TRIBHUVAN UNIVERSITY

FACULTY OF COMPUTER SCIENCE

AND

INFORMATION TECHNOLOGY

A

Report On

NCCS Chatbot

Submitted to:

Computer Science and Information Technology

National College of Computer Science and Information Technology

Paknajol, Kathmandu

Submitted by:

Kriti Ghimire

College Reg: NCCS CSIT 461

ACKNOWLEDGEMENT

For the partial fulfillment of this semester's project, I would like to express my sincere gratitude to everyone who has directly or indirectly supported me in developing this project. There were moments when the project seemed challenging, or even impossible, but I am deeply thankful to my respected supervisor, **Mr. Chhetra Bahadur Chhetri**, for his continuous guidance and encouragement, which enabled me to complete the project on time. I am also profoundly grateful to my teachers and friends who assisted me throughout this journey, helping me identify and correct my mistakes along the way.

Also, I extend my heartfelt thanks to Tribhuvan University for providing me with this invaluable opportunity through the Computer Science and IT program. This experience has not only helped me understand project ethics at an early stage but has also enabled me to evaluate and expand my knowledge further.

Thankyou,

Kriti Ghimire

BSc.CSIT 7th Semester

ABSTRACT

The NCCS Chatbot project aims to revolutionize how students access critical information at NCCS College by providing a virtual assistant powered by artificial intelligence. The chatbot is designed to streamline communication by offering quick, accurate responses to student inquiries related to admissions, fees, courses, and college facilities. Through the use of advanced Natural Language Processing (NLP) techniques and machine learning models, particularly Long Short-Term Memory (LSTM) networks, the chatbot can understand and respond to user queries more effectively. The system reduces reliance on in-person visits or lengthy phone calls, thereby improving efficiency and accessibility. The project follows the Agile development methodology, ensuring iterative progress and continuous improvements based on user feedback. The primary aim of this system is to enhance the student experience by providing real-time, automated support for essential college information. The system is limited by its focus on backend data processing and does not emphasize the user interface design. This project presents an innovative solution to information dissemination within educational institutions, with potential for future enhancements, such as voice command integration and multilingual support, to further improve user engagement and satisfaction.

Keywords: *chatbot, inquiries, LSTM, NLP, response, user experience etc.*

Table of Contents

Table of Figure

Chapter 1: Introduction

1.1 Introduction

The NCCS-Chatbot represents a sophisticated virtual assistant tailored specifically for NCCS College, designed to streamline information retrieval processes for prospective and current students alike. It offers a comprehensive range of features, including detailed insights into the college's location, fee structures, course offerings, campus environment, and parking facilities. By leveraging advanced technologies such as Machine Learning (ML) and Natural Language Processing (NLP), the chatbot provides an intuitive and conversational interface where users can interact naturally and receive prompt responses akin to speaking with college authorities. This AI-driven approach ensures that queries are addressed swiftly and accurately, enhancing user satisfaction and facilitating informed decision-making. Implemented using Python for backend development and incorporating HTML, CSS and JavaScript for frontend. The chatbot is robust and scalable. Future enhancements may include multilingual support, voice interaction capabilities, and further personalization to cater to the diverse needs of its users, solidifying its role as a pivotal tool in the NCCS College community.

1.2 Problem Statement

The problem statement for NCCS Chatbot is that the current methods of accessing college information, such as phone calls or emails, often prove inefficient and time-consuming for users. This limitation results in extended waiting periods, sometimes spanning hours or days, which can frustrate applicants and delay the enrollment process. Moreover, traditional communication channels like email and phone calls can feel impersonal and may not fully address users' specific needs, leading to repetitive queries and navigation through cumbersome phone menus. Particularly with complex academic inquiries concerning admission procedures, fee structures, scholarship opportunities, and document requirements, the process becomes further convoluted. Addressing these queries through conventional means requires substantial resources and can result in delays in providing accurate and comprehensive information to prospective students and their families.

1.3 Objectives

- To provide 24/7 access to college information like admissions, fees, courses and so on.
- To minimize the need for in-person visits or calls.

