Measuring effectiveness of RAG

Joudi Alakkad
Ruhr University Bochum
joudi.alakkad@rub.de

Sidra Saied Ali
Ruhr University Bochum
sidra.saiedali@rub.de

Miaoxuan Liang
Ruhr University Bochum
miaoxuan.liang@rub.de

Abstract

This report explores the implementation and evaluation of a RAG system designed to enhance LLMs by integrating external knowledge retrieval with text generation. Our implementation makes use of two LLMs—Llama-3.1-8B and Gemma-2-9B—to generate responses. Evaluation metrics, including BERTScore, ROUGE, BLEU and human evaluation, reveal that RAG significantly improves information relevance and reduces factual errors compared to standalone LLMs.

1 Key Information

- This is a student project as part of the course *Programming for modern machine learning*
- Mentor: Bilal Zafar

2 Introduction

As artificial intelligence advances, Large Language Models (LLMs), such as ChatGPT, have become deeply integrated into daily life. LLMs generate responses based on user input, but users should not blindly trust their output, since outdated information or incorrect details (hallucinations) may be included. This happens because traditional LLMs rely only on pre-trained knowledge, which makes it difficult for them to answer questions about recent events or domain specific tasks. We'll discuss two possible solutions that address these limitations:

- **Fine-Tuning** involves retraining an existing LLM on new domain-specific data. Although it improves performance on specialized tasks, this method is time-consuming, costly, and cannot be adapted to real-time updates.
- **Retrieval Augmented Generation (RAG)** connects the LLM to an external database containing up to date or domain-specific information. When a user submits a query, RAG first retrieves the relevant context from the database, then combines the query and context to guide the LLM response. This approach reduces inaccuracies by grounding answers in verified and up-to-date information.

However, it is unclear to what extent RAG improves LLM effectiveness. To evaluate this, we compare two models, Llama-3.1-8B and Gemma-2-9B, under two conditions:

- **Baseline:** Generating answers without external context
- **RAG-enhanced:** Generating answers using retrieved context

We measure performance using metrics such as BERTScore, ROUGE, BLEU and Human Evaluation. Our goal is to determine whether RAG consistently improves answer quality over standalone LLMs.

3 Approach

To be able to evaluate a LLM with RAG, we need to design and implement a RAG pipeline. The process begins by selecting documents to store it in database. We used a dataset that contains

question, reference answer, and context, in order to have ready questions and answers to compare with generated answers. Then, we split the context of the dataset into smaller chunks of size 512, using a Recursive Character Text Splitter. This ensures that the context is divided into coherent segments, making it easier for the model to process and retrieve relevant information. The chunks are then given unique ids to be saved in the database. After that, we create embeddings of the context chunks to store it in a vector database. An embedding model transforms the text into a format that can be easily compared and retrieved based on semantic similarity. This is realized with the `all-mpnet-base-v2` sentence transformer model provided by ChromaDB. These embeddings are then stored in a ChromaDB vector database. For this evaluation, we saved the context of up to 100 entries in the dataset to ensure a manageable scope. ChromaDB is an open-source vector store used for storing and retrieving vector embeddings.

To maintain consistency between retrieval and generation phases, we remove formatting artifacts like bullet points (*, •), numbered lists (1., 2.), or excessive newlines. This ensures that both the retrieved context and the model response follow a standardized format, facilitating the evaluation. Once the context is processed and saved, the system is ready to handle user queries. When a query is submitted, the system searches the vector database for context chunks that semantically match the query. This is done by comparing the embeddings of the query with those of the stored context chunks. The most relevant contexts are then retrieved and provided to the model. We limited the retrieval to be only 5, to avoid long prompts. The retrieved contexts are then encompassed into the prompt, instructing the model to generate answers precisely based on the retrieved context. This ensures that the model's responses are grounded in the provided information. For comparison, we also prompt the model to answer the same questions without the retrieved context. Both sets of answers along with the questions and retrieved contexts are saved in CSV files for easier and faster retrieval during evaluation.

Integrating the Groq API allowed us to easily use two different models "llama-3.1-8b-instant" and "gemma2-9b-it" for performance comparison. This step is crucial for understanding the strengths and weaknesses of each model in a RAG setup and decide the influence of the model on our RAG pipeline.

4 Experiments

With all the prerequisites in place, we can start with the evaluation.

4.1 Data

To train the model and evaluate the RAG, we need a dataset that has question-answer pairs with annotated contexts. We choose the neural-bridge/rag-dataset-1200 from HuggingFace.

4.2 Evaluation method

BERTScore. BERTSCORE takes 3 mandatory parameters: predictions (the answer generated by the model), references (the correct answer from the dataset) and the language (a shortcut indicating the language used). It then calculates a similarity score by comparing each token in the generated response with each token in the reference answer using contextual embeddings. (Zhang* et al. (2020))

ROUGE. ROUGE checks the word and phrase overlaps between the generated and reference answers. However, it does not consider the semantics, just the number of common words. There are different versions of ROUGE which only vary in how many words they compare. (Lin (2004))

- ROUGE-1 counts how many single words (unigrams) from the correct answer appear in the generated response.
- ROUGE-2 counts how many pair of words (bigrams) matches.
- ROUGE-L examines the longest common subsequence.

