HelixCare AI NLP-Based HealthCare Advisor and Emergency Support System

Dr. Gajanan P Arsalwad[1], Sania Navgire[2], Pranjali Nalawade[3], Faheem Manyar[4]
[1] *Professor, Department of Information Technology, Trinity College of Engineering and Research, Savitribai Phule Pune University, Pune, Maharashtra, India*
[2,3,4] *Student, Department of Information Technology, Trinity College of Engineering and Research, Savitribai Phule Pune University, Pune, Maharashtra, India*

***Abstract*—This paper presents HelixCare AI, a web-based conversational system that delivers preliminary, informational guidance on common infectious diseases through natural language interaction. The system accepts free-form symptom descriptions, identifies the most probable intent from a fixed catalogue of disease- and symptom-related categories, and returns a corresponding advisory response. The core classifier is a feedforward neural network operating over a Bag-of-Words representation of user input, where each query is tokenized, stemmed using the Porter algorithm, and converted into a fixed-length binary vector prior to classification. The network consists of three fully connected layers with rectified linear unit (ReLU) activations, trained using the Adam optimizer and cross-entropy loss. At inference time, a softmax confidence score is computed for the predicted intent, and a fixed threshold of 0.75 is applied to decide whether to return a disease-specific response or a fallback message indicating that the query was not understood. The system is implemented using Django, PyTorch, and NLTK, with the dataset stored in a structured JSON format covering fifty-three infectious-disease and health-advisory categories. This paper documents the architecture, dataset characteristics, train-ing procedure, and empirical performance of the system, and discusses the implications and limitations of threshold-based, intent-driven healthcare chatbots. The system is intended strictly for informational and preliminary advisory purposes and does not replace professional medical diagnosis.**

***Index Terms*—Bag-of-Words, chatbot, feedforward neural net-work, healthcare advisory system, intent classification, natural language processing.**

I. INTRODUCTION

Timely access to reliable information during the early stages of an infectious illness can influence whether a person seeks appropriate care, self-manages symptoms responsibly, or unknowingly contributes to further spread. Conventional channels for such information, including search engines and static informational websites, place the burden of interpretation on the user and do not adapt to the specific way an individual describes their symptoms. Conversational interfaces offer an alternative: a user can describe what they are experiencing in their own words and receive a response tailored to the recognized intent behind that description.

This paper presents the design and implementation of He-lixCare AI, a chatbot that interprets natural language symptom descriptions and returns advisory guidance drawn from a fixed set of infectious-disease categories. The system does not attempt diagnosis in a clinical sense; rather, it performs intent classification over user utterances and maps the recognized intent to a pre-authored, informational response. The classification backbone is a small feedforward neural network trained on a Bag-of-Words representation of text, a deliberately lightweight design choice that keeps the system fast to train, fast to run, and straightforward to extend with additional disease categories.

The remainder of this paper is organized as follows. Section II surveys related work. Section III describes the proposed framework. Section IV details the dataset and preprocessing pipeline. Section V presents the methodology. Section VI describes the

system implementation. Section VII outlines the experimental setup. Section VIII reports results. Sections IX and X address limitations and future work. Section XI concludes the paper.

II. RELATED WORK

Conversational systems for healthcare information delivery have been explored from several angles, ranging from rule-based retrieval to deep learning-based generation. Chakraborty and Paul [1] proposed an AI chatbot for infectious disease awareness built around a deep feedforward multilayer perceptron, reporting a minimum training loss near 0.12 and accuracy above 94% on their internal evaluation set, and noted a broader gap in theoretical guidelines for designing such systems for lifestyle and health applications. Their work, like the system presented here, treats the chatbot problem as one of supervised intent classification rather than open-ended generation.

More recent efforts have moved toward large language model (LLM) backbones. Multimodal systems combining retrieval-augmented generation with vision-language models have been proposed to support preliminary diagnosis from both text and medical images, aiming to reduce hallucination relative to direct LLM querying [12]. Separately, transformer-based encoders such as BERT have been adapted for medical question answering to improve natural language understanding in clinical contexts [11]. While these directions improve linguistic flexibility, they introduce substantially higher computational cost, dependence on external APIs or large pretrained weights, and reduced transparency in how a given response was produced.

