Journal of Information Systems Engineering and Management
2025, 10(27s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Leveraging The Performance of Large Language Models in Systems Engineering Applications

Shaul E. Niv

Faculty of Aerospace Engineering, Technion Israel Institute of Technology, Haifa, Israel
seliahou@technion.ac.il

ARTICLE INFO

Received: 15 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

The integration of Artificial Intelligence in systems engineering marks a significant evolutionary step in managing complex projects. Artificial Intelligence can optimize system designs and processes, leading to cost savings and efficiency improvements. One of the most groundbreaking advancements is the rise of Foundation Models and the Large Language Models, a recent innovation in Artificial Intelligence. These models represent a new approach where a single model is trained on a massive, diverse corpus of data to handle multiple tasks. This document investigates the use of Large Language Models within systems engineering, the study employs a Retrieval-Augmented Generation framework to address the complexities of specialized engineering tasks effectively. We outline a structured approach to enhance retrieval effectiveness and Leveraging large language model. This approach enables us to reduce the error rate in systems engineering prediction of different Large Language Models by 33.33%. Our innovative setup paves the way for the future Systems Engineering tool uses unique enhanced data corpus, laying on efficient and controlled real time expert knowledge sharing.

Keywords: Foundation Models, Large Language Models, Retrieval-Augmented Generation, Systems Engineering.

INTRODUCTION

The integration of Artificial Intelligence (AI) in systems engineering marks a significant evolutionary step in managing complex projects. AI can optimize system designs and processes, leading to cost savings and efficiency improvements. A unique feature of foundation models is their adaptability. These models can perform a wide range of disparate tasks with a high degree of accuracy based on input prompts. One application of AI that allows machines to extract knowledge from data and learn from it autonomously is Machine learning (ML). ML has traditionally emerged through supervised, unsupervised, and reinforcement learning approaches. A foundation model is a general-purpose AI model that is trained unsupervised, usually on large amounts of text or data sourced from the internet or other big data repositories. A Large Language Model (LLM) is derived from a foundation model by undergoing additional training and fine-tuning. The foundation model is a general-purpose AI model trained on vast amounts of diverse text data. To create an LLM, this foundation model is further trained on specific datasets, often focusing on conversational data, to enhance its ability to generate human-like text and maintain context in dialogues. This additional training allows the LLM to specialize in tasks such as chatbots and virtual assistants.

Leveraging foundation models and LLM technology in multi-domain systems engineering can significantly accelerate the systems engineering processes by optimizing decisions [1]. It is acknowledged that focused interdisciplinary exposure and thinking in any domain contribute to better outcomes. These advancements also empower work teams to streamline communication, enhance collaboration, and expedite problem-solving.

AI and ML encompass a variety of techniques and methodologies designed to enable machines to mimic cognitive functions associated with human intelligence, such as learning, reasoning, and problem-solving. AI and ML bring powerful benefits to deal with tasks of all shapes and sizes, with new possibilities constantly emerging. In particular, as the amount of data grows in size and complexity, automated and intelligent systems are becoming vital to automate tasks, unlock value, and generate actionable insights to achieve better outcomes [2][3].

The field of AI benchmarks has seen significant growth, starting with simpler, foundational tasks and expanding to more complex, domain-specific challenges. Early benchmarks concentrated on basic tasks such as word relationships and semantic similarity, but over time, the scope has broadened to encompass more advanced subjects like medicine, physics, biology, computer science, and engineering. This progression reflects an increasing demand for models capable of handling diverse, complex topics.

Evaluation of LLMs within the field of systems engineering is almost nonexistent, with very few reports available. Our investigation concentrates on the use of LLMs within systems engineering process, employing a Retrieval-Augmented Generation (RAG) framework to address the complexities of specialized engineering tasks. We outline a structured approach to build a specific domain RAG dataset, applying techniques such as text splitting, cleaning, and relevance checking to enhance retrieval effectiveness. Key sections cover the construction and optimization of the RAG system, followed by an evaluation using the LM-Harness framework to assess model accuracy and confidence on systems engineering benchmarks. Initial results provide insights into the model's capabilities and limitations, while future work will expand the dataset, improve relevance filtering, and test larger models to further understand LLMs' applicability in systems engineering contexts. In our proposed RAG framework, the LLM's queries are enhanced with pertinent dataset segments to provide additional context to the foundational model. The model's predictions are subsequently evaluated using metrics such as accuracy and confidence scores to assess performance on specialized engineering tasks.

