ULTIMATRIX CHATBOT
MINI PROJECT

Submitted by

EGADARSHAN S 412522243047

VIGNESH R 412522243174

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SRI SAIRAM ENGINEERING COLLEGE

(An Autonomous Institution; Affiliated to Anna University, Chennai - 600 025)

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2024

SRI SAIRAM ENGINEERING COLLEGE
(An Autonomous Institution; Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report “ULTIMATRIX CHATBOT” is the Bonafide work of EGADARSHAN S (412522243047), and VIGNESH R (412522243174) who carried out the MINI PROJECT under my supervision.

SIGNATURE	SIGNATURE
Staff in Charge	Head of the Department
MS. Varalakshmi K	Dr. Swagata Sarkar
Assistant Professor	Professor & Head

Submitted for project Viva – Voce Examination held on _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

"Let our advance worrying become advance thinking and planning."
-Winston Churchill.

Such a successful personality is our beloved founder Chairman, **Thiru. MJF. Ln. LEO MUTHU**. At first, we express our sincere gratitude to our beloved chairman through prayers, who in the form of a guiding star has spread his wings of external support with immortal blessings.

We express our gratitude to our CEO **Dr. J. Sai Prakash Leomuthu** for having given us spontaneous and whole-hearted encouragement for completing this project.

We express our sincere thanks to our beloved Principal, **Dr. J. Raja** for having given us spontaneous and whole-hearted encouragement for completing this project.

We are indebted to our Head of the Department **Dr. Swagata Sarkar** for her support during the entire course of this project work.

We express our gratitude and sincere thanks to our subject (20AIPC503, Natural Language Processing and Chatbot) handling staff, Assistant Professor, **Ms. Varalakshmi K** for her valuable suggestions and constant encouragement for successful completion of this project.

We thank all the teaching and non-teaching staff members of the Department of Artificial Intelligence and Data Science and all others who contributed directly or indirectly for the successful completion of the project.

TABLE OF CONTENTS

ABSTRACT

This project is a chatbot application that utilizes deep learning and Natural Language Processing (NLP) techniques to enable intelligent and context-aware user interactions. The chatbot processes user input, predicts its intent, and generates accurate and relevant responses. By employing a pre-trained neural network built with Keras, the system transforms raw text into numerical representations through tokenization, lemmatization, and a Bag-of-Words (BoW) model, ensuring precise intent classification and response generation.

The chatbot is powered by a JSON-based intents file, which defines categories of user input and their corresponding responses. This structured format allows for easy customization and scalability, enabling developers to add or modify intents as needed. The system applies an error threshold to filter low-confidence predictions, ensuring high-quality responses. Additionally, dynamic placeholders are incorporated into the responses, allowing the chatbot to personalize interactions, such as addressing users by their names when provided.

This application is designed to provide a seamless and interactive experience for users, with features like conversational history and personalized responses enhancing engagement. Its modular design makes it adaptable for various applications, including customer service, virtual assistants, and educational tools. By integrating advanced NLP and deep learning techniques, this project demonstrates the potential of AI-powered chatbots to transform automated communication.

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVE

The objective of this project is to develop an intelligent chatbot capable of understanding and responding to user inputs through advanced Natural Language Processing (NLP) and deep learning techniques. The system leverages libraries such as Keras for building and loading the neural network model, NumPy for numerical computations, and NLTK (Natural Language Toolkit) for text preprocessing tasks like tokenization and lemmatization. Additionally, Streamlit is used to create an interactive user interface, and libraries like Pickle and JSON facilitate data handling for intents, words, and model files. This project is designed to be flexible and scalable, allowing for seamless integration into various applications such as customer support, virtual assistants, and educational platforms, while maintaining an engaging and user-friendly conversational experience.

1.2. OUTLINE OF THE REPORT

This project involves building an intelligent chatbot application, designed to deliver accurate, context-aware, and personalized responses. The process begins with structuring user intents and responses in a JSON file, forming the backbone of the chatbot’s knowledge base. Advanced NLP techniques, including tokenization and lemmatization via NLTK, are employed to preprocess user input, which is then transformed into a Bag-of-Words (BoW) representation using NumPy. A pre-trained neural network model, developed and loaded with Keras, predicts the user's intent with high accuracy. Dynamic and personalized responses are generated by mapping the predicted intent to corresponding replies in the intents file, facilitated by libraries like Pickle and JSON. An error threshold filters low-confidence predictions to maintain response quality. The chatbot features a user-friendly interface, enabling real-time interactions while preserving chat history to provide a coherent conversational flow. This modular and scalable design highlights the potential of integrating NLP, deep learning, and AI in creating versatile applications across customer service, virtual assistance, and education.