BLEU. BLEU measures how well the generated response matches the reference answer by evaluating n-gram overlap. Unlike ROUGE, BLEU focuses on precision, meaning it checks how much of the generated response is correct rather than how much of the reference answer is covered.

Human evaluation. We manually compare the generated answers with the reference answer, while considering the provided context. Each response is assigned a correctness percentage based on its accuracy and relevance. While we focus on meaning, this evaluation remains subjective.

4.3 Experimental details

Using Groq API, allows us to access and compare different models with no additional training or fine-tuning needed. We chose models with 8-9 billions parameters. To ensure high-quality retrieval and response generation, we set the temperature to 0.2. We experimented with different temperatures between 0 and 0.9, but we decided for a lower temperature. We left some space for model to be little deterministic because on temperature 0 the model without context generated a lot of "I don't know answers".

While setting up and running our experiments, we encountered several challenges.

- Retrieval Quality: The quality of retrieved context depends on the context quality, the vector database setup (e.g. chunk sizes, number of chunks) and the choice of embeddings model. Furthermore, it depends on the size window, gemma model expects only 8,192 tokens so we couldn't retrieve big chunks to avoid reaching maximum tokens.
- Generation Quality: The quality of generated responses depends on the model parameters (temperature, prompt), the retrieved context and the choice of generation model(s). A well-structured prompt with relevant context helps to highlight the user's intent.
- Latency: Running RAG increases processing time due to retrieval and embedding searches. To mitigate this, we used GPU acceleration (e.g. with Google Colab or Kaggle) to speed up retrieval and generation.

4.4 Results

After running several experiments and analyzing the data, we observed that RAG had a significant impact on model performance. The models' accuracy improved significantly when using RAG compared to answering questions independently.

The most noticeable improvement was seen in human evaluation, with the average gain from no RAG to RAG being just over 50%. The ROUGE metric also rose by approximately 40% on average for each ROUGE score. BLEU scores also showed improvement, though to a lesser extent.

With RAG, we noticed that responses became more precise and contained less extraneous information. Our manual evaluation of responses further confirmed these findings. Answers with RAG achieved an accuracy rate of almost 90 percent, compared to 30-35% percent for baseline responses. When addressing complex topics or historical events, RAG responses consistently included relevant details such as dates, names, and technical specifics, which were often absent in baseline responses.

A comparison of the two models revealed notable differences in how they made use of retrieved context. Both models performed very well with RAG but showed more hallucinations and incorrect responses without RAG. However, we noticed that due to the Llama model's style of responding, which is longer and more detailed, it was able to consistently outperform the Gemma model. These findings suggest that different model architectures may vary in their ability to integrate RAG effectively.
To visualize our findings, we created several comparative diagrams that demonstrate the performance differences between RAG and non-RAG results:
The bar charts for each metric show that both models achieved higher values with RAG enabled, indicating that RAG helped both models generate responses that were semantically closer to the reference answers.

The line graph of BLEU precision scores across different n-gram levels shows some interesting patterns:

- Both models show declining precision as n-gram size increases
- RAG consistently outperforms no-RAG in all n-gram levels