1.4 Scope and Limitation

1.4.1 Scope

The NCCS College chatbot aims to provide comprehensive and accessible information across a broad spectrum of topics crucial to students and stakeholders. It covers essential details such as campus location, fees, course offerings, college facilities, and parking, ensuring users have easy access to vital information. Through its AI-driven capabilities, the chatbot offers a conversational text interface that simulates natural human interaction, delivering prompt responses to user queries. This functionality streamlines communication and enhances user satisfaction by providing efficient access to accurate information regarding admissions, fee structures, scholarships, and document requirements. Multi-language support further extends accessibility, accommodating users who communicate in languages other than English. By integrating seamlessly with college systems like the student information system, the chatbot ensures data accuracy and eliminates redundant manual processes. Available 24/7, it offers continuous availability, serving as a reliable resource for users to obtain information anytime. Continuous improvements based on user feedback and data analysis ensure the chatbot evolves to meet changing user needs, maintaining a high standard of user experience and effectiveness in supporting the NCCS College community.

1.4.2 Limitations

- The chatbot may not respond to all user queries, especially if they fall outside its trained knowledge base or if it encounters ambiguity. Certain complex or highly specific queries may require further development or human assistance.

Chapter 2: Background Study and Literature Review

2.1 Background Study

A Chatbot is a software application that uses artificial intelligence (AI) and natural language processing (NLP) to simulate human conversation with users. Chabot can be designed to interact with users through a messaging interface, voice-enabled devices, or chat windows on websites and mobile apps.

The first Chatbot, ELIZA [1], was created in the 1960s by Joseph Weizenbaum at MIT. ELIZA was designed to simulate a therapist and could engage in simple conversational exchanges with users. It uses pattern matching and substitution methodology to simulate conversation. Designed to convincingly simulate the way a human would behave as a conversational partner. However, it was limited in its ability to understand the context of a conversation and provide meaningful responses.

A Chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. However, from a technological point of view, a Chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP). Formulating responses to questions in natural language is one of the most typical Examples of Natural Language Processing applied in various enterprises' end-use applications.

With the advancements in AI and NLP technologies, Chatbot have become more sophisticated and can now handle complex conversations, make recommendations, answer queries, and even perform tasks for users. Today, chatbots are used in various industries, such as customer service, healthcare, finance, education, and entertainment, to enhance user experience, streamline processes, and reduce costs.

2.2 Literature Review

A literature review for a college chatbot project would typically explore existing research, studies, and implementations related to chatbots in educational settings.

WorldLink's chatbot is a virtual assistant driven by artificial intelligence that can assist consumers with a variety of chores and inquiries. It can help you handle internet or NETTV issues, recover forgotten passwords, manage accounts, and more. The chatbot is available 24/7 via the WorldLink website and WhatsApp, allowing clients to obtain assistance at any time. It

gives prompt and individualized responses while also providing efficient and consistent service. The chatbot's straightforward and user-friendly design ensures that clients can quickly and easily solve their concerns [2].

A Literature Survey of Recent Advances on Chatbots provides an in-depth examination of the development and present condition of chatbot technologies, highlighting the incorporation of Artificial Intelligence (AI) and Natural Language Processing (NLP). It outlines essential developmental stages in chatbot systems, from initial rule-based designs to sophisticated deep learning techniques that have greatly enhanced response creation and context management.

A key emphasis of the paper is on advancements in algorithms, especially the implementation of models such as Sequence-to-Sequence (Seq2Seq) and Transformer architectures (for instance, BERT, GPT). These models have allowed chatbots to produce more coherent, contextually appropriate, and human-like replies, surpassing the constraints of conventional pattern-matching methods. [3]

The "Role of AI Chatbots in Education: Systematic Literature Review" offers a thorough exa mination of 67 papers to evaluate the use of AI chatbots in classrooms.
It emphasizes the advantages for students, including skill improvement, instant support, and i ndividualized learning experiences.
Teachers gain from improved instructional techniques and time-saving features.
On the other hand, the study also discusses issues with fair assessment procedures, data prote ction and responsible AI use, and the accuracy and dependability of chatbot responses.
To optimize the use of AI chatbots in education, the authors stress the necessity of cautious d eployment and continual assessment.

