HealthSense: A Deep Learning based Symptom Analyzer

Shashank Mishra
M24MAC011
Department of Mathematics
Indian Institute of Technology Jodhpur
m24mac011@iitj.ac.in

Md Moin Munir Ahmed
M24MAC004
Department of Mathematics
Indian Institute of Technology Jodhpur
m24mac004@iitj.ac.in

Souptik Dutta
M24MAC014
Department of Mathematics
Indian Institute of Technology Jodhpur
m24mac014@iitj.ac.in

Abstract

Establishing reliable diagnoses of human disease from symptoms reported by patients remains a persistent problem in healthcare due to variability in the expression of symptoms and subjective measurement of their intensity. Traditional rule-based systems and simple machine learning methods are susceptible to mismodelling the temporal behavior and clinical relevance of symptoms. The HealthSense project presents a new LSTM-based disease prediction model that explicitly involves symptom severity in the learning process. With a weighted embedding layer, the model prioritizes clinically relevant symptoms, leading to more accurate and contextually informed predictions. It is also robust to heterogeneous symptom sequences and missing data, more representative of real-world clinical cases. On testing with a real dataset, the model achieves a diagnostic accuracy of 96.45%, surpassing baseline machine learning methods. The method demonstrates the potential of combining deep learning with expert knowledge to develop more reliable and interpretable AI-based diagnostic systems.

1 Introduction

Accurate diagnosis of various human diseases from patient-reported symptoms remains to be a very challenging issue in the healthcare domain, primarily because of the variability in the presentation of the symptoms and the subjective nature of their severity. Tradionally, disease diagnostic systems have been reliant on static, rule-based approaches that fail to capture the temporal nature of the evolution of symptoms and its relative clinical importance. Machine learning studies have also been carried out but are mostly practically inefficient. Recent deep learning advances, specifically employing Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM) Networks have tremendous potential for tackling these challenges. This is because these are more efficient in handling dependencies of sequential data. However, most of the methods used till date haven't explicitly used the severity of symptoms, which could significantly influence proper diagnosis.

This project, HealthSense, tries to overcome these hurdles by constructing and LSTM-based disease prediction model, which incorporates the symptom severity weights in its learning. We introduce a weighted embedding layer that gives symptom representations their clinical importance weights, allowing the model to better focus on high-impact symptoms. The system dynamically processes

Preprint. Under review.

sequences of symptoms, providing flexibility in symptom order while being resilient to missing inputs. Our tests show that the method attains very high diagnostic accuracy in a real-world dataset, with better performance than traditional machine learning based methods. By bringing together domain knowledge and deep learning, this work contributes to more reliable and explainable AI-based diagnostic aids. Additionally, some precautions, possible medications and dietary measures have been suggested as well.

2 Literature Review

There have been very little works on real world datasets in the medical domain that can effectively work towards prediction of diseases. Furthermore, taking appropriate importance of symptom severity has been deemed important, but little progress made in that front. Traditional ML algorithms have been used till date, but it is the deep learning based algorithms and its effeciency that make the difference.

1. To address self-medication issues, a machine learning-based drug recommendation system is suggested by Avanthi, S.K. et. al, which predicts drugs based on user-inputted diseases or symptoms. Precision, recall, and F1 score was employed to evaluate the performance of the models.
2. This study by Meshram, et. al.,[1] proposes a general disease prediction system using symptoms and patient habits, applying K-Nearest Neighbor (KNN) and Convolutional Neural Network (CNN) algorithms. CNN achieved 84.5% accuracy, outperforming KNN in both accuracy, time, and memory efficiency.
3. This research conducted by Shahadat, et. al., examines 48 comparative studies of supervised machine learning algorithms employed in disease prediction. The most frequently utilized was the Support Vector Machine (SVM), and the most accurate was the Random Forest (RF) in 53% of applications.
4. This research by Shriya, et. al., [2]applies various classification algorithms to forecast heart disease, breast cancer, and diabetes using UCI datasets. Backward modeling with p-value testing was applied for selecting features. The findings indicate the efficiency of machine learning in disease early detection, resulting in a significant rise in patient survival rates through early diagnosis.

3 Problem Statement

The primary challenges that are normally faced in symptom-based disease prediction are:

1. **Symptom variability:** Patients generally report their symptoms in various combinations, and their ordering remains inconsistent.
2. **Severity Neglect:** Existing models fail to correctly account for the importance of different symptoms and the context in which they are presented.
3. **Limitations in data:** Sparse labeled datasets are often the reason for constraints in model training.
4. **Sequence modeling:** Traditional approaches struggle with temporal patterns in symptoms

Pure rule-based and/or data-driven approaches lack proper medical interpretability. Our proposed solution tries to bridge the gap between the enormous capabilities of deep learning based methods and medical expertise. This would enhance the diagnostic decision support.

4 Proposed Methodology

This section discusses about the approaches taken by leveraging the advances of deep learning models for accurate disease prediction. At the very beginning, a thorough data preprocessing was performed on our dataset. Followed by that, various DL models were tried and tested on the preprocessed dataset for training, validation and testing purposes. The following subsections contain detailed discussions about each and every step involved in this study.

4.1 Dataset Acquisition

The "Medicine Recommendation System dataset" at our perusal has been taken from Kaggle. This is a well-balanced dataset containing the following files:

- **Description:** Contains about 41 different kinds of diseases along with a short description of the disease
- **Symptoms:** Contains various kinds of combinations for a given disease
- **Symptom-Severity:** Contains about 140 different kinds of symptoms of diseases along with a severity measure
- **Precaution:** Contains various kinds of precautionary advices for a given disease
- **Medication:** Contains various kinds of possible medication that can be prescribed for a given disease
- **Diet:** Contains advisory diet to get rid of or protect oneself from a particular disease
- **Workout:** Contains advisory workout measures to keep safe from a particular disease

4.2 Data cleaning & Preprocessing

The most important step after collecting the data is data preprocessing. A prediction model would produce desired results when it would be supplied with clean and preprocessed data. Since we are dealing with textual data, hence standard natural language processing techniques were utilised, which are listed as follows:

4.2.1 Null value handling

Practical, real-world data generally contains a lot of null values, which needs to be handled efficiently. This generally happens due to failure in recording data properly or corruption within the data. **In our case, there were no cases of null values, hence this step was not required.**

4.2.2 Tokenization

The process of dividing an entire raw text sample into parts is called tokenization and the small parts or chunks produced are called tokens. As a result, an entire text is tokenized into sentences and further into words. In this case, the following steps were implemented:

- The comma-separated symptoms were first converted to lower case
- The whitespaces trimmed and replaced them with underscores for better representation
- A vocabulary was built and each of the symptom tokens were converted into some equivalent integer IDs

Hence, along with each disease, we had the corresponding possible combinations of symptoms as numerical token values.