1.3. SCOPE OF THE PROJECT

The scope of this project is to create a versatile and intelligent chatbot capable of understanding and responding to user queries in a human-like manner.
It is designed to handle a wide range of conversational tasks, including answering FAQs, providing personalized responses, and engaging users dynamically.
The chatbot is highly adaptable and can be deployed across various domains such as customer support, virtual assistance, education, and e-commerce
The project includes the following key areas:

- Intelligent Chatbot: Develop a chatbot capable of understanding and responding to user inputs with context-aware, personalized responses.
- Domains of Use: Applicable in multiple domains such as customer support, virtual assistance, education, and e-commerce.
- Modularity & Customization: Easy to customize and extend by updating the intents file, making it adaptable to specific needs and industries.
- Advanced NLP Techniques: Utilizes tokenization, lemmatization, and Bag-of-Words (BoW) models to process and interpret user input efficiently.
- Deep Learning Integration: Leverages a pre-trained neural network model built with Keras to accurately predict user intent.
- Dynamic Responses: Generates personalized, dynamic responses based on the user's input, including handling specific use cases like name insertion.
- Error Threshold for Accuracy: Incorporates an error threshold to filter low-confidence predictions and ensure high-quality responses.
- Scalability: Scalable to accommodate new features, such as sentiment analysis or multilingual support, and capable of being expanded for more complex applications.
- Real-time Interaction: Provides a user-friendly interface for seamless, real-time interactions with the chatbot, maintaining chat history for context.
- Foundation for Future Development: Acts as a solid foundation for building more advanced conversational AI systems with additional features like advanced context handling.

CHAPTER 2
SDG JUSTIFICATION

SDG 4: Quality Education

- This chatbot application supports inclusive and equitable quality education by providing an interactive learning assistant that can assist learners in real time. It can be adapted for educational purposes, offering tutoring, answering questions, and delivering personalized educational content. The scalability and adaptability of the chatbot make it an ideal tool for both formal and informal education settings, promoting lifelong learning opportunities for all.

SDG 8: Decent Work and Economic Growth

- The chatbot enhances economic growth and decent work by streamlining customer support and improving service efficiency across industries such as retail, healthcare, and finance. By automating responses to customer queries, businesses can reduce operational costs, improve customer satisfaction, and allocate human resources to more complex tasks. This project also demonstrates the potential for AI and automation to create new job opportunities in technology, customer service, and AI development.

SDG 9: Industry, Innovation, and Infrastructure

- This project aligns with building resilient infrastructure and fostering innovation by advancing the development of conversational AI and intelligent systems. The chatbot's use of deep learning, NLP, and AI-driven technology promotes innovation in industries such as e-commerce, healthcare, and customer service. Furthermore, it serves as a scalable model that can be integrated into existing infrastructure, driving further advancements in automation and intelligent systems across various sectors.

CHAPTER 3
PROPOSED SOLUTION

The solution involves creating an intelligent, context-aware chatbot utilizing NLP and deep learning with a pre-trained Keras model. It focuses on predicting user intent and generating personalized responses, incorporating tokenization, lemmatization, Bag-of-Words representation, error filtering, and dynamic intent management for a scalable, customizable system applicable to various uses.

Proposed Solution: Key Components

1. Natural Language Processing (NLP)
    - Tokenization and Lemmatization: Using NLTK for preprocessing user input by breaking it into individual words (tokenization) and reducing words to their base form (lemmatization).
    - Bag-of-Words (BoW): Transforming the user input into a numerical representation for the machine learning model.
2. Deep Learning Model
    - Keras Neural Network: Utilizing a pre-trained deep learning model built with Keras to classify user intents based on the processed input.
    - Intent Prediction: The model predicts the intent of the user's message, which drives the appropriate response generation.
3. Intent Management
    - JSON-based Intents File: Storing user intents and corresponding responses in a JSON file, making it easy to add or modify intents and responses.

