A Performance Evaluation of Retrieval Augmented Generation in Large Language Models

A RESEARCH PROJECT PROPOSAL FOR NLP COURSEWORK

Submitted by:

Lakshay Chandra (2K20/IT/79)

DEPT. OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
Bawana Road, Delhi-110042

INTRODUCTION

Retrieval-Augmented Generation with Advanced Language Models

In recent times, the utilization of advanced language models (such as GPT-4) has seen remarkable growth. These models, known as Large Language Models (LLMs), have become invaluable tools for a wide range of tasks, from simple functions like setting reminders and handling emails to more intricate activities such as drafting research papers, coding software, and aiding in artistic endeavors. LLMs have proven their adaptability across diverse domains, showcasing their capabilities in various applications.

Particularly in the field of medicine, LLMs have demonstrated potential in interpreting complex datasets, searching patient records, and generating synthetic text data. Their versatility is attributed to the extensive training datasets and sophisticated architectures that enable them to produce human-like textual responses in real-time.

However, similar to any tool, LLMs have their limitations. One significant challenge is the occurrence of "hallucination" errors, where the model generates information that is either incorrect or not present in its training data. In critical areas like medicine, such errors could lead to misinterpretations and, in severe cases, adverse outcomes for patients. The central issue lies in the fact that while LLMs can generate content that sounds plausible, they lack an inherent mechanism to verify the factual accuracy against reliable data sources.

This project delves into Retrieval Augmented Generation (RAG), an approach designed to address hallucination errors in LLMs. RAG combines the robust generative capabilities of LLMs with the precision of retrieval-based models. In the RAG process, when a query is initiated, the model first retrieves pertinent documents or data snippets from a vast pool of documents (whether pre-existing or provided by the user). Subsequently, this retrieved information is utilized in the generation phase to craft a response. By leveraging the strengths of both retrieval and generation models, RAG aims to offer more accurate and contextually relevant answers. In medical applications, the use of RAG holds the potential to ensure that responses are not only contextually rich but also grounded in accurate data, enhancing the reliability of the model's outputs.

METHODOLOGY

The project methodology consists of several key steps, aiming to implement Retrieval-Augmented Generation (RAG) using the LangChain library on a Question Answering informatics dataset obtained from Kaggle. The detailed methodology is outlined below without delving into the intricacies of the code.

Dataset Exploration:

- Utilized the Kaggle dataset "Question-Answer Dataset" (https://www.kaggle.com/datasets/rtatman/questionanswer-dataset).
- Displayed the number of unique questions and examined the structure of the dataset, revealing 2463 unique questions with columns providing information such as article title, question, answer, difficulty levels, etc.

Data Cleaning:

- Eliminated unnecessary columns, specifically the last column (extra Article Title).
- Addressed missing values in the "Question" or "Answer" fields to ensure clean and complete data.
- Post-cleaning, the dataset was reduced to 2190 unique and clean question-answer pairs ready for subsequent processing.

Data Collection and Preprocessing:

- Created a dataset for retrieval augmentation by collecting textual data from .clean files in the "text_data" directory.
- Initialized a directory path and stored the content of each text file in a dictionary, establishing a robust dataset for subsequent RAG model training.

Regular Question-Answering with LLAMA2:

- Established a baseline using LLAMA 2, a state-of-the-art open-source language model, to understand the performance of traditional Large Language Models (LLMs) without retrieval augmentation.
- Configured and loaded the LLAMA 2 model, setting the stage for later comparison with RAG.
- Used 4-bit quantization to load the model on limited memory.

Tokenization:

- Utilized tokenization to convert textual input into a format understandable by LLAMA 2.
- Initialized a tokenizer associated with the LLAMA 2 model to prepare questions, paragraphs, or other textual data for model processing.

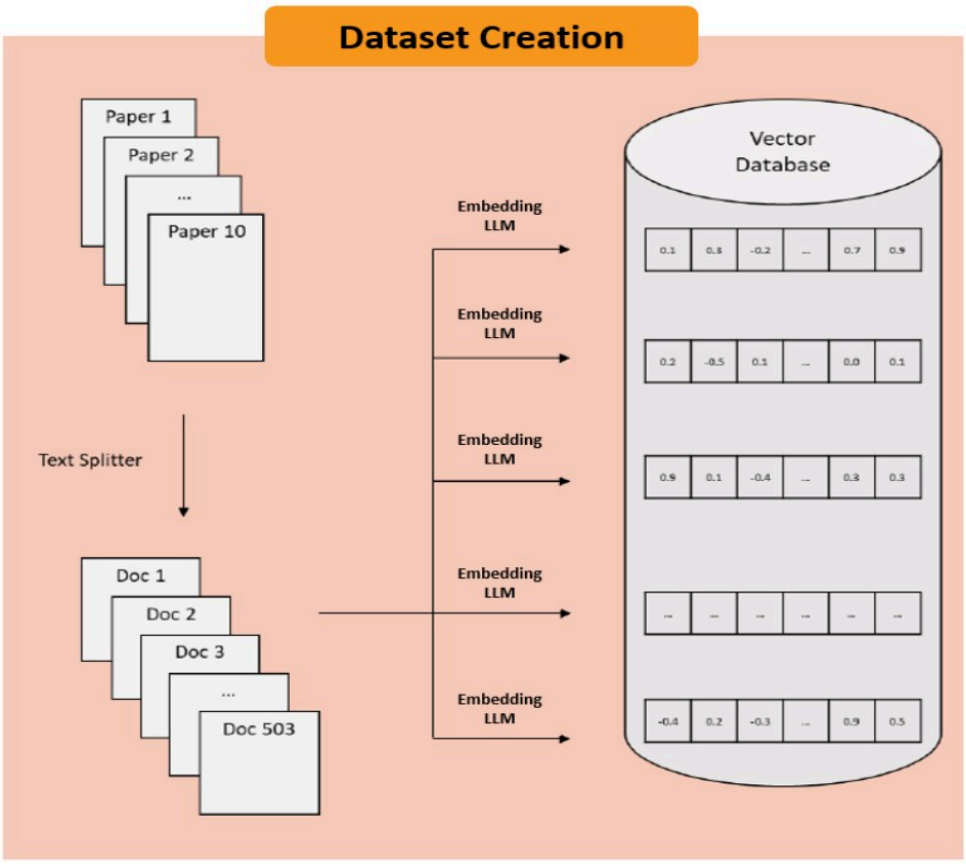
Language Model Pipeline for Text Generation:

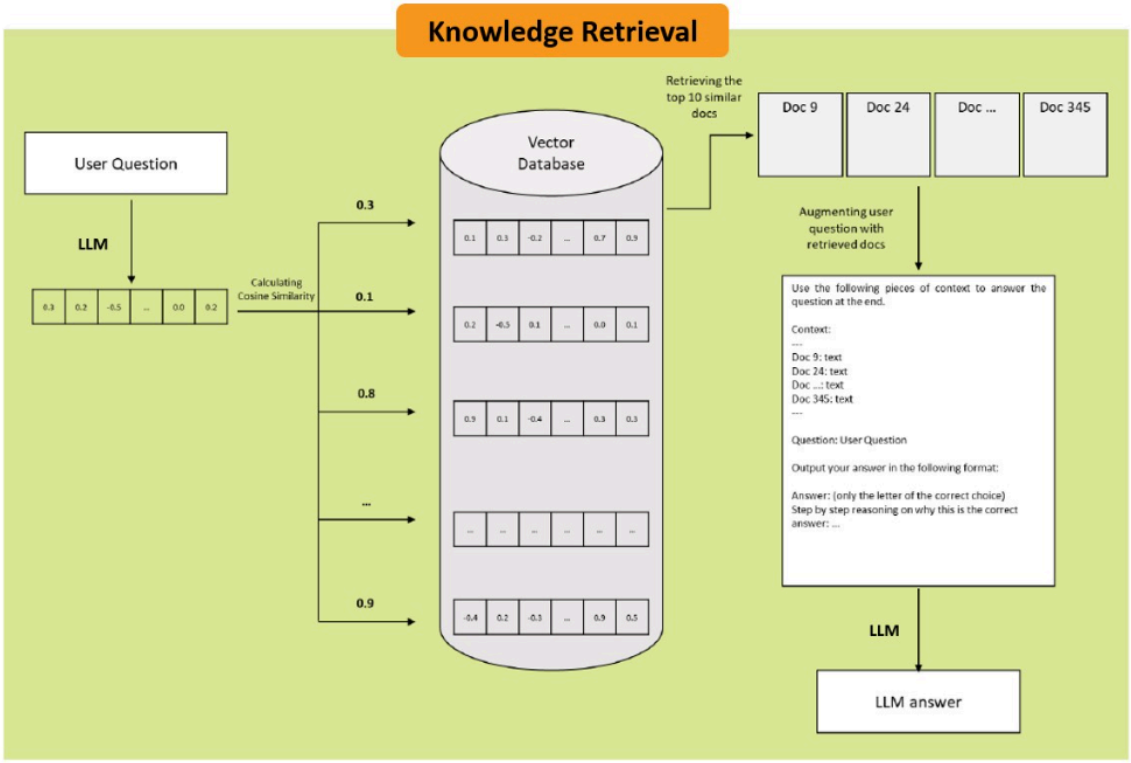
- Set up a text generation pipeline using the loaded LLAMA 2 model and its corresponding tokenizer.
- Configured parameters such as temperature, max_new_tokens, and repetition_penalty for controlled and efficient model output.

