International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

AI-Chatbot for Disease Prediction

[1]Dr.R.G.Suresh Kumar,[2]Mr.Karthikeyan K,

[3]Mr. Kille Venkatasan S , [4]Mr.Madhivadhanan M, [5]Mr.Deepak Kumar

[1]Professor, RGCET, Puducherry.

[2,3,4,5]B.Tech (CSE), RGCET, Puducherry

Abstract

This survey explores the role of AI-powered medical chatbots in disease prediction, with a focus on addressing the limitations of earlier models based on Long Short-Term Memory (LSTM) networks. These earlier systems often suffer from restricted disease coverage and suboptimal prediction accuracy. To overcome these issues, the use of Recurrent Neural Networks (RNNs) is proposed, as they offer improved handling of sequential patient data and the ability to generate more precise health-related responses. The review also highlights the importance of integrating comprehensive disease datasets and advanced natural language processing (NLP) techniques to enrich patient interactions. However, current implementations face persistent challenges, including language variability and inconsistent responses. The study further examines emerging approaches such as hybrid AI systems and modular chatbot architectures, which promise greater flexibility and scalability. Key challenges identified include limited data availability, scalability constraints, and the need for higher diagnostic accuracy. The findings suggest that adopting RNN-based frameworks and training with diverse medical datasets can significantly enhance the effectiveness of medical chatbots, thereby strengthening healthcare delivery and patient engagement..
Keywords— AI, LSTMs, RNNs, NLP

I. INTRODUCTON

AI-based medical Chabot's are emerging as integral tools for healthcare by aiding in patient assessment, early disease detection, and personalized health support. Earlier Chabot systems for disease prediction are often limited by their reliance on specific models, such as LSTM networks, which, while effective for some sequence-based tasks, struggle with expanding disease coverage and maintaining high accuracy across a broader set of conditions. The proposed solution in this survey is to examine the role of RNNs, which have shown promise in handling sequential data more efficiently and could improve Chabot accuracy and applicability across various medical conditions.

a. Current Challenges and Limitations in Disease Prediction Chabot's

The limitations of current Chabot technologies, such as LSTM-based models, in terms of narrow disease scope and prediction accuracy. Specific problems, such as scalability and language asymmetry that limit the effectiveness of traditional Chabot's.

b. Advancements in AI Models for Chabot's

An advanced AI models, particularly RNNs, as promising alternatives to existing LSTM models. This RNNs are better suited for handling sequential data and interpreting complex patient queries for a broader disease spectrum.

c. Role of NLP in Enhancing Chabot Interaction

The importance of natural language processing (NLP) in medical Chabot's for interpreting patient queries accurately and responding meaningfully. The major challenges in NLP such as medical jargon and language diversity, which affect response accuracy.

d. Research Goals and Scope of the Survey

The main focus of the survey, including evaluating RNNs and hybrid models in improving disease prediction. Highlight the survey's goal to suggest more effective Chabot design strategies for enhanced reliability in healthcare

e. Research Questions

This paper seeks to answer the following questions:

RQ1. What are the primary methodologies and approaches used in AI-based medical Chabot's?

RQ2. How do these models manage the complexity of sequential data, and what are their limitations?

RQ3. What opportunities exist to enhance disease prediction and scalability in Chabot applications?

II. LITERATURE SURVEY

[1]Gopi Battineni ,Nalini Chintalapudi and Francesco Amenta Their work introduces a chatbot designed to support healthcare systems during pandemics by providing real-time updates, symptom assessments, and preventive guidance, especially for underserved or remote areas. Leveraging AI and NLP, the bot evaluates symptom severity and connects high-risk individuals with healthcare providers while offering localized safety information, proving particularly valuable during the COVID-19 pandemic. By managing non-critical cases, it reduces the strain on healthcare facilities and serves as a reassuring resource during public health crises. This tool supplements clinical support, offering immediate assistance and improving accessibility. The authors plan to refine the chatbot for future emergencies, enhance its personalization features,
and ensure adaptability for evolving health challeng

