

**DEPARTMENT OF COMPUTER SCIENCE**

**FORMAN CHRISTIAN COLLEGE**

**(A Chartered University)**

**LAHORE, PAKISTAN**

**COMP-360 (A)**

**Intro to Artificial Intelligence**

**Project Report**

**AI-Powered University Admissions Chatbot**

**Group Members:**

**Abdullah Mehtab || 241607856**

**Ali Mujtaba ||** **261910678**

**Laiba something ||** **261905484**

**Zain ul abababa ||**

**Submitted to: Miss Aasia Khanum**

**1. Introduction**

The university admissions process can overwhelm prospective students with numerous queries about programs, scholarships, and deadlines. Traditional methods of addressing these queries are often inefficient, leading to delays and communication gaps.

This project introduces an AI-powered chatbot designed to streamline the admissions process. Leveraging Python, Keras, TensorFlow, and Tkinter, the chatbot aims to deliver instant and accurate responses to frequently asked questions (FAQs). By automating the query-resolution process, the chatbot reduces administrative workload and enhances the user experience.

**Technologies Overview**

* **Keras:** A high-level neural networks API written in Python. It simplifies the creation and training of machine learning models by providing intuitive APIs and layers for constructing deep learning models.
* **TensorFlow:** A powerful open-source library that powers the neural network model by enabling efficient training and prediction of intent classifications.
* **Tkinter:** A GUI framework for Python, used to create the chatbot’s user-friendly interface, making it accessible to non-technical users.

**2. Objectives**

The chatbot’s development is guided by the following SMART goals:

* **Specific:** Provide accurate responses to admission-related queries.
* **Measurable:** Achieve at least 85% accuracy in intent classification.
* **Achievable:** Utilize open-source libraries and complete the project within the timeline.
* **Relevant:** Reduce the workload on admission staff by automating FAQs.
* **Time-bound:** Ensure project completion by deadline.

**3. Work Division**

* **Abdullah Mehtab:**
* **Role:** Dataset Preparation, Data Preprocessing, and Model Training
* Prepared and preprocessed datasets for chatbot training, achieving an 98% accuracy in intent classification.
* Trained and fine-tuned a neural network using TensorFlow and Keras.
* Documented processes related to dataset preparation and model architecture.
* Made the GUI
* (everything)
* **Ali Mujtaba**
* **Role:** Information Gathering and Interface Development
* Conducted user surveys to shape chatbot features and FAQs.
* Designed and integrated a user-friendly interface with interactive buttons and themes on paper for code implementation to be done by Abdullah
* Introduced features like conversation saving.
* **Laiba**
* **Role:** Testing and Evaluation
* Developed test cases and conducted usability testing to refine chatbot responses.
* Made the "pink/UwU" theme with visual elements for a cohesive design.
* Improved response accuracy based on user feedback.
* **Zain-ul-Abidin**
* **Role:** Suggestions and Quality Assurance
* Proofread and structured the project report to meet academic standards.
* Conducted quality assurance for code and interface components.
* Suggested basic arithmetic functionality addition.
* Ran the project on his laptop (an achievement)

**4. Prerequisite Work**

**System Setup**

* **Software Requirements:**
  + Python 3.8+ || TensorFlow 2.0 || Keras Tkinter (built-in with Python)
* **Hardware Requirements:**
  + A machine with Python environment installed

**Installation Steps**

1. Install Python: Download and install Python 3.8+ from the official website.
2. Install dependencies: Run the following command in the terminal:

* pip install tensorflow keras nltk

1. Prepare project files: Place the provided files (‘intents.json’, ‘chatbot\_model.h5’, ‘words.pkl’, ‘classes.pkl’) in the project directory.

**Project Directory Structure**

AI\_project/

├── chatbot.py

├── chatbot\_model.h5

├── classes.pkl

├── intents.json

├── intents.py

├── training.py

├── words.pkl

├── cute\_background.png

**5. Implementation and Functionality**

**Overview of Code Files (Analysis with snippets and explanation)**

1. **intents.json**

Contains labeled data representing FAQs and their corresponding intents.

An intents.json file for a chatbot contains patterns of questions a user might ask, along with the possible responses and a tag for each question type. The file is used to create a neural network that classifies sentences as one of the tags in the file, and then displays a response from that group to the user. Example Structure:

{

"intents": [

{

"tag": "admissions",

"patterns": ["Tell me about admissions", "How do I apply?"],

"responses": ["You can apply online at our university website."]

