Bank Service Chatbot

Nafis Anzum
Department of Computer Science & Engineering
North South University
Dhaka, Bangladesh
nafis.anzum@northsouth.edu

Suhana Islam
Department of Computer Science & Engineering
North South University
Dhaka, Bangladesh
suhana.islam@northsouth.edu

Sihab Mahmud
Department of Computer Science & Engineering
North South University
Dhaka, Bangladesh
sihab.mahmud@northsouth.edu

Soleman Hossain
Department of Computer Science & Engineering
North South University
Dhaka, Bangladesh
soleman.hossain@northsouth.edu

***Abstract*—This paper presents the development of a bank service chatbot using a Large Language Model (LLM) and Retrieval-Augmented Generation (RAG) techniques. The main goal of the chatbot is to provide an efficient and automated solution for answering customer queries regarding banking services. The system integration with existing bank services through PDF-based data retrieval enables it to generate quick and accurate responses, helping reduce waiting time for customers and improving overall service efficiency. The chatbot is implemented using Python, Streamlit, and Ollama models, providing an ease of integration into real-world banking systems.**

Keywords—Bank service, Chatbot, Large language models, Retrieval–Augmented Generation, ChromaDB, Ollama, Customer support

I. Introduction

Recently, the integration of artificial intelligence (AI) into customer service systems has significantly improved the accessibility of services in various industries. Particularly, the banking sector has been seen to shift towards AI-based chatbots that can handle customer queries and provide instant responses. Traditional methods of customer support, such as call centers and email support, often led to long wait times and customer dissatisfaction. In order to address these challenges, this paper explores the development of an intelligent Bank Service Chatbot that provides a faster response to customers with the help of Retrieval-Augmented Generation (RAG) with a Large Language Model (LLM), specifically Ollama's Llama 3 (3B).

This project supports both RAG & ChromaDB as a vector database for efficient information retrieval. RAG models have the ability to retrieve relevant information from an external knowledge base to generate precise responses. The chatbot can provide answers based on the bank's policies, procedures, and services by processing PDF-based bank documents. This approach reduces operational costs and enhances customer support.

II. System Design and Architecture

A. Overall Architecture

The bank service chatbot is designed to interact with users through a simple user interface. The system architecture consists of three primary components:

1. User Interface (UI)
2. Language Model (LLM)
3. Information Retrieval (IR) System (ChromaDB)

The User Interface (UI) is implemented using Streamlit, allowing the users to interact with the chatbot. Upon receiving a query, the system retrieves relevant information from the database and generates a response through the Llama 3 model.

The Language Model (LLM), Ollama Llama 3, processes the retrieved information and generates natural language responses to user queries.

Finally, the Information Retrieval (IR) System (ChromaDB) is responsible for storing the bank's documents and providing fast and efficient retrieval of relevant data based on user queries.

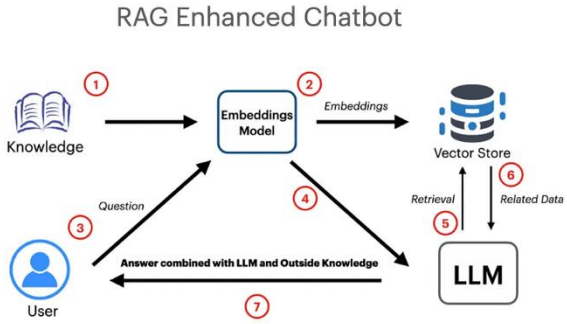

Fig. 1: *Bank Service Chatbot Pipeline*

B. Data Flow & Interaction

The interaction between the user and the chatbot takes place through the Streamlit interface. The system first performs a semantic search using the vector database to find chunks that are relevant to the query. Once the relevant chunks are identified, they are passed to the Llama 3 model, which generates a contextual answer based on the information retrieved. This response is then displayed to the user.

The MultiQueryRetriever was used to handle multiple queries in order to improve the efficiency and accuracy of retrieving information from the PDF by considering multiple queries at once.

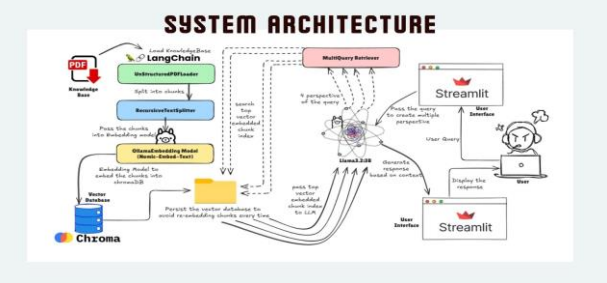

Fig. 2: *System Architecture*

III. Implementation

The implementation of the Bank Service Chatbot involved several key components including data preprocessing, vector database integration and chatbot development.

The entire system was designed with modular Python scripts to allow easy updates and scalability, ensuring that the chatbot could efficiently respond to a wide range of customer queries.

A. Data Collection and Preprocessing

We used our selected PDFs, which contain various customer service procedures, FAQs, and bank policies, as a base for our chatbot. The PDF was processed to extract text and convert it into a suitable format for LLM to understand.

Preprocessing steps:

- **PDF Extraction:** We used Python libraries such as UnstructuredPDFLoader to extract the text from the PDF.
- **Text Chunking:** We divided the extracted text into chunks of 1024 with an overlap of 256 to improve retrieval accuracy.
- **Vectorization:** The chunks were converted into vector forms using the Llama 3 model.