This template, modified in MS Word 2021 and saved as a "Word Document" for the PC, provides authors with most of the formatting specifications needed for preparing electronic versions of their papers. The Manuscript length should contain 5 pages at least. In general, a full manuscript consists of **"Introduction", "Objectives", "Methods", "Results", "Discussion" and "Conclusions"**. English is treated as the only written language throughout the text. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided.

BACKGROUND

The accuracy of outputs from generative artificial intelligence (AI) applications has been a subject of recent investigation, with several attempts made to evaluate the performance of these models across various fields of engineering. In the context of fluid motion phenomena, a comprehensive study conducted by Stanford University [4] examined the performance of several well-known AI applications, including Midjourney, Dall-E, Runway ML, Microsoft Designer, Gemini, Meta AI, and Leonardo AI. These applications, developed by leading companies such as Google, OpenAI, Meta, and Microsoft, were evaluated based on their ability to generate images and videos from text prompts describing common fluid mechanics phenomena, such as "Von Karman vortex street," "flow past an airfoil," "Kelvin-Helmholtz instability," and "shock waves on a sharp-nosed supersonic body." The study revealed significant shortcomings in the training of these AI models concerning fluid dynamics imagery. The generated outputs were often found to be misleading when compared to real images from laboratory experiments and numerical simulations. This discrepancy highlights a critical gap in the current capabilities of generative AI models in accurately representing complex fluid motion phenomena.In order to compare and evaluate LLMs performance, an objective evaluation test or benchmark is required. This test will also be used to examine the improvement of the model's performance through tools and infrastructure added for the purpose of improving performance for the specific purpose, which in our case is systems engineering. Recently, an attempt to introduce a benchmark in systems engineering was presented in a report published by the Acquisition Research Program at the Naval Postgraduate School [5]. In this work the SysEngBench Framework was introduced. This benchmark included a simple multiple-choice question (over a thousand) covers an introduction to systems engineering. This benchmark selects 10 different topics to reflect the core processes of systems engineering. The main system engineering areas includes Requirements, System Architecture and Design, Model-Based Systems Engineering, Cost Modeling, System Capability/Suitability Engineering, Safety Engineering, Human Factors Engineering, System Integration and Development, System Verification and Validation and Risk Management. All this questions and answers are described in detail and divided into categories at HaggingFace SysEngBench [6]. As part of this work, the performance of the Orca-2-7B the Mistral-7B the Llama-2-7B and the Gemma 2-2B models was tested by examining the percentage of correct answers compared to the number of questions asked. According to this method, the performance of the four models was rated.

METODOLOGY

The training data for major LLM's isn't specific to any one and often doesn't includes the user resources. If your content isn't in the training data it can only ever be approximated in the outputs. There are several known methods that can be used to integrate the user data or the user information and to shape the LLM's outputs. The known and accepted methods in use today are:

1. Prompt Engineering
2. RAG systems
3. Fine-tuning
4. Rebuilding foundational models.

Among these possibilities, considering that the specific dataset is small comparing to the large dataset on which the model is trained, it seems that the use of RAG infrastructure is the most practical. The method does not require intervention in the structure of the model, which is usually not possible and require large datasets compared to those on which the model is trained. The RAG infrastructure gives significant weight to the user's specific dataset in the LLM query.

In general RAG is an extension of Large Language Models (LLMs), RAG systems are specialized tools designed for LLMs to facilitate the process of gathering and using information. The user resources saved in a database (vector DB) The relevant data is retrieved and this information is used to augment the generation of the answer. the content should be represented in numerically as embeddings. Embeddings are a way of turning words or sections of documents into a numerical code (list of numbers). These numerical codes capture the semantics of what the text describes: the relationships between the codes reflect the relationships between the words they represent. Embeddings come from the statistical structure of language: words that hang out together or to describe things in the world that often occur together. Embeddings provide a computer-friendly representation of language that preserves much of the meaning. This allows the computer algorithms to work with language: for example, finding which chunk of a document has the closest embedding to a query prompt gives the chunk with the most relevant information. As part of the research we carried out, we investigated four LLM using different sets of about 100 questions from the SysEngBench Framework described in the previous chapter. We evaluated the models performance in terms of the various indices and then faced the creation of a unique RAG infrastructure for all its components. The performers were measured again and compared to the LLM performers without the use of the RAG infrastructure.

