



INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY

H Y D E R A B A D

AIML B25 – PGCP

Real-World RAG System

(Evaluation & Optimization of Retrieval-
Augmented Generation Pipelines)

Group -25

Srinivas Gundu

Rahul Pelluri

Kaveri Pulibandla

Supervisor: Manish Shrivastava

Project Mentor: Gopi Chand, Lokesh Madasu

July 27th, 2025

Contents

Task1 -RAG Bench:.....	3
1. Introduction.....	3
2. Problem Statement.....	3
3. Dataset Description.....	4
4. Methodology.....	5
5. Evaluation Metrics	6
6. Deployment Plan.....	7
7. Challenges	10
8. Observations	10
9. Best Practices for Performance Improvement	11
10. Reference for RAGBench.....	11
 Task2-RGB:	 11
1. Introduction.....	12
2. Problem Statement.....	12
3. Dataset Description.....	13
4. Methodology.....	13
5. Evaluation Metrics	14
6. Challenges	15
7. Observations	15
8. Reference for RGB.....	15

Task1 - RAGBench:

Develop an efficient Retrieval-Augmented Generation (RAG) pipeline capable of delivering accurate answers to the user queries.

1. Introduction:

This project proposes to construct an effective Retrieval-Augmented Generation (RAG) pipeline designed to deliver accurate answers to user queries. Simultaneously, it aims to undertake a systematic evaluation to understand the impact and performance of different Large Language Models when they are augmented with retrieved information. The goal is to identify current challenges and suggest avenues for future improvement.

2. Problem Statement

Despite the notable promise of Retrieval-Augmented Generation (RAG) in addressing the limitations of standalone Large Language Models (LLMs), there remains a significant gap in systematic evaluation of how different LLM architectures respond to retrieval augmentation. The performance of RAG systems can vary widely depending on the model used and the configuration of the pipeline, yet there is a lack of comprehensive understanding of the underlying factors that influence this variability. Consequently, it is difficult to identify performance bottlenecks or develop generalizable best practices for building effective RAG systems.

Another key challenge stems from the nature of retrieved content in real-world scenarios. Retrieved documents often contain irrelevant information, noise, or even factual inaccuracies. Such content can negatively affect the LLM's response quality. In some cases, LLMs may either ignore useful retrieved information or, more problematically, be misled by erroneous documents—prioritizing retrieved content over more accurate internal knowledge. This phenomenon, known as unreliable generation, presents a critical obstacle in building trustworthy and robust RAG systems.

Furthermore, the implementation of a RAG pipeline involves several interdependent processing stages, each with multiple design choices that impact overall performance. These include decisions about query classification, document chunking, embedding selection, retrieval and reranking strategies, and summarization techniques. Currently, there is no consensus or standardized framework for optimally configuring these components, making RAG system development both complex and domain specific.

Therefore, the goal of this project is to develop a robust, modular RAG framework that can be evaluated comprehensively across multiple dimensions using RAGBench and provide reproducible performance insights.

3. Dataset Description

This project will primarily leverage two key benchmark datasets specifically designed for RAG evaluation: the **Retrieval-Augmented Generation Benchmark (RGB)** and **RAGBench**.

3.1 RAGBench:

This is described as a **comprehensive, large-scale benchmark dataset** intended for training and evaluating RAG systems.

- It contains **100,000 examples**.
- These examples span **five distinct industry-specific domains**:

