SafetySam

Wilhelm Löfsten Oscarsson[1], Vilgot Åström[2], Oskar Danielsson[3]

Abstract
This project presents a chatbot called ”SafetySam” that leverages Retrieval-Augmented Generation (RAG) with the Llama text model to provide users with travel information about various countries. The chatbot integrates a React.js frontend, a Flask server, and a Python-based RAG backend to retrieve contextually relevant data from a preprocessed dataset. Despite challenges such as inference time with longer prompts and some limitations in handling conversational context, SafetySam demonstrates the potential of combining verified external data sources with LLMs to deliver concise and accurate travel advice. Future improvements, including enhanced dataset updates, context-awareness mechanisms, and optimized retrieval techniques, aim to make the system more robust and responsive.

Source code: https://gitlab.liu.se/willo509/tnm114project

Video: https://youtu.be/Pf9zS58ISbQ

Authors
[1] *Information Technology Student at Linköping University, willo509@student.liu.se*
[2] *Information Technology Student at Linköping University, vilas284@student.liu.se*
[3] *Information Technology Student at Linköping University, oskda863@student.liu.se*

Keywords: AI — Chatbot — RAG — NLP —Fullstack application

Contents

1. Introduction

Before traveling to new destinations, travelers often seek information on a wide range of topics, such as safety, health precautions, transportation options, and legal entry requirements. However, finding accurate, relevant, and personalized travel advice can be challenging. While Large Language Models (LLMs) like ChatGPT can provide a personalized experience, it can be difficult to determine the reliability and trustworthiness of the data they offer, especially when it comes to critical travel concerns [1]. Accurate and credible information is essential for ensuring the safety and well-being of travelers, which highlights the need for a solution that combines both personalization and verified advice. In this report, we present SafetySam, a chatbot that attempts to bridge this gap.

2. Theory

The Theory section describes relevant information about Retrieval-Augmented Generation, Natural Language Processing with text models, and Fullstack applications.

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a technique that combines the strengths of Large Language Models (LLMs) with external knowledge retrieval. It works by first mapping chunks of text from a large dataset or knowledge base into a vector space suitable for semantic search and storing these

vectors in a database. The prompt to the LLM is thereafter embedded in the same vector space and the most relevant chunks are retrieved from the database and augmented to the prompt. This provides the LLM with facts from external sources, making it possible for it to e.g. answer domain-specific questions or questions requiring more up-to-date information [2].

2.2 Natural Language Processing with text models

Natural Language Processing (NLP), is the field focused on enabling machines to interpret, understand, and generate human language. NLP models are widely used for tasks such as text classification, sentiment analysis, and machine translation, leveraging deep learning to capture complex linguistic structures and contexts. These models enable applications like chatbots to provide more accurate, relevant, and coherent responses [3]. For example, "LLaMA" is a collection of large-scale text models ranging from 3B to 65B parameters, and has good performance on several NLP benchmarks. Moreover, the models are trained exclusively on publicly available datasets, built with open-source and is a viable option to proprietary models like GPT-3 [4].

2.3 Fullstack applications

A full-stack application integrates both front-end and back-end technologies, enabling seamless communication between the user interface and the server. In frontend web applications, React.js provides an interactive, dynamic user experience by leveraging reusable components and simplifying the development process [5]. Furthermore, using premade open-source components like Material UI (MUI), developers can quickly build attractive and responsive user interfaces [6]. In comparison to server-side, Flask is a popular option, due to it being a lightweight Python-based micro-framework and is used to handle HTTP requests that manages communication between the client and the server. Flask's minimalism and flexibility, makes it useful for creating simple RESTful APIs. Additionally, it allows the front-end to request and receive real-time data from the Python-based backend, which then performs relevant data processing and retrieval tasks [7].

3. Method

This section describes the country information dataset for RAG, the LLM implementation with the mode, the RAG implementation for data retrieval and the fullstack development of the chatbot.

3.1 Dataset

For the RAG implementation, an online dataset from U.S. Data.gov was used, which provides comprehensive travel information for all countries. This dataset was well-suited to the project due to its structured format as a JSON file, ensuring consistent data coverage across all nations. However, the dataset does have some limitations. It is specifically tailored to U.S. citizens and was last updated in 2016, thus lacking information on more recent global events, such as the COVID-19 pandemic [8].