}

]

}

1. **training.py**
   * Prepares and trains the chatbot’s machine learning model.

* **Data Preparation:**
  + Load and preprocess the intents data.
  + Tokenize and lemmatize words.
  + Create a bag-of-words representation and one-hot encoding for output labels.
* **Model Training:**
  + Define a neural network architecture using TensorFlow and Keras.
  + Compile the model with appropriate loss functions and optimizers.
  + Train the model using the prepared data.
* **Model Saving:**
  + Save the trained model and preprocessed data (words and classes) for later use during inference.
* **Imports**

**A screen shot of a computer

Description automatically generated**

* **os:** Handles operating system interactions, such as setting environment variables.
* **warnings:** Suppresses Python warnings for cleaner outputs.
* **random:** Randomizes data shuffling during training.
* **json:** Parses and processes JSON files (e.g., intents.json).
* **pickle:** Serializes and saves Python objects (e.g., words and classes lists).
* **numpy:** Provides support for numerical operations and array handling.
* **tensorflow:** Used to build, train, and save the neural network model.
* **nltk:** A library for natural language processing, used for tokenization and lemmatization.
* **WordNetLemmatizer:** Reduces words to their base forms to standardize vocabulary.
* **Suppressing Warnings and Setting Up NLTK**

**A computer screen with text

Description automatically generated**

* **Suppressing Warnings:** Ensures a clean console output by hiding unnecessary warnings.
* **TF\_CPP\_MIN\_LOG\_LEVEL:** Reduces TensorFlow log verbosity.
* **NLTK Setup:** Specifies the directory for NLTK data and downloads required datasets (e.g., Punkt tokenizer and WordNet lemmatizer).
* **Loading the Intents File**

**A black background with text

Description automatically generated**

* **intents.json:** A JSON file that defines the chatbot’s intents. Each intent contains:
  + **tag:** A unique identifier for the intent.
  + **patterns:** Example phrases/questions triggering the intent.
  + **responses:** Predefined answers for the intent.
* **Tokenizing and Preprocessing Patterns**

A computer screen shot of text

Description automatically generated

* **words:** Stores all unique words from patterns after tokenization.
* **classes:** Holds all unique intent tags.
* **documents:** Links tokenized patterns with their corresponding tags.
* **ignoreLetters:** Filters out unnecessary characters like punctuation.
* **Logic:**
  + Tokenize each pattern into words.
  + Store (pattern, tag) pairs in documents.
  + Add unique tags to classes.
* **Lemmatizing and Sorting Words**

**A screen shot of a computer code

Description automatically generated**

* **Lemmatization:** Converts words to their base forms (e.g., "running" to "run").
* **Sorting:** Ensures a consistent order for words and classes.
* **Saving Preprocessed Data**

**A black background with white text

Description automatically generated**

* **words.pkl:** Saves the processed words list for future use.
* **classes.pkl:** Saves the intent classes list for later reference.
* **Creating Training Data**

A screen shot of a computer program

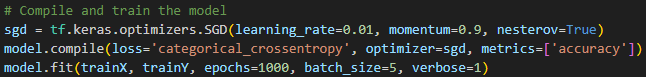
Description automatically generated

* **Bag of Words:** Encodes patterns as binary vectors indicating the presence of each word.
* **Output Labels:** Encodes intent tags as one-hot vectors.
* **Training Data:** Combines input features (bag of words) and output labels.
* **Neural Network Model**

A screen shot of a computer code

Description automatically generated

* **Dense Layers:** Fully connected layers for learning relationships.
* **Dropout:** Prevents overfitting by randomly deactivating neurons.
* **BatchNormalization:** Normalizes layer activations for faster convergence.
* **Softmax:** Outputs probabilities for each class.
* **Compiling and Training the Model**



* **SGD:** Stochastic Gradient Descent optimizer with momentum.
* **Categorical Crossentropy:** Loss function for multi-class classification.
* **Early Stopping:** Halts training if no improvement in loss for 10 epochs.
* **Saving the Trained Model**



* **chatbot\_model.h5:** Stores the trained model for future inference.
* **Purpose:** Allows the chatbot to use the trained model without re-training.
  + - **Output:** Saves trained model (‘chatbot\_model.h5’) and preprocessing files (‘words.pkl’, ‘classes.pkl’).

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Ran for 1000 steps, we don’t care if it overfits, memory isn’t an issue.

