VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI, KARNATAKA – 590018

[A STATE TECHNOLOGICAL UNIVERSITY]

A PROJECT REPORT ON

"BANKING CHAT BOT: AN INTELLIGENT ASSISTANT SYSTEM USING NLP"

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SUBMITTED BY

ABHISHEK ADANNAVAR USN : 2BA21AI005	**PRASANNA NAGARALE** USN : 2BA21AI030
SUKESH PADAGATTI USN : 2BA21AI054	**SUPREETH MADARI** USN : 2BA21AI055

UNDER THE GUIDANCE OF
PROF. JAYASHEELA D.K
ASSISTANT PROFESSOR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

B. V. V. SANGHA'S
BASAVESHWAR ENGINEERING COLLEGE, S. NIJALINGAPPA VIDYANAGAR
BAGALKOTE – 587102
[A GOVERNMENT AIDED COLLEGE, RECOGNIZED BY AICTE, PERMANENTLY AFFILIATED TO VTU, BELAGAVI & ACCREDITED BY NAAC WITH 'A' GRADE]

2024-25

B. V. V. Sangha's

BASAVESHWAR ENGINEERING COLLEGE,

BAGALKOTE-587102

CERTIFICATE

This is to Certify that the project work entitled **"BANKING CHAT BOT: AN INTELLIGENT ASSISTANT SYSTEM USING NLP"**, carried out by **MR. ABHISHEK ADANNAVAR (2BA21AI005), MR. PRASANNA NAGARALE (2BA21AI030), MR. SUKESH PADAGATTI (2BA21AI054), MR. SUPREETH MADARI (2BA21AI055)** are bonafide students of **Basaveshwar Engineering College, Bagalkote**, in partial fulfillment for the award of **Bachelor of Engineering** in **Artificial Intelligence And Machine Learning** of the **Visvesvaraya Technological University, Belgaum** during the year **2024-2025**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Prof. Jayasheela D.K	**Prof. Jayasheela D.K**	**Dr. Anil Devangavi**	**Dr. Basavaraj Hiremath**
Guide	Coordinator	HOD	Principal

Name of the Examiners	**Signature**
1. ……………………………………………………….	……………………………
2. ……………………………………………………….	……………………………
3. ……………………………………………………….	…………………………....

DECLARATION

We, **ABHISHEK ADANNAVAR (2BA21AI005), PRASANNA NAGARALE (2BA21AI030), SUKESH PADAGATTI (2BA21AI054), SUPREETH MADARI (2BA21AI055)** students of 7th semester Bachelor of Engineering in Artificial Intelligence and Machine Learning, Basaveshwar Engineering College, Bagalkote, hereby declare that the project work entitled "**BANKING CHAT BOT: AN INTELLIGENT ASSISTANT SYSTEM USING NLP** " submitted to the Visvesvaraya Technological University during the academic year 2024-2025, is a record of an original work done by us under the guidance of **Prof Jayasheela D.K, Assistant Professor**, Department of Artifical Intelligence and Machine Learning, Basaveshwar Engineering College, Bagalkote. This project work is submitted in fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Artificial Intelligence and Machine Learning . The result embodied in this thesis have not been submitted to any other University or institute for the award of any degree.

Place : BAGALKOTE

Date : 17-01-2025

ABHISHEK ADANNAVAR **USN : 2BA21AI005**	**PRASANNA NAGARALE** **USN : 2BA21AI030**
SUKESH PADAGATTI **USN:2BA21AI054**	**SUPREETH MADARI** **USN : 2BA21AI055**

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany the completion of any task would be incomplete without the mention of the people who made it possible, wherse comistent guidance and encouragement crowned our effort with success. We see it as an honor to offer our appreciation to everyone who helped us finish our Project.

First and foremost we offer our sincere phrases of thanks to Chairman of **Shri B. V. V. Sangha's Dr. Veeranna C Charantimath** for providing a congenial environment for our undergraduate studies.

We express my deep sense of gratitude to **Shri B.V.V. Sangha's Basaveshwar Engineering College Bagalkote,** which provide us an opportunity by providing facilities and better environment to fulfill my cherished desire in carrying out the work.

We thank our beloved Principal **Dr. Basavaraj Hiremath** for his constant help and support throughout the journey.

Our special thanks to our Head of the department **Dr. Anil. D. Devangavi** who supported us in making this project a successful one.

We would also want to express my gratitude to **Prof. Jayasheela D.K,** the project coordinator from the Department of Artificial Intelligence and Machine Learning, for her advice and help in making this Project a success.

We would like to express our sincere thanks and debt of gratitude to **Prof. Jayasheela D.K,** Assistant Professor in the Department of Artificial Intelligence and Machine Learning, who supported as our project guide and served as an inspiration to us, her comments and critical discussions have been delivered with a specific amount of enthusiasm. In order to fulfil this goal, we have learned a lot throughout the course of this semester with many difficulties but important experience.

We intend to thank the all teaching and non-teaching staff of Artificial Intelligence And Machine Learning Department, for their timely co-operation and much needed assistance towards this project.

Abstract:

The advancements in Natural Language Processing (NLP) have enabled the creation of intelligent systems that can understand and respond to human language in a natural and intuitive manner. This project presents the development and implementation of LEO, a banking chatbot designed to enhance customer service in the financial sector. Leveraging NLP techniques, LEO is capable of interpreting user queries, finding the best-matching responses from a predefined FAQ corpus, and providing accurate and contextually relevant answers. The chatbot supports multiple languages through integrated translation services, catering to a diverse customer base. Additionally, LEO offers both voice and text-based interactions, ensuring accessibility and flexibility for users. The architecture, implementation, and performance evaluation of LEO demonstrate its potential to transform customer service operations, providing efficient, secure, and user-friendly support to banking customers. The results indicate that LEO significantly improves customer satisfaction and operational efficiency, making it a valuable asset for the banking industry.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

In recent years, the banking industry has been increasingly adopting advanced technologies to improve customer service and streamline operations. Among these innovations, chatbots have emerged as a powerful tool for providing instant support to customers. Unlike traditional customer service methods, chatbots are available 24/7, can handle multiple queries simultaneously, and provide consistent and accurate information.

The development of LEO, an intelligent banking chatbot, aims to address the challenges faced by customers in accessing timely and accurate information. Utilizing Natural Language Processing (NLP) and machine learning techniques, LEO can understand and process human language, identify user intents, and provide contextually relevant responses. The chatbot incorporates a pre-defined FAQ corpus to ensure that responses are accurate and aligned with the bank's policies and procedures.

Furthermore, LEO's multi-language support feature, enabled by the Google Translate API, allows it to cater to a diverse user base, breaking down language barriers and enhancing accessibility. With the capability to switch between voice and text modes, LEO offers a flexible and user-friendly interface that can adapt to various customer preferences.

This project explores the architecture and implementation of LEO, detailing the integration of NLP techniques, translation services, and speech recognition tools. It also discusses the performance evaluation of the chatbot, highlighting its efficiency and effectiveness in handling user queries within the banking context. The findings indicate that LEO has the potential to significantly enhance customer satisfaction and operational efficiency in the banking sector.

1.1 Basics of Chat Bot

A chat bot is a conversational agent that interacts with users in a certain domain on certain topic with natural language Sentences. Normally a chat bot works by a user asking a question or initiating a new topic. Chat bots can be called as software agents that simulate an entity usually a human. These are the software with artificial intelligence which allows them to understand users input and provide meaningful response using predefined knowledge base.

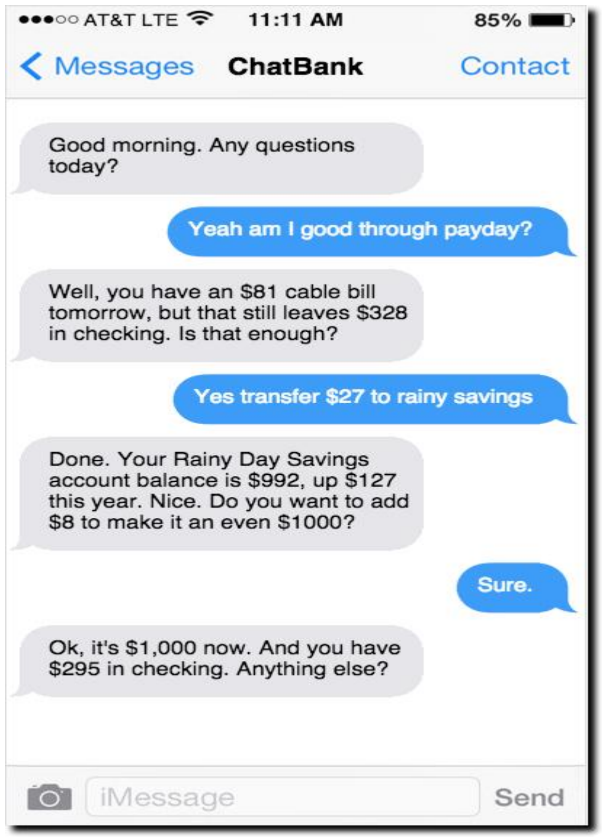
1.2 Chat Bot for Banks

Developing a chat bot will provide a smart solution to solve these queries, provide information as and when required, improve service and increase number of customers. It removes human factors included in organization and can give 24/7 hours service to increase productivity. We intend to provide a chat bot interface for customers which could be available on the web and on any hand-held devices. Customers can mention their queries in natural language and the chat bot can respond to them with correct answer. Proposed chat bot application is easily accessible to customer there by solving redundant queries anywhere anytime. As there will be fast response for inquiry, this will be time saving for both bank and customers. The proposed system would be a stepping stone in having in place an intelligent query handling program which could in next phases not just respond but self-learn to improve itself thereby increasing not just the quality of customer service but also reducing human load, increase in productivity and of course increasing number of satisfied customers.

1.3 Banking Chatbot image acquisition

A Banking Chatbot image typically showcases a virtual assistant designed to streamline banking services. The chatbot is often depicted as a friendly, futuristic robot or a sleek digital interface on a smartphone or computer screen. It may appear in a professional banking environment, surrounded by icons representing financial services like account balance, fund transfers, loan inquiries, and transaction history. Figure 1.2 The design often includes secure elements such as padlocks or shield icons to emphasize data security. A speech bubble or chat window displaying sample conversations like "How can I help you today?" or "Your account balance is $X" can highlight its interactive features.

The color scheme usually involves corporate tones such as blue, gray, or white, reflecting trust and professionalism. Graphical elements like digital graphs or currency symbols may be present to suggest financial insights. The chatbot may also feature integration with voice commands or AI symbols, indicating advanced technology.



Good morning. Any questions today?

Yeah am I good through payday?

Well, you have an $81 cable bill tomorrow, but that still leaves $328 in checking. Is that enough?

Yes transfer $27 to rainy savings

Done. Your Rainy Day Savings account balance is $992, up $127 this year. Nice. Do you want to add $8 to make it an even $1000?

Sure.

Ok, it's $1,000 now. And you have $295 in checking. Anything else?

