International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Exploring the Efficacy of LSTM Networks in Machine Translation: A Survey of Techniques and Applications

Neha Vaswani[1*], Krupa Mehta[2]

Submitted: 02/05/2024 **Revised**: 15/06/2024 **Accepted**: 22/06/2024

Abstract: Machine Translation (MT) has significantly advanced with the advent of neural network architectures, among which Long Short-Term Memory (LSTM) networks have garnered substantial attention. This paper presents a comprehensive survey on the LSTM networks in MT tasks. This Paper delve into the architecture of MLPs, RNNs and LSTMs, advantages of LSTMs over traditional recurrent neural networks (RNNs), and their suitability for capturing long-range dependencies. Furthermore, this paper examines various approaches of gates adopted in leveraging LSTM networks for MT. This work emphasizes the benefits, drawbacks, and possible directions for further research in this field through a critical review of the body of existing work.

Keywords: *Machine Translation, Neural Machine Translation, Recurrent Neural Network (RNN), Multi-Layered Perceptron, Long Short-term Memory (LSTM)*

1 Introduction

Language translation has been a long and crucial task in natural language processing (NLP) with applications ranging from global communication to cross-border commerce. The automatic translation of text between languages, known as machine translation, has advanced significantly in the last several years, primarily due to the emergence of neural network-based methods. Traditional statistical machine translation (SMT) systems relied on handcrafted features and linguistic rules, often struggling to capture the problems of human language. The advent of neural network-based approaches, particularly Recurrent Neural Networks (RNNs), has revolutionized the field, enabling more accurate and fluent translations. This paper provides an overview of RNNs, emphasizing their architecture, functioning, and training process. We discuss how RNNs handle sequential data by maintaining hidden states and updating them recursively for each input token. The difficulties in training RNNs—like the vanishing gradient problem—are also emphasized. This paper also introduces solutions like Long Short-Term Memory (LSTM) networks. Among all the approaches MT for Language Translation, Because of its capacity to represent long-range relationships and alleviate the vanishing gradient issue that standard RNNs sometimes face, LSTM networks have become a popular option. Our goal in this research is to present an extensive overview of LSTM network applications in machine translation jobs.

[1*]*Assistant Professor, GLS University, Ahmedabad, Gujrat, India. Email: neha.vaswani@glsuniversity.ac.in*
[2]*Assistant Professor, GLS University, Ahmedabad, Gujrat, India. Email: Krupa.mehta@glsuniversity.ac.in*
**** Corresponding Author:*** *Neha Vaswani*
Email: neha.vaswani@glsuniversity.ac.in

2 Neural Machine Translation (NMT)

Neural machine translation has demonstrated promising results as a deep learning technique in recent years.

One neural network is the foundation of a neural machine translation model. Translating between languages is most frequently done using a sequence-to-sequence approach. Using encoders and decoders, Sequence2Sequence converts variable-length sources into fixed-length targets by teaching the encoders how to do it. The decoders then determine the target sentences' length. Cho et al. recommend utilizing a recurrent neural network encoder-decoder to complete the encoding and decoding.

Once the input sentence has been encoded using an RNN model, the encoder uses the RNN decoder to decode the final hidden state, which is the target sentence. The RNN can handle data sequences in a manner akin to how text and video are based on word and frame sequences, respectively (Almansor, 2018).

This is because the Recurrent Neural Network considers the extensive history of prior observations and stores the crucial information in an internal state, or memory unit, where it is modifiable at each stage. For example, seven words in a sentence will result in seven-time steps.

The target language's output sequence (y1,⋯, yt) is read by the decoder in the sequence-to-sequence paradigm, while the source language's input sequence (x1,⋯, xt) is read by the encoder. Many NLP tasks, such as text translation, speech recognition, Q&A systems, and text categorization, have been handled by this approach.

3 Understanding Multi-Layered Perceptron (MLP)

An artificial neural network (ANN) known as a Multilayer Perceptron (MLP) is made up of several layers of neurons, each completely connected to the layer above it. MLPs are frequently employed to solve supervised learning problems because they can learn intricate input-to-output mappings.

3.1 Key Concepts of MLP

Neurons

The basic unit of a neural network, inspired by biological neurons. Each neuron receives inputs, processes them through a weighted sum, applies an activation function, and outputs a signal.

Layers

A hidden layer (or layers), an output layer, and an input layer make up an MLP.

- Input Layer: Takes input features.

- Hidden Layers: Execute intermediary calculations. More hidden layers allow the network to model more complex functions.

- Output Layer: Produces the final output of the network.

Weights

Parameters that are changed during network training in order to reduce error. Every neuronal connection has a weight attached to it.

Biases

Additional parameters that allow the activation functions to be shifted, improving the flexibility of the model.

Mechanisms of Activation

Each neuron's output can have functions applied to it to create non-linearity, which allows the network to represent intricate patterns. Sig-moid, Tanh, and ReLU are examples of common activation functions.

Forward Propagation

The process of calculating the output of the network given an input. Up until the final output is generated, the output of each layer serves as the input for the layer after it.

Loss(Diminished) Function

A function that quantifies the discrepancy between the target values and the network's forecast. Common loss functions for classification tasks are Cross-Entropy Loss and Mean Squared Error (MSE) for regression tasks.

Backpropagation

An algorithm with which the MLP is trained. Gradient descent can be used to update the weights because it computes the gradient of the loss function with respect to each weight by employing the chain rule.

Training Process

- Initialization: Weights are initialized randomly or using specific strategies.

- Forward Pass: The network processes input data in order to produce output data.

- Loss Computation: The loss function is evaluated.

- Backward Pass: Backpropagation is used to calculate the gradients of the loss with respect to each weight.