1. **chatbot.py**

The main script responsible for user interaction and response generation. It integrates the trained model with a Tkinter-based GUI, enabling users to interact with the chatbot. The analysis covers each line, function, and logic block in detail, ensuring a comprehensive understanding of its purpose and functionality. Additionally, an overview of the intents.json file is provided.

The key Features in it are:

1. **Natural Language Processing (NLP)**:
   * Uses NLTK for tokenization and lemmatization.
   * Loads and preprocesses intents (user queries) and trained model data.
   * Predicts the intent of user messages using a neural network model (chatbot\_model.h5).
2. **Intent-based Response Generation**:
   * Matches the user input with predefined intents and provides a relevant response.
   * Includes error handling for unrecognized input.
3. **Arithmetic Evaluation**:
   * Detects simple arithmetic expressions (e.g., addition, subtraction) and returns results.
4. **Graphical User Interface (GUI)**:
   * Built with Tkinter, featuring a chat window for interaction.
   * Displays chat history, a text input box, and a button to send messages.
   * Includes buttons for saving conversations and toggling between normal and a "pink" theme.
5. **Interactive Tips**:
   * Displays random examples from predefined patterns to guide users on what they can ask.
6. **Theme Customization**:
   * Allows users to toggle between a normal theme and a "pink/UwU" theme, changing the look of the interface.
7. **Conversation Saving**:
   * Allows users to save the conversation to a text file.
8. **Multithreading**:
   * Updates tips asynchronously in the background using a separate thread.

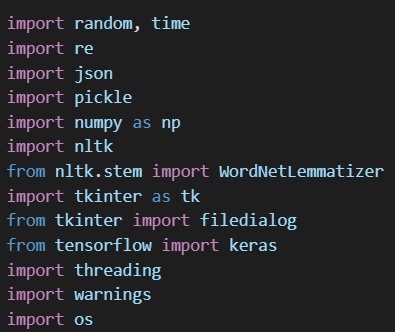
**Logic and Algorithm**

The program follows these steps:

* **Setup:**
  + Import libraries and modules.
  + Configure TensorFlow and NLTK.
* **Data and Model Loading:**
  + Load intents, preprocessed words and classes, and the trained model.
* **Natural Language Processing (NLP):**
  + Preprocess user input using tokenization and lemmatization.
  + Create a bag-of-words representation for input text.
  + Predict intent using the trained model.
  + Generate a response based on the predicted intent.
* **Additional Features:**
  + Handle arithmetic calculations.
  + Provide tips for user queries.
* **GUI Integration:**
  + Build a Tkinter-based graphical interface.
  + Allow user input, output display, and theme toggling.
  + Save conversation logs.

**3. Code Analysis with Snippets and Explanations**

* **Imports and Setup**



* **random, time:** Provide random choices for responses and manage asynchronous operations.
* **re:** Regular expressions for arithmetic pattern matching.
* **json:** Parse the intents.json file.
* **pickle:** Load preprocessed words and classes.
* **numpy:** Handle numerical computations.
* **nltk:** Tokenize and lemmatize user input.
* **tkinter:** Build the GUI.
* **keras:** Load the trained model.
* **threading:** Run asynchronous tasks (e.g., updating tips).
* **warnings, os:** Suppress unnecessary warnings and TensorFlow logs.
* **Loading Data and Model**

A black background with white text

Description automatically generated

* **intents.json:** Defines chatbot intents, including tags, patterns, and responses.
* **words.pkl, classes.pkl:** Contain preprocessed vocabulary and intent labels.
* **chatbot\_model.h5:** The trained neural network model for intent classification.
* **NLP Helper Functions**

**Function 1: clean\_up\_sentence**

A black background with white text

Description automatically generated

* **Purpose:** Tokenize and lemmatize user input for standardization.
* **Parameters:**
  + sentence (str): User input text.
* **Returns:**
  + List of lemmatized tokens.

**Function 2: bag\_of\_words**

A computer screen shot of a code

Description automatically generated

* **Purpose:** Create a binary vector representing the presence of words in the input.
* **Returns:** NumPy array (bag-of-words representation).

**Function 3: predict\_class**

A screen shot of a computer code

Description automatically generated

* **Purpose:** Predict the intent of user input based on the trained model.
* **Returns:** List of intents with probabilities.

