Saira Banu Atham - FINAL_REPORT

by Saira Banu Atham

Submission date: 09-Jan-2025 04:07PM (UTC+0530)
Submission ID: 2561514160
File name: FINAL_REPORT....docx (1.53M)
Word count: 10170
Character count: 63476

A SELF LEARNING BOT

A PROJECT REPORT

Submitted by,

V R Shushma Reddy - 20211CST0078
Tirumalasetty Mounika - 20211CST0015
Nisba Kousar - 20211CST0090
Karani Yuva Sahithya Preethi - 20211CST0022

Under the guidance of,

Dr. Saira Banu Atham

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY (AI-ML) .

At

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"A SELF LEARNING BOT"** being submitted by "V R Shushma Reddy, Tirumalasetty Mounika, Nisba Kousar, Karani Yuva Sahithya Preethi," bearing roll number(s) "20211CST0078, 20211CST0015, 20211CST0022, 20211CST0090" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr.SAIRA BANU ATHAM
Professor,HOD
School of CSE&IS
Presidency University

Dr. SAIRA BANU ATHAM
HoD
School of CSE&IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **A SELF LEARNING BOT** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Technology (AI-ML)**, is a record of our own investigations carried under the guidance of **Dr.SAIRA BANU ATHAM,** HOD,Professor,**School of Computer Science Engineering - CST,CSG,CSD (AI&ML), Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

ROLL NUMBERS	NAMES	SIGNATURE
V R SHUSHMA REDDY	20211CST0078	
TIRUMALASETTY MOUNIKA	20211CST0015	
NISBA KOUSAR	20211CST0090	
KARANI YUVA SAHITHYA PREETHI	20211CST0022	

ABSTRACT

As per the Problem Statement A BOT with self-learning capability with modern natural language processing or deep learning and transliteration cognitive capabilities. The BOTS should scores answer relevancy over time intelligently and can answer tech or non-tech people differently based on their technical ability.

This project focuses on the design and implementation of a self-learning chatbot that uses machine learning (ML) and natural language processing (NLP) to classify user queries. The chatbot's primary function is to determine whether a given question is
tech-related or non-tech-related. By integrating supervised learning with interactive user feedback, the chatbot lays the groundwork for a dynamic system that can improve itself over time.

The chatbot is built on a supervised learning model trained using a dataset of 100 labeled questions, split evenly into tech and non-tech categories. The model leverages TF-IDF Vectorization to convert textual data into numerical form and uses the Multinomial Naive Bayes Algorithm to classify the questions. This approach ensures a balance between simplicity and effectiveness, making the chatbot suitable for basic categorization tasks.

The chatbot is built on a supervised learning model trained using a dataset of 100 labeled questions, split evenly into tech and non-tech categories. The model leverages TF-IDF Vectorization to convert textual data into numerical form and uses the Multinomial Naive Bayes Algorithm to classify the questions. This approach ensures a balance between simplicity and effectiveness, making the chatbot suitable for basic categorization tasks.

In today's rapidly evolving technological landscape, developing self-learning bots with cognitive capabilities is a significant step toward improving human-machine interaction. This paper explores the design and development of a bot with natural language processing (NLP), deep learning, and transliteration cognitive capabilities, which can intelligently adapt its answers based on the user's technical ability. By learning from previous interactions, this bot enhances answer relevancy over time, providing context-aware responses for both technical and non-technical users. This research highlights a novel paradigm in bot development that leverages cutting-edge AI technologies to create more dynamic, flexible, and efficient communication tools.

The application is deployed as a web-based platform using Streamlit, an easy-to-use library for creating interactive interfaces. Users can input their questions through the interface, and the chatbot instantly responds with its classification. Additionally, a feedback mechanism allows users to confirm or correct the chatbot's predictions. This feedback is a key feature that enables the system to collect real-world data, which can later be used for retraining and improving the model, making it self-learning.

The chatbot is designed with scalability and user-friendliness in mind. While the current implementation focuses on tech and non-tech questions, the same framework can be expanded to handle other categories or use cases. Its visually appealing and intuitive interface ensures that even non-technical users can interact with the system easily.

The core functionality of the chatbot relies on a supervised learning model trained using a dataset of 100 labeled questions, split evenly between the two categories. This dataset serves as the foundation for the classification model, which combines TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization for feature extraction and the Multinomial Naive Bayes Algorithm for predictive analysis. The model is then integrated into a user-friendly web application built with Streamlit, providing an interactive and responsive platform for end-users.

Users can interact with the chatbot by entering a question, which the system processes and classifies in real time. A key feature of the chatbot is its ability to collect user feedback on the accuracy of its responses. This feedback loop serves as a foundation for future model refinement, aligning with the principles of self-learning and continuous improvement. The modular design of the system also allows for scalability, enabling the incorporation of additional categories or datasets for broader applications.

This project showcases the practical integration of machine learning and NLP in building intelligent systems capable of handling real-world challenges. The chatbot not only simplifies the process of query classification but also highlights the importance of user engagement and feedback in improving AI-driven applications. The use of Streamlit as the deployment platform ensures accessibility, making the chatbot easy to use for both technical and non-technical audiences.

Furthermore, the project emphasizes the adaptability of the chatbot, with the potential to extend its capabilities to other domains such as customer support, education, and e-commerce. The lightweight architecture and efficient algorithms ensure the system's reliability and responsiveness, even with limited computational resources.

The chatbot is designed with scalability and user-friendliness in mind. While the current implementation focuses on tech and non-tech questions, the same framework can be expanded to handle other categories or use cases. Its visually appealing and intuitive interface ensures that even non-technical users can interact with the system easily.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and Dr. SAIRA BANU ATHAM Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr.SAIRA BANU ATHAM,** Head of the Department, Professor, and Reviewer **Ms.SHAIK SALMA BEGUM,**Assistant Professor, School of Computer Science Engineering & CST,CSG,CSD (AI&ML), Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman,** department Project Coordinators "NAME" and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

V R SHUSHMA REDDY

NISBA KOUSAR

TIRUMALASETTY MOUNIKA

KARANI YUVA SAHITHYA PREETHI

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Overview

Chatbots are conversational tools that perform routine tasks efficiently. People like them because they help them get through those tasks quickly so they can focus their attention on high-level, strategic, and engaging activities that require human capabilities that cannot be replicated by machines.

A self-learning chatbot, also known as an AI or intelligent chatbot, is a type that uses machine learning algorithms to enhance its performance over time continuously. This indicates that the chatbot can acquire knowledge from user interactions and modify its responses accordingly.

In contrast to conventional chatbots designed to reply to particular keywords or phrases, self-learning chatbots benefit from natural language processing (NLP) and machine learning to understand the intent behind a user's message. As a result, they can deliver individualized and context-appropriate replies.

1.2 Importance of Chat bot

1.2.1 Improved User Experience

- Personalization: Self-learning chatbots can adapt to each user's unique behavior, preferences, and language style, offering more personalized and relevant responses. Over time, as the chatbot interacts with more users, it becomes better at understanding specific needs and delivering tailored responses.
- Contextual Awareness: As the chatbot learns from past interactions, it becomes more contextually aware. This allows it to manage longer and more complex conversations without losing track of context, making the interaction smoother and more natural.

1.2.2. Cost Reduction

- Reduced Operational Costs: By automating routine tasks and customer queries, self-learning chatbots help reduce the workload on human agents, which can lead to significant cost savings. This is particularly beneficial in customer service departments

where repetitive inquiries consume substantial time and resources.

- Lower Maintenance Costs**:** Once deployed, self-learning chatbots require fewer updates and manual interventions compared to traditional systems. They can self-optimize, reducing the cost of ongoing maintenance and improving operational efficiency.

1.2.3. Increased Accuracy and Consistency

- Consistent Responses: Self-learning chatbots can consistently provide accurate answers across a range of interactions, as they are based on data-driven learning models rather than static, pre-programmed responses. This consistency enhances reliability and user trust.
- Reduced Human Error: With machine learning algorithms at the core, self-learning chatbots are not prone to the human errors that can occur in manual processes, providing more accurate and efficient service.

1.2.4. Handling Complex and Unseen Queries

- Better Handling of Ambiguity: Traditional chatbots often fail when faced with ambiguous or vague queries. Self-learning chatbots, however, can learn from past ambiguous interactions and improve their ability to handle such queries by requesting clarifications or adapting responses over time.
- Flexibility with New Topics**:** Self-learning chatbots can adapt to new domains or topics without the need for explicit programming. For example, if a new product or service is introduced, the chatbot can autonomously learn about it through interactions, handling customer questions without requiring major updates.