- Weight Update: Using an optimization technique like Adam or stochastic gradient descent (SGD), weights are modified to minimize the loss.

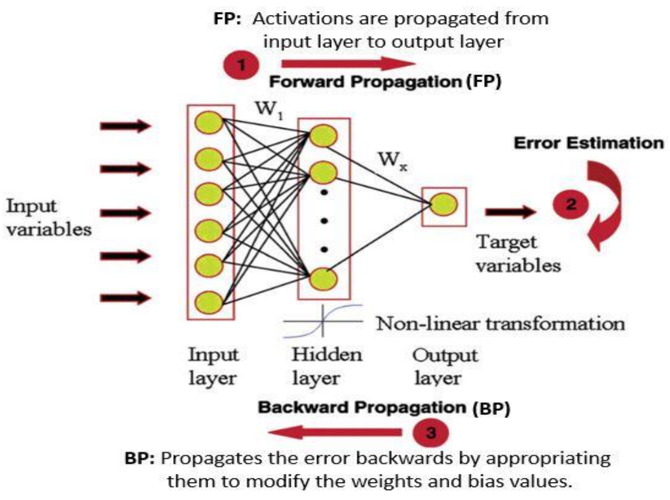

FP: Activations are propagated from input layer to output layer
1
Forward Propagation (FP)
W₁
Wₓ
Error Estimation
2
Input variables
Target variables
Non-linear transformation
Input layer
Hidden layer
Output layer
Backward Propagation (BP)
3
BP: Propagates the error backwards by appropriating them to modify the weights and bias values.

Fig 1 Perceptron with Multiple Layers (Singhal, 2020a)

Working Process

The three layers that make up an MLP are an input, a hidden layer, and an output layer. Each node in the following layer is given a certain weight, which connects the nodes to one another (Singhal, 2020a). A feed-forward neural network called a multilayer perceptron train using both forward propagation (FP) and back propagation (BP) methods. The following is the MLP illustration (Singhal, 2020a).

4 Understanding Recurrent Neural Networks (RNN)

One popular type of artificial neural network is the recurrent neural network (RNN). The definition of "recurrent" is "each element in the sequence has an output that is determined by the calculations made earlier".

The human brain is tenacious, so when people read an essay, they are able to comprehend the entire essay's context and every word. They also wouldn't start over by discarding the knowledge that has been ingrained in their minds. Given that it seems like a significant drawback, this cannot be accomplished with traditional neural networks. Classifying the events in the movie at every point, for example, is possible for humans. Conventional neural networks may have trouble updating later scenes in the movie based on their understanding of prior events. Recurrent Neural Networks are included here to help tackle this problem. According to Colah (2015), networks with loops allow for the persistence of data, just like in RNN.

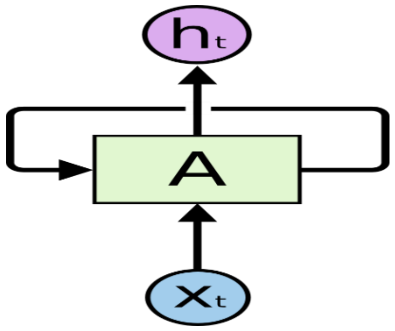


Figure 2 RNN's with loops (colah, 2015)

The input "xt" in the above picture feeds into the "A" component of the neural network, producing the output value "ht." Additionally, a loop in this RNN enables information to flow from one network to another.

4.1 Simple RNN’s

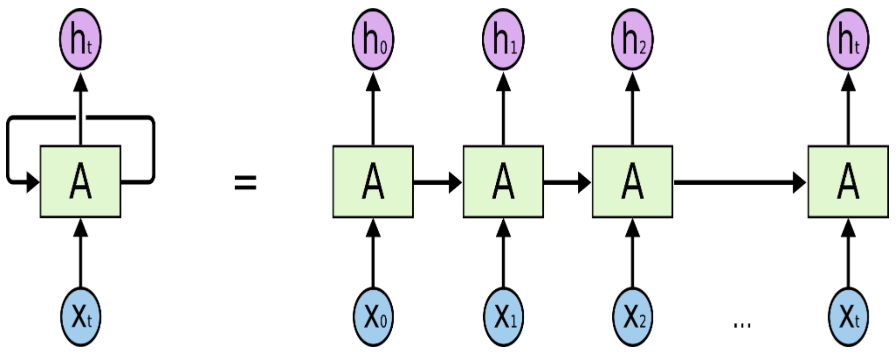


Fig 3 Unrolled RNN (colah, 2015)

Loops make the Recurrent Neural Network illustration look a little strange. Unroll this RNN loop into several copies of the same network, though, to gain a deeper understanding of it. Every network node will transmit data to its successor.

Given that lists and sequences are the default architecture that neural networks employ when dealing with this type of data, the networks of these unrolled recurrent neural networks are closely related to each other. In recent years, A great deal of problems, including language modeling, speech recognition, image classification, picture captioning, and language translation, have been resolved with surprising efficacy by using RNN implementation.

4.2 Vanishing Gradient & Back Propagation through time (BPTT) in RNN

Recurrent Back propagation through time (BPTT) is used to train recurrent neural networks. To signify that the approach is to be applied to a temporal neural network

(RNN), the phrase "through time" is added to the word "backpropagation". BPTT seeks to identify a point with the minimum amount of error. The weights can be changed to achieve this. We refer to this complete procedure as Gradient Descent. The chain rule, derivatives, and partial derivatives are used to determine the gradient. However, there are certain challenges with backpropagation through time when determining the long-term dependencies. The challenges posed by long-term dependencies should theoretically be resolved by including stacked RNNs. However, adding additional RNNs practically leads to a vanishing gradient issue. Since the gradient in BPTT would be so tiny, weights would not be able to change, therefore preventing neural networks from learning.

