu, “Recall and learn: Fine-tuning deep pretrained language models with less forgetting,” *arXiv preprint arXiv:2004.12651*, 2020.
- [8] B. Saha and U. Saha, “Enhancing international graduate student experience through aidriven support systems: A llm and rag-based approach,” in *2024 International Conference on Data Science and Its Applications (ICoDSA)*, North Dakota State University. IEEE, 2024, pp. 300–304, in press.
- [9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K'uttler, M. Lewis, W.-t. Yih, T. Rockt"aschel et al., “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

- [10] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy et al., “Text and code embeddings by contrastive pre-training,” arXiv preprint arXiv:2201.10005, 2022.
- [11] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, “Retrieval-augmented generation for large language models: A survey,” arXiv preprint arXiv:2312.10997, 2023.
- [12] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?” in Proceedings of the 2021 ACM conference on fairness, accountability, and transparency, 2021, pp. 610–623.
- [13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., “Language models are unsupervised multitask learners,” OpenAI blog, vol. 1, no. 8, p. 9, 2019.
- [14] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann et al., “Palm: Scaling language modeling with pathways,” Journal of Machine Learning Research, vol. 24, no. 240, pp. 1–113, 2023.
- [15] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., “Llama 2: Open foundation and fine-tuned chat models,” arXiv preprint arXiv:2307.09288, 2023.
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in neural information processing systems, vol. 30, 2017.
- [17] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili et al., “A survey on large language models: Applications, challenges, limitations, and practical usage,” Authorea Preprints, 2023.
- [18] S. Huo, N. Arabzadeh, and C. L. Clarke, “Retrieving supporting evidence for llms generated answers,” arXiv preprint arXiv:2306.13781, 2023.
- [19] L. Fan, L. Li, Z. Ma, S. Lee, H. Yu, and L. Hemphill, “A bibliometric review of large language models research from 2017 to 2023,” ArXiv, vol. abs/2304.02020, 2023.
[Online]. Available: <https://api.semanticscholar.org/CorpusID:257952516>
- [20] V. Bolotova, V. Blinov, F. Scholer, W. B. Croft, and M. Sanderson, “A non-factoid questionanswering taxonomy,” in Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1196–1207.

- [21] A. Rogers, M. Gardner, and I. Augenstein, “Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension,” ACM Computing Surveys, vol. 55, no. 10, pp. 1–45, 2023.
- [22] S. Ruder and A. Sil, “Multi-domain multilingual question answering,” Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts, 2021. Available: <https://api.semanticsscholar.org/CorpusID: 245289877>
- [23] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, “Survey of hallucination in natural language generation,” ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023.
- [24] G. Mialon, R. Dess`1, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozi`ere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz et al., “Augmented language models: a survey,” arXiv preprint arXiv:2302.07842, 2023.
- [25] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, “Internet-augmented language models through few-shot prompting for open-domain question answering,” arXiv preprint arXiv:2203.05115, 2022.
- [26] P. Lewis, Y. Wu, L. Liu, P. Minervini, H. Kuttler, A. Piktus, P. Stenetorp, and S. Riedel, “Paq: 65 million probably-asked questions and what you can do with them,” Transactions of the Association for Computational Linguistics, vol. 9, pp. 1098–1115, 2021.
- [27] V. Sharma and V. Raman, “A reliable knowledge processing framework for combustion science using foundation models,” arXiv preprint arXiv:2401.00544, 2023.
- [28] I. Siragusa and R. Pirrone, “Conditioning chat-gpt for information retrieval: The unipagpt case study,” in Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023), 2023. [29] M. Kulkarni, P. Tangarajan, K. Kim, and A. Trivedi, “Reinforcement learning for optimizing rag for domain chatbots,” arXiv preprint arXiv:2401.06800, 2024.
- [30] S. Campese, I. Lauriola, and A. Moschitti, “Quadro: Dataset and models for questionanswer database retrieval,” arXiv preprint arXiv:2304.01003, 2023.
- [31] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, “Mteb: Massive text embedding benchmark,” arXiv preprint arXiv:2210.07316, 2022.
- [32] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” arXiv preprint arXiv:1904.09675, 2019.

- [33] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “Ragas: Automated evaluation of retrieval augmented generation,” arXiv preprint arXiv:2309.15217, 2023.
- [34] S. Kumar, A. Solanki, and N. Z. Jhanjhi, “Rouge-ss: A new rouge variant for the evaluation of text summarization,” 2024.
- [35] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65– 72.
- [36] M. Barbella and G. Tortora, “Rouge metric evaluation for text summarization techniques,” Available at SSRN 4120317, 2022.