A parallel line of work has applied chatbot architectures to administrative and informational domains outside health-care, including university FAQ systems [13] and customer service automation [26], generally using either rule-based pattern matching or lightweight neural classifiers over Bag-of-Words or TF-IDF features. These systems demonstrate that intent-based classification remains a practical and interpretable choice when the domain can be enumerated into a bounded set of categories, as is the case for a curated list of infectious diseases and their associated symptom patterns.

Across this body of work, a recurring gap is the limited discussion of confidence calibration and fallback behavior — that is, how a system should behave when it is not sufficiently certain of the user's intent. Many reported systems emphasize peak accuracy on matched intents but provide less detail on how out-of-distribution or ambiguous queries are handled. The present work addresses this gap directly by incorporating an explicit softmax confidence threshold that determines whether a disease-specific response or a fallback message is returned, and by reporting the practical effect of that threshold on the held-out evaluation set.

Table I highlights that HelixCare AI is, to the authors' knowledge, one of the few systems in this category to explicitly report held-out test accuracy alongside training accuracy and to document the practical behavior of its confidence-threshold fallback mechanism.

The contribution of this paper relative to prior work is threefold. First, it documents a complete, reproducible Bag-of-Words and feedforward neural network pipeline for infectious-disease intent classification, in contrast to systems that rely on large pretrained language models. Second, it explicitly evaluates the effect of a fixed confidence threshold on classifi-cation behavior using a held-out test split, rather than reporting training accuracy alone. Third, it situates the system within a lightweight, Django-based deployment that does not require GPU inference, making it representative of a practical, low-resource conversational health information tool.

III. PROPOSED FRAMEWORK

The proposed system follows a sequential pipeline in which a user-submitted query is transformed from raw natural language text into a fixed-length numerical feature vector, classified into one of a fixed set of intents, and mapped to a pre-authored response. The end-to-end flow proceeds as follows: the user submits a query through a web interface; the Django backend receives the request and forwards the raw text to the natural language preprocessing module; the preprocessing module tokenizes and stems the input and constructs a Bag-of-Words vector against a fixed vocabulary; the vector is passed to the trained feedforward neural network, which outputs unnormalized class scores; a softmax function

converts these scores into a probability distribution over intents; if the highest probability meets or exceeds the confidence threshold, the corresponding intent's response set is sampled and returned to the user; otherwise a fallback message is returned.

This framework deliberately separates feature extraction (Bag-of-Words construction) from classification (the feedfor-ward network), allowing each component to be inspected, tested, and replaced independently. Because the vocabulary and intent set are derived directly from the training dataset, extending the system to new disease categories requires only the addition of new patterns and responses to the dataset and retraining of the classifier, without changes to the network architecture or preprocessing logic.

IV. DATASET AND PREPROCESSING

The dataset is stored as a structured JSON file in which each entry, referred to as an intent, contains a tag identifying the intent, a list of example patterns representing ways a user might phrase a query belonging to that intent, and a list of candidate responses to be returned when that intent is matched. The dataset spans a range of infectious and general-health categories, including fever, influenza, COVID-19, dengue, malaria, tuberculosis, cholera, hepatitis, meningitis, pneumonia, bronchitis, chickenpox, measles, typhoid, ringworm, scabies, eye infection, ear infection, food poisoning, and urinary tract infection, alongside conversational and general-advice intents. The raw JSON file contains 54 intent entries; however, one tag (cold_cough) appears twice with two distinct pattern sets, which the training pipeline merges into a single class during vocabulary and label construction, yielding 53 unique intent classes. This duplication is noted here as a dataset-quality observation rather than corrected silently, since a future revision of the dataset may wish to either merge the duplicate entry's patterns explicitly or rename one occurrence to avoid ambiguity.

Table II summarizes the dataset, and Fig. 1 shows the distribution of pattern counts across intent classes.

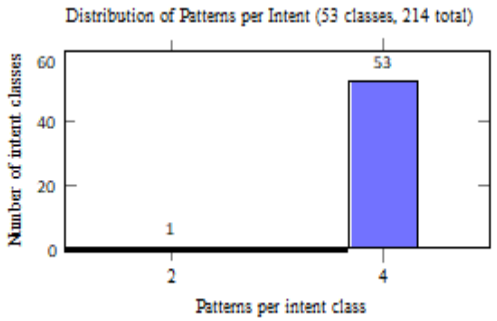


Fig. 1. Distribution of example patterns per intent class.