Understanding RAG: A Simplified Overview:

Here's a step-by-step breakdown of the process:

1. **Embedding Knowledge**: Initially, the RAG system scans all the books (or documents) in the library, understanding their content, and converting each page into a digital fingerprint, or "vector". These vectors are stored in a special digital catalog.
2. **Question Analysis**: Now, when you ask the librarian a question, she instantly translates your question into a similar digital fingerprint to know what to look for in the catalog.
3. **Finding Relevant Information:** Using the fingerprint of your question, the librarian quickly searches the catalog to find the pages (or chunks of data) most closely related to your query, as if comparing the similarities between the patterns of two fingerprints.
4. **Crafting the Response:** With the relevant pages in hand, the librarian now composes a well-informed answer, ensuring it's based on the information from the library. This answer is not just from memory but is augmented by the recent information she retrieved.
5. **Delivering the Answer:** The librarian then shares her comprehensive response with you.


Dataset Creation
Paper 1
Paper 2
...
Paper 10
Text Splitter
Doc 1
Doc 2
Doc 3
...
Doc 503
Embedding LLM
Embedding LLM
Embedding LLM
Embedding LLM
Embedding LLM
Vector Database


Knowledge Retrieval
User Question
LLM
Calculating Cosine Similarity
0.3
0.1
0.8
...
0.9
Vector Database
Retrieving the top 10 similar docs
Doc 9
Doc 24
Doc ...
Doc 345
Augmenting user question with retrieved docs
Use the following pieces of context to answer the question at the end.
Context:

Doc 9: text
Doc 24: text
Doc ...: text
Doc 345: text

Question: User Question
Output your answer in the following format:
Answer: (only the letter of the correct choice)
Step by step reasoning on why this is the correct answer: ...
LLM
LLM answer

Loading Texts into Langchain Documents:

- Employed the LangChain library to preprocess data for RAG, initializing an empty list to store processed documents.
- Created metadata for each source, converted text content into LangChain Document objects, facilitating better management and querying.

Chunking the Texts:

- Implemented text chunking for efficient document retrieval, especially for large text files like medical records or literature.
- Split documents into smaller text chunks, enhancing indexing, storage, and retrieval.

Setting Up the Embedding Model:

- Established embeddings for retrieval tasks, transforming textual data into numerical vectors in a high-dimensional space.
- Utilized ChromaDB, a vector store, to store and retrieve vector embeddings, ensuring efficient searching and matching of related content.
- Populated the vector store with document embeddings and saved them for future usage.

Demonstrating Retrieval Process:

- Explored the retrieval process using cosine similarity, demonstrating how the model identifies and fetches relevant information from the vector database based on a sample query.

Setting Up a RAG Pipeline:

- Assembled a RAG pipeline using LangChain, combining the Large Language Model (LLM) and the vector database for retrieval-augmented text generation.
- Ensured that responses generated by the pipeline are grounded in chunks of texts extracted from the vector database, enhancing the reliability and relevance of the generated content.

This comprehensive methodology sets the foundation for implementing Retrieval-Augmented Generation on the medical informatics dataset, incorporating data cleaning, preprocessing, model training, and vector-based retrieval for enhanced question-answering capabilities.

Performance Metrics

When evaluating the performance of Retrieval-Augmented Generation (RAG) in Large Language Models (LLM), several metrics can be considered to assess the effectiveness of the model in generating relevant and coherent responses. Here are some performance metrics used in the project:

Primary Metrics: *The initial set of metrics utilized to quantify hallucinations in the model, both with and without retrieval augmentation, are statistically derived and are not based on external language models.*

1. F1 Score:

- **Definition:** F1 Score is the harmonic mean of precision and recall. In the context of hallucination detection, it measures the balance between identifying true positives (non-hallucinated content) and avoiding false positives (incorrectly flagging hallucination).
- **Pros:**
    - Simple and intuitive metric.
    - Takes both false positives and false negatives into account.
- **Cons:**
    - F1 Score may not provide a detailed understanding of the model's performance, especially in scenarios where false positives and false negatives have different implications.

2. BLEU Score:

- **Definition:** BLEU (Bilingual Evaluation Understudy) Score measures the similarity between the predicted text and reference text based on n-gram overlap.
- **Pros:**
    - Sensitive to n-gram matches, providing a nuanced evaluation.
    - Widely used in machine translation and natural language generation tasks.
- **Cons:**
    - Ignores word order, which may be important for hallucination detection.
    - Can be sensitive to small variations in generated content.

3. ROUGE Score (ROUGE-N and ROUGE-L):

- **Definition:** ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score measures the overlap of n-grams (ROUGE-N) and the longest common subsequence (ROUGE-L) between the predicted and reference text.

- **Pros:**
    - ROUGE-N captures n-gram overlap, providing granularity.
    - ROUGE-L considers the longest common subsequence, useful for capturing content overlap.
- **Cons:**
    - Similar to BLEU, ROUGE may not consider word order adequately.
    - Sensitivity to specific n-gram lengths may affect results.

4. METEOR Score:

- **Definition:** METEOR (Metric for Evaluation of Translation with Explicit ORdering) Score considers precision, recall, stemming, synonymy, and word order.
- **Pros:**
    - Takes into account various linguistic aspects, including word order.
    - Offers a comprehensive evaluation of generated text.
- **Cons:**
    - Requires pre-processing, including stemming and synonymy, which may introduce complexities.
    - Interpretability may be challenging due to the combination of multiple factors.

5. Context Overlap Score:

- **Definition:** Context Overlap Score calculates the Jaccard similarity between sets of tokens in the reference and predicted text.
- **Pros:**
    - Focuses on the overlap of content, addressing hallucination concerns.
    - Considers the set of tokens, providing flexibility.
- **Cons:**
    - May not capture subtle contextual differences.
    - Sensitive to tokenization choices and may require careful pre-processing.

General Considerations:

- **Subjectivity:** Hallucination detection involves subjective judgment. Metrics may not fully capture the qualitative aspects of hallucination, requiring additional human evaluation.
- **Trade-offs:** Different metrics may emphasize precision, recall, or linguistic aspects. Choosing a metric depends on the specific goals and priorities of the evaluation.