4.2.3 Shuffling the rows

All the symptoms and their corresponding list of symptom token IDs were shuffled so that the mapping between the symptom and tokens remain consistent.

4.2.4 Pad Sequences

Considering the fact that the input of a patient about their symptoms won't be consistent, i.e., some may show less number of symptoms while others may experience more of them. The dataset reflects similar situations as well where all the rows aren't consistent in terms of number of symptoms. To tackle this, pad sequences were applied which ensures that the lengthiest symptom sequence was taken as the maximum, and all other shorter sequences were padded with zeros at the end.

4.2.5 Encoding Disease Labels

The disease label here is a categorical feature and was encoded with OneHot Encoding, resulting in one-hot vectors. This ensures the algorithm would learn bias-free and enhances the model interpretability.

These were the few primary steps for data preprocessing implemented, which helped to obtain a clean and proper numerical representation of the dataset. Even then, some more advanced methods were utilized to help our models learn in a better way. The following subsection discusses those in detail.

4.3 Embedding and Weighting the Tokens

Although tokenized values of the symptoms would suffice for the models to learn and interpret about the diseases, it is important that they are initialized with weights as embeddings to properly capture the essence of the data. This is demonstrated with the help of an example below.

1. Consider the symptoms as a raw textual input from a patient:

   "nausea, spinning movements, vomiting, headache"

2. As retrieved from the dataset, they have the following weights or symptom-severity:

   [5, 6, 5, 3]

3. **Tokenization mapping**:

   $$\text{nausea} \rightarrow 1$$
   $$\text{spinning movements} \rightarrow 2$$
   $$\text{vomiting} \rightarrow 3$$
   $$\text{headache} \rightarrow 4$$

4. **Token Embedding:** Assume that a **3-dimensional embedding vector** is defined and initialized for each of the four symptoms, which would look like this:

$$\begin{aligned} E(1) &= [0.1,\ 0.2,\ 0.3] && \text{(nausea)} \\ E(2) &= [0.4,\ 0.5,\ 0.6] && \text{(spinning\_movements)} \\ E(3) &= [0.7,\ 0.8,\ 0.9] && \text{(vomiting)} \\ E(4) &= [1.0,\ 1.1,\ 1.2] && \text{(headache)} \end{aligned}$$

   Thus, the embedding matrix, X would look like this:

$$X = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \\ 1.0 & 1.1 & 1.2 \end{bmatrix}$$

5. Next, we multiply the embedding matrix with the corresponding weights from above:
   - For "nausea" (weight 5):
   $$5 \times [0.1, 0.2, 0.3] = [0.5, 1.0, 1.5]$$
   - For "spinning_movements" (weight 6):
   $$6 \times [0.4, 0.5, 0.6] = [2.4, 3.0, 3.6]$$
   - For "vomiting" (weight 5):
   $$5 \times [0.7, 0.8, 0.9] = [3.5, 4.0, 4.5]$$
   - For "headache" (weight 3):
   $$3 \times [1.0, 1.1, 1.2] = [3.0, 3.3, 3.6]$$

   Hence, the resulting weighted embedding matrix, X' would look like this:

$$X = \begin{bmatrix} 0.5 & 1.0 & 1.5 \\ 2.4 & 3.0 & 3.6 \\ 3.5 & 4.0 & 4.5 \\ 3.0 & 3.3 & 3.6 \end{bmatrix}$$

The idea shown in the above example was actually implemented in large vector spaces with large embedding dimensions. While training on models such as LSTMs, each of the rows from the resulting embedding matrix would be used as the input embedding vectors. This weighted embedding approach ensures that the symptoms that have greater clinical importance or significance will tend to influence the prediction more. At the same time, it also preserves the semantic relationships between the symptoms.

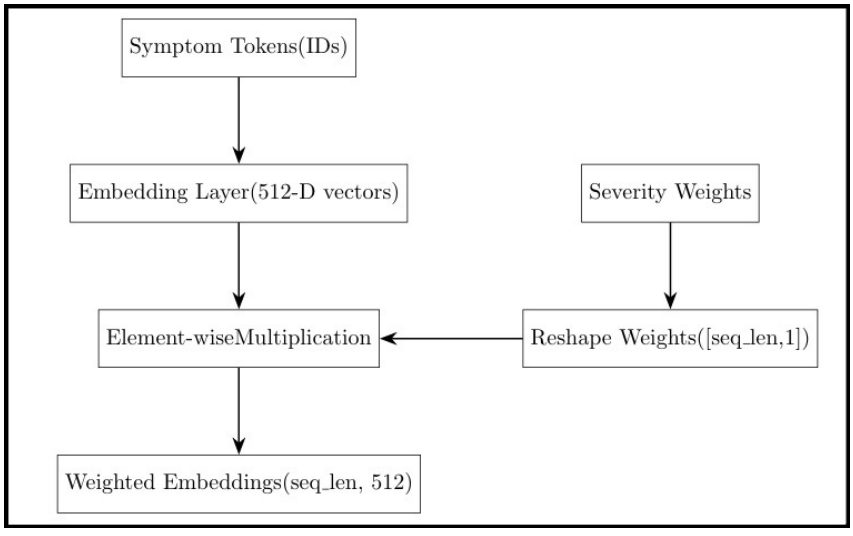

Figure 1: Embedding Workflow

4.4 Model Training

Once the data was fully preprocessed and prepared, we proceeded to the modeling stage. Firstly, the entire dataset consisting of symptom sequences as embedding vectors and the corresponding disease labels were split into training and testing sets. The training data consisted of 80% of the entire data, while 20% of it were kept for testing the performance of our DL models. Furthermore, 10% of the training data was used as a validation set for validating the model performance against the training performance.