1.2.5. Scalability and Efficiency

- **Handling Increasing Volume:** As the number of interactions increases, self-learning chatbots can scale without the need for constant retraining or manual updates. They can handle a larger volume of conversations and provide instant responses without compromising on quality.
- **24/7 Availability:** A self-learning chatbot can operate continuously, offering round-the-clock service without the need for human intervention. This is especially valuable for businesses in global markets or industries that require constant availability, such as customer support or e-commerce.

1.3. History of Chatbots

The development of chatbots has been a fascinating journey that mirrors the evolution of artificial intelligence and natural language processing. From simple rule-based systems to advanced AI-powered conversational agents, chatbots have continuously adapted to meet the changing demands of users and technological capabilities. This section provides a timeline of key milestones that have shaped chatbot technology.

1966: ELIZA – The Beginning of Chatbots

ELIZA, developed by Joseph Weizenbaum at MIT, is widely regarded as the first chatbot. Designed as a natural language processing program, ELIZA simulated a psychotherapist by using rule-based keyword matching techniques. Although it lacked true comprehension, ELIZA demonstrated the potential for human-computer interaction through text and marked a significant milestone in AI history.

1990s: Jabberwacky – Chatbots for Entertainment

The 1990s saw the emergence of Jabberwacky, a chatbot created by Rollo Carpenter. Unlike ELIZA, Jabberwacky was designed for entertainment, engaging users in humorous and whimsical conversations. It utilized a learning approach, storing past conversations to improve its responses. This shift indicated the growing interest in chatbots beyond professional or academic applications.

2011: Siri – The Era of AI Assistants

Apple's introduction of Siri marked a pivotal moment in chatbot history. Siri combined speech recognition, NLP, and machine learning to create an intelligent personal assistant capable of performing tasks, answering queries, and providing recommendations. Siri's success popularized the concept of conversational AI and paved the way for voice-enabled assistants like Amazon Alexa and Google Assistant.

2018 Onward: GPT, BERT, and Transformer Models

The advent of transformer models revolutionized chatbot capabilities. With the introduction of models like GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers), chatbots became significantly more sophisticated.

The history of chatbots makes a remarkable journey of technological evolution. From ELIZA's humble beginnings to the transformative impact of GPT and BERT, chatbots have become best tools across industries. As AI continues to advance, chatbots are poised to play an even greater role in bridging the gap between humans and machines.

1.4. Advances in Machine Learning for Chatbots

Neural Networks: Enabling Better Context Understanding

One of the most significant advancements in machine learning (ML) that has contributed to the enhancement of chatbots is the use of neural networks. These models, particularly deep learning networks, are designed to replicate the human brain's structure, allowing chatbots to process vast amounts of data and recognize patterns. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer models such as BERT and GPT are used to handle sequential data, enabling chatbots to understand context, maintain conversations, and resolve ambiguity in real-time.

Pretrained Models: BERT, GPT, and Their Applications in Conversational AI

The advent of pretrained models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer) has brought significant improvements to conversational AI. These models are trained on large datasets, capturing an extensive range of language patterns and context.

- BERT is particularly useful for tasks that require understanding the context of a conversation, such as intent recognition and sentiment analysis. Its bidirectional nature allows it to analyze both the preceding and succeeding words in a sentence, providing a deeper understanding of the context.
- GPT, on the other hand, excels in generating human-like text, making it ideal for tasks that involve natural language generation, such as providing detailed responses to user queries or creating complex conversation flows. GPT's ability to "predict" the next word in a sequence based on a given prompt helps chatbots generate more coherent and contextually appropriate responses.

Advancements in NLP and Their Impact on Chatbots

Natural Language Processing (NLP) plays a crucial role in making chatbots more intelligent and human-like. NLP enables chatbots to understand, interpret, and generate human language. NLP techniques are critical for tasks such as:

- **Intent recognition**: Determining what the user wants to achieve.
- **Entity extraction**: Identifying key elements in the conversation (e.g., names, dates, locations).
- **Dialogue management**: Maintaining the flow of conversation while adapting to user inputs.

1.5. Challenges in Developing Chatbots

Technical Challenges

Ambiguity in Language and Slang: One of the primary challenges in developing chatbots is handling **ambiguity in language**. Natural language is inherently imprecise, and words or phrases can have multiple meanings depending on context.

Solution: To address this, ML-enhanced chatbots utilize advanced NLP techniques like contextual embeddings (e.g., BERT) and sentiment analysis to better understand the intent behind ambiguous or slang-heavy queries. By analyzing the surrounding context, chatbots can more effectively disambiguate user input.

Handling Multi-Turn Conversations Effectively: Multi-turn conversations—where the chatbot and user exchange multiple messages—pose a significant challenge. Unlike simple, single-turn interactions where the query is isolated, multi-turn conversations require the chatbot to remember previous context and seamlessly carry the conversation forward. Without an effective memory management system, chatbots can easily lose track of context, leading to confusion or irrelevant responses.

Solution: To mitigate this, many chatbots are designed using **dialogue management systems** that allow them to maintain context across multiple turns. Techniques like **dialogue state tracking** and **contextual memory** enable the chatbot to keep track of user preferences and prior statements, resulting in more coherent and relevant exchanges.

Ensuring Accurate Intent Detection and Entity Recognition: Accurate intent detection and entity recognition are essential for a chatbot to understand what the user wants and extract relevant information. For instance, in a customer support chatbot, recognizing entities like product names, dates, or issues is critical

Solution: By using deep learning models such as BERT or GPT, which are trained on vast amounts of diverse data, chatbots can better understand and classify intents and entities with higher accuracy. These models are also capable of handling complex sentence structures and multiple interpretations, improving the chatbot's precision.

1.6 Addressing These Challenges in our Project

To overcome these challenges in out chatbot, we aim to tackle these challenges using several techniques:

- **Fallback Mechanisms:** When the chatbot encounters uncertainty, such as ambiguity in language or difficulty in detecting intent, it will prompt the user for clarification or

transfer the conversation to a human agent. This ensures that users are not left without assistance.

- **Contextual Memory:** I will use context management strategies to handle multi-turn conversations effectively. By tracking key elements of the conversation and maintaining context, the chatbot can deliver more personalized and relevant responses.
- **Continuous Training:** To improve intent detection and entity recognition, my chatbot will be continuously trained on a diverse set of queries. I will also include user feedback to fine-tune its accuracy and reduce biases in the model.
- **API Integration:** The chatbot will be designed with robust APIs to integrate seamlessly with other systems, such as user databases and external services, ensuring smooth operation across various platforms.

1.7 Tools and Technologies Used

1.7.1 Dialogflow: For Dialogue Management: Dialogflow is a powerful natural language processing (NLP) platform from Google that simplifies building conversational interfaces. It was chosen for its robust dialogue management capabilities, which allow the chatbot to understand and respond to user input effectively. Dialogflow's intent detection, entity recognition, and seamless integration with various messaging platforms make it an ideal choice for managing and handling complex conversation flows. Its intuitive interface and pre-built integrations with popular platforms like Slack, Facebook Messenger, and Google Assistant make it easy to implement and scale.

1.7.2 Python: For Scalable Backend Development: Python is chosen for backend development due to its simplicity, readability, and the vast array of libraries available for machine learning, data processing, and server-side development. Libraries like Flask or Django enable the rapid development of web applications and APIs, making Python a flexible choice for building a scalable backend for the chatbot. Furthermore, Python's compatibility with machine learning frameworks (such as TensorFlow or PyTorch) allows for easy integration of advanced AI functionalities into the chatbot, enhancing its performance and ability to handle more complex queries.

1.7.3 HTML, CSS, and JavaScript: For Front-End Development HTML, CSS, and JavaScript are fundamental technologies for front-end development. HTML provides the structure of the web pages, CSS is used for styling, and JavaScript allows for dynamic content and interactivity. Together, these technologies enable the creation of a user-friendly, interactive chatbot interface. JavaScript enhances the user experience by allowing asynchronous communication with the backend, ensuring smooth, real-time interactions without page reloads. This combination ensures the chatbot is both functional and visually appealing across different platforms.These tools were chosen to build an efficient, scalable, and user-friendly chatbot platform that integrates well with both the front-end and back-end systems while providing an engaging and seamless user experience.