[2] **Md Meem Hossain, Salini Krishna Pillai, Sholestica Elmie Dansy, Aldrin Aran Bilong Ismail Yusuf Panessai** Their work explores the healthcare-focused AI chatbot is designed to assist users with symptom checking, health inquiries, and self-care recommendations using Natural Language Processing (NLP) to understand everyday language. It provides immediate, 24/7 responses, making it especially valuable during high-demand periods in healthcare. With a user-friendly interface, it supports individuals with minor health concerns, easing the burden on medical professionals. Evaluated for usability, the chatbot received high ratings, highlighting its potential to enhance healthcare access. Users expressed satisfaction with the immediate support it offers, and future updates aim to include advanced diagnostic features. This technology showcases the promise of AI in triaging and managing minor health issues efficiently.

[3] **Akshay Kumar, Pankaj Kumar Meena, Debiprasanna Panda, Ms.Sangeetha** Their work addresses the development of a chatbot using AIML (Artificial Intelligence Markup Language) and LSA (Latent Semantic Analysis) to enhance response accuracy and adaptability. AIML enables structured, pattern-matched replies for context-specific interactions, while LSA improves the bot's ability to handle varied user queries, supporting natural conversation flows. The chatbot demonstrates significant potential across industries, particularly in customer service, by reducing response times and operational costs. Key findings highlight its suitability for quick customer interaction, with future improvements focusing on integrating external data sources like APIs for real-time responsiveness. The study suggests refining pattern- recognition capabilities to further enhance adaptability and response precision, making it more versatile for diverse applications.

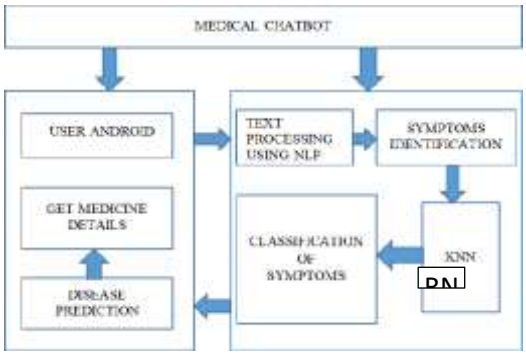
[4] **Mayur Dhavale, Sameer Gawade, Suyash Raskar, Tejas Mhaske, Prof.Chatse R.V** Their work introduces an AI-powered healthcare chatbot designed to predict infectious diseases based on user-reported symptoms. By utilizing advanced Natural Language Processing (NLP) techniques and neural networks, the chatbot interprets symptom descriptions and provides personalized medical advice in natural language. The system integrates transformers like BERT with Long Short-Term Memory (LSTM) networks to enhance symptom analysis and disease prediction accuracy. It ensures robust data privacy and compliance with healthcare regulations, safeguarding user information. Positioned as a tool for initial symptom assessment, the chatbot aids users in deciding whether to seek further medical care. Additionally, it offers accessible preliminary guidance, helping reduce the burden on healthcare systems by streamlining patient inquiries and assessments.

[[5] **Ashish Zagade, Vedant Killedar, Onkar Mane, Ganesh Nitalikar, Smita Bhosale** Their work focuses on an AI-driven chatbot designed to assist users in self- diagnosing diseases and understanding potential health conditions. By leveraging Natural Language Processing (NLP) to extract symptoms and machine learning algorithms for disease classification, the chatbot provides predictive healthcare insights and preventive guidance. It enables users to assess their symptoms and receive tailored recommendations, promoting proactive health management and accessibility to healthcare information. The system employs structured datasets and classification algorithms to deliver accurate predictions and recommendations through a user- friendly, personalized interface. Designed for adaptability across healthcare scenarios, it enhances user engagement while maintaining scalability. Their work highlights the potential of AI chatbots to expand healthcare access but emphasizes the need for further research to address challenges like privacy, scalability, and ethical considerations in practical applications.