4. Error Filtering and Confidence Threshold
    - Threshold for Confidence: Implementing a confidence threshold to filter out low-confidence predictions, ensuring that only high-probability responses are used.
5. Chat History and Context Management
    - Chat History: Maintaining a conversation history for better context handling, enabling the chatbot to remember previous interactions and provide more coherent responses.
6. User Interface
    - Interactive Interface: A user-friendly interface that allows users to interact with the chatbot in real-time, providing seamless communication.
    - Scalable Integration: The chatbot is designed to be easily integrated into existing systems or platforms, ensuring broad applicability across various industries

3.1. SOURCE CODE:

```python
import random
import numpy as np
import pickle
import json
import streamlit as st
from keras.models import load_model
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import RegexpTokenizer

# Initialize necessary components
lemmatizer = WordNetLemmatizer()
tokenizer = RegexpTokenizer(r'\w+')
model = load_model(r"C:\Users\egadarshan\Desktop\NLP\chatbot\chatbot_model1.h5")
words = pickle.load(open(r"C:\Users\egadarshan\Desktop\NLP\chatbot\words.pkl", "rb"))
classes = pickle.load(open(r"C:\Users\egadarshan\Desktop\NLP\chatbot\classes.pkl", "rb"))

# Load the intents file
with open(r"C:\Users\egadarshan\Desktop\NLP\chatbot\intents.json") as file:
    intents = json.load(file)

# Clean up sentence for tokenization
def clean_up_sentence(sentence):
    sentence_words = tokenizer.tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# Convert sentence into a bag of words
def bow(sentence, words, show_details=True):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                bag[i] = 1
    return np.array(bag)
```

```python
# Predict the intent of the sentence
def predict_class(sentence, model):
    p = bow(sentence, words, show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = [{"intent": classes[r[0]], "probability": str(r[1])} for r in results]
    return return_list

# Get response from intents file
def getResponse(ints, intents_json):
    if not ints:
        return "No valid intent found."
    tag = ints[0]["intent"]
    list_of_intents = intents_json["intents"]
    for i in list_of_intents:
        if i["tag"] == tag:
            return random.choice(i["responses"])
    return "Sorry, I don't understand that."

# Define the chatbot response
def chatbot_response(user_input):
    if user_input.startswith('my name is'):
        name = user_input[11:]
        ints = predict_class(user_input, model)
        res = getResponse(ints, intents)
        return res.replace("{n}", name)
    elif user_input.startswith('hi my name is'):
        name = user_input[14:]
        ints = predict_class(user_input, model)
        res = getResponse(ints, intents)
        return res.replace("{n}", name)
    else:
        ints = predict_class(user_input, model)
        return getResponse(ints, intents)

```

```python
st.title("Chatbot Interface")
st.markdown("Welcome to the chatbot! Ask me anything.")

# Chat history
if "messages" not in st.session_state:
    st.session_state.messages = []

# Input box
user_input = st.text_input("You:", "")

if user_input:
    # Display the user's input
    st.session_state.messages.append({"role": "user", "message": user_input})

    # Get the bot's response
    bot_response = chatbot_response(user_input)
    st.session_state.messages.append({"role": "bot", "message": bot_response})

# Display chat history
for message in st.session_state.messages:
    if message["role"] == "user":
        st.markdown(f"**You:** {message['message']}")
    else:
        st.markdown(f"**Bot:** {message['message']}")
```

CHAPTER 4
REQUIREMENT SECIFICATION

4.1. Hardware Requirements:

Computer/Server:

- A machine with at least 4GB RAM and a multi-core processor is recommended for optimal performance
- A stable internet connection is required to load external resources (e.g., pre-trained models or datasets) and for deployment if integrated with cloud-based services.

4.2. Software Requirements:

Python:

Python 3.6 or higher is required, as the project relies on Python-based libraries for machine learning and NLP tasks.

Libraries and Dependencies:

- Keras: Build, train, and deploy deep learning models.
- TensorFlow: Backend for Keras to execute models.
- NumPy: Handle numerical computations and model inputs.
- NLTK: Perform Natural Language Processing tasks .
- Scikit-learn: Use for model evaluation and text vectorization.
- Pickle: Save and load trained models and other objects.
- JSON: Manage structured data for intents and responses.
- Streamlit: Create an interactive user interface for the chatbot.
- Pandas (optional): Manipulate data for complex handling.