STEP-BY-STEP WORKFLOW

Workflow Diagram

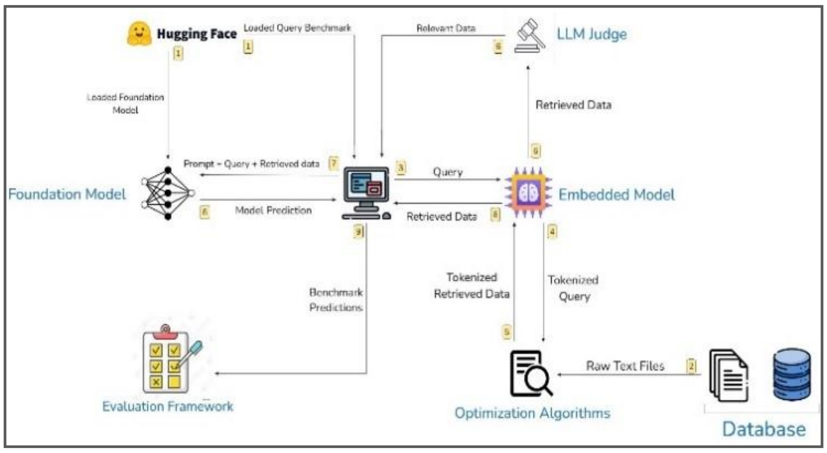


Figure 1. Workflow Diagram. This diagram outlines the RAG system's process, starting with

loading the foundation model and leading through data retrieval, prediction, and evaluation stages.

The process of using RAG which was developed and tested by us is described below using 9 main steps:

Step 1. Loading the Foundation Model. The system begins by loading a foundation model from Hugging Face.

A benchmark dataset is also loaded, providing queries to test the performance of the models with and without the use of the RAG framework.

Step 2. Database Integration. A domain-specific database of raw text files is accessed. These documents form the corpus for retrieval.

The basic corpus utilized for retrieval consists of the following assortment of domain-specific documents:

- Wikipedia entree: Systems engineering [12].
- USA Department of Defence: Systems Engineering Guidebook [13].
- MITRE: Systems Engineering Guide [14].
- DAU: Systems Engineering Fundamentals [15].
- INCOSE: SE Handbook, Version 2a [16]
- Systems Engineering Body of Knowledge (SEBoK) [17].

Compared to typical industry datasets, which are significantly larger, our corpus is relatively small and narrowly focused. Achieving results with this dataset demonstrates the retrieval system's capability to work effectively with concise, targeted information. However, the limited size of the corpus does present challenges, such as reduced diversity and coverage, which can lead to gaps in information. Future work could address these limitations by incorporating additional domain-specific documents, crowd-sourced annotations, and other sources to expand and enrich the dataset. The raw database is preprocessed using: Chunking algorithms to divide large documents into manageable segments. Relevance filtering is used to remove redundant or low-value chunks.

Step 3. Query Input. Queries are selected from the benchmark dataset and passed into the system for processing.

Step 4. Vectorization. Both the queries and the preprocessed database are vectorized using a semantic embedding model with the database saved as a ChromaDB's vector store.

Step 5. Document Retrieval. The system performs a similarity search between the query embedding and the database embeddings to identify the top-k most similar documents. Retrieved documents are augmented with Framing and Chain-of-Thought Reasoning instructions to enhance their relevance and guide the model's structured reasoning.

Step 6. Relevance Judging (Optional). The top-k retrieved documents are evaluated for relevance:

Without Judge - The retrieved documents are returned directly to the main program for use in the next step.

With Judge - The top-k documents are passed to an LLM Judge, which determines their relevance to the query. Only the judged relevant documents are returned to the main program.

Step 7. Prediction. The query, retrieved data, and prompt instructions are provided to the foundation model for generating predictions.

Step 8. Output. The foundation model returns its prediction based on the query and retrieved documents.

Step 9. Benchmark Evaluation. The system evaluates the model's predictions in comparison to the benchmark dataset answers in order to estimate the accuracy and the system's overall performance.

HILIGHTS AND DETAILS OF THE RAG PROCESS STEPS

A. Human prompt vs. system

A Human Prompt is the input directly provided by a user to the system, such as a question, query, or instruction describing the task to be performed. It is static, manually crafted, and serves as the starting point for the system's processing. In contrast, a System Prompt is an automatically generated input constructed by the system "behind the scenes." It combines the human prompt with additional context, such as retrieved documents or predefined instructions, to optimize the input for the model. While the human prompt defines the problem, the system prompt refines and structures it to enhance the model's ability to generate accurate and relevant responses.

We will now briefly describe the method by which this works in our framework: The Human Prompt in our framework is the query provided by a user or in our case the System Engineering benchmark. It defines the task or question for the system to address, For example:

- "What are the key principles of Systems Engineering... ?"
- "How does risk management contribute to Systems Engineering processes... ?"

