Int. J. Simul. Multidisci. Des. Optim. **11**, 25 (2020)

https://doi.org/10.1051/smdo/2020019

IJSMDO
Available online at:
https://www.ijsmdo.org

RESEARCH ARTICLE OPEN ACCESS

LSTM based Ensemble Network to enhance the learning of long-term dependencies in chatbot

Shruti Patil, Venkatesh M. Mudaliar, Pooja Kamat*, and Shilpa Gite

Symbiosis Institute of Technology, Symbiosis International (Deemed) University, Pune, Maharashtra, India

Received: 30 July 2019 / Accepted: 10 November 2020

Abstract. A chatbot is a software that can reproduce a discussion portraying a specific dimension of articulation among people and machines utilizing Natural Human Language. With the advent of AI, chatbots have developed from being minor guideline-based models to progressively modern models. A striking highlight of the current chatbot frameworks is their capacity to maintain and support explicit highlights and settings of the discussions empowering them to have human interaction in real-time surroundings. The paper presents a detailed database concerning the models utilized to deal with the learning of long haul conditions in a chatbot. The paper proposes a novel crossbreed Long Short Term Memory based Ensemble model to retain the information in specific situations. The proposed model uses a characterized number of Long Short Term Memory Networks as a significant aspect of its working as one to create the aggregate forecast class for the information inquiry and conversation. We found that both of the ensemble methods LSTM and GRU work well in different dataset environments and the ensemble technique is an effective one in chatbot applications.

Keywords: Chatbot / AI / LSTM / Ensemble Method / GRU / RNN / Neural turing machine

1 Introduction

A Conversational Agent is otherwise called 'Chatbot' is a software program which leads a discussion by means of sound-related or literary strategies in a characteristic language, for example, English. Chatbots are being coordinated universally into our lives in a type of Virtual collaborators and messaging applications. In 1950, Alan Turing proposed 'Turing Test' as a benchmark of a chatbot program to imitate a human in a discussion [3]. ELIZA, Jaberwacky, ALICE were not many of the underlying chatbots created dependent on principle-based methodology [1]. Eliza was considered to be the first-ever chatbot to simulate a psychotherapist, in the 1960s. Although it was capable of establishing a dialogue, simulating a human being, his virtual model was based on rephrasing the user input whenever a collection of hand-crafted infusions matched. Eliza was not designed to model human cognitive capabilities; however, it demonstrated how software could create a significant impact through the mere illusion of comprehension. The 'Measurable Revolution' contingent upon Machine Learning blossomed in the late 1980s and mid-1990s. There has been a significant powerful movement in the zone of chatbot inferable from the development of human-made reasoning. The presentation of the influx of Artificial Intelligence-based chatbots has introduced another time of conversational interfaces. The other factor adding to advancement is the noticeable change of the elements of human discussion leaning toward short informing over different types of correspondence. Most chatbots work through remote helpers, advising applications or association's sites. Right now, the market of cutting edge conversational specialists is shared by IBM's Watson, Apple's Siri, Google Assistant, Amazon Alexa, Microsoft's Cortana to give some examples. Endeavours have been made to typify the usefulness of chatbot consistently into administrations alongside contracting the uniqueness contrasted with human discussions. The incorporation of an inductive memory practically equivalent to the human cerebrum into the engineering of a chatbot encourages the chatbot to keep up the edge of setting for longer durations. Protecting the highlights identified with the relationship for longer lengths is named as adapting Long term dependencies. This model acquires a factor of commonality and lucidness throughout the discussion between the end client and chatbot. Wide and unambiguous data about the advancement of memory incorporated models utilized in chatbot could give an exhaustive comprehension and bits of knowledge into the

* e-mail: pooja.kamat@sitpune.edu.in

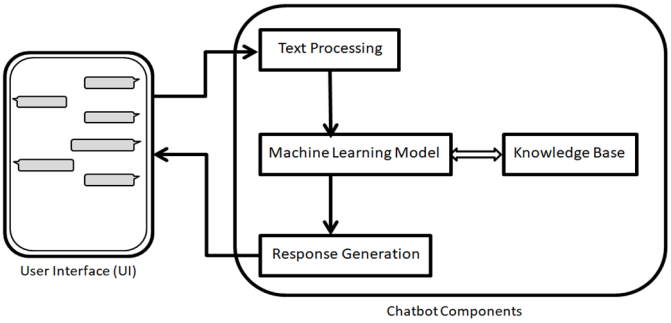


Fig. 1. Chatbot operation.

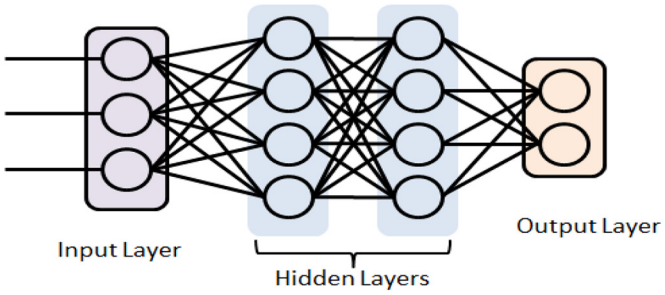


Fig. 2. Artificial neural network structure.

Table 1. Timeline of Technology Development.

Year	Author	Contribution
1943	W. McCulloch, W. Pitts	Artificial Neural Network (ANN)
1990	Elman	Simple Recurrent Neural Network (RNN)
1990	L.K. Hansen, P. Salamon	Ensemble Learning
1994	Y. Bengio	The issue with long term dependencies
1997	S. Hochreiter	Long Short Term Memory (LSTM)
2000	F.A. Gers	LSTM with forget gates
2014	K. Cho	Gated Recurrent Unit (GRU)
2014	A. Graves	Neural Turing Machine (NTM)

eventual fate of chatbot inquire about. The design and advancement strategy of a run of the mill chatbot relies upon fundamental ideas as determined in Figure 1 [2].

A brief look at the concepts to comprehend the variations possible at the formative stage is presented below.

- **Text Processing**: Word embeddings are the vector representations of words within the specific vocabulary enabling better implementation and utilization of statistical machine learning models.
- **Machine Learning Model**: The concept of Artificial Neural Network is extensively employed in dealing with input processing, classification and generating the most appropriate response for the input query. Chatbots like Deepprobe, Superagent utilize the Long short-term memory (LSTM) model with Seq2Seq, while Rubystar uses Seq2Seq with Gated recurrent unit (GRU) [2].
- **Knowledge Base**: The dataset used for training the model can be either Open or Close in its bounds. The open-domain chatbot was found to be compromised on relevance and accuracy of the responses, and Closed domain chatbot performs well owing to limited yet definite confines of the dataset [2].
- **Response Generation**: The response returned for input is either retrieved or generated. The former selects the appropriate response from a collection while the latter generates the response depending on the features of input vectors, dictionary a trained classifier. The hybrid RNN-Seq2Seq model has progressed to become a popular choice in chatbot architecture [3].

2 Literature survey

This section aims to provide an overview of different concepts employed in handling long-term dependencies and discuss their corresponding nuances. The timeline of some fundamental technologies is listed in Table 1.

2.1 Artificial neural networks

Artificial Neural Networks (ANNs) are information-processing models inspired by the biological neural system and the capability of the brain to process information. The work on Neuron Circuit and Perceptron by Warren Pitts, Warren McCulloch and F. Rosenblatt served groundwork for ANN to evolve and induct over the traditional computer frameworks in the 1970s [1]. ANN is composed of a large number of densely interconnected mathematical function units called 'Neurons' clustered into three types of layers, as shown in Figure 2. The input layer is responsible for the initial processing of input data, whereas the output layer deals with aggregating the final outputs and presenting the result. The weighted connections between neurons in the hidden layer form the basis of the learning process, providing variable strength to the input data traversing forward towards output neurons. An activation function like Sigmoid, ReLU or tanh is applied on the summation of weighted inputs in a neuron [4].

The model is trained using 'Back Propagation' where the error calculated leads to optimal updation of weights. Gradient (Δw) can be calculated as change in error with