We have tried out 3 model architectures for our study, listed below:

- **A simple LSTM model using standard parameters and hyperparameters**
- **An LSTM employing multi-head attention layers**
- **A hybrid approach with (LSTM + Attention) model to extract features and then trained on a Support Vector Machine (SVM) classifier with the extracted features**

An in-depth working and analysis of the models are discussed in the following parts.

4.4.1 LSTM based model

A typical Long Short Term Memory (LSTM) works in a gated structure, which makes it more efficient in handling sequential data compared to other architectures such as RNNs. It has the ability to capture what is important and what is not relevant. The memory cell of an LSTM architecture consists of three gates, specialised in regulating the flow of information. In this case, the sequential symptom data is processed through these gates. These are:

1. **Forget gate (F_t):**
    - Decides what information to keep and what to discard from the previous state
    - Computed as: $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$ and $f_t \odot C_{t-1}$
    - Uses sigmoid activation to determine update proportions (0-1)
2. **Input gate (I_t):**
    - Controls how new symptom information updates the cell state
    - Enables the model to "reset" when encountering critical symptoms
    - Computed as: $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$
    - Contains a candidate memory having information of the current time step, which is calculated as: $\bar{C}_t = tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$
    - The information from $\bar{C}_t$ is filtered by $f_t \oplus c_{t-1}$
    - The cell state is calculated as: $C_t = [f_t \odot C_{t-1}] \oplus [i_t \odot \bar{C}_t]$
3. **Output gate (O_t):**
    - Determines what information should be output to the next layer
    - Filters relevant diagnostic information for prediction
    - Expressed as: $O_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$
    - Hidden state is computed as: $h_t = O_t \odot tanh(C_t)$

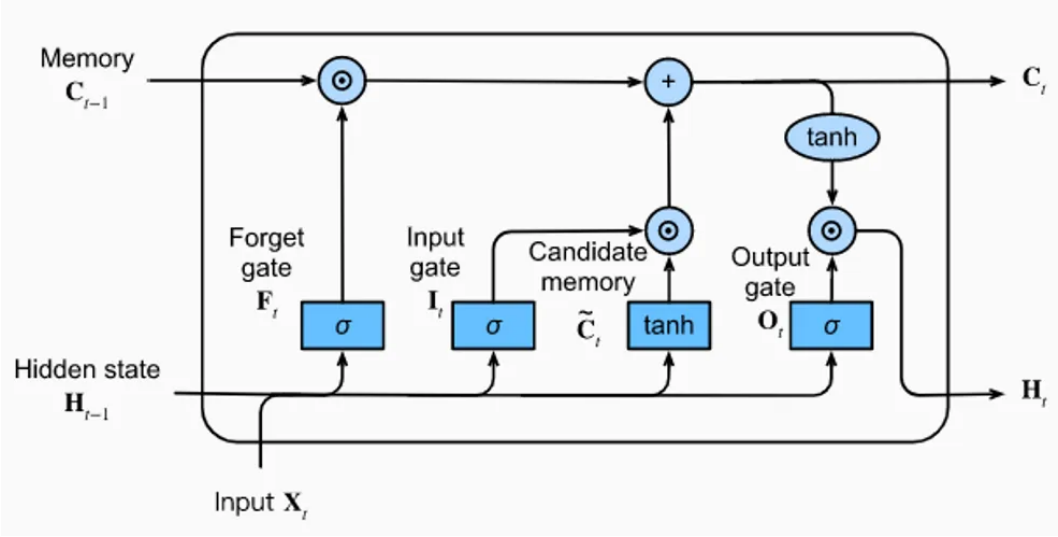

Figure 2: Architecture of an LSTM unit

The specifics of the implementation:

- Embedding dimensions: 512
- Recurrent Dropout of 0.2 applied for generalization of model and prevent overfitting
- Adam optimizer used for model optimisation
- Trained over 100 epochs with categorical cross entropy as loss function and batch size 32

4.4.2 LSTM model with Multi-head attention

Attention mechanism has been a milestone of revolutionary progress in the realm of deep learning, especially for sequence modeling tasks and transduction problems. **An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.** The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.[3]

In the case of a single scaled dot-product attention, we have keys and queries of dimension d_k. The attention value of a query from the set of queries is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

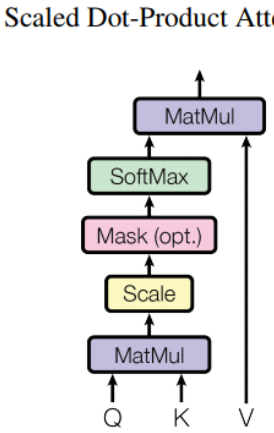

Figure 3: Representation of scaled dot-product attention

It is more beneficial to use parallely use more than one attention head. In case of multi-head attention, it helps the model to be more attentive towards information from different representations. This is calculated as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

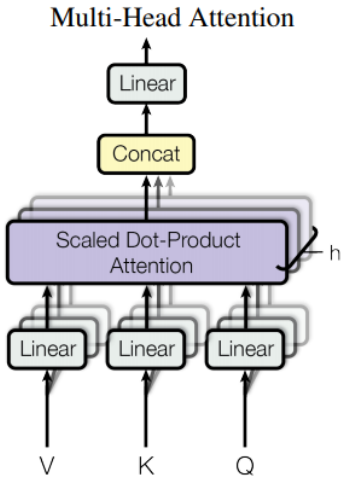

Figure 4: Representation of Multi-head attention

Our model was implemented using two multi-head attention layers:

1. **First Multi-Head Attention Block:**
   - 8 heads with $d_k = 64$
   - Layer normalisation with $\epsilon = 10^{-6}$ applied

2. **Second Multi-Head Attention Block**

    - 8 heads with $d_k = 64$
    - Layer normalisation with $\epsilon = 10^{-6}$ applied
    - Sequence of features are aggregated into a single vector using global average pooling
    - Dropout of 0.5
    - ReLU activation for the Dense layer and softmax activation for the output layer
    - Adam optimizer used for model optimisation
    - Trained over 100 epochs with categorical cross entropy as loss function and batch size 32