3.2 LLM implementation

This project uses a Large Language Model (LLM) to generate text given a prompt. The specific model used is Llama3.2 with 3.21B parameters released by Meta in September 2024 [9]. The model is downloaded and run locally through the Ollama platform [10], which allows interaction with the pre-trained model through HTTP POST requests. With each request, the model receives a payload consisting of a list of messages, including a system prompt and previous messages in the conversation. The system prompt is the information about the model itself and can be seen below.

System Prompt

> You are a quirky travel advice assistant called SafetySam, and your job is to give concise and clear advice about foreign travel. Limit your answers to one short paragraph maximum. Keep your answer ground in the facts of the CONTEXT.

The model also receives a list of the current chat history, with each message annotated to indicate whether it was sent by the user or the model itself.

3.3 RAG implementation

The dataset was parsed and split into chunks of text. Therefore, we implemented a chunking strategy that segmented the dataset into non-overlapping sections, each limited to 300 characters and containing only complete sentences. As the dataset was structured, each chunk had a straightforward mapping to which field of the dataset it belonged to and what country the text was about. This information was augmented to each text chunk to help the retriever understand what each chunk was about. An example is shown below where the red substring is the annotated part of the chunk.

Example Chunk

> safety_and_security in Norway: As in any other urban area, you should exercise basic security awareness at all times. Don't buy counterfeit and pirated goods, even if they are widely available.

Each chunk was embedded to a vector space using the sentence transformer model "all-MiniLM-L6-v2", available on Huggingface [11]. These vectors were stored in a "ChromaDB" database. Retrieval was done by embedding the prompt to the vectorspace with the same sentence transformer model, whereby a cosine similarity based search was performed to find the most relevant chunks for a given prompt. Each chunk was given a value between zero and one indicating their similarity, where values closer to one are more similar then values closer to zero. The top five chunks which passed

a similarity threshold of 0.5 were retrieved, and augmented to the top of the prompt following the keyword "CONTEXT:" which also appears in the system message. An example of a final, augmented prompt after retrieval is shown below.

Example Chunk

CONTEXT:
safety_and_security in Norway: As in any other urban area, you should exercise basic security awareness at all times. Don't buy counterfeit and pirated goods, even if they are widely available.

Is it safe in norway?

3.4 Fullstack implementation

The full-stack application SafetySam utilizes React.js for the frontend, with the MUI library providing pre-built components. This frontend is connected to a Flask server that facilitates communication between the React.js frontend and the Python backend via RESTful APIs. In the backend, the users text queries are sent to the Flask server, where they get processed by the RAG model. Thereafter the text is divided into relevant chunks, which are then passed to the LLaMA model for interpretation. Lastly, the LLaMa model use both the RAG chunks and the user's input to generate a response, which is then returned to the frontend along with the processed chunk data.

4. Result

This section presents the outcomes of implementing the SafetySam chatbot, including its qualitative performance in delivering accurate responses and quantitative evaluation of its performance on some well-established evaluation metrics.

4.1 Qualitative results

The chatbot responds well when prompted clearly. It gives suitable responses using retrieved relevant context. However, it performs worse when asking follow-up, generic, or no questions. This is evident in the back-and-forth nature of a chat, which limits the usability

4.2 Quantitative results

The quantitative evaluation was carried out using the three metrics context relevancy, faithfulness, and answer relevancy. Context relevancy measures the similarity between the question and the retrieved context. Faithfulness measures the similarity between the context and the answer. Answer relevance measures the similarity between the question and the generated answer. The evaluation was done using 10 questions that we think SafetySam should be able to answer. Whereby the relevant context was extracted, the answer was generated and the scores were calculated. All similarity scores where calculated by embedding the text and using cosine similarity. The measurements are shown in Figure 1. A full overview of the evaluation can be seen in Table 1 in the Appendix.