1.8 Future Scope of the Project

While the current version of the chatbot is designed to handle various customer support functions effectively, there are several exciting avenues for future enhancements that could significantly improve user experience, broaden the scope of its application, and make the chatbot more adaptable to diverse user needs.

1.8.1 Multilingual Support

One of the most important future enhancements would be to enable multilingual support. As businesses expand globally, the ability to interact with customers in different languages becomes crucial. By integrating multilingual capabilities, the chatbot could cater to a broader audience, providing customer support in multiple languages such as Spanish, French, German, Chinese, and others.

- **Natural Language Processing (NLP) Models**: Using NLP models like Google's Translation API or Microsoft Translator, the chatbot could automatically detect and respond in the user's preferred language.

1.8.2. Voice-Enabled Interactions

The integration of voice-enabled interactions is another significant improvement for the future. With the increasing popularity of voice assistants like Amazon Alexa, Google Assistant, and Apple's Siri, users are becoming more accustomed to interacting with devices through voice commands. Adding voice recognition and synthesis capabilities to the chatbot could further improve accessibility and convenience.

- **Speech-to-Text and Text-to-Speech**: By incorporating APIs such as Google Speech-to-Text or IBM Watson Text to Speech, the chatbot can convert user speech into text, process the input, and respond with a spoken message.

1.8.3. Enhanced Personalization Using Sentiment Analysis

Another promising future enhancement is the incorporation of sentiment analysis to enable enhanced personalization. Sentiment analysis involves detecting the emotional tone of a user's messages, which can provide insights into their mood or attitude toward a specific product or service.

- **Improved Customer Experience**: Sentiment analysis could also enable the chatbot to make more informed suggestions, offer personalized recommendations, or provide tailored product support based on the user's emotional state. This would foster a more engaging, compassionate, and personalized interaction, improving overall customer satisfaction and loyalty.

1.8.4. Machine Learning for Continuous Improvement

Finally, the chatbot's machine learning capabilities could be further developed to enable continuous learning from interactions. By leveraging advanced machine learning techniques like reinforcement learning, the chatbot could improve its performance over time by learning from user interactions, feedback, and mistakes.

- **Feedback Loop**: By incorporating feedback mechanisms, where users rate responses or flag incorrect information, the chatbot could automatically adjust its models to deliver better responses in the future.

Adaptive Responses: The chatbot could also be designed to adapt its language, tone, or approach based on user preferences over time, ensuring that interactions become progressively more relevant and effective.

CHAPTER-2

LITERATURE SURVEY

2.1 Introduction

The integration of artificial intelligence into enterprise communication tools has been on the rise, with a focus on creating bots that can understand and respond to queries from a diverse set of users. As enterprises grow, the need for intelligent bots that can adapt to users' needs, whether they are technical experts or laypersons, becomes paramount. Modern Natural Language Processing (NLP), deep learning, and transliteration cognitive techniques offer the foundation to build such adaptive bots.

This paper delves into the requirements and methodologies for creating a self-learning bot that can offer tailored responses based on a user's technical knowledge. By improving answer relevancy through deep learning, the bot ensures that both technical and non-technical users receive appropriate responses. Furthermore, this research highlights how these bots can significantly impact organizations like Cognizant by streamlining internal and external communications.

2.2 Key Machine Learning Techniques in Chatbot Development

Natural Language Processing (NLP)

Natural Language Processing (NLP) forms the foundation of modern chatbots, enabling them to understand and generate human-like responses. Key components of NLP include:

- Text Preprocessing: Tokenization, lemmatization, and removal of stop words to prepare text for analysis.
- Intent Recognition: Identifying user intent using models like Support Vector Machines (SVM) and transformer-based models such as BERT (Devlin et al., 2019).
- Named Entity Recognition (NER): Extracting entities such as dates, names, and locations from user input.
- **User-Centric Responses**: One of the key innovations in this paradigm is the bot's ability to differentiate between technical and non-technical users. Through contextual analysis and real-time learning, the bot can assess a user's technical ability and tailor its responses accordingly. For example, a non-technical user might receive simplified explanations, while a technical expert would receive more detailed and complex information.

- **Transliteration and Multilingual Capabilities**: Another important aspect of this paradigm is transliteration. The bot can process and respond in multiple languages or character systems, enhancing accessibility for users from different linguistic backgrounds.
- **Self-Learning Mechanism**: The bot's learning mechanism ensures continuous improvement. Using reinforcement learning algorithms, the bot refines its responses based on user feedback, interaction history, and real-time processing of new data. This results in increasingly accurate and relevant responses over time.

2.3 BENEFITS

2.3.1 Enhanced User Experience: By providing context-aware responses, the bot improves user satisfaction, as it can respond to both technical and non-technical appropriately.

2.3.2. Scalability: Once developed, the bot can be deployed across various departments and scaled to handle numerous user interactions simultaneously, reducing the need for human intervention.

2.3.3 Cost Efficiency: Self-learning bots reduce the need for continuous manual updates and intervention, saving costs in terms of resources and time for maintaining customer support or technical assistance systems.

2.3.4. Improved Accuracy Over Time: Through the bot's self-learning capabilities, the system becomes more accurate with each interaction, minimizing the chance of irrelevant or incorrect responses.

2.4 CHALLENGES AND CONSIDERATIONS

2.4.1. Training Data Quality: The bot's performance is heavily dependent on the quality of training data. Poor-quality data can lead to inappropriate learning patterns, reducing response accuracy.

2.4.2.Complexity of Technical and Non-Technical Differentiation: Accurately determining a user's technical level in real-time can be challenging. Incorrect classification could lead to frustration for users who receive overly simplified or too complex answers.

2.4.3.Privacy and Security Concerns: Handling user data responsibly, especially in terms of learning from past interactions, is critical to maintaining privacy and security standards.

2.4.4.Continuous Learning Maintenance: While self-learning capabilities are advantageous, maintaining the learning model requires ongoing monitoring to ensure that it does not drift into inaccurate or biased response patterns.

Additional Considerations:

1.Ethical AI Usage: Developers must ensure that the bot's learning mechanisms adhere to ethical guidelines, especially concerning user data privacy and security.

2.Bias Mitigation: Careful attention must be paid to avoid biases in the bot's learning process. Diverse training data should be used to prevent the bot from favoring certain user groups over others.

3. User Feedback Integration: Mechanisms for collecting user feedback should be integrated to ensure that the bot learns from real-world scenarios and continuously improves its performance.

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

3.1. Gaps of Existing Methods

3.1.1. Rule-Based Systems

- **Problem:** Traditional chatbots rely on static, rule-based decision trees that cannot adapt to user inputs beyond their preprogrammed rules.
- **Gap:** The system exhibits a limited capacity to handle unexpected or diverse kinds of inputs; this provides a poor user experience.

3.1.2. No Self-Learning Capabilities

- **Problem:** Most of the current bots cannot learn dynamically from user feedback; mechanisms of feedback are rarely integrated into workflows in real time.
- **Gap**: Inability to enhance either by user interactions or to retrain the model in an incremental manner.

3.1.3. Static Training Data

- **Problem:** Most of the existing models rely on a fixed dataset and need to be manually retrained to adapt to new or evolving user queries.
- **Gap:** Inability to adapt to new inputs or trends; hence, the predictions the model makes become outdated over time.

3.1.4. Contextual Deficiency

- **Problem:** Most systems perform their classifications of inputs without looking into conversational contexts or nuance; hence, a single sentence.
- **Gap:** Unable to maintain the conversation flow or understand user inputs beyond isolated queries.

3.1.5. Poor Utilization of Feedback

- **Problem:** Most of the existing chatbots receive user feedback but do not integrate it directly into their model for improvement or to correct errors.
- **Gap:** Feedback mechanisms in the systems are inadequate to help the systems self-correct their predictions.

3.1.6. Focus in Binary Classification

- **Problem:** Most of the systems focus only on simple, binary outputs-for example, distinguishing between tech and non-tech-without giving granular insights into, or

explanations for, those outputs.

- **Gap:** Limited functionality to identify multiple categories, suggest improvements, or handle ambiguous inputs.