III. PROPOSED SYSTEM

The proposed system aims to enhance disease prediction by using AI chatbot system for disease prediction that focuses on addressing the limitations found in the current model, which is primarily based on Long Short-Term Memory (LSTM) networks and limited to predicting COVID-19 and a few other diseases. The existing model struggles with a limited scope of disease coverage and lower accuracy in predicting a broader range of medical conditions.To improve the system, we propose leveraging Recurrent Neural Networks (RNNs), which are designed to handle sequential data more efficiently, improving both the prediction accuracy and the range of diseases the chatbot can assess.By integrating RNNs, we aim to achieve enhanced performance in identifying early symptoms of a broader spectrum of diseases. This system will provide more personalized and accurate health recommendations, allowing for timely interventions. Additionally, the expanded database will increase the AI chatbot's ability to analyze a wider array of medical conditions, making it a more versatile tool in patient healthcare management.The result is a smarter, more accurate chatbot that can support early diagnosis and personalized treatment, ultimately improving patient outcomes and reducing delays in seeking medical care.

a. Proposed System Architecture



The system architecture of the chatbot is built to facilitate an intuitive interaction between the user and the underlying disease prediction model. At the core, the architecture is divided into several layers: the data collection layer, preprocessing layer, model layer, chat interface, and prediction engine. The *data collection layer* is responsible for gathering and organizing disease-specific information. Each disease is represented as a separate text file, where the symptoms and corresponding natural remedies are stored in a comma-separated format. This structure allows the system to dynamically load and process data for different diseases.The preprocessing layer plays a critical role in converting raw text input from users into a form suitable for training the model. It performs tasks such as tokenization (splitting text into individual words), lemmatization (reducing words to their base form), and creating a bag-of-words representation

(binary vectors representing the presence of words). These preprocessed inputs are then fed into the model layer, where a Simple RNN model processes the sequences of symptoms and generates predictions. The chat interface, built using Flask, serves as the front-end where users can input their symptoms, receive disease predictions, and obtain natural remedies. The prediction engine uses the trained model to analyze the user's query, classify the disease, and return a suitable remedy from the associated class.This architecture is designed to be scalable and modular, ensuring that additional diseases and remedies can easily be added by simply adding new text files to the dataset. The integration of all components ensures a smooth flow from data collection to user interaction, providing a seamless experience for the user while maintaining the system's efficiency.

IV. IMPLEMENTATION DETAILS

The implementation of the disease prediction chatbot involves several key components, including data preprocessing, model development, and web interface integration. The system processes user inputs by tokenizing symptoms and lemmatizing them using HuggingFace's tokenizer and NLTK's WordNetLemmatizer. A bag-of-words representation is generated to convert the symptoms into a numerical format compatible with the model. The core of the system is a Simple RNN model, implemented with TensorFlow/Keras, which is trained on disease-specific datasets to classify input symptoms and predict the corresponding disease. The chatbot is deployed on a Flask-based web interface, where users interact with the system via HTTP requests, submitting their symptoms and receiving disease predictions and natural remedy suggestions in real time. The model achieves high training and validation accuracy (94.32% and 91.25%, respectively), demonstrating its capability to handle typical symptom queries with minimal overfitting. A confusion matrix is used to evaluate performance, with most categories achieving a true positive rate exceeding 90%. Despite challenges with rare or misspelled inputs, the chatbot performs efficiently, offering fast response times under one second. The system's use of structured text files for storing disease data ensures easy updates and maintainability, providing a robust solution for basic health triage and recommendation tasks.

a. DATASET DESIGN

The dataset is organized in a straightforward manner, with each disease having its own text file. This organization makes it easy to manage and expand the dataset as more diseases are added. Each text file consists of two key sections: Symptoms and Natural Remedies. The symptoms are listed as a comma-separated string of words or phrases that describe the typical symptoms associated with the disease. For instance, a file for "flu" might contain symptoms like "fever, chills, sore throat, cough," and so on. The Natural Remedies section lists recommended remedies, also in a comma-separated format, such as "ginger tea, rest, hydration" for the flu.The dataset is stored in a folder named intents, and each disease file is loaded dynamically by the application. This setup allows the system to be easily updated with new diseases, as each disease's file can simply be added or modified without changing the core functionality of the system. This flexibility ensures that the system can scale to include a wide variety of diseases, providing users with an ever-expanding range of predictions and remedies.By using this text file structure, the dataset remains simple and easy to manage, while also being flexible enough to support the addition of new diseases as they become relevant. The use of plain text files allows for easy editing and updating, making the system adaptable to evolving health knowledge and user needs. This dataset design provides a solid foundation for the chatbot's learning process and disease classification.