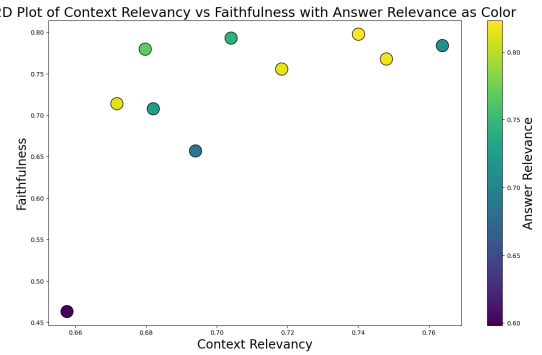

Figure 1. Context Relevancy vs Faithfulness vs Answer Relevancy.

There is a clear correlation between context relevancy and faithfulness. Answer relevance does not seem to correlate as clearly to any of the other metrics, but there is a slight correlation between answer relevancy and context relevancy.

4.2.1 Inference Time

Inference time depends on the prompt's length and the use of the RAG implementation. From Figure 2, we can note some of these example results. It seems that the RAG implementation makes the response times at least twice as long, sometimes reaching times four times as long. The length of the prompt also plays a significant role in the time taken for the user to receive a response. The response time scales poorly as the length of the prompt increases.

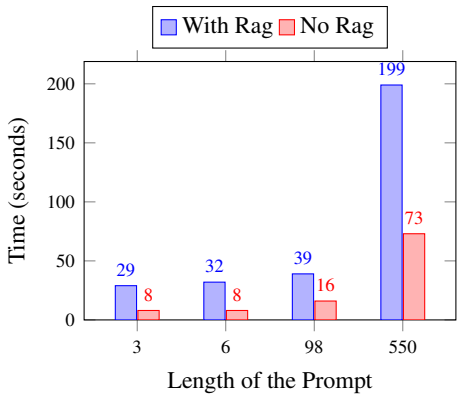

Figure 2. Comparison of time (in seconds) for different prompt lengths with and without Rag.

5. Discussion

This section discusses the results and evaluation of SafetySam, addressing inference time challenges, limitations in message history handling, and potential future improvements for enhanced performance and user experience.

5.1 Result and Evaluation

The correlations seen in Figure 1 indicate that the system is able to generate answers more faithful to the context when the context is more relevant to the question. This means that the context is more useful to the chatbot when it is relevant to the question, which means it will not be as eager to use irrelevant information to answer a question. In order for us to draw more conclusions from this data, the scale of the evaluation would have to increase significantly, with more than only ten questions.

5.2 Inference time

SafetySam's inference time faces challenges due to delays from both the RAG pipeline and the LLaMA 3.2 model. The LLaMA model's large size results in some noticeable latency when processing prompts. However, the RAG implementation, which uses ChromaDB for storing embeddings and the 'all-MiniLM-L6-v2' model for similarity search, adds significant overhead. This can be noticed when having longer conversations with the chatbot, and can be noticed in the uploaded video screencast [12].

To address this issue, some optimizations can be made, like switching to faster vector search libraries like Faiss [13] or using more efficient embedding models like InstructorXL [14]. These alternatives provide improvements in retrieval and embedding efficiency, significantly reducing the overhead of the RAG pipeline. By enabling faster similarity searches and leveraging task-optimized embeddings, the time required to process queries and retrieve context can be minimized. Consequently, these enhancements would lead to a smoother user experience by decreasing inference times and ensuring SafetySam responds more efficiently to user inputs.

5.3 Message History and RAG

In this implementation, the message history is supplied to the LLM responsible for chat completion, but not as input to the RAG implementation. That means that follow-up questions cannot use context from the dataset, as it is not found. For example, suppose the user starts by stating a preferred destination in one prompt, followed by detailed questions in another prompt. In that case, the model is not given any context in order to answer the second prompt, often leading to hallucinations. An example to this can be seen in Figure 3, where the user first informs the model of the intended destination Norway, and then follows up with a query about the safety there. The information given to the RAG implementation after the second prompt contains no mention of the destination itself, making it impossible to retrieve relevant context to answer the question. The model is still able to answer the question, but it is impossible to know whether the answer is true or not, as no reference to data is given.

To get relevant context for the second prompt, all the relevant information would have to be explicitly added to the query itself. The difference can be seen in Figure 4, and this time the model receives relevant context, as seen in Figure 5.