Secondary Metrics: The supplementary set of metrics, primarily derived from large language models, includes the following:

1. **Natural Language Inference (NLI):**
    - **Definition:** NLI assesses the model's ability to understand the logical relationships between statements. It typically involves tasks such as recognizing entailment, contradiction, or neutral relationships between pairs of sentences.
    - **Pros:**
        - Provides insights into the model's comprehension of nuanced language relationships.
        - Useful for evaluating the logical consistency of generated content.
    - **Cons:**
        - Requires labeled datasets for training and evaluation.
        - This is also sensitive to choice of tokenizer as we see in the results
2. **BERTScore:**
    - **Definition:** BERTScore measures the similarity between the predicted and reference text using contextual embeddings from BERT (Bidirectional Encoder Representations from Transformers).
    - **Pros:**
        - Leverages contextual embeddings for a more sophisticated evaluation.
        - Considers word order and context, addressing limitations of traditional metrics.
    - **Cons:**
        - Computationally intensive, especially for large datasets.
        - This is also sensitive to choice of tokenizer as we see in the results
3. **Diversity (Based on SentenceBERT):**
    - **Definition:** Diversity metrics, particularly those based on SentenceBERT, assess the variety and distinctiveness of generated content from a reference text. This involves evaluating how different the generated sentences are from each other and from the expected response.
    - **Pros:**
        - Provides insights into the diversity of responses generated by the model.
        - Useful for avoiding repetitive or redundant answers.
    - **Cons:**
        - Definition of diversity may vary based on the specific metric used.

- Sensitive to the choice of similarity measures and thresholds.

---

*This insight is leveraged by SelfCheckGPT (Manakul, P., Liusie, A., & Gales, M. J. F. (2023). SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models.), as several samples of the same prompt are drawn, and used to detect inconsistencies among them. The higher the inconsistencies, the more likely the LLM is hallucinating.

RESULT AND DISCUSSION

I) Primary Metrics

Metric	Mean (LLM)	Mean (RAG)
F1 Score	0.43	0.59
BLEU Score	0.29	0.39
Meteor Score	0.29	0.42
Context Overlap	0.25	0.36
Rouge L Score	0.32	0.42
Rouge N Score	0.29	0.39

1. **F1 Score:**

   RAG achieves a higher F1 Score than LLM. This indicates that RAG strikes a better balance between precision and recall, demonstrating improved accuracy in identifying relevant information and minimizing false positives or negatives.

2. **BLEU Score:**

   RAG surpasses LLM in BLEU Score, suggesting that RAG's generated content aligns more closely with the reference text. The contextual embeddings from retrieval contribute to a more accurate assessment of similarity, resulting in a higher BLEU Score.

3. **Meteor Score:**

   RAG exhibits a superior Meteor Score compared to LLM. This implies that RAG's responses not only align well with the reference but also incorporate various linguistic aspects, including stemming and synonymy, leading to a more comprehensive evaluation.

4. **Context Overlap:**

RAG demonstrates a higher Context Overlap than LLM. This suggests that the content generated by RAG exhibits greater overlap with the reference, indicating improved relevance and a reduced likelihood of hallucination.

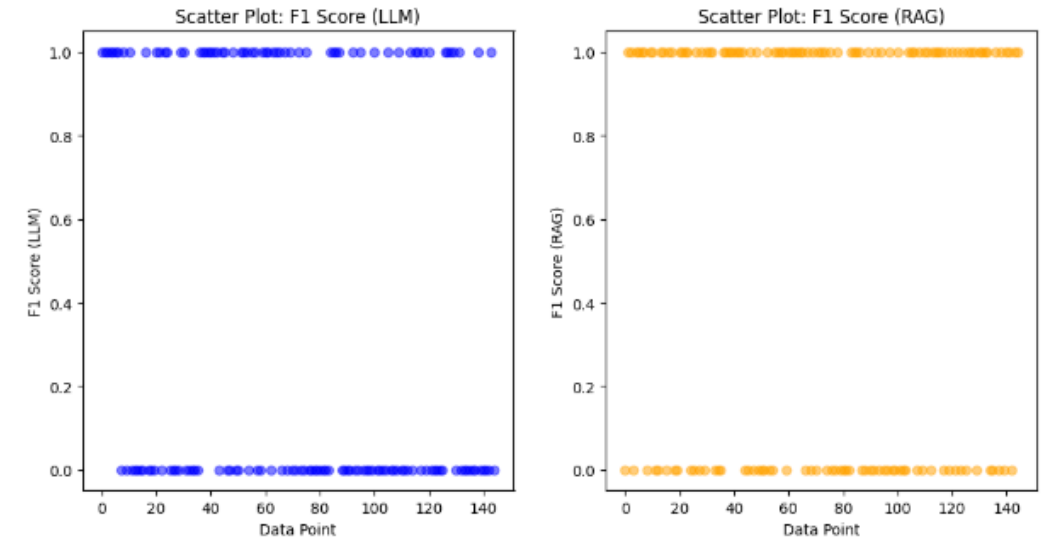
5. **Rouge L Score:**

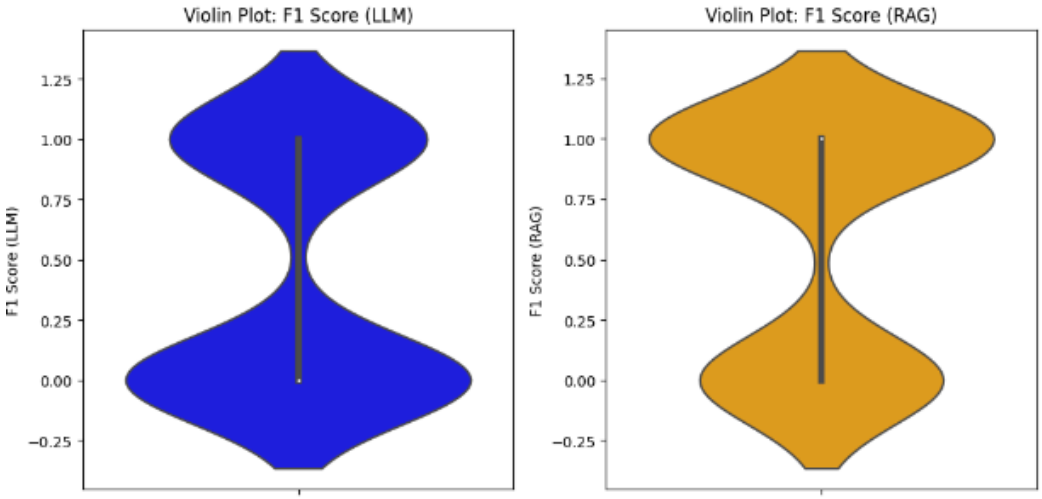
   RAG outperforms LLM in Rouge L Score, indicating that RAG generates responses with a higher overlap in the longest common subsequence with the reference. This reflects better content alignment and coherence in RAG-generated responses.

6. **Rouge N Score:**

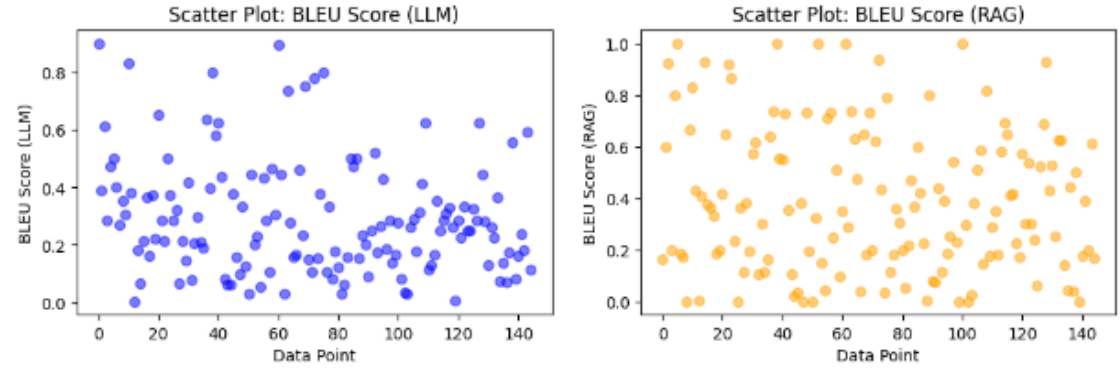
   RAG achieves a higher Rouge N Score compared to LLM. This signifies that RAG generates content with improved overlap in n-grams, capturing more relevant information from the reference and reducing the likelihood of hallucination.

