International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 04 | Apr 2024 www.irjet.net p-ISSN: 2395-0072

CHATBOT APPLICATION USING NLTK AND KERAS

T.Ranjith Kumar[1], Anishka Akarapu[2] , Sathwik Rao Allam[3], Naga Chandana Gattepally[4], Pooja yashaswi Nellutla[5]

[1]Assistant professor, Computer Science and Engineering, Kakatiya Institute of Technology and Science, Warangal, India

[2,3,4,5] Computer Science and Engineering, Kakatiya Institute of Technology and Science, Warangal, India

---

***Abstract*— This paper presents an advanced chatbot application designed to provide comprehensive first aid guidance and support to users. With the increasing digitalization of healthcare, chatbots offer promising solutions for delivering timely and accessible information. Leveraging state-of-the-art technologies such as the Natural Language Toolkit (NLTK) for natural language processing (NLP) tasks and the Keras deep learning framework, the developed chatbot demonstrates a sophisticated understanding of user queries and generates contextually relevant responses. The development process involves several key phases. Initially, a dataset comprising first aid intents is collected and meticulously preprocessed using NLTK's tokenization, part-of-speech tagging, and syntactic parsing functionalities to extract meaningful information from user inputs. Subsequently, multiple neural network architectures including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM, and Dense Neural Network are implemented and trained on this preprocessed dataset. Performance evaluation metrics such as accuracy, precision, recall, and user feedback are employed to assess the efficacy of the developed chatbot. Comparative analyses with existing chatbot systems are conducted to highlight its strengths and identify potential areas for improvement. Notably, after thorough evaluation, the Dense Neural Network model is chosen for its superior performance in understanding user intent, maintaining context in conversations, and generating responses that mimic human-like interactions.This paper contributes to the ongoing evolution of chatbot technology by presenting a sophisticated solution tailored for first aid guidance. The findings underscore the potential of chatbots in providing valuable support in healthcare and emergency response domains, paving the way for further enhancements and applications in digital healthcare services.**

Keywords——Natural Language Tool kit (NLTK), Dense neural network, Keras, Long Short-Term memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM.

I. INTRODUCTION

Chatbots have become increasingly popular in the tech industry, providing users with personalized and interactive experiences. These computer programs simulate conversation with human users, offering a unique interface for communication [1]. Initially, chatbots were rule-based systems triggering predefined responses with specific keywords [2]. However, advancements in machine learning and neural networks have enabled chatbots to generate more nuanced responses, enhancing their conversational capabilities [3]. One of the primary applications of chatbots is in customer service, where they handle routine queries and provide instant support [4]. They have also found utility in diverse sectors like healthcare, education, and e-commerce, streamlining tasks and enhancing user experiences. In recent years, the integration of chatbots into various platforms has transformed the way businesses interact with their customers, offering round-the-clock support and personalized assistance. This shift towards automated communication reflects the growing reliance on technology to streamline processes and enhance user experiences.

Moreover, the adaptability of chatbots to different industries underscores their versatility and potential for innovation. As technology advances, chatbots are expected to understand context, emotions, and user intent more accurately, ushering in a new era of conversational interfaces. The increasing demand for efficient communication solutions has led to heightened interest in chatbot development. This project aims to contribute to this landscape by implementing a chatbot using the Natural Language Toolkit (NLTK) and Keras, two powerful libraries in natural language processing (NLP) and deep learning. NLTK, renowned for its comprehensive suite of tools, offers functionalities for tokenization, stemming, and syntactic analysis. Keras simplifies the development of deep neural networks, providing an intuitive interface for building and training complex models. The synergy between NLTK and Keras positions our project at the forefront of conversational AI research and implementation.

Chatbots, powered by NLP techniques, have gained popularity across various domains due to their human-like interaction capabilities. They offer a promising avenue for enhancing communication and user engagement, contributing to the evolution of human-computer interaction.

MOTIVATION

The motivation for this project stems from the increasing demand for efficient communication solutions across diverse sectors. In today's digital era, businesses, organizations, and individuals seek innovative tools to streamline customer interactions and enhance user experiences. Chatbots, driven by artificial intelligence, offer

a promising solution to address these needs. Despite the growing adoption of chatbots, there remains a need for advanced solutions capable of understanding user intent and delivering contextually relevant responses. This project aims to leverage cutting-edge technologies, particularly Natural Language Processing (NLP) and deep learning frameworks like NLTK and Keras, to develop a sophisticated chatbot application. By addressing practical challenges in chatbot development, such as managing conversational context and improving response coherence, this project seeks to contribute to the advancement of conversational AI. The ultimate goal is to create intelligent and adaptive conversational agents that empower users and drive technological innovation.