The entire workflow of this model can be depicted as below:

1. The LSTM output sequence (each time step is a vector) goes into the attention layer
2. Each time step is scored using learned weights and biases
3. Softmax is applied to obtain normalized attention weights
4. Each LSTM vector is multiplied by its attention weight, and the sum over time steps produces a context vector

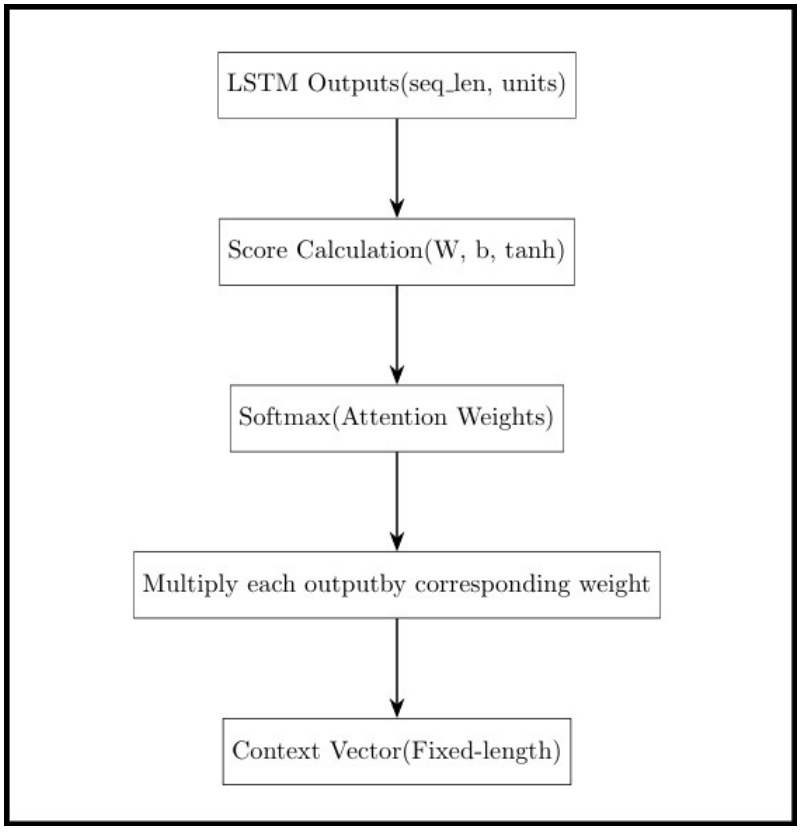

Figure 5: Workflow in attention layer

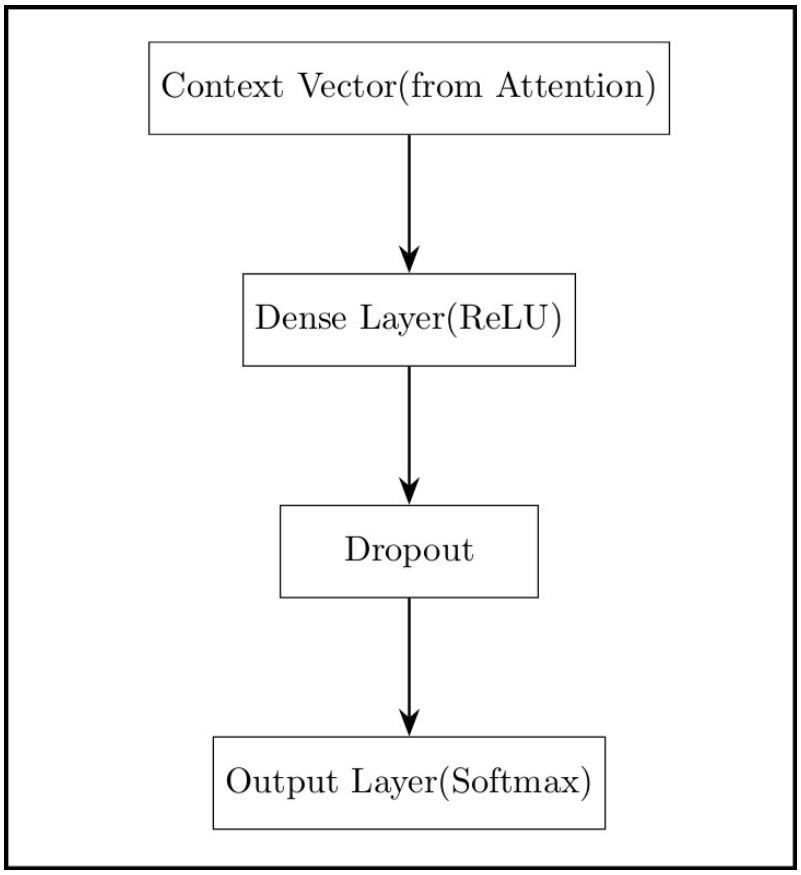

Figure 6: Workflow in dense layer

4.4.3 LSTM + SVM Hybrid Model

This document explains in detail a hybrid model combining deep feature extraction using an LSTM with an attention mechanism, followed by a Support Vector Machine (SVM) classifier for disease prediction based on patient symptom data.