These queries are simple and contain no additional context or instructions - they initiate the retrieval and reasoning process. The System Prompt is dynamically generated by our framework based on the human prompt. It includes:

The Human Prompt: The core query or task.

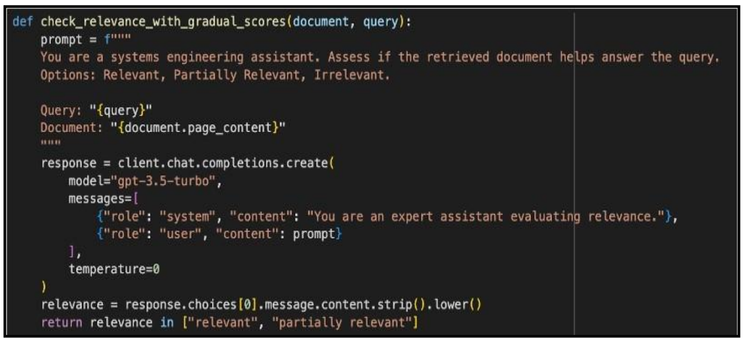
Retrieved Documents: Relevant text retrieved from the corpus using semantic similarity and a LLM as a judge.

Predefined Instructions: Additions like Framing or Chain-of-Thought (CoT) Reasoning to guide the model's reasoning process.

B. Retrieving DOCS Filtering

Various strategies are employed to improve the effectiveness of the RAG system in managing noise. These strategies focus on controlling data retrieval and applying filtering techniques to ensure that only relevant information is processed. We employed Python Script for Text Relevance Assessment. This script utilizes various regular expressions and logical conditions to determine the irrelevance of text based on content and formatting criteria. The system segments the dataset into chunks, which are then stored in a vector database for efficient retrieval. By adjusting the number of chunks retrieved, it is possible to balance knowledge diversity with noise reduction. The Python code initializes a retriever to find the top-4 most similar documents to a given query, showcasing the use of similarity-based search parameters.

An LLM is used to filter out irrelevant chunks from the retrieval set. This procedure helps to reduce noise and ensures that the data being processed is of the high quality, which aims to improves the accuracy and reliability of the system's outputs. We used the Python Relevance Assessment Function. This Python function checks the relevance of a document to a query using a system-engineering assistant role within an AI model, determining if the document is relevant or partially relevant based on the content interaction.



```python
def check_relevance_with_gradual_scores(document, query):
    prompt = f"""
    You are a systems engineering assistant. Assess if the retrieved document helps answer the query.
    Options: Relevant, Partially Relevant, Irrelevant.

    Query: "{query}"
    Document: "{document.page_content}"
    """
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are an expert assistant evaluating relevance."},
            {"role": "user", "content": prompt}
        ],
        temperature=0
    )
    relevance = response.choices[0].message.content.strip().lower()
    return relevance in ["relevant", "partially relevant"]
```

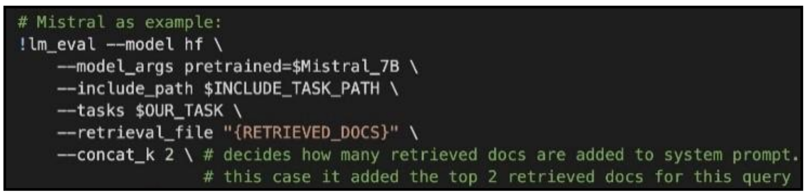
Figure 2. Relevance Assessment Function.

Example of a query from retrieved docs json file.

Question: “What methodologies are commonly used in the risk identification phase of systems engineering projects?” relevant retrieved text: “Imagine you are a Systems Engineering expert. Based on the retrieved information, evaluate the following carefully. Analyze each piece of information step-by-step to answer the question: identified risks within the systems engineering process. The risk management process requires the involvement of several disciplines and encompasses empirical, quantitative, and normative judgmental aspects of decision-making. Furthermore, risk assessment and management should be integrated and incorporated within the broader holistic approach so technology management can help align the risk management requirements to the overall systems engineering requirements. Thus, the inclusion of a well-defined risk management plan that deals with the analysis of risks, within the systems engineering master plan, is vital for the long-term and sustained success of any system (Blanchard and Fabrycky [9]). Noise retrieved text: “Based on the retrieved information, evaluate the following carefully. Analyze each piece of information step-by-step to answer the question: Hillson [10], Olsson [11], and Chapman and Ward [8] provide highly cited introductions. Additional information on risk management for systems engineering projects can be found in the Risk Management article in Part 3: Systems Engineering and Management.”