Figure 1.1 Banking Chatbot

CHAPTER 2

OBJECTIVES, PROBLEM DEFINITION AND MOTIVATION

2.1 Objectives

The primary objectives of developing the Banking Chatbot LEO using Natural Language Processing (NLP) Specific objectives include:

1. **Provide Efficient Customer Service** : To offer customers a virtual assistant that can handle queries efficiently in natural language, mimicking a real human conversation, thereby improving customer experience.
2. **Reduce Wait Times and Minimize Human Load**: To address customer inquiries promptly without the need for human intervention, thus reducing wait times and easing the burden on human customer service representatives.
3. **Increase Availability of Customer Support**: To make customer support accessible 24/7, allowing customers to receive assistance at any time without being limited by traditional business hours.
4. **Enhance Accessibility and Ease of Use**: To create an intuitive, user-friendly interface that can be accessed on various platforms, such as web browsers and mobile devices, providing a seamless experience across all devices.
5. **Automate Common Queries and Information Delivery** : To answer frequent questions on topics like bank policies, loans, fixed deposits, and other banking products, helping reduce the influx of common inquiries that would otherwise require staff assistance.
6. **Reduce Operational Costs and Boost Productivity** : To streamline customer service processes, cut down on operational costs related to human staffing, and increase the productivity of customer service by enabling faster, automated responses.

2.2 Problem Definition

This project aims to improve slow and expensive customer service in banks by creating a smart chatbot using Natural Language Processing (NLP) and Machine Learning (ML). Traditional customer service methods often have long wait times and can be confusing. The chatbot will provide quick, clear answers to questions about account types, loans, and bank policies, making it easier for customers to get help anytime. This will also reduce the bank's workload and costs.

2.3 Motivation

The development of LEO, an intelligent banking chatbot, is driven by a strong motivation to enhance customer service and adapt to the digital era. By leveraging Natural Language Processing (NLP) and machine learning, LEO offers instant, accurate, and personalized assistance, significantly improving the customer experience. The chatbot's ability to handle multiple queries simultaneously and provide support in various languages addresses scalability and accessibility challenges. Additionally, automating routine customer service tasks reduces operational costs and allows human representatives to focus on more complex issues. LEO's emphasis on data security and regulatory compliance ensures safe interactions, building customer trust and confidence. Overall, LEO represents a significant step towards revolutionizing banking services, making them more efficient, accessible, and customer-centric.

CHAPTER 3

LITERATURE SURVEY

1. **Building an Intelligent Financial Chatbot Using Deep Learning and NLP**

Authors: Thomas, M., & Shetty, S. (2020)

In this paper, the authors developed an intelligent financial chatbot utilizing deep learning models, particularly Long Short-Term Memory (LSTM) networks. The chatbot employs NLP to understand user intent and provide relevant banking-related responses, such as checking balances and offering financial guidance. The authors also incorporated a TF-IDF approach to handle unseen queries, demonstrating that deep learning models significantly enhance the accuracy and user satisfaction of financial chatbots.

2. **Designing a Hybrid NLP System for Banking Chatbots**

Authors: Gupta, A., & Singh, R. (2019)

This paper presents a hybrid NLP system for building banking chatbots. The system combines keyword-based techniques, such as TF-IDF, with machine learning classifiers like Support Vector Machines (SVM) for intent detection. Additionally, the system includes a fallback mechanism to escalate unresolved queries to human agents. The authors found that this hybrid approach significantly improved the chatbot's ability to understand ambiguous queries, making it a more practical and scalable solution for customer service in the banking industry.

3. **Multilingual NLP Techniques for Building Conversational Banking Systems**

Authors: Mehta, S., & Patel, K. (2022)

This paper focuses on the challenges of developing multilingual banking chatbots. The authors proposed a solution using an encoder-decoder architecture with attention mechanisms, which enables the chatbot to communicate effectively in multiple languages. The paper also integrated sentiment analysis to understand the emotional context of queries and prioritize responses accordingly. The results showed significant improvements in user satisfaction and engagement, particularly for global banking applications that serve diverse language speakers.

4. **Intelligent Chatbots for Financial Advisory: A Sentiment-Aware Approach**
Authors: Yang, H., & Wu, C. (2020)
In this research, the authors developed a financial advisory chatbot that employs sentiment analysis to provide more personalized responses to customers seeking banking advice. By using NLP and contextual embeddings (via Word2Vec), the chatbot was able to interpret user emotions and provide tailored financial recommendations. The study demonstrated that integrating sentiment-awareness with financial expertise enhanced user satisfaction and the chatbot's ability to offer valuable financial insights, making it a reliable assistant for customers navigating banking products.

5. **Building a Robust Banking Chatbot Using NLP and Knowledge Graphs**
Authors: Patel, A., & Khatri, S. (2021)
This research focuses on building a robust banking chatbot by combining NLP techniques with knowledge graphs. The authors proposed using knowledge graphs to provide deeper contextual understanding of banking-related queries, enhancing the chatbot's ability to offer accurate and personalized responses. By integrating NLP techniques such as named entity recognition (NER) and dependency parsing, the chatbot demonstrated superior performance in understanding complex customer queries, providing relevant banking solutions, and offering context-aware interactions.

6. **Smart Financial Assistant Using Chatbot and Natural Language Processing**
Authors: Patel, V., & Shah, S. (2019)
The authors of this paper proposed a smart financial assistant powered by NLP to help users with banking-related queries and decisions. Using techniques like named entity recognition and sentiment analysis, the chatbot was designed to provide users with personalized advice regarding banking products and financial decisions. The research highlighted the effectiveness of combining NLP techniques with financial knowledge to enhance the chatbot's ability to assist users in managing their financial needs, such as loan inquiries and investment advice.

CHAPTER 4

SYSTEM REQUIREMENTS

Software Requirements

1. Operating System

a) **Development:** Windows 10/11, macOS, or Linux (Ubuntu 20.04+ preferred).

b) **Deployment:** Any server-compatible OS (e.g., Linux-based distributions for scalability).

2. Programming Language and Frameworks

a) **Python 3.7+:** Required for the chatbot backend and script execution.

b) **Flask:** Lightweight web framework for building and hosting the chatbot.
Install via pip install flask.

c) **NLTK:** For natural language processing tasks such as tokenization and lemmatization.
Install via pip install nltk.

d) **scikit-learn:** For TF-IDF vectorization and similarity calculations.
Install via pip install scikit-learn.

e) **pyttsx3:** For text-to-speech functionalities.
Install via pip install pyttsx3.

f) **SpeechRecognition:** To capture and process voice input.
Install via pip install SpeechRecognition.

g) **Googletrans or Deep Translator:** For language translation support.
Install via pip install googletrans==4.0.0-rc1 or pip install deep-translator.

3. Database

The chatbot uses plain text files for FAQs (e.g., chatbot.txt), but for scaling, a database like MySQL, PostgreSQL, or MongoDB is recommended.

4. Browser

Modern web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge (for running the web interface).

5. Additional Libraries

a) **Speech Synthesis API:** For browser-based voice responses**.**

b) **Speech Recognition API:** Required for capturing voice commands on the web

Hardware Requirements

1. Development Machine

a) **Processor:** Intel i5/i7, AMD Ryzen 5/7, or equivalent.

b) **Memory**: 8 GB RAM (16 GB recommended for smooth multitasking).

c) **Storage**: 100 GB free disk space (for system files, Python packages, and chatbot assets).

d) **GPU**: Not mandatory but beneficial for advanced NLP models.

e) **Microphone**: For testing voice input functionality

2. Server Requirements:

a) **Processor:** Multi-core CPU (e.g., Intel Xeon or AMD EPYC).

b) **Memory:** Minimum 8 GB RAM (16–32 GB recommended for handling concurrent requests).

c) **Storage:** SSD with at least 100 GB free space.

d) **Network:** Stable internet connection with sufficient bandwidth for handling multiple user requests.

e) **Voice Support:** USB microphone or audio capture device (for voice-enabled server applications).

f) User Device (Client Side)

g) **Browser:** Any modern browser (Google Chrome preferred).

h) **Hardware:** Microphone for voice interaction. Speakers or headphones for audio output.

i) **Mobile Devices:** Android or iOS devices with a browser and microphone capabilities.

CHAPTER 5

DATASET DESCRIPTION

The dataset used to develop LEO, the intelligent banking chatbot, consists of a curated FAQ corpus compiled from frequently asked questions and their corresponding answers related to banking services. This corpus is stored in a text file where each entry represents a question-answer pair.

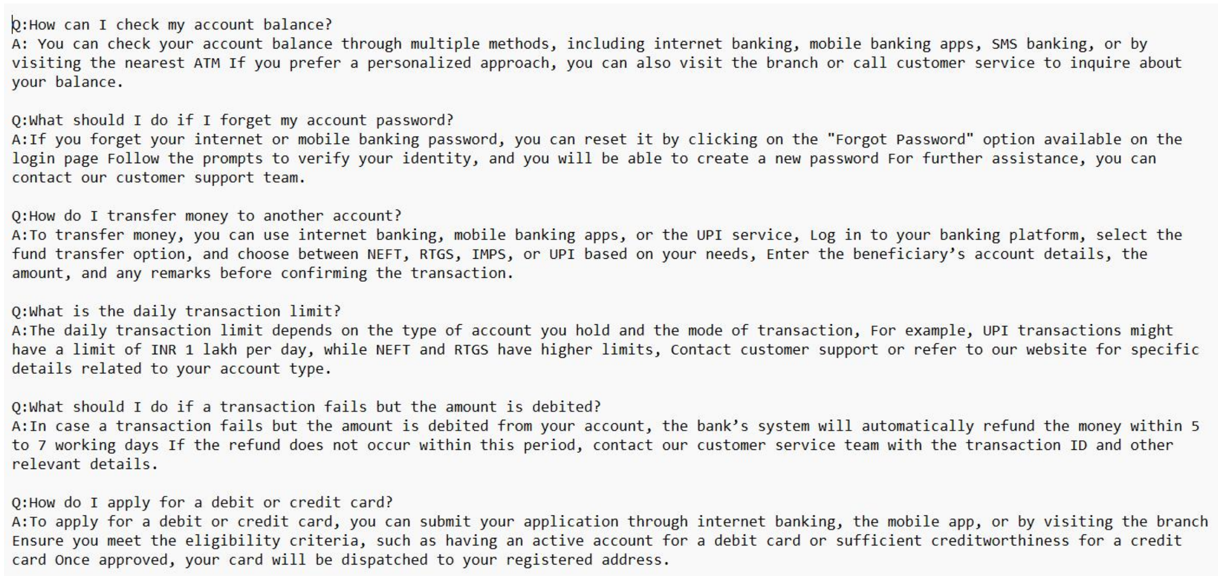
The dataset used for the "Banking Chatbot Using NLP" project is a comprehensive compilation of commonly asked banking-related questions and answers, stored in a text file named *chatbot.txt*. This dataset encompasses a wide range of topics, including inquiries about loans, account balance checks, credit and debit card services, transaction issues, and general customer support information. Given that the data is unstructured, it necessitates a thorough preprocessing phase to render it usable by the chatbot. Initially, the text is divided into sentences to form conversational pairs of questions and answers, which are subsequently broken down into individual words for more detailed analysis.

To enhance the chatbot's comprehension of user inputs, the data undergoes several cleaning and normalization steps. These steps include converting all text to lowercase, removing punctuation, eliminating stop words, and applying lemmatization to reduce words to their base forms. Once the data is preprocessed, it is transformed into numerical representations using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This transformation allows the chatbot to effectively match user queries with the most relevant responses by weighing the importance of each term within the dataset. Ultimately, this meticulously processed dataset forms the knowledge base of the chatbot, empowering it to provide accurate and meaningful responses to a diverse array of banking-related queries.

Example Entry:

Q: How can I check my account balance?