While acknowledging the role of advanced AI technologies, like natural language processing (NLP) and machine learning (ML), in enabling chatbots to simulate human-
like interactions and provide contextually appropriate responses, the paper does not go into gr eat detail about specific algorithms or technical methodologies used in chatbot development, despite its primary focus on the applications and implications of AI chatbots in education. Thi s suggests that future research should investigate the technical aspects of chatbot design and i mplementation to further enhance their effectiveness in educational contexts. [4]

Chapter 3: System Analysis and Design

3.1 System Analysis

The project will be explained using dataflow diagrams, flowcharts, use-case diagrams, connection and entity diagrams, and so on.

3.1.1Requirement Analysis

The NCCS Chatbot requirement analysis entails identifying and defining the features, functionalities, and requirements necessary for the website to run effectively and efficiently.

3.1.1.1Functional Requirement

- The user shall ask the queries.
- The user shall be able to view the responses.
- The user shall be able to listen the response.

3.1.1.2 Non-functional Requirement

- The system must have simple and user-friendly UI.
- The system is available 24/7.
- The system must have a good response.
- The system must be maintainable.

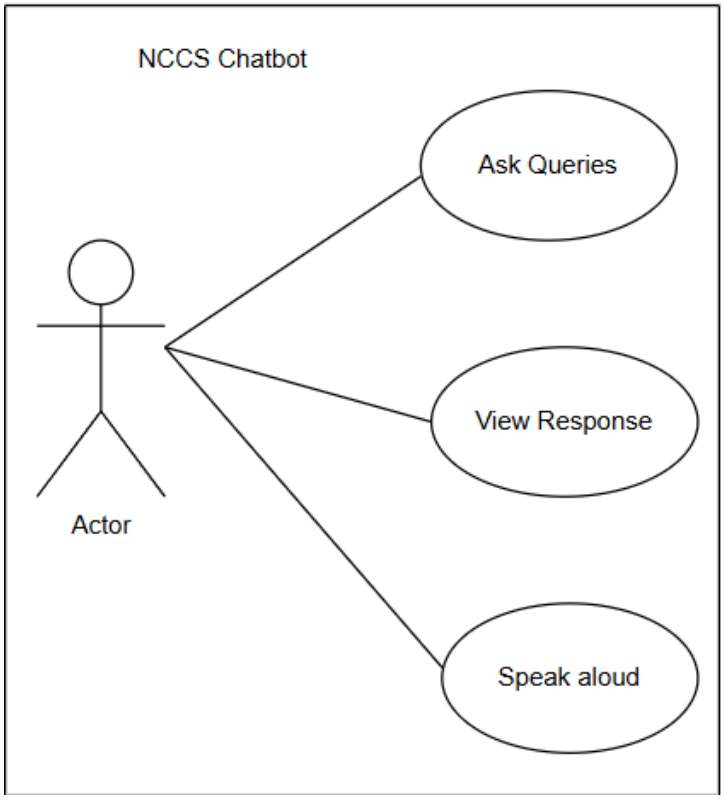

Figure 1:Use case diagram of Chatbot

3.1.2 Feasibility Analysis

A feasibility study for the NCCS College chat-bot involves evaluating whether developing and deploying the chatbot is practical and beneficial.

3.1.2.1 Technical Feasibility

Our project was developed within Intel Core i5 2.4 GHz, 4GB RAM and windows 11 OS. So that it can support every user's device.

3.1.2.2 Operational Feasibility

The chatbot should be easy for students, staff, and stakeholders to use. It must provide clear, accurate answers to common questions, such as admissions processes, fees, course details, and more. The system must be scalable, capable of handling a large number of simultaneous users, especially during peak times like enrollment periods.

3.1.2.3 Economic Feasibility:

The project can be developed in a very cost-effective way because the project would be using open-source software like python which is available free online. Also, the benefits of the project outweigh the cost. Hence the project can be deemed economically feasible.