This example of retrieved text, highlights how different models can respond differently to the same system parameters. How different models may respond differently to the same systemic parameters. The first chunk provides actionable insights into risk management methodologies, while the second references external sources without detailing specific methods. While the selection process prioritizes relevance, some models may handle additional length and minor noise better than others, emphasizing the need to tailor the RAG system to the specific model’s characteristics.

The evaluation system also allows control over the number of chunks retrieved per query, enabling a balance between retrieving comprehensive information and noise reduction, based on the specific needs of the query. This command line script showcases how to evaluate a language model using specific model arguments, tasks, and a retrieval component to include the top two documents relevant to a query in the system prompt.



```
# Mistral as example:
!lm_eval --model hf \
    --model_args pretrained=$Mistral_7B \
    --include_path $INCLUDE_TASK_PATH \
    --tasks $OUR_TASK \
    --retrieval_file "{RETRIEVED_DOCS}" \
    --concat_k 2 \ # decides how many retrieved docs are added to system prompt.
                   # this case it added the top 2 retrieved docs for this query
```

Figure 3. LM Evaluation Command.

Although the system works within a specific domain and relies on a small dataset, noise remains a challenge. The effectiveness of noise management strategies can vary depending on the model’s architecture and the task at hand, highlighting the need for tailored approaches to optimize performance. Framing and Chain-of-Thought Reasoning: enhancing document utilization in RAG systems. Our RAG framework Introduces two techniques, Framing and CoT Reasoning, as will be described in detail below. These two techniques aimed at addressing challenges with retrieving and utilizing complex or irrelevant data. These techniques work in tandem with our retrieval system, ensuring that the retrieved documents are effectively leveraged by the models. By optimizing how retrieved information is structured and contextualized, the framework enhances the model's ability to process and apply this data to benchmark questions.

C. Framing: Providing Domain Context to the Model

Framing involves explicitly contextualizing queries or inputs to guide the language model’s reasoning within a specific domain. This technique is particularly effective when paired with retrieved documents, as it "primes" the model to interpret and apply the retrieved information as a subject-matter expert. Framing reduces ambiguity in the retrieved data and focuses the model’s attention on relevant domain knowledge. In our framework, framing is introduced through a static instruction added to the retrieved data: *Framing context = "Imagine you are a Systems Engineering expert."*

D. Chain-of-Thought Reasoning: Structured Query Analysis

CoT Reasoning is a technique where the model is guided to break down its reasoning process into intermediate steps before arriving at a final answer. When applied to retrieved documents, this technique ensures that the model systematically evaluates each piece of information, enabling more robust and accurate reasoning. In our framework, CoT reasoning is introduced through a separate contextual instruction appended to each retrieved document aimed at emphasizing step-by-step analysis: *cot reasoning context = "Based on the retrieved information, evaluate the following carefully. Analyze each piece of information step by step to answer the question"*

Both Framing and CoT Reasoning are automatically incorporated into the system prompt. These instructions are dynamically added alongside the retrieved documents, ensuring the model receives clear guidance on how to interpret and process the information. By automating this process, the framework consistently applies domain-specific context and step-by-step reasoning to every query, enhancing the model's ability to generate accurate and relevant answers.

E. Confidence Level calculation from the Raw Results

In the LM-Harness framework, the output for each answer to question is provided as raw scores called logits, representing the initial assessment of each answer's likelihood. Logits are the raw outputs of the final layer in a neural network before any activation function like softmax is applied. These scores are directly processed using the softmax function, which converts them into a set of probabilities ensuring that the combined total is exactly one. The softmax function is mathematically represented as:

$$P(y=i)=\frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \qquad (1)$$

where $P(y=i)$ is the probability of choice i, z_i is the logit corresponding to choose i, and N is the total number of the choices. Subsequently, these probabilities are multiplied by 100 to convert them into percentage confidence scores. This method enhances numerical stability and clarity. By transforming logits into probabilities, it prevents numerical underflow or overflow, thereby ensuring more reliable outputs.