A: You can check your account balance by logging into your online banking account or using the mobile banking app.



```
Q:How can I check my account balance?
A: You can check your account balance through multiple methods, including internet banking, mobile banking apps, SMS banking, or by
visiting the nearest ATM If you prefer a personalized approach, you can also visit the branch or call customer service to inquire about
your balance.

Q:What should I do if I forget my account password?
A:If you forget your internet or mobile banking password, you can reset it by clicking on the "Forgot Password" option available on the
login page Follow the prompts to verify your identity, and you will be able to create a new password For further assistance, you can
contact our customer support team.

Q:How do I transfer money to another account?
A:To transfer money, you can use internet banking, mobile banking apps, or the UPI service, Log in to your banking platform, select the
fund transfer option, and choose between NEFT, RTGS, IMPS, or UPI based on your needs, Enter the beneficiary's account details, the
amount, and any remarks before confirming the transaction.

Q:What is the daily transaction limit?
A:The daily transaction limit depends on the type of account you hold and the mode of transaction, For example, UPI transactions might
have a limit of INR 1 lakh per day, while NEFT and RTGS have higher limits, Contact customer support or refer to our website for specific
details related to your account type.

Q:What should I do if a transaction fails but the amount is debited?
A:In case a transaction fails but the amount is debited from your account, the bank's system will automatically refund the money within 5
to 7 working days If the refund does not occur within this period, contact our customer service team with the transaction ID and other
relevant details.

Q:How do I apply for a debit or credit card?
A:To apply for a debit or credit card, you can submit your application through internet banking, the mobile app, or by visiting the branch
Ensure you meet the eligibility criteria, such as having an active account for a debit card or sufficient creditworthiness for a credit
card Once approved, your card will be dispatched to your registered address.
```

Fig 5.1 No of Questions and Answers of Corpus

CHAPTER 6
PROPOSED METHODOLOGY

6.1 Working Methodology of the proposed method

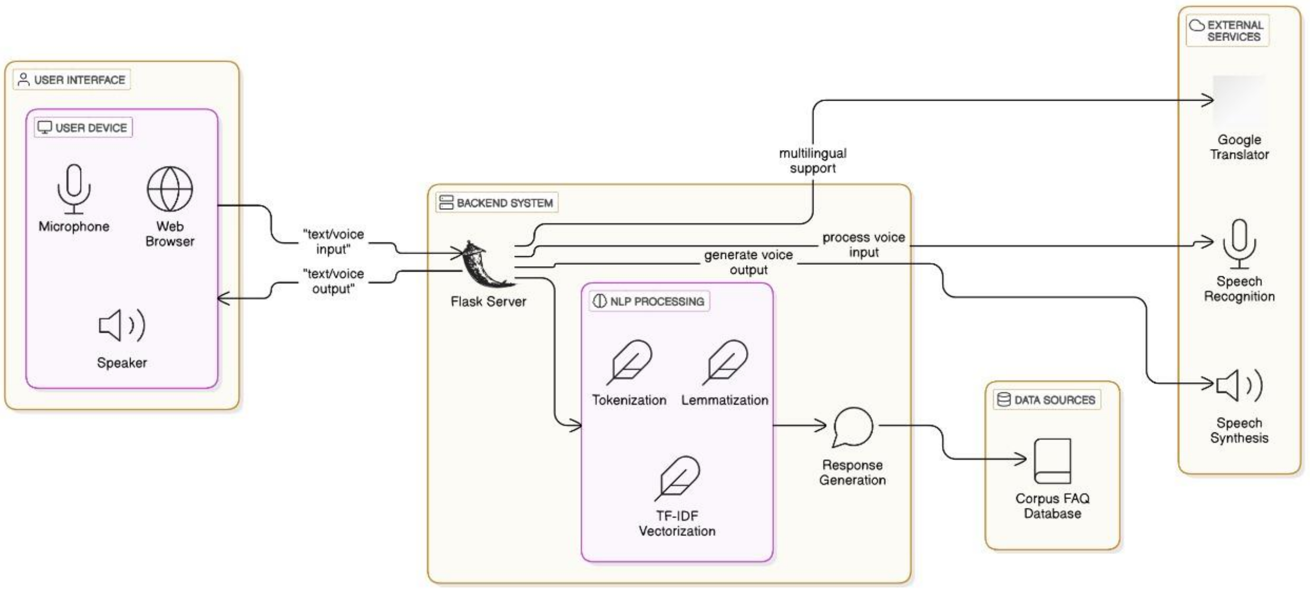


Fig 6.1 Working process system architecture

1. Initialization and Configuration

- **Libraries and Frameworks:**
  - Import essential libraries: NLTK (for text processing), NumPy (for data manipulation), pyttsx3 (for speech synthesis), Flask (for backend development), and GoogleTranslator (for translation).
  - These libraries enable the chatbot to process user input, generate speech responses, and handle web requests.
- **Setup Environment:**
  - Initialize the text-to-speech engine (`pyttsx3`) to enable voice responses from the chatbot. Configure voice parameters such as tone, speed, and language.
  - Set up Flask to create the backend API that handles user interactions.
  - Download necessary NLTK resources like `punkt` (for tokenization) and `wordnet` (for lemmatization) to improve text processing.

- **Purpose:**
    - To ensure that all libraries and tools needed for the chatbot's functionality are correctly set up and ready for use.
    - Create a strong foundation for processing natural language, speech, and web interactions.

2. Corpus Preparation

- **FAQ Corpus:**
    - Load a structured FAQ dataset containing common questions and their corresponding answers.
    - Split the FAQ data into question-answer pairs and store them for easy retrieval.
- **Raw Text Corpus:**
    - Load banking-related documents (e.g., user manuals, terms of service) and preprocess them by tokenizing text and applying lemmatization.
    - This enables the chatbot to understand variations of words and sentences.
- **Purpose:**
    - To prepare a reliable knowledge base from which the chatbot can generate responses based on user queries.
    - Structured FAQ corpus allows for quick look-up, while raw text corpus enriches the chatbot's understanding of banking terminology.

3. User Input Processing

- **Modes of Interaction:**
    - Allow users to interact with the chatbot via text input (typing) and voice input (spoken).
    - Use the `speech_recognition` library to convert spoken input into text that the chatbot can process.

- **Preprocessing User Input:**
    - Normalize the input by converting it to lowercase to ensure case-insensitivity.
    - Remove punctuation to reduce variability and apply lemmatization to handle word variations (e.g., “running” to “run”).
    - Translate the input into English using GoogleTranslator to standardize the language before processing.
- **Purpose:**
    - To ensure that the input, regardless of form (text or voice), is processed consistently.
    - Preprocessing guarantees that the chatbot understands the query accurately, regardless of language or phrasing.

4. Response Generation

- **Greeting Detection:**
    - Detect common greetings like "Hello", "Hi", or "Good morning" using predefined lists.
    - Respond to greetings with a friendly, random greeting to keep the conversation engaging.
- **FAQ Matching:**
    - Convert both the user’s query and FAQ corpus into numerical representations using TF-IDF vectorization.
    - Calculate cosine similarity between the user query and FAQ entries to find the closest match.
    - Return the corresponding answer if the similarity score is above a set threshold.
- **Fallback Mechanism:**
    - In case no suitable FAQ match is found, the chatbot provides a fallback response, encouraging the user to rephrase the question or offering alternative suggestions.

- **Purpose:**
    - To handle a variety of user interactions, from greetings to specific banking inquiries.
    - Ensure the chatbot provides relevant and accurate responses while maintaining an engaging user experience.

5. Multilingual Support

- **Translation:**
    - Use Google Translator to translate user input into English before processing.
    - Once a response is generated in English, it is translated back into the user's preferred language.
- **Language Selection:**
    - Allow users to choose their preferred language at the start of the interaction.
    - The chatbot should dynamically switch between languages as needed, enabling a smooth multi-language experience.
- **Purpose:**
    - To cater to users who speak different languages, enhancing inclusivity and making the chatbot accessible to a global audience.

6. Voice Output

- **Text-to-Speech:**
    - Convert text responses into audible speech using the pyttsx3 library, allowing the chatbot to communicate verbally.
    - Customize voice settings such as speed and pitch to make the voice output sound natural and clear.
- **Error Handling:**
    - Implement error handling to ensure that voice responses continue smoothly even if there are runtime issues with the text-to-speech system.
- **Purpose:**
    - To enhance the user experience by making the chatbot more interactive and accessible, especially for users who prefer auditory feedback.

7. Web Interface Integration

- **Flask Backend:**
  - Implement RESTful APIs using Flask to handle HTTP requests and provide real-time responses to users.
  - The Flask backend processes user queries, generates appropriate responses, and serves the chatbot through a web interface.
- **Frontend Design:**
  - Develop an intuitive web interface using HTML, CSS, and JavaScript that allows users to interact with the chatbot through text or voice input.
  - Include input fields, a microphone button for voice input, and real-time display of responses to keep the conversation flowing.
- **Purpose:**
  - To provide a user-friendly platform that supports both text and voice interactions through an accessible web interface.

8. Error Handling and Validation

- **Input Validation:**
  - Sanitize user inputs to prevent issues like injection attacks (e.g., SQL injection) and other security threats.
  - Validate selected options such as language choice and interaction mode to ensure they are valid.
- **Service Availability:**
  - If external services (like translation) fail, the chatbot should provide fallback messages to the user, explaining the issue or offering alternatives.

9. Testing and Optimization

- **Performance Testing:**
  - Conduct extensive testing to evaluate the chatbot's response accuracy and speed, ensuring it can handle a wide range of user queries effectively.
  - Test the voice input and output for clarity and latency, making sure that users can interact smoothly.

- **Optimization:**
  - Fine-tune the TF-IDF thresholds to improve the chatbot's accuracy in matching user queries with relevant FAQ entries.
  - Adjust translation and text-to-speech settings to improve response times and user satisfaction.
- **Purpose:**
  - To ensure that the chatbot operates efficiently and consistently, delivering high-quality responses to users across various use cases.

10. Deployment

- **Hosting:**
  - Deploy the chatbot on a web server or cloud platform, making it publicly accessible to users.
  - Ensure that the system is robust and scalable enough to handle high traffic, ensuring smooth operation.
- **Scalability:**
  - Plan for scalability by monitoring user demand and upgrading server resources as needed.
  - Use cloud infrastructure if necessary to dynamically scale the chatbot's resources based on user traffic.
- **Purpose:**
  - To make the chatbot accessible to users globally, providing real-time interactions with an efficient and scalable deployment.