OBJECTIVES

- Create an operational chatbot proficient in understanding and generating responses akin to human interaction.
- Employ the Natural Language Toolkit (NLTK), a prominent Python framework tailored for handling human language data, to augment the chatbot's natural language processing skills.
- Incorporate Keras, an accessible deep learning framework, to enable the construction of a neural network model, thus enhancing the chatbot's comprehension of conversations and the generation of responses.

II. LITERATURE SURVEY

The literature survey reveals a dynamic landscape in chatbot development, marked by notable advancements and emerging challenges. Chatbots leverage sophisticated natural language processing (NLP) techniques, as highlighted by Jurafsky and Martin (2019), to comprehend user inputs and generate contextually relevant responses [5]. Deep learning methodologies, as discussed by Goodfellow et al. (2016), have further enhanced their capability to handle intricate conversational scenarios [6]. Frameworks like NLTK and Keras, as described by Bird et al. (2009), have emerged as prominent choices, offering comprehensive tools for NLP tasks and neural network development [7].

Despite these advancements, existing chatbot systems encounter limitations that impede their effectiveness and user satisfaction. One such limitation, as identified by Tang et al. (2019), is the inability to handle ambiguous queries adeptly [9]. Additionally, many chatbots exhibit limited adaptability to dynamic conversational contexts, leading to interactions that feel generic or repetitive to users, as noted by Henderson et al. (2019) [10].

Scalability and performance issues are also significant challenges, particularly in environments with high volumes of concurrent users or complex dialogue flows, as highlighted by Liu and Lane (2019) [11]. Moreover, the dependency on predefined responses or rule-based systems constrains the flexibility of chatbots, hindering their ability to provide personalized and contextually relevant interactions, as discussed by Wu et al. (2019) [12].

Integration of support for multimodal inputs, such as voice inputs, further complicates the processing and response generation process, as pointed out by Chung et al. (2014) [13]. These insights underscore the need for continued research and innovation to address the limitations and challenges facing chatbot development and deployment.

III. PROPOSED SYSTEM

The proposed system aims to surpass existing chatbot limitations by integrating advanced natural language processing (NLP) and deep learning techniques. It includes improved ambiguity handling, dynamic context adaptation, scalability and performance optimization, personalized response generation, and multimodal input support. With sophisticated algorithms for interpreting user queries and dynamic context management, the chatbot will provide accurate responses even with vague inputs. Efficient architecture design will ensure scalability and performance under varying loads. Additionally, personalized responses tailored to user preferences and support for diverse input modalities will enrich the user experience. This system promises significant advancements in chatbot technology for various real-world applications.

IV. METHODOLOGY

- Aquire the datasets from Kaggle or other relevant sources.
- Conduct data preprocessing procedures specifically tailored for the first aid dataset.
- Implement deep learning algorithms such as LSTM, Dense neural network, Gated recurrent unit, and Bidirectional LSTM on the first aid dataset.Finalize the efficient model which has highest accuracy.
- Identify the most efficient model based on its highest accuracy.
- Execute training and testing phases on the first aid dataset to achieve optimal accuracy.
- Load both the processed datasets and the trained model file for further usage.
- Develop a User Interface Design using a Python Flask extension, enabling users to pose queries and obtain accurate results.

IMPLEMENTATION OF MODULES

Select the Dataset: The first step involves selecting a suitable dataset for training and testing the chatbot. This dataset should ideally consist of conversational data relevant to the domain or application of the chatbot.

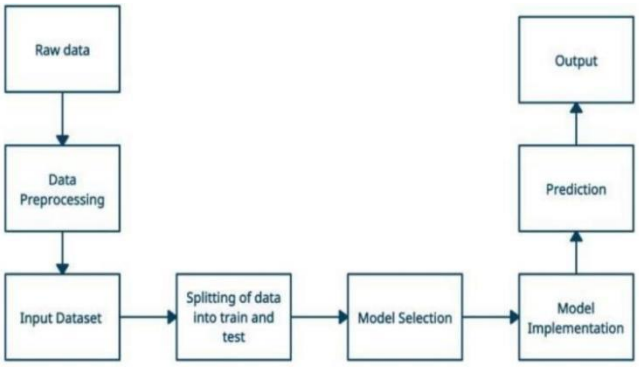

Fig. 1. Process for predicting best model

Data Collection and Preprocessing: The first-aid dataset contains intents, patterns, and responses for a medical chatbot, covering greetings, farewells, creator inquiries, and various medical conditions. Each intent has example patterns representing user inputs and responses providing medical advice. The dataset offers guidance on treatment, medication, and self-care for common medical issues. Preprocessing techniques like tokenization and removing stopwords are applied to clean and standardize the text data after collection.