RESULTS

The results for each model obtained by using specific parameters, including employing or un employing of an LLM judge. The number of retrieved documents were chosen to optimized the maximum accuracy. The models we chose to test includes the following: Mistral 7B, Orca 2 7B, Llama 2 13B and Gemma 2 2B. The performance of the models was tested on the benchmark described earlier, the accuracy and confidence score of each of the models were calculated

A. Mistral 7B

Mistral is our best performing model on the current version of our corpus and RAG Framework with an accuracy scores of 87.5% compared to 79.38% without using the RAG Framework. We were able to reduce the error rate by 39.38%, we achieved the highest results using RAG with an LLM “as a judge” as well as retrieving the top 3 most relevant files, we believe that the model is less sensitive to longer and denser prompts as long as they contain relevant text, therefore benefiting from “a lot” of information. we believe the model is less sensitive to longer and denser prompts as long as they contain relevant text therefore benefiting from “a lot” of information, this is also suggested by worse performs on the same corpus without relevance and from tests with less retrieved text. For the purpose of illustration only, we will present graphical results of Mistral 7B and for the other models' numerical results without the graphical description. It is important to note that the results are presented in a division according to the main categories of the question subjects in the Bench Mark.

Figs no. 4 and 5 describe the accuracy charts of the Mistral 7B model without the use of the RAG infrastructure and with the use of the infrastructure respectively. The division into categories is emphasized, where the Systems Engineering Management category is described in red color. MBSE concepts, languages and tools, described in green color. Integrated Systems Design, analysis and evaluation in pink color and Defense acquisition life- cycle in the light blue color.

We intend to carry out in a follow-up work to this study an analysis and evaluation of the models performance considering these categories.

Fig. 6 shows that despite the significant improvement in accuracy resulting from the application of the RAG framework, there was generally a reduction in the confidence score of the model. Improving the accuracy of LLMs can be very important, but it's essential to consider the impact on the confidence score.

The performance of models Orca 2 7B, Llama 2 13B and Gemma 2 2B were also tested using the benchmark and the RAG framework and the results obtained are detailed below