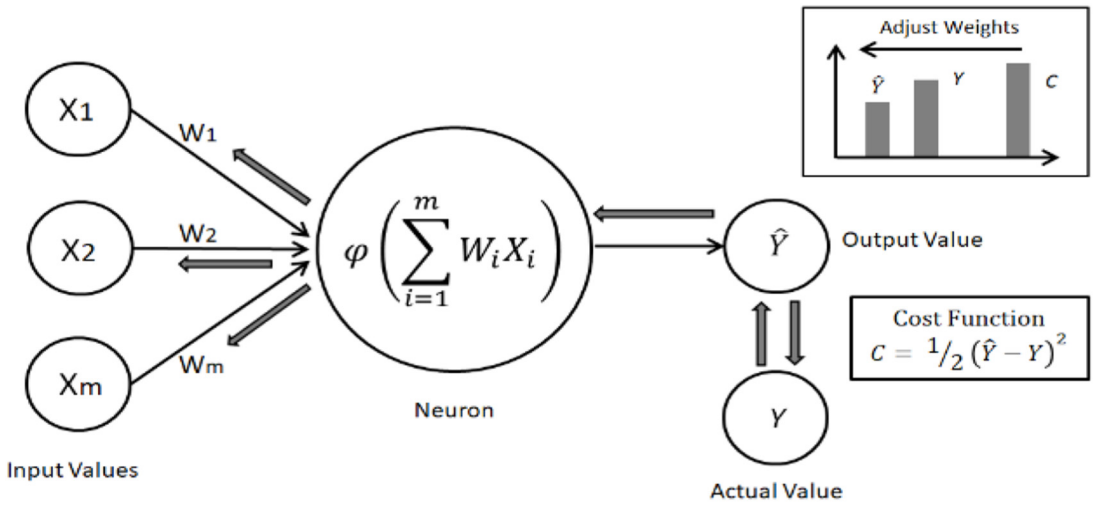
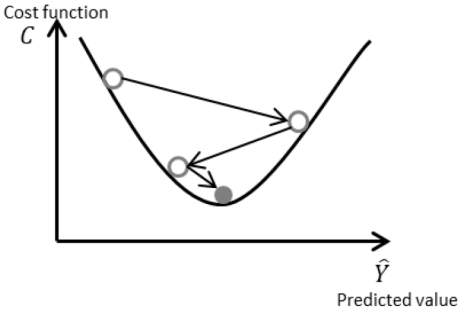
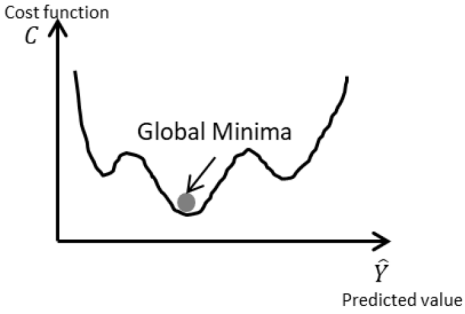


Fig. 3. Backpropagation process.

Table 2. Comparison between GD and SGD.

Gradient Descent	Stochastic Gradient Descent
	
GD computes gradient using a batch from the dataset	SGD computes gradients using single rows of training examples
It follows a deterministic approach	It follows a random approach
It converges slower on large training samples	It converges faster on large training samples
Steps: For every iteration – Traverse entire dataset – Evaluate gradient – Return	**Steps:** For every iteration – Iterate over each value in the dataset – Evaluate Gradient – Return

respect to change in weights $\left(\frac{de}{dw}\right)$. Values for new weights is determined by adding weight (w) and the gradient (Δw). The entire process is depicted in Figure 3.

However, the brute force approach for updating weights suffers from 'Curse of dimensionality' [5]. Gradient Descent (GD) and Stochastic Gradient (SGD) descent offer a faster way to find optimum weights. Both these methods determine the global minima by finding the point where the slope of the cost function is zero hence resulting in the error to be minimum. GD and SGD are compared in Table 2.

ANN suffers from both overfitting and underfitting. Overfitting is an outcome of an overly accurate or complicated model showing low bias but high variance. Underfitting is a result of a too simple model showing low variance but high bias [6]. ANNs deal with fixed-sized vectors only and they do not possess a dedicate memory element to handle sequential data hence making them an inappropriate choice for a chatbot handling dependent vectors.

2.2 RNN

Recurrent Neural Network (RNN) is the class of Artificial Neural Network supplemented by the integration of edges spanning adjacent timestamps. Psychologist David

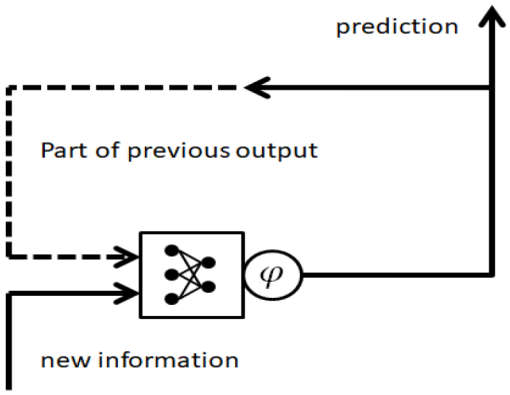
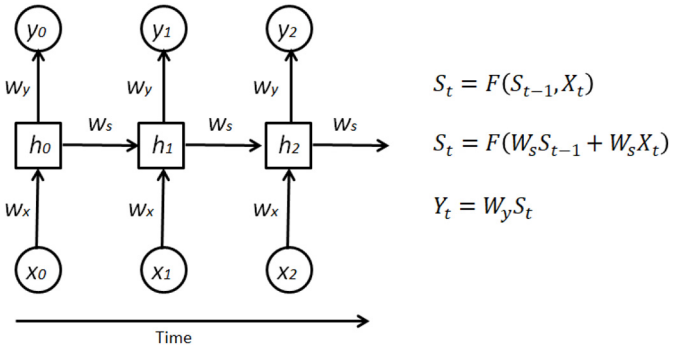


Fig. 4. RNN overall architecture.



$$S_t = F(S_{t-1}, X_t)$$

$$S_t = F(W_s S_{t-1} + W_s X_t)$$

$$Y_t = W_y S_t$$

Fig. 5. RNN unrolled structure.

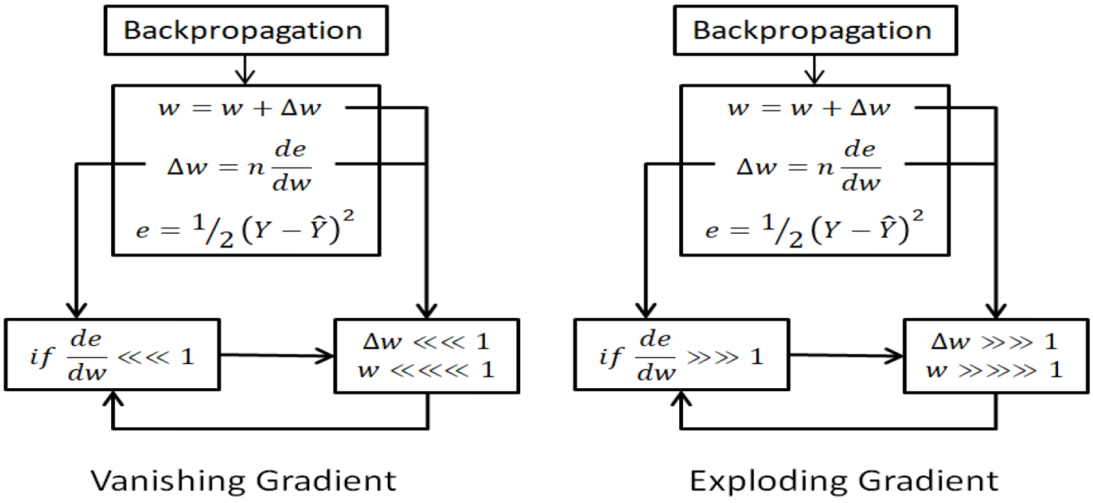


Fig. 6. Vanishing gradient and exploding gradient.

Rumelhart's work on symbolic artificial intelligence from 1986 formed base for the development of RNN. RNN has two inputs, the present values and values from recent past, enabling it to capture the dynamics of a sequence of inputs in scenarios like handwriting recognition, stock price prediction, etc. Owing to the variable size of input and output vectors, RNN has shown significant improvement over traditional feed-forward networks in Chatbots as RNNs are capable of exploiting a dynamically changing contextual window over input sequences. The overall architecture of RNN is specified in Figure 4.

At given time t, output for the state S_t is calculated applying a function on the portion of the output from the previous state S_{t-1} and current input X_t. It can be termed mathematically as $S_t = F(S_{t-1}, X_t)$ where F is activation function like $tan\,h$ or ReLU. This process continues forming an information loop for a given state concerning time. The unrolled structure of RNN is shown in Figure 5, along with equations.

Like Feed-forward networks, RNNs use backpropagation for training the difference being the additional parameter 'time'. Hence it is termed as 'Backpropagation through time (BTT)' as shown in Figure 5 [7].

The range of context to be used practically is limited as each prediction looks at one step prior state value. While back-propagating the recurrent connections, the influence of given input vector on the corresponding hidden layer and hence overall network output either decays or blows up exponentially giving rise to Vanishing Gradient and Exploding Gradient problem respectively as shown in Figure 6. Both these problems cause the model to train poorly and performance degradation.

A prediction of a state at the time 't' depends on the input presented at earlier time T where T << t. When the gap between T and t grows large, it becomes challenging for the model to attain convergence causing the failure of RNN to handle 'Long Term Dependencies' which makes it unfitting model for chatbots dealing with time series conversations [8].

2.3 LSTM

Long Short Term Memory networks are an extension for Recurrent Neural Networks with explicitly extended memory capability well suited to handle long term dependencies [9]. LSTM networks were proposed by

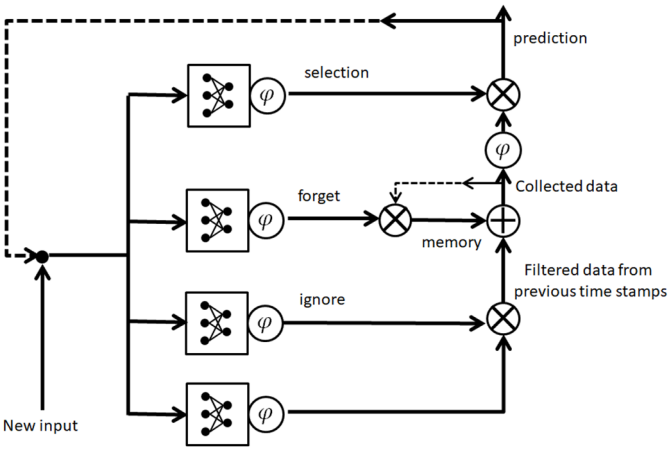


Fig. 7. LSTM network.