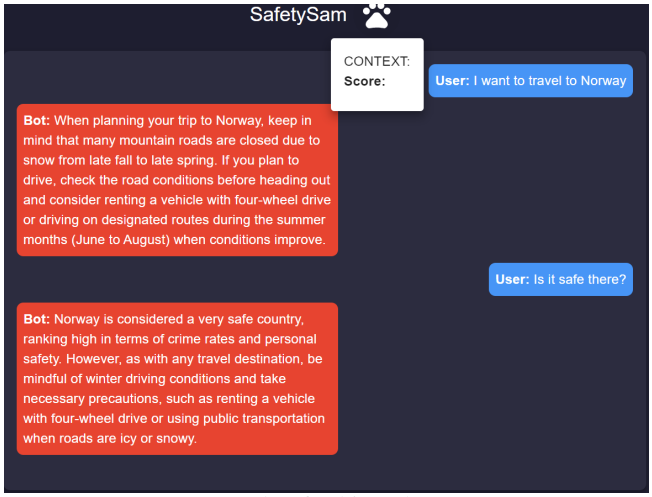

Figure 3. A case example of asking about Norway. Here, no context is supplied to the model.

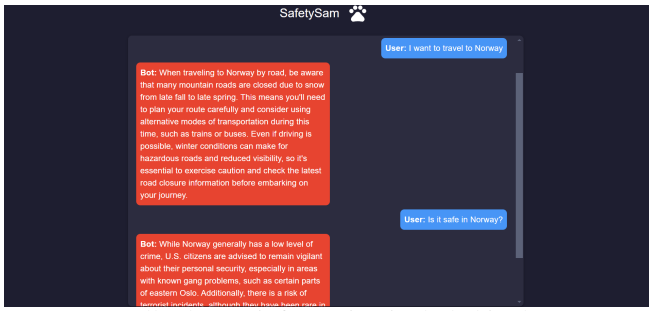

Figure 4. All relevant information included in the prompt.

To solve this issue, one solution could be to include the whole message history as input to the RAG implementation. This was tried briefly, but the model often ended up confused as to what parts of the history was relevant. For example, if a user asked about road conditions in one query and followed up with a question about safety, the RAG implementation would return context about the road conditions for the second query. The performance of this version was deemed worse and the idea was scrapped.

A better solution might be to somehow identify and provide the country in question to the RAG implementation before context retrieval, although it is not clear how the identification would be implemented.

5.4 Future Work

To enhance SafetySam's performance and usability, several improvements could be made. Integrating larger, more advanced models, such as updated LLaMA versions, could improve contextual understanding and response quality, while optimization could mitigate increased computational demands. Updating the dataset to include more recent and globally diverse travel information is crucial to improving relevance, especially post-pandemic. To address inference time issues, Meta's faster vector search libraries like Faiss and task-specific embedding models like InstructorXL could streamline re-

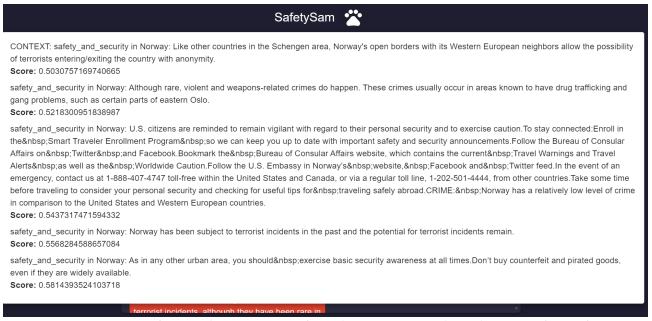

Figure 5. Retrieved context in the example shown in Figure 4.

trieval processes. Moreover, improving context awareness by dynamically identifying key information, such as the destination, would enhance follow-up query handling. Finally, implementing caching for frequent queries and precomputing responses could further reduce latency, improving scalability and user experience.

6. Conclusion

This project successfully demonstrates the potential of combining LLMs with RAG model to create a chatbot that delivers contextually relevant and accurate travel advice. SafetySam integrates a user-friendly frontend with a python backend, effectively creating a well-structured and efficient application. While the chatbot performs well in providing concise and accurate responses, challenges such as inference time delays (especially with messages with prompts) and limitations in conversational context management remain. Addressing these issues through techniques like dataset updates, improved context-awareness mechanisms, and optimized retrieval processes will enhance the system's scalability and user experience. Overall, SafetySam highlights the value of leveraging verified external data sources in conjunction with LLMs to provide reliable information, making it a strong foundation for further development and refinement in the travel advisory domain.