3.1.2.4 Schedule Feasibility:

The Gantt chart of the project is as follow:

Activities/Duration	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Planning										
Research										
Data Gathering										
Requirement Analysis										
Coding										
Testing										
Validation										
Documentation										

Figure 2: Gantt Chart

3.1.3 Object Modeling

3.1.3.1 Object modeling using class and object diagram

The figure given below shows the class diagram of the NCCS College Chatbot project shows how different components of the chatbot system work together. There are four main parts: Model, ChatBot, Admin Panel, and Users. The Model stores the chatbot's knowledge, including the different questions (intents) and their corresponding answers (responses). The ChatBot itself interacts with users by receiving their queries and generating appropriate responses based on the model. The Admin Panel allows administrators to manage the chatbot's knowledge by adding or editing intents and responses. Finally, Users are the people who interact with the chatbot, asking questions and receiving answers. The diagram shows that each chatbot uses a single model, can handle multiple users at once, and is managed by the admin through the panel. This structure helps in keeping the system organized and easy to update or expand.

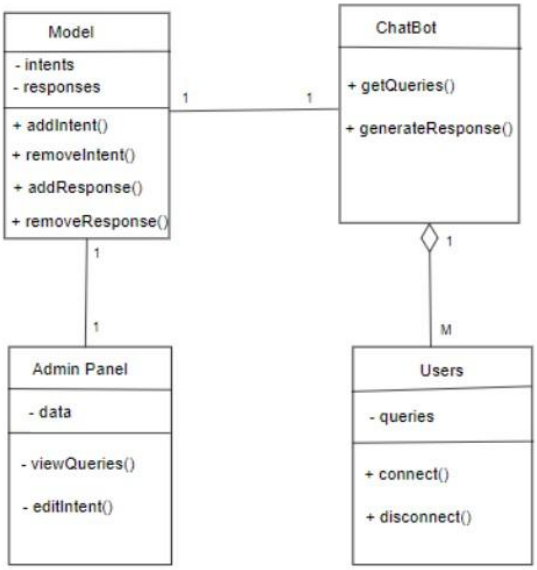

Figure 3:Class Diagram of NCCS Chatbot

3.1.4 Dynamic modeling

3.1.4.1 Sequence Diagram

In the figure the sequence diagram starts when the user sends the query and the chatbot respond the query, while responding the query it performs different mechanism which are cleaning, tokenization, lemmatization and create the sequence of bag of words and with the help of trained dataset by LSTM algorithm, those which has highest probability of the input it will generate the respective output. And if the user query untrained data, then the query is stored in the database.