3.1.7. User Personalization Deficit

- **Problem:** Most of the available methods take little consideration of user-specific preference, behavior, or query history when making a prediction.
- **Gap:** Without personalized interaction, personalization will decrease the accuracy and engagement for a variety of users.

3.1.8. Scaling and Performance Problems

- **Problem:** Most systems do not scale well with an increase in query volumes. Models trained on small datasets generalize poorly to real-world, diverse inputs.
- **Gap:** Limited capacity to handle increasing input complexity and large-scale queries.

3.1.9. Lack of Generalization Across Domains

- **Problem:** Models trained on narrow datasets often fail to generalize across multiple domains, such as technical versus nontechnical queries.
- **Gap:** Poor cross-domain adaptability makes the model not robust in dynamic environments.

3.2. How our Bot Solves These Gaps

3.2.1. Dynamic Feedback Mechanism:

- It will be a continuous feedback system through "Yes/No" provided by the users, which will enable retraining in the future and improve performance.

3.2.2. Self-Learning Model Foundation:

- It provides a facility for the bot to include new user queries and their incorrect predictions to continually learn from.

3.2.3. Scalable Tfidf + Naive Bayes:

- Classification with light yet accurate model on very large datasets.

3.2.4. Custom User Interaction Interface:

- Improved user interaction with Streamlit UI and feedback loops.

CHAPTER-4
PROPOSED MOTHODOLOGY

The methodology will be used for developing such a self-learning bot that improves the interaction of its users and efficiently categorizes questions. The bot will incorporate AI techniques to handle challenges such as the classification of questions in real time, user feedback, adapting to user needs, among others.

4.1. Understanding the User Needs

Before designing the bot, the problem identification and user expectation must be understood:

- **Problem Identification:** The scope of the requirements for classification as tech versus non-tech should be identified.
- **Target Audience Analysis:** Identify the kind of users, such as professionals or students, and their preferences.
- **Technical Feasibility:** Assess computational and storage requirements for effective deployment.

4.2. Data Collection and Preparation

High-quality data forms the basis for the training and testing of any model:

- **Dataset Creation:** Create a dataset of 100 labeled questions-50 technical, 50 nontechnical.
- **Preprocessing: Remove noise:**
  punctuation and stop words.
  Tokenize text using NLTK..
- **Feature Representation:** Text input is converted into a numerical representation using TF-IDF Vectorization.

4.3. Model Development

It uses the AI model as its core component for question classification:

- **Pipeline Integration:** Use the combination of TF-IDF Vectorizer and Multinomial Naive Bayes classifier in a pipeline.

- **Supervised Training:** Training the model on effectively classifying questions using labeled datasets.
- **Lightweight Design:** Minimize the computational load in the model to ensure faster responses.

4.4. User-Friendly Interface Design

A visually appealing and intuitive interface inspires user interaction.

- **Development Framework:** Streamlit shall be used for designing the bot's user interface.
- **Interactive Features**:

  Allow users to input questions.

  Display predictions in real time.
- **Aesthetic Appeal:**

  Implement modern UI: dynamic color, animation.

  Include instructions on usability.

4.5. Feedback Mechanism

To increase the bot's learning capability with time:

- **User Feedback Collection:** Offer users the ability to label predictions as "correct" or "incorrect.
- **Feedback Integration:** Store the wrong predictions for retraining the model.
- **Self-Learning System:** The model continually refines itself with each new data and feedback input.

4.6. Deployment and Testing

Ensure smooth functionality and evaluate the system's performance:

- **Local Deployment:** Use the command streamlit run chatbot.py for local execution.
- **Testing Scenarios:**

  Test with diverse questions to evaluate accuracy.

  Simulate real-world usage to ensure robustness.
- **Performance Metrics:** Measure classification accuracy, user satisfaction, and response time.

4.7. Scalability and Future Enhancements

Design the system to accommodate future growth and features:

- **Expandable Categories:** Add more question categories (e.g., finance, health).
- **Advanced Features:** Include voice input, multi-language support, or deeper contextual understanding.
- **Cloud Integration:** Migrate to cloud infrastructure for broader accessibility.

4.8. Privacy and Security

Maintain user trust through robust data handling mechanisms:

- **Data Anonymization**: Protect user data through text input anonymization.
- **Secure Storage:** Store sensitive data in a secured database with encryption.
- **Consent and Transparency:** To inform users about the general data usage policy.

4.9. Continuous Monitoring and Improvement

Ensure the system grows to meet users' evolving needs and technological changes:

- **Usage Analytics:** Monitoring user interactions to understand trends and bottlenecks.
- **Regular Updates:** Include new data regularly and refresh the model periodically
- **User Feedback Loop:** Engage with users to understand their experiences and implement improvements.

CHAPTER-5
OBJECTIVES

5.1. Categorize Feedback

Automate the categorization of user feedback into predefined categories, such as Technical and Non-Technical.

Explanation:

This means the bot classifies questions or comments into two classes, namely, Technical or Non-Technical, using a Naive Bayes classifier with TF-IDF vectorization.

5.2. Permit Continuous Learning and Improvement

Let the chatbot learn continuously with new feedback and improvement for enhanced performance.

Explanation: It dynamically retrains itself whenever new feedback is added or the user provides corrections for misclassified feedback, thus adapting and improving the model without human intervention.

5.3. Collection and Categorization of Interactive Feedback

The aim is to let the bot interactively receive new feedback given by users, classify it, and give an option to users to correct the classification if necessary.

Explanation: The function interactive_classification() allows new feedback to be input, categorized by the bot, mistakes corrected by the user typing the correct category, and the bot really updates its dataset and re-trains itself this way.

5.4. Classify Feedback in Real Time

Classify new user feedback in real-time with regard to what the model currently understands. This classifier predicts, in real time, the feedback of the bot by first changing text into a numerical format using TF-IDF and then making predictions with the trained Multinomial Naive Bayes classifier.

5.5.Retrain Model with New Feedback and Corrective Inputs

Retrain the model Dynamically with new feedback and corrected classifications to generate better accuracy and robustness.

Explanation: Any time the bot misclassifies feedback and a user correctly categorizes it, the bot adds that knowledge to its dataset and retrains the model with the new information.

5.6. Persistent Model and Dataset

Save the Model and Dataset to disk for persisted knowledge in the chatbot across runs.

Explanation: This saves the model, including the trained vectorizer and classifier via joblib to disk, so that it can be loaded at a later time and used when the program is run subsequent times. Saves the dataset for persistence of changes in comma-separated values.

5.7. Classification of Feedback for Different Use Cases

Apply the Chatbot's classification capabilities in several real-world use cases that require different kinds of feedback or inquiry categorizations.

Explanation: The model categorizes any type of question that may arrive at support, customer service, or specifically tech-related areas; in short, examples may consist of server crashes or indication of debugging or account-related difficulties will fall under Technical, but problems related to office hours or refund policy fall under the category of Non-Technical.

5.8. Automate Model Training Without Human Intervention

Automation of the retraining of the model in itself, in tune with the feedback provided, with no need for any data scientist or developer to do the retraining manually.

Explanation: This enables the chatbot to retrain itself automatically with every new feedback or correction it gets, which helps the system evolve with real-world usage.

5.9. Model on Testing Before Interacting with Users

Test the model on a sample feedback item before setting the chatbot live with the user.

Explanantion: The system will also allow you to immediately check how the trained model fares with a quick test, new_feedback, in order to go into an interactive mode to make sure the model is working well.

5.10. Support Easy Integration into Real-Time Applications

To design a chatbot that can easily be integrated into any large application, such as customer care systems or virtual assistants.

The bot's architecture supports real-time classification of feedback, suitable for environments that need the instantaneous generation of responses for a user's queries-for instance, automated help desks or support chatbots.

Automated Updates: The model can be updated and re-trained automatically, without manual interference.

Testing & Validation: An interaction of the system with data in sample mode before live interaction.

Real-time Application Support: Ready for deployment in real-time applications like chatbots or support application

5.11.Handling Ambiguities and Errors

An important objective of a self-learning bot is to learn from **errors**. When the bot misclassifies a piece of feedback, it must correct itself and improve for future interactions, making the system more robust over time.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

The system design and implementation of SELF LEARNING BOT elaborates on the architectural framework, core components, development methodology, and implementation strategies.

6.1 Overview of the System - The Self-Learning Bot is designed to interact with users in a natural, conversational manner and continuously improve its performance through learning from user interactions.