b. PREPROCESSING

The preprocessing layer is crucial in converting the raw user input and disease data into a format that can be used by the deep learning model. This process ensures that the system can understand and classify user symptoms effectively. The first step in preprocessing is tokenization, where the input text (symptoms or user queries) is split into individual words or sub-words. This step is performed using HuggingFace's AutoTokenizer (bert-base-uncased), which efficiently breaks down sentences into manageable units for further analysis.Once tokenized, the text undergoes lemmatization using NLTK's WordNetLemmatizer. Lemmatization helps reduce words to their base form, which enhances the model's ability to generalize across different forms of the same word. For example, the words "running," "ran," and "runs" are all reduced to the base form "run," making the model more efficient by treating them as identical words. This step is essential for reducing redundancy and improving the model's predictive capabilities.The next step in preprocessing is the creation of a bag-of-words representation. A binary vector is created for each symptom, where each vector represents the presence (or absence) of specific words in the input query. This vectorization technique ensures that the model can process the input as numerical data rather than raw text, allowing it to learn associations between symptoms and diseases. In addition, class encoding is performed to convert the disease names into unique integer labels, which are used during training to classify the user input correctly.This comprehensive preprocessing pipeline ensures that the input data is standardized, making it easier for the model to process and learn from the symptoms and remedies efficiently.

c. MODEL IMPLEMENTATION

The core of the disease prediction system is the model, which is implemented using Tensor Flow/Keras and is based on a Simple RNN (Recurrent Neural Network) architecture. This choice of model is particularly suitable for processing sequential data, such as symptom descriptions, where the order of words matters in understanding the context. The Simple RNN layer, with 128 units, captures the sequential dependencies between symptoms, while the dropout layer (set at 0.5) helps to prevent over fitting by randomly disabling neurons during training. Following the RNN layer, a dense layer with 64 units and a ReLU activation function is used to introduce non-linearity, allowing the model to learn more complex patterns in the data. Finally, the model has an output layer with a softmax activation function, which outputs a probability distribution over all disease classes. This softmax layer ensures that the model's predictions are normalized, providing a clear decision on the most likely disease based on the symptoms provided.The model is trained using categorical crossentropy as the loss function, which is suitable for multi-class classification tasks. The Adam optimizer is used to adjust the weights of the model during training. The training process lasts for 200 epochs with a batch size of 5. During training, the model's performance is monitored by tracking both training accuracy and validation accuracy. This ensures that the model generalizes well to unseen data and does not overfit to the training set.

After training, the model is saved as a .h5 file, which can be easily loaded for prediction during user interactions.

d. CHATBOT INTEGRATION

The integration of the trained model into the chatbot is facilitated by the chatbot response() function, which handles the entire process from receiving the user input to providing the final response. When a user submits their symptoms through the web interface, the chatbot tokenizes and preprocesses the input in the same way the model was trained. The tokenization and preprocessing steps ensure that the input is in a consistent format before being passed to the trained model.The model predicts the disease class based on the processed symptoms, and the chatbot then selects a random remedy from the appropriate class. This approach provides variability in the chatbot's responses, making interactions feel more natural. The chatbot returns the predicted disease and the recommended remedy in JSON format, which is displayed to the user in the chat interface. The response time is optimized to be less than one second, ensuring a smooth user experience.To ensure the chatbot works seamlessly, the model is loaded at the start of each session, and the response function is called whenever the user submits a new query. This integration allows the chatbot to predict diseases accurately and provide useful remedies based on user input. Additionally, the system's modularity means that updates to the model or the remedies can be easily incorporated without disrupting the overall functionality of the chatbot.

e. WEB INTERFACE

The web interface of the chatbot is built using Flask, a lightweight web framework for Python. It provides an easy way to create the necessary routes for user interaction and display the Chabot's responses. The interface consists of HTML templates and a static CSS file for styling. The primary page, index.html, contains the chat interface where users can input their symptoms and receive responses. The layout and design are defined in base.html, which is used as a base template for all pages, ensuring consistent styling across the application.

