Analysis of Automated text generation using Deep learning

Dr. Manoj Kumar
Dept. of Computer Science and Engineering, Delhi Technological University, New Delhi, India
mkumarg@dce.ac.in

Arnav Kumar
Dept. of Computer Science and Engineering, Delhi Technological University, New Delhi, India
arnavkumar1999@gmail.com

Abhishek Singh
Dept. of Computer Science and Engineering, Delhi Technological University, New Delhi, India
abhisheksanu26@gmail.com

Ankit Kumar
Dept. of Computer Science and Engineering, Delhi Technological University, New Delhi, India
ankitkumar50417@gmail.com

Abstract—

A chatbot is a computer program that can converse with humans using artificial intelligence in messaging platforms. The goal of the project is to use and optimize deep learning techniques for making an efficient chat bot. Among current chat bots many are developed using rule-based techniques, simple machine learning algorithms or retrieval-based techniques which doesn't generate good results.in this paper, we will be comparing performance of three chatbots build by using RNN, GRU and LSTM. These conversation chatbots are mostly used by different businesses, government organizations and non-profit organizations.

Index Terms— Natural Language Processing, Deep learning, Chatbots, Gated Recurrent Units, Long short-term memory.

I. INTRODUCTION

Text generation is task of giving our sentences as input and getting a response from the machine/computer. It is done using deep learning algorithms which are subset of Neural networks and give the best result and accuracy. It is interesting problem in the field of Natural Language Processing. We only have a supervised learning approach for text generation and that is present in sequence to sequence models of the RNNs. This Project is intended to apply analysis different types of sequence to sequence models for generating text which should be meaningful to the user (not random text). We applied three methods i.e Simple Recurrent Neural Network, LSTM and GRU and compared them to find out the difference in their performance we have also looked other research papers for the results on basic RNN and LSTM. We used the Cornell Movies Dataset for this paper. We tried to figure out the using existing learning algorithm like GRU how can we improve it to get it on par with LSTM Ease of Use

II. HISTORY AND RELATED WORK

A. *Early Approach*

They are many open source bot API on the internet, providing an easy way to developers to create bots. Bots like Mitsuku were among the smartest bot created. But in the beginning it was like a pattern matching the developer used to write a pattern a template such that when matching ours it responds with one of the templates. Rule Based make it easy for everyone to create a bot but it is quite inconvenient hard to make bot that answers complex queries. There are many approaches in making a chat box some are AI approaches and other are Non-AI approaches in this paper we will be talking about the AI approaches i.e. Retrieval Based and Generative Models.in the title or heads unless they are unavoidable.

B. *Intelligent Models*

1) ***Retrieval Based*****:** Retrieval Based Chat bot has a collection of predefined responses and the bot is trained to give the best fitting response from the collection of responses. In this model all the responses are to manually entered by a human. It doesn't generate any new responses so we don't need to worry about grammar. It usually tends to make mistake as it has a limited set of responses due to which it tends to be rigid responses appears to non -human. It's basic job was to select the most optimal response from the pool of predefined responses.

2) ***Generative Model:*** This model doesn't uses any predefined responses instead it generate response itself word by word due to which they are prone to grammatical errors. This method is used for developing a smart more human like bot that are quite advanced in nature.Its responses are still unpredictable the algorithms are quite complex. we will be comparing three of the generative model algorithms further in the paper.

C. ***Related Work***

Previously, Seq2Seq AI chat bots were developed using
encoder-decoder architecture. The encoder-decoder was built using a modified Recurrent Neural Network know as Long Short Term Memory cell units. Seq2Seq model predicts utterances unit by unit not considering what could occur in future which also leads to many frequent responses like "I don't get what you are saying" which was also because of the dataset having such sentences to avoid these issues Deep Reinforcement learning is used which is beyond the scope of this paper. Adam optimizer were also used as computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradient. It was also selected as it is a combination of best properties of AdaGrad and RMSProp [8]. Bidirectional LSTM were also used before in encoder side and attention mechanism used in decoder side to improve model performance. The advantage that the Bidirectional LSTM is that it works like two independent Recurrent Neural Network put together enabling the network to have both forward backward information of the sequence at every step of the word generation [10] i.e. one transfer information from past to future the other transfer it from future to past as it runs backwards it preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

III. PRELIMINARIES

Here we will discuss some various deep learning techniques
used for implementing chatbots.

A. Text Generation

Text generation comes under the branch of Natural Language processing where we use different deep learning algorithms and methods like sequence-to-sequence model with Recurrent neural networks, Gated recurrent units and long short-term memory respectively [1]. We can also different classification techniques to categorize words and generate sentences like Naive Bayes classifier.