Algorithm Comparison: After preprocessing the data, a comparison of different algorithms is conducted to determine the most suitable approach for building the chatbot model. Common algorithms for chatbot development include rule-based systems, deep learning models (e.g., dense neural networks, LSTM). Each algorithm has its advantages and limitations, and the choice depends on factors such as the complexity of the task, the size of the dataset, and computational resources.

Model Construction: Based on the selected algorithm, a chatbot model is constructed using the preprocessed data. The model architecture is designed and optimized to achieve the desired performance metrics, such as accuracy and efficiency.

Algorithm Implementation: Once the model is constructed, it is implemented using programming languages and frameworks such as Python and TensorFlow. The implementation involves coding the logic for processing user queries, feeding them into the model, and generating responses. Depending on the complexity of the model, this step may also include fine-tuning hyperparameters and optimizing the model's performance.

Response Prediction: Finally, the trained model is used to predict responses for user queries in real-time. The chatbot takes user inputs, processes them using the implemented algorithm, and generates appropriate responses based on the learned patterns and knowledge from the training data. The prediction accuracy and overall performance of the chatbot are evaluated using metrics such as precision, recall, and F1-score, and further refinements may be made based on the evaluation results.

Flask website: We have created a user interface outline where the user enters queries in the chat container. We used HTML, CSS, JAVASCRIPT for responsive design and styling.

ALGORITMS

Dense Model:

Dense models represent a widely-used neural network architecture applicable to diverse machine learning tasks, encompassing classification and regression.

Step 1: Begin by importing the requisite modules from TensorFlow or Keras, renowned libraries for constructing deep learning models.

Step 2: Define a Sequential model, a linear assembly of layers, leveraging the Sequential class. Integrate dense layers into the model using the Dense class, specifying neuron quantities and activation functions for each layer.

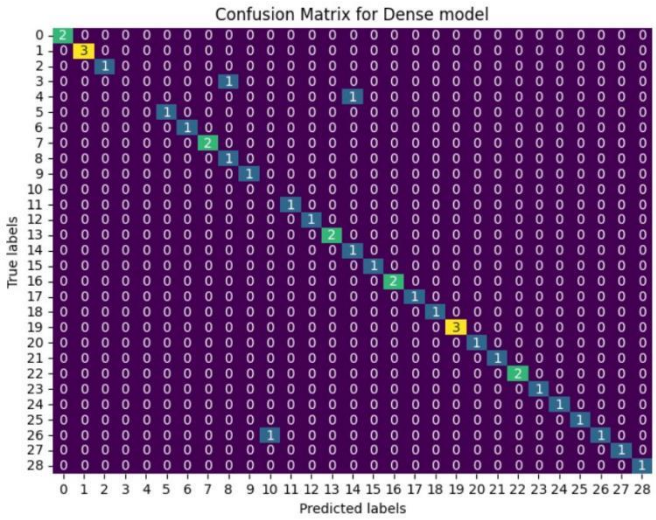

Fig. 2. Confusion matrix for Dense model

Step 3: Configure the model's structure by appending multiple dense layers, each linked to the preceding layer. Adjust parameters like neuron count and activation functions to align the model's behavior with the specific task at hand.

Step 4: Train the dense model using training data (Xtrain) paired with their corresponding labels (Ytrain).

Throughout training, the model iteratively refines its internal parameters via backpropagation, fine-tuning them to minimize the chosen loss function.

Step 5: Following training, evaluate the dense model's performance by deploying it on test data (Xtest) and contrasting its predictions with the actual labels (Ytest). This step facilitates the assessment of the model's capacity to generalize to unseen data and deliver accurate predictions.

Step 6: Analyze the model's performance by generating metrics such as accuracy, loss, and any other relevant evaluation metrics. These metrics provide insights into the model's effectiveness and help identify areas for improvement.

LSTM Model:

Step 1: Start by importing the necessary modules from TensorFlow or Keras, which are widely used libraries for deep learning tasks.