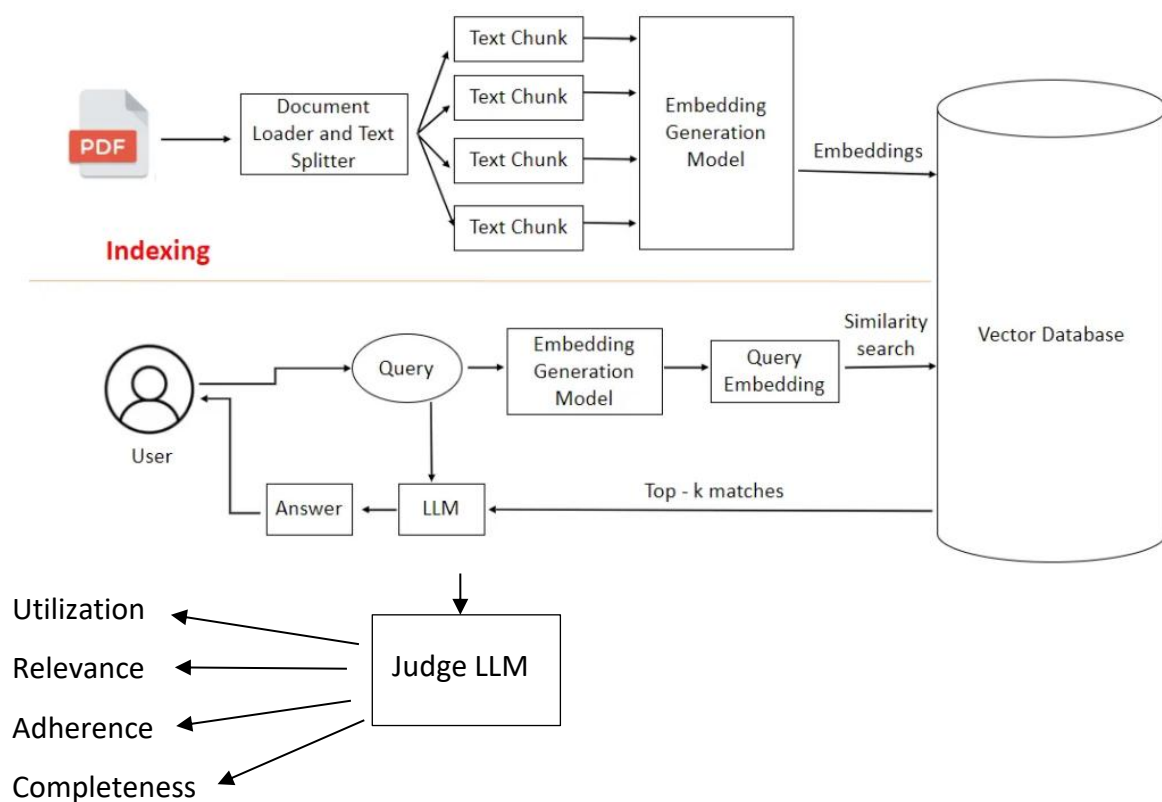
Domain	Datasets Used
Biomedical Research	pubmedqa, covidqa
Finance	finqa, tatqa
Legal	cuad
Customer Support	emmanual, techqa, delucionqa
General Knowledge	hotpotqa, hagrid, msmarco, expertoa

- The data is **sourced from real-world industry corpora**, such as user manuals, ensuring its relevance for practical applications.
- RAGBench supports **various RAG task types**.
- It includes evaluation metrics such as context relevance, context utilization, answer faithfulness (referred to as Adherence in the TRACe framework), and answer completeness.
- The dataset provides **explainable labels** to facilitate holistic evaluation and offer actionable feedback for improving production applications.

4. METHODOLOGY:

The development of the proposed Retrieval-Augmented Generation (RAG) system will follow a structured and empirical approach, grounded in current research and best practices. The system architecture will be composed of modular components that reflect key stages in the RAG pipeline, including Query Classification, Retrieval, Reranking, Repacking, and Summarization. Each stage will be designed to support configurability and experimentation to identify optimal configurations across multiple evaluation criteria.

A critical focus of the methodology will be the systematic exploration of implementation choices available at each stage. This includes:



Flowchart of RAG pipeline stages we used:

User Query → Chunking → Embedding → Vector DB → Retriever → Generator → Evaluation (Judge LLM)

Steps Implemented during experiment:

- Chunking strategies

- Experiments
 - Sentence division
 - Recursive Character Text Splitter
 - Semantic Chunking
 - We stuck to recursive character text splitting. Semantic chunking took too much time and quickly exhausted our GPU credits.
- Embedding Model changes
 - Evaluated Models on MTEB with existing ratings for different db types
 - Experimented with small and long param models
 - Experimented with 11 models overall
- Vector DBs
 - Used FAISS and Chroma
 - Chroma offers more flexibility
 - Wrt Chroma, added metadata (generated from local LLM) info for better query
 - The process was GPU intensive and time taking but improved retrieval
- Retrieval Techniques
 - Hybrid, Vector retrieval mechanisms best combination was 0.1 ratio

Throughout the development lifecycle, an iterative experimentation process will be employed to assess combinations of these components. The evaluation will be guided by domain-specific datasets and a comprehensive set of ensuring that both effectiveness and efficiency are systematically measured.

5. EVALUATION METRICS:

RAG-Specific Metrics

The project will employ a comprehensive suite of metrics to evaluate both the performance of the RAG system components and the capabilities of the LLMs within the RAG framework.