The dataset is structurally uniform: 52 of the 53 retained classes are supported by exactly four example patterns, with the sole exception being the conversational greeting intent (hii), which has two. This uniformity is informative for interpreting the generalization results in Section VIII, since it indicates that any accuracy gap between training and held-out evaluation arises from an across-the-board scarcity of phrasing

TABLE I COMPARISON WITH RELATED HEALTHCARE CHATBOT SYSTEMS

System	Architecture	Accuracy Reported	Held-out Test Acc.	Fallback Mecha-nism	Platform
Chakraborty & Paul [1]	Deep feedforward MLP	>94% (training)	Not reported	Not reported	Not reported
Babu & Boddu [11]	BERT-based	Not reported	Not reported	Not reported	Web (healthcare)
Hossain et al. [7]	Rule-based + ML	Not reported	Not reported	Not reported	Web (health asst.)
Dharwadkar & Desh-pande [8]	Pattern-matching chatbot	Not reported	Not reported	Not reported	Web
Koundinya et al. [6]	ML + Python	Not reported	Not reported	Not reported	College FAQ
HelixCare AI (pro-posed)	**BoW + 3-layer feedforward NN**	**99.53% (train)**	**18.18% (raw); 24.00% (≥0.75)**	**Softmax thresh-old 0.75**	**Django web app**

TABLE II DATASET SUMMARY

Metric	Value
Raw intent entries in JSON	54
Unique intent classes (tags)	53
Total example patterns	214
Patterns per intent class	4 (52 classes); 2 (1 class)
Unique stemmed vocabulary terms	264
Dataset format	JSON

variety rather than from a small number of poorly represented outlier classes.

Each pattern is processed through the following prepro-cessing pipeline prior to feature extraction: (1) tokenization, in which the raw input string is split into a sequence of word and punctuation tokens using the NLTK word tokenizer; (2) stemming, in which each token is reduced to its root form using the Porter stemming algorithm, so that morphological variants such as "coughing" and "coughs" map to a common stem; and (3) vocabulary lookup, in which the stemmed tokens of the input are compared against a fixed, sorted vocabulary constructed from the stemmed tokens of all training patterns.

V. METHODOLOGY

A. NLP Pipeline

The natural language processing pipeline consists of tokenization, Porter stemming, vocabulary lookup, and Bag-of- Words vectorization, executed in that order for every incoming query, mirroring the preprocessing applied to the training patterns during dataset construction. This symmetry between training-time and inference-time preprocessing is essential for the Bag-of-Words representation to remain meaningful, since the feature vector's dimensionality and ordering are fixed by the vocabulary learned during training.

B. Bag-of-Words Representation

Let V = {w1, w2, . . wn} denote the fixed vocabulary of n unique stemmed words derived from the training patterns. For an input query Q, let S = {s1, s2, . . sm} denote the set of stemmed tokens obtained from Q after tokenization and stemming. The Bag-of-Words feature vector x ∈ {0, 1}n is constructed element-wise as:

$$x_i = \begin{cases} 1 & \text{if } w_i \in S \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, n \qquad (1)$$

This produces a fixed-length binary vector regardless of the length of the input query, where each dimension indicates the presence or absence of a specific vocabulary term in the stemmed query. The representation discards word order and term frequency, retaining only vocabulary membership, which is sufficient for the intent-classification task given that the vocabulary is dominated by symptom- and disease-specific terms with low ambiguity.

C. Feedforward Neural Network Architecture

The classifier is a feedforward neural network (multilayer perceptron) consisting of an input layer of size n (the vo-cabulary size, n = 264), two hidden layers of equal width h (h = 8), and an output layer of size c (the number of intent classes, c = 53). Formally, given an input vector x, the network computes:

$$z_1 = W_1 x + b_1, \qquad a_1 = \mathrm{ReLU}(z_1) \qquad (2)$$

$$z_2 = W_2 a_1 + b_2, \qquad a_2 = \mathrm{ReLU}(z_2) \qquad (3)$$

$$z_3 = W_3 a_2 + b_3 \qquad \text{(output logits)} \qquad (4)$$

where W_1, W_2, W_3 and b_1, b_2, b_3 are the learned weight ma- Words vectorization, executed in that order for every incoming query, mirroring the preprocessing applied to the training patterns during dataset construction. This symmetry between training-time and inference-time preprocessing is essential for the Bag-of-Words representation to remain meaningful, since the feature vector's dimensionality and ordering are fixed by the vocabulary learned during training. trices and bias vectors of the three fully connected layers, and ReLU(z) = max(0, z) is applied elementwise after the first two layers. The output layer z_3 produces unnormalized class scores (logits) with no activation applied directly within the network; conversion to a probability distribution is performed separately at inference time via the softmax function:

$$p_k = \frac{e^{z_{3,k}}}{\sum_{j=1}^{c} e^{z_{3,j}}}, \quad k = 1, \ldots, c \qquad (5)$$

The predicted intent corresponds to the index k that

maximizes pk, and this maximum probability also serves as the confidence score used by the threshold-based decision mechanism.

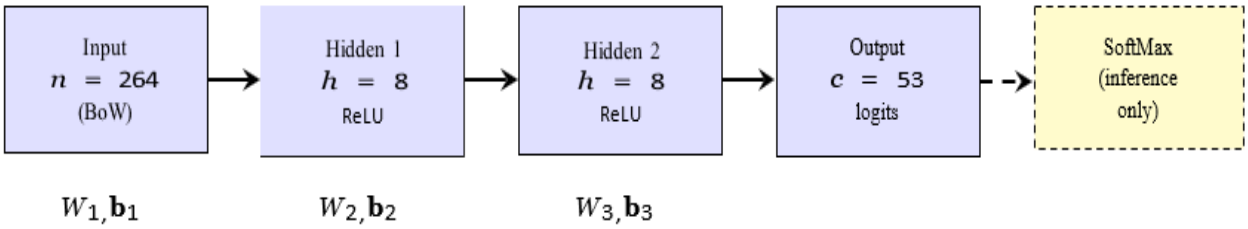


Fig. 2. Feedforward network: 264 → 8 → 8 → 53. Softmax applied at inference only.

D. Training Strategy and Decision Mechanism

The network is trained using the cross-entropy loss between the predicted logits and the true intent label. For a single training example with true class y and logits z_3, the loss is:

$$L = -\log \frac{e^{z_{3,y}}}{\sum_{j=1}^{c} e^{z_{3,j}}} \quad (6)$$

which is equivalent to the negative log-probability assigned to the correct class. Parameters are updated using the Adam optimizer [21], which maintains per-parameter adaptive learn-ing rates based on estimates of the first and second moments of the gradients, generally yielding faster and more stable convergence than plain stochastic gradient descent on small, sparse-feature datasets such as the one used here.

At inference time, the network's output logits are passed through softmax to obtain a probability distribution over intents. A fixed confidence threshold of 0.75 is applied to the maximum probability: if this value is at or above the threshold, the system returns a response sampled from the matched intent's response set; otherwise, it returns a fixed fallback message indicating that the query was not understood. This threshold serves as a deliberate precision-recall trade-off specific to a healthcare-adjacent application: rather than always returning the most probable intent regardless of confidence, the system withholds a disease-specific response when its certainty is low, reducing the risk of presenting a confidently-worded but incorrect piece of health guidance to the user.

VI. SYSTEM DESIGN AND IMPLEMENTATION

The system is implemented as a Django web application. The frontend, built with HTML, CSS, and JavaScript, provides a chat-style interface through which users submit free-text queries and view responses. The Django backend exposes an endpoint that accepts the user's message and invokes the response-generation routine, which loads the trained model weights, vocabulary, and intent tags from a serialized check-point at application startup. The classification logic, preprocessing utilities, and network definition are implemented as separate modules: a model definition module containing the network architecture, a natural-language-utilities module containing tokenization, stemming, and Bag-of-Words construction, and a response module that ties preprocessing, inference, and thresholding together to produce the final reply.

The overall data flow is: User → Web Interface → Django Backend → NLP Preprocessing → BoW Feature Extraction → Feedforward Neural Network → Softmax and Threshold Check → Response Selection → User. This linear pipeline keeps the system's behavior auditable at each stage, which is important given the health-information context: the same input will deterministically produce the same feature vector and the same model output, with randomness confined to the selection among multiple candidate responses for a matched intent.