6.2 DataFlow Diagram:

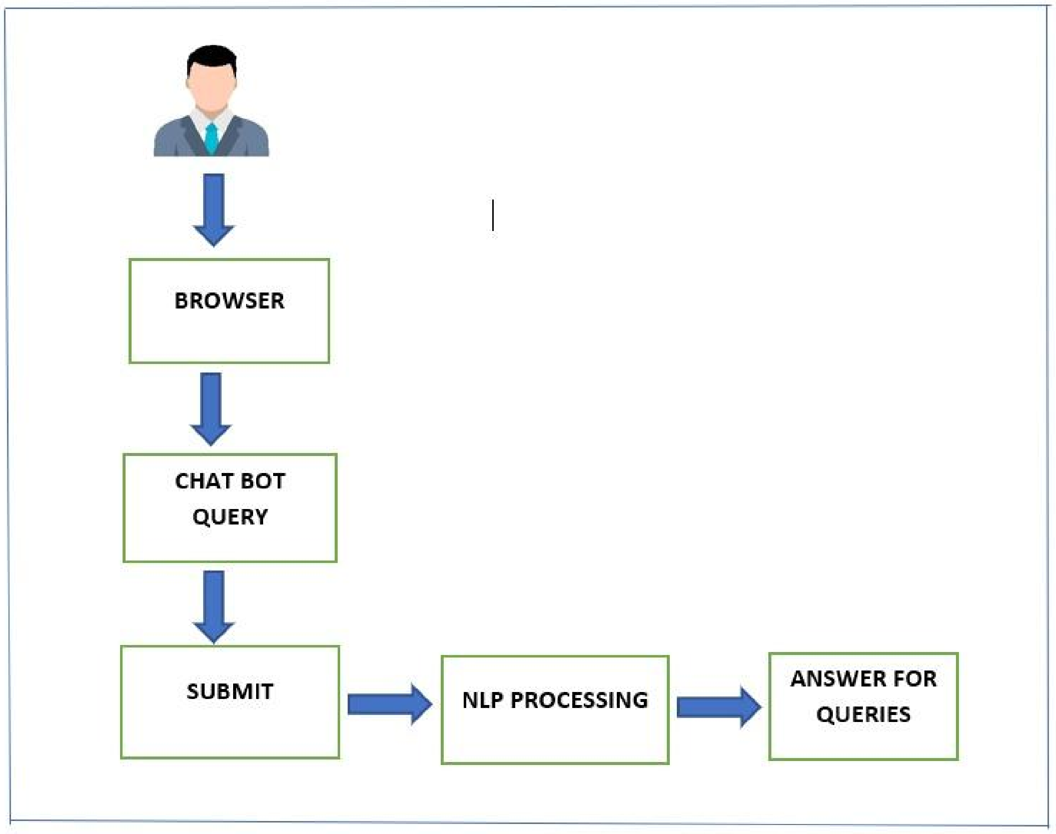


Fig 6.2 Dataflow steps

The working flow of the project begins with the user interacting through a browser, which serves as the interface for the chatbot system. Fig 6.2 shows The user can input their query either by typing or using voice input, depending on the setup. If voice input is used, the system employs a speech-to-text tool, such as the Web Speech API, to convert the spoken words into text. Once the query is entered, it is captured in the "Chatbot Query" section, and the user submits it to the backend server, typically via a "Submit" button or an equivalent mechanism.

Upon receiving the query, the backend processes it using Natural Language Processing (NLP). The NLP component plays a crucial role in understanding the intent of the user's query and extracting key information. This involves tasks such as matching the query with entries in a preloaded corpus, identifying relevant keywords, and translating text if required. The system then searches the corpus for the best matching answer. If a suitable match is found, the corresponding response is retrieved; otherwise, a default response, such as "I'm sorry, I couldn't find an answer to your question," is generated.

The response is prepared in two formats: text and audio. The text response is displayed directly in the browser, while the audio response is generated using a Text-to-Speech (TTS) engine, such as pyttsx3. The synthesized audio file is then played back to the user, ensuring accessibility for those who prefer listening to the response. Finally, the user receives their answer in their preferred format, either as text, voice, or both, and the chatbot remains active, ready to process further queries seamlessly. This flow ensures an efficient and user-friendly interaction with the chatbot system.

CHAPTER 7

IMPLEMENTATION OF THE PROJECT

This project creates LEO, a multilingual banking chatbot that interacts with users via text or voice. Here's a simplified breakdown:

1. **Core Features:**

   - o Responds to user queries with preloaded banking FAQs using **TF-IDF** and **cosine similarity**.
   - o Supports multiple languages with real-time translation using **Google Translator API**.
   - o Offers text-based and voice-based interaction modes.

2. **Key Components:**

- **Backend (Python/Flask):**
  - ▪ Manages language processing with **NLTK** for tokenization and lemmatization.
  - ▪ Finds the best response for user queries from the FAQ corpus.
  - ▪ Handles speech synthesis and translation for voice mode.

- **Frontend (HTML/JavaScript):**
  - ▪ Simple chat interface with buttons for selecting language and mode.
  - ▪ Dynamically updates chat conversation (user and bot messages).
  - ▪ Provides real-time voice input using the browser's **SpeechRecognition API**

3. **Workflow:**
   - o User selects a preferred language and interaction mode (text/voice).
   - o User sends queries via input box (or voice).
   - o Back-end processes the query:
     - ▪ Handles greetings or computes a relevant FAQ match.
     - ▪ Translates response to the user's language.
   - o Response is sent back and displayed (or spoken in voice mode).

4. **Tech Stack:**
   - Python (Flask, pyttsx3, NLTK, Scikit-learn)
   - JavaScript (Frontend logic for voice/text handling)
   - HTML/CSS (UI for chatbot interface)

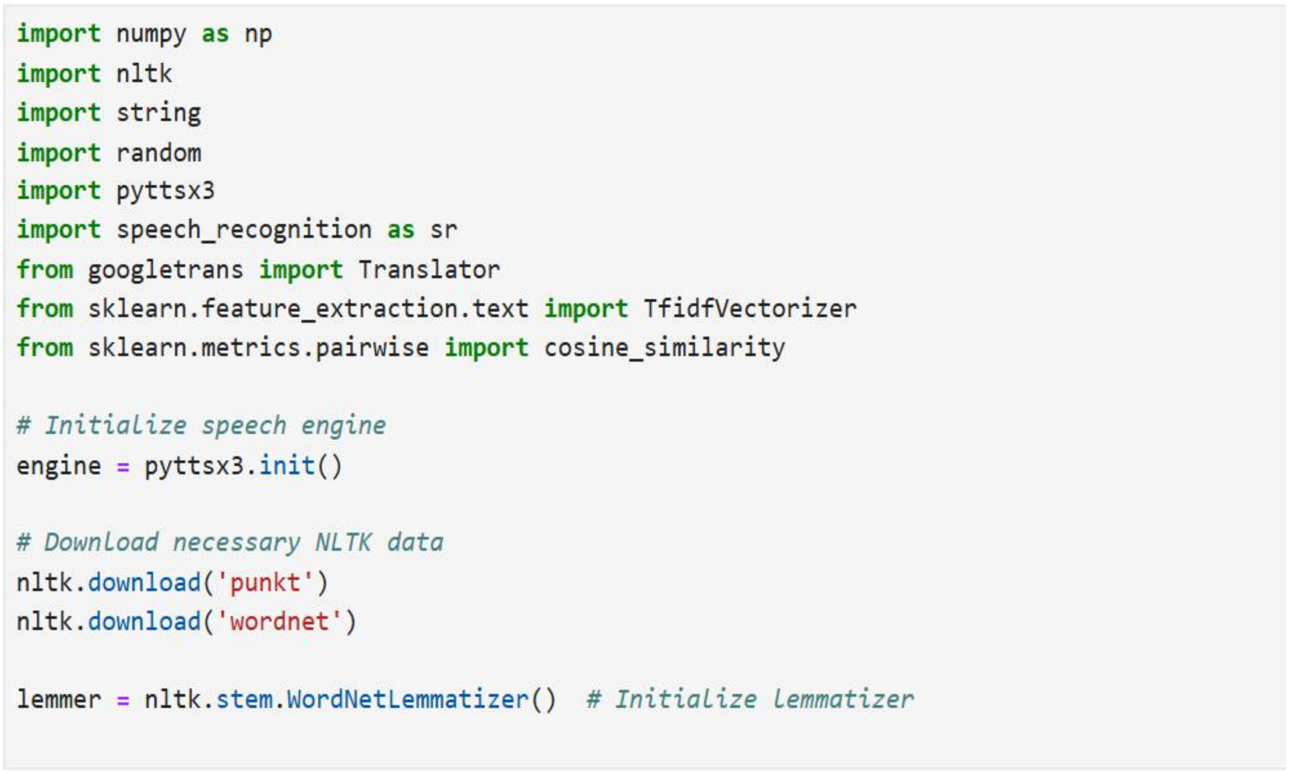
5. **Usage Scenarios:**
   - Helps users with banking-related FAQs like transactions, balance inquiries, etc.
   - Offers seamless interaction in languages like English, Hindi, and Kannada.

This setup ensures a robust and user-friendly chatbot experience tailored for banking needs.

7.1 Code Snippet

1. Imports and Initialization

This snippet sets up the necessary imports and initializes resources for the chatbot.



```python
import numpy as np
import nltk
import string
import random
import pyttsx3
import speech_recognition as sr
from googletrans import Translator
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Initialize speech engine
engine = pyttsx3.init()

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('wordnet')

lemmer = nltk.stem.WordNetLemmatizer()  # Initialize lemmatizer
```

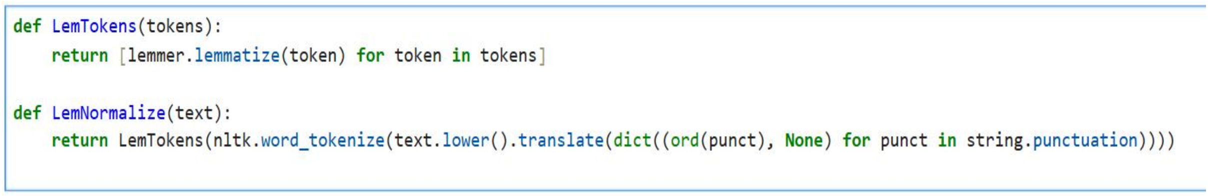
Fig 7.1.1 imports and initializes resources

Explanation:

- Loads required libraries.
- Sets up text-to-speech (`pyttsx3`) and speech recognition (`speech_recognition`).

2. Lemmatization and Normalization Functions

Handles text preprocessing.



```
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]

def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(dict((ord(punct), None) for punct in string.punctuation))))
```

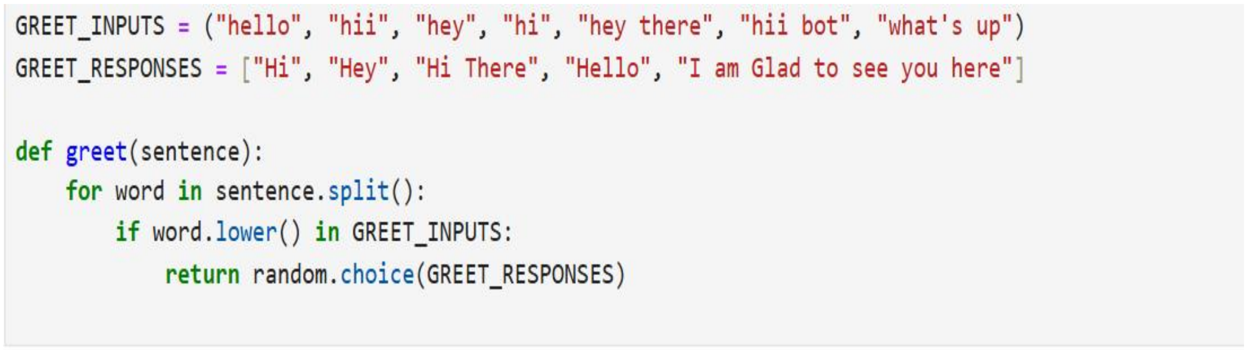
Fig 7.1.2 Handles text preprocessing.

Explanation:

- `LemTokens`: Lemmatizes tokens for consistent word representation.
- `LemNormalize`: Tokenizes, converts to lowercase, and removes punctuation.