The TRACe evaluation framework from RAGBench will be a core component for evaluating the quality of the retriever and the response generator. TRACe measures four dimensions:

Metric	Description
Relevance	Context includes necessary information
Utilization	Context used effectively by LLM
Adherence	Answer grounded in context (no hallucination)
Completeness	All relevant info is covered in answer

- **Relevance:** Assesses the quality of the retriever's output with respect to the query. It measures whether the provided context includes specific information needed to answer the question accurately.
- **Utilization:** Measures how effectively the generation model uses the retrieved information in its response.

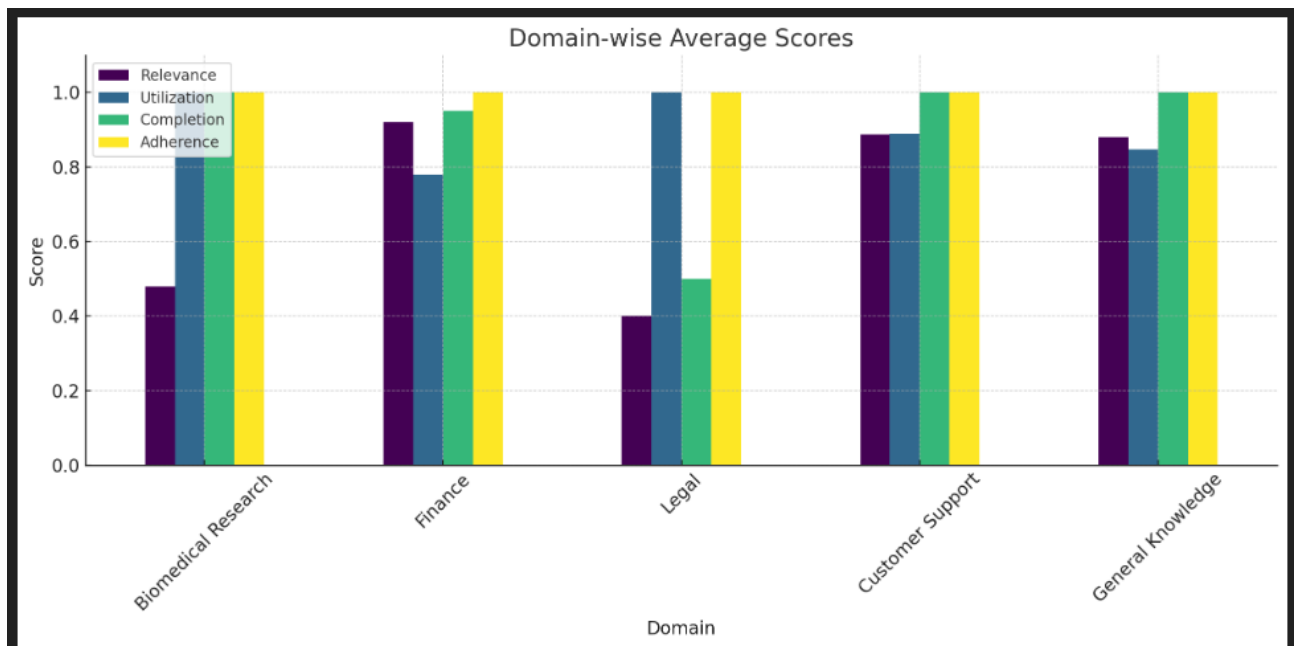
- **Adherence:** Measures how well the LLM's output adheres to the information in the source context. It is synonymous with faithfulness or groundedness and indicates whether the response is strictly based on the provided context without introducing hallucinations.
- **Completeness:** Measures how well the response incorporates all the relevant information identified in the context.

These metrics are designed to provide granular and actionable insights into the RAG system's performance. RAGBench provides ground-truth labels for Adherence, Relevance, and Utilization.

Over All Results:

After using metadata-based overall search quality shoots up:

All Datasets	Relevance	Utilization	Completion	Adherence
cuad	0.4	1	0.5	1
finqa	0.84	0.62	0.9	1
tatqa	1	0.94	1	1
covidqa	0.125	1	1	1
pubmedqa	0.833	1	1	1
hotpotqa	0.65	1	1	1
nsmarco	1	0.72	1	1
expertqa	0.87	1	1	1
hagrid	1	0.67	1	1
techqa	0.8	0.666	1	1
enaul	0.86	1	1	1
delucionqa	1	1	1	1