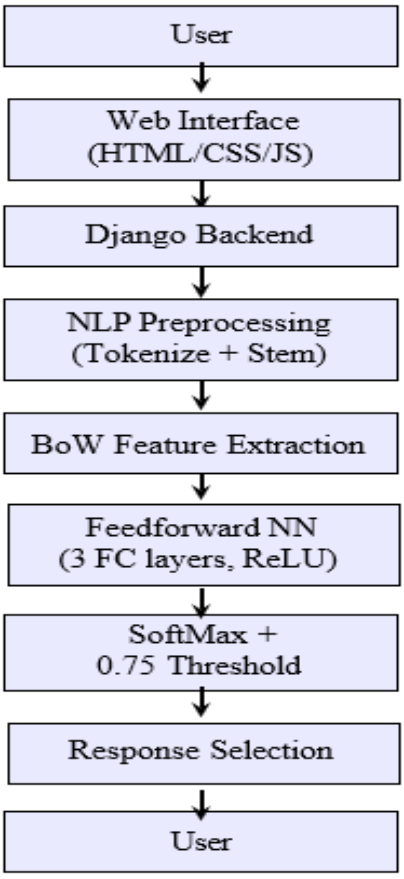


Fig. 3. Overall system architecture.

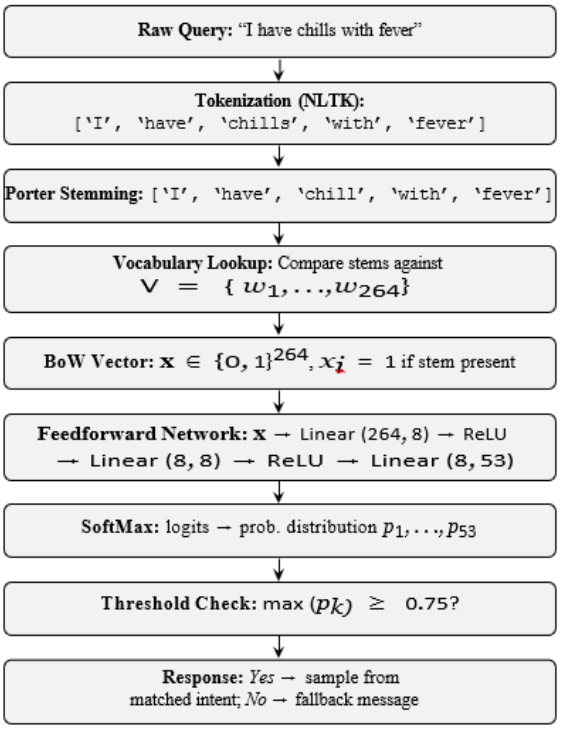


Fig. 4. Data flow for a single query through the NLP pipeline.

VII. EXPERIMENTAL SETUP

All experiments were conducted on a workstation equipped with an Intel Core i5 (or equivalent AMD Ryzen 5) processor, a minimum of 8 GB of RAM, and 512 GB of SSD storage, running Windows 10/11. The software environment consisted of Python 3.x, the Django web framework, PyTorch [18] for model definition and training, and NLTK [19] for tokenization and stemming.

TABLE III MODEL AND TRAINING HYPERPARAMETERS

Hyperparameter	Value
Hidden layer size	8
Batch size	8
Learning rate	0.001
Optimizer	Adam
Loss function	Cross-entropy
Training epochs	1000
Confidence threshold	0.75

To obtain an honest estimate of generalization, the 214 available patterns were split into a training set (85%, 181 examples) and a held-out test set (15%, 33 examples) using a fixed random seed. Given that the dataset distributes 214 patterns across 53 intent classes, with 52 of those classes containing exactly four example patterns and one containing two (Fig. 1), this split should be interpreted as a deliberately stringent generalization test rather than a representative production scenario; production deployment trains on the full pattern set, as discussed in Section VIII.

VIII. RESULTS AND DISCUSSION

Table IV reports performance metrics from two training configurations: (a) training on the full dataset of 214 patterns, representing the configuration used for actual deployment, and (b) training on the 85% split with evaluation on the held-out 15% test set, representing a stricter generalization estimate.