B. Recurrent Neural Network

Special Type of Neural Network designed for natural language processing. Here we have our word embeddings or the bag of words model given as input to the recurrent neural network, then they'll produce an output based on the model (for our case it is text generation). Although the structure of this Neural Networks may seem complex it is not very good when we have different contexts. RNN normally will find it difficult to distinguish after the plural or singular words gets past as it doesn't have any memory to store the form of sentence.

C. Long Short Term Memory

Short-term, long-term memory network is a type of recurrent neural network (RNN). LSTM (Sepp Hochreiter et al [4]) are ideal for learning, processing and classifying sequential data. Its networks aim to overcome the problem of fading gradients by using gateways to selectively keep the information up to date and forget irrelevant details [3]. Reduced sensitivity to time intervals makes LSTM networks better for sequential data analysis than individual RNNs.

1) Architecture: The architecture of the LSTM (Sepp Hochreiter et al [4]) block is shown below. An LSTM unit typically has a storage cell, intake valve, exhaust valve, and bypass valve in addition to the hidden state of the RNN. The entry gate weights and offsets control how far the new value enters the cell. Likewise, the weights and offsets for the Oblivion gate and output control the extent to which the value remains in the cell and the extent to which the value in the cell is used to calculate the output activation of the LSTM-Block respectively [4].

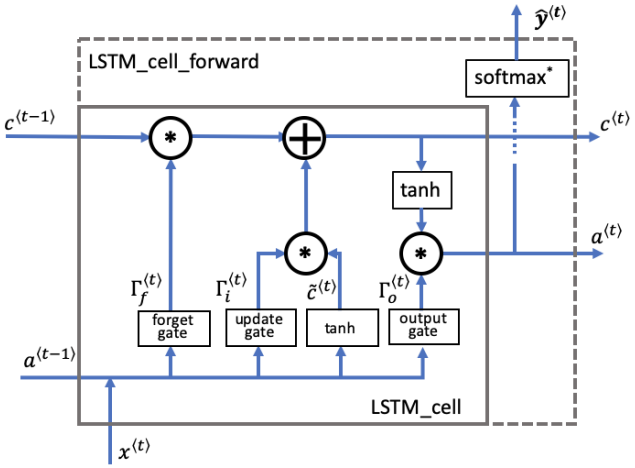

Fig. 1: LSTM unit

The Important part of LSTM is its cell state. The line passing through all the LSTM Units. Cell state acts as a memory unit to carry forward certain aspects of our sentence [3] as we know plural words cause issues for basic RNN, LSTM doesn't suffer from that.

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \quad (1)$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \quad (2)$$

$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \quad (3)$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (5)$$

$$h_t = o_t \odot tanh(c_t) \quad (6)$$

where h_t is hidden state at time t, c_t is the cell state at time t , x_t is the input at time t, h_{t-1} is the hidden state of the layer at time t-1 or the initial hidden state at time o, and i_t, f_t, g_t, o_t are the input, forget, cell and output gates respectively [4] . σ is the sigmoid function, and $\odot$ is the Hadamard product. In the multi layer LSTM, the input $x_t^{(l)}$ of the l^{th} layer (l>=2) is the hidden layer $h_t^{(l-1)}$ of the previous layer multiplied by dropout $\delta_t^{(l-1)}$ where each $\delta_t^{(l-1)}$ is a Bernoulli random variable which is 0 with probability dropout

D. Gated Recurrent Unit

The gated recurrent block (GRU) is a gating mechanism in recurrent neural networks (RNN), similar to the LSTM block, but without the outlet value. GRU (Jun young Chung et al [5]) tries to solve the problem of the vanishing gradient that can occur with standard recurrent neural networks. It can be thought of as a type of LSTM block as both are similar in design and in some cases give similar results. GRU can solve the disappearing gradient problem with the update and reset gate. The update gateway manages information flowing into memory, and the reset gateway contains information flowing out of memory [5]. GRU is a useful mechanism to solve the problem of vanishing gradients in recurrent neural networks. The vanishing gradient problem occurs in machine learning when the gradient becomes vanishingly small, which prevents the weight from changing its value.

1) ***Architecture:*** The architecture of the GRU block is shown below. An GRU unit typically has a forget, update gate and reset gate in addition to the hidden state of the RNN it lacks the output gates. The update gateway and the reset valve are two vectors that decide which information is transmitted to the output.

We can train this to remember information from the past or to delete information unrelated to the forecast. [5]

2) Advantages:

- GRU perform better than LSTM when working with small data sets.
- GRU is Computationally less expensive as compared to LSTM as we have fewer matrices to compute. it Works well with Smaller Datasets.

3) Disadvantages:

- Performs worse in case of larger datasets as compared to LSTM.
- Lesser bleu score than LSTM.