German researchers Sepp Hochreiter and Juergen Schmidhuber in 1997 as a solution to the vanishing gradient problem [10]. In comparison, LSTM can learn to bridge the features in excess about 1000 definite time steps by imposing constant error flow through the units termed as 'cells' effectively dealing with Long Term dependencies [11].

LSTM contain information from a context in a gated cell. The cells control the data to be written, stored, read and erased using Forget, Input and Output gates which are implemented with element-wise multiplications by sigmoids, as shown in Figure 7 [7]. The forget gate learns the weights controlling the decay rate of values stored in memory cells. For the instance when the input and output gates are off, and the forget gate is not causing decay, the memory cell maintains its value over time causing the gradient of the error to stay constant during back-propagation. This enables the model to remember information for more extended periods. The overall architecture of LSTM is shown in Figure 7.

Mathematically each step can be explained as follows [7]:

– In the first step Forget Gate layer decides the features to be flushed out from cell state looking at h_{t-1} and new input x_t.

$$f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \tag{1}$$

– In the second step, deciding the information be stored in the cell state is done in two steps. Input Gate layer i_t which is a sigmoid layer establishes the values to be updated. Then a $\tan h$ layer generates the vector of new candidate values $\tilde{C}_t$.

$$i_t = \sigma\left(w_i\left[h_{t-1,}x_t\right] + b_i\right) \tag{2}$$

$$c_t = \tan h\left(w_c\left[h_{t-1,}x_t\right] + b_c\right) \tag{3}$$

– The old cell state C_{t-1} is updated to a new cell C_t summing the output from Forget gate layer function f_t and $i_t \times \tilde{C}_t$.

$$c_t = f_t \times c_{t-1} + i_t \times c_t$$

– The output is determined in two steps – First, the sigmoid layer decides the parts of cells to output o_t. The product of the new cell state C_t through $\tan h$ and the output of sigmoid gate outputs h_t the selectively decided parts.

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{4}$$

$$h_t = o_t \times \tan h(c_t) \tag{5}$$

Hyperparameters tuning and optimization is an arduous and experimental task [4]. The training of the LSTM model is expensive in terms of memory and computational power. In the domain of chatbots for time series conversations, LSTM is shown to perform well and maintain the context for longer durations.

2.4 GRU

A Gated recurrent unit (GRU) is a specific model of Recurrent Neural Network introduced by Kyunghyun Cho in 2014 as a variation of an intermediate company like LSTM enabling the recurrent unit to capture dependencies of different time steps.

Unlike LSTMs, GRU has two gates as Reset and Update to control the flow of information and refine the outputs. When compared to LSTM, the update gate can be considered a combination of Forget and Input gate from LSTM. Update gate determines the portion of information from previous time steps needs to be passed to the next states. This gives GRU an edge over LSTM as the model can decide to maintain all features of earlier timestamps. Reset gate is used to determine the irrelevant part of the information which needs to be discarded [12]. GRU works in the following steps:

– Update gate $[z_t]$ at a time 't' is calculated.

$$z_t = \sigma(W_z.[h_{t-1}, x_t]) \tag{6}$$

– Reset gate $[r_t]$ calculates the information to be forgotten using

$$r_t = \sigma(W_r.[h_{t-1}, x_t]) \tag{7}$$

– New memory content is introduced which uses the reset gate to store the relevant information

$$\tilde{h_t} = \tan h(W.[r_t \times h_{t-1}, x_t]) \tag{8}$$

– $[h_t]$ is calculated, which holds information for the current unit using update gate output and memory content from previous steps $[h_{t-1}]$.

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h_t} \tag{9}$$

GRU exposes complete memory content without control gate when compared to controlled exposure of LSTM using Output gate. GRU explicitly holds the influx of information while calculating new memory content using the Update gate. Owing to less complicated nature and few tensor operations, GRU is computationally more effective and faster to train.

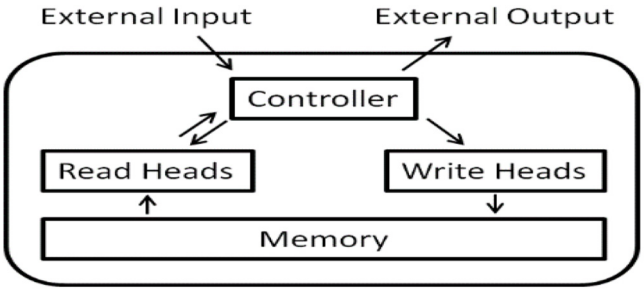


Fig. 8. NTM architecture.

2.5 NTM

Neural Turing Machine (NTM) explores the concept of evidently extending the context accumulator of RNN with an addressable external memory. They are an example of Memory Augmented Neural Networks, which decouple the computation from memory [13]. NTM have been shown to outperform LSTMs on sequence learning tasks demanding large memory for handling memorization of more extended contexts.

Controller and Memory matrix are primary components in NTM ash shown in Figure 8. The controller is a recurrent or feed-forward neural network which takes input and returns the output. External memory unit constitutes of $N \times W$ memory matrix where N is the number of memory locations, and W is the dimension of each memory cell. The interaction between the Controller and Memory matrix is carried out by reading and write heads. The memory matrix is initialized using schemes like Constant initialization or Truncated Normal distribution [13]. The NTM model can be trained by variants of stochastic gradients using backpropagation through time in case of an RNN based controller.

Algorithmic tasks like priority sort, Associative Recall, Copy, Repeat Copy, etc. can be performed to test if the NTM could be trained via supervised learning for efficient performance. NTM models generalize reasonably well to longer inputs.

2.6 Ensemble learning

The concept of ensemble learning was popularized in 1990 by Hansen and Salamon [14] over the idea that performance of a set of classifiers outweighs that of a single classifier. The individual models work in unison, where the outputs are combined with a particular decision fusion strategy to output a single answer [15]. Owing to the combination of various learning models, the generalization ability turns to be healthier. The basic architecture of the Ensemble model is depicted in Figure 9. The variation in the member models is a critical factor for classification performance [16]; hence strategies, as follows, were proposed for boosting the diversity scale among the member learners:

– Employing different learning algorithms for diverse learners or using the same algorithm with variation in parameters.

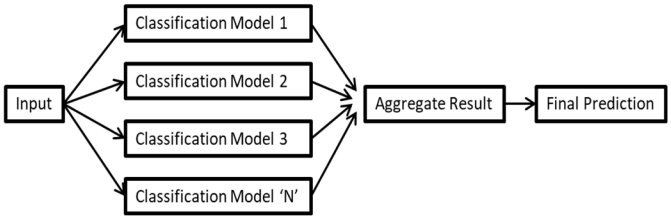


Fig. 9. Ensemble learning model.

– Training the members with varied datasets by subsampling or changing the attributes.
– Combination of the above two methods is used simultaneously.

An overall comparison between the concepts discussed along with the problem statements each methodology is well suited for is stated in Table 3.

The authors now present a comprehensive review of some of the recent works carried out in this domain (Tab. 4):

3 Dataset

3.1 Data source

Dataset plays a prominent role in making the conversation with chatbots as realistic as possible. For this purpose, we have to teach the chatbot model how to understand the text entered by a user and how to answer back. The format and syntax of this conversation, depending upon the purpose of the chatbot will also vary. For this purpose, various categories of datasets are available for the training of chatbots. An overview of the same is given in Figure 10 below:

The dataset used in the fundamental research is Cornell Movie Dialog Corpus. Cornell Edu distributes it. The dataset consists of different metadata-rich files. The conversations in the dataset are extracted from movie scripts. The dataset in whole has 220579 exchanges between 10292 characters collected from 617 movies. Two files are used for establishing the conversation data. 'Movie_lines.txt' contains texts from the dialogues and it has attributes like lineID, CharacterId, movieID, character name and the actual text. 'movie_conversations.txt' forms the structure of the conversation. It maps the conversation between two characterIDs together along with the movieID of the movie. The '+++$+++' acts as the field separator between the attributes mentioned for each file utilized.

3.2 Data preprocessing

Python with NumPy is used to preprocess the dataset to institute the conversation dictionaries. Dictionaries are created to map each line and the corresponding 'id', creating a list of all conversations, separating questions and answers. Individual, conjoint words are cleaned and replaced with simple words. A dictionary is also designed to map each word with its number of occurrences and for mapping the questions words and answer words to unique integer values.

Table 3. Overall comparison of Concepts from Literature Survey.