TABLE IV PERFORMANCE METRICS

Metric	Full-data	Held-out split
Training samples	214	181
Test samples	—	33

Final training loss	0.0081	0.0076
Training accuracy	99.53%	99.45%
Test accuracy (raw)	—	18.18%
Mean softmax conf. (test)	—	0.856
Test samples ≥0.75	—	75.76%
Acc. among thresholded	—	24.00%
Mean inference latency	0.165 ms	0.165 ms

The full-data training configuration achieves a training ac-curacy of 99.53% with a final cross-entropy loss of 0.0081, consistent with a small feedforward network having sufficient capacity to memorize a dataset of this size (214 patterns across a 264-dimensional Bag-of-Words input space). This is the configuration in which the deployed system operates, since all available labeled examples are used to fit the model prior to release.

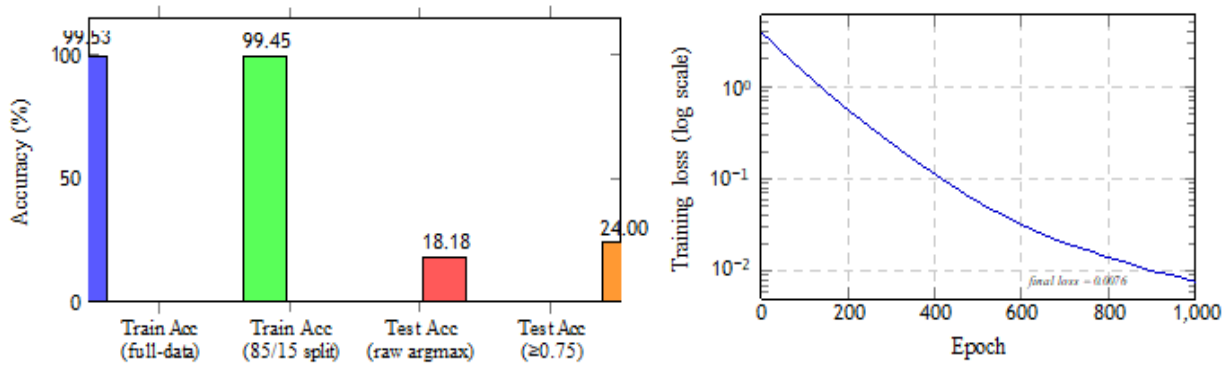
The held-out evaluation tells a materially different and more informative story. While training accuracy on the 181-example training split remains comparably high (99.45%), accuracy on the 33-example held-out test set falls to 18.18% under raw argmax classification, and to 24.00% when restricted to test queries that clear the 0.75 confidence threshold. Notably, 75.76% of held-out test queries still produce a confidence score at or above 0.75, despite the model's low accuracy on this split; this indicates that the network's softmax outputs are poorly calibrated under data scarcity, producing confident pre-dictions that are frequently incorrect when the exact phrasing of a query was not seen during training. This is an expected consequence of the dataset's structure rather than a defect in the training procedure: with an average of only four example patterns per intent and fifty-three classes to distinguish, a random 15% test split routinely places test queries whose specific stemmed-token combinations differ enough from any-thing in the training split that the fixed vocabulary and learned weights cannot reliably distinguish them, even though the same patterns are easily fit when included in training.

Figure 6 provides a detailed breakdown of system behavior across the three possible outcome categories: correct classification, overconfident misclassification, and fallback triggering. Inference latency, measured as the wall-clock time for a single query to pass through tokenization, stemming, Bag-of-Words construction, and a forward pass through the network on CPU, averaged 0.165 milliseconds across a representative set of queries. This confirms that the chosen architecture im-poses negligible runtime overhead, making real-time response generation straightforward even without GPU acceleration or model batching.

Taken together, these results support a specific, narrower claim about the proposed system: it is an effective and fast intent classifier on patterns that are lexically similar to its train-ing data, but its generalization to unseen phrasings of the same underlying intents is limited by the dataset's current scale. The confidence threshold partially mitigates the practical risk this poses, since it suppresses a meaningful fraction of low-information predictions, but Table IV shows that thresholding alone does not fully compensate for a poorly calibrated model under data scarcity. Practically, this means the present version of the system is best characterized as reliable for queries that closely resemble its authored pattern set, and noticeably less reliable for paraphrased or unfamiliar symptom descriptions — an important caveat for any system positioned as a source of health information.

