A Comparison of Large Language Models in a Retrieval-Augmented Generation (RAG) System for Machine Repair and Maintenance Prompts

Shane Tikasingh and Patrick Hosein
Department of Computing and Information Technology
The University of the West Indies, St Augustine, Trinidad
shane.tikasingh@gmail.com, patrick.hosein@uwi.edu

***Abstract*—We constructed a Retrieval Augmented Generation (RAG) system for industrial machine manuals to quantify the performance of various open-source Large Language Models (LLMs). Using technical manuals from a flour mill, textual data was extracted using MinerU. The data was processed using Natural Language Processing (NLP), embedded with "Qwen3-Embedding-0.6B" and entered into a PostgreSQL database using LangChain's pgVector API. Our RAG pipeline integrated a tag extractor, three query expansion techniques (Multiquery, Decomposition, Step back) and one scoring algorithm (BM25), a PostgreSQL Database, a Reranker (mxbai-rerank-xsmall-v1), a Prompt template and an LLM. We evaluated the BERTScore performance of five open-source LLMs (Gemma3:1B, Gemma3:4B, Llama3.1:8B, Llama3.2:1B, Llama3.2:3B) in this RAG pipeline. One key finding is that model performance is task-dependent: Llama3.1:8b excelled in precision, while Gemma3:4b excelled in recall. Smaller models can outperform larger models in specific tasks, as demonstrated by Llama3.2:3B superior performance in identifying unanswerable questions. This study provides a framework for selecting cost-effective and high-performing open-source LLMs in industrial applications and demonstrates the effectiveness of smaller parameter models, a critical finding for compute limited businesses.**

***Index Terms*—Retrieval-Augmented Generation (RAG), Open-Source, Large Language Models (LLMs), Natural Language Processing, BERTScore, Self-Hosted.**

I. Introduction

The past decade has seen an increase in the popularity and capability of Artificial Intelligence (AI). This is signified by broken benchmarks, significant investment, and integration into daily life, creating a valuable opportunity for business applications [9], [11]. This is particularly important in industrial settings where technical documents are essential for operations, but traditional retrieval methods such as perusing hard copies or its digitized equivalent remain inefficient, leading to costly downtime. The urgent need for intelligent solutions is exemplified by the explosive growth fore-casted for the RAG market, which is projected to grow from $1.2 billon in 2024 to $11 billion by 2030 as companies seek AI systems capable of providing semantically coherent answers [7].

LLMs have an amazing ability to perform diverse tasks, this ability stems from their parameterized knowledge obtained from training on large corpora datasets. However, LLMs are known to perform poorly when dealing with niche areas such as domain-specific applications [10]. RAG systems improve the output of LLMs by consulting an external knowledge base which contains information outside the model's training data, and then integrating this information with the user query to generate responses. RAG systems allow the extension of LLMs to niche domains [1]. Much of the prominent research on AI has focused on powerful proprietary models hosted on cloud infrastructure [2]. These cloud-hosted systems present key considerations for business, such as concerns about data privacy, the availability of a stable and consistent internet connection, and operational costs. Therefore, a key research niche is created as there is a lack of empirical studies quantifying the performance of smaller open-source LLMs within locally hosted RAG systems designed for specialized industries with limited resources such as those commonly found in the Caribbean region.

This study develops an open source, fully local RAG system for querying industrial machine manuals to address the gap in practical, cost-effective solutions. Guided by three research questions, we first construct the external knowledge base using technical manuals for a flour mill and then the RAG pipeline. We then evaluate and quantify the performance of various open source LLMs (Gemma3:1B, Gemma3:4B, Llama3.1:8B, Llama3.2:1B, Llama3.2:3B) using BERTScore. This allows us to determine the most effective models for this manufacturing domain.

978-1-6654-7808-3/25/$31.00 ©2025 IEEE

II. Literature Review

Recent research on RAG has explored both technical optimization and real-world application, with a particular emphasis on technical documentation and domain-specific support systems. One stream of work has investigated how LLMs can enhance retrieval and generation processes in technical support contexts. For example, [3] examined the role of LLMs in relevance feedback and multi-agent communication patterns, finding that larger LLMs generally outperform smaller ones in mimicking human relevance judgments. Importantly, conversational structures such as reflection and planning led to better performance than simpler interaction patterns, underscoring the value of multi-agent orchestration in RAG pipelines.