Model	Advantages	Suited problem statements	Chatbot applications using the model
ANN [17–19]	– Self-organizing to changes in information – Fault-tolerant to the corruption of cells and missing input values	ANN is well suited for classification and regression dealing with a large number of variables. Character recognition, Image processing are some of the applications for ANN	eHealth Chatbot, Tourism Chatbot, Quiz Generation and Answering Chatbot
RNN [20–22]	– Adapts wells to quick changes in the input nodes – Variable size of input and output vectors – Works well with contextual input sequences – Excel at modelling temporal structure	Appropriate for sequence prediction, classification prediction, and Natural language processing and generative model. Hence they can be used in text generation, prediction of the values of an attribute in a problem statement.	Combating Depression, Emotion Recognition, Technical Support Automation
Ensemble Learning [23–25]	– Better generalization ability – Weak models can be boosted to efficient learners – Because of growing computations power, the Ensemble model can be well utilized	It can be used to enhance the performance of existing models like RNN, LSTM and GRU.	Diabetes Personalised Treatment Chatbot, Fashion Recommender Chatbot
LSTM [26–28]	– Extended memory capabilities than RNN – Handles Long-term dependencies well. – More robust to vanishing gradients that RNN	The right choice for problem statements like Time series forecasting. LSTM can be applied in Conversation agent, handwriting generation, Language translation, Image captioning.	Twitterbot, AgriBot, Midoriko Anime Chatbot
GRU [29,30]	– Handles Long-term dependencies effectively – Robust to Vanishing gradient problem – Computationally effective than LSTM	GRU can be used in applications related to time series prediction like text generations, classification, etc	Cultural Heritage website chatbot, Papaya English dialogue chatbot
NTM [31,32]	– Generalize well to longer inputs as compared to LSTM – Presence of external memory complements the RNNs existing memory	NTM is well suited for models with massive and more extended sequences of data. NTM has demonstrated the solutions to be generalizing well for basic algorithms like copying and sorting.	Agnostic Chatbot, Arabic Chatbot

Table 4. Comprehensive review of recent works involving AI for chatbot implementation.

Sr no.	Research paper	Year	Algorithm used	Research findings
1.	M. Nuruzzaman and O. K. Hussain, 'A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks,' [33]	2018	ANN	Artificial Neural Network (ANN) owing to its capability to handle the complicated combination of features provides the most appropriate base to work upon for a problem statement such as Conversational Agents or Chatbots.
2.	Lee, M. C., Chiang, S. Y., Yeh, S. C., & Wen, T. F. 'Study on emotion recognition and companion Chatbot using deep neural network' [21]	2020	RNN	RNN provides a better response to problem statements about Seq2Seq framework of RNN built over Domain-Specific Knowledgebase.
3.	Bali, M., Mohanty, S., Chatterjee, S., Sarma, M., & Puravankara, R. Diabot: 'A Predictive Medical Chatbot using Ensemble Learning' [24]	2019	Ensemble Learning	Ensemble Learning as a meta-algorithm has the potential to provide better generalization. The increased performance can be mapped with strong correlations with a humane sense of conversation
4.	Pathak, K., & Arya, A. 'A Metaphorical Study Of Variants Of Recurrent Neural Network Models For A Context Learning Chatbot' [34]	2019	LSTM	LSTM is the most appropriate choice when the states of dialogues & responses in a conversation need to be tracked and predicted.
5.	G. Dzakwan and A. Purwarianti, 'Comparative Study of Topology and Feature Variants for Non-Task-Oriented Chatbot using Sequence to Sequence Learning,' [35]	2018	GRU	The Encoder-Decoder framework over GRU comes in as an alternative to LSTM for the chatbot. GRU offers similar if not better performance over LSTM when compared over the parameters like the size of the dataset, the resources being consumed.

4 Proposed method

Ensemble learning forms the basis of the proposed methodology. Classifiers like Support vector machines, Linear regression were used in the Ensemble model initially. With the onset of Deep learning, a more elaborate approach can be followed to improve the overall performance of the Ensemble model. The idea is to define a number of LSTM networks with variation in hyperparameters as part of the ensemble model. The member models work together in parallel, and their individual outputs are aggregated to generate the output of the overall model. As a fine-tuning measure, the concept of Pruning is also employed. An architectural overview is presented in Figure 9, followed by detailed Ensemble Network algorithm. Segmentation, Vector Space Model (VSM), Classification algorithm & Response generation forms the primary components of the chatbot. The flow of operations is shown in Figure 11. After the output class is predicted, the output of Chabot is returned to the user.

Components of the LSTM based Ensemble network are described in Figure 12.

- **Input Data**: Input sentences are segmented into terms. These terms are transformed into vectors by VSM corresponding to Vector space. The vectors generated are fed into the classifier model.
- **LSTMs based Ensemble Network**: The ensemble network consists of a defined number of LSTM networks working concurrently to generate the overall output prediction. The variation in hyperparameters like the number of hidden layers, number of neurons forms the basis of the distinction between each LSTM model. This leads to the training of models with different generalization features and accuracy metrics.
- **Prediction**: The LSTM models in the ensemble generate the predicted output, which is converted to a predicted class.

The Encoder-Decoder LSTM acts the base for defining the even single LSTM as well as the combination of LSTMs

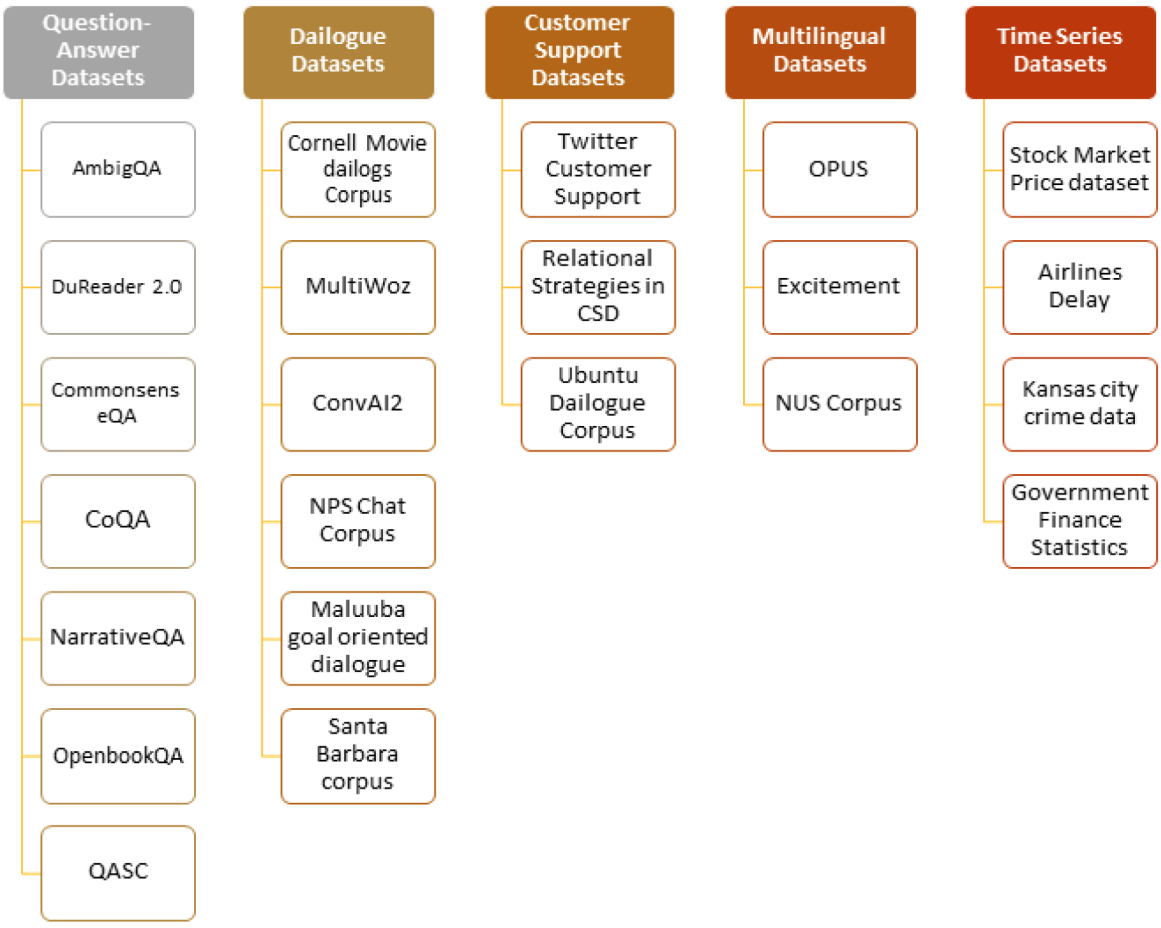


Fig. 10. Overview of chatbot datasets.

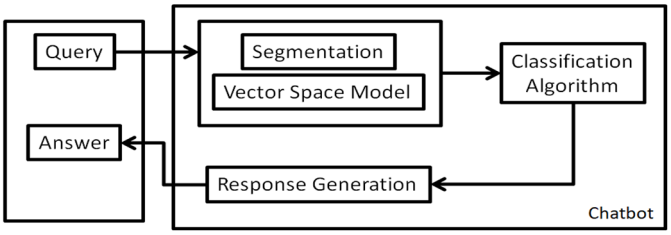


Fig. 11. Proposed chatbot architecture.

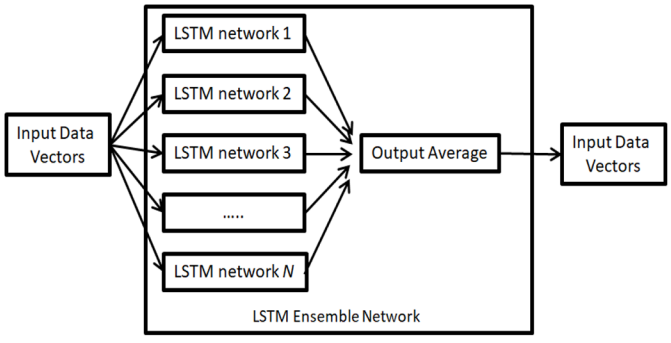


Fig. 12. LSTM based ensemble network classifier.

working in unison as part of the ensemble. The entire process of implementation can be broken down in specific steps, as discussed below. The proposed model includes training phase and the testing phase, which are shown in Figures 11 and 12, respectively. In the training phase, a definite number of LSTM networks are generated and trained using variations in training data. The models with lower accuracy are filtered out. In the testing phase, the models with higher accuracy work in conjunction to predict the output class from the calculated output weights. The detail of the two steps is presented in the following sections. The notations used in the algorithm specifications are specified in Table 5.