**Function 4: get\_response**

A computer screen shot of a code

Description automatically generated

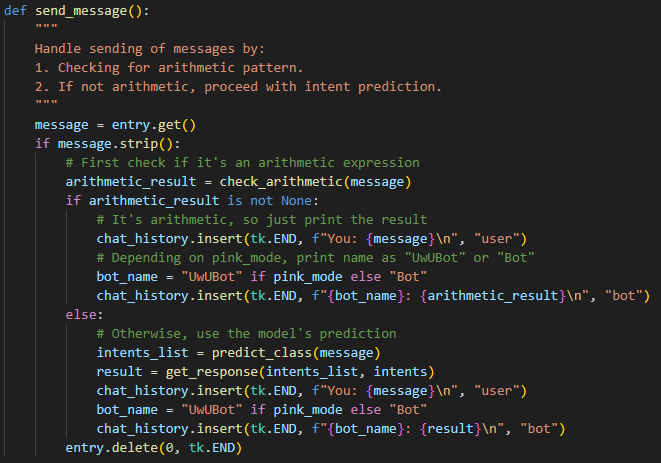
* **Purpose:** Generate a response based on the predicted intent.
* **Returns:** A random response string from the matched intent.
* **Arithmetic Check**

A screenshot of a computer program

Description automatically generated

* **Purpose:** Identify and evaluate arithmetic expressions in user input.
* **Returns:** Arithmetic result as a string or None if not an arithmetic expression.
* **GUI Functions**

**send\_message:** Handles user input and generates responses (via arithmetic check or intent prediction).



* **Get User Input**: Retrieves the message typed by the user.
* **Check for Arithmetic**: If the message is an arithmetic expression (like "5 + 3"), it calculates the result and shows it.
* **Intent Prediction**: If not arithmetic, it predicts the intent using the model and shows the relevant response.
* **Display in Chat**: Displays both the user's message and the bot's response in the chat history.
* **Clear Input**: Clears the input box after sending the message.

**save\_conversation:** Saves the chat history to a file.

A computer screen shot of text

Description automatically generated

**toggle\_theme:** Toggles between normal and pink themes, updating GUI elements accordingly.

* + - Pointlessly long code to paste here, just see in code

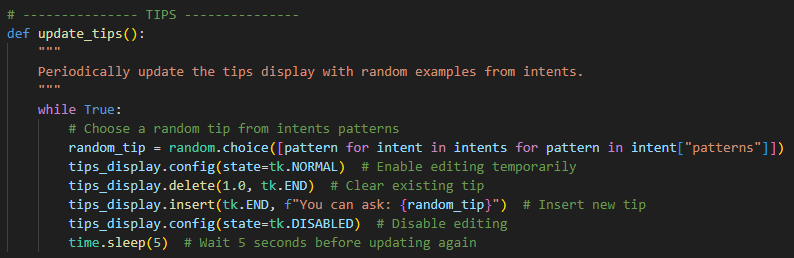
**Variables:**

* **pink\_mode**: Tracks if the pink theme is on (True) or off (False).
* **bg\_label**: Stores the background image label for the pink theme.
* **normal\_title**: Title of the window when using the normal theme.
* **pink\_bg**: Stores the pink background image.

**toggle\_theme() Function:**

* **Switch to Pink Theme**: Changes the window title to "UwUBot", sets a pink background, and changes colors to pink for chat, buttons, and input box. Sets **pink\_mode** to True.
* **Revert to Normal Theme**: Reverts all colors to normal and removes the pink background. Sets **pink\_mode** to False.

**update\_tips:** Displays random tips periodically from the intents file.



* **Building the GUI**

The chatbot GUI is built using Tkinter and includes:

* **Chat Frame:** Displays user and bot messages with scrollable history.

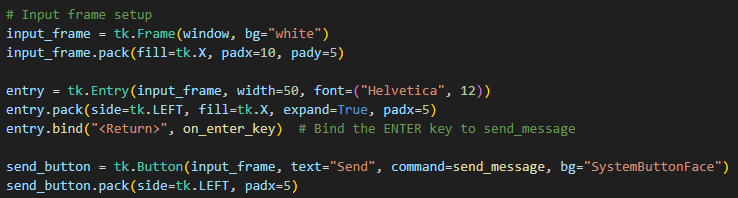
The **chat frame** serves as the main area where user inputs and bot responses are displayed. It includes a scrollable text widget for seamless navigation through the conversation history.