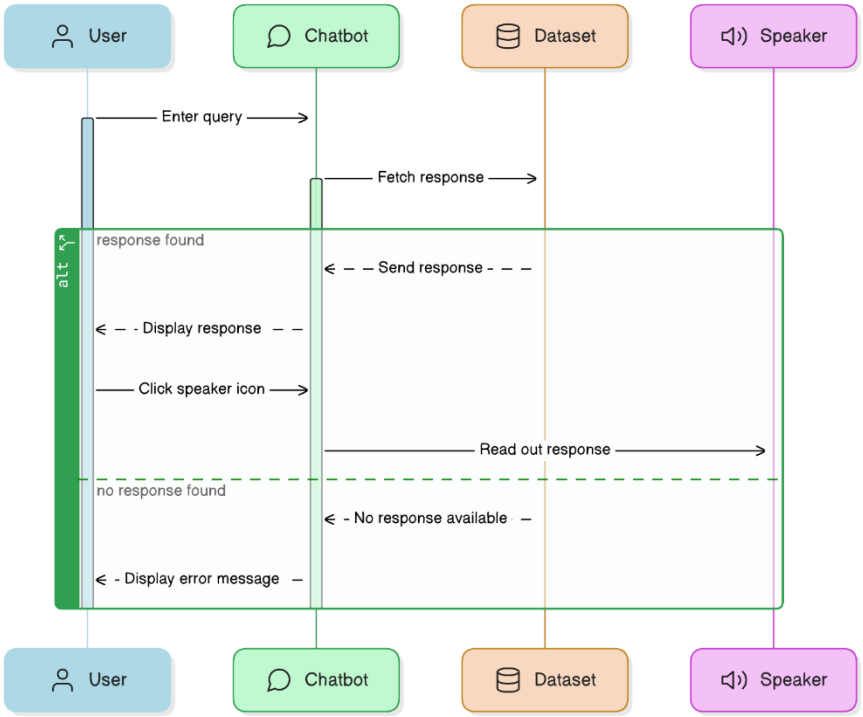

Figure 4: Sequence Diagram of NCCS Chatbot

3.1.4.2 State Diagram

The state diagram of the NCCS Chatbot illustrates the workflow for handling user interactions in a streamlined process. It begins with the User Query, where the user inputs a question or request into the system. The chatbot transitions to the Process Input state, where the query is analyzed and understood to determine the appropriate response. Once the input is processed, the system moves to the Generate Output state, where it formulates a relevant reply based on the user's query. Finally, in the Reply Response state, the chatbot delivers the generated response back to the user, completing the interaction. The arrows in the diagram depict the sequential flow from receiving the user's input to providing the response, ensuring a clear and structured exchange.

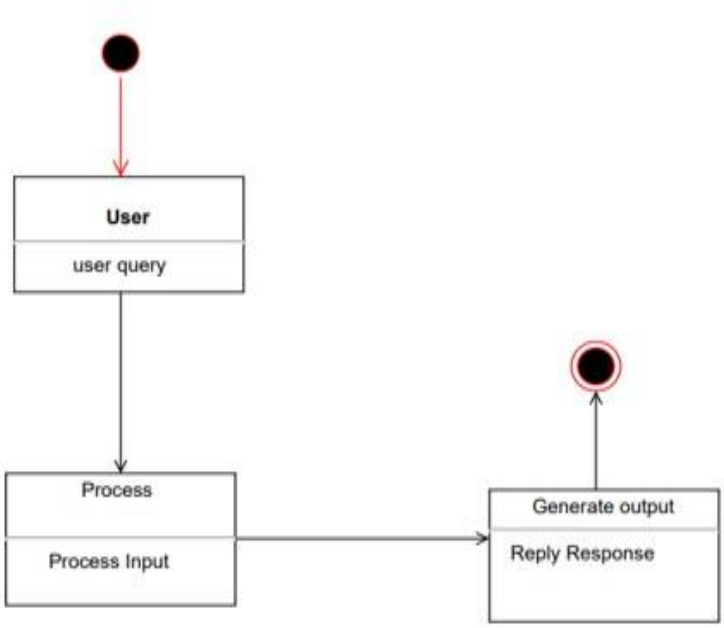

Figure 5: State Diagram of NCCS Chatbot

3.1.5 Process modeling using Activity Diagrams

The activity diagram outlines the workflow of the NCCS chatbot system. When a user submits a query through the homepage, the system first checks if the required dataset is loaded. If the dataset is unavailable, it sends an error message to the user indicating the issue. If the dataset is loaded, the system processes the query by tokenizing and encoding it, then pads the sequence to make it compatible with the LSTM model. The model predicts a tag based on the query. If a valid tag is found, a matching response is fetched from the dataset. If no tag is identified, the system sends a default message indicating it cannot provide an answer. For valid responses, the text is converted into speech using gTTS, saved as an audio file, and both the text and audio are sent back to the user.

