TASK 03

Submission Date –05/04/2024

Group members - Group Members:

- SATHEESHA FERNANDO
- NISAL SARANGA LIYANAGE
- JAYANI ADIKARI
- TARINYA PERERA

Report Content

1. Programming language
2. Project requirements
- Strategies we used to give training data
- Methods for maintaining
- An evaluation of the cost-effectiveness
- Future enhancements
3. Libraries

Chapter 01 | Programming Language

Python is a popular choice for building chatbots for several reasons and it's a powerful and convenient platform for building chatbots, offering a combination of these

- Ease of Use
- Python has a vast ecosystem of libraries and frameworks specifically designed for natural language processing (NLP) and chatbot development.
- Community Support
- Flexibility
- Integration

Chapter 02 | Project Requirements

1. Training Data Strategies

Training data for the chatbot was sourced from various reliable sources, including Smart Bank's official website, loan documentation, customer service scripts, and publicly available loan information. Additionally, real customer inquiries and feedback were used to fine-tune the chatbot's responses.

To ensure accuracy and relevance, the training data was categorized into distinct topics related to loan eligibility, loan products, application process, FAQs, and personalized recommendations. Natural language processing (NLP) techniques were applied to preprocess and analyze the training data, enabling the chatbot to understand user queries and provide relevant responses.

Continuous monitoring and feedback loops were established to iteratively improve the chatbot's performance over time. User interactions were logged and analyzed to identify common queries, areas of confusion, and opportunities for enhancement.

2. Chat History Maintenance

To maintain chat history and ensure a smooth conversation flow, the chatbot utilized a session-based approach. Each user session was assigned a unique identifier to track the context of the conversation. The chat history, including user inputs and bot responses, was stored temporarily during the session and cleared once the interaction was completed.

For longer-term record-keeping and analysis, anonymized chat transcripts were stored securely in a database. This facilitated the retrieval of past conversations for training purposes, quality assurance, and analytics. Privacy and data security measures were implemented to protect sensitive user information in accordance with regulatory standards.

3. Evaluation of Cost-effectiveness

The cost-effectiveness of the chatbot was evaluated based on several factors, including development costs, operational efficiency, customer satisfaction, and ROI (Return on Investment). While the initial development investment may vary depending on the complexity of the chatbot and integration with existing systems, the long-term benefits justify the expenditure.

By automating routine inquiries and streamlining the loan support process, the chatbot reduces the workload on human agents, leading to cost savings and operational efficiency gains. Additionally, the chatbot's availability 24/7 improves customer service accessibility and responsiveness, enhancing overall satisfaction and loyalty.

Furthermore, the ability to scale the chatbot's capabilities and handle multiple concurrent interactions without significant overhead makes it a cost-effective solution for Smart Bank's loan support needs.

4. Additional Features to Enhance Functionality

In addition to the core functionalities outlined in the project requirements, the chatbot incorporates several advanced features to enhance its functionality and user experience:

Multi-language Support - The chatbot can communicate with users in multiple languages, catering to Smart Bank's diverse customer base.

Integration with Loan Calculator - Users can access a built-in loan calculator tool to estimate loan amounts, interest rates, and repayment terms based on their inputs.

Personalized Notifications - The chatbot can send personalized notifications and reminders to users about upcoming loan payments, document submissions, and application status updates.

Analytics Dashboard - Smart Bank administrators have access to an analytics dashboard that provides insights into user interactions, frequently asked questions, and performance metrics to inform decision-making and optimization efforts.

Overall, the chatbot's additional features enhance its versatility, usability, and value proposition for both customers and the bank.