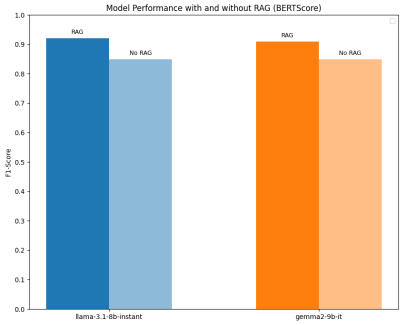

Figure 1: BERTScore

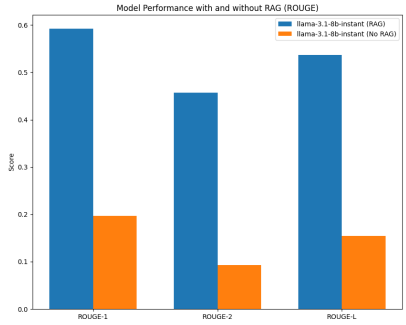

Figure 2: ROUGE

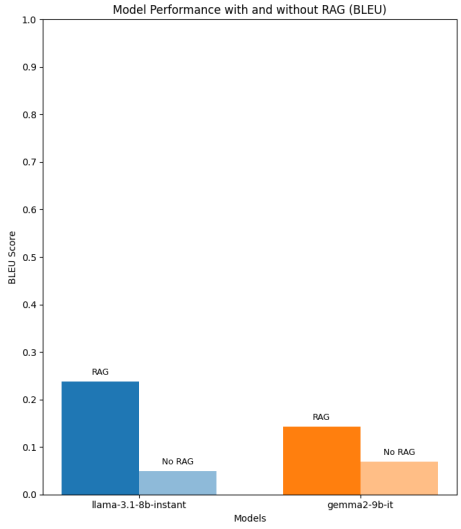

Figure 3: BLEU

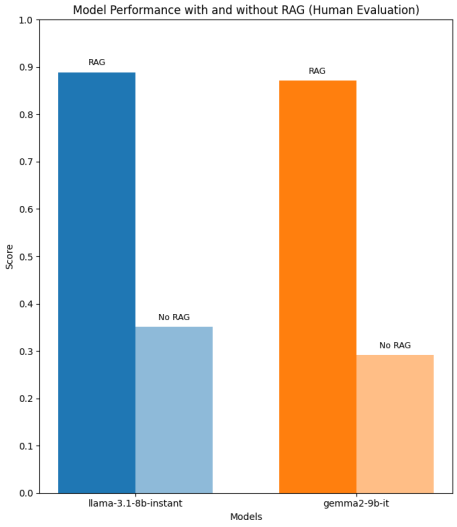

Figure 4: Human Evaluation

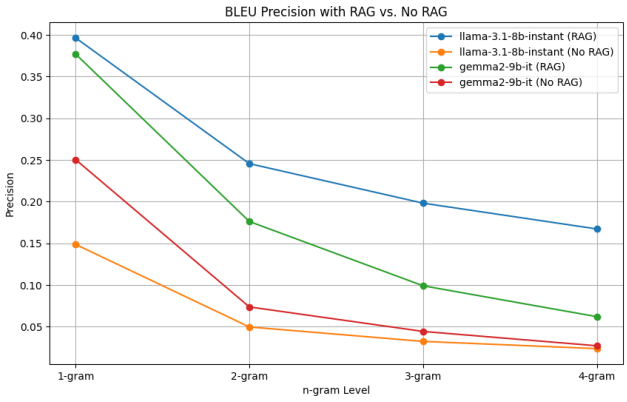

Figure 5: BLEU Precision

- Llama-3.1-8b-instant maintains higher precision compared to gemma2-9b-it, particularly at higher n-gram levels

5 Analysis

The RAG system performs well in situations where retrieved context is highly relevant to the query. In success cases, answers with RAG reduce hallucination and fully align with the reference answers. In failure cases, the model may extract less relevant keywords, which makes the answer to focus on the wrong aspect.

However, the model does not always hallucinate when the context is unclear. When the model is aware that it needs specific knowledge, it will ask the user for more information than giving an made up answer. In rare cases, where the query is highly specific but based on general knowledge, the model might give an answer that matches partly with the reference.

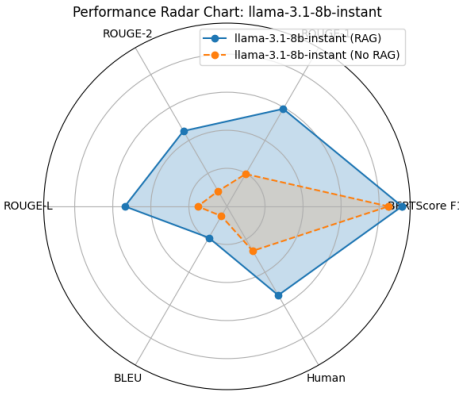

Figure 6: llama-3.1-8b Performance

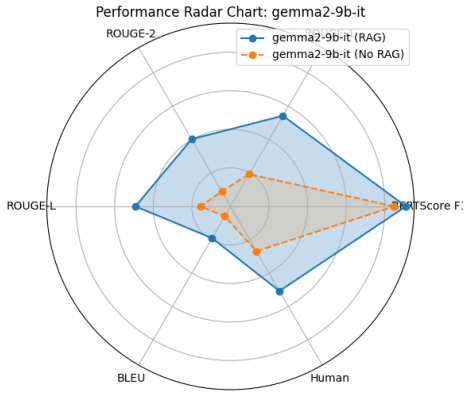

Figure 7: gemma2-9b-it Performance

6 Conclusion

Our findings demonstrated that integrating an external retrieval mechanism significantly enhances response quality by reducing hallucinations and ensuring more fact-based, context-sensitive answers.

As learning outcomes, We got to know how to implement a RAG pipeline, how to deploy vector database and embeddings. Moreover, we payed attention to influence of prompt engineering on the results and how to evaluate LLM including the differences between metrics.

However, our work has several limitations, we were able to document only two models, due to usage limit errors in the API. Additionally, the limits of GPU units available online, we used both Google Colab and Kaggle. Furthermore, we used only 100 entries from the dataset, due the mentioned computational limitations. Finally, the evaluation metrics we employed were not entirely reliable in ensuring that the generated responses fully aligned with user intent.

References

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.