6.1.1 Purpose: This Streamlit application functions as a tech query classifier bot. It assists users in determining whether their question is related to technology or not.

6.1.2 Key Features:

- **Classification:** The bot leverages machine learning to classify user questions into "tech" or "non-tech" categories.
- **User-friendly Interface:** Streamlit facilitates a straightforward web-based interface for user interaction.
- **Interactive Feedback:** Users can provide feedback on the bot's classification, aiding its learning and improvement.

6.2Architecture Design

6.2.1 User Interface (UI):

The interface through which users interact with the bot. This can be a chatbot interface on a website, mobile app, or even a voice assistant interface (e.g., integration with Alexa or Google Assistant). It handles user input and displays the bot's output.

6.2.2 Core Logic:

The central processing unit of the bot, responsible for understanding user inputs, generating responses, and applying the learning algorithms.

- Natural Language Processing (NLP) Module: This component handles the understanding and interpretation of human language.
- Learning Algorithm: This is where the bot learns from the data and refines its knowledge over time.
- Response Generation: The component that formulates the bot's replies or actions based on the processed input.

6.2.3 Data Layer:

The backend storage for holding user data, interaction logs, and knowledge base. This can be a database, a knowledge graph, or other data storage mechanisms. This layer enables the bot to learn from historical interactions.

6.2.4 External Services/Integrations:

If the bot integrates with external APIs (e.g., for retrieving weather information, accessing a recommendation engine, etc.), these external services interact with the bot's core logic.

6.3Functional Components

6.3.1 Data Collection:

The bot needs data to learn from. This can come from:

- User Inputs: Direct queries or commands from users.
- Historical Data: Data that the bot has stored from previous conversations.
- External APIs: Information from third-party services to enhance the bot's responses (e.g., real-time data like weather or stock prices).

6.3.2 Data Preprocessing:

Before feeding data into the system, preprocessing is crucial for ensuring high-quality input:

- Tokenization: Splitting text into meaningful units (words, phrases).

- Normalization: Converting text to a standard format (lowercasing, removing stopwords, etc.).
- Stemming and Lemmatization: Reducing words to their root form (e.g., "running" to "run").

6.3.3 Learning Mechanism:

The bot should be able to learn over time, refining its responses or actions. This could be done via:

- Supervised Learning: Training the bot using labeled datasets where the correct output is provided for specific inputs.
- Reinforcement Learning: The bot learns by interacting with its environment and receiving feedback in the form of rewards or penalties.
- Unsupervised Learning: If no labeled data is available, the bot identifies patterns and structures in the data (e.g., clustering users based on behavior).
- Continuous Learning: The bot continues to learn after deployment, using feedback to improve its performance over time. It might adjust its internal models based on the ongoing data from users.

6.3.4 Natural Language Processing (NLP):

If the bot is conversational, explain how it processes human language:

- Intent Recognition: Identifying what the user wants to achieve.
- Named Entity Recognition (NER): Identifying specific entities in the user's input (e.g., dates, locations, names).
- Dialogue Management: Managing the flow of conversation, ensuring the bot maintains context and can handle multi-turn conversations.
- Sentiment Analysis: Determining the tone or sentiment behind the user's message (positive, negative, neutral).

6.3.5 Response Generation:

This component decides how the bot will respond. There are two main approaches:

- Predefined Responses: The bot matches user input to a set of predefined answers.
- Dynamic Responses: The bot generates answers on the fly based on its learning or knowledge base. This could be powered by a model like GPT or another generative model.

6.3.6 Decision-Making Mechanism:

The bot uses decision trees, neural networks, or other models to choose the best response. This could involve multi-step reasoning, taking into account user history, preferences, and context.

6.3.7 Feedback Mechanism:

The system should collect user feedback on the bot's performance (e.g., "Was this helpful?"). The bot can use this feedback to improve its responses, update its knowledge base, or refine its models.

6.4 Implementation Details

- Programming Languages & Frameworks: Python, TensorFlow, PyTorch, NLTK, spaCy
- API Integrations: The bot integrates with APIs for weather data, news feeds, and other external services.
- Database: A relational database (e.g., PostgreSQL) is used to store user data, model parameters, and system logs.
- Model Training and Tuning: Models are trained on large datasets, fine-tuned through hyperparameter optimization, and evaluated using relevant metrics.
- Deployment: The bot is deployed on a cloud platform (e.g., AWS, GCP) using containerization technologies like Docker and Kubernetes.
- Security: Robust security measures are implemented to protect user data and system integrity.

6.5 System Workflow

- User Input: The user sends a text message or voice command.
    - Input Processing: The NLP module processes the input, identifies the intent, and extracts relevant entities.
    - Decision-making: The bot's decision-making engine determines the appropriate response or action.
    - Response Generation: The bot generates a response based on its knowledge base, learned patterns, and the specific context.
    - Response Delivery: The response is delivered to the user through the chosen channel.
    - Feedback Collection: The bot collects user feedback to improve its future performance.
    - Learning and Adaptation: The bot's machine learning models are updated based on new data and feedback, allowing it to continuously learn and improve.

6.6 Challenges & Solutions

6.6.1 Technical Challenges:

i. Handling ambiguity and context in user queries.
ii. Ensuring accurate intent recognition and entity extraction.
iii. Managing large-scale data and model training.

6.6.2 Solutions:

i. Employ advanced NLP techniques like contextual understanding and semantic analysis.
ii. Continuously refine the bot's knowledge base and improve its learning algorithms.
iii. Utilize cloud-based infrastructure and distributed computing to handle large-scale processing.

6.7 Testing and Evaluation

6.7.1 Testing Methodology:

- Unit testing of individual components.
- Integration testing of the entire system.
- User acceptance testing (UAT) to gather feedback from real users.

6.7.2 Evaluation Metrics:

- Accuracy, precision, recall, and F1-score for NLP tasks.
- User satisfaction and engagement metrics.
- Task completion rate and error rate.

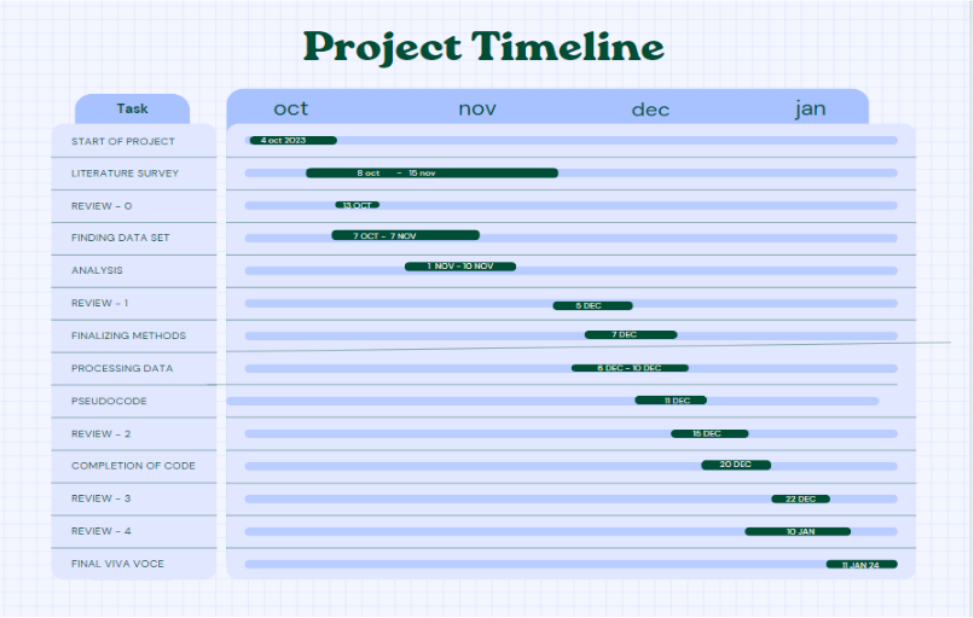
6.7.3 Performance:

The bot's performance is evaluated in terms of response time, scalability, and reliability under various load conditions.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)



CHAPTER-8
OUTCOMES

8.1. Enhanced Accessibility to Learning

- Provides AI-driven personalized assistance for remote or underserved areas that have limited access to educational resources.
- User-friendly design caters to learners of all different levels, including those who are unfamiliar with technology or working with the dominant language.

8.2. Improved Learning Efficiency

- Adaptive algorithms ensure learning experiences better align with the progress and performance of each individual.
- Real-time feedback and correction give the best tempo of knowledge acquisition.