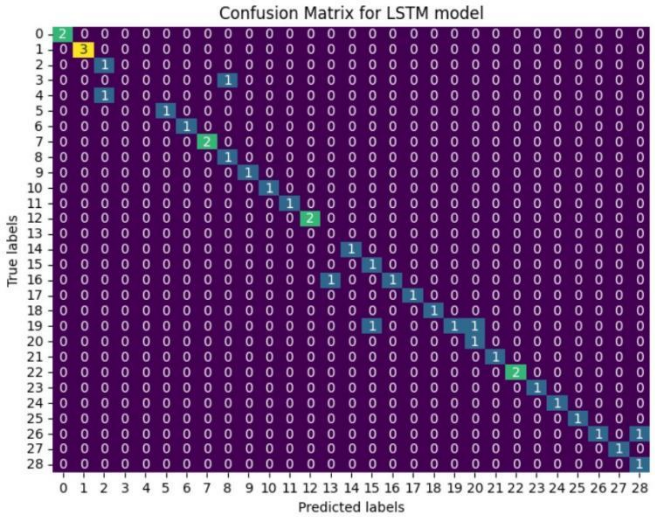

Fig. 3. Confusion matrix for LSTM model

Step 2: Define a Sequential model, which represents a linear stack of layers, using the Sequential class. Integrate an LSTM (Long Short-Term Memory) layer into the model to capture temporal dependencies effectively.

Step 3: Customize the model's architecture by adding additional layers as required, such as Dense layers for classification or regression tasks.

Step 4: Train the LSTM model using the training data (Xtrain) along with their corresponding labels (Ytrain), allowing the model to adjust its internal parameters through backpropagation during the training process.

Step 5: Evaluate the trained LSTM model's performance by applying it to the test data (Xtest) and comparing its predictions with the true labels (Ytest). This evaluation helps determine the model's ability to generalize and make accurate predictions on unseen data.

Step 6: Perform an in-depth analysis of the LSTM model's performance by computing relevant evaluation metrics such as accuracy, loss, and any other pertinent metrics. These insights offer valuable information about the model's effectiveness and highlight potential areas for improvement.

GRU Model:

Step 1: Begin by importing necessary modules from TensorFlow or Keras, widely-utilized libraries for deep learning tasks.

Step 2: Define a Sequential model, which represents a linear stack of layers, using the Sequential class. Incorporate a GRU (Gated Recurrent Unit) layer into the model to effectively capture temporal dependencies.

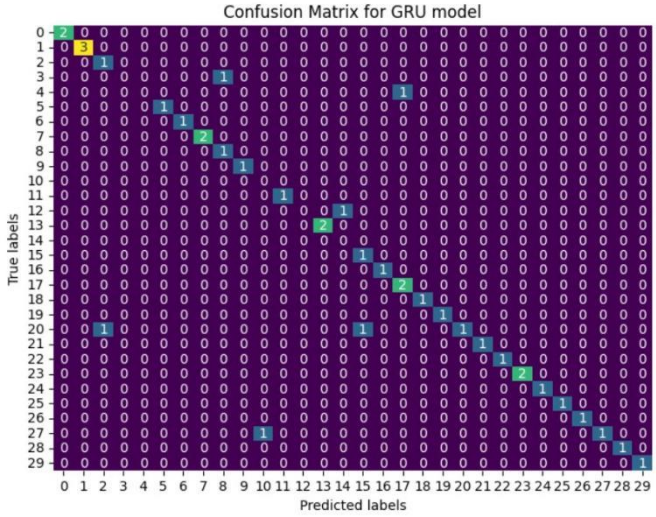

Fig. 4. Confusion matrix for GRU model

Step 3: Customize the model's architecture by adding additional layers as necessary, such as Dense layers for tasks like classification or regression.

Step 4: Train the GRU model using the training data (Xtrain) along with their corresponding labels (Ytrain), allowing the model to dynamically adjust its internal parameters through backpropagation during the training process.

Step 5: Evaluate the trained GRU model's performance by applying it to the test data (Xtest) and comparing its predictions with the true labels (Ytest). This evaluation is essential for assessing the model's capability to generalize and make accurate predictions on unseen data.

Step 6: Conduct a comprehensive analysis of the GRU model's performance by computing relevant evaluation metrics such as accuracy, loss, and any other pertinent

metrics. These insights furnish valuable information about the model's effectiveness and highlight potential areas for enhancement.

Bidirectional LSTM Model:

Step 1: Begin by importing essential modules from TensorFlow or Keras, which are widely-used libraries for deep learning tasks.