4.1 Data preprocessing

The dataset is preprocessed as defined in the section NUMBER. Different functions are created for the generation of dictionaries and cleaning the text.

4.2 Building the model

Encoder LSTM is responsible for reading the input sequence and encoding the same into a vector essentially

to map the corresponding vector from the vector space defined. Decoder LSTM deals with decoding the vector generated and outputting the predicted sequence. Encoder–Decoder LSTM generates a continual representation of data from a considerable number of data attributes from previous timestamps. This architecture of Encoder–Decoder LSTM was found useful on long and continuous data influx. We split the dataset into training and validation dataset as an attempt to carry out cross-validation. As seen in Figure 12, three different decoder LSTMs are created to decode the training data, decode the validation data, and the actual decoder for the encoder created.

Table 5. Notations used in the algorithm.

Notation	Description
N	The number of LSTM networks
$h_{\max}$	Maximum number of hidden layers in an LSTM model
$p_{\max}$	Maximum number of neurons in a hidden layer
$w_{\text{threshold}}$	A threshold value for the accuracy
h_m	Number of hidden layers in LSTM model
p_m	Number of neurons in a hidden layer
T	Total number of partitions of the training dataset
A_m	Average Accuracy
k	LSTM models remaining after pruning
C_k	Output of k^{th} LSTM model

4.3 Training phase

The hyperparameters like the number of epochs, batch size, LSTM size, number of layers in Encoder and Decoder LSTM, Learning rate are initialized for single LSTM as well as the Ensemble LSTM model. A session for training is initialized, and the models are training for both portions of the dataset that, that is, the Training dataset and validation dataset as well. As the training progresses, the model generates the weights. The model generalizes the data for patterns and features and stores it in the model to be utilized while testing. The training phase is also depicted in Figure 13.

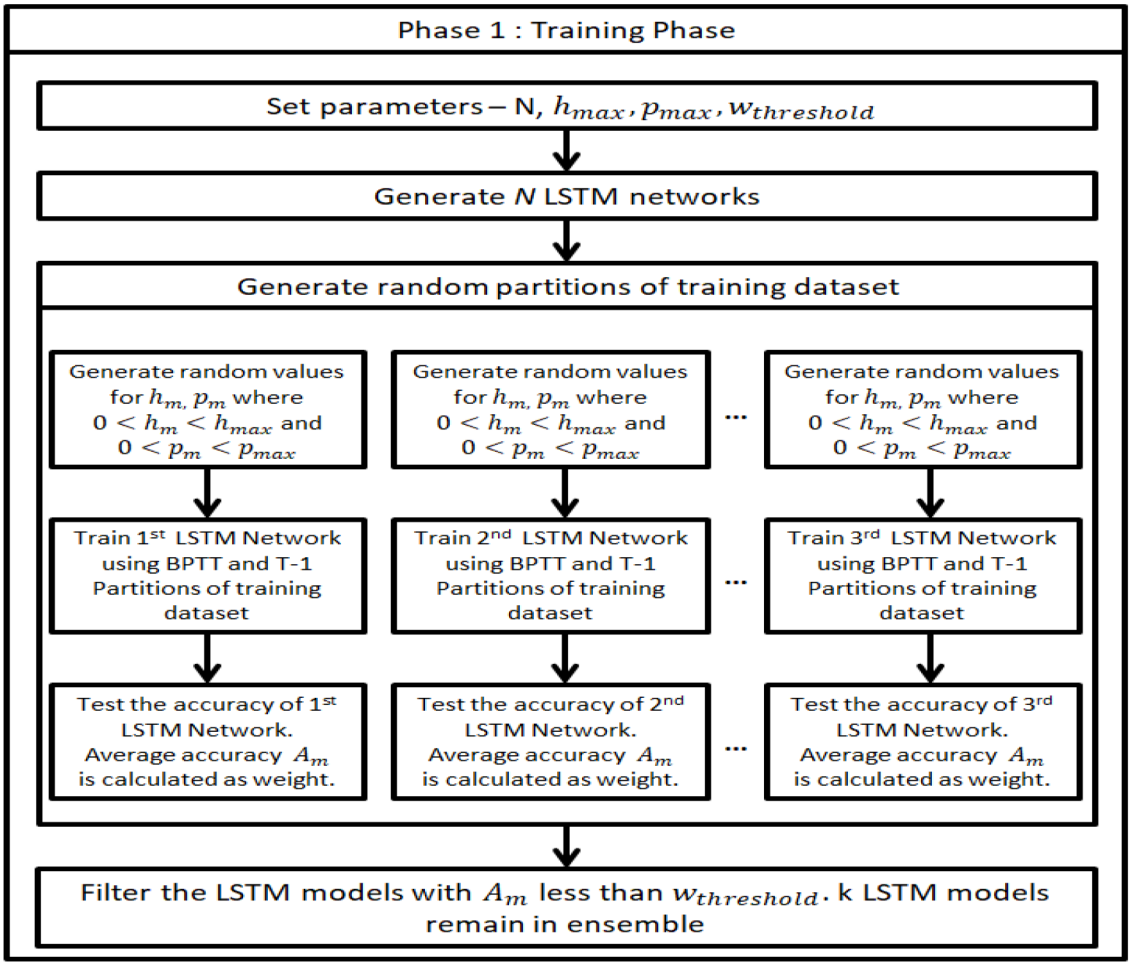


Fig. 13. Training phase algorithm.

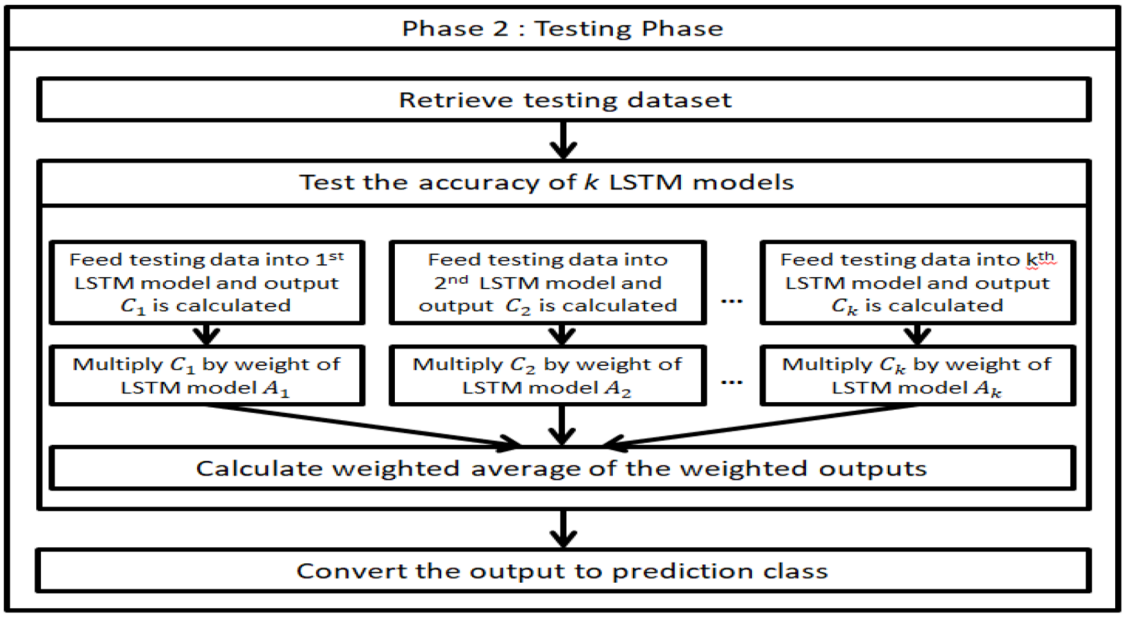


Fig. 14. Testing phase algorithm.

Step 1: The parameters like the number of LSTM networks N, the maximum number of hidden layers in the dmodel $h_{\max}$, a maximum number of neurons in a hidden layer $p_{\max}$ and the accuracy threshold for each LSTM model $w_{\text{threshold}}$ are initialized.

Step 2: Generate N LSTM models with variations in hyperparameters. The number of hidden layers h_m and the number of neurons in a hidden layer p_m are assigned random values between 0 and corresponding maximum bounds.

Step 3: The training dataset is split into T partitions. $T-1$ partitions are used to train each LSTM model. Every different LSTM model trains and learns using BPTT.

Step 4: The performance of each LSTM model along with the average accuracy A_m is evaluated using the remaining single partition from testing dataset.

Step 5: The LSTM models with accuracy lower than the accuracy threshold $w_{\text{threshold}}$ are dropped, leaving k models in the ensemble network.

4.4 Testing phase

The weights generated in training are loaded, and a session is initiated running the Encoder–Decoder LSTM model as a part of both the single model and Ensemble LSTM. The incoming queries are cleaned and preprocessed using the functions defined in Data Preprocessing. The predicted answer is returned to the user. This phase deals with applying the patterns and features learned on the testing dataset. The testing dataset is fed into the Ensemble model to predict the output classes. The testing phase is depicted in Figure 14.

Step 1: The testing dataset is retrieved and fed into the LSTM models of the ensemble.

Step 2: The output C_k for the individual model is calculated. C_k is multiplied with corresponding weights of each model. A weighted average of the sum of total weighted outputs is calculated.