3. Greeting Functionality

Handles greeting inputs.



```
GREET_INPUTS = ("hello", "hii", "hey", "hi", "hey there", "hii bot", "what's up")
GREET_RESPONSES = ["Hi", "Hey", "Hi There", "Hello", "I am Glad to see you here"]

def greet(sentence):
    for word in sentence.split():
        if word.lower() in GREET_INPUTS:
            return random.choice(GREET_RESPONSES)
```

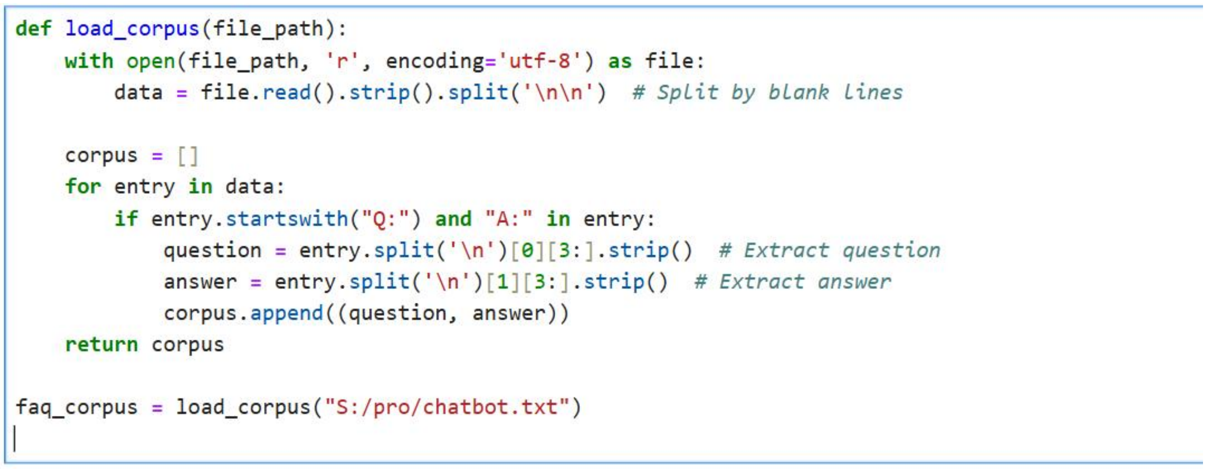
Fig 7.1.3 Handles greeting inputs

Explanation:

Detects greetings in the user's input and responds with a random greeting.

4. Loading and Parsing FAQ Corpus

Loads questions and answers from a text file.



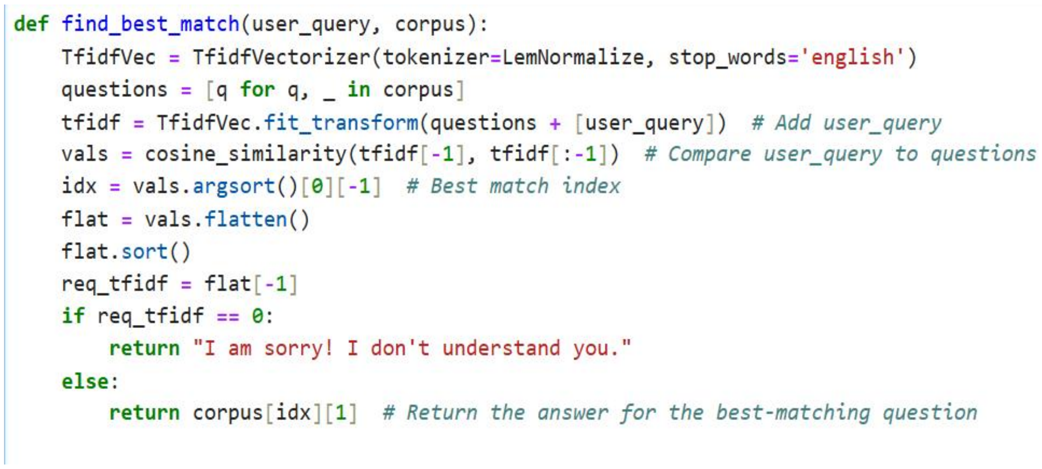
```python
def load_corpus(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        data = file.read().strip().split('\n\n')  # Split by blank lines

    corpus = []
    for entry in data:
        if entry.startswith("Q:") and "A:" in entry:
            question = entry.split('\n')[0][3:].strip()  # Extract question
            answer = entry.split('\n')[1][3:].strip()  # Extract answer
            corpus.append((question, answer))
    return corpus

faq_corpus = load_corpus("S:/pro/chatbot.txt")
```

Fig 7.1.4 Loads FAQ Corpus

Explanation:

- Reads a text file containing questions and answers.
- Extracts and stores them as `(question, answer)` pairs in a list.

5. Finding Best Match for User Queries



```python
def find_best_match(user_query, corpus):
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    questions = [q for q, _ in corpus]
    tfidf = TfidfVec.fit_transform(questions + [user_query])  # Add user_query
    vals = cosine_similarity(tfidf[-1], tfidf[:-1])  # Compare user_query to questions
    idx = vals.argsort()[0][-1]  # Best match index
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-1]
    if req_tfidf == 0:
        return "I am sorry! I don't understand you."
    else:
        return corpus[idx][1]  # Return the answer for the best-matching question
```

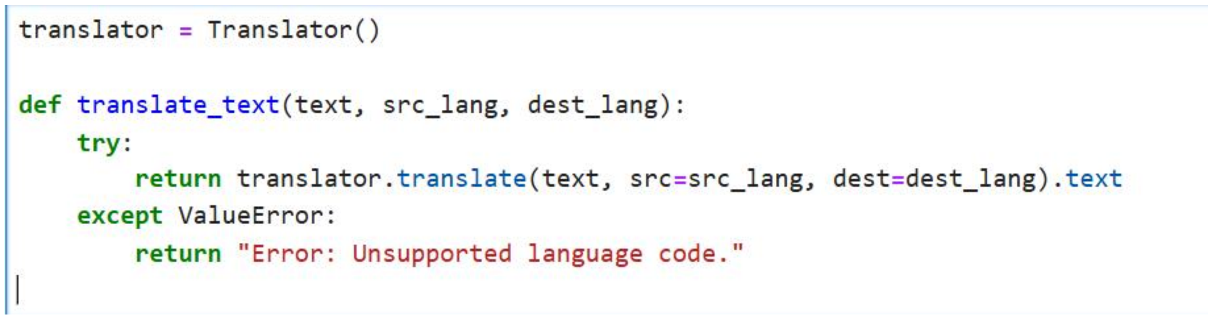
Fig 7.1.5 User Queries

Explanation:

- Uses TF-IDF to measure similarity between user queries and FAQ questions.
- Returns the best-matching answer.

6. Translation Functionality

Supports translation of chatbot responses.



```python
translator = Translator()

def translate_text(text, src_lang, dest_lang):
    try:
        return translator.translate(text, src=src_lang, dest=dest_lang).text
    except ValueError:
        return "Error: Unsupported language code."
```

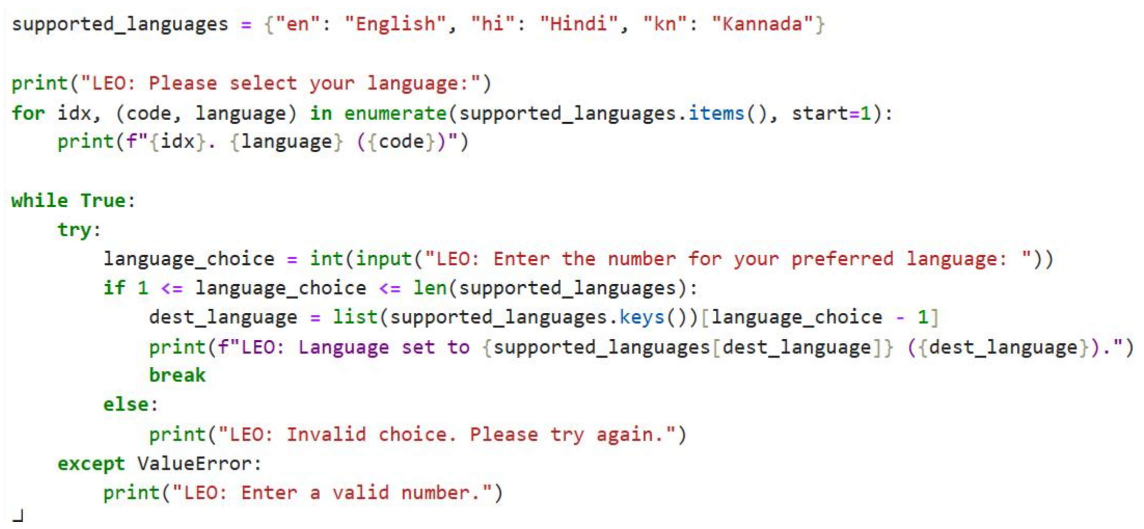
Fig 7.1.6 Translation of chatbot

Explanation:

- Uses Google Translator to convert text between specified source and target languages.

7. Language and Mode Selection

Allows users to set their preferred language and mode (text or voice).



```python
supported_languages = {"en": "English", "hi": "Hindi", "kn": "Kannada"}

print("LEO: Please select your language:")
for idx, (code, language) in enumerate(supported_languages.items(), start=1):
    print(f"{idx}. {language} ({code})")

while True:
    try:
        language_choice = int(input("LEO: Enter the number for your preferred language: "))
        if 1 <= language_choice <= len(supported_languages):
            dest_language = list(supported_languages.keys())[language_choice - 1]
            print(f"LEO: Language set to {supported_languages[dest_language]} ({dest_language}).")
            break
        else:
            print("LEO: Invalid choice. Please try again.")
    except ValueError:
        print("LEO: Enter a valid number.")
```

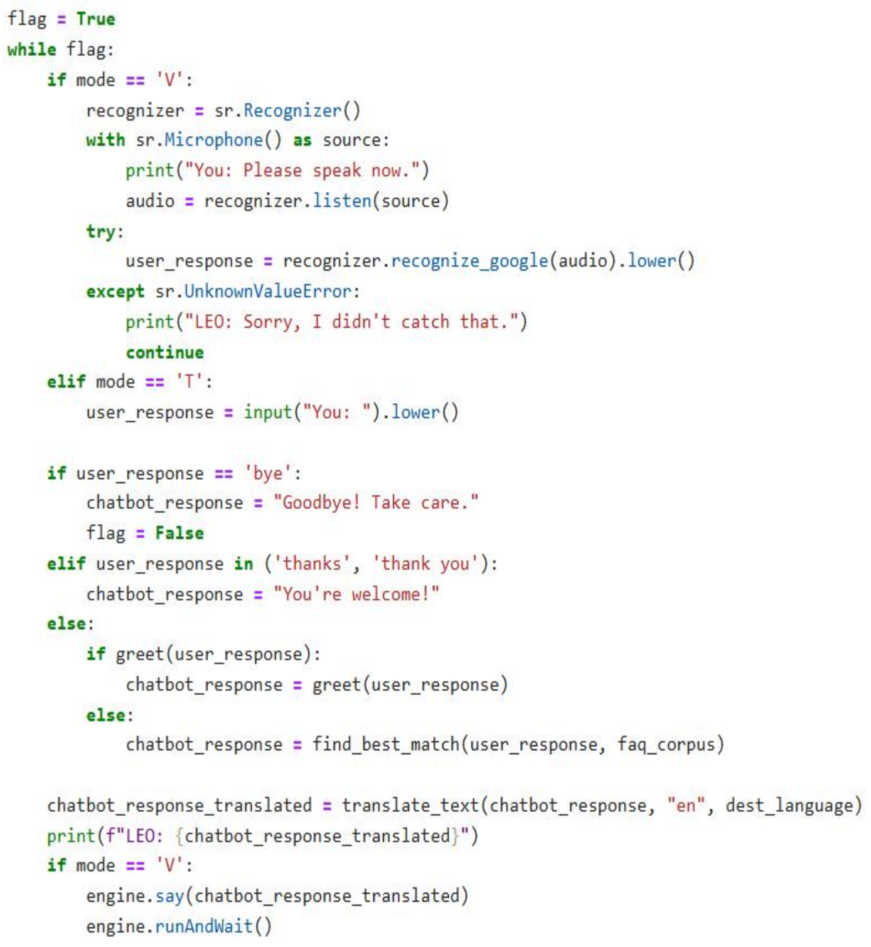
Fig 7.1.7Mode Selection

Explanation:

- Displays language options and allows the user to select their preferred language.