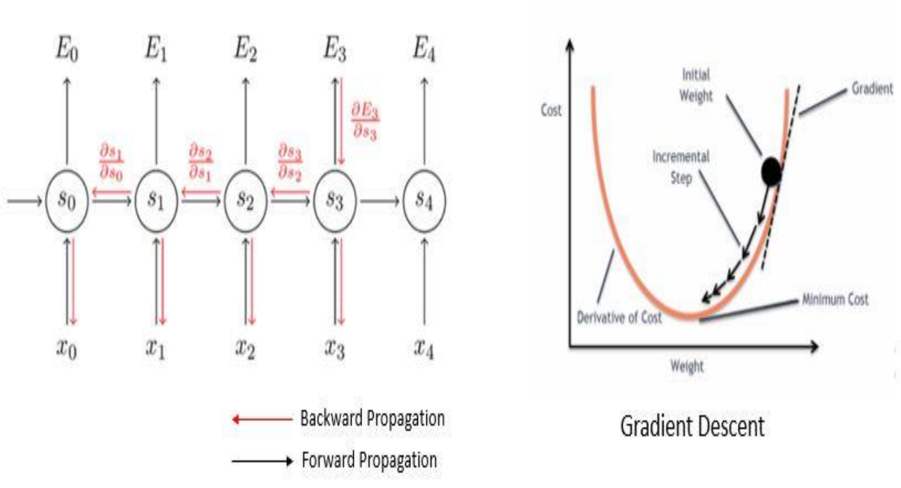


Fig 4 BPTT & Vanishing Gradient (Singhal, 2020a)

4.3 The Problem of Long-Term Dependencies in RNN

One of its appealing aspects is the potential for RNNs to be able to connect previous information to the current task—for instance, by using previous video frames to better understand the current frame. If RNNs could achieve this, it would be incredibly beneficial. But can they really? It fluctuates.

Sometimes a quick check at recent data is all that is required to finish the present assignment. Consider a language model that, given the words that come before it, predicts the word that will come next. The final word in "the clouds are in the sky" is obvious to guess—we don't need any other context to figure it out. The word that follows will definitely be "sky." When there is a short distance between the relevant information and the location where it is needed, RNNs can learn to use the prior knowledge.

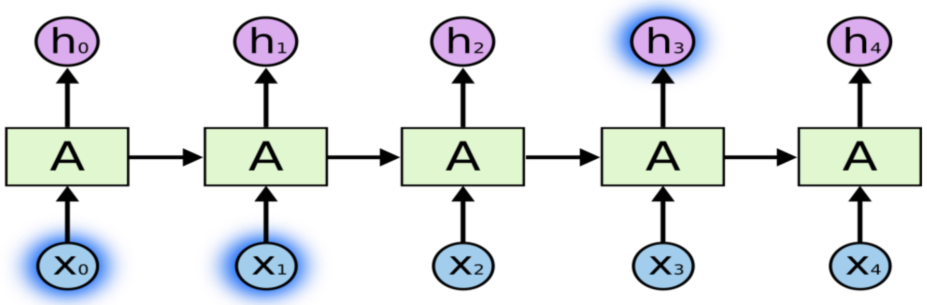


Fig 5 In small gap RNN learn to use the past information

However, in other situations, additional background is required. The last sentence in the text, "I grew up in France... I speak fluent French," might be worth predicting. The following word is most likely the name of a language, according to recent evidence, but in order to determine which language it is, we need to go further back in time and examine the French context. There is every chance that there will be a significant discrepancy between the pertinent information and the point at which it is required. Regretfully, RNNs lose their ability to connect the previously acquired data as that gap widens.

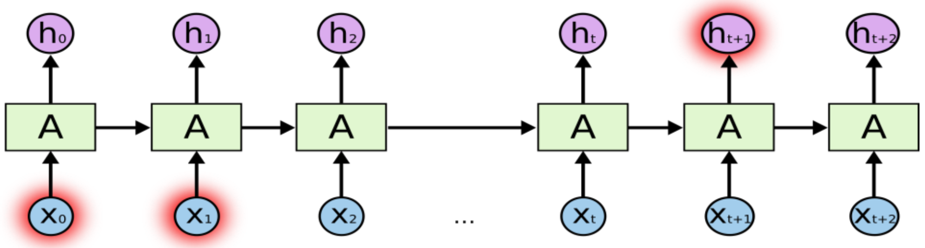


Figure 6 As the distance widens, RNN is unable to connect the previously acquired data.

Such "long-term dependencies" can theoretically be handled by RNNs with complete ease. Toy difficulties of this kind could be solved by a human carefully choosing parameters. Regretfully, it doesn't appear that RNNs can learn them in practice. Fortunately, this is not an issue with LSTMs!

5 Introduction to Long Short-term Memory (LSTM)

A specific kind of RNN designed to understand long-term dependencies is called an LSTM. Since their introduction by Hochreiter and Schmidhuber in 1997, LSTMs have been enhanced and made more well known by several researchers. Their outstanding performance in a range of applications has led to their widespread use today. Long-term dependency is a challenge that LSTMs are specifically made to address. Rather than struggling to learn, they inherently retain information over extended periods. The structure of any recurrent neural network is made up of neural network modules that repeat. These repeating modules are commonly found in conventional RNNs, where they are represented by a single tanh layer.