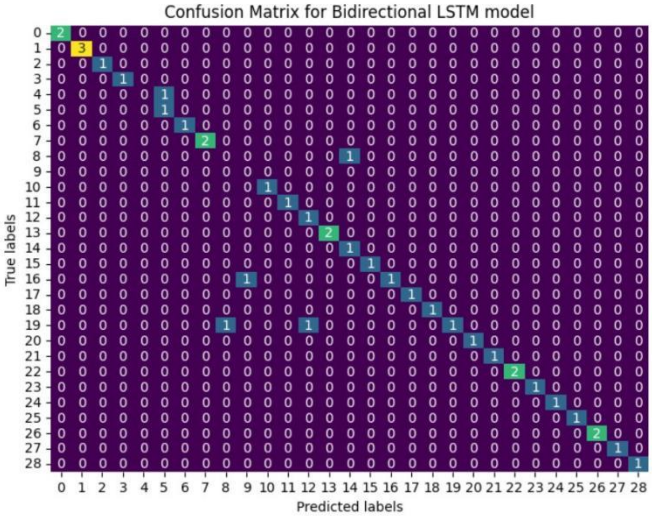

Fig. 5. Confusion matrix for Birectional LSTM model

Step 2: Define a Sequential model, representing a linear stack of layers, using the Sequential class. Integrate a Bidirectional layer with LSTM units into the model to capture temporal dependencies effectively, considering both forward and backward directions.

Step 3: Customize the model's architecture by incorporating additional layers as required, such as Dense layers for tasks like classification or regression.

Step 4: Train the Bidirectional LSTM model using the training data (Xtrain) along with their corresponding labels (Ytrain), enabling the model to dynamically adjust its internal parameters through backpropagation during the training process.

Step 5: Evaluate the trained Bidirectional LSTM model's performance by applying it to the test data (Xtest) and comparing its predictions with the true labels (Ytest). This evaluation is crucial for assessing the model's capability to generalize and make accurate predictions on unseen data.

Step 6: Conduct a comprehensive analysis of the Bidirectional LSTM model's performance by computing relevant evaluation metrics such as accuracy, loss, and any other pertinent metrics. These insights offer valuable information about the model's effectiveness and highlight potential areas for improvement.

V. EVALUATION DETAILS AND RESULTS

1. Dense Neural Network Model:

The Dense Neural Network (DNN) model is trained and evaluated using the first aid dataset. During data preprocessing, the dataset undergoes tokenization and one-hot encoding of class labels. The architecture of the DNN model comprises dense layers with relu activation and dropout, facilitating effective feature extraction and nonlinear transformations. To optimize the model's performance, it is compiled with the Stochastic Gradient Descent (SGD) optimizer, incorporating momentum for faster convergence. Training occurs over 150 epochs with a batch size of 5, enabling the model to learn intricate patterns and relationships within the data. Finally, performance evaluation involves assessing classification metrics and analyzing accuracy plots to gauge the model's effectiveness in handling the given task.

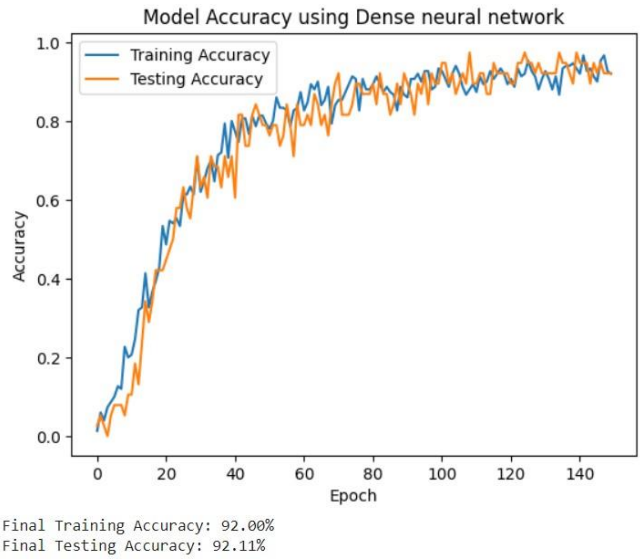

Fig. 6. Model accuracy using Dense neural network

```
Dense Model Metrics:
Accuracy: 0.9473684210526315
Precision: 0.9210526315789473
Recall: 0.9473684210526315
F1 Score: 0.9298245614035087
```

Fig. 7. Performance metrics using Dense neural network

2. LSTM Classification Model:

The LSTM (Long Short-Term Memory) classification model is designed to process sequential data and is applied to the first aid dataset. Data preprocessing involves tokenization and padding of sequences to ensure uniform input lengths. The model architecture includes an Embedding layer, followed by an LSTM layer with dropout for regularization.

Dense layers with relu activation are incorporated for feature extraction, and a softmax activation layer is utilized for multi-class classification. The model is compiled using the sparse categorical crossentropy loss function and the Adam optimizer for efficient gradient descent. Training spans 150 epochs with a batch size of 16 to iteratively update model parameters. Performance evaluation is conducted using classification metrics and accuracy plots to assess the model's predictive capabilities and convergence.