Complementing these architectural advances, other research has focused on novel retrieval mechanisms. The study [8] introduced qulm-RAG, a system that implements an inverted question matching approach to improve accuracy and reduce hallucination. In this method, potential questions are generated from document chunks using an LLM, embedded into a high-dimensional space, and quantized to their nearest prototype vector before being indexed. When a user query is submitted, it is matched against this index of questions rather than raw text chunks to retrieve the most semantically relevant context for the answer-generating LLM. Evaluated on a custom, domain-specific dataset from a university website, qulm-RAG demonstrated significant improvements over traditional RAG systems across all measured metrics including faithfulness, answer relevance, and BERTScore, highlighting the effectiveness of both its unique retrieval technique and the use of a prepared domain-specific dataset.

Parallel to this, [5] demonstrated the feasibility of building high-performing, open-source RAG systems for technical manual analysis. Using LLamaIndex and PostgreSQL with PGVector for retrieval alongside llama.cpp backend for generation, the authors compared different combinations of embedding and LLM models. Results indicated that higher-dimensional embedding models performed better, indicating an ability to capture more nuanced information, with the combination of MxBai Colbert (Large), Llama3:8b, and the presence of a reranker yielding the strongest results in terms of factual correctness, clarity, and user satisfaction. This highlights the critical role of embedding dimensionality and retrieval architectures in shaping system performance, especially in technically dense domains.

Beyond technical evaluations, research has also explored RAG deployment in practical, real-world settings. [4] presented domain-specific RAG systems in government, cybersecurity, agriculture, industrial research, and healthcare. User evaluations showed strong ratings for ease of use and accuracy, though transparency and trust varied by domain. Key lessons emerged around infrastructure and design choices: self-hosting improved compliance and speed, clean and stable data pipelines were essential for retrieval quality, and incorporating explicit source references in output improved trustworthiness. The study emphasized that user feedback and dataset design are just as critical as model performance in determining adoption and usability.

Taken together, these studies illustrate a trajectory in RAG research that moves from optimizing model-level components (LLMs, embeddings, conversation strategies) and inventing novel retrieval paradigms toward engineering and evaluating full systems in applied settings. While technical optimizations from multi-agent orchestration and higher-dimensional embeddings to inverted indexing techniques, demonstrate clear gains in retrieval accuracy and response quality, the broader adoption of RAG in real-world domains hinges on considerations of transparency, trust, and infrastructure robustness.

III. Methodology

A. Dataset Description

The dataset, curated from the maintenance and asset manager, consisted of 23 technical machine manuals. Each of these manuals represented a component machine that collectively built a flour mill. The manuals contained a host of information such as general information, procedures, transportation, handling, installation, operation, maintenance, spare parts, and disposal.

B. Extract, Transform and Load (ETL)

- **Extract:** The MinerU extraction tool from [6] was applied to each document with the resultant markdown files placed in a common directory.
- **Transform:** The gathered markdown files were then cleaned by evicting: image and table URLs, table of contents, special/invalid characters ("\n", "\t", email and phone numbers, non-english words), invalid headers and headers with empty fields. The title of each section was then appended to its content. After cleaning, for each file the sections were joined to produce the file corpus. Next, construction of the knowledge base began with the chunking and embedding of the data (belonging to Transformation step), followed by the initiation of the PostgreSQL database and its update with the LangChain API (belonging to Load step). Chunking using Semantic Text Splitter was applied with a token (Tokenizer: `bert-base-uncased`) range of 100 to 120 and an overlap of 20, to segment the respective cleaned text with these results stored. The respective chunks were embedded using "Qwen/Qwen3-Embedding-0.6B" from Hugging Face.
- **Load:** With the data being transformed, it was moved out of the staging area and persisted in the database: A PostgreSQL Database, using pgAdmin4, was registered, a database was created, and the pgvector extension was enabled. Langchain's pgvector API was used to upload precomputed document embeddings along with their corresponding documents and metadata. To optimize storage and retrieval, the column with the embeddings was fixed with a vector length of 1024 and a Hierarchical Navigable Small Worlds (HNSW) index was constructed for the embedding column.

