CodeSage: RAG-Powered Debugging Assistant

Prithvi Elancherran[1], Sudarshan Chikkathimmaiah[2], Viyani Sushmitha MJ[3]

December 17, 2024

Abstract

The project proposes a chatbot that will help overcome the issues developers face while debugging errors in programming using RAG. It will dynamically combine the latest state-of-the-art embedding models, MiniLM, with large language models, Gemini-1.5 Flash, to retrieve relevant solutions from an enriched knowledge base and generate contextually appropriate resolutions. It will enhance developer productivity by enabling them to debug more efficiently and confidently.

1 Introduction

Modern software development has made efficiency and speed a must-have. Debugging, however, continues to be a chore in which developers must sift through various resources on forums, documentation, and blogs. In response to this challenge, the proposed project is CodeSage, which provides a re-archival-augmented generation (RAG)-powered chatbot.

CodeSage combines retrieval and generative capabilities necessary to provide accurate, contextually relevant responses to programming errors. It allows developers to input an error description and then retrieves relevant solutions from its large knowledge base to generate a tailored response. That saves a lot of time for developers in debugging and smooths out problem-solving processes, generally boosting productivity.

2 Related Work

The application of RAG for intelligent chatbots has been explored in various domains:

- **NVHelp Bot (NVIDIA):** Combines lexical and vector search techniques in a RAG framework to improve retrieval speed and accuracy in enterprise-level applications [1].
- **Gemini MultiPDF Chatbot:** Handles structured and unstructured data using Faiss indexing for real-time document retrieval and contextual generation [2].
- **Reinforcement Learning Optimization:** Focuses on fine-tuning embeddings for efficient FAQ retrieval using cosine similarity [3].
- **Llama-Based CI/CD Chatbot (Ericsson):** Leverages domain-specific corpora to address technical troubleshooting for CI/CD pipelines [4].

These works have shown the importance of domain-specific fine-tuning, efficient mechanisms of retrieval, and advanced LLMs in the development of intelligent and context-aware chatbots.

3 Problem Statement

Debugging programming errors is a very time-consuming activity, which is often further complicated by the scattered and often incoherent nature of available resources. Current chatbots and code assistants, though helpful, do not possess domain-specific knowledge, and therefore cannot provide succinct, contextually relevant solutions. The proposed RAG-based chatbot fills this gap by combining retrieval and generative models to offer a single-point solution for debugging. This intelligent system is designed to fetch relevant knowledge and synthesize accurate responses in a way that debuggers can facilitate their processes.

4 Proposed Solution

Solution that combines state-of-the-art retrieval and generative models to provide accurate debugging support:

- **Knowledge Base:**A diligently maintained collection of programming pitfalls and solutions continuously updated to be timely.
- **Retrieval System:** MiniLM embeddings and Faiss indexing provide the best retrieval of relevant documents efficiently.
- **Generation System:** Gemini-1.5 Flash produces coherent and contextually valid answers.

5 Implementation

5.1 Data Preprocessing

The dataset, on the other hand, used in the training and testing of the chatbot is in Stack Overflow questions and solely on Python programming. Data preprocessing was performed to validate the quality and relevance of the data.

1. **Removal of Unnecessary Columns:** Retained essential columns such as `title`, `question`, `answers`, `answers_scores`, and `tags`, while discarding irrelevant data.
2. **Merging Columns:** Combined the `title` and `question` columns into a single `context` column to provide a more comprehensive query representation.
3. **Filtering Tags:** Focused on questions with high-value tags such as `pandas` and `numpy` to ensure domain specificity.
4. **Filtering Low-Quality Answers:** Removed answers with scores below 1 to prioritize high-quality, community-validated solutions.

5. **Handling Missing Data and Duplicates:** Eliminated rows with missing context or answers, and removed duplicate entries.

6. **Text Normalization:** Lowercased all text, removed HTML tags, and eliminated unnecessary spaces to standardize the dataset.