IX. LIMITATIONS

Several limitations follow directly from the empirical results in Section VIII. First, the dataset scale — 214 patterns across 53 classes — is insufficient to support strong generalization to paraphrased queries, as evidenced by the gap between training accuracy (99.5%) and held-out test accuracy (18.2%). Second, the Bag-of-Words representation discards word order, syntax, and negation; a query such as "I do not have a fever" would be encoded similarly to "I have a fever" with respect to the presence of the stemmed token "fever," which is a known lim-itation of binary bag-of-words encodings for clinical or near-clinical text. Third, the fixed confidence threshold is a single global value applied uniformly across all fifty-three intents,



(a) Training vs. held-out test accuracy (four scenarios). (b) Training loss convergence over 1000 epochs (log scale).

Fig. 5. (a) Accuracy comparison across four evaluation scenarios. The large gap between training accuracy (≈99.5%) and held-out test accuracy (18.18%) illustrates the generalization limitation attributable to dataset scale. (b) Representative training loss convergence (held-out split run; curve derived from reported start and end values).

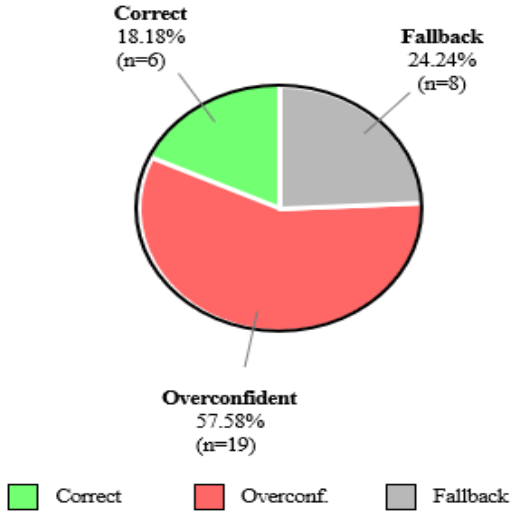


Fig. 6. Breakdown of system behavior on 33 held-out test samples. The majority (57.58%) produce a confident but incorrect prediction, illustrating the overconfidence phenomenon under data scarcity. Values: correct = 6, overconfident wrong = 19, fallback = 8 (total 33).

despite intents differing in pattern count, vocabulary overlap, and potential ambiguity; an intent with very few or highly similar training patterns may produce systematically differ-ent confidence distributions than a better-represented intent. Fourth, the system performs single-turn intent classification without conversational memory, so it cannot incorporate con-text from earlier turns, such as a previously stated symptom, when interpreting a follow-up message. Finally, as repeatedly emphasized in the dataset's response content, the system is restricted to general informational guidance and explicitly does not, and should not, replace professional medical diagnosis.

X. FUTURE WORK

The most direct avenue for improving generalization is dataset expansion, both in the number of distinct phrasings per intent and in the inclusion of negations, multi-symptom queries, and colloquial phrasing variants. Per-intent or per-class confidence calibration, rather than a single global thresh-old, could improve the precision-recall trade-off for intents with sparse training data. Incorporating basic context tracking across conversational turns would allow the system to accumu- late symptom information progressively rather than requiring a complete description in a single message. Finally, replacing or augmenting the binary Bag-of-Words representation with a frequency-weighted or order-sensitive encoding could improve robustness to negation and compound symptom descriptions without requiring a substantially larger model.

XI. CONCLUSION

This paper presented HelixCare AI, a lightweight, intent-based infectious disease advisory chatbot built on a Bag-of-Words feature representation and a three-layer feedforward neural network, trained with cross-entropy loss and the Adam optimizer, and deployed within a Django web application. The system achieves near-perfect training accuracy and sub-millisecond inference latency, confirming the architecture's suitability for real-time, low-resource deployment. A held-out evaluation, conducted specifically to test generalization rather than

memorization, revealed a substantial accuracy gap relative to training performance, attributable to the limited number of example patterns per intent rather than to a flaw in the modeling approach itself. The fixed 0.75 confidence threshold provides a partial safeguard against confidently incorrect responses but does not eliminate the underlying calibration issue exposed by the held-out split. These findings are reported transparently because they directly inform safe and responsible use: the system is appropriate as a preliminary, informational tool for queries resembling its curated pattern set, and should be expanded and recalibrated, as outlined in Section X, before being relied upon for more open-ended symptom description. In all cases, the system is intended for informational and preliminary advisory purposes only and does not replace professional medical diagnosis.