The hybrid model consists of two main parts:

1. **Deep Feature Extraction (LSTM + Attention):**
   - Process symptom sequences using an embedding layer, an LSTM network, and an attention mechanism
   - The output is a fixed-length feature vector summarizing the clinical information that are important
2. **Classification using SVM:**
   - The extracted feature vector serves as input for a Support Vector Machine (SVM) classifier
   - The SVM uses the kernel trick to learn a decision boundary that separates the disease classes

For feature extraction, first the attention mechanism was applied as before, to produce the attention values and normalised weights. Further, the context vector was applied as:

$$v = \sum_{t=1}^{T} \alpha_t h_t$$

The context vector is further transformed via a dense layer:

$$f = ReLU(W_d v + b_d)$$

where $f \in \mathbb{R}^n$ is the extracted feature vector

The feature vector f is used as input for the SVM classifier. In its simplest form, an SVM finds a hyperplane that maximizes the margin between class. It is defined by:

$$w^\top f + b = 0$$

For multi-class problems, techniques such as one-vs-rest are used. When using a non-linear kernel (e.g., RBF kernel), the kernel function is defined as:

$$K(f_i, f_j) = \exp\left(-\gamma \|f_i - f_j\|^2\right)$$

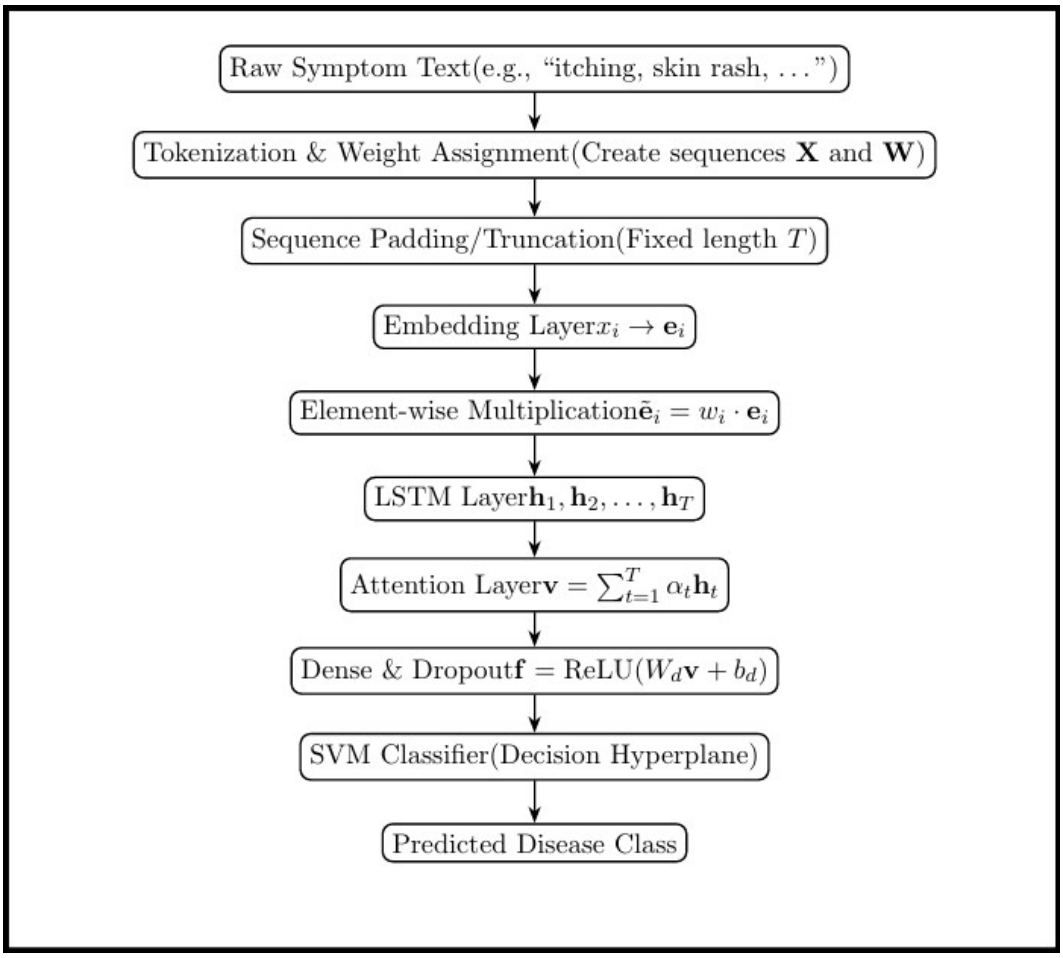

Figure 7: Worflow of Hybrid model

5 Experiments and Results

This section discusses about the various results we have got after training our models. The models were evaluated based on their acuracies they produce. The train_loss, train_accuracy, validation_loss, validation_accuracy, test_loss and test_accuracy are depicted in tabular form below.

Model	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
LSTM	0.0477	98.05%	0.2050	96.19%
LSTM + Attention	0.0882	96.42%	0.1918	95.18%
Hybrid model	0.0542	97.77%	0.1457	96.45%

Table 1: Model Performance Table

Model	Test Loss	Test Accuracy
LSTM	0.2481	94.72%
LSTM + Attention	0.2800	93.09%
Hybrid model	0.1619	96.54%

Table 2: Model Performance Table

The accuracy and loss curves of the results of all the models are shown as follows:

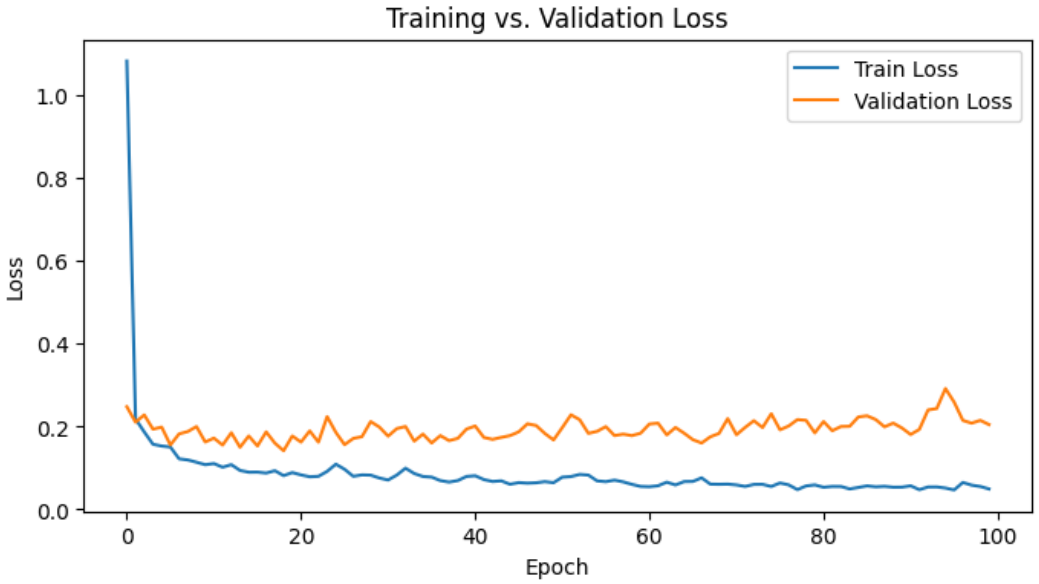

Figure 8: Loss curve of LSTM model

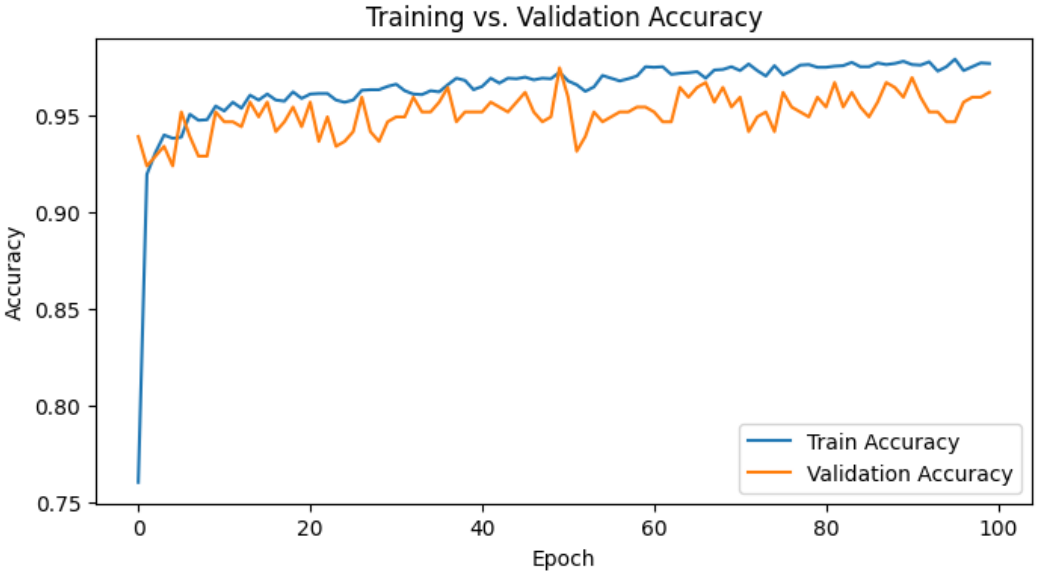

Figure 9: Accuracy curve of LSTM model

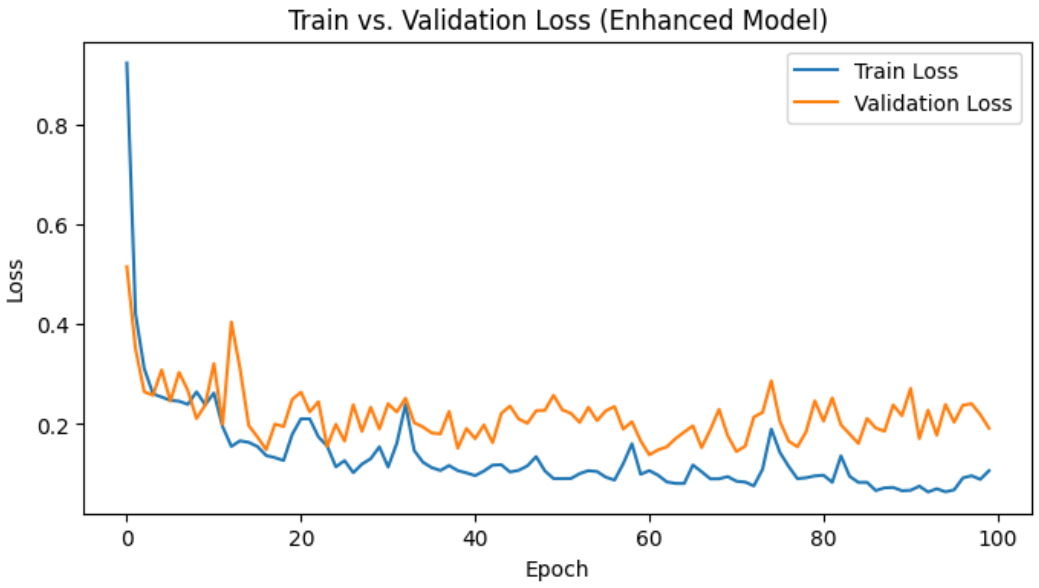

Figure 10: Loss curve of LSTM + Attention model

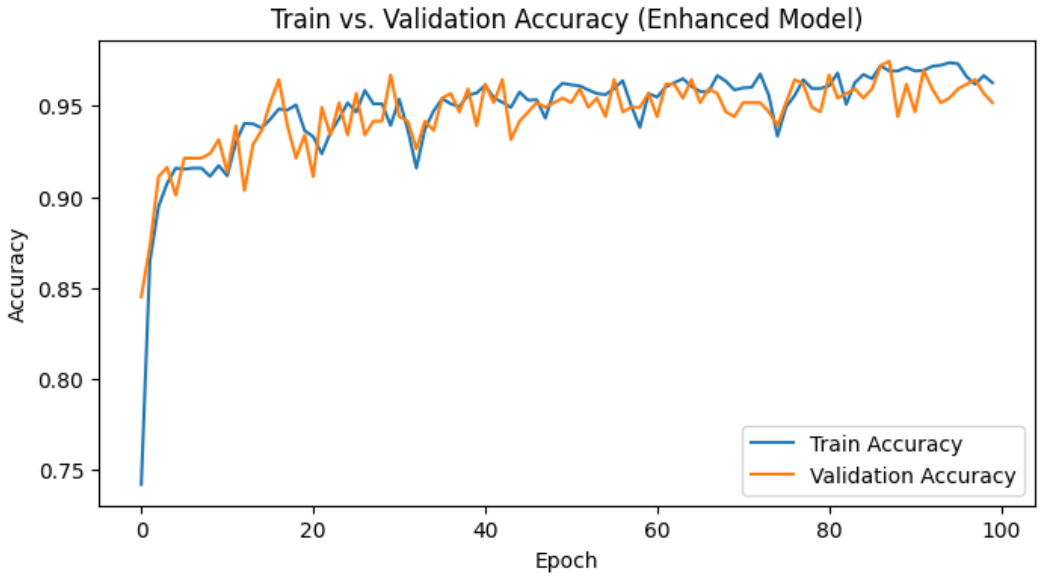

Figure 11: Accuracy curve of LSTM + Attention model

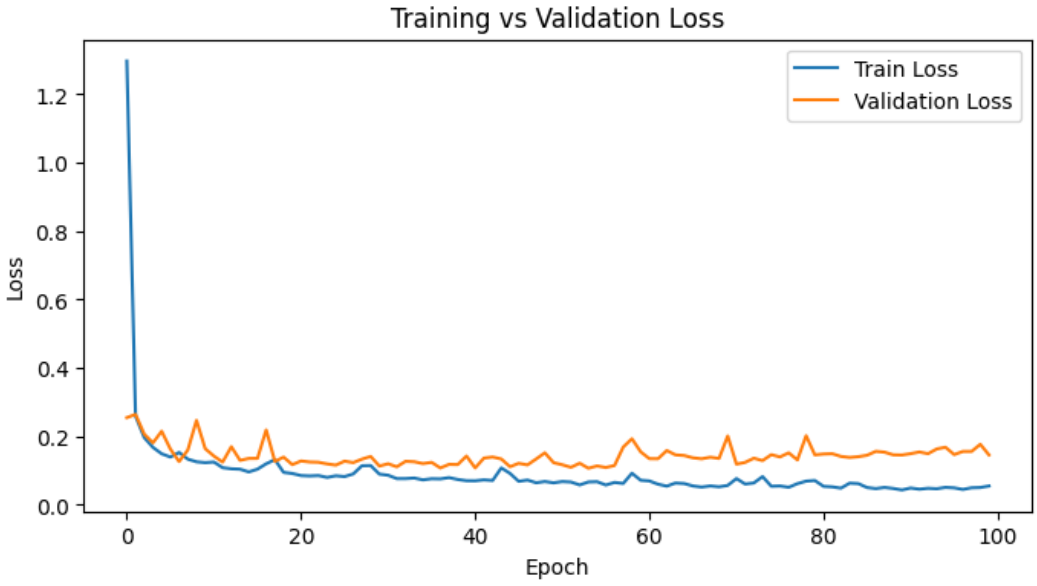

Figure 12: Loss curve of Hybrid model

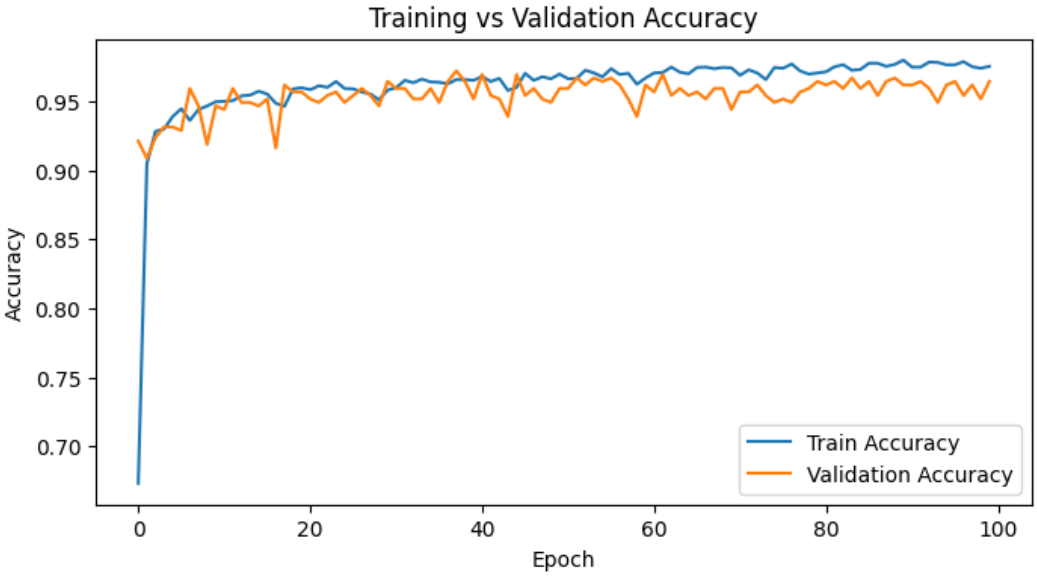

Figure 13: Accuracy curve of Hybrid model

6 Insights and Observations

6.1 Data & Feature Engineering

1. **Weighted Embeddings:** Incorporating symptom severity as weights is an innovative idea. By multiplying the standard embeddings with a severity weight, the model is forced to give more prominence to severe symptoms—a potentially powerful way to incorporate expert domain knowledge.
2. **Token Shuffling and Ordering:** The code experiments with shuffling the symptom tokens within each record. Although the intent is to explore if altering the order (e.g., putting high-severity symptoms in specific positions) might help the LSTM "remember" important information better, it is also a data augmentation technique that could reduce bias from a fixed token order.

6.2 Model Selection and Trade-offs

1. **Sequential Processing with LSTMs:** LSTMs are well-suited for sequential data where order might matter. However, their capacity to capture long-term dependencies can sometimes be limited when sequences are long or the order is not inherently meaningful.
2. **Attention Mechanisms:** The use of Multi-Head Attention layers (inspired by Transformer models) provides a mechanism for the model to learn which parts of the input are most important regardless of their position in the sequence. This can be especially useful if the order of symptoms is arbitrary or if certain symptoms carry a disproportionate amount of information.

7 Model Deployment

The best performing model was deployed to tackle real world scenarios and the results are quite fascinating. In addition to prediction of diseases based on the symptoms, we have also provided some more information beneficial for patients. These are:

- A short description of the disease based on the symptoms reported
- Necessary precautionary measures one could take to avoid the disease
- Probable medications (although it is advised to consult a doctor before consumption of any kind of medication)
- Diet and exercise that can help a patient recover and stay safe from such diseases

The link to the web-app is: https://healthsense-dl.streamlit.app/ The following images show a demo of the UI.