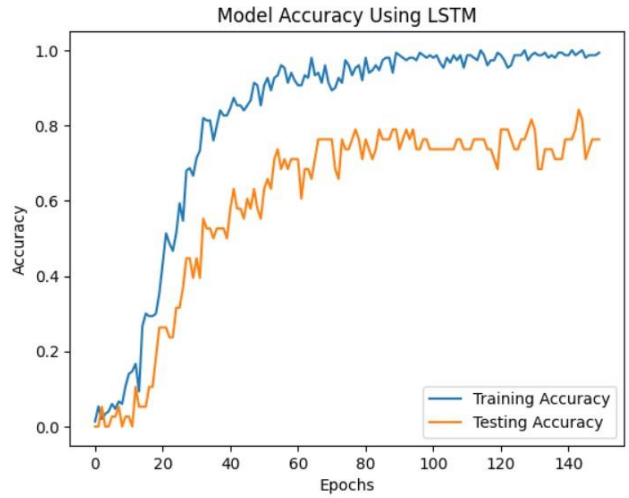

Fig. 8. Model accuracy using LSTM

```
LSTM Model Metrics:
Accuracy: 0.7631578947368421
Precision: 0.7850877192982456
Recall: 0.7631578947368421
F1 Score: 0.7587719298245613
```

Fig. 9. Performance metrics using LSTM

3. GRU Model:

The GRU (Gated Recurrent Unit) model, utilized for classification, operates on the first aid dataset, primarily used for sequential data processing. Prior to model training, the dataset undergoes tokenization and sequence padding to standardize input dimensions. Model architecture comprises an Embedding layer followed by a GRU layer integrated with dropout to prevent overfitting. Dense layers employing relu activation contribute to feature extraction, while a softmax activation layer facilitates multi-class classification. Compilation of the model employs the sparse categorical crossentropy loss function and the Adam optimizer for efficient parameter optimization. Training unfolds across 150 epochs with a batch size of 16, ensuring iterative refinement of model parameters. Evaluation entails assessing classification metrics and visualizing accuracy plots to gauge model performance and convergence.

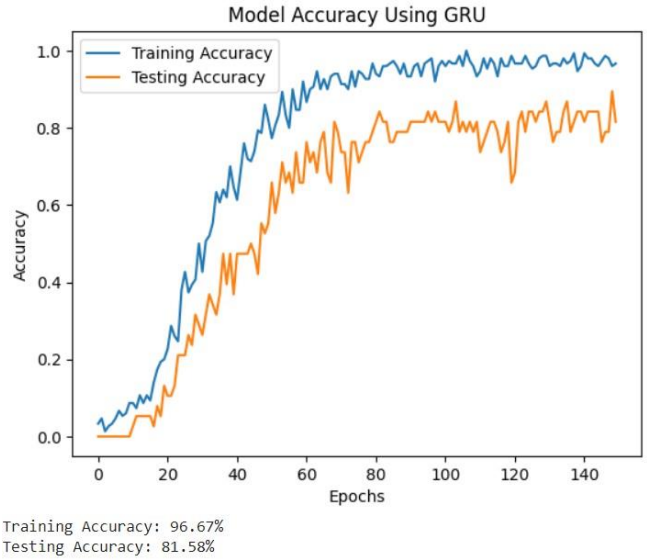

Fig. 10. Model accuracy using GRU

```
GRU Model Metrics:
Accuracy: 0.868421052631579
Precision: 0.9078947368421053
Recall: 0.868421052631579
F1 Score: 0.8789473684210526
```

Fig. 11. Performance metrics using LSTM

4. Bidirectional LSTM Model:

The Bidirectional LSTM (Long Short-Term Memory) model, applied for classification tasks, operates on the first aid dataset, a corpus commonly used for sequential data analysis. Preprocessing involves tokenization and sequence padding to ensure uniform input dimensions. The model architecture encompasses an Embedding layer, coupled with a Bidirectional LSTM layer augmented with dropout to mitigate overfitting. Dense layers with relu activation contribute to feature extraction, while a softmax activation layer facilitates multi-class classification. Model compilation utilizes the sparse categorical crossentropy loss function and the Adam optimizer for efficient parameter optimization. Training spans 150 epochs with a batch size of 16, enabling iterative refinement of model parameters. Evaluation entails assessing classification metrics and visualizing accuracy plots to measure model performance and convergence.