C. RAG Pipeline

The overall workflow of the RAG pipeline is illustrated in Figure 1. The process consists of two main stages: information retrieval and generation. The following sections describe each stage in detail.

1) Information Retrieval System:

a) Tag Extractor: Users are required to submit their query with the specific machine of interest attached to a hashtag (eg, "#bucketelevator"). The hashtag is then captured using regular expressions, and Levenshtein's distance is applied safeguard against errors. The machine of interest is used for metadata filtering when consulting the database.

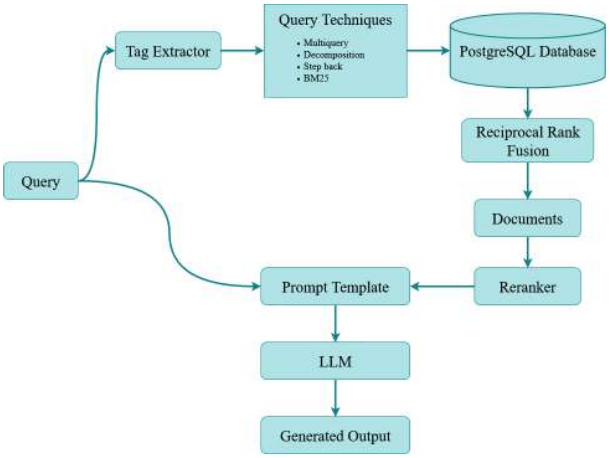

Fig. 1. RAG Pipeline

b) Query Techniques: The query expansion techniques of multi-query, decomposition, and step-back were applied alongside a BM25 scoring algorithm. The query expansion techniques were implemented with LangChain's "MultiQueryRetriever" class, where "gemma3:1b" was employed to reformulate queries with respective instructions. Query embeddings are generated with "Qwen/Qwen3-Embedding-0.6B".

For the query expansion techniques, each multi-retriever generates its own set of queries, which are then used to search the database using Maximal Marginal Relevance (MMR). The results of all queries are combined by taking the union of the returned documents.

For the scoring algorithm, all documents that passed the metadata filters are retrieved from the database and LangChain's BM25 is applied with the top 60 results returned.

c) Document Ranking: All query techniques above are wrapped in LangChain's "EnsembleRetriever" implementation, which executes all retrievers, collects the documents associated with each retriever, and performs reciprocal rank fusion to produce the top 60 final output documents. These sets of documents are then passed to the "`mixedbread-ai/mxbai-rerank-xsmall-v1`" reranker model where the document relevance to the query is computed and the top 20 results are returned.

2) Generative System: The user query and top relevant documents are passed to the prompt template. The prompt template constructed with LangChain's "ChatPromptTemplate" included instructions for the LLM, the role it plays, important rules to follow, and language-style constraints. Notably, the LLM (implemented with Ollama) is instructed to work step by step (chain-of-thoughts prompting) to arrive at the final answer and to respond in a JSON format that separates the reasoning from the final answer. The filled prompt template was input into the LLM model which was set to a temperature of 0.5. The final output is then generated by the LLM.

3) Evaluation: Evaluating the performance of the RAG model requires a systematic comparison between reference answers and candidate answers. This section outlines the methodology for generating reference question-answer pairs, producing candidate answers through the RAG pipeline, and assessing their reference-candidate answer similarity using BERTScore.

BERTScore is a text evaluation metric that measures the similarity score between tokens of the candidate and reference sentence using contextual embeddings. BERTScore outputs, as described in [12] contains three values, precision, recall, and F1 as follows:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j \tag{1}$$

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j \tag{2}$$

$$F_{\text{BERT}} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{3}$$

The overall process for generating reference questions and answers is illustrated in Figure 2. It consists of two main stages: (i) corpus partitioning and (ii) question–answer generation.