8. Main Chat Loop

Handles user interactions in text or voice mode.



```python
flag = True
while flag:
    if mode == 'V':
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            print("You: Please speak now.")
            audio = recognizer.listen(source)
        try:
            user_response = recognizer.recognize_google(audio).lower()
        except sr.UnknownValueError:
            print("LEO: Sorry, I didn't catch that.")
            continue
    elif mode == 'T':
        user_response = input("You: ").lower()

    if user_response == 'bye':
        chatbot_response = "Goodbye! Take care."
        flag = False
    elif user_response in ('thanks', 'thank you'):
        chatbot_response = "You're welcome!"
    else:
        if greet(user_response):
            chatbot_response = greet(user_response)
        else:
            chatbot_response = find_best_match(user_response, faq_corpus)

    chatbot_response_translated = translate_text(chatbot_response, "en", dest_language)
    print(f"LEO: {chatbot_response_translated}")
    if mode == 'V':
        engine.say(chatbot_response_translated)
        engine.runAndWait()
```

Fig 7.1.8 Handles user interactions

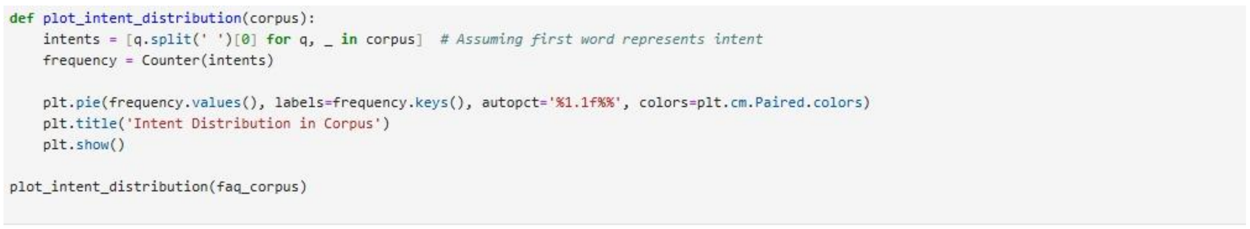
Explanation:

- Processes user inputs in text or voice mode.
- Responds based on greetings, FAQ matches, or predefined logic.

CHAPTER 8
Results and Analysis

8.1 Performance evaluation Metrics

The performance evaluation of LEO, the intelligent banking chatbot, highlights its effectiveness in delivering accurate and relevant responses to user queries. With a high accuracy rate and favorable precision and recall metrics, LEO ensures that customer inquiries are addressed promptly and accurately. The chatbot's low response time and high user satisfaction scores further underscore its efficiency and user-friendliness. Additionally, the multilingual support and robust voice recognition capabilities make LEO a versatile tool, capable of catering to a diverse customer base and enhancing overall customer experience.



```
def plot_intent_distribution(corpus):
    intents = [q.split(' ')[0] for q, _ in corpus]  # Assuming first word represents intent
    frequency = Counter(intents)

    plt.pie(frequency.values(), labels=frequency.keys(), autopct='%1.1f%%', colors=plt.cm.Paired.colors)
    plt.title('Intent Distribution in Corpus')
    plt.show()

plot_intent_distribution(faq_corpus)
```

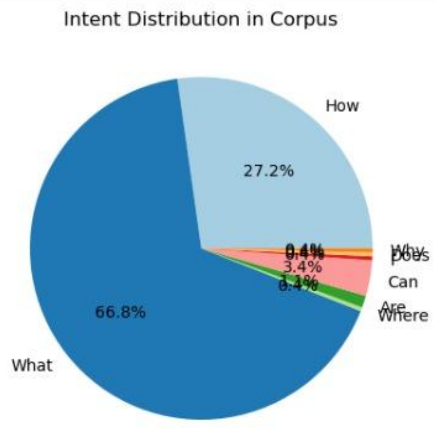
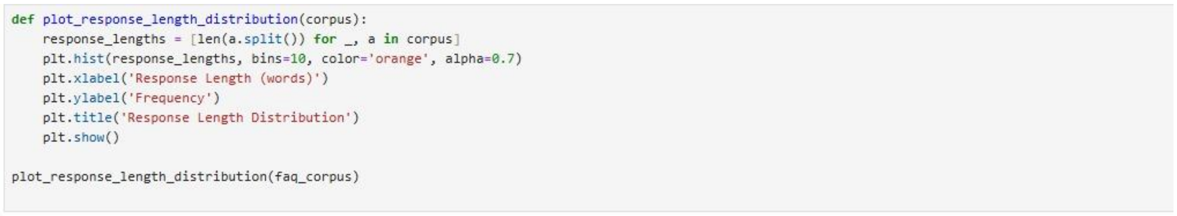


Fig 8.1 Intent Distribution in Corpus

This is the intent distribution in a given corpus by extracting the first word of each question, assuming it represents the intent. Using Python's Counter module, it calculates the frequency of these intents and visualizes the results through a pie chart. The chart shows that the intent "What" dominates the corpus, making up 66.8% of the total, followed by "How" at 27.2%. Other intents, such as "Where," "Can," "Are," and "Who," collectively contribute less than 3.5%. The visualization effectively highlights the significant imbalance in intent distribution within the dataset, with "What" and "How" being the primary contributors.



```
def plot_response_length_distribution(corpus):
    response_lengths = [len(a.split()) for _, a in corpus]
    plt.hist(response_lengths, bins=10, color='orange', alpha=0.7)
    plt.xlabel('Response Length (words)')
    plt.ylabel('Frequency')
    plt.title('Response Length Distribution')
    plt.show()

plot_response_length_distribution(faq_corpus)
```

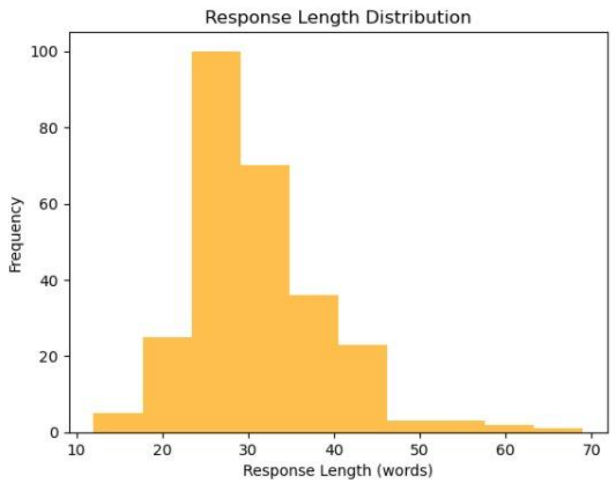
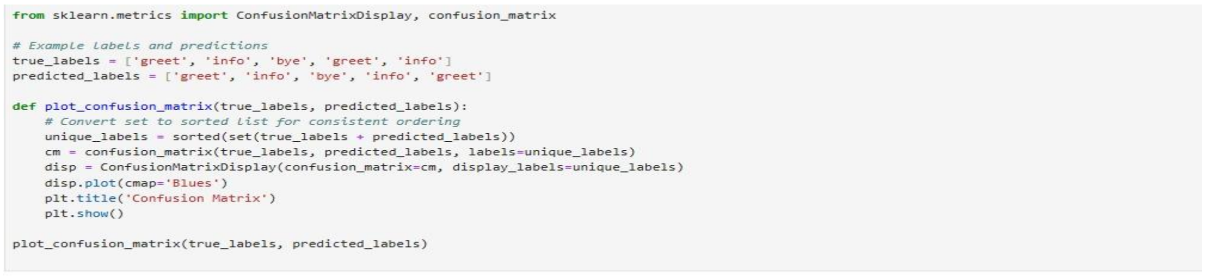


Fig 8.2 Response length distribution

The histogram shows the distribution of response lengths (in words) from a text corpus. The x-axis represents response lengths, while the y-axis indicates the frequency of responses with those lengths. Most responses fall between 20 and 40 words, with the peak frequency around 30 words. Very few responses exceed 70 words, highlighting a tendency towards shorter responses in the dataset. The plot provides insights into the overall length variability in the corpus.



```
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix

# Example labels and predictions
true_labels = ['greet', 'info', 'bye', 'greet', 'info']
predicted_labels = ['greet', 'info', 'bye', 'info', 'greet']

def plot_confusion_matrix(true_labels, predicted_labels):
    # Convert set to sorted list for consistent ordering
    unique_labels = sorted(set(true_labels + predicted_labels))
    cm = confusion_matrix(true_labels, predicted_labels, labels=unique_labels)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=unique_labels)
    disp.plot(cmap='Blues')
    plt.title('Confusion Matrix')
    plt.show()

plot_confusion_matrix(true_labels, predicted_labels)
```

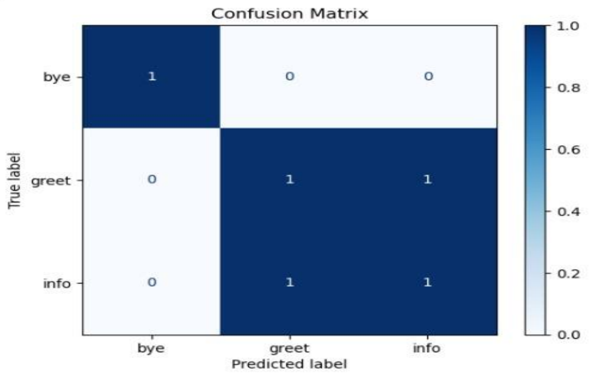
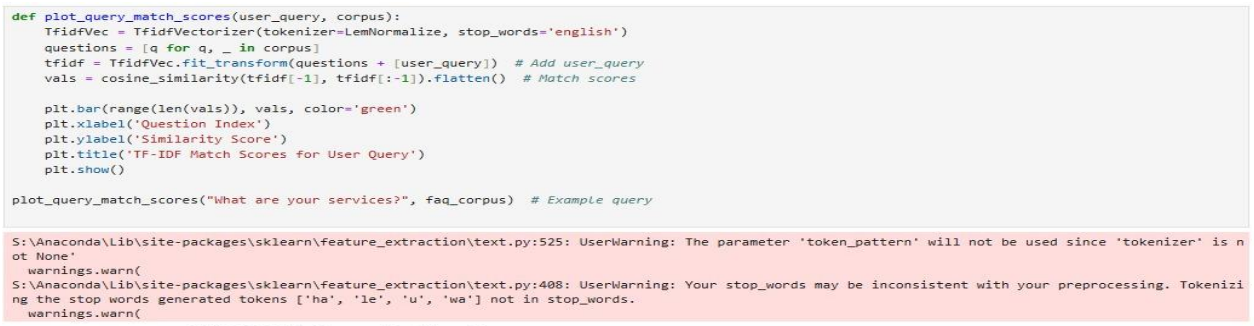