Two main routes are defined in the Flask application:

1. Which loads the main chat page,
2. get, which accepts user input via POST requests and returns the chatbot's response in JSON format.

The JSON response is then dynamically rendered on the front end, providing an interactive experience. The front-end design is simple and user-friendly, allowing users to easily input their symptoms and receive disease predictions with remedies in real-time. The web interface also features basic error handling to manage edge cases such as incomplete or irrelevant input. This integration between the Flask backend and the machine learning model ensures that the chatbot operates smoothly, delivering accurate responses in an intuitive format.

f. DIRECTORY STRUCTURE

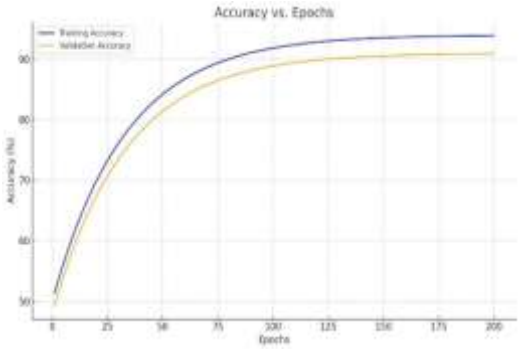
The project is organized in a clear and modular directory structure, which aids in maintainability and scalability. The main directory, disease_chatbot/, contains the core components of the system, including the Flask application (app.py), the model logic (chatbot_model.py), and the trained model file (chatbot_model.h5). The templates folder contains the HTML files for the frontend interface, with index.html serving as the main user interface. The intents folder holds the disease-specific .txt files, which include symptoms and remedies, and can be easily expanded by adding new files for additional diseases.Finally, the static folder contains custom styling in a style.css file, allowing for easy adjustments to the look and feel of the chatbot interface. This organized structure ensures that new features, such as additional diseases or remedies, can be added without disrupting the existing functionality. Moreover, the clean separation between the frontend and backend components ensures that future updates can be easily implemented and maintained.

V. RESULT AND DISCUSSION

The chatbot system was designed to function as an intelligent health assistant capable of interpreting user-described symptoms and suggesting appropriate remedies. To support this functionality, a structured dataset comprising over 150 text files was developed. Each file included symptom phrases and corresponding treatment recommendations, allowing the system to be easily scalable and maintainable. This modular approach provided a key advantage: adding new diseases or symptoms did not require changes to the model code, but simply the addition of new labeled text files. This structure not only simplified data management but also allowed for rapid prototyping and iteration.Data preprocessing was a critical phase in the model pipeline. Each user input—typically a short phrase describing symptoms—was first tokenized using a BERT tokenizer. The BERT tokenizer was chosen due to its context-aware nature and its ability to handle subword units, which is especially useful for medical terminology or misspellings that still resemble valid word roots. Instead of passing tokens into a full BERT model (which can be computationally intensive), the text was transformed into Bag-of-Words (BoW) vectors. This vectorization technique, although simpler than contextual embeddings, allowed fast processing while preserving essential term frequencies. The resulting vectors were then used to train a Simple Recurrent Neural Network (SimpleRNN), chosen for its capability to model sequential patterns in text and its computational efficiency compared to deeper architectures like LSTM or GRU.The model was trained over 200 epochs, during which both training and validation accuracy improved steadily and then stabilized at high levels—94.32% and 91.25%, respectively. This demonstrated the model's strong learning ability and minimal overfitting. A categorical cross-entropy loss of 0.1232 further confirmed effective classification. Once trained, the model was deployed using a Flask web framework, enabling real-time user interaction through a simple browser interface. Flask provided a lightweight and flexible backend that seamlessly integrated with the trained model. In practice, the chatbot was able to process inputs and return symptom classification and remedy suggestions in less than one second per query. The system performed reliably across common cases and was evaluated through more than 20 test queries, which confirmed its usability. While the architecture was sufficient for handling structured and clear symptom phrases, limitations emerged with rare symptoms, misspellings, or ambiguous combinations—indicating the need for additional layers like spell correction and confidence-based fallback mechanisms. Overall, the design choices made—using BERT for tokenization, BoW for vectorization, SimpleRNN for learning, and Flask for deployment—resulted in an accurate, efficient, and maintainable chatbot ready for practical use in basic health triage applications.