References

[1] Maximilian Mozes, Xuanli He, Bennett Kleinberg, and Lewis D. Griffin. Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities. *arXiv preprint arXiv:2308.12833*, 2023. [Accessed 14-10-2024].

[2] IBM Research. What is retrieval-augmented generation (rag)? *IBM Research Blog*, 2023. [Accessed 12-10-2024].

[3] The Batch. Natural language processing matters. *DeepLearning.AI*, 2020. [Accessed 14-10-2024].

[4] H. Touvron et al. Llama: Open and efficient foundation language models. *Hugging Face Documentation*, 2023. [Accessed 14-10-2024].

[5] Jordan Walke. React top-level api. [Accessed 15-10-2024].

[6] MUI Team. Material ui documentation, 2024. Accessed: 15-10-2024.

[7] M. Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2 edition, 2018. [Accessed 15-10-2024].

[8] US Data.Gov. Country travel information. `https://catalog.data.gov/dataset/country-travel-information`, 2016. [Accessed 10-10-2024].

[9] Llama 3.2 — Model Cards and Prompt formats — llama.com. `https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2`. [Accessed 21-10-2024].

[10] Ollama - llama3.2. `https://ollama.com/library/llama3.2`. [Accessed 21-10-2024].

[11] Hugging Face. all-minilm-l6-v2, 2024. Accessed: 15-10-2024.

[12] Oskar Danielsson Wilhelm Löfsten Oscarsson, Vilgot Åström. Safetysam - screencast. YouTube video, 2024. Accessed: 2024-10-15.

[13] Facebook AI Research. Faiss: A library for efficient similarity search and clustering of dense vectors. `https://github.com/facebookresearch/faiss`, 2023. Accessed: 2024-11-26.

[14] Hugging Face. Instructorxl: Instruction-finetuned text embedding model. `https://huggingface.co/hkunlp/instructor-xl`, 2023. Accessed: 2024-11-26.

Appendix

Table 1. Evaluation Results with asked questions and their respective scores.

Question	Contexts (excerpt)	Answer (excerpt)	Context Relevancy	Faithfulness	Answer Relevance
Do I need a visa when traveling to Japan?	*entry_exit_requirements in Japan: For further ...*	*For U.S. citizens, no visa is required ...*	0.740	0.797	0.823
In French Guiana, what is the name of the threat system?	*safety_and_security in French Guiana: French ...*	*The threat rating system in French Guiana is ...*	0.748	0.768	0.817
Is it dangerous in Afghanistan?	*safety_and_security in Afghanistan: Private ...*	*Yes, Afghanistan is considered a high-risk ...*	0.704	0.793	0.743
Do I need any vaccination going to Somalia?	*health in Somalia: Credit cards are typically ...*	*If you're planning a trip to Somalia, you ...*	0.718	0.755	0.815
Is it safe when traveling on the countryside in Iceland?	*travel_transportation in Iceland: Further ...*	*When traveling on the countryside in Iceland, ...*	0.672	0.714	0.813
What specific laws do I need to know when traveling to Vatican City?	*safety_and_security in Vatican City: Be aware that ...*	*When traveling to Vatican City, be aware that ...*	0.694	0.657	0.688
What visa do I need to travel to China?	*entry_exit_requirements in China: The ...*	*As a U.S. citizen traveling to China for multiple ...*	0.682	0.708	0.723
What local laws exist in Germany?	*local_laws_and _special_circumstances in Germany: ...*	*When it comes to local laws in Germany, there ...*	0.658	0.463	0.598
How are the road conditions in Thailand?	*local_laws_and _special _circumstances in Thailand: ...*	*The road conditions in Thailand can be challenging ...*	0.680	0.779	0.769
What travel safety information is there in Bangladesh?	*travel_transportation in Bangladesh: Road conditions ...*	*Bangladeshi roads can be hazardous due to overloading ...*	0.764	0.784	0.709