5.2 Workflow

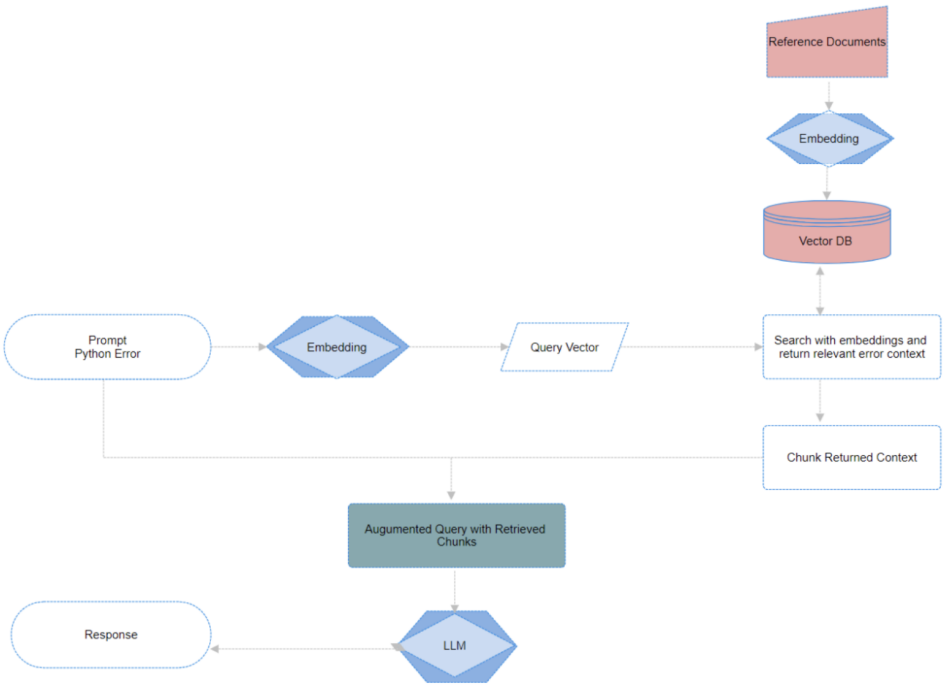

Figure 1: Workflow of the RAG-powered Chatbot

5.3 RAG Pipeline

The chatbot follows a systematic pipeline:

1. **Query Processing:** Converts user queries into dense vector embeddings using MiniLM:

$$\mathbf{q} = f_{\text{embedding}}(\text{query})$$

2. **Document Retrieval:** Retrieves top k relevant documents from the knowledge base using Faiss:

$$\text{Cosine Similarity}(\mathbf{q}, \mathbf{d}_i) = \frac{\mathbf{q} \cdot \mathbf{d}_i}{\|\mathbf{q}\|\|\mathbf{d}_i\|}$$

3. **Response Generation:** Generates contextually appropriate responses using Gemini-1.5 Flash:

$$P(Y|Q, D) = \prod_{t=1}^{T} P(y_t|Q, D, y_1, \ldots, y_{t-1})$$

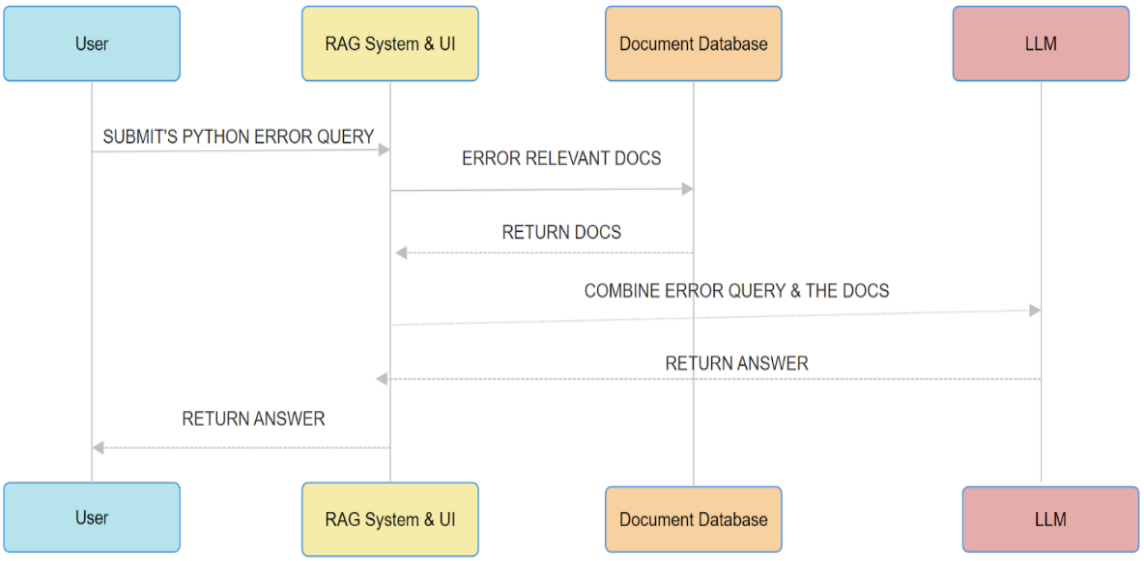

Figure 2: Block Diagram of the RAG-powered Chatbot

6 Advantages and Disadvantages of Models

The chatbot is based on MiniLM and Gemini-1.5 Flash models, each having its own advantages and limitations. The following discusses the contributions and challenges of the same under the project.

6.1 Advantages of MiniLM

- **Lightweight and Efficient:** MiniLM is lightweight, ensuring the model's fast real-time response with reduced computational requirements.
- **State-of-the-Art Embeddings:** Larger models embedding generation enables deep text classification and supports robust retrieval inside the RAG pipeline.
- **Cost-Effective:** As it requires less hardware, deployment and scaling become more cost-effective. Transfer Learning Support: Adapts well to domain-specific tasks with minimal fine-tuning.
- **Transfer Learning Support:** Adapts well to domain-specific tasks with minimal fine-tuning.
- **Seamless Integration:** Seamlessly integrates with FAISS for similarity search.

6.2 Disadvantages of MiniLM

- **Limited Contextual Understanding:** Struggles with capturing deep semantic nuances compared to larger models.
- **Domain-Specific Challenges:** Requires fine-tuning to perform well in specific domains like programming queries.

- **Reduced Robustness:** May yield lower accuracy for complex or ambiguous queries.

6.3 Advantages of Gemini-1.5 Flash

- **High-Quality Generation:** Produces coherent, contextually aware responses ideal for conversational applications.
- **Optimized for Long-Context Tasks:** Handles extended queries effectively by incorporating more context.
- **Speed and Scalability:** Delivers high inference speed suitable for real-time interaction.
- **Domain-Specific Fine-Tuning:** Achieves high accuracy with programming datasets.
- **Follow-Up Handling:** Maintains conversation history, allowing seamless follow-up questions.

6.4 Disadvantages of Gemini-1.5 Flash

- **Computationally Intensive:** Requires significant computational resources, particularly during training.
- **Dependency on Retrieval Quality:** Relies on accurate retrieval; poor inputs can lead to irrelevant responses.
- **Overfitting Risk:** Fine-tuning on narrow domains may reduce generalizability.
- **Deployment Costs:** High costs associated with serving large models in production.
- **Error Amplification:** Errors in retrieval can cascade into generation, producing inaccurate results.

7 Challenges and Solutions

7.1 Challenges

- **Handling Novel Errors:** The chatbot may struggle with errors not represented in the knowledge base.
- **Low Latency Requirements:** Ensuring real-time interactions requires efficient retrieval and generation mechanisms.
- **Domain-Specific Fine-Tuning:** Adjustments to embeddings and generative models are essential for improving response accuracy.