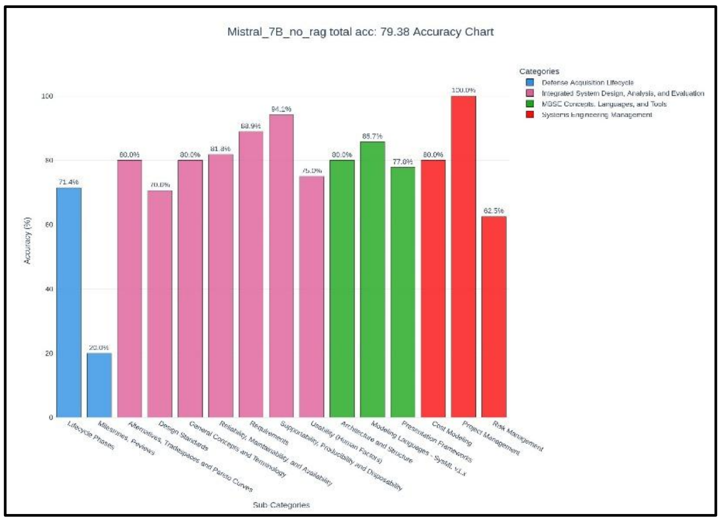


Figure 4. The Accuracy of Mistral 7B without using the RAG infrastructure

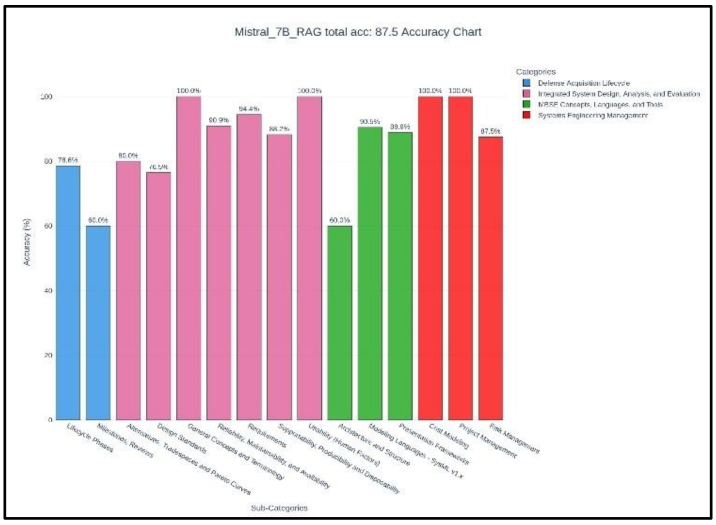


Figure 5. The Accuracy level of Mistral 7B using the RAG infrastructure

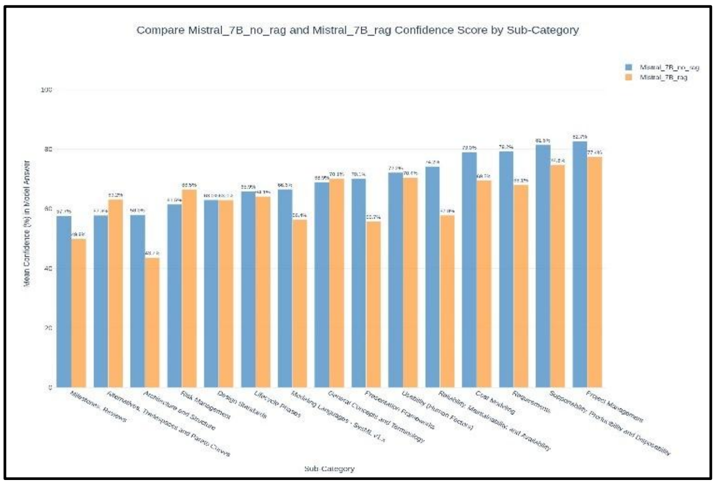


Figure 6. The confidence score of Mistral 7B with and without using the RAG framework

B. Orca 2 7B

Orca 2 7B also improved its accuracy using the RAG system, with an accuracy scores of 82.5% compared to 78.75 without using the RAG Framework. We were able to reduce the error rate by 17.65%. The highest scores were achieved using the current corpus without using LLM as a judge and with only the top 1 retrieved text from the RAG, we believe this model is sensitive to longer prompts but performs well when provided with concise and relevant information, this can be seen in the custom corpus section where we provide the model with just that and achieved the highest accuracy from all the models so far.

C. Llama 2

Llama 2 showed similar improvement to Orca 2 with reduction in error rates of 14.31%. an accuracy scores of 81.25% compared to 78.12% without using the RAG Framework. Similarly, the highest scores were achieved using the current corpus without using LLM as a judge and with the top 1 retrieved text from the RAG, we believe that like Orca this model is sensitive to longer prompts but performs well when provided with concise and relevant information, and again we achieved drastic improvement with the custom corpus.

D. Gemma 2 2B

Gemma 2 2B demonstrated significant improvement, with accuracy increasing from 63.75% to 75.0% corresponding to a reduction of 31.03% in the error rate. The best results were achieved using LLM as a judge and retrieving only the top 1 relevant document. On the custom corpus, Gemma 2 reached an accuracy of 80%, showcasing its ability to handle concise and highly relevant information effectively.

DISCUSSION

Accuracy and confidence score are both crucial metrics, and there is often a delicate balance between them. Enhancing accuracy might lead to a lower confidence score if the model becomes more conservative in its responses. On the other hand, a high confidence score might result in less accurate answers if the model is overly confident in its incorrect responses. In practice, it's important to find the right balance between accuracy and confidence score based on the specific use case of the model. For example, in critical application dealing with complex engineering tasks, it might be preferable to prioritize high accuracy even if it means a lower confidence score. In systems engineering, the choice between high accuracy and high confidence score depends on the specific goals and requirements of the project. For critical projects such as aerospace and defense, accuracy is often the top priority. In these cases, it's crucial for the system to provide precise and reliable results, even if it means having a lower confidence score. High accuracy ensures that the system meets stringent requirements and prevents critical failures. This performance highlights the potential of smaller models to achieve results comparable to those of much larger architectures, such as Llama 2 13B, when paired with an optimized RAG system. By combining techniques like targeted retrieval, the use of an LLM judge for filtering, and concise input formatting, the limitations of smaller parameter counts and simpler architectures can be effectively mitigated. These results suggest that improvements in retrieval and data handling can close the gap between small and large models, showcasing the importance of retrieval-augmented techniques in performance optimization. This is especially important for environments with limited computational resources, where smaller models like Gemma 2 can be used more cost-effectively than larger ones. Additionally, achieving high accuracy with smaller models addresses the increasing demand for energy-efficient and sustainable technology. By prioritizing retrieval methods and input optimization over simply increasing model size, these findings demonstrate a practical approach to making high-performing systems more accessible, particularly in specialized fields such as Systems Engineering.

FUTURE WORK

To further enhance our retrieval system, we are suggest focusing on several key areas:

- Reducing Noise: By improving data corpus chunking, we aim to meticulously label data chunks and develop a robust scoring system to ensure their relevance.
- Improving Relevance Judgment: This involves summarizing data chunks and refining our prompt engineering techniques to ensure that only the most relevant information is retrieved.
- Exploring Generalized Systems: We aim to design a retrieval system that is not tailored to specific models but instead works effectively across a wide range of models, ensuring flexibility and adaptability.