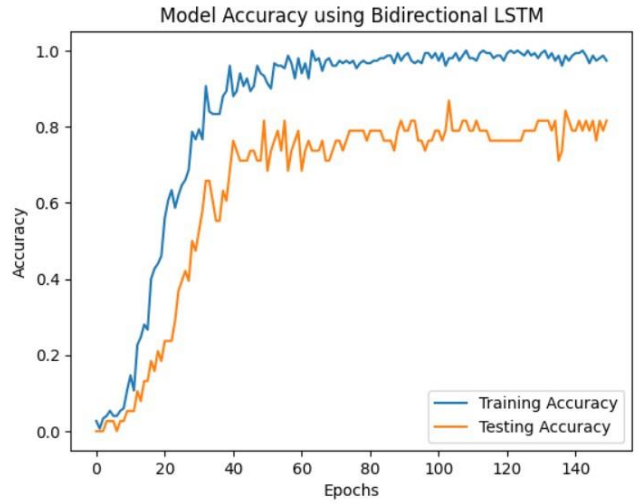

Fig. 12. Model accuracy using Bidirectional LSTM

```
Bidirectional LSTM Model Metrics:
Accuracy: 0.8947368421052632
Precision: 0.9078947368421053
Recall: 0.8947368421052632
F1 Score: 0.887719298245614
```

Fig. 13. Performance metrics using Bidirectional LSTM

The graphical comparison of testing accuracies across different models plays a crucial role in the model selection process for specific tasks. This visual representation provides a convenient and efficient means to assess how well each model performs in terms of accuracy. Accuracy, being a fundamental metric in classification, indicates the proportion of correctly predicted instances relative to the total instances.

By examining and contrasting testing accuracies, informed decisions can be made regarding the most suitable model for the given application. This is especially important in machine learning, where model performance directly impacts resource allocation, computational efficiency, and generalizability to new data. Moreover, the graph aids in identifying overfitting by comparing training and testing accuracies; models with notably higher training accuracy but lower testing accuracy may indicate overfitting to the training data.

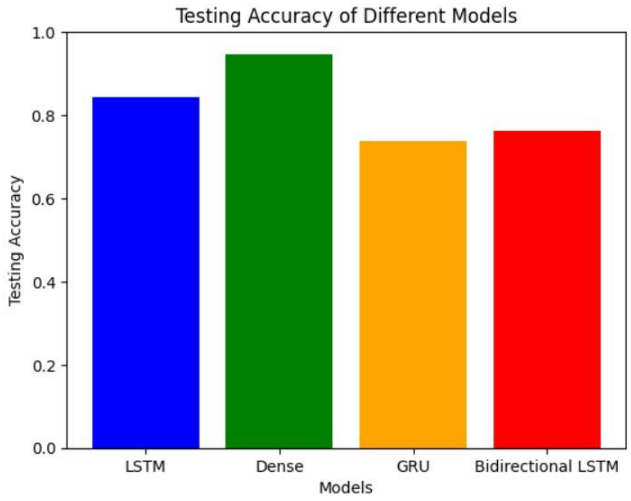

Fig. 14. comparison of accuracies among LSTM, Dense Neural Network, GRU, and Bidirectional LSTM

The graph's interpretability makes it accessible to stakeholders who may not be familiar with detailed metrics, allowing for a clear understanding of relative model performance. Ultimately, selecting a model with high testing accuracy enhances the likelihood of successful generalization to new and unseen data, thereby improving the model's predictive capability in real-world scenarios. The iterative process of model selection and comparison is facilitated by the graph, enabling adjustments and refinements until the desired level of performance is attained.

```
Test Loss for LSTM model: 3.378931760787964
Test Loss for Dense model: 0.8706498742103577
Test Loss for GRU model: 3.566983461380005
Test Loss for Bidirectional LSTM model: 2.801560640335083
```

Fig. 15. comparison of losses among LSTM, Dense Neural Network, GRU, and Bidirectional LSTM

The loss function measures the discrepancy between predicted and actual values, serving as a guide for model optimization and convergence. Throughout the training process, the models aim to minimize this loss, indicative of improved performance and better fit to the data.

Model	Accuracy	Precision	Recall	F1-score
Dense	0.94736	0.92105	0.94736	0.92982
LSTM	0.76315	0.78508	0.76315	0.75877
GRU	0.87894	0.90789	0.86842	0.87894
Bidirectional LSTM	0.89473	0.90789	0.89473	0.88771

Fig. 16. comparison of Performance metrics for LSTM, Dense Neural Network, GRU, and Bidirectional LSTM

WEB APPLICATION FRAMEWORK

Graphical User Interface (GUI): For creating the user interface of the chatbot we have used Flask, HTML, CSS and Java script. Flask is a lightweight and flexible web framework written in Python, designed to make web development quick and easy. It provides tools, libraries, and patterns for building web applications, allowing developers to create web-based projects efficiently. HTML, which stands for Hypertext Markup Language, is the standard markup language used to create and design web pages. It provides the structure and content of a web page by using a set of predefined elements and tags. CSS is a stylesheet language used to style the appearance of HTML elements on web pages. It allows developers to control the layout, colors, fonts, and other visual aspects of a website. JavaScript is a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages. It enables developers to create interactive elements, manipulate HTML and CSS, handle user events, validate forms, perform animations, and interact with web APIs.