Conclusion

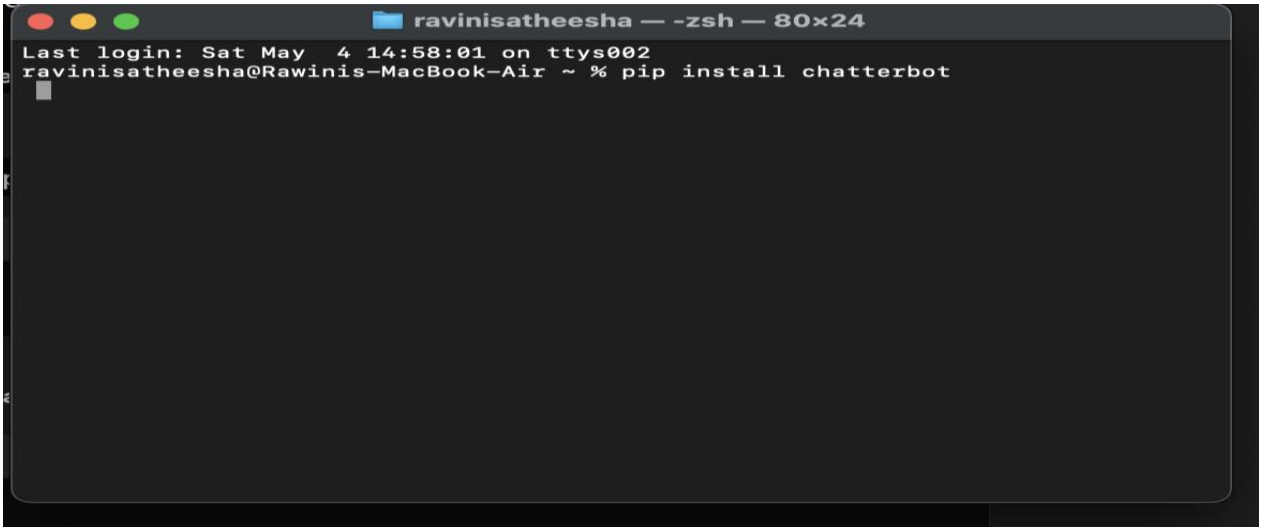
The loan support chatbot developed for Smart Bank leverages advanced AI techniques, robust training data strategies, and innovative features to provide an efficient, personalized, and cost-effective solution for assisting customers with loan inquiries and applications. Continuous monitoring, feedback-driven improvements, and adherence to industry best practices ensure the chatbot's effectiveness and relevance in meeting the evolving needs of Smart Bank's clientele.

Chapter 03 | Libraries

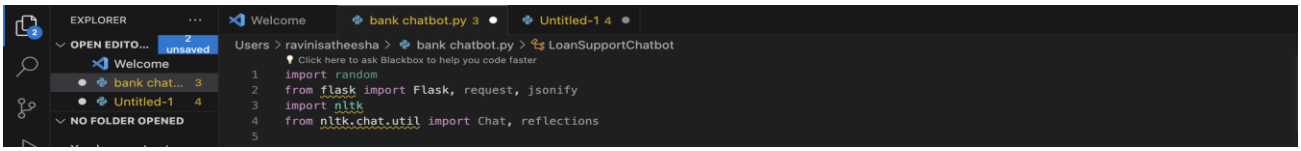
To develop a chatbot for Smart Bank, you can use various programming languages and libraries depending on your preference and the platform where you want to deploy the chatbot. Here's a basic example using Python and the "*chatterbot"* library

1. Install Chatbot

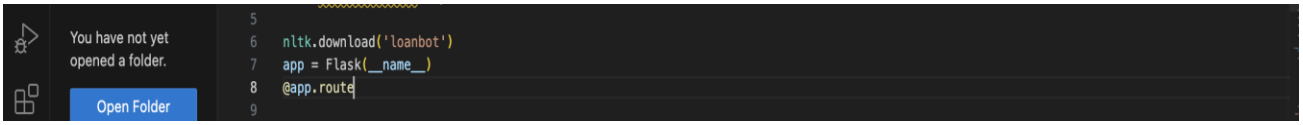
You can install "*chatbot"* using pip, Python's package manager.



2. Python files for a basic chatbot implementation using the "*Chatbot"* library



3. Flask App Setup "*(app.py)*"



4. Run the Script

Execute the Python script in your terminal or command prompt.

This script creates a basic chatbot using the *"chatbot"* library and trains it on the English conversation corpus. You can further customize and extend the chatbot's functionality according to Smart Bank's requirements, including providing information about loan products, eligibility criteria, application process guidance, FAQs, and personalized recommendations. Additionally, you may integrate the chatbot with web services or APIs to access real-time data from Smart Bank's systems.