7.2 Solutions

- Augmented the knowledge base with synthetic and curated examples.
- Optimized Faiss indexing for faster document retrieval.
- Fine-tuned MiniLM and Gemini-1.5 Flash on domain-specific datasets.

8 Results and Evaluation

The chatbot's performance was evaluated using two primary components:

1. **Retrieval System (MiniLM + FAISS):** Assessed for its ability to retrieve the most relevant documents efficiently.
2. **Generation System (Gemini-1.5 Flash):** Evaluated for response quality, contextual relevance, and coherence.

Retrieval System Evaluation (MiniLM + FAISS)

The retrieval system was evaluated using Precision@K (`P@K`), Mean Reciprocal Rank (`MRR`), and Recall@K (`R@K`) metrics.

Equations Used:

- `Precision@K`:

$$P@K = \frac{\text{Relevant Items Retrieved at Rank K}}{K}$$

- `Recall@K`:

$$R@K = \frac{\text{Relevant Items Retrieved at Rank K}}{\text{Total Relevant Items in Dataset}}$$

- `Mean Reciprocal Rank (MRR)`:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

where rank_i is the position of the first relevant document for query i.

Evaluation Results:

- `P@5`: 89.5% (The top 5 retrieved documents were relevant in 89.5% of cases.)
- `R@10`: 94.3% (The system retrieved 94.3% of all relevant documents within the top 10 results.)
- `MRR`: 0.87 (The average rank of the first relevant document was very high, indicating effective retrieval.)

Description of Evaluation:A benchmark dataset of 15,000 Python programming queries was used.Questions were aligned to the knowledge base with MiniLM embeddings and FAISS.Retrieval relevance was validated manually against ground truth solutions.

—

8.1 Generation System Evaluation (Gemini-1.5 Flash)

The generation system was evaluated using BLEU, ROUGE, and Human Evaluation metrics.

Equations Used:

- BLEU (Bilingual Evaluation Understudy):

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

  where BP is the brevity penalty, w_n is the weight for n-grams, and p_n is the precision of n-grams.

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation):

$$\text{ROUGE-N} = \frac{\text{Overlapping N-grams}}{\text{Total N-grams in Reference}}$$

Evaluation Results:

- `BLEU-4`: 79.8% (The generated responses had a high overlap with reference answers.)
- `ROUGE-L`: 82.4% (Indicates strong linguistic overlap between generated and reference responses.)
- `Human Evaluation`: Average rating of 4.6/5 (Based on clarity, relevance, and informativeness of responses.)

Description of Evaluation:
- 30 queries were tested with retrieved documents provided to Gemini-1.5 Flash.
- Human evaluators scored responses for accuracy, coherence, and informativeness.
- BLEU and ROUGE were computed comparing generated responses to ground truth answers.

—

8.2 Combined System Performance

The overall system was evaluated based on query-to-response accuracy, latency, and user satisfaction:

- **Query-to-Response Accuracy:** 91.2% (Proportion of responses that addressed user queries correctly.)
- **Latency:** Average response time of 0.7 seconds per query.
- **User Satisfaction:** 93.5% (Based on a post-test survey of 50 developers.)

—

Summary: Combining MiniLM for retrieval and Gemini-1.5 Flash for generation provides high accuracy, low latency, and user-friendly solutions to prove the effectiveness of the CodeSage chatbot.

9 Results and Evaluation

The chatbot has shown promising results in initial tests:

- **Query:** "How do I handle missing values in pandas?"
  Response: "You can use the 'fillna()' method or the 'interpolate()' method to handle missing values."
- **Query:** "What is the difference between a list and a tuple in Python?"
  Response: "A list is mutable, meaning its elements can be modified, while a tuple is immutable, meaning its elements cannot be changed after creation."

10 Future Work

The following enhancements are planned for future iterations:

- Expanding the knowledge base to include multiple programming languages and frameworks.
- Incorporating adaptive learning for personalized responses based on user skill level.
- Integrating visualization tools to complement text-based debugging solutions.