4.3. Model Requirements:

The model requires a pre-trained deep learning model built with Keras and TensorFlow, which can be loaded for prediction tasks. It should be capable of handling text input preprocessed using tokenization, lemmatization, and Bag-of-Words representation. The model must be trained to predict user intent based on the input, with a confidence threshold to filter low-probability responses.

4.4. System Requirements:

Operating System:

- Compatible with Windows, macOS, and Linux. The
- system must support Python and the required libraries.

Deployment Environment:

- Local deployment on personal computers is sufficient. For advanced
- functionalities, cloud-based services can be integrated, requiring at least 8GB of RAM for server setups.

Internet Connection:

- A stable internet connection is required to load external resources (e.g., pre-trained models or datasets) and for deployment if integrated with cloud-based services.

CHAPTER 5

IMPLEMENTATION

5. IMPLEMENTATION TECHNOLOGY

5.1. Library Installations

- Install required libraries using Python's package manager pip or conda
- Keras (pip install keras)
- TensorFlow (pip install tensorflow)
- NLTK (pip install nltk)
- NumPy (pip install numpy)
- Streamlit (if used) (pip install streamlit)
- Pickle and JSON (standard Python libraries, no installation required)

5.2. Importing Required Libraries

- Import necessary libraries into the project:
- Keras for deep learning model loading and prediction.
- NLTK for text preprocessing (tokenization, lemmatization).
- NumPy for numerical data handling.
- Pickle for saving/loading model and data.
- JSON for handling structured intent and response data.

5.3. Initialization and Configuration

- Initialize the NLTK lemmatizer and tokenizer for text preprocessing.
- Load the pre-trained Keras model and Pickle files containing words and classes.
- Load the JSON file containing intents and responses.
- Set up configuration options, such as the confidence threshold for intent prediction.

5.4. Command Processing and Execution

- Process user input by tokenizing and lemmatizing the text.
- Convert processed input into a Bag-of-Words representation.
- Predict the user intent using the trained Keras model.
- Fetch the corresponding response from the intents JSON file based on predicted intent.

5.5. Interface Design and Usage

- Design a simple interface (using Streamlit or another framework) to display user inputs and chatbot responses.
- Provide a text input box for users to type queries.
- Display the chatbot's response in real-time as the conversation progresses.
- Optionally, store the conversation history for context.

5.6. Launching and Usage

- Launch the chatbot application by running the script through a command prompt or terminal.
- Open the interface in a web browser (if using Streamlit).
- Interact with the chatbot by typing queries and receiving instant responses.
- For deployment, host the chatbot on a local or cloud server for production usage.