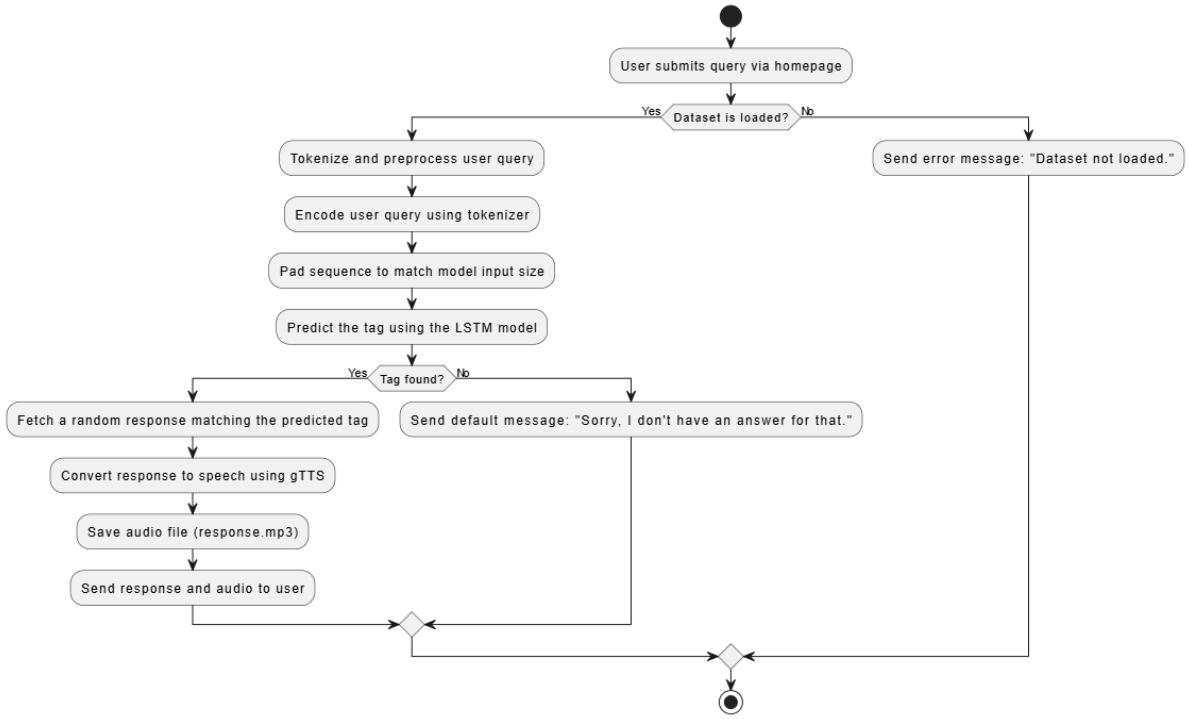

Figure 6: Activity Diagram of NCCS Chatbot

Chapter 4: Implementation

4.1 Implementation

The NCCS Chatbot is designed to streamline the process of accessing information at NCCS College, making it more efficient and user-friendly. It features a comprehensive architecture that includes a frontend, backend, and database components.

The frontend is built using HTML, CSS, and JavaScript, which together create a visually appealing and interactive user interface. This interface allows users to type in queries and receive responses from the chatbot in a conversational format. JavaScript enhances this interactivity by enabling real-time communication with the backend without needing to refresh the page. The backend is developed in Python and serves as the brain of the chatbot. It uses advanced Natural Language Processing (NLP) techniques to understand user queries. Specifically, the chatbot employs Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) that excels at handling sequential data. This allows the chatbot to interpret user input accurately and generate appropriate responses. Python libraries such as NLTK might be used for text processing, while TensorFlow or Keras are used to build and train the LSTM model. Where datasets are used to store essential data such as user interactions, training data for the chatbot, and responses.

Overall, the NCCS Chatbot represents a significant step towards improving information accessibility and efficiency for NCCS College's community, leveraging cutting-edge technologies to offer a more responsive and user-friendly experience.

4.1.1 Tools Used

- PYTHON

  Python is a popular, high-level programming language that is widely utilized in diverse applications like scientific computing, web development, data analysis, artificial intelligence, and more. It supports a wide range of object-oriented programming (OOP) concepts, including encapsulation, inheritance, and polymorphism. Python also boasts a rich collection of third-party frameworks and standard libraries for various applications, making it a popular choice for developers and organizations.

- HTML

  HTML, which stands for Hypertext Markup Language, is a markup language that is primarily used to create web pages and web applications.HTML is used to create the simple college website in this project.

- CSS

  Cascading Style Sheets (CSS) is a style sheet language that is utilized to describe the presentation of a web document written in HTML or XML. It is used to style the website which is created with HTML in this project.

- Visual Studio Code

  This is our code editor where we have written our all of codes. This tool is very user friendly and have lots of extensions which helps for making the coding process more efficient.

- JavaScript

  JavaScript can be used in this project to enhance the chatbot's user interface and interactivity. It can control audio playback for the chatbot's TTS responses and handle user events like button clicks or key presses, providing a smoother and more responsive user experience.

- Snipping Tool

  This is a pre-installed tool on our Windows Machine which helps to take the screenshots of all required figures, documents and user interfaces.

- Microsoft Word

  This tool is used to do all the documentation of our project from the scratch to the very end.

- Microsoft PowerPoint

  This tool is used to make the PowerPoint slides to do presentation of our project.