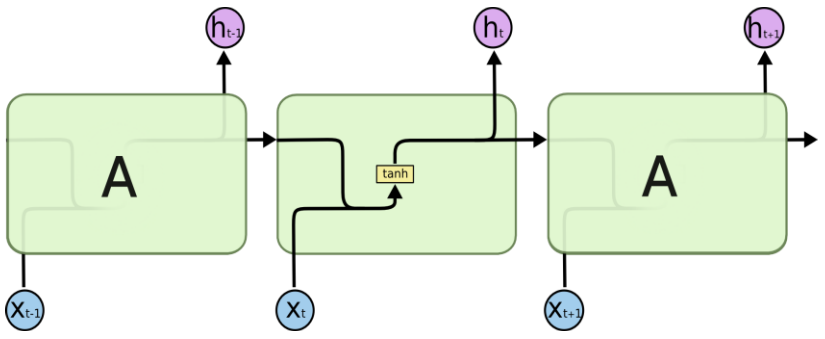


Fig 7 A conventional RNN has a single layer in its repeating module.

Although LSTMs' repeating module is organized differently, they nonetheless have this chain-like structure.

Neural network layers unnecessary space added are not one, but four, and each of them interacts with the others in a different way.

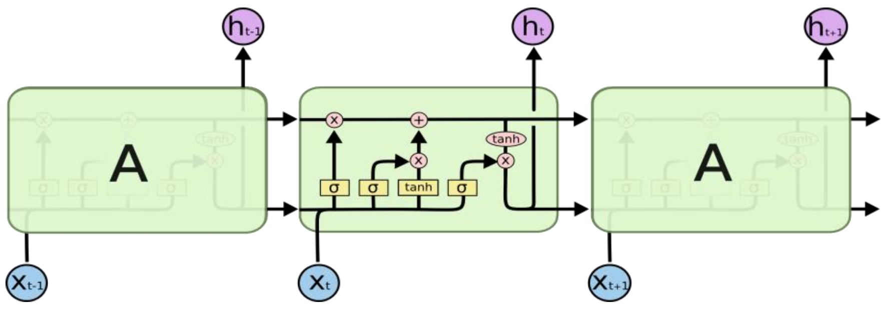


Fig 8 Four interacting layers make up an LSTM's repeating module.

The diagram below shows a complete vector, with each line joining the inputs of other nodes to the outputs of one node. The pink circles indicate pointwise operations like vector addition, and the yellow boxes represent learned neural network layers. Lines that merge indicate concatenation; lines that fork indicate that a line's content is copied and transmitted to many destinations.

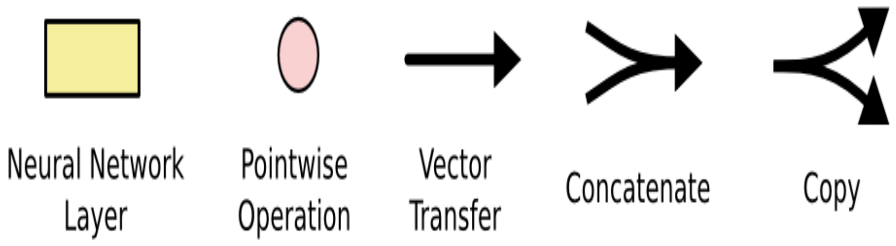


Fig 9 Working of a Complete Vector

5.1 LSTM's Fundamental Concepts

The horizontal line at the top of the figure, representing the cell state, is crucial to LSTMs. For LSTMs, the cell state, shown by the horizontal line at the top of the picture, is essential. The status of a cell is comparable to a conveyor belt. It goes straight down the entire chain with very little in the way of linear interactions. It's easy for information to pass through it unchanged.

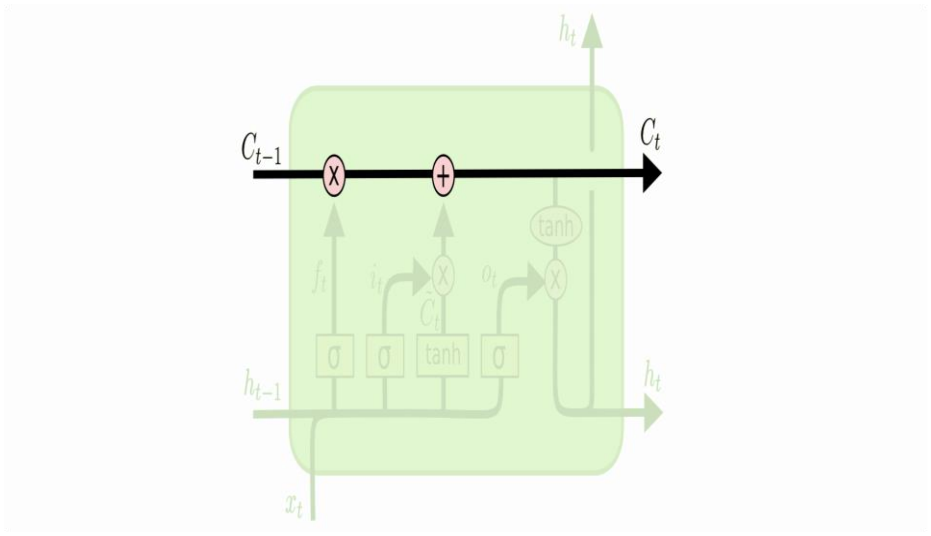


Fig 10 The horizontal line at the top of the figure, representing the cell state.

The LSTM can add or delete information from the cell state by carefully manipulating structures called gates. Information can be optionally allowed via gates. Their construction consists of a pointwise multiplication operation and a sigmoid neural net layer.