CHAPTER 6

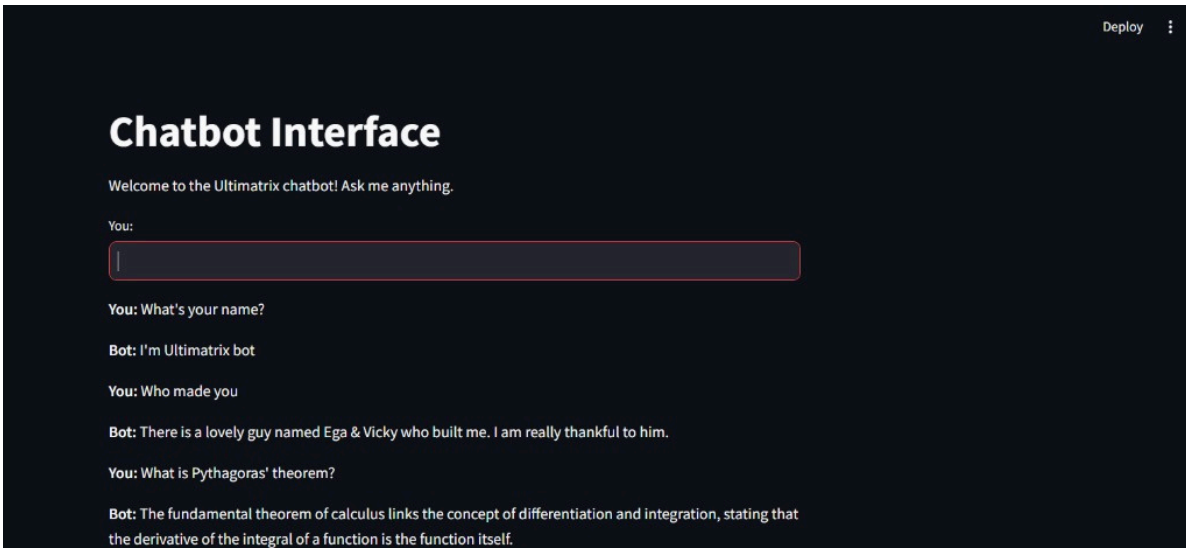
RESULT AND CONCLUSION

6.1 Results

1. Accurate Intent Prediction: The chatbot predicts user intents effectively, using a Keras deep learning model with tokenization and lemmatization.
2. Personalized Responses: It provides dynamic, personalized responses, addressing users by name and adapting based on inputs.
3. Confidence Filtering: Low-confidence predictions are filtered out, ensuring reliable, accurate responses.
4. Real-Time Interaction: The system offers real-time responses, providing a smooth user experience.
5. Modular Design: The chatbot's structure allows for easy customization and updates to handle new topics or queries.
6. Scalability: The chatbot can be expanded for more advanced features like sentiment analysis and multilingual support.
7. Efficient Resource Use: The model is optimized for quick responses and minimal computational overhead.
8. User-Friendly Interface: The interface is intuitive, making it easy for users to interact without technical knowledge.

6.2. CONCLUSION

In conclusion, the chatbot successfully implements an intelligent, real-time conversational system that accurately predicts user intent and generates personalized responses. Its modular design, scalability, and efficient resource usage make it adaptable to a wide range of applications, from customer service to education. With robust error handling, user-friendly interaction, and the ability to easily integrate new features, the chatbot demonstrates strong potential for practical deployment, offering a scalable and customizable solution for enhancing user engagement and automating interactions across various domains.



Output image

CHAPTER 7

REFERENCES

7.1. REFERENCES:

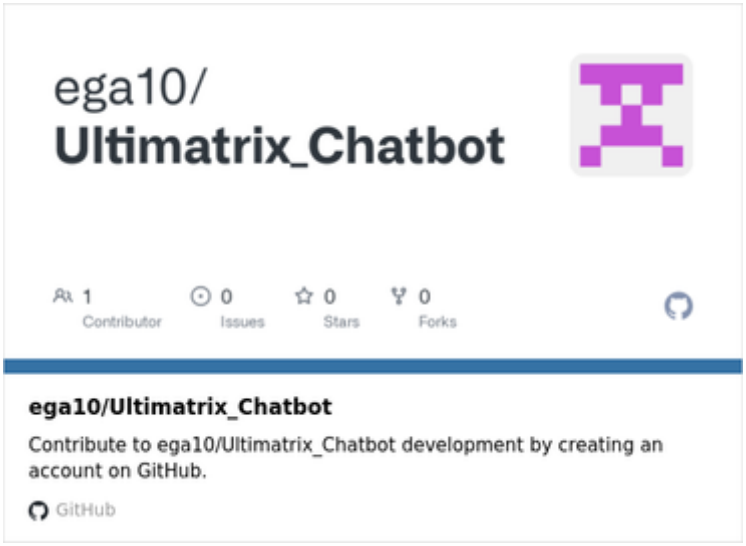
- Chollet, F. (2015). Keras: The Python deep learning library. Retrieved from https://keras.io
- Abadi, M., Barham, P., Chen, J., et al. (2016). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283.
- Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2
- Olson, E. (2018). Streamlit: A faster way to build and share data apps. Retrieved from https://streamlit.io
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. Computing in Science & Engineering, 13(2), 22-30.
- Python Software Foundation. (2024). Python documentation. Retrieved from https://www.python.org/doc/
- PyTorch Documentation. (2024). PyTorch: An open-source deep learning platform. Retrieved from https://pytorch.org/docs/stable/
- Zhang, Z., & Li, P. (2019). Natural Language Processing with deep learning: A survey. arXiv preprint arXiv:1910.05647.

APPENDIX

GitHub Repository

The source code and project files for the Chatbot Project can be found in the following GitHub repository:

https://github.com/ega10/Ultimatrix_Chatbot



The GitHub repository for this project contains the complete source code, required models, and other resources necessary to run the chatbot application. The repository serves as a version-controlled platform for managing the codebase and allows other developers to contribute, test, or modify the project.