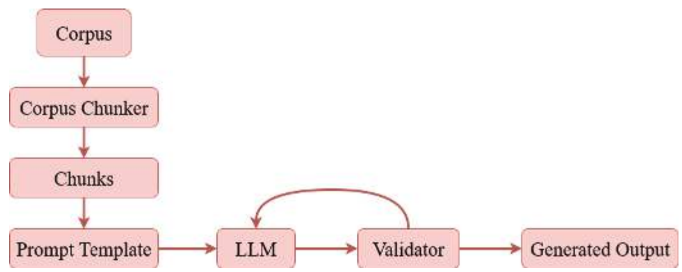

Fig. 2. Reference Question-Answer Generation Process

To generate reference question-answer pairs we apply corpus partitioning. Each file corpus was partitioned into nearly equal chunks based on their token count using a function built with the `bert-base-uncased` tokenizer and semantic text splitter. Four questions were generated for each corpus partition: factual, inference, multihop reasoning, and unanswerable. A structured prompt template instructed the `llama3.1:8b` model (temperature of 0.5) to generate these questions while adhering to specific requirements such as JSON formatting, language style, and structural constraints. In practice, the model occasionally produced erroneous output, such as deviations from the JSON schema or inclusion of explanatory text despite explicit instructions. To address this, a validator function was implemented to parse the model's output into JSON, verify the presence of all question types, ensure the presence of question-answer pairs, and confirm that these question-answer pairs contained content. When the output failed these checks, the same corpus partition was resubmitted to the model until a valid response was obtained. All validated outputs were aggregated into a reference JSON object.

The process of generating candidate answers from the reference questions is illustrated in Figure 3. This stage ensures that all questions are passed through the RAG pipeline and that the outputs conform to the expected JSON structure.

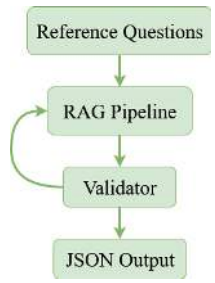

Fig. 3. Candidate Answer Generation Process

The reference questions were extracted from the reference JSON object and passed into the RAG pipeline. In practice, the model occasionally produced errors such as repetition loop, invalid JSON or explanatory text. To address these issues, the model was initialized with a predicted token count of 300. A validator function was then applied to the output which parsed them into JSON, checked for the presence of ”reasoning” and ”final answer” keys and ensured they contained content. If the specifications were not met, the RAG pipeline was re-prompted with the same query. After every three unsuccessful re-prompts the predicted token count was increased by 50 tokens. All valid outputs were collected in a candidate JSON object, which also contained the reference question they answered.

To assess performance across different language models, the candidate question-answer generation process was repeated using the following models : gemma3:1b, gemma3:4b, llama3.1:8b, llama3.2:1b, llama3.2:3b. Each model's output was stored in their respective candidate JSON object.

The performance of each model in the RAG system was quantified by comparing the candidate answers and the reference answers (ground truth) using BERTScore. To arrive at the final precision, recall and F1 scores, a structured aggregation process was employed. First, the candidate answers from each model were aligned with their corresponding reference question-answer pairs by merging the candidate and reference JSON objects. BERTScore was then computed for each individual pair, generating a set of precision, recall, and F1 values. Subsequently, a three-tier averaging procedure was implemented:

1) File-Level Averages (per question type): For each technical manual (file), the scores of all its constituent corpus partitions were averaged, yielding a single representative score per question type for that file.
2) Model-Level Averages (per question type): These file-level averages were then aggregated by averaging across all 23 files, producing the overall model-level precision, recall and F1 scores for each question type.
3) Model-Level Overall Average: A final average was calculated across all four question types to produce a single overall precision, recall, and F1 score for each model, representing its general performance irrespective of question type.

IV. Results

The built RAG system was evaluated using BERTScore (precision, recall, and F1) across five different models (gemma3: 1b, gemma3: 4b, llama3.1: 8b, llama3.2: 1b and llama3.2: 3b) and four different question types (factual, inference, multihop reasoning, and unanswerable). The results are summarized in Figure 4 which presents model-level averages. The key results for each metric in Figure 4, broken down by question type, are summarized as follows:

- **Precision scores (Figure 4 (top panel):** For factual, inference and multihop reasoning questions the llama3.1:8b model achieved the highest precision score while for unanswerable questions the smaller llama3.2:3b model performed better. Across all question types, the llama3.2:1b model consistently achieved the lowest precision score.
- **Recall scores (Figure 4 (middle panel):** For factual, inference and multihop reasoning questions, the gemma3:4b model achieved the highest recall score while for unanswerable questions the smaller llama3.2:3b model performed better. The llama3.2:1b model recorded the lowest recall score in inference, multihop reasoning and unanswerable questions, while the gemma3:1b model had the lowest recall in factual questions.
- **F1 scores (Figure 4 (bottom panel):** For factual questions, the llama3.1:8b model achieved the highest F1 score. For inference and multihop reasoning questions, the best performance came from gemma3:4b, while for unanswerable questions, the llama3.2:3b model attained the top score. The llama3.2:1b model had the lowest F1 score for all question types.

To enable direct comparison of overall performance and determine the overall best LLM model for the RAG system, we averaged precision, recall and F1 scores across all question types, producing a single set of metrics per model (model-level overall average). The main results for each metric are as follows:

- **Precision Score:** The llama3.1:8b model achieved the highest overall precision score. It performed slightly better than the llama3.2:3b model (Δ = 0.02) and modestly better than the gemma3:4b model (Δ = 0.04).
- **Recall Score:**: The gemma3:4b model achieved the highest overall recall score. It performs modestly better than the llama3.1:8b model (Δ = 0.05) and noticeably better than the llama3.2:3b model (Δ= 0.06).
- **F1 Score:** The llama3.1:8b and the smaller model gemma3:4b recorded the same F1 score and were negligibly better than the llama3.2:3b (Δ = 0.01).

V. Discussion

This study constructed a RAG system for industrial technical manuals and methodically evaluated the performance of five open-source LLMs across four different question types.