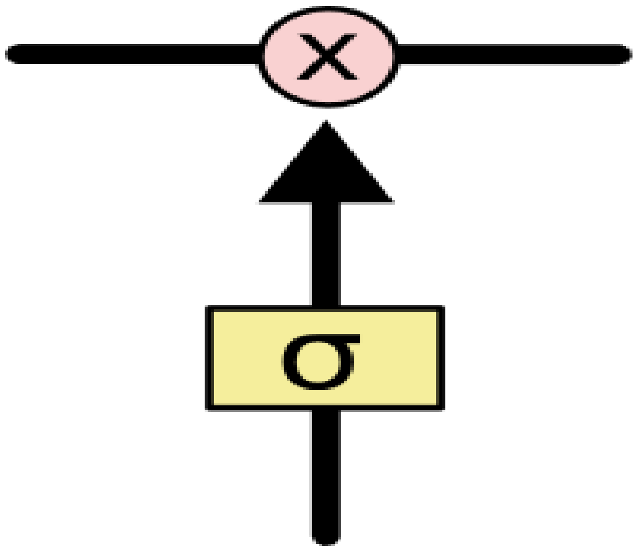


Fig 11 Concept of Gate in LSTMs

In Long Short-Term Memory (LSTM) networks, the tanh and sigmoid activation functions play crucial roles in managing and regulating the flow of information through the network. These activation functions are used in the gates and cell states of the LSTM to enable effective learning of long-term dependencies.

Tanh Activation Function

Tanh is a non-linear activation function. The network's values are governed by the Tanh function, which ranges in value from -1 to 1. To stop information from fading, a function that can hold onto its second derivative for an extended amount of time is required. In certain situations, values can grow to huge proportions, which further reduces their significance. Because of this tanh activation function, the number 5 in the figure below remains inside the boundaries (Singhal, 2020b)

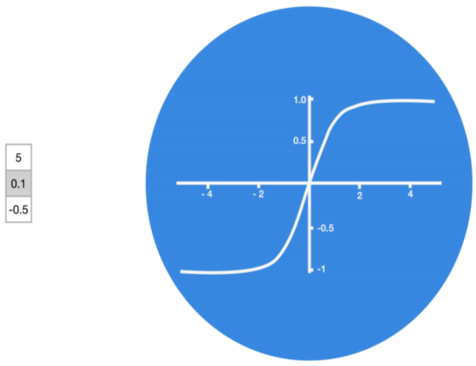


Figure 12 Tanh Activation Fuction (Singhal, 2020b)

Activation Function of Sigmoid

The activation function of sigmoid is another non-linear function. Within the gate lies the sigmoid activation function. According to Singhal (2020b), the sigmoid function has values between 0 and 1. The network will benefit from this by updating or remembering the information. If their product is 0, the information is lost, and if the value is 1, it remains untouched. In doing so, the network is better able to decide which data should be discarded and which should be remembered (Singhal, 2020b).

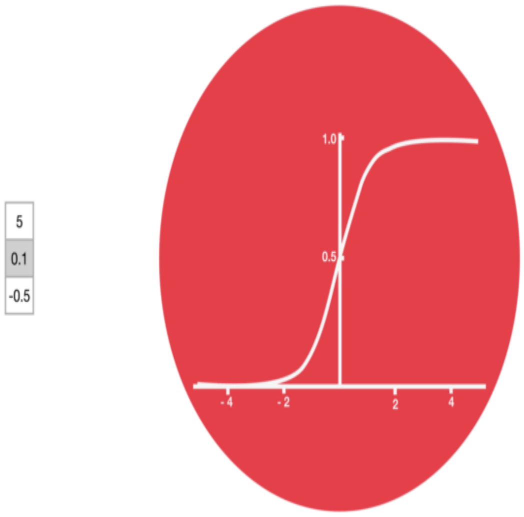


Fig 13 Sigmoid Action Functionn (Singhal, 2020b)

6 Gate Mechanism in LSTM

The forget gate, input gate, and output gate are the three separate gates found inside an LSTM cell. Below is an illustration of an LSTM cell.

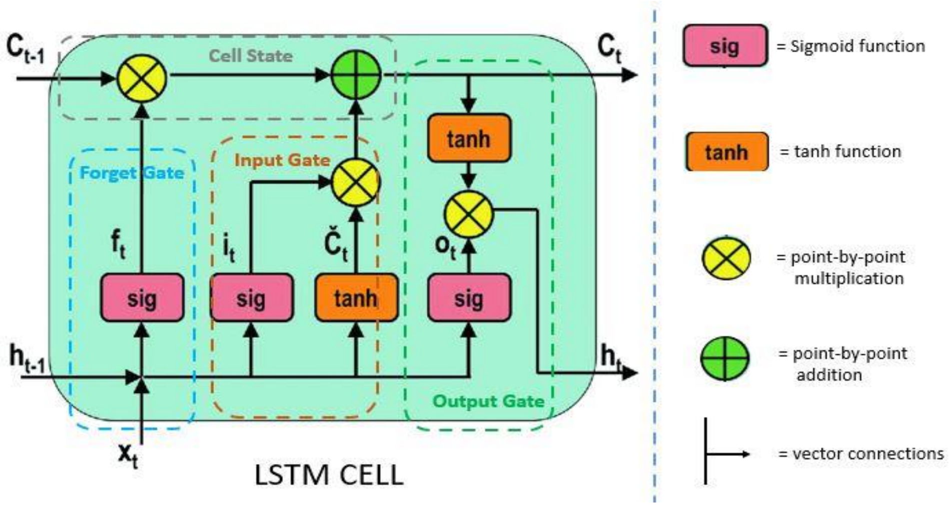
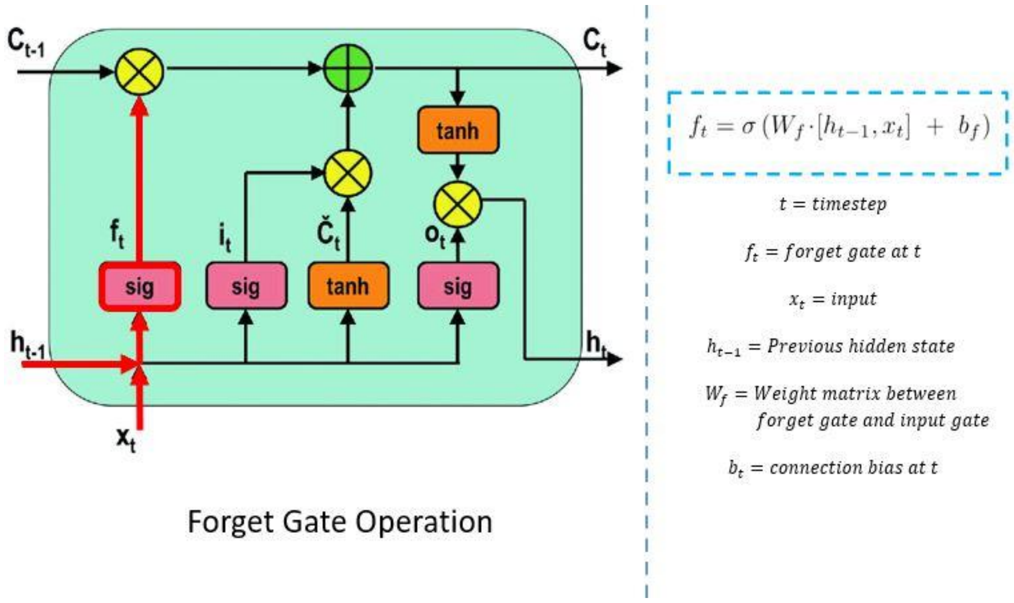