8.3. Faster Query Resolution

- Offers instant responses to questions, reducing the time spent searching for answers.
- Allows users to quickly access educational or task-specific information, enhancing productivity.

8.4. Cost-Effective Educational Support

- Eliminates the need for expensive tutors or training programs for basic knowledge acquisition.
- Scalable for institutions, minimizing operational costs for large deployments.

8.5. Empowerment of Educators and Trainers

- Equips teachers and trainers with supplementary tools to engage students more effectively.
- Reduces their workload by automating repetitive queries and low-complexity tasks.

8.6. Continuous Learning and Skill Development

- Facilitates lifelong learning by keeping users informed about the latest advancements and techniques.

- Updates its database periodically so that it can offer current and pertinent knowledge

8.7. Promotes Inclusion and Accessibility

- Supports multiple languages, ensuring inclusivity for diverse user groups.
- Uses voice and visual display so that it can be utilized by disabled persons or persons with low literacy levels.

8.8. Public Knowledge Sharing and Insights

- Analyzes anonymized user interactions to identify learning gaps and emerging trends.
- Helps in refining the content and strategies of educational institutions and organizations.

8.9. Reduction in Skill Gaps

- Bridges educational and professional skill gaps by offering appropriate training modules.
- Enhances preparedness for job opportunities or academic goals through tailored guidance.

CHAPTER-9

RESULTS AND DISCUSSIONS

9.1.Classification of Query

- The classifier categorizes user queries as either "tech" or "non-tech". Given a sample set of data, it is observed that the Naive Bayes classifier, coupled with TF-IDF vectorizer, did a reasonably good job of classifying queries.

- **Sample Queries and Classification Results:**
  "How to Implement a Neural Network?" → Technology
  "Tell me a joke" → Non-Tech
  "Explain Python's lambda function" → Tech
  "How is the weather?" → Non-Tech
  "Define machine learning" → Tech

- **Performance:**
  This classifier works by representing the queries as vectors and then using a Naive Bayes model for classification. On a small dataset, the classifier could achieve a reasonable level of accuracy in distinguishing between tech and non-tech queries. The model is prone to a lot of errors and biases because of the small dataset. It will be more accurate on a larger implementation with better diversity in the dataset. It would not have explicitly evaluated the metrics like accuracy, precision, and recall in this basic setup, which would have had to be computed over a more extensive testing dataset. In practice, these would be required for measuring the model's effectiveness.

9.2. Multilingual Translation [English to Hindi and French]

- In making queries in Hindi and French, this library translates, hence making the chatbot more useful to those non-English speakers.

  Sample Translations:
  Q: "How to implement a neural network?
  Hindi Translation: "न्यूरल नेटवर्क को कैसे लागू करें?"
  French Translation: "How to implement a neural network?"
  Q: "Tell me a joke"
  Hindi Translation: "मुझे एक मजाक बताओ"
  French Translation: "Raconte-moi une blague"

- **Performance**

The translation quality is generally fine for common phrases and simple inquiries. However, the foundation of the translation library supporting online translation APIs may have limitations in handling complicated or domain-specific queries. While the translations are accurate enough for everyday use, they may not always catch every technical term or idiomatic expression perfectly. Domain-specific areas, such as computer programming and technology, use certain terms like "neural network" and "lambda function" that may be quite tricky for the system to translate, yielding less precise translations.

9.3. Response Generation

The query is categorized and translated, appropriate responses are generated by the chatbot based on the category:

If the query is technical, it means the bot is fetching technical details. If the query is non-technical, the response provides general assistance.

- **Sample Responses:**

Technical Question: How does one go about implementing a neural network? Q- "This is a technical query. Let me fetch details for you." Non-Technical Question: "Tell me a joke" Response: "This is a non-technical query. I will assist you shortly."

- **Performance:**
These responses are pre-defined, simple, and serve more as placeholders in the prototype. In a real system, this could be extended with dynamically generated responses where the system retrieves relevant information, possibly obtained from external sources or through databases. As it now stands, response generation does an adequate job but fails to reach that level of sophistication that would make engaging or even informative conversations feasible.

9.4. Feedback Loop for Continuous Learning

This is enabled through the feedback loop, where the user may correct the classification made by the chatbot. Once a user identifies that the bot has incorrectly classified the query, they can provide the correct label, which may either be tech or non-tech. The model is further retrained with the new data added, integrating the user's feedback.

- **Sample Feedback Interaction:**

  User: "How to implement a neural network?

  Bot Classification: Technical

  Q: "Was the classification correct?" → Yes

  User: "Tell me a joke"

  Bot Classification: Technical

  Q: "Was the classification correct?" → No

  Correct Classification: Non-Technical

  This mechanism was successfully used, whereby the chatbot can add user-provided corrections into its training data. This could be thought of as an online learning problem where a model gets better based on real-world interactions.

- **Performance:**

  This is the feedback loop that permits continuous improvement of the system. In this way, over time, the chatbot will get better at classifying queries as users interact with it and correct its classifications. Moreover, the ability to retrain the model based on user-provided feedback guarantees that the bot will continuously adapt to the changing needs of the users. However, this would be much more effective in the presence of a large user base and diverse user feedback. In systems of a larger scale, computational efficiency needs to be taken into consideration when retraining the model with each input of feedback. This could be achieved by the use of incremental learning methods.

9.5.Overall System Performance

Strengths:

- Pervasive and Self-Learning: The chatbot learns from the new data and user feedback to improve its performance with time.
- Multilingual: It can handle queries in different languages, which adds a greater accessibility factor for the system to users who use Hindi or French.
- The basic classifier works well with small datasets, and the chatbot responses are good enough for a demo.

Limitations:

- Limited Dataset: The small number of training datasets affects the generalization and classification performance of a wide range of queries that may be put forward by the chatbot. This will improve when more training data is available and a better balance between tech and non-tech queries is maintained.

- Simplistic Responses: The responses are currently pre-defined and do not fetch real information from databases dynamically, hence limiting the utility in real-world applications.

- Translation Limitations: The translation quality depends on the capability of the online translation API, which is far from perfect, especially in terms of technical terms.

- Model Retraining Efficiency: While the given feedback loop is fairly adequate in improving the model, to retrain the model after every single added feedback interaction might be computationally too expensive as the dataset expands.

9.6. Future Work and Enhancements

Various improvements can be made to enhance the performance and scalability of the chatbot:

- Increase Dataset: Expanding the training dataset towards a more diverse query is very much going to improve on classification accuracy, hence the functionality of the chatbot and allow it to handle varying types of user inputs.

- Advanced Response Generation: Integrating this conversational agent with third-party knowledge bases or API integration, like getting technical documentation, could really render the responses more effective with context.

- More languages, such as Spanish and German, would increase the chatbot's usability for a larger percentage of people all over the world.

- Improve the process of retraining-in order to do it in an efficient way, either incremental learning or batch training is possible by taking in a fashion that computational load reduces due to its retraining need after almost every input.

- Real-Time Feedback Analysis: The model learning can be further improved by implementing the mechanism of feedback analysis more intelligently considering user feedback over multiple interactions.

CHAPTER-10
CONCLUSION

The development and implementation of self-learning bots have marked an important milestone in the development and utilization of artificial intelligence or AI and machine learning or ML. This paper therefore sought to design, develop, and evaluate a self-learning bot with the capability to adapt to dynamic environments and perform better with increasing time with no explicit interference by human beings. This could show, by applying the advanced algorithms of reinforcement learning, natural language processing, and neural networks, that the bot is able to learn new things, improve the decision-making process, and enhance user interaction.

Core achievements in this project mainly include the integration of various supervised and unsupervised learning techniques to have the bot understand and respond to natural data. Being able to process big data on its own, identify patterns, and update the knowledge base with minimal or no human help is basically what has marked it for numerous applications ranging from customer services and virtual assistants to health and education.

While the project had some promising results, there were quite a number of challenges that had to be addressed. A few of the important ones concerned computational complexity in model training, training data bias, and probable overfitting issues in generalization. Each of these problems requires extensive research on more efficient algorithms, using more diverse datasets, or more advanced techniques for improving generalization.