4.1.2 Implementation Details of the Module

LSTM Model:

Old Model Explanation

```
model = Sequential ([
Embedding (1000, 64, input_length=max_length),
LSTM (64),
Dense (64, activation='relu'),
Dense(len(np.unique(encoded_tags)), activation='softmax')
])
```

- Embedding

  The Embedding layer in the model, defined as Embedding (1000, 64, input_length=max_length), is used to convert input words, which are represented as integers (tokens), into dense vectors of a fixed size.

  Here, 1000 specifies the size of the vocabulary, meaning the tokenizer is limited to using only the top 1,000 most frequently occurring words in the dataset. Any word that is not among these top 1,000 words is either ignored or replaced with an Out-of-Vocabulary (OOV) token if defined earlier.

  The second parameter, 64, defines the size of the embedding vector, meaning each word in the vocabulary is represented by a 64-dimensional vector. This vector captures the semantic meaning of the word in a dense format, enabling the model to understand relationships between words.

  Finally, input_length=max_length sets a fixed length for all input sequences, ensuring that each input sequence fed into the model has exactly max_length tokens. If a sequence is shorter than this length, it is padded (usually with zeros), and if it is longer, it is truncated, allowing for consistent input sizes and efficient batch processing. This embedding layer is crucial for transforming sparse, high-dimensional word representations into dense, continuous vectors that are easier for neural networks to process.

- LSTM (64)

  The LSTM layer with 64 units is used to capture patterns in sequential data by processing each element in the sequence one at a time. The parameter 64 specifies the number of output units, also known as hidden states, in the LSTM. This determines the dimensionality of the layer's output. Essentially, the LSTM takes sequential input data and processes it to produce a fixed-length representation of size 64, capturing meaningful temporal relationships and dependencies within the sequence.

- Dense (64, activation='relu')

  The Dense layer with 64 neurons is a fully connected layer that connects each input from the previous layer to all 64 neurons. Each of these neurons applies a weighted sum of inputs followed by an activation function to produce its output. The activation function used here is ReLU (Rectified Linear Unit), which outputs the input directly if it is positive; otherwise, it outputs zero. This introduces non-linearity into the model, enabling it to learn complex patterns and relationships in the data. The use of ReLU is popular because it helps mitigate issues like the vanishing gradient problem, allowing the model to train faster and perform better, especially in deeper networks.

- Dense(len(np.unique(encoded_tags)), activation='softmax')

  The Dense Layer with softmax activation, which is particularly well-suited for multi-class classification tasks. Softmax converts the raw output of the network into a probability distribution, where each neuron corresponds to a possible class (in this case, a unique tag). The number of neurons in the output layer matches the number of unique tags (encoded_tags), so each neuron represents a different tag. The softmax function ensures that the sum of all output probabilities is 1, with each probability indicating the likelihood of the input belonging to a specific tag. This allows the model to classify the input into one of the predefined tags based on the highest probability.

New Model Explanation (Improved Version)

```
model = Sequential ([
    Embedding (1000, 128, input_length=max_length),
    LSTM (128, return_sequences=True),
    Dropout (0.2),
    LSTM (64),
    Dense (64, activation='tanh'),
    BatchNormalization (),
    Dense(len(np.unique(encoded_tags)), activation='softmax')
])
```

- Embedding (1000, 128, input_length=max_length)

  The Embedding layer with 1000 as the vocabulary size and 128 as the embedding dimension is similar to the old model but uses a larger embedding size. The vocabulary size of 1000 means the model will consider the top 1,000 most frequent words in the dataset. By increasing the embedding dimension to 128, the model can represent each word as a 128-dimensional vector, allowing it to capture richer, more detailed semantic relationships between words. This enhanced representation helps the model better understand and differentiate between subtle meanings and contexts, leading to improved performance, especially for complex text inputs.

- LSTM (128, return_sequences=True)

  The LSTM (128, return_sequences=True) layer consists of 128 units, which enables the model to capture complex patterns in sequential data. The parameter return_sequences=True ensures that the LSTM layer outputs a sequence of hidden states for each time step in the input, rather than just the final hidden state. This is crucial when stacking multiple LSTM layers, as it allows the next LSTM layer to receive and process the entire sequence of outputs from the previous layer, rather than only the final output. This setup helps the model learn more detailed temporal relationships within the data.