Fig 14 LSTM Cell (Singhal, 2020b)

The Vanishing Gradient problem, one of the shortcomings of simple RNNs, led to the development of LSTM. LSTM will employ a gated mechanism, which has the ability to regulate memorization, to address this problem. The information can be written, read, and saved in LSTMs thanks to the opening and closing gates. These gates are used to achieve element-wise multiplication for sigmoid ranges with values between 0 and 1. Because this analog has a differentiable character, it is suitable for back propagation. In order to protect and control the cell state, an LSTM has three gates.

6.1 Forget Gate

The first gate in an LSTM cell is the forget gate. This gate will determine whether or not the required information is taken into consideration. A sigmoid function is used to process the input data (X(t)) and the hidden data (h(t-1). After then, values between 0 and 1 will be reduced by the sigmoid activation function. When the information is needed, the sigmoid function will release a value that is closer to 1 (Singhal, 2020b). After then, point-by-point multiplication will be performed using the forget gate value at time step f(t), which will be represented as (×). Below is a diagram of the Forget Gate.



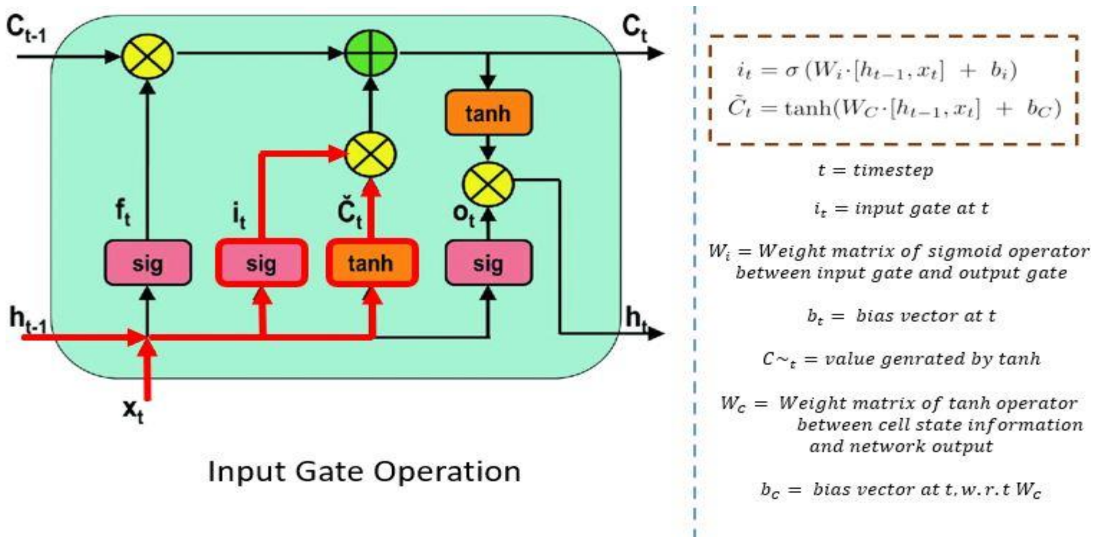
C_{t-1}
C_t
tanh
f_t
i_t
$\check{C}_t$
o_t
sig
h_{t-1}
h_t
x_t
Forget Gate Operation
$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$
t = timestep
f_t = forget gate at t
x_t = input
h_{t-1} = Previous hidden state
W_f = Weight matrix between forget gate and input gate
b_t = connection bias at t

Fig 15 Forget Gate of LSTM (Singhal, 2020b)

6.2 Input Gate

The LSTM cell's state will change as a result of the input gate's subsequent actions. First, the hidden state data (h(t-1)) and the input data (Xt) are sent to the second sigmoid activation function. The output values vary between 0 and 1. Singhal (2020b) asserts that if the value is 1, the data is not significant. In the event that it is 0, the data is significant. The current state (Xt) and the concealed state h(t-1) will then be subjected to the tanh activation function. The tanh operator builds a vector of ţ(t), whose values range from -1 to 1, in order to manage the network.