F1

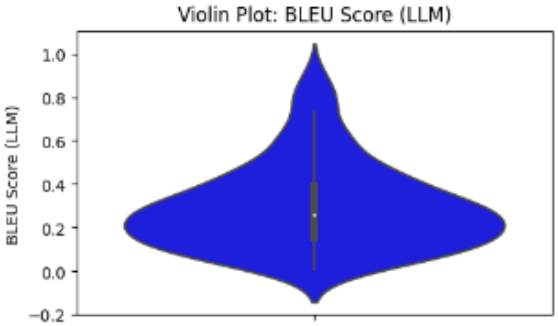

Scatter Plot: F1 Score (LLM)
F1 Score (LLM)
Data Point
Scatter Plot: F1 Score (RAG)
F1 Score (RAG)
Data Point

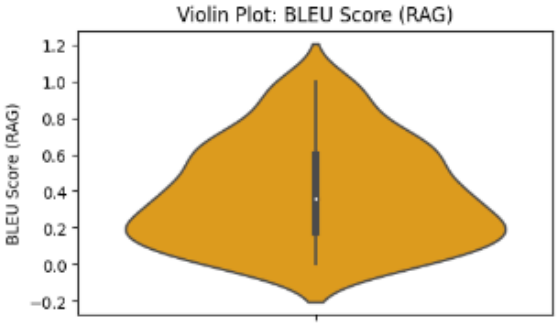

Violin Plot: F1 Score (LLM)
F1 Score (LLM)
Violin Plot: F1 Score (RAG)
F1 Score (RAG)

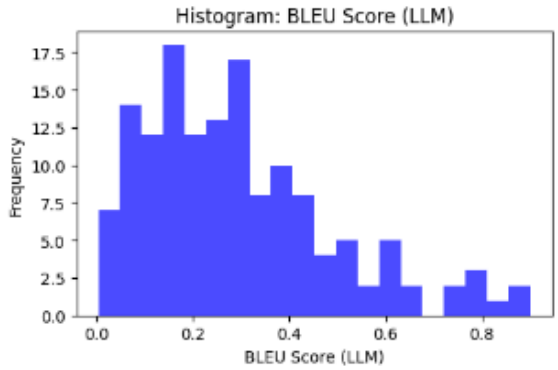
BLEU:

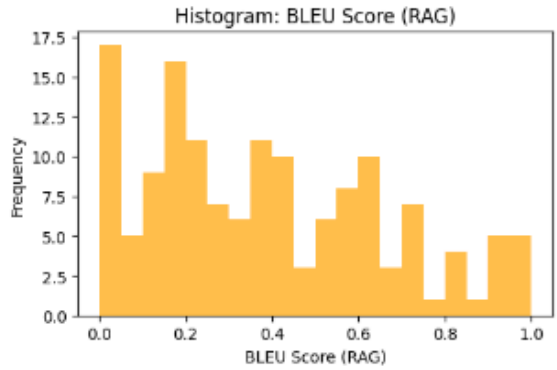

Scatter Plot: BLEU Score (LLM)
BLEU Score (LLM)
Data Point

Scatter Plot: BLEU Score (RAG)
BLEU Score (RAG)
Data Point

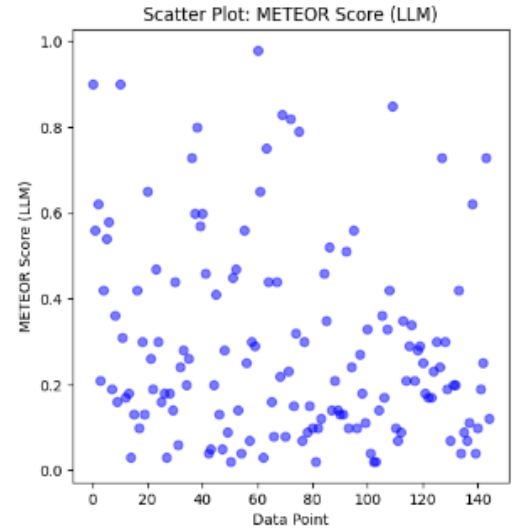

Violin Plot: BLEU Score (LLM)
BLEU Score (LLM)

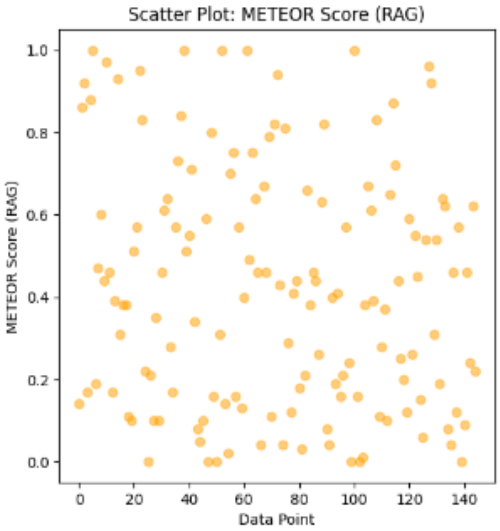

Violin Plot: BLEU Score (RAG)
BLEU Score (RAG)

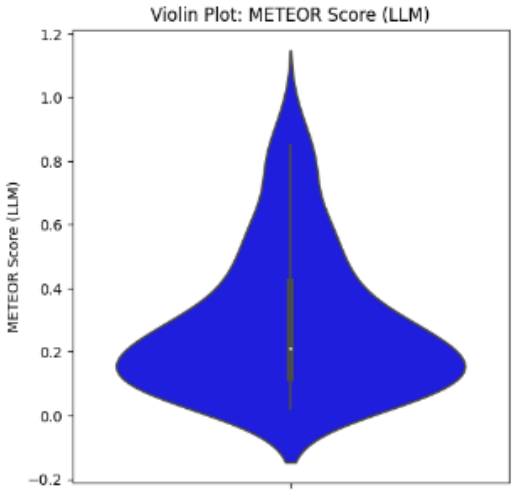

Histogram: BLEU Score (LLM)
Frequency
BLEU Score (LLM)

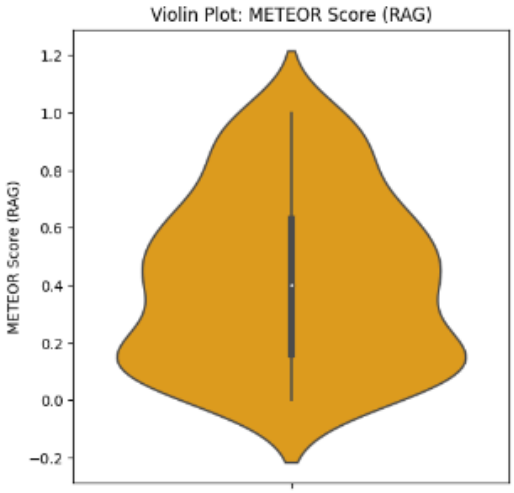

Histogram: BLEU Score (RAG)
Frequency
BLEU Score (RAG)

METEOR:

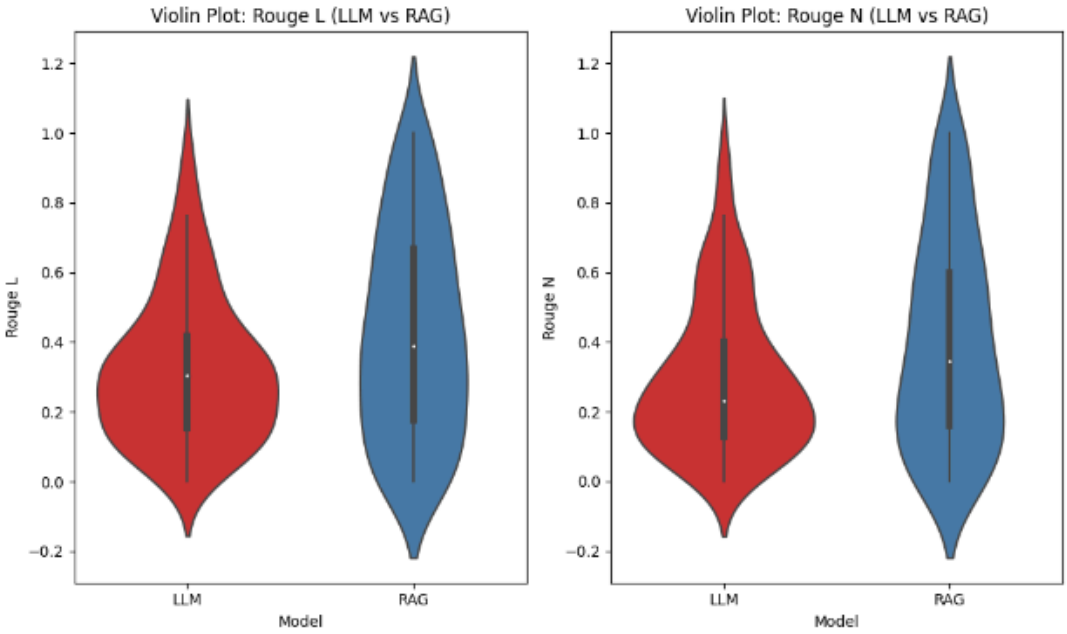

Scatter Plot: METEOR Score (LLM)
METEOR Score (LLM)
Data Point

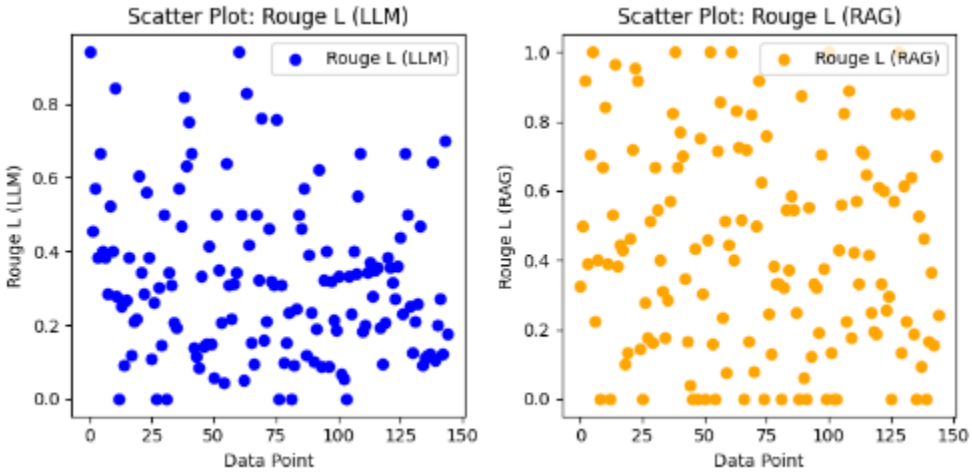

Scatter Plot: METEOR Score (RAG)
METEOR Score (RAG)
Data Point


Violin Plot: METEOR Score (LLM)
METEOR Score (LLM)

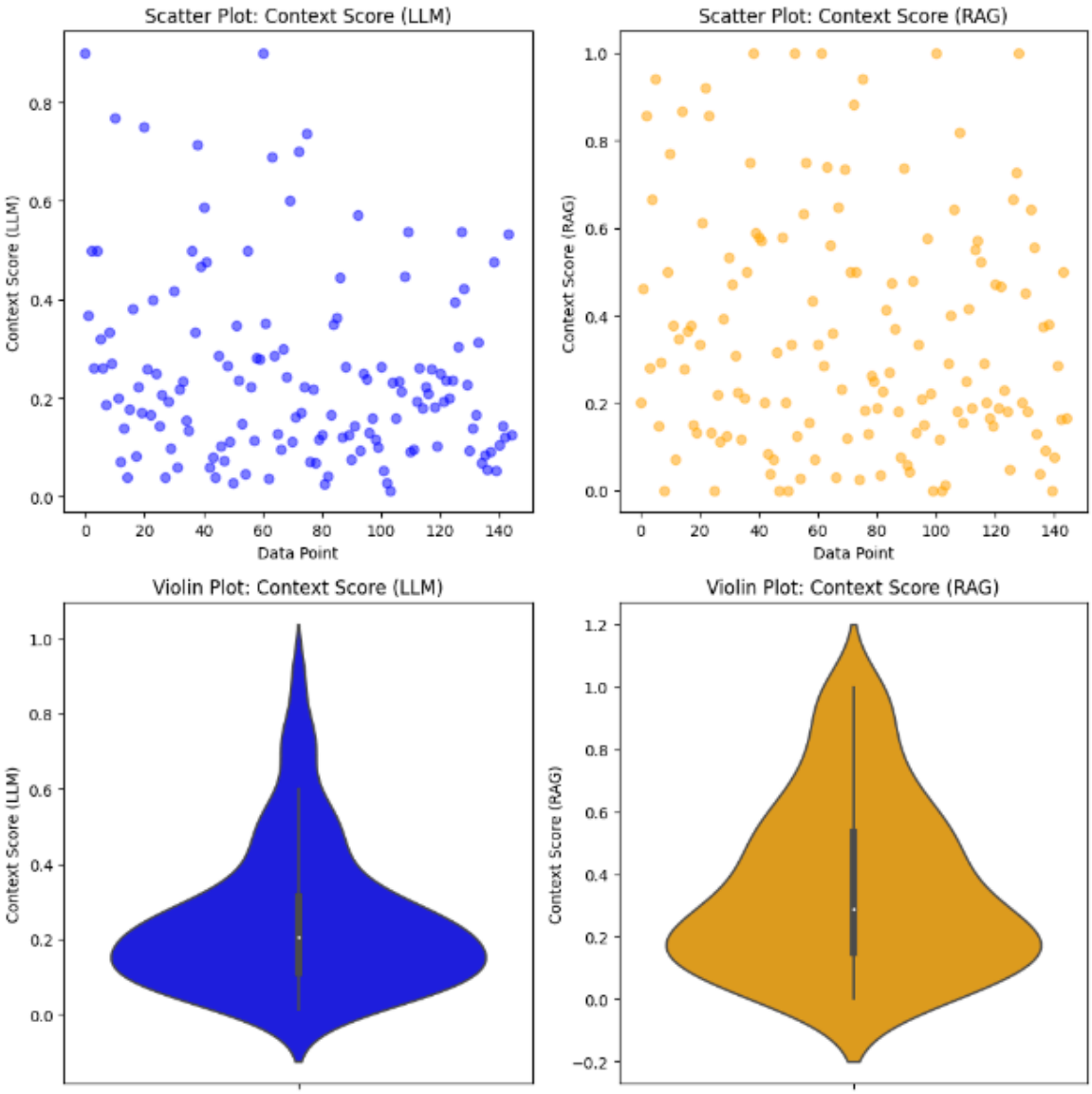

Violin Plot: METEOR Score (RAG)
METEOR Score (RAG)

ROUGE





Context Score:



Explanation:

The visual comparison between two text-generating models, namely the Large Language Model (LLM) and Retrieval-Augmented Generation (RAG), is presented through scatter and violin plots focusing on the context score—a metric indicating the similarity between generated and actual responses.

Scatter Plots: The top row exhibits scatter plots with the y-axis representing the context score and the x-axis showcasing the data points. The left scatter plot corresponds to the LLM model, denoted by blue points, while the right scatter plot pertains to the RAG model, marked with orange points. Notably, the LLM scatter plot reveals higher density for lower values (0-0.3) and significantly lower density for higher values. Conversely, the RAG scatter plot is more spread out, indicating a greater number of data points with higher context scores.

Violin Plots: The bottom row features violin plots illustrating the context scores on the y-axis and density. The width of each violin signifies the density of data points at different values, with wider sections representing higher density. The height of the violin reflects the range of the data. The left violin plot corresponds to the LLM model (blue violin), while the right violin plot corresponds to the RAG model (orange violin). Notably, the RAG model's violin exhibits a slower decrease in width, particularly in the higher value range (0.8-1), suggesting a larger number of data points with elevated context scores.

Consequently, the observation indicates that the RAG model performs **slightly better** than the LLM model in terms of context score distribution.