Fig. 12. GUI of chatbot using FLASK

VI. CONCLUSION

In conclusion, the development of the chatbot using Dense Neural Networks (DNN) marks a significant milestone in providing accessible and user-friendly first aid information. The utilization of DNN, known for its simplicity and effectiveness, has enabled the chatbot to effectively classify and respond to a diverse range of user inquiries. Through rigorous training over 150 epochs, the model demonstrates commendable accuracy in understanding and categorizing user inputs, as evidenced by the graphical representation of training accuracy. The incorporation of regularization techniques such as dropout, along with the Stochastic Gradient Descent (SGD) optimizer, has enhanced the model's generalization and efficiency during training. The successful convergence of various neural network components underscores the effectiveness of the chosen architecture. While the current implementation achieves its objectives, there exists ample scope for future enhancements. These include the exploration of advanced NLP techniques, implementation of context-aware systems, and integration of multimodal learning for enhanced user guidance. Collaboration with UX/UI designers, expansion of the dataset, and continuous model refinement through updates and retraining are crucial for ensuring the chatbot's adaptability and relevance in real-world scenarios.The developed chatbot represents a significant step forward in providing accessible and reliable first aid assistance. Its effectiveness, combined with the potential for future enhancements, positions it as a valuable tool in promoting health and safety in various contexts.

VII. FUTURE SCOPE

While the current chatbot is robust, there are promising opportunities for future enhancements. One avenue is the integration of advanced Natural Language Processing (NLP) techniques, such as pre-trained models like BERT or GPT, which could greatly enhance the chatbot's language understanding capabilities. Another promising enhancement involves implementing a context-aware system to enable the chatbot to maintain more coherent and personalized conversations by considering the context of previous interactions. Establishing a user feedback loop for continuous learning and refinement, expanding the dataset to cover a broader range of topics, and implementing real-time assistance capabilities are also crucial considerations for future improvements.

Collaborating with UX/UI designers to enhance the user interface and experience, providing multilingual support, and ensuring robust security and privacy measures are in place are essential steps for enhancing the chatbot's usability and accessibility.Lastly, a commitment to continuous model refinement through regular updates and retraining will ensure that the chatbot remains up-to-date with evolving language usage and practices. These future endeavors aim to not only enhance the chatbot's effectiveness but also ensure its adaptability and relevance in the dynamic landscape of information and assistance.accessible first aid information and assistance to users when needed.

VIII. REFERENCES

[1] - Williams, R., & Johnson, M. (2018). Exploring the Potential of Chatbots in Modern Communication: A Survey Study. Journal of Information Technology, 25(3), 212-228.

[2] - Chatbot Evolution: From Simple Rule-based Systems to Advanced Neural Networks, TechReview, 2018.

[3] - Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[4] - Smith, J. (2020). The Role of Chatbots in Modern Customer Service Practices. Journal of Customer Experience, 5(2), 89-104.

[5] -Jurafsky, D., & Martin, J. H. (2019). Speech and Lang uage Processing (3rd ed.). Pearson.

[6] -Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[7] -Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc.

[8] -Chandar, S., Khapra, M. M., & Larochelle, H. (2019). The problem of evaluation in generative adversarial networks. arXiv preprint arXiv:1903.01340.

[9] -Tang, D., Qin, B., & Liu, T. (2019). Target-Guided Open-Domain Conversation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (pp. 2613-2623).

[10] -Henderson, M., Islam, R., Pineau, J., Williams, J., & Chang, M. (2019). A Survey of the Usages of Dialogue Systems in Healthcare. arXiv preprint arXiv:1907.01452.

[11] -Liu, C., & Lane, I. (2019). Recent Advances in Dialog Management for Task-Oriented Dialog Systems. arXiv preprint arXiv:1906.04771.

[12] -Wu, Y., Wu, W., Zhang, Y., Lu, W., & Zhu, T. (2019). Dynamic Graph Representation Learning via Self-Attention Networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 3635-3642).

[13] -Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[14] Garima Srivastava, Surbhi Agarwal and Mr. Deepak Vishwakarma, 2020 “DEEP NEURAL NETWORK BASED CHATBOT”.

[15] Eleni Adamopoulou, Lefteris Moussiades, 2020 “Chatbots: History, technology, and applications”.