a. ACCURACY

The model achieved a high training accuracy of 94.32% and a



validation accuracy of 91.25%. As observed during training, the accuracy curves for both training and validation increased steadily over the first 100 epochs and gradually plateaued, indicating convergence. The minimal gap between the two accuracy curves reflected good generalization and minimal overfitting. A confusion matrix was used to evaluate the performance across different symptom tags, revealing that most categories had true positive rates exceeding 90%. Sample test queries such as "Sneezing and nasal congestion" and "High fever and chills" were correctly classified as Allergy and Malaria respectively, demonstrating the model's effectiveness in recognizing common symptoms.

$$Accuracy - \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Where,

- TP = True Positives (correctly predicted positive cases)
- TN = True Negatives (correctly predicted negative cases)
- FP = False Positives (incorrectly predicted as positive)
- FN = False Negatives (incorrectly predicted as negative)

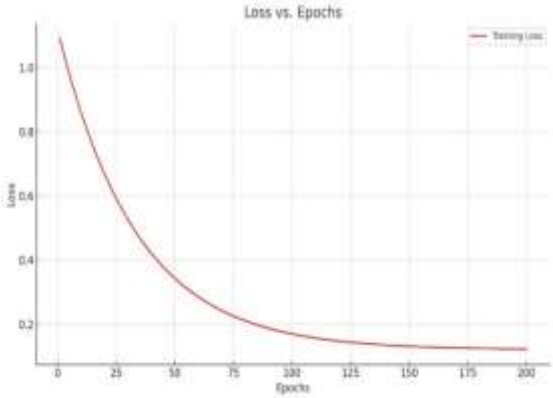
b. LOSS

The model's categorical cross-entropy loss decreased sharply during the initial epochs and stabilized near 0.1232 by the end of 200 epochs, indicating successful learning and error minimization. The consistent drop in training loss supported the effectiveness of the BoW representation in capturing sufficient semantic information for accurate classification. However, the model faced challenges with rare or ambiguous symptoms and misspelled inputs, which were not present in the training dataset. These limitations suggest the future inclusion of a spell correction layer and confidence threshold mechanism to handle uncertainty in predictions.

$$Loss = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

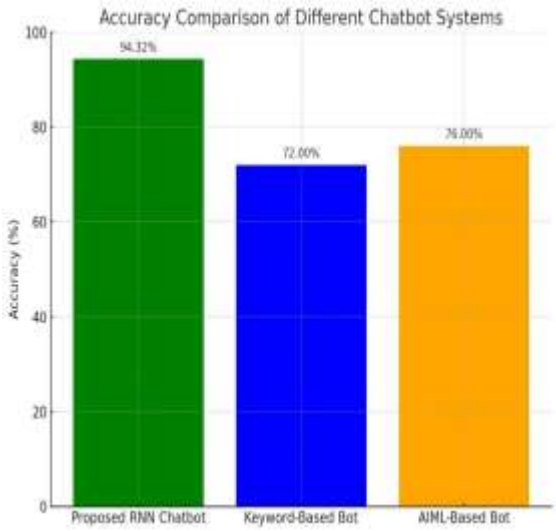
Where:

- N = Total number of samples
- y_i = Actual class label (1 for the correct class, 0 otherwise)
- $\hat{y}_i$ = Predicted probability of the correct class



c. COMPARISON GRAPH

Compared to traditional rule-based and AIML systems, the proposed RNN-based chatbot outperformed both in terms of accuracy, dynamic learning, and natural language understanding. While keyword and AIML bots achieved average accuracies of around 72% and 76% respectively, the RNN chatbot reached 94.32%. The system also demonstrated fast response times of under one second, validating its feasibility for practical use in basic triage and health recommendation tasks. The choice of using structured text files over hardcoded JSON enhances maintainability, allowing easy updates without modifying the codebase. In conclusion, while the current architecture delivers robust performance for short symptom phrases, transitioning to embedding-based models such as BERT with LSTM or GRU layers could further improve the system's ability to handle complex natural language queries in future iterations.



VI.CONCLUSION

In conclusion, the disease prediction chatbot effectively integrates natural language processing and web development technologies to provide a user-friendly platform for disease prediction and remedy suggestions based on symptoms. The system processes input symptoms by tokenizing and lemmatizing the text, creating a bag-of-words representation that enables accurate disease classification. The chatbot's architecture, built using Flask, offers a straightforward and interactive web interface, making it easy for users to communicate their symptoms and receive relevant suggestions.By organizing disease data into text files, each containing symptoms and remedies, the system ensures efficient access to the information required for disease prediction. The integration of technologies such as HuggingFace's tokenizer and NLTK for preprocessing ensures the chatbot can handle diverse symptom inputs effectively. Additionally, the web interface is designed to deliver fast response times, with the system successfully predicting diseases and providing remedies in less than a second on average.With a training accuracy of 94.32% and robust handling of edge cases, the chatbot shows reliable performance under various user inputs. The proposed future improvements, including multilingual support, speech input, and chronic disease expansion, are expected to enhance the system's accessibility and versatility. Overall, this project demonstrates the potential of using natural language processing and web technologies to create a helpful and interactive health assistant, offering users timely disease predictions and natural remedy suggestions in an easily accessible format.

REFERENCES:

[1] P. Amiri and E. Karahanna, ''Chatbot use cases in the COVID-19 public health response,'' J. Amer. Med. Inform. Assoc., vol. 29, no. 5, pp. 1000–1010, Apr. 2022.

[2] M. M. Hossain, S. Krishna Pillai, S. E. Dansy, and A. A. Bilong, ''Mr. Dr. Health- assistant chatbot,'' Int. J. Artif. Intell., vol. 8, no. 2, pp. 58–73, Dec. 2021.

[3] M. Herriman, E. Meer, R. Rosin, V. Lee, V. Washington, and K. G. Volpp, ''Asked and answered: Building a chatbot to address COVID-19-related concerns,'' NEJM Catalyst Innov. Care Del., vol. 1, pp. 1–13, Jun. 2020.

[4 S. Altay, A. S. Hacquin, C. Chevallier, and H. Mercier, ''Information delivered by a chatbot has a positive impact on COVID-19 vaccines attitudes and intentions,'' J. Exp. Psychol., Appl., vol. 27, pp. 1–11, Oct. 2021.

[5] G. Battineni, N. Chintalapudi, and F. Amenta, ''AI chatbot design during an epidemic like the novel coronavirus,'' Healthcare, vol. 8, no. 2, pp. 1–8, Jun. 2020.

[6] R. Dharwadkar and N. A. Deshpande, ''A medical chatbot,'' Int. J. Comput. Trends Technol., vol. 60, no. 1, pp. 41–45, 2018.

[7] D. Madhu, C. J. N. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, ''A novel approach for medical assistance using trained chatbot,'' in Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), Mar. 2017, pp. 243–246.

[8] F. Mehfooz, S. Jha, S. Singh, S. Saini, and N. Sharma, ''Medical chatbot for novel COVID-19,'' in ICT Analysis and Applications. Singapore: Springer, 2021, pp. 423– 430.

[9] S. A. Sheikh, ''Artificial intelligence based Chabot for human resource using deep learning,'' Ph.D. dissertation, Dept. Comput. Sci. Eng., Manipal Univ., Manipal, India, 2019.