II) Secondary Metrics

Metric	LLM	RAG
Mean NLI Score	0.74	0.46
Mean BERTScore	0.78	0.67
Mean Diversity	0.54	0.43
MIN(BERTSCORE, NLI)	0.62	0.39

1. **Mean NLI* Score:**

   RAG(0.46) has a lower mean NLI compared to LLM(0.74). A lower NLI score indicates better entailment between reference text (greedily sampled) and sample responses thereby a lower rate of hallucination. There is on an average 74 % chance of contradiction between reference text and generated response by LLM compared to a lesser (though still significant) 46% chance of contradiction in case of RAG.

2. **Mean BERTScore:**

   RAG(0.67) has a lower mean BERTScore compared to LLM(0.78). SelfCheckGPT uses BERTScore to calculate the average BERTScore of a sentence when compared with the most similar sentence from each selected sample. If the information contained in a sentence is found across many selected samples, it is reasonable to conclude that the information is factual. On the other hand, if the statement does not appear in any other sample, it may likely be a hallucination or an outlier. In our definition of bertscore, a lower bertscore implies a lower rate of hallucination.

*NLI score here corresponds to contradiction, hence lower the better, we need to subtract this score form 1 otherwise to get the entailment score

3. **Mean Diversity:**

   These values represent the average semantic diversity scores calculated using the cosine similarity metric between the ground truth answer and the generated responses for each model.

   The higher the diversity score, the more varied and different the responses are from the ground truth. In this case, LLM has a higher mean diversity score (0.54) compared to RAG (0.43). This suggests that, on average, LLM generates responses that are more diverse in semantic content compared to RAG.

4. **MIN(BERTSCORE, NLI) :**

   Hallucination Detection:

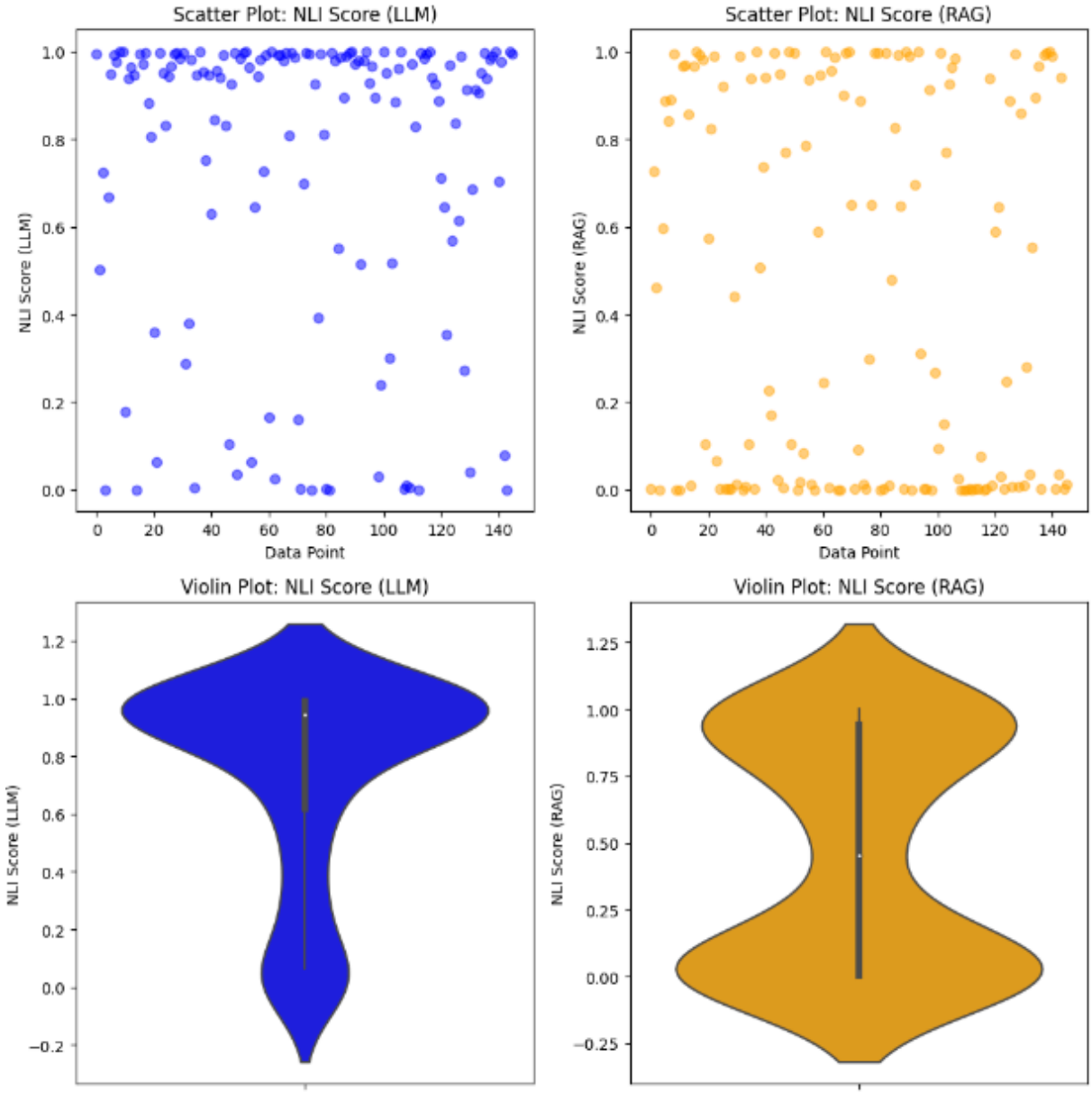
   - Hallucination refers to the generation of information that is not present in the input or context. Inconsistencies between semantic content and lexical similarity may indicate potential hallucination.
   - Taking the minimum helps identify cases where high BERTScore might be driven by lexical overlap while NLI suggests a contradiction or inconsistency.

   Robustness to Metric Biases:

   - Different metrics may capture different aspects of language quality and may exhibit biases. Taking the minimum helps mitigate the impact of individual metric biases and provides a more balanced evaluation.
   - In the course of our experiments, we noticed that NLI (Natural Language Inference) scores and BERTScores often provided unexpected or inconsistent results for some set of samples. This discrepancy could be attributed to factors such as suboptimal tokenization, or biases and limitations inherent in the evaluation metrics.
   - To address this challenge, we found that taking the minimum of both NLI and BERTScore metrics proved to be a valuable strategy. By selecting the lower (better) of the two scores for a given sample, we aimed to overcome potential issues arising from individual metric peculiarities.

   LLM (0.62) has a higher combined metric (minimum between BERTScore and NLI) compared to RAG (0.39). This lower value suggests that RAG, in this particular aspect, has better alignment between semantic content and lexical similarity compared to LLM.

NLI



Explanation:

The visual comparison between two text-generating models, namely the Large Language Model (LLM) and Retrieval-Augmented Generation (RAG), is presented through scatter and violin plots focusing on the NLI score—a metric indicating the entialment between generated responses and actual response.