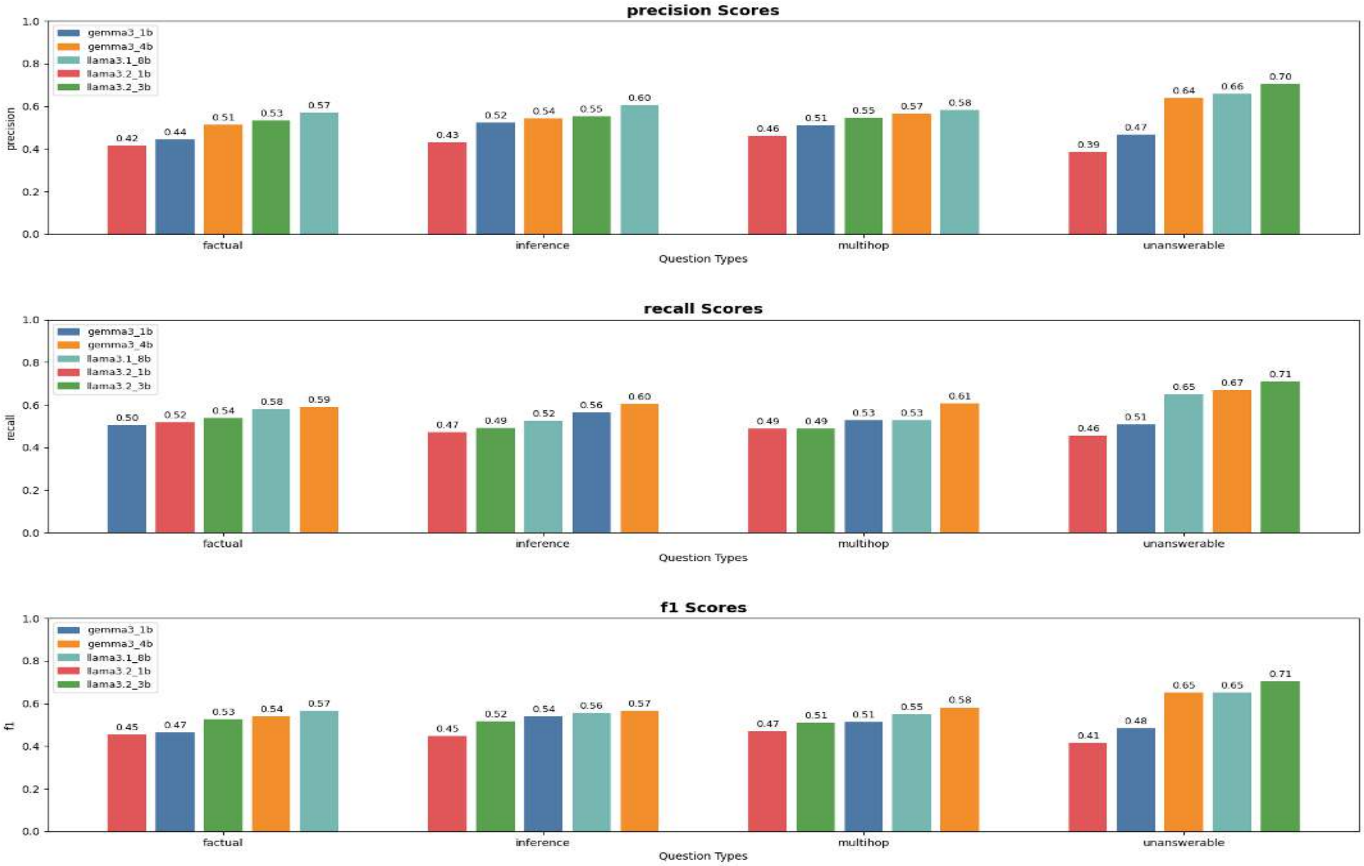

Fig. 4. Model Level Average: Precision, Recall and F1 Scores for Each Model against Question Type

The results revealed an intricate performance landscape in which no single model dominated across all metrics and question categories, highlighting important compromises in model selection for applied RAG systems.

Our key finding is that larger parameter models generally but not universally gave superior performance. The 8 billion parameter Llama3.1:8b model achieved the highest overall precision (0.60) and shared the highest overall F1 score (0.58). This performance dramatically outperformed ($\Delta > 0.20$) the traditional RAG baseline reported by [8] and supports the findings of prior work that larger LLMs generally excel in tasks demanding precise and faithful generation of content [3] [8]. However, the 4 billion parameter Gemma3:4b model attained the highest overall recall (0.62), also signifying a major improvement ($\Delta > 0.20$) over [8]'s traditional baseline. This indicates an exceptional ability to generate output most representative of the reference answer which is suggestive of a potential trade-off. Larger models are better able to generate more precise answers, while mid-sized models are better able to generate answers with more coverage of the ground truth. The fact that these two models achieved the same overall F1 score (0.58) which also substantially exceeds the traditional baseline of [8], illustrates this balance. The choice between the respective models will depend upon the specific need for accuracy of generated output (precision) or coverage of information (recall).

However, the most intriguing result challenges the primitive "bigger is better" narrative. The smaller Llama3.2:3b model outperformed all others, including the larger Llama3.1:8b, on unanswerable questions. This is an important capability of technical Question-Answer systems where confident hallucination of erroneous answers could have the potential to be far deadlier than correctly identifying disparity in knowledge. This finding suggests that the ability to identify limits in knowledge may be a specialized skill not solely dependent on model size, potentially relating to more targeted training or reinforcement learning from human feedback (RLHF) that emphasizes response caution.

These results align with and extend the current literature. The strong performance of our models, which consistently and substantially surpassed the traditional RAG benchmarks in [8], confirming the findings of studies which highlight the role of LLMs in relevance feedback and complex reasoning [3]. A detailed comparison with the more advanced QulM-RAG model from [8] reveals an interesting performance landscape. Our system demonstrated a modest advantage (Δ = 0.05) in precision on their traditional dataset, yet was slightly less performant (Δ = 0.03) on their custom dataset. For recall and F1 QulM-RAG performed negligible better for their traditional dataset (Δ = 0.01 for both), but held a notable advantage (Δ =

0.09) on their custom dataset. This illustrates that a highly performant system can be built with architectural innovations and without proprietary models. Ultimately, our results showcase that the optimal model is not a one-size-fits-all proposition but is conditional on the specific metric, question type and domain specificity of the targeted application.