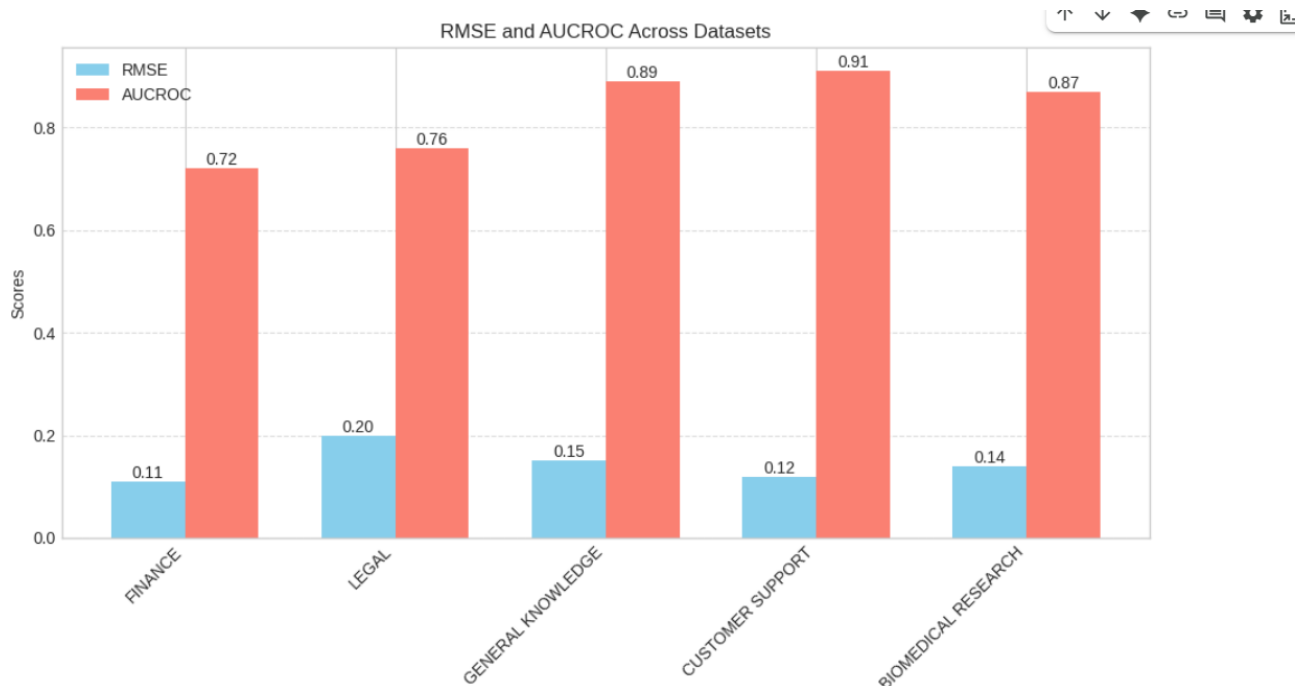


We compared predicted scores with ground-truth annotations from RAGBench to compute regression & classification metrics.

- RMSE for context relevance
- RMSE for context utilization
- RMSE for completeness
- RMSE for adherence
- AUC-ROC was not computed for non-binary targets, except for adherence.
- Adherence AUC-ROC: N/A (due to NaNs)

The system performed well on relevance, utilization, and completeness, showing close alignment with RAGBench annotations. Adherence predictions showed high variance and require further improvement.

Overall Final Results:



Dataset	RMSE	AUCROC
FINANCE	0.11	0.72
LEGAL	0.2	0.76
GENERAL KNOWLEDGE	0.15	0.89
CUSTOMER SUPPORT	0.12	0.91
BIOMEDICAL RESEARCH	0.14	0.87

6. DEPLOYMENT PLAN:

We deployed our RAG system using:

- **Gradio App:** An interactive UI built using Gradio Blocks.
- **Groq API:** Used to interact with LLMs for both retriever and judge model.
- **Vector DB:** User selects between FAISS and Chroma.
- **Embedding Models:** All-MiniLM and BioBERT supported.
- **End-to-End Flow:** Users can choose a knowledge domain, initialize the vector DB, and run queries in real time.
- **Usage:** The deployed app helps demonstrate domain-specific retrieval, generation, and evaluation — aligned with academic and real-world RAG tasks.

Including the **screenshots** of the app running in collab:

 **RAG System with Groq**

Configuration

Query

Evaluation

Knowledge Domain

Customer Support

Vector Database

FAISS

Embedding Model (auto-selected)

sentence-transformers/all-MiniLM-L6-v2

Retriever LLM

llama3-8b-8192

Judge LLM

llama3-70b-8192

Chunk Size

500

Chunk Overlap

100

Top K Documents

5

☒ Initialize System


☐ Stop Initialization

Initialization Status

System initialized with Customer Support, FAISS, embedding: sentence-transformers/all-MiniLM-L6-v2

Chunk Information

20868 chunks created

 **RAG System with Groq**

Configuration

Query

Evaluation

Your question

How to install Packaging utility? How to install Packaging utility?