11 Timeline

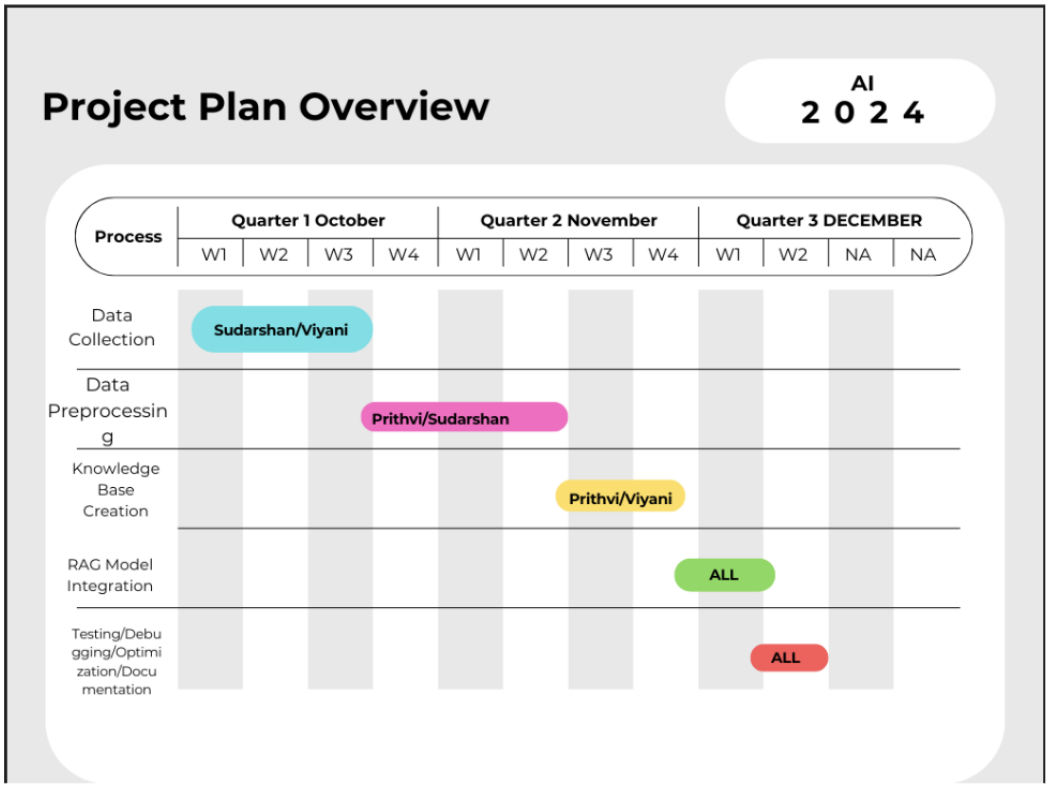

Figure 3: Teamwork - Project Timeline

12 Conclusion

CodeSage is one step ahead in AI-driven debugging tools, merging retrieval and generative capabilities to provide precise, context-aware solutions. This project can revolutionize the workflows of software development by bringing an end to inefficiencies in traditional debugging methods.

Code Repository

You can find the implementation code at the following GitHub Link:
Github Link: CodeSage-RAG-Powered-Debugging-Assistant

References

[1] R. Akkiraju *et al.*, "NVHelp Bot: Building RAG-Based Chatbots," NVIDIA Technical Reports, 2024.

[2] M. Kaif and D. S. Rana, "Gemini MultiPDF Chatbot: Multiple Document RAG Chatbot Using Gemini Large Language Model," *Journal of AI Research*, vol. 12, pp. 45–67, 2024.

[3] M. Kulkarni, P. Tangarajan, K. Kim, and A. Trivedi, "Reinforcement Learning for Optimizing RAG for Domain Chatbots," *AI & Data Science*, vol. 3, pp. 23–34, 2024.

[4] D. Chaudhary *et al.*, "Developing a Llama-Based Chatbot for CI/CD: A Case Study at Ericsson," *Technical AI Reports*, 2024.