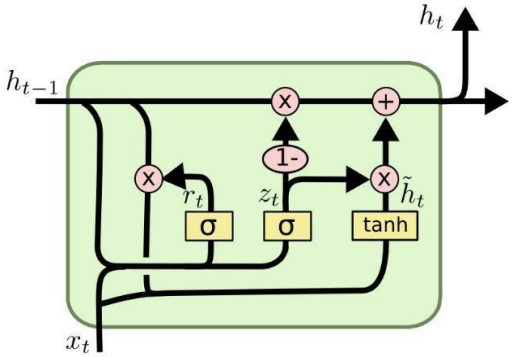

Fig. 2: GRU unit

4) Equations:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \quad (7)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \quad (8)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (9)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (10)$$

E. Encoder Decoder Model with Attention layer

1) Encoder: It is a Stack of a recurrent units (of whichever model being used like LSTM or GRU) which takes element of input sentence or sequence and pass on its information to further layers Equation for hidden states vector [6], [7].

$$h_t = f(W^{hh} h_t - 1 + W^{hx} x_t) \quad (11)$$

This equation denotes final hidden layer vector produced by the encoder this helps in combining all the information from the input and then providing it to decoder for respective prediction or in this case generating text. This vector is directly given as first input layer of the decoder [6], [7]. hidden layer and memory value of each unit is given to the attention layer [2].

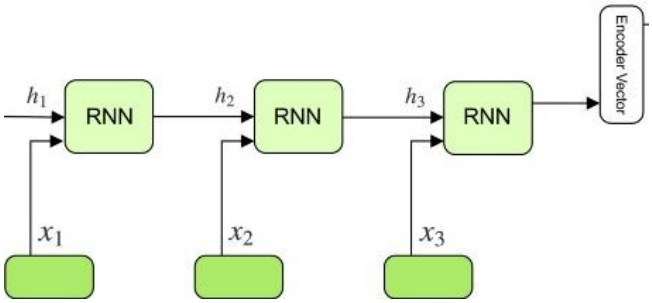

Fig. 3: Encoder in Seq2Seq model

2) Attention Layer: Computes 4 important elements which are score of each encoder state, attention weights using SoftMax function over the aforementioned scores, Context vector by multiplying hidden layer vector and attention weights and concatenate them with previous time step and send this to the decoder along with last hidden layer vector [2]. We have 2 types of attention models which are as follows:

1. **Global Attention layer:** Here all encoder states are considered for obtaining the context vector

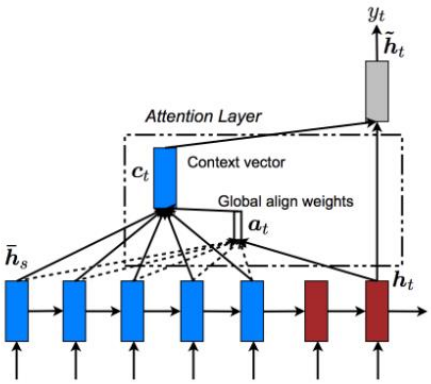

Fig. 4: Global attention layer - l – at each time step t, the model infers a variable-length alignment weight vector at based on the current target state h_t and all source states h_s. A global context vector c_t is then computed as the weighted average, according to at, over all the source state (Minh-Thang Luong et al [2]).

2. Local Attention layer: Global attention has a few drawbacks like it attends all words on the encoder side for all target words which could potentially be computationally
expensive. Here local attention comes into action where we have a small subset of source positions per target word [2]. Equations for attention weights and context vector are referred from Minh-Thang Luong et al [2].

3) Decoder: Decoder is used to generate text in case of Seq2Seq model where it takes the context vector from the attention layer and last hidden vector from the encoder and gives output accordingly.

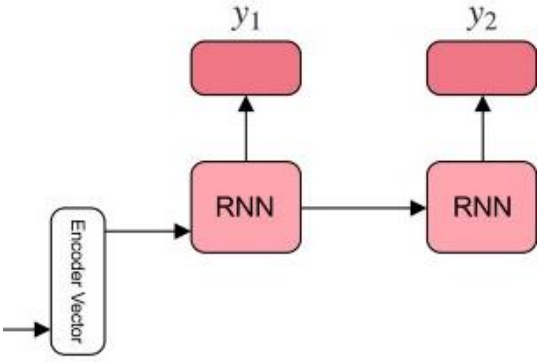

Fig. 5: Decoder in Seq2Seq model

IV. METHODOLOGY

Here we will judge performance of aforementioned algorithms and we will use our model in the mix as well.

Algorithm: Deep Neural Network, Recurrent Neural Network.

Main Technique: We have used the Seq2Seq model with attention layer [2] with GRU for this paper We will be comparing our outputs with paper which has implemented same model on same dataset using LSTM.

frameworks used: We have used various deep learning frameworks like PyTorch for respective implementation.