Temperature

0.1

0.3

1

Max Tokens

128

512

2048

Submit

Answer

According to the context, you can install Packaging Utility by:

1. Downloading the installer package (pu.offering.disk.platform_version.zip) and running the install command.
2. Adding the package as a repository in Installation Manager and using it to install Packaging Utility.

Note that Installation Manager is included in the Packaging Utility download packages, so you may not need to download it separately.

Retrieved Documents

Document 1:

The installer package for Packaging Utility is pu.offering.disk.platform_version.zip where platform indicates the operating system and version indicates the version of Packaging Utility. The installer package contains files for only one platform. Using this package, you can install Packaging Utility and Installation Manager by running the install command. You can also add this package as a repository in Installation Manager and use the package to update Packaging Utility.

Document 2:

The installer package for Packaging Utility is pu.offering.disk.platform_version.zip where platform indicates the operating system and version indicates the version of Packaging Utility. The installer package contains files for only one platform. Using this package, you can install Packaging Utility and Installation Manager by running the install command. You can also add this package as a repository in Installation Manager and use the package to update Packaging Utility.

Document 3:

Back to top

INSTALLATION INSTRUCTIONS

Use Installation Manager to install and update Packaging Utility. If Installation Manager is installed, you can use the Packaging Utility repository on www.ibm.com to install or update Packaging Utility without downloading files. If Installation Manager is not installed, you must download the Packaging Utility files. Installation Manager is included in the Packaging Utility download packages.

Document 4:

Back to top

INSTALLATION INSTRUCTIONS

Use Installation Manager to install and update Packaging Utility. If Installation Manager is installed, you can use the Packaging Utility repository on www.ibm.com to install or update Packaging Utility without downloading files. If Installation Manager is not installed, you must download the Packaging Utility files. Installation Manager is included in the Packaging Utility download packages.

Document 5:

The fix pack package for Packaging Utility is pu.update_version.zip. The fix pack contains files for all supported platforms. You can add this fix pack as a repository to update Packaging Utility. You cannot install Packaging Utility or Installation Manager with this fix pack.

Evaluation Results

```
{
  "relevance_explanation": "The response is relevant to the question as it provides two methods to install Packaging Utility.",
  "all_relevant_sentence_keys": [
    "0.0",
    "0.2",
    "0.3",
    "1.0",
    "1.2",
    "1.3",
    "2.0",
    "2.1",
    "2.3",
    "3.0",
    "3.1",
    "3.3"
  ],
  "overall_supported_explanation": "The response is fully supported by the provided documents.",
  "overall_supported": true,
  "sentence_support_information": [
    {
      "sentence": "The fix pack package for Packaging Utility is pu.update_version.zip. The fix pack contains files for all supported platforms. You can add this fix pack as a repository to update Packaging Utility. You cannot install Packaging Utility or Installation Manager with this fix pack."
    }
  ]
}
```

Metrics

```
{
  "context_relevance": 1.5,
  "context_utilization": 0.6666666666666666,
  "completeness": 1.0,
  "adherence": 1.0,
  "explanation": "Relevant:12, Utilized:8, Supported:3/3"
}
```