Step 3: The weighted average is used to predict the response class.

An example of the variations in the hypermeters for multiple LSTM multiples as a part of the Ensemble Network is presented in Figure 15.

5 Performance analysis

RNN, LSTM and GRU serve as the best choice for the classifier in Ensemble Network. LSTM and GRU provide an edge over RNN owing to the presence of a dedicated memory control unit enabling the learning of long term dependencies. The selection of an appropriate model between them depends on the key differences and the dataset. GRU exposes complete memory content without control gate when compared to controlled exposure of LSTM using Output gate. LSTM doesn't control the amount of information flowing in from previous time steps while computing new memory content. On the other hand, GRU explicitly controls the influx of information while calculating new memory content using the Update gate. An experiment was performed to compare LSTM and GRU for their performance in the time series prediction.

5.1 Comparison between LSTM and GRU

5.1.1 Prediction comparison

LSTM and GRU are closely related mechanisms for handling long term dependencies. A comparison between both for their performance provided important insight for the selection of LSTM over GRU, as seen in Figure 16. Two models constituting a single LSTM and a single GRU were created for the comparison. To make the comparison more

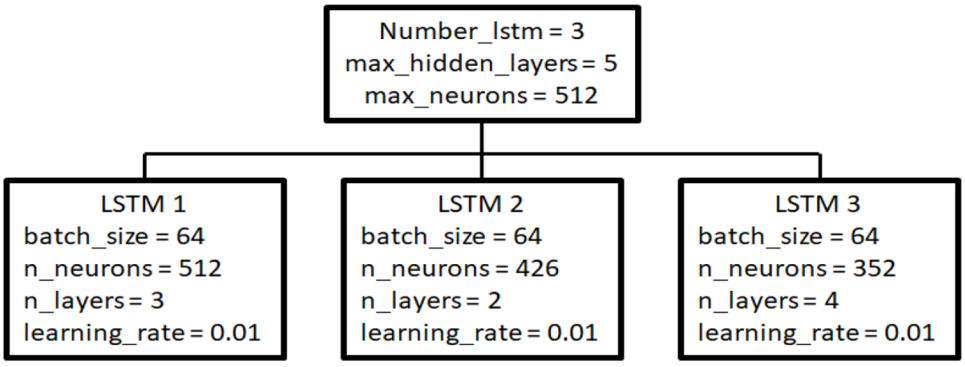


Fig. 15. Hyperparameter variations in LSTM.

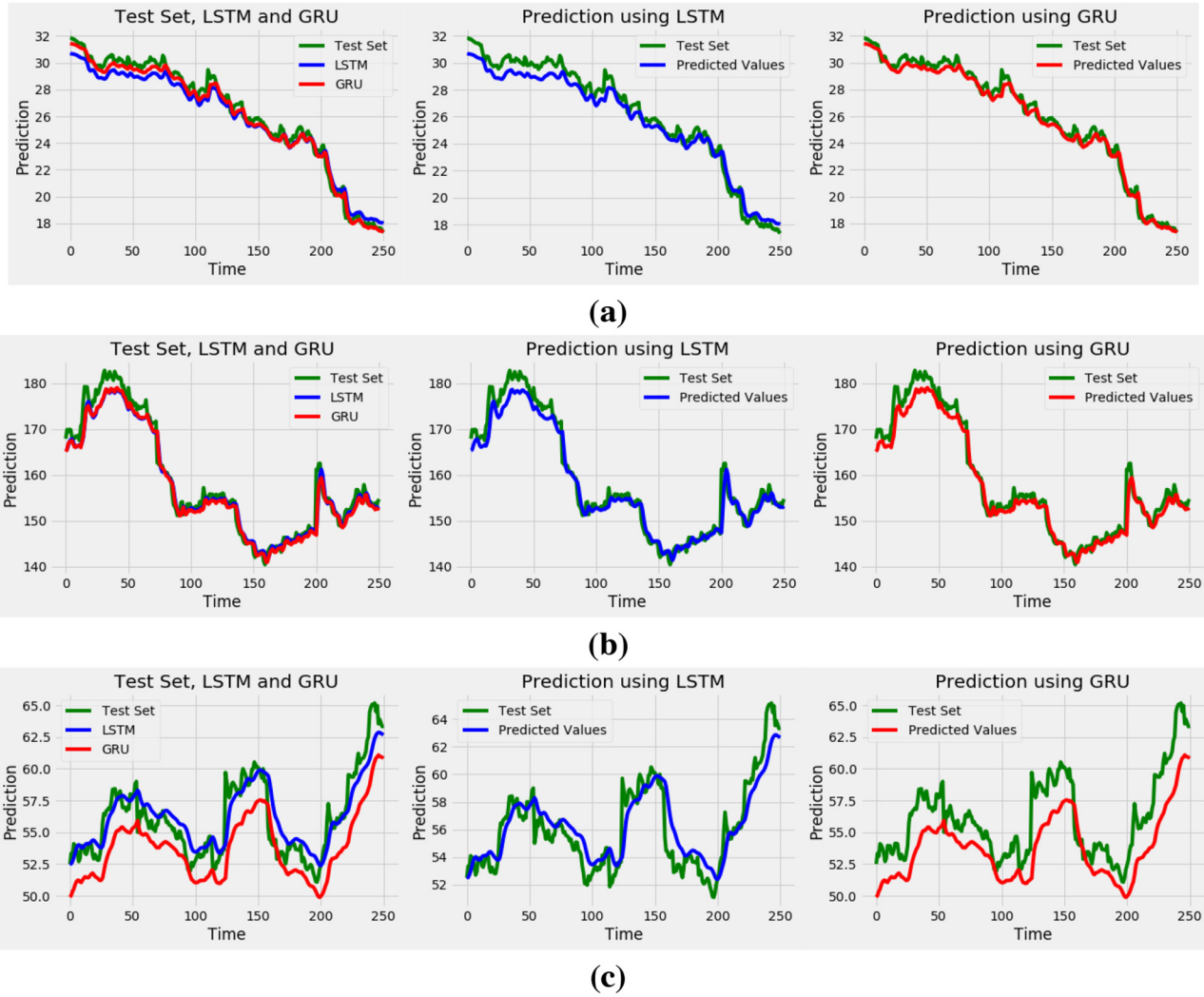


Fig. 16. Outputs for dataset variations. (a) Dataset 1. (b) Dataset 2. (c) Dataset 3.

just, both LSTM and GRU models had four hidden layers with 50 neurons each with 0.2 dropout rate. Both the models were trained with 50 epochs and a batch size of 32. The dataset used to select the appropriate model for time series data analysis is Stock Price Dataset. The Dataset is split into portions randomly to generate a more stable and evenly spread out output values. In the first split of the dataset, there are 700 entries in total. In the second split, the dataset contains 1400 entries of data points. The final dataset consists of 3000 data points. The models developed

Table 6. Mean squared error for LSTM and GRU on Datasets.

	LSTM	GRU
Dataset 1	0.7516197811968257	0.4601261472286205
Dataset 2	2.374904252912631	2.2920254537142664
Dataset 3	1.4342299451204816	2.6693156947773167

Table 7. Loss values for each dataset.

	Dataset 1	Dataset 2	Dataset 3
LSTM	0.002012	0.013321	0.001093
GRU	0.00151	0.00129	0.02667

were executed on the three variations of the dataset to analyze the parameters like Mean Squared Error, Accuracy, Loss and Time taken for training and ultimately deciding the most applicable model for handling time series data as well as long term dependencies.

For Dataset 1 with700 data points, the performance of GRU seems better than LSTM as the predicted values by GRU, and the Testing values map well than that of LSTM. In iteration for Dataset 2 with 1400 data points, the performance of LSTM and GRU both seem almost equal as the values predicted by both the models correspond with the testing values. In the iteration for Dataset 3, GRU shows more error and deflect away from the Testing Data points. On the other hand, LSTM offers better performance over 3000 data points from Dataset 3.

5.1.2 Mean squared error analysis

The mean squared error is calculated for each dataset variation for both LSTM and GRU model. Mean squared error is one way to calculate the error during back-propagation which is the basis for or training for both LSTM and GRU. Higher error value indicates performance degradation and improper training. The mean squared error for both models on three datasets is presented in Table 6.

From the values of Mean Squared Error, we can conclude that GRU performs well and generates a standard error for datasets with small size with lesser data points. LSTM was found performing well for datasets with massive data points.

5.1.3 Average loss calculation

The loss values are calculated for each epoch for both LSTM and GRU. For a given model, the lesser the loss, the better is the training of the model. Failure is a summation of errors made for each batch of training dataset over an epoch. The average of loss values over the complete training procedure for 50 epochs is specified in Table 7 for the variations in a dataset.

Owing to less computational steps, GRU generates less error than LSTM. LSTM executes well over the last dataset variation having the highest number of data points.

Table 8. Training Time in minutes for LSTM and GRU.

	LSTM	GRU
Dataset 1	9.51	8.28
Dataset 2	12.91	11.59
Dataset 3	16.8	14.97

5.1.4 Training time comparison

The training time for the complete training process in minutes is specified for both LSTM and GRU in Table 8.

For all the disparity in the dataset, LSTM model takes more time to train over GRU. When LSTM and GRU were used to train the chatbot models, the model with LSTM showed overall better accuracy of 71.69% over 70.12% accuracy of GRU during Training. During Testing, the LSTM model showed better accuracy of 71.59% over 70.75% of GRU model.