- Expanding the Dataset: Our current dataset is relatively small, and we plan to significantly extend it by incorporating a diverse range of data relevant to systems engineering. This expansion is intended to help our system handle a wider variety of queries and address more complex scenarios, improving its accuracy and utility.
- Real-World Testing: We would like to evaluate the system in real-world, day-to-day systems engineering tasks, moving beyond benchmark-specific testing. This will help us assess its practical utility and performance in real-life scenarios.

While it is not the main purpose of this work, there is potential to establish an open-source System Engineering vector database (vectorDB) in the future. Such a resource could be valuable for those looking to specialize their models in systems engineering. This vectorDB might include a variety of materials, such as articles, academic papers, books, system designs, and technical drawings, forming a comprehensive knowledge base. Although this concept is still in its early stages, it offers a compelling direction for enhancing the availability of high-quality resources and supporting broader advancements in the field.

CONCLUSION

Notably, with the RAG system, Gemma 2 was able to perform on par with or even exceed models with significantly larger architectures, such as Llama 2 13B. This demonstrates how the combination of targeted retrieval, filtering through an LLM judge, and concise input can compensate for a smaller parameter count and simpler architecture. The custom corpus, with its minimal noise, further highlights the model's ability to utilize the retrieved data effectively, closing the performance gap with more complex models. A key challenge identified in our system was the retrieval of excessive irrelevant data, which hindered the model's ability to generate accurate predictions. To address this, we developed a custom corpus using GPT-4, containing concise, question-relevant information. This approach significantly improved performance, It is recommended to integrated "LLM as a Judge" technique, where a secondary LLM evaluates and filters retrieved documents to ensure their relevance to the query.

ACKNOWLEDGEMENTS

The author would like to thank Mr. Asaf Shiloah and Mr. Yotam Gardosh, undergraduate students from the Computer Department at HUJI, for their assistance with the software issues.

REFERENCES

[1] Q. Zhu, X. Zhang, J. Luo, " Biologically Inspired Design Concept Generation Using Generative Pre-Trained Transformers. Journal of Mechanical Design", 145(4), 2023.

[2] J. Han, S. Sarica, F. Shi, and J. Luo, "Semantic networks for engineering design: state of the art and future directions. Journal of Mechanical Design", 144(2), p.020802. 2022.

[3] Y. Han, M. Moghaddam, "Eliciting attribute-level user needs from online reviews with deep language models and information extraction". Journal of Mechanical Design, 143(6), 2021.

[4] A. Kashefi, "A Misleading Gallery of Fluid Motion by Generative Artificial Intelligence" https://www.researchgate.net/publication/380895437_ 24 May 2024.

[5] R. Bell, R. Longshore, R. Madachy, "Introducing SyesEngBench: A novel Benchmark for assessing Large Language Models in Systems Engineering". Acquisition Research Symposium Naval Postgraduate School, 9 May 2024.

[6] HaggingFace SysEngBench https://huggingface.co/datasets/rabell/SysEngBench/tree/main

[7] https://github.com/EleutherAI/lm-evaluation-harness

[8] S. Ward, C. Chapman, " Transforming project risk management into project uncertainty management" International Journal of Project Management, 21(2), Pages 97-105, February 2003.

[9] B.S. Blanchard, W.W.J. Fabrycky, "Systems Engineering and Analysis". USA: Pearson Education Limited, 2011.

D.Hillson, " Effective Opportunity Management for Projects Exploiting Positive Risk" New York CRC Press, November 2003.

[10] R. Ollson, " In search of opportunity management: Is the risk management process enough?" International Journal of Project Management 25(8), , Pages 745-752, November 2007.

"Systems engineering." Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Systems_engineering

[11] USA Department of Defense. "Systems Engineering Guidebook."

https://ac.cto.mil/wp-content/uploads/2022/02/Systems-Eng-Guidebook_Feb2022-Cleared-slp.pdf

[12] MITRE Corporation. "Systems Engineering Guide."

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=aff2aa7c9402dd04739cebd6abd32f5cac462c47

[13] Defense Acquisition University. "Systems Engineering Fundamentals."

https://dml.armywarcollege.edu/wp-content/uploads/2023/01/DAU-Systems-Engineering-Fundamentals-2001.pdf

[14] INCOSE. "Systems Engineering Handbook", Version 2a.

http://www.las.inpe.br/~perondi/21.06.2010/SEHandbookv3.pdf

[15] "Guide to the Systems Engineering Body of Knowledge (SEBoK)."

https://sebokwiki.org/w/images/sebokwiki-farm!w/8/83/Guideto_the_Systems_Engineering_Body_of_Knowledge_v2.9.pdf