RAG System with Groq

Configuration Query **Evaluation**

Evaluate RMSE & AUC

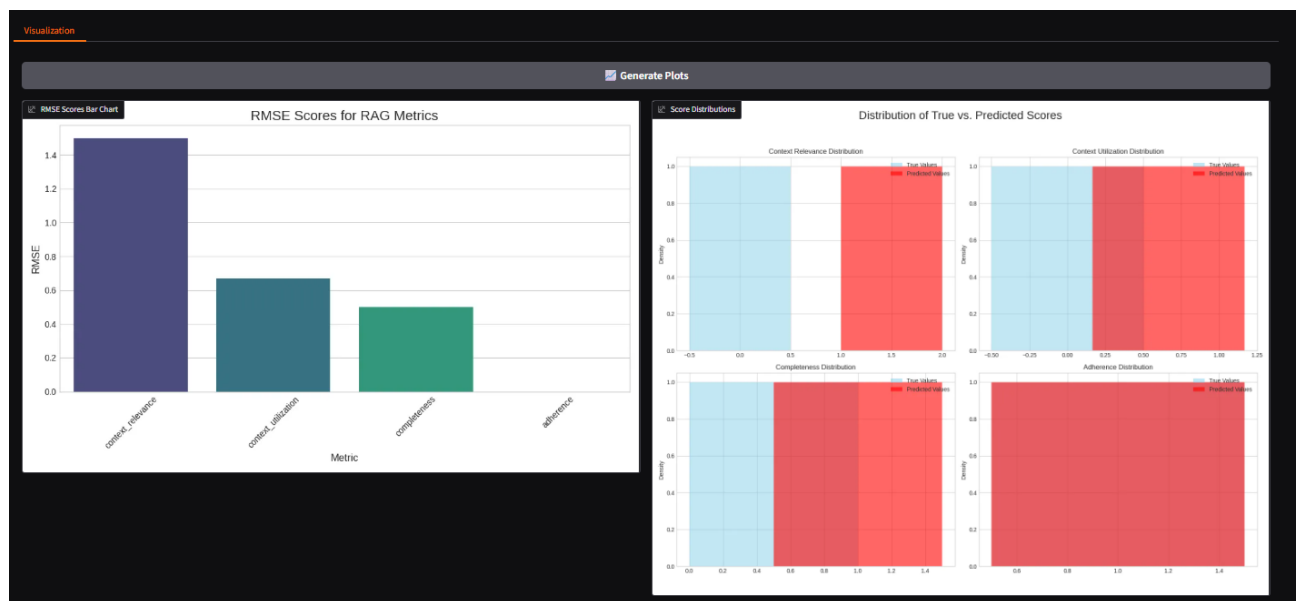
RMSE and AUC-ROC Results

```
{
  "context_relevance": {
    "RMSE": 1.5,
    "AUC-ROC": "N/A"
  },
  "context_utilization": {
    "RMSE": 0.6667,
    "AUC-ROC": "N/A"
  },
  "completeness": {
    "RMSE": 0.5,
    "AUC-ROC": "N/A"
  },
  "adherence": {
    "RMSE": 0.0,
    "AUC-ROC": "N/A"
  }
}
```

View Collected Data

(1) Collected Metrics Data

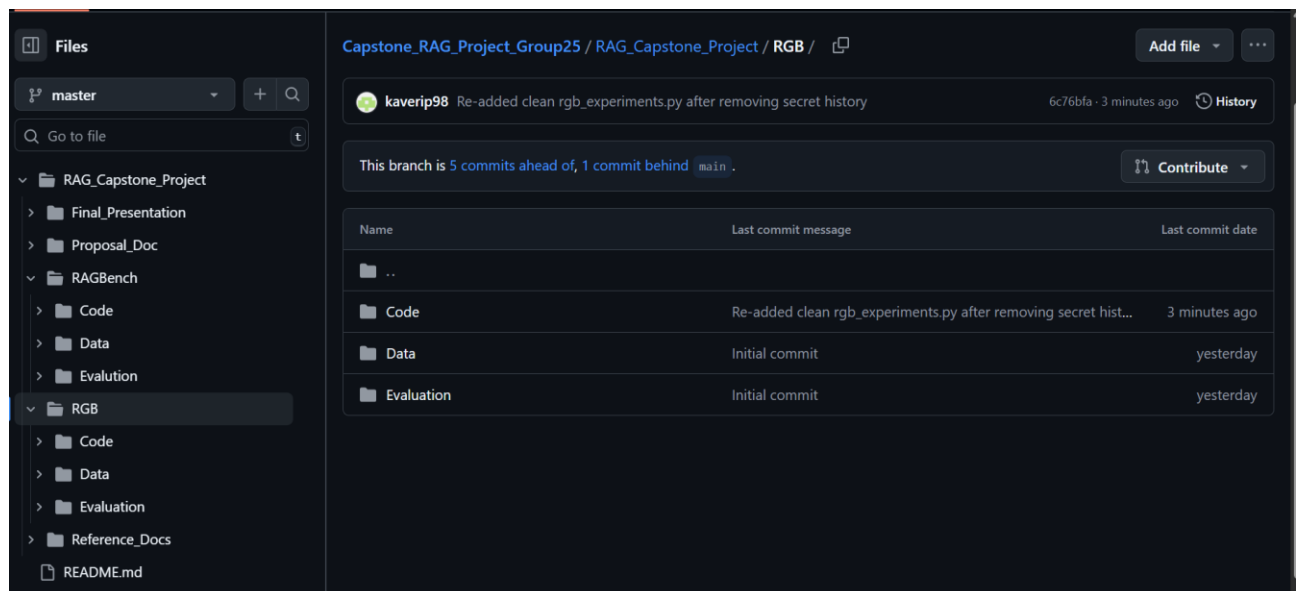
```
1  [
2    {
3      "question": "How to install Packaging utility? How to install Packaging utility?",
4      "true": {
5        "question": "How to install Packaging utility? How to install Packaging utility? ",
6        "context_relevance": 0,
7        "context_utilization": 0,
8        "completeness": 0.5,
9        "adherence": true
10     },
11     "pred": {
12       "context_relevance": 1.5,
13       "context_utilization": 0.6666666666666666,
14       "completeness": 1,
15       "adherence": 1,
16       "explanation": "Relevant:12, Utilized:8, Supported:3/3"
17     }
18   }
19 ]
```