Optimization technique: For this paper we have used Adam optimization technique to get the best results which we can.

V. CONFIGURATION

We have used 3 different algorithms 2 of which are implemented by use other is from the base paper [9].

TABLE I: Configuration Table

Parameters	*LSTM [9]*	*RNN*	*GRU*
batch size	32	128	128
embedding size	1024	1024	1024
RNN Size	1024	1000	1250
learningrate	0.001	0.0001	0.00001
epochs	50	32	46
minlearningrate	0.0001	0.00001	0.000001
learning rate decay	0.9	0.9	0.9

VI. TRAINING MODEL

We trained the model for 2 configurations as we discussed above using Adam Optimization which can vary the learning
rate using learning rate decay technique [8]. we had around
82000 sentences in Cornell movies dataset after tokenization. We used Bidirectional GRU in encoder side and decoder is made with help of Luong attention layer [2].

VII. HARDWARE SPECIFICATION

Training on google colab for GRU and RNN Basic Seq2seq
and my local machine configuration is stated below

TABLE II: Specification

Hardware	Specification
Processor	Core-i5 6400 @3.3 GHz
RAM	DDR4 16GB @ 2400MHz
GPU	Radeon RX 5700 XT 8GB

VIII. RESULT

Here a test is given from all 3 models after seeing training time and responses GRU is the best option for this dataset to create chatbot but if the dataset is more complex then LSTM is the best choice but work is still being done on GRU so it has scope to improve even further.

Human :hey
Bot :hi
Human :what doing
Bot :sleep
Human :when will you be okay
Bot :just few days
Human :where are you from
Bot :southern California
Human: you want to come with me?
Bot: how is that for you?
Human :how much do you like me
Bot :you are not cold mostly
Human :who am i
Bot :doctor Livingston
Human :what's your hobby
Bot :eat
Human :wow you are best
Bot :I have done
Human :really well
Bot :goodbye

(a) LSTM [9] conversation

```
> hey
Bot: there
> what ypu doing
Bot: there
> when will you be okay
Bot: i m sorry .
> where are you from
Bot: i m sorry .
> you want to come with me
Bot: i m not .
> how much do you like me
Bot: i m sorry .
> who am i
Bot: you re not .
> what about weather in NY
Bot: there
> what's your hobby
Bot: i m sorry .
```

(b) RNN with Seq2Seq

```
> hey
Bot: hey
> what you doing
Bot: i m going to see you
> where are you from
Bot: gosh
> you want to come with me
Bot: no
> how much do you like me
Bot: i don t know
> who am i
Bot: gosh
> what about weather in NY
Bot: right here
> what's your hobby
Bot: gosh
> are you happy?
Bot: i m not sure
> Oh isee
Bot: right here
> bye
Bot: bye
```

(c) GRU

FIG. 6: COMPARISON OF ALL 3 ALGORITHMS

REFERENCES

[1] Peng H., Parikh A. P., Faruqui M., Dhingra B., and Das D. (2019). "Text generation with exemplar-based adaptive decoding," arXiv preprint arXiv:1904.04428.

[2] Luong M.-T., Pham H., and Manning C. D. (2015). "Effective approaches to attention-based neural machine translation," arXiv preprint arXiv:1508.04025.

[3] Sherstinsky A. (2020). "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," Physica D: Nonlinear Phenomena, vol. 404, p. 132306.

[4] Hochreiter S. and Schmidhuber J. (1997). "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780.

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

[6] Sutskever I., Vinyals O., and Le Q. V. (2014). "Sequence to sequence learning with neural networks," arXiv preprint arXiv:1409.3215.

[7] Cho K., Van Merri¨enboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., and Bengio Y. (2014). "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078.

[8] Kingma D. P. and Ba J. (2014). "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980.

[9] Sojasingarayar A. (2020). "Seq2seq ai chatbot with attention mechanism," arXiv preprint arXiv:2006.02767.

[10] Graves A., Fern´andez S., and Schmidhuber J. (2005). "Bidirectional lstm networks for improved phoneme classification and recognition," in International conference on artificial neural networks. Springer, pp. 799–804.

[11] Sundermeyer Martin, Schl¨uter Ralf, and Ney Hermann (2012). "LSTM
Neural Networks for Language Modeling," In Interspeech, pages 194–197.

[12] Bengio Y., Ducharme Rejean, Vincent Pascal, and Janvin Christian (2000) . "A neural probabilistic language model," 13th International Conference on Neural Information Processing Systems, Pages 893–899.

[13] Bengio Y., Boulanger-Lewandowski N., and Pascanu R. (2013). "Advances in optimizing recurrent networks,", DOI: 10.1109/ICASSP.2013.6639349

[14] Csaky R. (2019). "Deep learning based chatbot models," arXiv reprint arXiv:1908.08835.