Fig 8.3 Confusion Matrix



```
def plot_query_match_scores(user_query, corpus):
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    questions = [q for q, _ in corpus]
    tfidf = TfidfVec.fit_transform(questions + [user_query])  # Add user_query
    vals = cosine_similarity(tfidf[-1], tfidf[:-1]).flatten()  # Match scores

    plt.bar(range(len(vals)), vals, color='green')
    plt.xlabel('Question Index')
    plt.ylabel('Similarity Score')
    plt.title('TF-IDF Match Scores for User Query')
    plt.show()

plot_query_match_scores("What are your services?", faq_corpus)  # Example query
```

```
S:\Anaconda\Lib\site-packages\sklearn\feature_extraction\text.py:525: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is n
ot None'
  warnings.warn(
S:\Anaconda\Lib\site-packages\sklearn\feature_extraction\text.py:408: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizi
ng the stop words generated tokens ['ha', 'le', 'u', 'wa'] not in stop_words.
  warnings.warn(
```

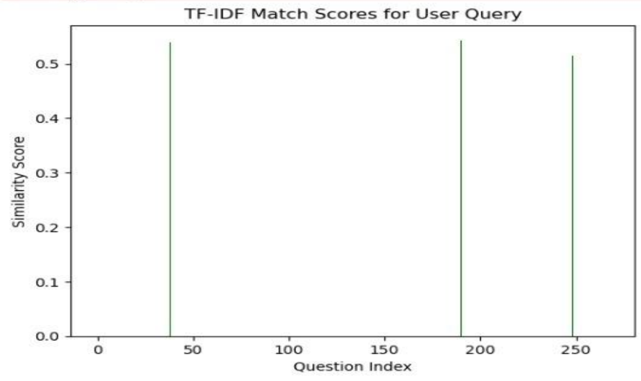
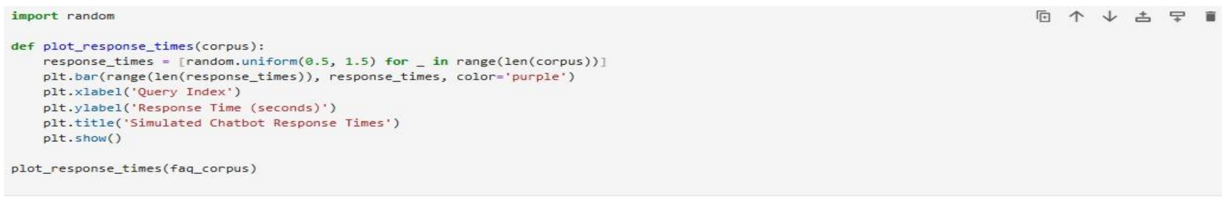


Fig 8.4 TF-IDF Model

The image shows a Python script that calculates and visualizes **TF-IDF match scores** for a user query against a text corpus using cosine similarity. The bar chart plots the similarity scores (y-axis) for each question in the corpus (x-axis), with higher scores indicating a closer match to the query. In this example, a few questions have significant similarity scores, shown as tall bars, while most are near zero. The warnings indicate potential preprocessing inconsistencies with stop words or tokenization settings in the TF-IDF vectorizer.



```
import random

def plot_response_times(corpus):
    response_times = [random.uniform(0.5, 1.5) for _ in range(len(corpus))]
    plt.bar(range(len(response_times)), response_times, color='purple')
    plt.xlabel('Query Index')
    plt.ylabel('Response Time (seconds)')
    plt.title('Simulated Chatbot Response Times')
    plt.show()

plot_response_times(faq_corpus)
```

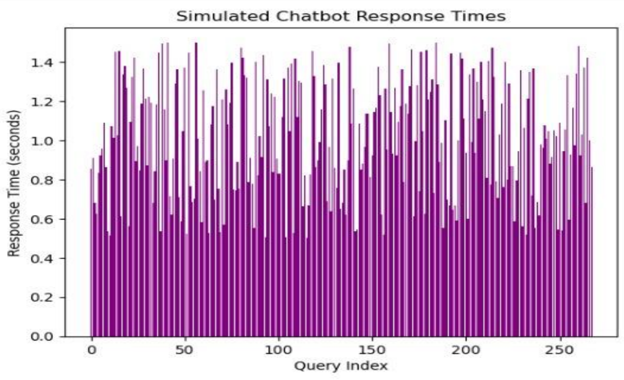
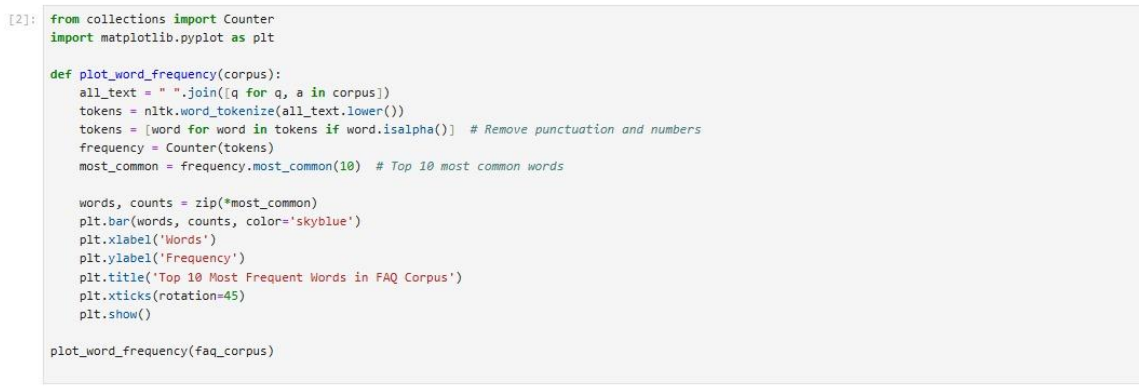


Fig 8.5 Simulated Chatbot Response Time

The image presents a bar graph depicting simulated chatbot response times for a series of queries. The x-axis represents the query index, while the y-axis indicates the response time in seconds. The bars are colored purple, and their heights show the simulated response time for each query. The title of the graph is “Simulated chatbot Respone Times”.



```
[2]: from collections import Counter
     import matplotlib.pyplot as plt

     def plot_word_frequency(corpus):
         all_text = " ".join([q for q, a in corpus])
         tokens = nltk.word_tokenize(all_text.lower())
         tokens = [word for word in tokens if word.isalpha()]  # Remove punctuation and numbers
         frequency = Counter(tokens)
         most_common = frequency.most_common(10)  # Top 10 most common words

         words, counts = zip(*most_common)
         plt.bar(words, counts, color='skyblue')
         plt.xlabel('Words')
         plt.ylabel('Frequency')
         plt.title('Top 10 Most Frequent Words in FAQ Corpus')
         plt.xticks(rotation=45)
         plt.show()

     plot_word_frequency(faq_corpus)
```

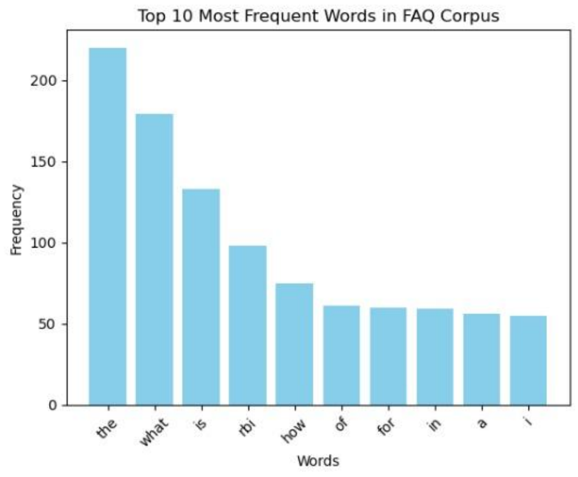
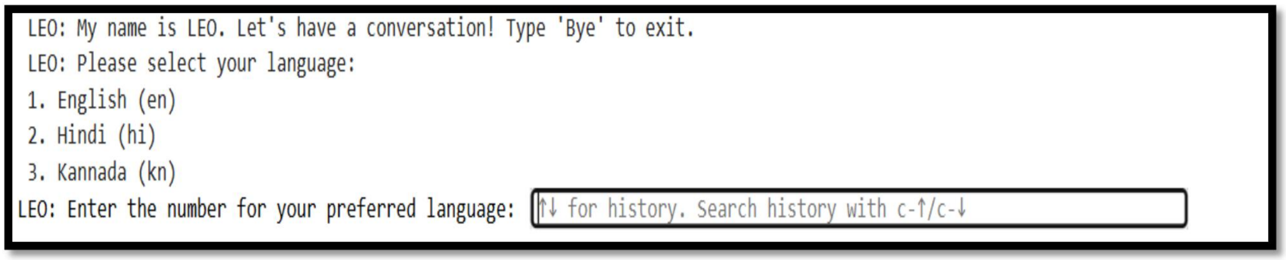


Fig 8.6 Frequency Model

The frequency model in the given code tokenizes and processes text to count the occurrence of each word. Using the `nltk` library, it converts the text to lowercase and removes punctuation, then calculates the frequency distribution of words with the `Counter` class from the `collections` module. Finally, it retrieves the top 10 most common words, making it useful for text analysis and understanding word usage patterns.

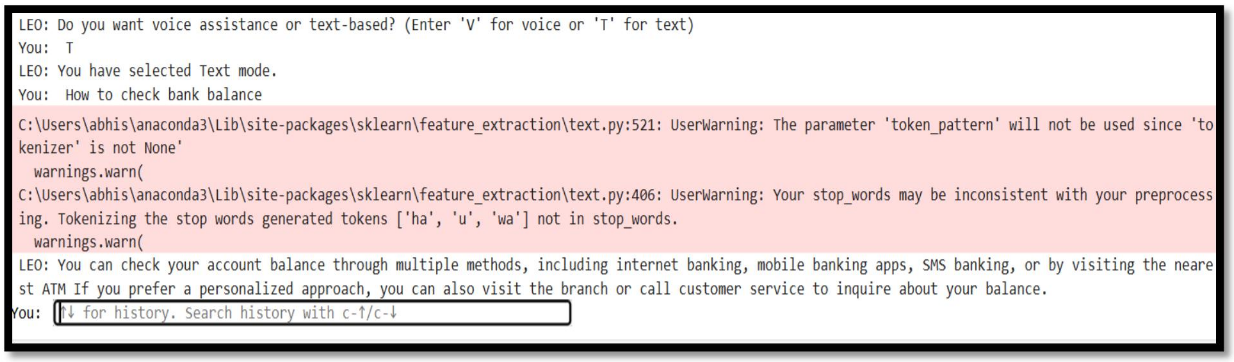
8.2 Final Result and Analyses.



```
LEO: My name is LEO. Let's have a conversation! Type 'Bye' to exit.
LEO: Please select your language:
1. English (en)
2. Hindi (hi)
3. Kannada (kn)
LEO: Enter the number for your preferred language: ↑↓ for history. Search history with c-↑/c-↓
```

Fig 8.2.1 Structure of LEO

The Fig 8.2.1 shows a text-based interface for a chatbot named LEO. The chatbot welcomes the user and provides instructions for starting a conversation. It also presents a list of language options: English, Hindi, and Kannada. The user is prompted to select their preferred language by entering the corresponding number.