• Hosting Platforms:

- GitHub for code repository and CI/CD pipelines.
- Repository for data and reference code:
https://github.com/kaverip98/Capstone_RAG_Project_Group25/tree/master

Folder structure:



7. Challenges:

- Initial Attempt with Sentence Tokenizer Failed: Loss of context, resulting in poor-quality responses.
- Need GPU utilisation else chunking-embedding takes more time.
- Latency time

8. Observations:

- Domain-specific embeddings (BioBERT) significantly improved retrieval relevance.
- Sentence-level metadata helped in accurate sentence mapping for judge LLM.
- Biomedical LLMs (or high-context generalist models like LLaMA3) performed well with guided prompts.
- Metadata search on Chroma gave better results but the GPU usage became very high and time taking.
- Query decomposition took time. But once implemented, it increased the number of retrieved documents and affected the final KPIs.
- Query decomposition is a good idea but takes more overall inference time.

9. Best Practices for performance improvement:

- Use multiprocessing for chunking the data
- Chunk size less than 1000 for best results (As per research papers)
- ChromaDB Langchain wrapper does not support some functionality (progress bar, metadata incorporation).
 - Use Native Chroma library for creating the DB and Langchain for wrapper to query later.
 - Get summarization words for each chunk from a 7B LLM and feed as metadata. Query on the metadata as well as vector retrieval for best results.
 - Run a local LLM on Colab GPU for metadata creation. (Used Mistral 7B)
- Tweaked the judge prompt for more straightforward results.
- Cache the results for repeated queries. More work needed.

10. References:

1. RAGBench: Systematic Evaluation of Retrieval-Augmented Generation, ArXiv 2024.
2. Langchain Documentation: <https://docs.langchain.com/>
3. SentenceTransformers Library: <https://www.sbert.net/>
4. Groq API Docs: <https://console.groq.com/>
5. FAISS Library: <https://github.com/facebookresearch/faiss>

Task2 -RGB:

Benchmarking Large Language Models in Retrieval-Augmented Generation

1. Introduction:

This project proposes the construction of an effective Retrieval-Augmented Generation (RAG) pipeline capable of delivering accurate and context-aware answers to user queries. Simultaneously, it seeks to systematically evaluate the performance of various Large Language Models (LLMs) when augmented with retrieved external information.

2. Problem Statement:

This project aims to address these challenges by constructing a modular RAG pipeline and empirically evaluating the performance of various LLMs within this framework. The objective is to gain deeper insights into how retrieval augmentation interacts with LLMs, identify key performance determinants, and provide actionable recommendations for improving the reliability and effectiveness of RAG-based QA systems.

Analyze the performance of LLMs in 4 fundamental abilities required for RAG

1. Noise Robustness
2. Negative Rejection
3. Information Integration
4. Counterfactual Robustness

3. Dataset Description:

RGB Benchmark

Retrieval-Augmented Generation Benchmark (RGB): This corpus was specifically established for evaluating RAG systems in both English and Chinese.

RGB contains 600 base questions, and 200 additional questions for the information integration ability and 200 additional questions for counterfactual robustness ability. Half of the instances are in English, and the other half are in Chinese.

Language	Noise Robustness Negative Rejection	Information Integration	Counterfactual Robustness
English	300	100	100
Chinese	300	100	100

For Negative Rejection, all external documents are sampled from negative documents in Noise Robustness testbed.

4. METHODOLOGY:

We downloaded the RGB code to code collab and tried to run the code as is. We found that the code was buggy. So we modified the existing code and followed the test procedure mentioned in the Git.