Considering four parameters like Mean Squared Error, Loss, Training Time and Accuracy, we can deduce LSTM is a better choice over GRU as a whole.

5.2 Comparison between LSTM and Ensemble LSTM

Following graphs present a comparative look at the performance of LSTM and Ensemble Model for the same time series data analysis. The Ensemble Model consists of three LSTMs with variation in hyperparameters. The graphs depict the mapping between the values from the testing dataset and the predicted values from the corresponding model. The maps related to LSTM models are specified in Figure 17, and the graphs associated with Ensemble LSTM models are depicted in Figure 18.

From the graphs, we can conclude the performance of Ensemble Model is similar in the aspect of the performance of the LSTM model and in some cases, better. With fine-tuning the Ensemble models, the performance can be improved over standalone LSTM model.

5.3 Comparison between GRU and Ensemble GRU

The following graphs in Figures 19 and 20 provide overall performance analysis for single GRU model and Ensemble GRU model in respective sections. The Ensemble GRU model consists of three individual GRU models with variations in hyperparameters.

We derive from the graphs, the performance of single GRU and the Ensemble GRU is comparatively similar. The performance could be better enhanced with the diversity in the hyperparameters of singular models constituting the Ensemble model.

5.4 Comparison between Ensemble LSTM and Ensemble GRU

This section provides the differentiation in the performance between the Ensemble LSTM model and Ensemble GRU model against the prediction values from the testing

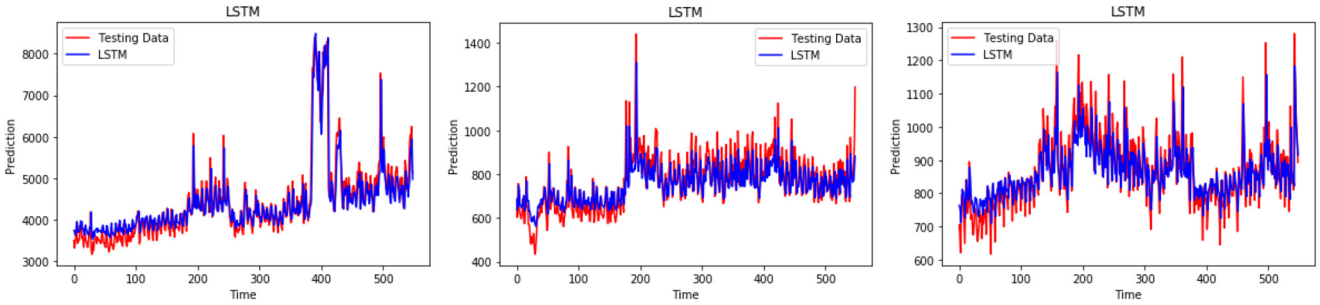


Fig. 17. Single LSTM with dataset variations.

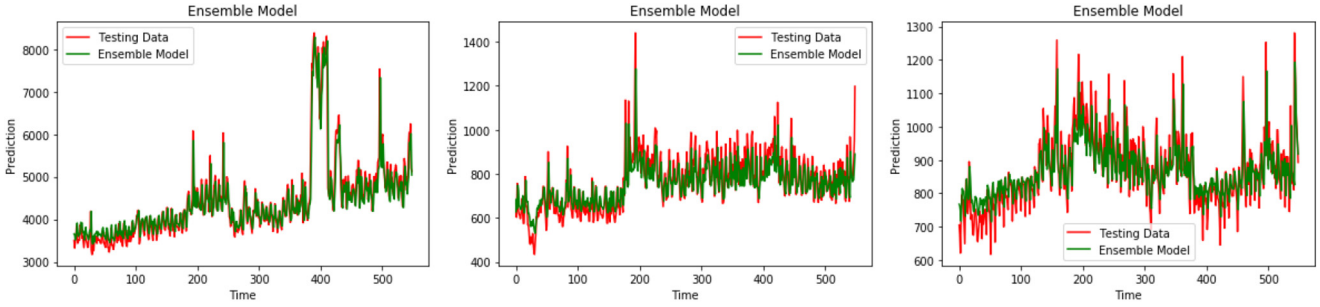


Fig. 18. Ensemble LSTM model with dataset variations.

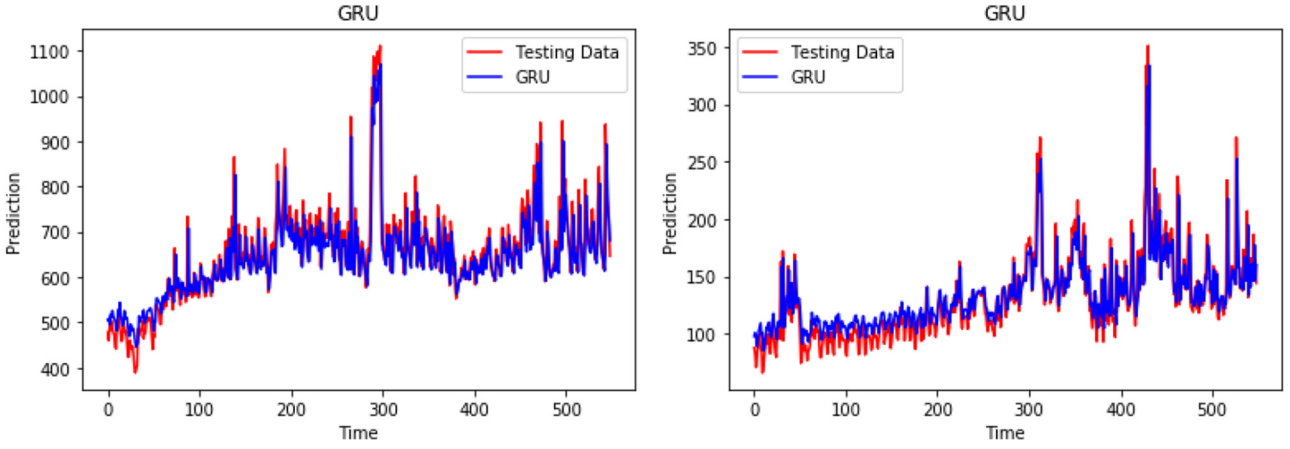


Fig. 19. Single GRU model with dataset variations.

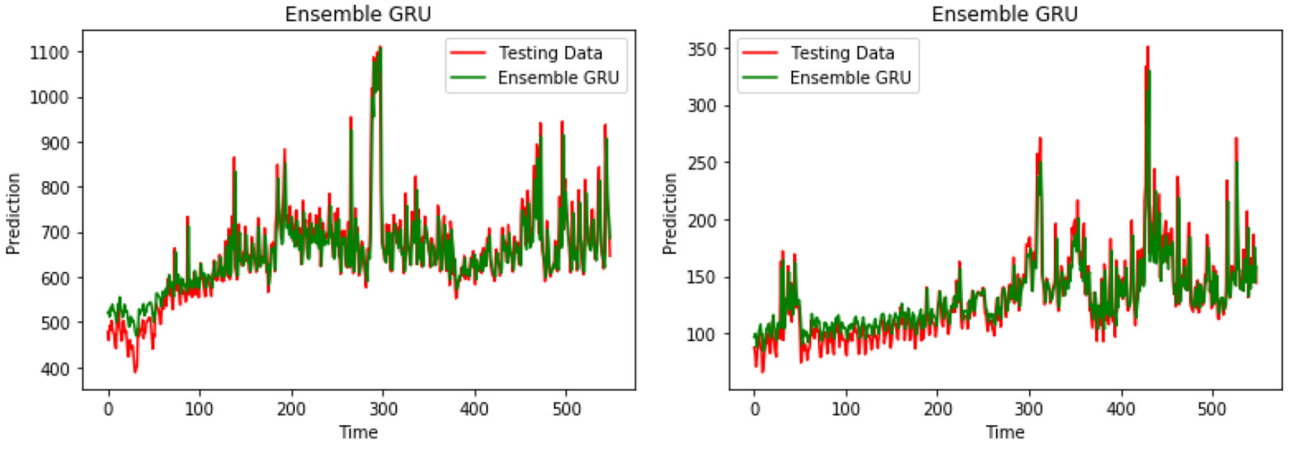


Fig. 20. Ensemble GRU with dataset variations.

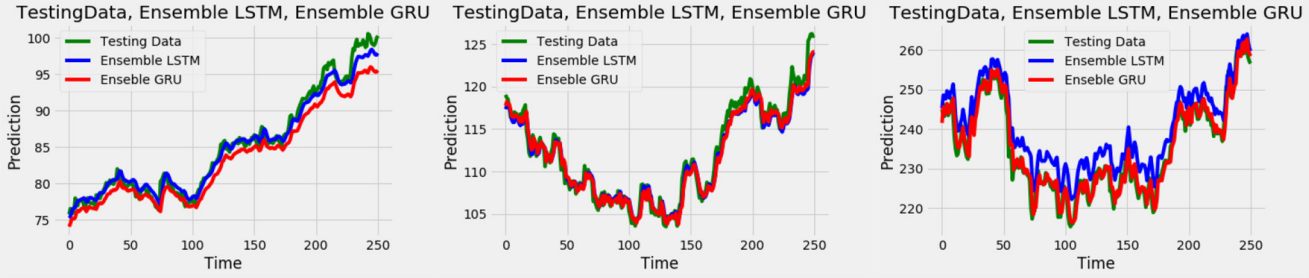


Fig. 21. Results over dataset variations for ensemble LSTM and ensemble GRU.