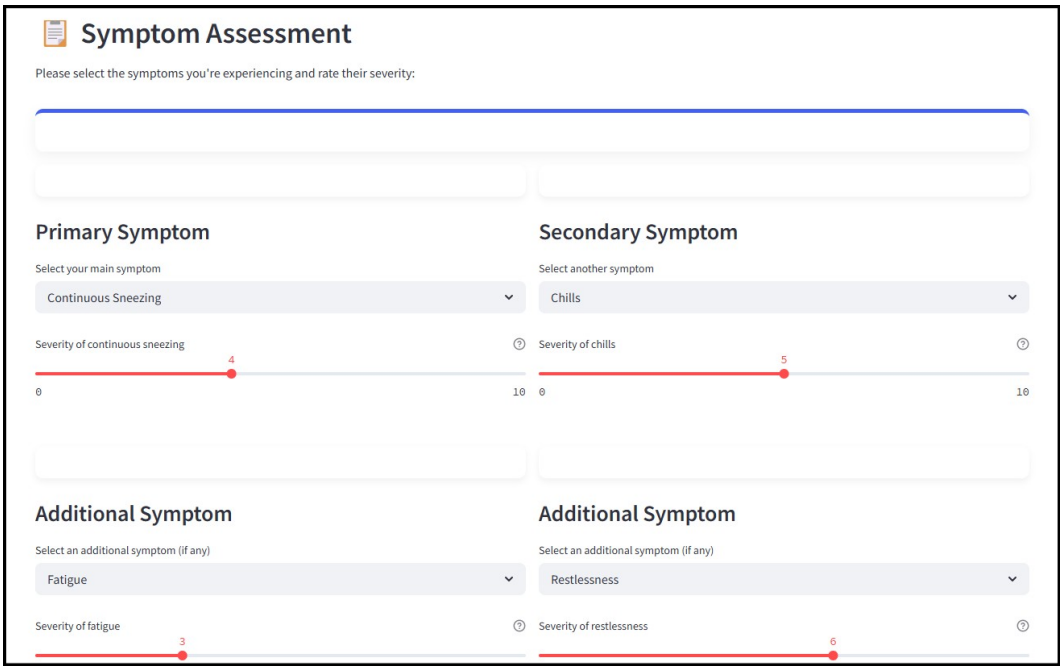

Figure 14: Input of Symptoms

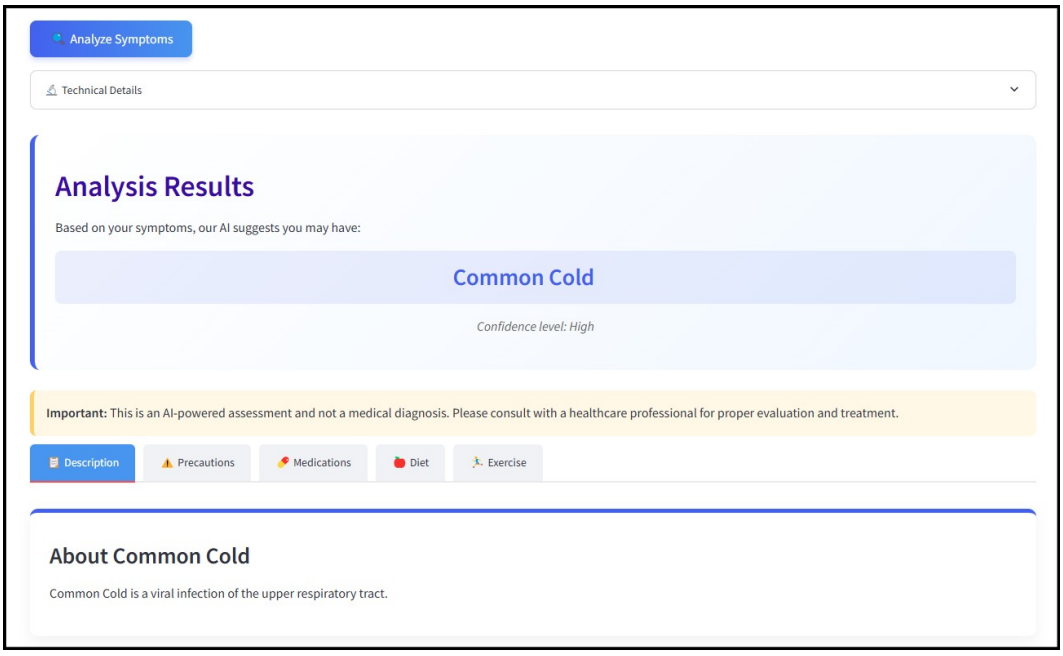

Figure 15: Predictions of the model

8 Conclusions

1. The basic LSTM model uses a weighted embedding mechanism to give more importance to severe symptoms
2. An enhanced model with attention layers improves upon this by allowing the network to selectively focus on parts of the input sequence that are more important, even if they appear early or late in the sequence
3. The LSTM + SVM hybrid model extracts deep features using a custom attention mechanism and then leverages a traditional SVM classifier, which might offer robustness in certain situations or with limited data

Each approach has its own strengths and trade-offs, and the choice of model could ultimately depend on the size and nature of the dataset, computational resources, and the specific performance metrics that are most important for the application. The code also includes careful preprocessing and thoughtful incorporation of domain-specific knowledge (symptom severity), both of which are crucial in building high-performance predictive models.

Overall, the code demonstrates an advanced and multifaceted approach to combining natural language processing, sequence modeling, and hybrid classification methods in a real-world domain.

References

[1] Dhiraj Dahiwade, Gajanan Patle, and Ektaa Meshram. "Designing Disease Prediction Model Using Machine Learning Approach". In: *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. 2019, pp. 1211–1215. DOI: 10.1109/ICCMC.2019.8819782.

[2] Pahulpreet Singh Kohli and Shriya Arora. "Application of Machine Learning in Disease Prediction". In: *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. 2018, pp. 1–4. DOI: 10.1109/CCAA.2018.8777449.

[3] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.