A computer screen shot of text

Description automatically generated

* **tk.Frame(window, bg="white"):** Creates a frame to group related widgets (conversation display and scrollbar).
* **tk.Text(chat\_frame, wrap=tk.WORD):** A multi-line text widget for displaying chat history.
  + **Attributes:**
    - wrap=tk.WORD: Ensures text wraps at word boundaries for better readability.
    - font=("Helvetica", 12): Sets a clean and legible font.
* **Scrollbar:** Linked to the chat\_history widget using the command property for vertical scrolling.
* **Tag Configuration:** Differentiates user and bot text visually by setting distinct colors and styles.
* **Input Frame:** Contains an entry widget for user input and a send button.

The **input frame** allows users to type messages and submit them for processing by the chatbot.



* **Entry Widget:** Captures user input.\n
  + **tk.Entry(...) Attributes:**
    - width=50: Specifies the entry box width in characters.
    - font=("Helvetica", 12): Matches the chat frame style for consistency.
  + **Event Binding:** Links the <Return> (Enter) key to the on\_enter\_key function, allowing users to send messages without clicking the send button.
* **Send Button:**
  + Triggers the send\_message function when clicked.
  + **Styling:** Default system style (SystemButtonFace).
* **Button Frame:** Provides options to toggle the theme and save conversations.

This frame contains buttons for switching between themes (normal and pink) and saving the chat history.

A computer screen shot of text

Description automatically generated

* **Toggle Theme Button:**
  + Toggles between the default and pink (UwU) themes using the toggle\_theme function.
* **Save Conversation Button:**
  + Opens a file dialog to save the conversation history using the save\_conversation function.
  + **Purpose:** Allows users to store chat logs for future reference.
* **Tips Display:** Shows helpful example queries.

The **tips display** periodically provides users with example queries to guide them in using the chatbot.

A computer code on a black background

Description automatically generated

* **Tips Frame:** Groups the tip display area.
* **Tips Display Widget:**
  + **Attributes:**
    - wrap=tk.WORD: Ensures tips are displayed neatly in a single line.
    - state=tk.DISABLED: Prevents users from editing the displayed tips.
    - bg="light yellow": Highlights tips for better visibility.
* **Tip Updates:**
  + A separate thread (update\_tips) periodically updates the display with random examples from the intents.json file.
* **Main Window and Event Loop**

A screen shot of a computer code

Description automatically generatedThe main window and event loop manage the GUI's structure and ensure it responds to user actions.

* **Main Window (tk.Tk()):** The root widget for the GUI.
  + **Attributes:** title, geometry, minsize
* **Event Loop (mainloop()):** Continuously listens for user interactions and updates the GUI accordingly.

1. **intents.py**
   * Handles data loading and utility functions for ‘intents.json’.
2. **chatbot\_model.h5, words.pkl, classes.pkl**
   * **chatbot\_model.h5:** Stores the trained neural network weights.
   * **words.pkl:** Contains preprocessed words for input tokenization.
   * **classes.pkl:** Maps intents to their labels.

**6. Conclusion**

The development of the AI-powered University Admissions Chatbot provided a comprehensive learning experience in building an intelligent system. Through this project, we were able to address real-world challenges in the admissions process, reducing administrative workload and enhancing the user experience. The chatbot, equipped with natural language processing and machine learning capabilities, proved to be a valuable tool for automating FAQs, demonstrating a significant improvement in efficiency and user satisfaction.

**What We Learned**

This project deepened our understanding of various technical and teamwork aspects:

1. **Technical Skills**: Implementing Python libraries like TensorFlow and Keras enhanced our knowledge of machine learning and neural networks. Using Tkinter taught us how to build user-friendly graphical interfaces.
2. **Problem-Solving**: Developing the intents and responses required careful planning to ensure accuracy and relevance, teaching us how to tackle real-world user needs.
3. **Collaboration**: The division of responsibilities among team members showed us the importance of effective communication and teamwork in achieving a common goal.

**Resources/References:**

1. ResearchGate: *Development of an AI Chatbot to Support Admissions and Career Guidance*. <http://dx.doi.org/10.22662/IJEMR.2020.4.2.013>
2. *How universities are using chatbots to improve the student admissions process | Full Fabric*. (n.d.). <https://www.fullfabric.com/articles/how-universities-are-using-chatbots-to-improve-the-student-admissions-process>
3. Rahman, A. (2024, March 22). A guide to building a custom GPT chatbot for university admissions. *Medium*. <https://medium.com/@ahmadrahman/a-guide-to-building-a-custom-gpt-chatbot-for-university-admissions-227b605e9e8c>