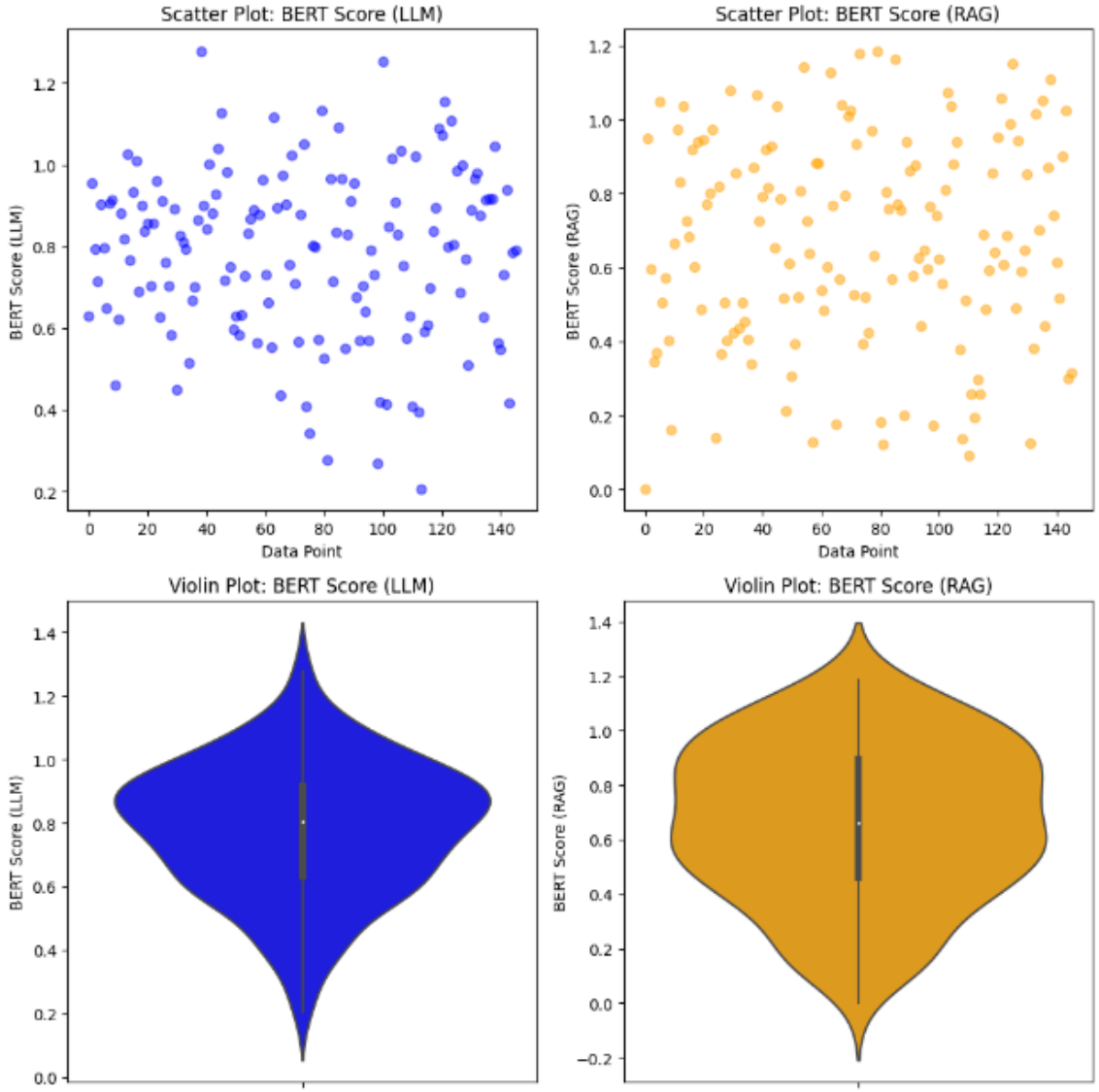
Scatter Plots: In the upper row, scatter plots depict the NLI scores on the y-axis against corresponding data points on the x-axis. The scatter plot on the left represents the LLM model (blue points), while the one on the right corresponds to the RAG model (orange points).

Notably, the LLM scatter plot demonstrates a higher concentration of data points at higher NLI values (0.8-1.0) and a noticeably lower density at lower values, suggesting a greater prevalence of higher NLI scores. Conversely, the RAG scatter plot exhibits higher density at the lower end (0-0.2) and a more dispersed pattern, indicating a larger number of data points with lower NLI scores.

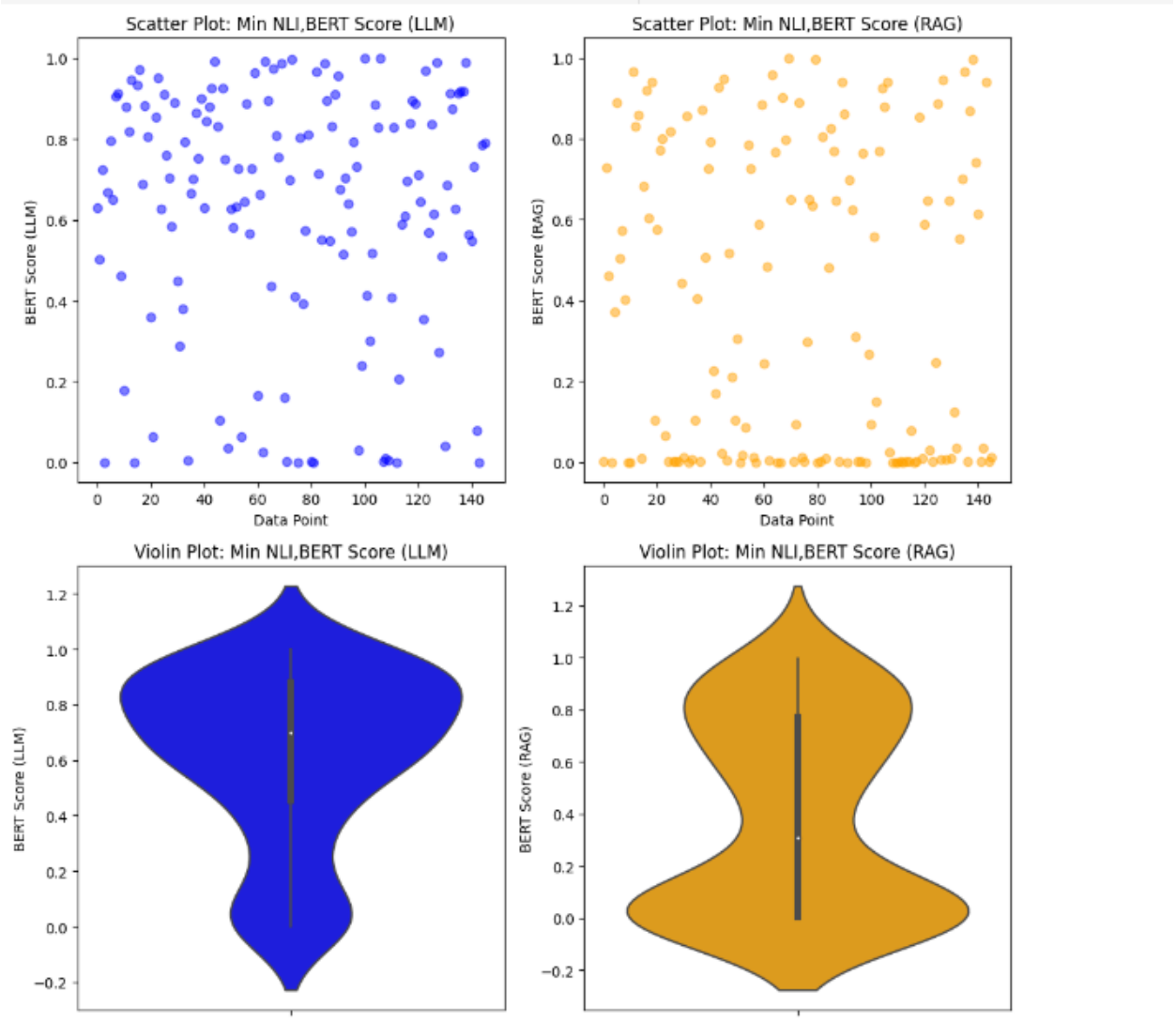
Violin Plots: The bottom row displays violin plots illustrating NLI scores on the y-axis and density as a function of width. The width of each violin represents the density of data points at different NLI values, with wider sections indicating higher density. The height of the violin reflects the range of the data. The left violin plot corresponds to the LLM model (blue violin), while the right violin plot corresponds to the RAG model (orange violin). Notably, the LLM model's violin is wider near 1.0 and narrower at 0, indicating a concentration of data points at higher NLI scores. In contrast, the RAG model's violin displays the opposite behavior, though to a lesser extent.

Consequently, these observations suggest that **the RAG model outperforms the LLM model** in terms of NLI, given its more favorable distribution of scores across the evaluated range.