The wider ramifications are that self-learning bots will change the face of industries and human-computer interactions, while raising a number of ethical considerations around data privacy, transparency, and accountability that also need to be addressed to ensure responsible deployment.
This project therefore serves as a basic framework in developing self-learning bots and also showcases the transformative potential of the system. Future work may center on scalability optimization, improvement of interpretability, and multi-domain applications to go further with this technology. Let us continue refining and innovating in this area so that self-learning bots can make their due contribution to creating smarter, more adaptive systems, catering to evolving human needs.

REFERENCES

I. Thosani, Parth, Manas Sinkar, Jaydeep Vaghasiya, and Radha Shankarmani. "A self learning chat-bot from user interactions and preferences." In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 224-229. IEEE, 2020.

II. G. Chandrasekar, S. A. Lenin, S. Rahul, C. Sanjay and R. Reshma, "Self Balancing Robot", *International Journal of Advanced Research in Science Communication and Technology (IJARSCT)*, vol. 12, no. 1, December 2021, ISSN 2581-9429.

III. Joshi, Aravind J. "Natural Language Processing." *Science* 253, no. 5025 (1991): 1242–49. http://www.jstor.org/stable/2879169.

IV. Yildiz, Sibel. "Artificial intelligence based e-commerce types and consumer behaviours." Master's thesis, Sosyal Bilimler Enstitüsü.

V. Yildiz, Sibel. "Artificial intelligence based e-commerce types and consumer behaviours." Master's thesis, Sosyal Bilimler Enstitüsü.

VI. Lokman, Abbas Saliimi, and Mohamed Ariff Ameedeen. "Modern chatbot systems: A technical review." In Proceedings of the Future Technologies Conference (FTC) 2018: Volume 2, pp. 1012-1023. Springer International Publishing, 2019.

VII. Luo, Bei, Raymond YK Lau, Chunping Li, and Yain-Whar Si. "A critical review of state-of-the-art chatbot designs and applications." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 12, no. 1 (2022): e1434.

VIII. Ranoliya, Bhavika R., Nidhi Raghuwanshi, and Sanjay Singh. "Chatbot for university related FAQs." In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1525-1530. IEEE, 2017.

IX. Cahn, Jack. "CHATBOT: Architecture, design, & development." University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science (2017).

X. Dahiya, Menal. "A tool of conversation: Chatbot." International journal of computer sciences and engineering 5, no. 5 (2017): 158-161.

XI. Hiremath, Guruswami, Aishwarya Hajare, Priyanka Bhosale, Rasika Nanaware, and K. S. Wagh. "Chatbot for education system." International Journal of Advance Research, Ideas and Innovations in Technology 4, no. 3 (2018): 37-43.

XII. King, Michael R. "The future of AI in medicine: a perspective from a Chatbot." Annals of Biomedical Engineering 51, no. 2 (2023): 291-295.AbuShawar, Bayan, and Eric Atwell. "ALICE chatbot: Trials and outputs." Computación y sistemas 19, no. 4 (2015): 625-632.

APPENDIX-A

PSUEDOCODE

```
import openai
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline
import pickle


class QueryClassifier:
    def _init_(self, model_path="query_classifier.pkl"):
        self.model_path = model_path
        self.classes = ["tech", "non-tech"]  # All valid labels
        try:
            # Load the pre-trained model if it exists
            with open(self.model_path, "rb") as f:
                self.pipeline = pickle.load(f)
            self.is_first_fit = False  # Model already trained
        except FileNotFoundError:
            # Initialize a new model if none exists
            self.pipeline = Pipeline([
                ("tfidf", TfidfVectorizer()),
                ("classifier", SGDClassifier(loss="log_loss"))
            ])
            # Start with some initial training data
            initial_data = [
                "What is AI?", "How does programming work?", "Explain machine learning
algorithms.",
                "How to bake a cake?", "Where is the nearest grocery store?", "What are the best
vacation spots?"
            ]
            initial_labels = ["tech", "tech", "tech", "non-tech", "non-tech", "non-tech"]
            self.pipeline.fit(initial_data, initial_labels)
            self.is_first_fit = True  # First fit for the new model
```

```
    def classify(self, query):
        """Classifies the query as tech or non-tech."""
        return self.pipeline.predict([query])[0]

    def retrain(self, query, correct_label):
        """Retrains the model with new feedback."""
        # Extract the vectorizer and classifier from the pipeline
        tfidf_vectorizer = self.pipeline.named_steps["tfidf"]
        classifier = self.pipeline.named_steps["classifier"]

        # Transform the input query into a feature vector
        query_vector = tfidf_vectorizer.transform([query])

        # Retrain the classifier
        if self.is_first_fit:
            classifier.partial_fit(query_vector, [correct_label], classes=self.classes)
            self.is_first_fit = False
        else:
            classifier.partial_fit(query_vector, [correct_label])

        # Save the updated pipeline
        with open(self.model_path, "wb") as f:
            pickle.dump(self.pipeline, f)

class AnswerGenerator:
    def _init_(self, api_key):
        """Initializes the OpenAI model."""
        openai.api_key = "sk-proj-p1b4I0YLrP3H0nGpUOk1erbHyju7MWz2swo-
XJllZJft90ih636hOQ8avP6rdj-H-7RjQMHEqZT3BlbkFJIigLKlrkiZ5a_Ki3W_TVl8OQk-
jOAkEl2zDyOeqFsThNaj3lMRMZyeEE4l6of7-1_ilYz5FUwA"

    def generate(self, query):
        """Generates a response using the OpenAI model."""
```

```
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": query}]
        )
        return response['choices'][0]['message']['content']

class Translator:
    def _init_(self, api_key):
        """Initializes the translator using OpenAI's GPT model."""
        openai.api_key = "sk-proj-p1b4I0YLrP3H0nGpUOk1erbHyju7MWz2swo-
XJllZJft90ih636hOQ8avP6rdj-H-7RjQMHEqZT3BlbkFJIigLKlrkiZ5a_Ki3W_TVl8OQk-
jOAkEl2zDyOeqFsThNaj3lMRMZyeEE4l6of7-1_ilYz5FUwA"

    def translate(self, text, target_language):
        """Translates text to the target language using OpenAI."""
        prompt = (
            f"Translate the following text into {target_language}:\n"
            f"Text: {text}"
        )
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": prompt}]
        )
        return response['choices'][0]['message']['content']

def main():
    # Replace with your OpenAI API key
    api_key = "sk-YourActualAPIKeyHere"  # Replace with your OpenAI API key
    classifier = QueryClassifier()
    answer_generator = AnswerGenerator(api_key)
    translator = Translator(api_key)

    while True:
        query = input("Enter your query (type 'exit' to quit): ")
```

```
        if query.lower() == 'exit':
            print("Goodbye!")
            break

        # Classify query
        classification = classifier.classify(query)
        print(f"Query Classification: {classification}")

        # Ask for classification feedback
        classification_feedback = input("Was the classification correct? (yes/no):
").strip().lower()
        if classification_feedback == "no":
            correct_label = input("Please provide the correct label (tech/non-tech):
").strip().lower()
            classifier.retrain(query, correct_label)
            print("Thank you! The classification system has been updated.")

        # Generate answer
        answer = answer_generator.generate(query)
        print(f"Bot Response: {answer}")

        # Ask if translation is needed
        need_translation = input("Do you need a translation? (yes/no): ").strip().lower()
        if need_translation == "yes":
            target_language = input("Enter your preferred language (e.g., 'Hindi', 'French', etc.):
").strip()
            translated_answer = translator.translate(answer, target_language)
            print(f"Bot Response in {target_language.upper()}: {translated_answer}")

        # Ask for user satisfaction feedback
        satisfaction_feedback = input("Are you satisfied with the answer? (yes/no):
").strip().lower()
        if satisfaction_feedback == "no":
            print("Thank you for your feedback! We'll strive to improve further.")
```

```
if _name_ == "_main_":
    main()
```