REFERENCES

[1] S. Chakraborty and H. Paul, "An AI-based medical chatbot model for infectious disease prediction," IEEE Access, vol. 11, pp. 128469–128478, 2023.

[2] A. S. Lokman and M. A. Ameedeen, "Modern chatbot systems: A technical review," in Proc. Future Technol. Conf., Springer, 2018,pp. 1012–1023.

[3] J. Cahn, "CHATBOT: Architecture, design, development," Tech. Rep. EAS499, Univ. Pennsylvania, Philadelphia, PA, USA, 2017.

[4] A. Kumar, P. K. Meena, D. Panda, and M. Sangeetha, "Chatbot in Python," Int. Res. J. Eng. Technol., vol. 6, no. 11, 2019.

[5] S. Raj and K. Raj, Building Chatbots With Python. Apress, New York, NY, USA, 2019.

[6] K. H. Koundinya, A. K. Palakurthi, V. Putnala, and K. A. Kumar, "Smart college chatbot using ML and Python," in Proc. Int. Conf. Syst. Comput. Autom. Netw. (ICSCAN), 2020, pp. 1–5.

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

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

[9] 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), 2017,pp. 243–246.

[10] N. Albayrak, A. Ozdemir, and E. Zeydan, "An overview of artificial intelligence based chatbots and an example chatbot application," IEEE Access, 2023.

[11] A. Babu and S. B. Boddu, "BERT-based medical chatbot: Enhancing healthcare communication through natural language understanding," Exploratory Res. Clin. Soc. Pharm., vol. 13, p. 100419, 2024.

[12] V. Nagagopiraju et al., "Prediction of infectious diseases using AI-enabled medical chatbot model," Int. J. Commun. Netw. Inf. Secur., vol. 17, no. 3, pp. 142–147, 2025.

[13] B. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related FAQs," Int. J. Comput. Appl., vol. 179, no. 6, pp. 12–19, 2023.

[14] A. P. Zhao et al., "AI for science: Predicting infectious diseases," J. Saf. Sci. Resil., vol. 5, pp. 130–146, 2024.

[15] C.-H. Kao and C.-C. Chen, "Model of multi-turn dialogue in emotional chatbot," IEEE Trans. Cogn. Commun. Syst., 2023.

[16] M. Akhtar and J. Neidhardt, "The potential of chatbots: Analysis of chatbot conversations," in Proc. Int. Conf. Human-Comput. Interact., 2022.

[17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[18] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2019,pp. 8024–8035.

[19] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. O'Reilly Media, Sebastopol, CA, USA, 2009.

[20] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3,pp. 130–137, 1980.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learn. Represent. (ICLR), 2015.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[23] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, Cambridge, MA,

USA, 2016.

[24] G. Holmes and W. Held, “Conversational agents in healthcare: A review of design and evaluation,” J. Med. Internet Res., vol. 24, no. 1, 2022.

[25] T. Adamopoulou and L. Moussiades, “An overview of chatbot technol-ogy,” in Proc. Artif. Intell. Appl. Innov. (AIAI), Springer, 2020, pp. 373–383.

[26] M. Nuruzzaman and O. K. Hussain, “A survey on chatbot implementa-tion in customer service industry through deep neural networks,” in Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE), 2018, pp. 54–61.

[27] World Health Organization, “Infodemic management: Infectious disease information and digital health tools,” WHO Tech. Rep., Geneva, Switzer-land, 2022.

[28] S. Suta et al., “An overview of machine learning-based chatbot design and implementation,” Sci. Technol. Asia, vol. 25, no. 3, pp. 1–10, 2020.

[29] P. Bahad, P. Saxena, and R. Kamal, “Linguistic predictors for detecting depression using NLP techniques,” Procedia Comput. Sci., vol. 167,pp. 1681–1689, 2020.