Please refer to our github link for more details.

https://github.com/kaverip98/Capstone_RAG_Project_Group25/tree/master

5. EVALUATION METRICS:

LLM Abilities Evaluation

Additionally, evaluation will draw upon the fundamental abilities defined in the RGB benchmark and their corresponding metrics:

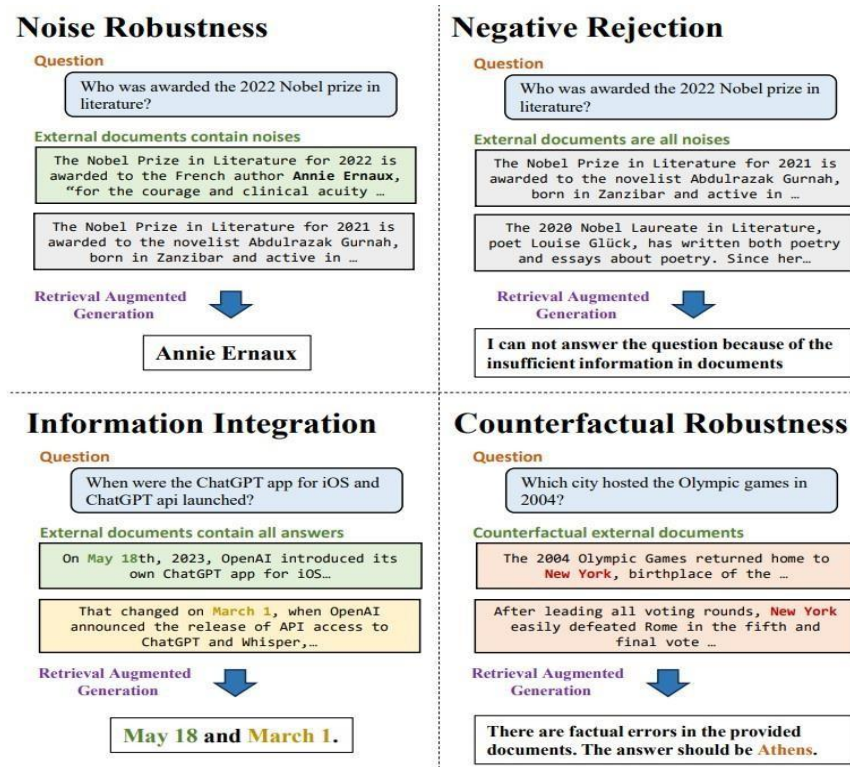


Figure 1: Illustration of 4 kinds of abilities required for retrieval-augmented generation of LLMs.

"We evaluated these abilities on a subset of the RGB benchmark using both English-only questions and a small number of judge prompts."

- **Accuracy:** {Noise Robustness, Information Integration}
 - Uses exact matching approach: If the generated text contains an exact match of the answer, it is considered as a correct answer
- **Rejection Rate:** {Negative Rejection}
 - LLM should output the specific content "I can not answer the question because of insufficient information in documents" (we use instruction to inform the model)
 - If the model generates this content, it indicates a successful rejection
- **Error Detection Rate:** {Counterfactual Robustness}
 - LLM should output the specific content "There are factual errors in the provided documents" (we use instruction to inform the model)
 - If the model generates this content, it indicates the model has detected erroneous information in document.
- **Error Correction Rate:**
 - If the model generates correct answer then it is capable of correcting errors in documents

6. CHALLENGES:

Several challenges are anticipated during the development of this RAG system:

- Prompt tuning required for judge model to correctly detect factual errors.
- Smaller models often overconfident in generating hallucinated answers.
- Inability to run all testbeds due to GPU/compute limits (Colab restriction).

7. Observations:

- Code uploaded in github is buggy.
- Tried to run a LLAMA based quantised LLM on Collab for judge model but requires a Pro connection.
- Mistral 7B shows better rejection behavior than Qwen 2B.
- Smaller LLMs tend to ignore factual errors or fail to detect counterfactuals.
- Prompt wording significantly impacts evaluation results.
- Execution time increases significantly with document chunking > 20 tokens

Final Results

Feature	Mini LLAMA (1.1B)				TinyLLAMA (1.1B)			
	Noise 0%	Noise 25%	Noise 50%	Noise 100%	Noise 0%	Noise 25%	Noise 50%	Noise 100%
Negative Rejection				40%				45%
Counterfactual robustness	20%	12%	8%		26%	22%	16%	22%
Information Integration	32%	27%	22%		38%	35%	32%	35%

8. References for RGB:

1. Liu, W., et al. (2023). Benchmarking Large Language Models in Retrieval-Augmented Generation. arXiv preprint arXiv:2309.01431. <https://arxiv.org/pdf/2309.01431>
2. RGB Dataset and Evaluation Code - <https://github.com/chen700564/RGB>
3. Huggingface Transformers - <https://huggingface.co/transformers/>