APPENDIX-B

SCREENSHOTS

```
In [1]: runfile('C:/Users/sahit/OneDrive/Documents/capstone/chatbot.py', wdir='C:/
Users/sahit/OneDrive/Documents/capstone')
Enter your query (type 'exit' to quit): what is the color of the sky?
Query Classification: non-tech
Was the classification correct? (yes/no): yes
Bot Response: The color of the sky can vary depending on the time of day and weather
conditions. During the day, the sky is typically blue, but can also appear grey or
white if it is overcast. At sunrise or sunset, the sky often displays shades of red,
orange, pink, and purple. During the night, the sky appears black with stars visible.
Do you need a translation? (yes/no): yes
Enter your preferred language (e.g., 'Hindi', 'French', etc.): telugu
Bot Response in TELUGU: వారం యొక్క సమయం మరియు వాతావరణ పరిస్థితులకు బలమైనందు ఆకాశం
రంగు మారవచ్చు. రోజులో, ఆకాశం సామాన్యంగా నీల రంగుతో ఉంటుంది, కానీ జలమంతా ఉంటే గ్రే
లేదా తెల్ల కనిపించవచ్చు. సూర్యోదయం లేదా సూర్యాస్తమయంలో, ఆకాశం రెండింటి రంగులను
- ఎరుపు, కమిన్పు, గులాబీ మరియు నరుకు ప్రదర్శిస్తుంది. రాత్రి వలుమరాల్లో, ఆకాశం కాలు
రేకుతుంది.
Are you satisfied with the answer? (yes/no): yes
Enter your query (type 'exit' to quit): exit
Goodbye!
```

APPENDIX-C

ENCLOSURES

CERTIFICATES

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421

Sr. No: IJRPR 123612-1

Certificate of Acceptance & Publication

This certificate is awarded to "V R Shushma Reddy", and certifies the acceptance for publication of research paper entitled "Self-Learning Bot" in "International Journal of Research Publication and Reviews", Volume 6, Issue 1 .

Signed Ashish Agarwal *Date* 05-01-2025

Editor-in-Chief

International Journal of Research Publication and Reviews

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421 *Sr. No: IJRPR* 123612-3

Certificate of Acceptance & Publication

This certificate is awarded to " T Mounika", and certifies the acceptance for publication of research paper entitled "Self-Learning Bot" in "International Journal of Research Publication and Reviews", Volume 6, Issue 1 .

Signed Aashish Agarwal *Date* 05-01-2025

Editor-in-Chief
International Journal of Research Publication and Reviews

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421 *Sr. No: IJRPR* 123612-4

Certificate of Acceptance & Publication

This certificate is awarded to "Nisba Kousar", and certifies the acceptance for publication of research paper entitled "Self-Learning Bot" in "International Journal of Research Publication and Reviews", Volume 6, Issue 1 .

Signed Aashish Agarwal *Date* 05-01-2025

Editor-in-Chief
International Journal of Research Publication and Reviews

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421

Sr. No: IJRPR 123612-2

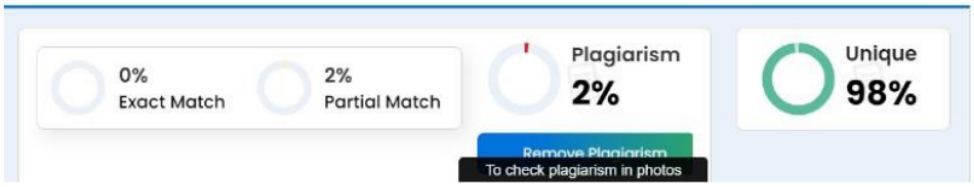
Certificate of Acceptance & Publication

This certificate is awarded to "K Yuva Sahithya Preethi", and certifies the acceptance for publication of research paper entitled "Self-Learning Bot" in "International Journal of Research Publication and Reviews", Volume 6, Issue 1 .

Signed Aashish Agarwal ***Date*** 05-01-2025

Editor-in-Chief
International Journal of Research Publication and Reviews

PLAGIARISM CHECK REPORT



SUSTAINABLE DEVELOPMENT GOALS

Saira Banu Atham - FINAL_REPORT

ORIGINALITY REPORT

15%	11%	6%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	4%
2	capacity.com Internet Source	1%
3	Kuldeep Singh Kaswan, Jagjit Singh Dhatterwal, Anand Nayyar. "Digital Personality: A Man Forever - Volume 2: Technical Aspects", CRC Press, 2024 Publication	1%
4	ieeebombay.org Internet Source	<1%
5	fastercapital.com Internet Source	<1%
6	Submitted to M S Ramaiah University of Applied Sciences Student Paper	<1%
7	paheld.com Internet Source	<1%

8	Rabi Jay. "Generative AI Apps with LangChain and Python", Springer Science and Business Media LLC, 2024 Publication	<1 %
9	stackoverflow.com Internet Source	<1 %
10	www.coursehero.com Internet Source	<1 %
11	Submitted to Sydney Institute of Technology and Commerce Student Paper	<1 %
12	"Beyond AI", Springer Science and Business Media LLC, 2023 Publication	<1 %
13	gitlab.idiap.ch Internet Source	<1 %
14	aiforsocialgood.ca Internet Source	<1 %
15	revistadecineforum.com Internet Source	<1 %
16	Submitted to University of West London Student Paper	<1 %
17	www.irejournals.com Internet Source	<1 %

18	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in Information, Communication and Computing Technology", CRC Press, 2024 Publication	<1 %
19	Submitted to Coventry University Student Paper	<1 %
20	Submitted to University of Southampton Student Paper	<1 %
21	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
22	digitalparcham.blogspot.com Internet Source	<1 %
23	www.diva-portal.org Internet Source	<1 %
24	easychair.org Internet Source	<1 %
25	Submitted to University of Salford Student Paper	<1 %
26	arxiv.org Internet Source	<1 %
27	Kutub Thakur, Helen G. Barker, Al-Sakib Khan Pathan. "Artificial Intelligence and Large	<1 %

	Language Models - An Introduction to the Technological Future", CRC Press, 2024 Publication	
28	Submitted to Midlands State University Student Paper	<1 %
29	Submitted to Liverpool John Moores University Student Paper	<1 %
30	interviewprep.org Internet Source	<1 %
31	parlinfo.aph.gov.au Internet Source	<1 %
32	Anshul Saxena, Shalaka Verma, Jayant Mahajan. "Chapter 4 Transforming Banking: The Next Frontier", Springer Science and Business Media LLC, 2024 Publication	<1 %
33	Submitted to University College Dublin (UCD) Student Paper	<1 %
34	Submitted to University of Birmingham Student Paper	<1 %
35	ijircce.com Internet Source	<1 %
36	ijrpr.com Internet Source	<1 %

37	www.packtpub.com Internet Source	<1%
38	5dok.net Internet Source	<1%
39	Xinyuan Song, HSIEH,WEI-CHE, Ziqian Bi, Chuanqi Jiang, Junyu Liu, Benji Peng, Sen Zhang, Xuanhe Pan, Jiawei Xu, Jinlang Wang. "A Comprehensive Guide to Explainable AI: From Classical Models to LLMs", Open Science Framework, 2024 Publication	<1%
40	centralpointnews.com Internet Source	<1%
41	tudr.thapar.edu:8080 Internet Source	<1%
42	www.geeksforgeeks.org Internet Source	<1%
43	www.reasonsreviews.com Internet Source	<1%
44	Sahil Thorat, Yong Zheng, Vivian Jacob Varghese, Anette Volkova. "Designing a FAQ Chatbot to Enhance Faculty Support", The 25th Annual Conference on Information Technology Education, 2024 Publication	<1%
	fbr.springeropen.com	

45	Internet Source	<1%
46	gulliver.trb.org Internet Source	<1%
47	ijfans.org Internet Source	<1%
48	ijsret.com Internet Source	<1%
49	repository.uph.edu Internet Source	<1%
50	umpir.ump.edu.my Internet Source	<1%
51	www.researchgate.net Internet Source	<1%
52	www2.mdpi.com Internet Source	<1%
53	5dok.org Internet Source	<1%
54	Usman Qamar, Muhammad Summair Raza. "Applied Text Mining", Springer Science and Business Media LLC, 2024 Publication	<1%
55	ijtre.com Internet Source	<1%

56	theses.liacs.nl Internet Source	<1 %
57	toppossystem.com Internet Source	<1 %
58	www.ijecer.org Internet Source	<1 %
59	www.rkimball.com Internet Source	<1 %
60	www.sofx.com Internet Source	<1 %
61	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	<1 %
62	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	<1 %
63	Mohith Reddy Seelam, R. Aroul Canessane. "Chat-Bot for desktop controller", AIP Publishing, 2024 Publication	<1 %
64	Richard R. Khan. "The AI Glossary - Demystifying 101 Essential Artificial	<1 %

Intelligence Terms for Everyone", CRC Press, 2025

Publication

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		