These activation functions result in values that can be multiplied point-by-point. The notation will look like this (×). Below is an illustration of this input gate.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$t = timestep$

$i_t = input\ gate\ at\ t$

$W_i = Weight\ matrix\ of\ sigmoid\ operator\ between\ input\ gate\ and\ output\ gate$

$b_t = \ bias\ vector\ at\ t$

$C\sim_t = value\ genrated\ by\ tanh$

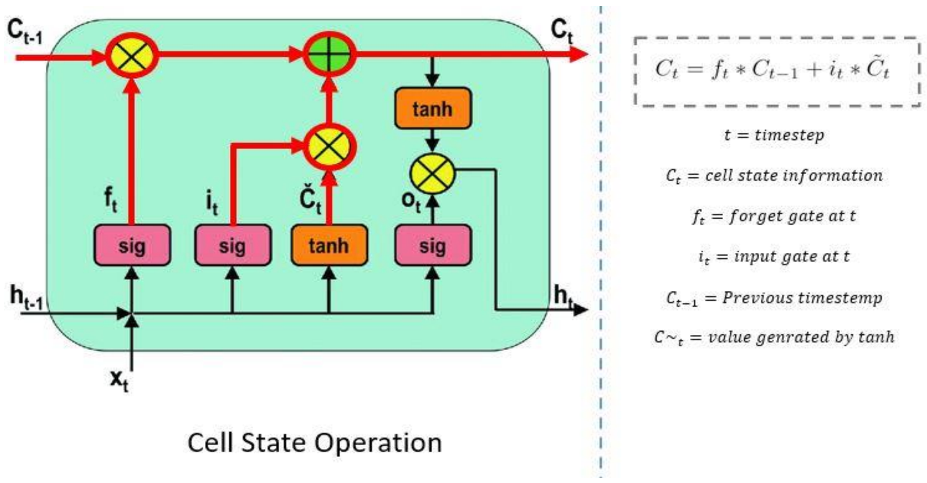
$W_c = \ Weight\ matrix\ of\ tanh\ operator\ between\ cell\ state\ information\ and\ network\ output$

$b_c = \ bias\ vector\ at\ t.w.r.t\ W_c$

Fig 16 Input Gate of LSTM (Singhal, 2020b)

Cell State Operation in Input Gate

The input gate and forget gate supply enough data for the network. The pertinent data is then transferred from the new state to the cell state as part of the decision-making process. The forget gate cell state C(t-1) multiplies the forget vector f(t). If the result is zero, these values are eliminated from the cell state (Singhal, 2020b). The current cell state will then be changed into a new cell state C(t) by performing a point-by-point addition operation (+) using the output value of the input vector i(t) (Singhal, 2020b). An example of the cell state can be seen below.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$t = timestep$

$C_t = cell\ state\ information$

$f_t = forget\ gate\ at\ t$

$i_t = input\ gate\ at\ t$

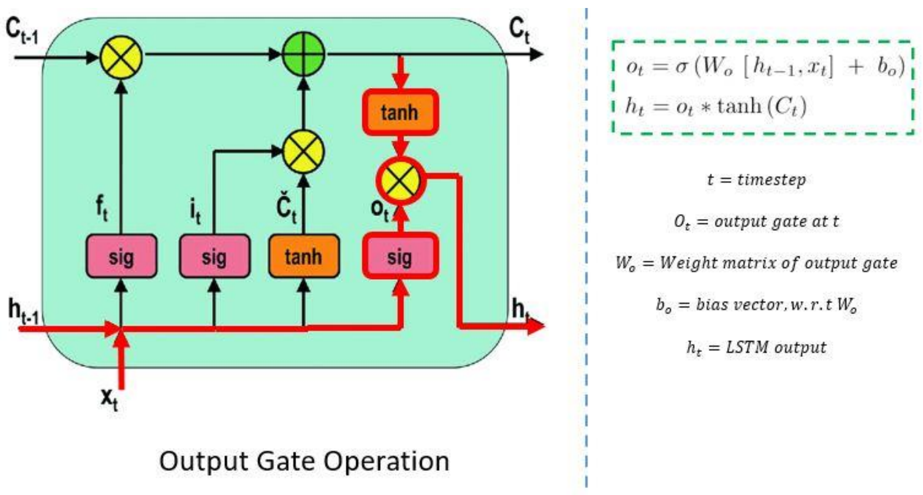
$C_{t-1} = Previous\ timestemp$

$C\sim_t = value\ genrated\ by\ tanh$

Fig 17 Input gate Cell state of LSTM (Singhal, 2020b)

6.3 Output Gate

The value of the next concealed state is set by the output gate. The history of earlier input data was stored in this hidden state. The input data (X(t)) and the previous hidden state data (h(t-1)) are first sent to the third sigmoid function. The new cell state created by point-by-point addition in the input gate action is given to the tanh activation function. The output values generated by the sigmoid and tanh functions are multiplied point-by-point. The information that should be in the hidden state will depend on the network's final value, or h(t), which is the LSTM output. It predicts the word using this hidden state, h(t). Next, the new cell state C(t) and the new hidden state h(t) are moved to the subsequent time step. Below is an illustration of this output gate.