Table 9. Mean squared error analysis for ensemble LSTM and ensemble GRU.

	Ensemble LSTM	Ensemble GRU
Dataset 1	1.060990337370966	2.0204261732384396
Dataset 2	1.249621120822286	1.1770008219011074
Dataset 3	5.661777383126808	2.6198470374584173

Table 10. Average loss calculation for ensemble LSTM and ensemble GRU.

	Dataset 1	Dataset 2	Dataset 3
Ensemble LSTM	0.001069	0.034829	0.000179
Ensemble GRU	0.002512	0.00278	0.01917

dataset. The training was carried out on variations of the same dataset hence employing the strategy of cross-validation to make it most even comparison.

5.4.1 Prediction comparison – Ensemble LSTM, Ensemble GRU

From Figure 21, it can be deduced from the graph i and ii that the performance of both ensemble models is almost equal in terms of close mapping with the prediction values. However, in graph iii, it is observed that Ensemble GRU performance better than Ensemble LSTM. The values predicted by Ensemble GRU are closely mapped with the actual values from the testing dataset.

5.4.2 Mean squared error analysis

The final Mean Squared Error for both Ensemble models is calculated. Lower the values, better the performance of the model. The values are defined in Table 9.

From the values calculated, we can deduce that no specific model performs better than the other in every variation of the dataset. In two cases, the performance of both models was comparatively equal while in one case the Ensemble GRU performs better than Ensemble LSTM.

5.4.3 Average loss calculation

For each epoch during the training, Loss values are calculated. Lesser the loss values, better is the training of the model. The average loss values calculated for each dataset variation over the complete training is presented in Table 10.

From the values in the table, we deduce the average loss during the training of Ensemble GRU is lesser than the values of Ensemble LSTM. That indicates Ensemble GRU trains well with the dataset compared to Ensemble LSTM.

Based on the analysis of three parameters, we can conclude the performance of both Ensemble LSTM and Ensemble GRU is not definitively better than each other. In some cases Ensemble GRU performed better while in some cases Ensemble LSTM. With tweaking the parameters of individual models working together as an ensemble, the best performance can be achieved.

6 Conclusion

This paper presents a review of the evolution of technologies applied in Chatbots handling time series conversations in the labels of Architectural Design and Implementation. The paper also intends to contribute to developing a sturdy groundwork on the concepts utilized in learning long term dependencies hence providing a roadmap towards further enhancements being inclined towards minimalistic yet alike requisites. These primal conditions can be considered as 1. We are designing the word embedding schema not constrained by the knowledge base. 2. Flexible and accurate conversational model. 3. Reaching the true peak of imitating the human conversation requiring no human intervention. The proposed LSTM based Ensemble Network architecture attempts at enhancing the user experience by providing a sense of continuance of context in a series of conversations. The algorithm does so by generalizing the features imperative to making the conversation human. Future scope for this concept would be to think of applying similar approach using GPT-1, GPT-2 or GPT-3 that are widely applied nowadays in NLP domain.

References

1. S. Natale, If the software is narrative: Joseph Weizenbaum, artificial intelligence and the biographies of ELIZA, New Media Soc. **21**, 712–728 (2019)
2. E. Adamopoulou, L. Moussiades, An overview of chatbot technology, in: *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, Cham, 2020, pp. 373–383.
3. P. Goyal, S. Pandey, K. Jain, Developing a chatbot, in: *Deep Learning for Natural Language Processing, Apress, Berkeley*, 2018, pp. 169–229
4. H. Salehinejad, S. Sankar, J. Barfett, E. Colak, S. Valaee, Recent advances in recurrent neural networks, 2017. arXiv preprint arXiv:1801.01078
5. A. Prieto, J. Cabestany, F. Sandoval, Computational intelligence and bioinspired systems, Neurocomputing **70**, 16–18, 2701 (2007)
6. S. Lawrence, C. Giles, Overfitting and neural networks: conjugate gradient and backpropagation, presented at *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000
7. Z.C. Lipton, J. Berkowitz, C. Elkan, A critical review of recurrent neural networks for sequence learning, 2015. arXiv preprint arXiv:1506.00019
8. J.F. Kolen, S.C. Kremer, Gradient flow in recurrent nets: The difficulty of learning longterm dependencies, 237–243 (2001)
9. C. Olah, Understanding LSTM networks, 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
10. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735–1780 (1997)
11. F. Gers, Learning to forget: continual prediction with LSTM, presented at *9th International Conference on Artificial Neural Networks: ICANN 99*, 1999
12. J. Chung, C. Gulcehre, K. Cho, Y. Bengio, J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. arXiv:1412.3555
13. M. Collier, J. Beel, M. Collier, J. Beel, Implementing neural turing machines, 2018. arXiv:1807.08518
14. D. Opitz, R. Maclin, Popular ensemble methods: an Empirical Study, J. Artif. Intell. Res. **11**, 169–198 (1999)
15. L. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. **12**, 993–1001 (1990)
16. C. Chen, C. Wu, C. Lo, F. Hwang, An augmented reality question answering system based on ensemble neural networks, IEEE Access **5**, 17425–17435 (2017)
17. S. Ayanouz, B.A. Abdelhakim, M. Benhmed, A smart chatbot architecture based NLP and machine learning for health care assistance, *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*, Association for Computing Machinery, New York, 2020, pp. 1–6
18. S. Dey, D. Shukla, Analytical study on use of AI techniques in tourism sector for smarter customer experience management, *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2020, IEEE, pp. 1–5
19. A.S. Sreelakshmi, S.B. Abhinaya, A. Nair, S.J. Nirmala, A question answering and quiz generation chatbot for education, *2019 Grace Hopper Celebration India (GHCI)*, 2019, IEEE, pp. 1–6
20. F. Patel, R. Thakore, I. Nandwani, S.K. Bharti, Combating depression in students using an intelligent chatBot: a cognitive behavioral therapy, *2019 IEEE 16th India Council International Conference (INDICON)*, 2019, IEEE, pp. 1–4
21. M.C. Lee, S.Y. Chiang, S.C. Yeh, T.F. Wen, Study on emotion recognition and companion Chatbot using deep neural network, Multimed. Tools. Appl. **79**, 19629–19657 (2020)
22. G. Aalipour, P. Kumar, S. Aditham, T. Nguyen, A. Sood, Applications of sequence to sequence models for technical support automation, *2018 IEEE International Conference on Big Data (Big Data)*, 2018, IEEE, pp. 4861–4869
23. H. Cuayáhuitl, D. Lee, S. Ryu, Y. Cho, S. Choi, S. Indurthi, S. Yu, H. Choi, I. Hwang, J. Kim, Ensemble-based deep reinforcement learning for chatbots. Neurocomputing **366**, 118–130 (2019)
24. M. Bali, S. Mohanty, S. Chatterjee, M. Sarma, R. Puravankara, Diabot: a predictive medical chatbot using ensemble learning, Int. J. of Recent Technol. and Eng. **8**, 2277–3878 (2019)
25. R. Chakraborty, K. Vats, K. Baradia, T. Khan, S. Sarkar, S. Roychowdhury, Recommendence and fashionsence: online fashion advisor for offline experience, *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, Association for Computing Machinery, New York, 2019, pp. 256–259
26. Y.M. Çetinkaya, İ.H. Toroslu, H. Davulcu, Developing a Twitter bot that can join a discussion using state-of-the-art architectures, Soc. Netw. Anal. Min. **10**, 1–21 (2020)
27. B. Arora, D.S. Chaudhary, M. Satsangi, M. Yadav, L. Singh, P.S. Sudhish, Agribot: a natural language generative neural networks engine for agricultural applications, *2020 International Conference on Contemporary Computing and Applications (IC3A)*, 2020, IEEE, pp. 28–33
28. Y.T. Wan, C.C. Chiu, K.W. Liang, P.C. Chang, Midoriko Chatbot: LSTM-Based Emotional 3D Avatar, *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, IEEE, pp. 937–940
29. G. Sperlí, A deep learning based chatbot for cultural heritage, *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, Association for Computing Machinery, New York, 2020, pp. 935–937
30. G. Dzakwan, A. Purwarianti, Comparative study of topology and feature variants for non-task-oriented chatbot using sequence to sequence learning, *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, 2018, IEEE, pp. 135–140
31. P. Rivas, C. Chelsi, N. Nishit, L. Ravula, Application-agnostic chatbot deployment considerations: a case study, *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, IEEE, pp. 361–365
32. S. Al Humoud, A. Al Wazrah, W. Aldamegh, Arabic chatbots: a survey, Int. J. Adv. Comp. Sci. Appl. **9**, 535–541 (2018)

33. M. Nuruzzaman, O.K. Hussain, A survey on chatbot implementation in customer service industry through deep neural networks, *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, 2018, IEEE, pp. 54–61
34. K. Pathak, A. Arya, A metaphorical study of variants of recurrent neural network models for a context learning chatbot, *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 2019, IEEE, pp. 768–772
35. G. Dzakwan, A. Purwarianti, Comparative study of topology and feature variants for non-task-oriented chatbot using sequence to sequence learning, *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA), Krabi*, 2018, pp. 135–140

Cite this article as: Shruti Patil, Venkatesh M. Mudaliar, Pooja Kamat, Shilpa Gite, LSTM based Ensemble Network to enhance the learning of long-term dependencies in chatbot, Int. J. Simul. Multidisci. Des. Optim. **11**, 25 (2020)