B. Database Creation & Integration

- Using ChromaDB as a vector Database, each chunk is indexed based on its vector representation, allowing for efficient similarity search. In the create_database.py script, appropriate parameters for chunk size and overlap were set, ensuring that the database is optimized for quick retrieval.

C. Chatbot Functionality

Streamlit framework was used as an interactive web-based interface. The user can type in queries, and the chatbot responds with contextually relevant answers based on the data stored in the ChromaDB.

The interaction with the chatbot follows these steps:

1. **User Input:** A customer submits a query.
2. **Search in ChromaDB:** ChromaDB searches for the most relevant document chunks based on the query.
3. **Response Generation:** The Llama 3 model generates an answer based on the retrieved documents.
4. **Display Answer:** The answer is displayed in the UI.

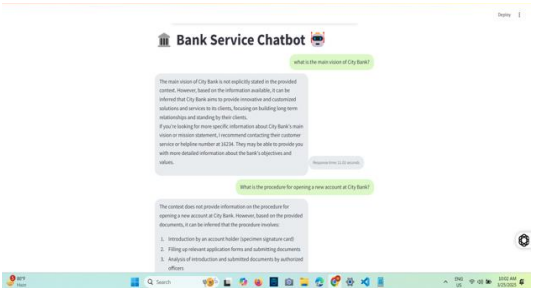

Fig. 3: *Chatbot UI Design*

D. Algorithm

- Start
- Load the ChromaDB vector database
- Receive user query from the Streamlit interface
- Preprocess the query (e.g., remove special characters, lowercasing)
- Convert the query into an embedding vector using the Nomic Embed Text model
- Perform a similarity search in ChromaDB to retrieve the top-k relevant document chunks
- Format the retrieved context as input for the LLM (Ollama Llama 3)
- Generate a response using the LLM with the retrieved context
- Display the response on the Streamlit interface
- End

IV. Result and Discussion

The chatbot was evaluated based on its accuracy and response time with the help of BLEU, ROUGE and F1 score. Automation testing of 1179 test cases was performed, and the chatbot provided accurate and appropriate responses based on the content mentioned in the PDF.

Fig. 4: *Automation Testing Results*

A. Evaluation Metrics

To assess the performance of our chatbot, we employed several standard Natural Language Processing (NLP) evaluation metrics: **BLEU**, **ROUGE**, and **F1-score**. These

metrics help us to quantify the quality, relevance, and accuracy of the generated response.

*1) BLEU (Bilingual Ev*aluation Understudy): BLEU is a precision-based metric used for evaluating machine translation and text generation. A higher BLEU score indicates greater similarity to the reference.

2) ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE measures the overlap between the generated response and the reference text, focusing on recall rather than precision.

3) F1-Score: The F1-score is the harmonic mean of precision and recall. This metric helps us to evaluate how accurately the chatbot identifies and responds with relevant information.

B. Performance Evaluation

To access the effectiveness and reliability of the banking assisstant chatbot, we evaluated its performance based on several key metrics.These critieria are crucial for determining the chatbot's suitability for real-time customer service environment.

- **Accuracy:** The chatbot's responses were compared to expected answers to evaluate the correctness of the generated responses. The chatbot achieved high accuracy in understanding and responding to domain specific questions, particularly those related to account queries, balance inqueries, and general banking information.
- **Response Time:** The time taken by the chatbot to generate a response was measured to ensure that it meets the expectations of real-time customer support systems. Reponse time is critical in real-time systems where user patience is limited.
- **User Satisfaction:** User feedback was collected to gauge satisfaction levels with the chatbot's performance.

TABLE I. EVALUATION METRICS FOR CHATBOT PERFORMANCE

Metrics	Metrics Evaluation		
	Score	***Type***	***Remarks***
BLEU	0.0562	Automatic	Low score typical for short responses.
ROUGE-1	0.2544	Automatic	moderate lexical similarity.
ROUGE-2	0.1119	Automatic	lower score indicates variability in phrasing.
ROUGE-L	0.2289	Automatic	reflects structural similarity between responses.

The results demonstrated that the chatbot exhibits commendable performance in both accuracy and speed. The chatbot effectively leverages the RAG pipeline and LLM to generate accurate responses to customer queries. The chatbot offers a scalable and efficient solution for enhancing customer support in the banking sector.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our supervisor, Shafin Rahman, of the Department of Computer Science, for his invaluable guidance and unwavering support throughout this project. Additionally, we extend our deepest appreciation to the developers of Ollama and ChromaDB for providing the robust tools that enabled the successful completion of this endeavor.

REFERENCES

[1] *APSURSI 2023 Paper Format Guide, [Online]. Available: https://www.google.com/url?q=https://2023.apsursi.org/Papers/PaperFormat/SPC_Submission_Regular.pdf&source=gmail-imap&ust=1745837054000000&usg=AOvVaw2v9gpKNuyWxINWKYNn-ZNo*

[2] M. S. Islam, A conversational agent-based customer support service, BRAC University, 2010. [Online]. Available: https://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/2967/10104074.pdf

[3] Overleaf, "IEEE Conference Template," [Online]. Available: https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn

[4] Ollama, "Run open-source large language models locally," [Online]. Available: https://ollama.com