C_{t-1}
C_t
tanh
f_t
i_t
$\check{C}_t$
o_t
sig
sig
tanh
sig
h_{t-1}
h_t
x_t
Output Gate Operation
$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$
$h_t = o_t * \tanh\left(C_t\right)$
$t = timestep$
$O_t = output\ gate\ at\ t$
$W_o = Weight\ matrix\ of\ output\ gate$
$b_o = bias\ vector, w.r.t\ W_o$
$h_t = LSTM\ output$

Fig 18 Output Gate in LSTM (Singhal, 2020b)

6.4 Challenges and Limitations of LSTM

Despite their efficacy, LSTM-based MT systems encounter certain challenges and limitations. It has issues such as computational complexity, data sparsity, domain adaptation, and the tendency for LSTM networks to overfit on small datasets. Additionally, it addresses concerns related to model interpretability and the generation of fluent and contextually appropriate translations.

7 Conclusion

In conclusion, machine translation (MT) has come a long way since the development of neural network topologies, especially Long Short-Term Memory (LSTM) networks. This thorough analysis has brought to light the distinct benefits of Long Short-Term Memory (LSTM) networks over conventional Recurrent Neural Networks (RNNs), particularly with regard to their capacity to detect long-range dependencies and alleviate the vanishing gradient issue. We have demonstrated the higher performance and suitability of LSTMs for MT tasks by analyzing the architecture and operation of RNNs, Multilayer Perceptrons (MLPs), and LSTMs. The paper also delved into the various gating mechanisms within LSTMs that facilitate efficient information retention and processing, making them ideal for handling the complexities of language translation. Despite their effectiveness, LSTM-based MT systems face challenges such as computational complexity, data sparsity, and potential overfitting on small datasets. Addressing these issues is crucial for further enhancing their performance and applicability.

Through critical analysis of existing literature, this survey has underscored both the strengths and limitations of LSTM networks in MT. Future research should focus on improving the computational efficiency, robustness, and scalability of LSTM-based models. Additionally, exploring hybrid models and integrating LSTMs with other advanced neural network architectures could pave the way for more accurate and contextually appropriate translations.

Ultimately, LSTMs represent a powerful tool in the arsenal of neural machine translation, with significant potential for further advancements and applications in natural language processing.

References

[1] Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In IEEE 1st International Conference on Neural Networks, San Diego, volume 2, pages 609-618

[2] Marino, D. L., Amarasinghe, K., and Manic, M., Building energy load forecasting using deep neural networks. in IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, pp. 7046– 7051.

[3] Cherrier, N., Castaings, T., and Boulch, A., Deep sequence-to-sequence neural networks for ionospheric activity map prediction. in International Conference on Neural Information Processing, 2017, pp. 545–555.

[4] Lindemann, B., Jazdi, N., and Weyrich, M., Detektion von Anomalien zur Qualitätssicherung basierend auf Sequence-to-Sequence LSTM Netzen. atAutomatisierungstechnik, vol. 67, no. 12, 2019, pp. 1058–1068.

[5] Du, S., Li, T., and Horng, S.-J., Time Series Forecasting Using Sequenceto-Sequence Deep Learning Framework. in 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), 2018, pp. 171–176.

[6] Gu, J., Lu, Z., Li, H., and Li, V. O. K., Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393, 2016.

[7] Jaitly, N., Le, Q. V., Vinyals, O., Sutskever, I., Sussillo, D., and Bengio, S., An online sequence-to-sequence model using partial conditioning. in Advances in Neural Information Processing Systems, 2016, pp. 5067– 5075.

[8] Cinar, Y. G., Mirisaee, H., Goswami, P., Gaussier, E., Aït-Bachir, A., and Strijov, V., Position-based content attention for time series forecasting with sequence-to-sequence rnns. in International Conference on Neural Information Processing, 2017, pp. 533–544.

[9] Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).

[10] Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999).

[11] Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010).

[12] LNCS Homepage, http://www.springer.com/lncs, last accessed 2016/11/21.

[13] ALMANSOR, E. H. 2018. Translating Arabic as low resource language using distribution representation and neural machine translation models.

[14] ANJU, E. & MANOJ KUMAR, K. 2014. Malayalam to English machine translation: An EBMT system. IOSR Journal of Engineering (IOSRJEN), 4, 18-23.

[15] BIGALHI94 2022. Language-translation-with-transformer-model. BROWNLEE, J. 2017. What Are Word Embeddings for Text? Available from: https://machinelearningmastery.com/what-are-word-embeddings/.

[16] COLAH. 2015. Understanding LSTM Networks. Available from: https://colah.github.io/posts/2015-08- Understanding-LSTMs/.

[17] GARG, A. & AGARWAL, M. 2018. Machine translation: a literature review. arXiv preprint arXiv:1901.01122.

[18] KHAN, N. J., ANWAR, W. & DURRANI, N. 2017. Machine translation approaches and survey for Indian languages. arXiv preprint arXiv:1701.04290.

[19] KLAPPENBACH, A. 2022. The 12 most spoken languages in the world [Online]. Available: https://blog.busuu.com/most-spoken-languages-in-the-world/ [Accessed].

[20] MOREL, R. 2019. Why Do We Need Translation? [Online]. Available: https://lighthouseonline.com/blog-en/why-do-we-need-translation/ [Accessed].

[21] MOSES-SMT 2019. Nonbreaking prefixes.

[22] NGUYEN, T. T. 2019. Machine translation with transformers.

[23] REDDIT 2017. Question about Positional Encodings used in "Attention is all you need" paper.

[24] SCIONOFTECH 2019. English-Telugu-Bilingual-Sentence-Pairs.

[25] SINGHAL, G. 2020a. Getting Started with RNN. Available from: https://www.pluralsight.com/guides/getting-started-with-rnn.

[26] SINGHAL, G. 2020b. Introduction to LSTM Units in RNN. Available from: https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn.

[27] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł. & POLOSUKHIN, I. 2017. Attention is all you need. Advances in neural information processing systems, 30.

[28] WEAVER, W. Translation. Proceedings of the Conference on Mechanical Translation, 1952.