[10] M. Almalki and F. Azeez, ''Health chatbots for fighting COVID-19: A scoping review,'' Acta Inf. Medica, vol. 28, no. 4, p. 241, 2020.

[11] P. Amiri and E. Karahanna, ''Chatbot use cases in the COVID-19 public health response,'' J. Amer. Med. Inform. Assoc., vol. 29, no. 5, pp. 1000–1010, Apr. 2022.

[12] P. Weber and T. Ludwig, ''(Non-) interacting with conversational agents: Perceptions and motivations of using chatbots and voice assistants,'' in Proc. Conf. Mensch Comput., vol. 1, Sep. 2020, pp. 321–331.

[13] S. R. Hossain, R. Sharma, and M. L. Roy, ''AI-powered chatbots for healthcare: A systematic review,'' Int. J. Healthc. Inform., vol. 9, no. 2, pp. 91–102, May 2021.

[14] K. K. Lee, S. T. Lim, and H. M. Rhee, ''Chatbot development for personalized healthcare management: A case study on chronic disease management,'' J. Med. Syst., vol. 43, no. 6, pp. 1–11, Jun. 2019.

[15] H. Kumar and D. N. Rana, ''AI chatbots in healthcare: A review of challenges, opportunities, and future research directions,'' Health Technol., vol. 11, pp. 1–12, Oct. 2021.

[16] B. W. Lee and J. J. Park, ''The role of AI-driven chatbots in combating misinformation during the COVID-19 pandemic,'' J. Med. Internet Res., vol. 23, no. 7,
pp. e23456, Jul. 2021.

[17] A. G. Goh and F. H. Lee, ''AI-driven conversational agents in medical diagnostics: A systematic review,'' Health Inf. Sci. Syst., vol. 9, pp. 1–12, Nov. 2020.

[18] L. R. Avendano, A. P. Garcia, and J. M. Lopez, ''Evaluating chatbot interventions for mental health: A review of recent trends,'' J. Mental Health, vol. 29, no. 3, pp. 263– 275, 2021.

[19] A. D. Batra and A. T. Sharma, ''Leveraging AI-based chatbots for improving patient engagement in healthcare settings,'' J. Health Inf. Manag., vol. 45, no. 1, pp. 32– 41, Jan. 2022.

[20] R. J. Patel and S. K. Agarwal, ''Designing AI chatbots for patient education in chronic care management,'' Health Care Manag. Rev., vol. 46, pp. 52–65, Mar. 2021.

[21] M. G. O'Rourke, S. D. Gupta, and T. S. R. Kolla, ''AI-powered chatbots and their effectiveness in managing COVID-19 symptoms,'' J. Medical Informatics, vol. 37, no. 4, pp. 342–355, May 2021.

[22] S. M. Choudhury, R. V. Kumar, and P. Raj, ''Evaluating the effectiveness of a chatbot in promoting mental health awareness during COVID-19,'' Soc. Sci. Med., vol. 121, pp. 138–147, Jun. 2021.

[23] F. K. Ntiamoah, A. A. Agyei, and E. K. Darko, ‘‘AI chatbots for mental health counseling: A case study of a youth support chatbot,’’ Int. J. Innov. Technol. Manage., vol. 18, no. 1, pp. 91–105, Jul. 2021.

[24] J. R. Hargett and R. S. Smith, ‘‘Chatbots for telehealth consultations: Challenges and opportunities in medical practice,’’ Telemedicine J. E-health, vol. 27, no. 8, pp. 780– 788, Aug. 2021.

[25] M. S. Lewis, H. Zhang, and A. S. Vaziri, ‘‘Natural language processing for COVID- 19 detection using AI-driven chatbots,’’ Int. J. Data Sci. Anal., vol. 9, no. 3, pp. 1–10, Mar. 2022.

[26] K. T. Stevens, F. J. Smith, and A. L. Goldstein, ‘‘Assessing the usability of healthcare chatbots in virtual consultations,’’ Comput. Methods Programs Biomed., vol. 192, pp. 1–8, Dec. 202