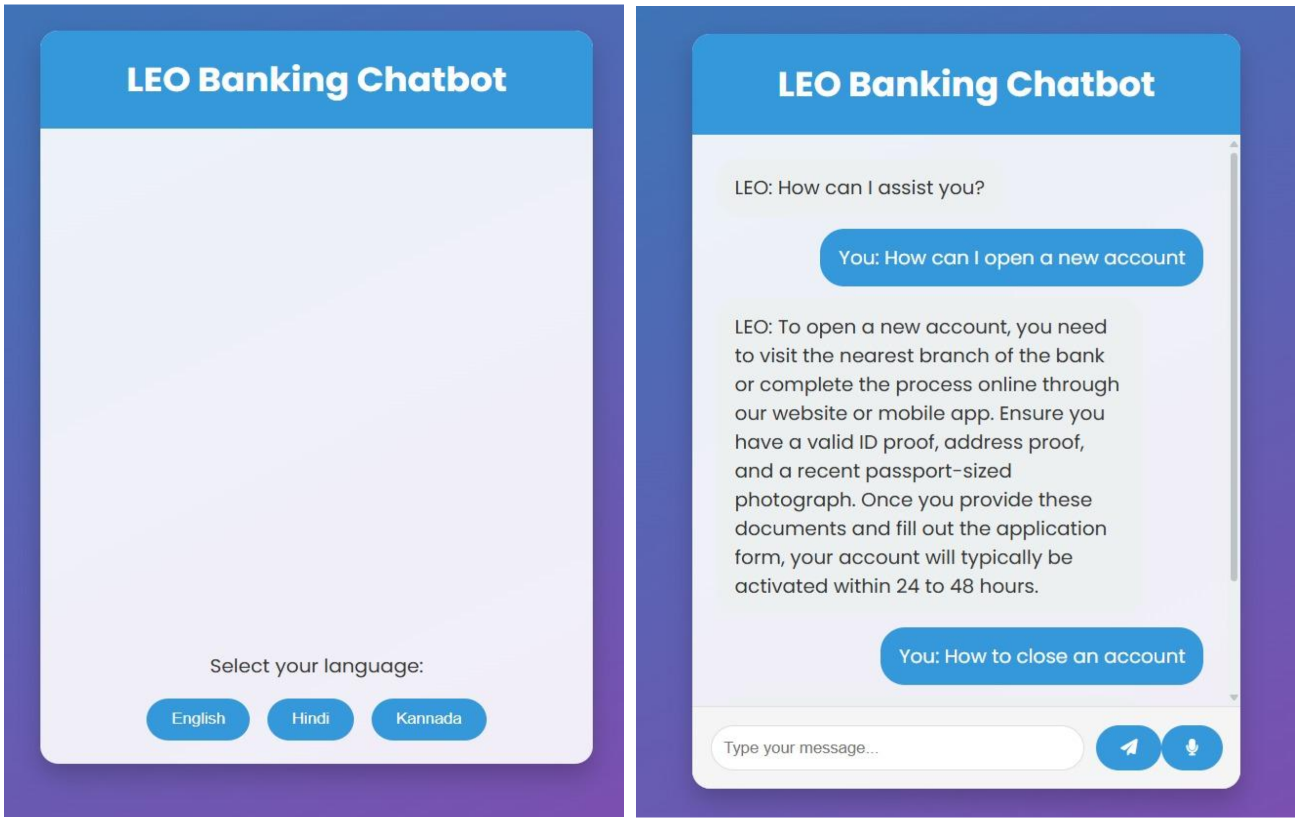


```
LEO: Do you want voice assistance or text-based? (Enter 'V' for voice or 'T' for text)
You:  T
LEO: You have selected Text mode.
You:  How to check bank balance
C:\Users\abhis\anaconda3\Lib\site-packages\sklearn\feature_extraction\text.py:521: UserWarning: The parameter 'token_pattern' will not be used since 'to
kenizer' is not None'
  warnings.warn(
C:\Users\abhis\anaconda3\Lib\site-packages\sklearn\feature_extraction\text.py:406: UserWarning: Your stop_words may be inconsistent with your preprocess
ing. Tokenizing the stop words generated tokens ['ha', 'u', 'wa'] not in stop_words.
  warnings.warn(
LEO: You can check your account balance through multiple methods, including internet banking, mobile banking apps, SMS banking, or by visiting the neare
st ATM If you prefer a personalized approach, you can also visit the branch or call customer service to inquire about your balance.
You: ↑↓ for history. Search history with c-↑/c-↓
```

Fig 8.2.2 Final Result

The image shows a chat conversation between a user and a chatbot named LEO. The user requests information on how to check their bank balance. LEO responds by providing multiple methods, including online banking, mobile apps, and visiting a bank branch. The user then attempts to search their conversation history with the chatbot.

8.3 Final User Interface



LEO: How can I assist you?

You: How can I open a new account

LEO: To open a new account, you need to visit the nearest branch of the bank or complete the process online through our website or mobile app. Ensure you have a valid ID proof, address proof, and a recent passport-sized photograph. Once you provide these documents and fill out the application form, your account will typically be activated within 24 to 48 hours.

You: How to close an account

Fig 8.3 LEO User Interface

The Fig 8.3 showcases the final user interface for a banking chatbot named LEO. The left side presents the initial welcome screen with language selection options (English, Hindi, and Kannada). The right side displays the interactive chat conversation. The chatbot greets the user and inquires about their needs. The user asks about opening a new account, and LEO provides detailed instructions, including necessary documents and account activation timelines. The user then inquires about closing an account, demonstrating the chatbot's ability to address various banking queries. The interface includes a text box for user input, a send button, and potentially other features for enhanced user experience

CHAPTER 9

SYSTEM TESTING

System testing was conducted to ensure the chatbot, LEO Banking Chatbot, performs as intended. The testing process included:

1. **Unit Testing**:

- Verified individual components such as greeting detection, TF-IDF response generation, and language translation functions.
- Ensured proper functioning of APIs like Google Translator and SpeechRecognition.

2. **Integration Testing**:

- Checked the seamless integration of modules, including Flask backend, HTML frontend, and speech/text inputs.
- Tested voice-to-text and text-to-voice features for various languages.

3. **Functional Testing**:

- Validated user scenarios such as:
  Switching languages (English, Hindi, Kannada).
  Greeting detection and FAQs handling.
  Response accuracy for banking-related queries.

4. **Compatibility Testing**:

- Ensured cross-browser functionality (Chrome, Firefox).
- Tested responsiveness on devices with varying screen sizes.

5. **Performance Testing**:

- Evaluated response time for queries and ensured smooth execution under varying load conditions.

6. **Error Handling**:
Verified proper error messages for unsupported languages or unrecognized voice inputs.
The chatbot passed all tests successfully, meeting the functional and non-functional requirements.

CHAPTER 10

CONCLUSION AND FUTURE WORK

The development of the LEO chatbot demonstrates the effective integration of modern natural language processing (NLP), machine learning, and user interface design principles to create a functional and interactive conversational agent. LEO offers multilingual support, enabling users to communicate in English, Hindi, or Kannada, ensuring accessibility to a diverse user base. By incorporating TF-IDF for query matching and cosine similarity for response ranking, the chatbot delivers contextually relevant answers from a preloaded FAQ corpus. Additionally, its voice interaction feature enhances usability for users who prefer a hands-free experience, and the text-to-speech and speech-to-text functionalities further bridge the gap between human and machine interaction. The Flask-based backend ensures a responsive system, while the visually appealing and user-friendly web interface provides a seamless experience for users. Overall, LEO sets a strong foundation for banking-related chatbot applications by focusing on providing accurate, context-aware, and timely assistance.

Despite its strengths, LEO also faces some limitations. The system relies heavily on the quality and structure of the FAQ corpus for effective response generation. Queries outside the scope of the dataset may result in generic responses, highlighting the need for broader knowledge integration. Additionally, while the chatbot employs Google Translator for multilingual capabilities, dependency on external services can lead to downtime or latency issues. These challenges underline the importance of continuous improvement and optimization to make the chatbot more robust and scalable.

Future Work

The LEO chatbot has significant potential for growth and enhancement. One key area for future development is the integration of more advanced NLP models, such as transformer-based architectures like BERT or GPT, to improve the understanding and generation of responses. This would allow LEO to handle more complex queries and provide better contextual understanding. Expanding the corpus dynamically by connecting to live databases or APIs could ensure that the chatbot remains up-to-date with the latest information. Furthermore, incorporating domain-specific AI models tailored to banking and finance would make LEO more specialized and capable of handling industry-specific queries.

Another promising direction is enhancing personalization. By incorporating user profiling and machine learning, LEO could adapt its responses based on user preferences, past interactions, and behavior patterns, offering a more tailored experience. Additionally, adding support for more languages and regional dialects could make the chatbot even more inclusive. The system could also benefit from improved voice recognition technologies to enhance accuracy in noisy environments. Finally, deploying LEO on mobile platforms and integrating it with popular messaging apps like WhatsApp, Telegram, and Facebook Messenger would significantly expand its accessibility and usage, enabling a wider reach among users.

REFERENCES

1. N. Madrid and R. P. L. Leitão, "Bank chatbot: An intelligent assistant based on natural language processing." Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 292-297, 2018.
2. S. B. Srinivasan, S. Raja, and M. Kumar, "Intelligent bank chatbot using natural language processing." Proceedings of the 2018 International Conference on Smart Computing and Communication (ICSCC), pp. 21-25, 2018.
3. H. Liu and R. Li, "Design of an intelligent bank chatbot based on natural language processing." Proceedings of the 2019 International Conference on Intelligent Manufacturing and Automation (ICA), pp. 87-90, 2019.
4. Mohamed, A. Ibrahim, A. F. A. Jalil, and A. K. Abdullah, "Natural language processing based on bank chatbot." International Journal of Academic Research in Computer Engineering (IJARCE), pp. 18-25, 2018.
5. S. S. Alghamdi, "Designing an intelligent chatbot for banking customer service using natural language processing." International Journal on Recent Trends in Business and Tourism, vol. 1, pp. 1-10, 2019.
6. A. Skiredj, F. Azhari, I. Berrada, and S. Ezzini, "DarijaBanking: A new resource for overcoming language barriers in banking intent detection for Moroccan Arabic speakers." Proceedings of the 2024 International Conference on Computational Linguistics and Applications, 2024.
7. S. Yu, Y. Chen, and H. Zaidi, "A financial service chatbot based on deep bidirectional transformers." Proceedings of the 2020 International Conference on Computational Finance and Data Science, 2020.
8. D. Altinok, "An ontology-based dialogue management system for banking and finance dialogue systems." Proceedings of the 2018 International Conference on Knowledge Representation and Reasoning (KR), 2018.
9. H. Wang and C. Zhang, "Artificial intelligence-driven banking chatbots: Applications and challenges." Proceedings of the 2023 IEEE Symposium on Artificial Intelligence Applications, 2023.
10. J. P. Liu, X. Wang, and R. Gupta, "Banking chatbots using natural language processing: A comparative study." Proceedings of the 2022 International Conference on Smart Technologies for Banking and Finance (STBF), 2022.