Our findings also identify with the practical lessons from real-world RAG deployment [4]. The varying strengths of different models imply that for a production system, complex multiagent approaches or convoluted routing mechanisms can be applied using models like Gemma3:4b for high-recall generations and LLama3.1:8b for precise generations could achieve better results than any single model. Furthermore, the strong performance of Llama3.2:3b on unanswerable questions makes it a compelling candidate as a "safety check" agent within a multi-agent framework, a strategy suggested by [3].

VI. Limitations and Future Work

This study, while comprehensive, has limitations that present opportunities for future research:

- This study was carried out on a private dataset of technical machine manuals from a flour mill. The generalizability of our findings to other technical domains, such as automotive, petrochemical, or administration, remains to be tested. Future work should validate the RAG architecture on a wider and more diverse set of technical documents.
- Our retrieval pipeline, while sophisticated, relied on a single embedding model ("Qwen/Qwen3-Embedding-0.6B"). The performance of the entire system is highly dependent on the quality of these embeddings. Comparative analysis with different open-source embedding models could further optimize the retrieval branch.
- The evaluation used in this study was BERTScore. While this provides an objective measure of semantic similarity, it may not fully capture critical aspects of technical responses such as factual correctness, clarity and conciseness. Human evaluation with domain experts would provide a holistic assessment of the system's practical application.
- This project was conducted on a Lenovo LOQ which constituted 32GB DDR5 ram and a NVIDIA RTX 4050 which has 6GB vram. The employment of more powerful GPU's would enable the use of larger LLMs and bigger embedding models. This would enable the evaluation of the performance landscape as model size grows to determine if the observed trends hold as model size increases significantly.
- Computational power is a constraint present in Caribbean businesses. Future work can explore model quantization, alternative model serving backends and other optimizations to enhance performance, making high-quality RAG systems more accessible.

VII. Conclusion

In conclusion, we successfully constructed an open-source RAG system for industrial technical manuals and quantified the performance of five LLMs. We found that:

- Model performance is task-dependent: Larger models (Llama3.1:8b) performed well in precision, while medium sized models (Gemma3:4b) achieved a higher recall.
- Smaller models can have superior performance in specific tasks: The Llama3.2:3b model was more effective in identifying unanswerable questions, a key safety feature.
- The best model is a complex choice: For overall balanced performance, both Llama3.1:8b and Gemma3:4b are good choices, but optimal selection depends on the main metric of interest for a specific application.

  This work demonstrates that effective RAG systems for technical domains can be built with open-source models but careful model selection based on outcome constraints is important. Future work will focus on integrating these findings into a multi-agent orchestration framework and validate system performance through human evaluation with industry experts.

References

[1] Amazon. What is rag? - retrieval-augmented generation explained - aws, 2024.

[2] R. Bommasani et al. On the opportunities and risks of foundation models, 2022.

[3] Mats Finsås and Joachim Maksim. Optimizing rag systems for technical support with llm-based relevance feedback and multi-agent patterns. Master's thesis, NTNU, 2024.

[4] Md Toufique Hasan, Muhammad Waseem, Kai-Kristian Kemell, Ayman Asad Khan, Mika Saari, and Pekka Abrahamsson. Engineering rag systems for real-world applications: Design, development, and evaluation, 2025.

[5] Kunal Menon. Utilizing open-source ai to navigate and interpret technical documents: leveraging rag models for enhanced analysis and solutions in product documentation. *VAASAN AMMATTIKORKEAKOULU UNIVERSITY OF APPLIED SCIENCES*, 2024.

[6] opendatalab. Github - opendatalab/mineru: A high-quality tool for convert pdf to markdown and json, 2024.

[7] Gramd View Research. Retrieval augmented generation market size report, 2030, 2023.

[8] Binita Saha, Utsha Saha, and Muhammad Zubair Malik. Quim-rag: Advancing retrieval-augmented generation with inverted question matching for enhanced qa performance. *IEEE Access*, 12:185401–185410, 2024.

[9] Andrea Sestino and Andrea De Mauro. Leveraging artificial intelligence in business: Implications, applications and methods. *Technology Analysis & Strategic Management*, 34(1):16–29, 2022.

[10] Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. Fine tuning vs. retrieval augmented generation for less popular knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2024, page 12–22. ACM, December 2024.

[11] Stanford University. The 2025 ai index report, 2025.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.