- Dropout (0.2)

  The Dropout (0.2) layer randomly sets 20% of the input units to zero during training. This technique helps prevent overfitting by reducing the model's dependence on specific neurons, forcing it to learn more robust and generalized features. By randomly deactivating

parts of the network, dropout encourages the model to not rely on any single neuron, which improves its ability to generalize to unseen data and reduces the risk of overfitting to the training set. This results in a more reliable and adaptable model.

- LSTM (64)

  The LSTM (64) layer is a second LSTM layer with 64 units, which processes the sequential data output from the previous LSTM layer. By stacking multiple LSTM layers, the model can learn more complex patterns and relationships in the data. The first LSTM layer captures initial features of the sequence, and the second LSTM layer refines this understanding, allowing the model to recognize higher-level patterns and dependencies. This layered approach enhances the model's ability to capture intricate temporal relationships in the data, improving its performance for tasks like sequence prediction or classification.

- Dense (64, activation='tanh')

  The Dense (64, activation='tanh') layer is a fully connected layer with 64 neurons that uses the tanh (hyperbolic tangent) activation function. Unlike ReLU, which outputs values between 0 and positive infinity, the tanh function produces values in the range of -1 to 1. This allows the model to handle both positive and negative values, making it useful for learning sequences where the data may have both positive and negative trends. The tanh function helps in normalizing the output, which can improve the learning process, especially for tasks that require capturing complex relationships in the data.

- BatchNormalization ()

  The BatchNormalization() layer normalizes the output of the previous layer, ensuring that the activations have a mean of 0 and a standard deviation of 1. This normalization helps reduce internal covariate shift, which can occur when the distribution of layer inputs changes during training. By stabilizing the training process, batch normalization speeds up convergence and can lead to better performance. It also helps in improving the model's generalization ability, preventing overfitting, and making the learning process more efficient.

- Dense (len(np.unique(encoded_tags)), activation='softmax')

  The final output layer uses a softmax activation function to produce a probability distribution over the classes. It outputs the probabilities for each class, making it suitable for multi-class classification tasks.

Key Differences Between the Old and New Model:

- Embedding Layer: The new model uses a larger embedding size (128 vs. 64), which is used to capture more detailed word relationships.
- Stacked LSTM Layers: The new model has two LSTM layers (128 and 64 units) instead of one, enabling it to learn more complex patterns in the data.
- Dropout Layer: The new model includes a Dropout layer for regularization, which helps reduce overfitting.
- Batch Normalization: This is added in the new model to improve training speed and model stability.
- Activation Functions: The new model uses tanh in one of its Dense layers, providing a more balanced output range compared to relu.

Chapter 5 : Conclusion

In conclusion, the NCCS Chatbot Project successfully integrates cutting-edge AI techniques with user-friendly frontend components, creating an interactive and informative chatbot tailored specifically for NCCS College. The implementation of dual AI models, LSTM, enhances the accuracy and reliability of responses, providing a versatile interaction experience. With the completion of backend integration, thorough testing, and deployment, the chatbot promises significant improvements in accessibility and information dissemination for students and staff at NCCS College. Future developments could further expand functionality, enabling real-time analytics and advanced personalized interactions, reinforcing its role as an essential tool in academic and administrative communication.

Chapter 7: References

[1] "ELIZA," 13 september 2024. [Online]. Available: https://en.wikipedia.org/wiki/ELIZA.

[2] "WorldLink ChatBot: Transforming the best service with AI," Wordlink, 4 october 2023. [Online]. Available: https://worldlink.com.np/worldlink-chatbot/.

[3] S. J. ,. M. Guendalina Caldarini, "A Literature Survey of Recent Advances in Chatbots," 15 January 2022. [Online]. Available: https://www.mdpi.com/2078-2489/13/1/41. [Accessed 20 May 2025].

[4] M. G. M. Lasha Labadze, "Role of AI chatbots in education: systematic literature review," 23 October 2023. [Online] Available: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-023-00426-1. [Accessed 20 May 2025].