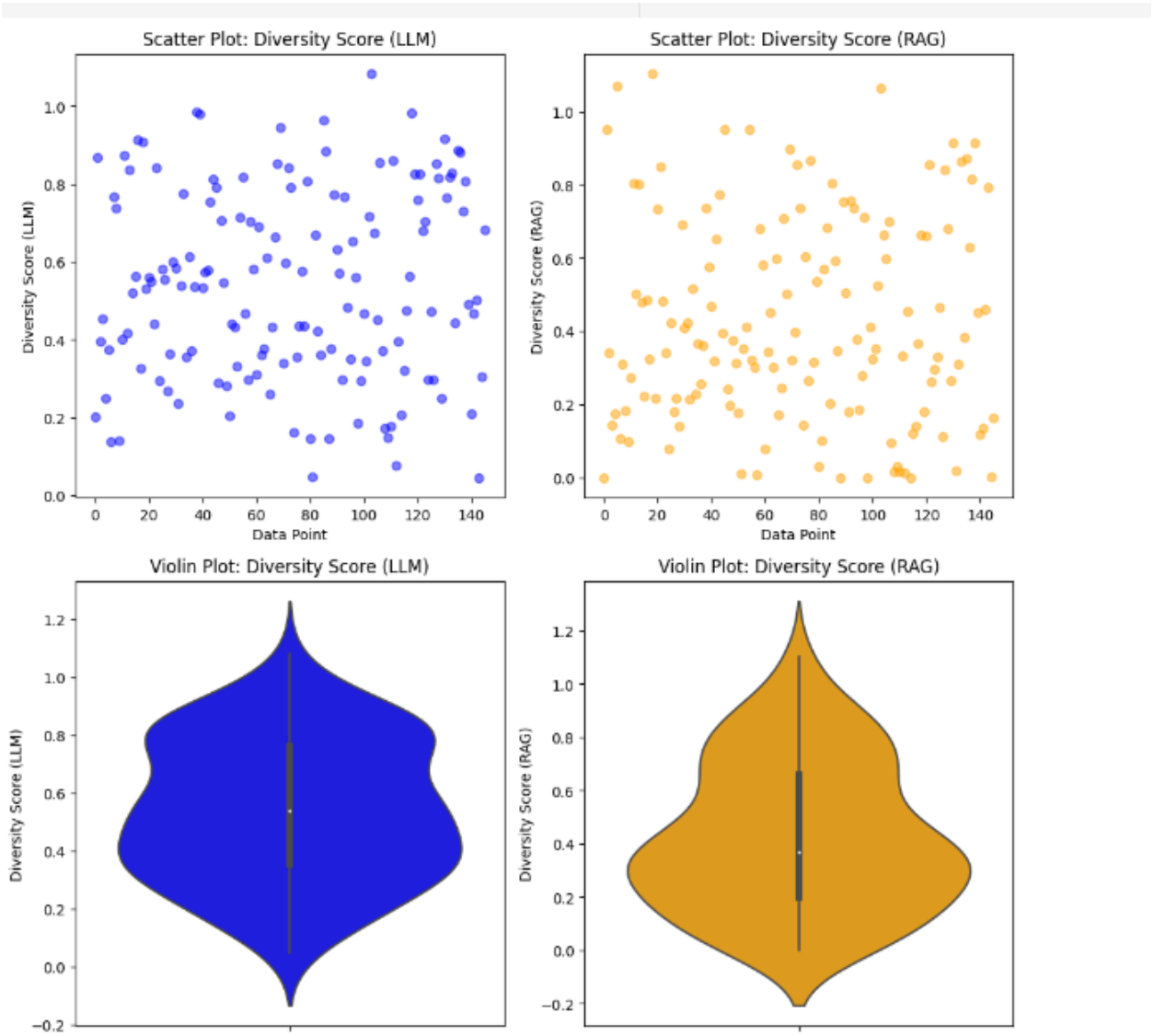
BERTSCORE


Scatter Plot: BERT Score (LLM)
BERT Score (LLM)
Data Point
Scatter Plot: BERT Score (RAG)
BERT Score (RAG)
Data Point
Violin Plot: BERT Score (LLM)
BERT Score (LLM)
Violin Plot: BERT Score (RAG)
BERT Score (RAG)

MIN(NLI, BERT)


Scatter Plot: Min NLI,BERT Score (LLM)
BERT Score (LLM)
Data Point
Scatter Plot: Min NLI,BERT Score (RAG)
BERT Score (RAG)
Data Point
Violin Plot: Min NLI,BERT Score (LLM)
BERT Score (LLM)
Violin Plot: Min NLI,BERT Score (RAG)
BERT Score (RAG)

DIVERSITY


Scatter Plot: Diversity Score (LLM)
Diversity Score (LLM)
Data Point
Scatter Plot: Diversity Score (RAG)
Diversity Score (RAG)
Data Point
Violin Plot: Diversity Score (LLM)
Diversity Score (LLM)
Violin Plot: Diversity Score (RAG)
Diversity Score (RAG)

CONCLUSION

Retrieval-augmented generation has shown superior performance compared to standalone generators across various evaluation metrics. This advancement, however, comes with a set of crucial considerations that impact its effectiveness.

- **Dependency on Augmentation Data Quality:** The quality of augmentation data, which includes the additional information retrieved to enhance generation, plays a pivotal role. The success of retrieval-augmented generation heavily relies on the relevance, accuracy, and completeness of the augmentation data. Poor-quality or irrelevant augmentation data may undermine the overall performance and efficacy of the RAG model.
- **Importance of Augmentation Data Indexing and Organization:** The indexing and organization of augmentation data are equally critical. Efficient retrieval during the augmentation phase depends on well-structured and properly indexed data. A thoughtful organization ensures that the retrieval process is streamlined, allowing the model to access relevant information promptly. Inadequate indexing or disorganized data may lead to suboptimal results.
- **Significance of the RAG Pipeline:** The RAG pipeline, encompassing both the retrieval and generation processes, plays a pivotal role in determining the success of the overall system. The effectiveness of information retrieval during the initial phase and the seamless integration with the generation phase contribute to the model's ability to produce accurate and contextually relevant responses. A well-designed and optimized RAG pipeline is essential for achieving superior results.
- **Impact of Database Quality:** While a high-quality database is a valuable asset, it is essential to recognize that its effectiveness is contingent on the overall strength of the RAG pipeline. A high-quality database coupled with a weak retrieval or generation process may not yield the desired improvements. The synergy between a robust database and an optimized pipeline is crucial for maximizing the benefits of retrieval-augmented generation.
- **Limitations of Evaluation Metrics:** The evaluation metrics employed to assess RAG's performance provide an overall perspective but come with inherent limitations. Metrics may not capture all nuances of the model's capabilities and limitations. Users and developers must interpret results with an understanding of these constraints and consider additional qualitative assessments for a comprehensive evaluation.

In conclusion, retrieval-augmented generation offers enhanced performance over standalone generators, but its success is contingent on critical factors. The quality of the database, efficiency of the RAG pipeline, and the careful selection and curation of augmentation data are paramount. Addressing these considerations ensures that retrieval-augmented generation can fulfill its potential in providing accurate, contextually rich, and reliable responses. As the field continues to evolve, attention to these factors will be crucial for advancing the effectiveness of retrieval-augmented generation systems.

REFERENCES

1. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks : https://arxiv.org/pdf/2005.11401.pdf
2. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models : https://arxiv.org/pdf/2303.08896.pdf
3. Understanding and Mitigating LLM Hallucinations | by Felipe de Pontes Adachi | Oct, 2023
4. QLoRA: Efficient Finetuning of Quantized LLMs : https://arxiv.org/pdf/2305.14314.pdf
5. RAG Tutorial https://colab.research.google.com/github/PouriaRouzrokh/machine-learning/blob/master/Education/Retrieval_Augmented_Generation/RAG_Tutorial.ipynb *by Pouria Rouzrokh, MD, MPH, MHPE; Shahriar Faghani, MD, MPH, MHPE; Felipe Kitamura, MD, PhD; Timothy L. Kline, PhD* :
6. Automatic Hallucination detection with SelfCheckGPT NLI