

Introduction to Long Short-Term Memory (LSTM)



Archit Saxena · Follow

Published in Analytics Vidhya · 4 min read · Jan 18, 2023



527



3

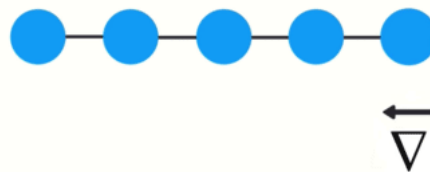


In my previous article on [Recurrent Neural Networks \(RNNs\)](#), I discussed RNNs and how they work. Towards the end of the article, the limitations of RNNs were discussed. To refresh our memory, let's quickly touch upon the main limitation of RNNs and understand the need for modifications of vanilla RNNs.

The **main limitation** of RNNs is that RNNs can't remember very long sequences and get into the problem of **vanishing gradient**.

What is the vanishing gradient problem?

The gradients of the loss function in neural networks approach zero when more layers with certain activation functions are added, making the network difficult to train.



Vanishing gradient; Source: [Medium](#)

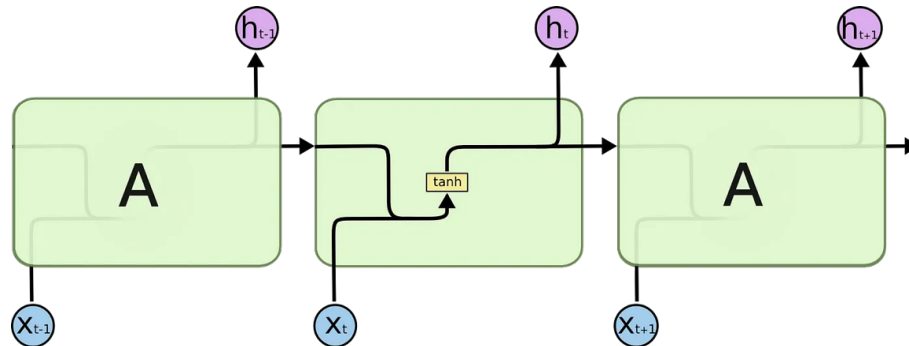
Long Short-Term Memory (LSTM)

LSTMs come to the rescue to solve the vanishing gradient problem. It does so by ignoring (forgetting) useless data/information in the network. The LSTM will forget the data if there is no useful information from other inputs (prior sentence words). When new information comes, the network determines which information to be overlooked and which to be remembered.

LSTM Architecture

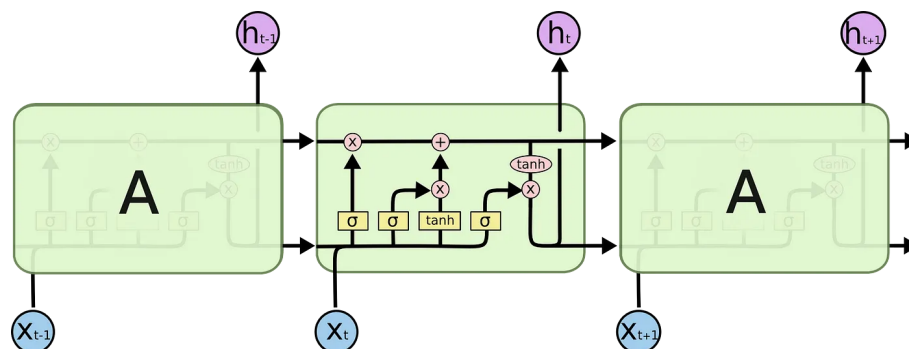
Let's look into the difference between RNNs and LSTMs.

In RNNs, we have a very simple structure with a single activation function (\tanh).

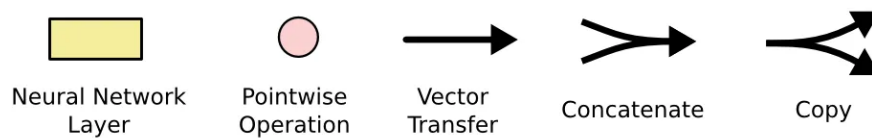


RNN network; Source: [colah's blog](#)

In LSTMs, instead of just a simple network with a single activation function, we have multiple components, giving power to the network to forget and remember information.



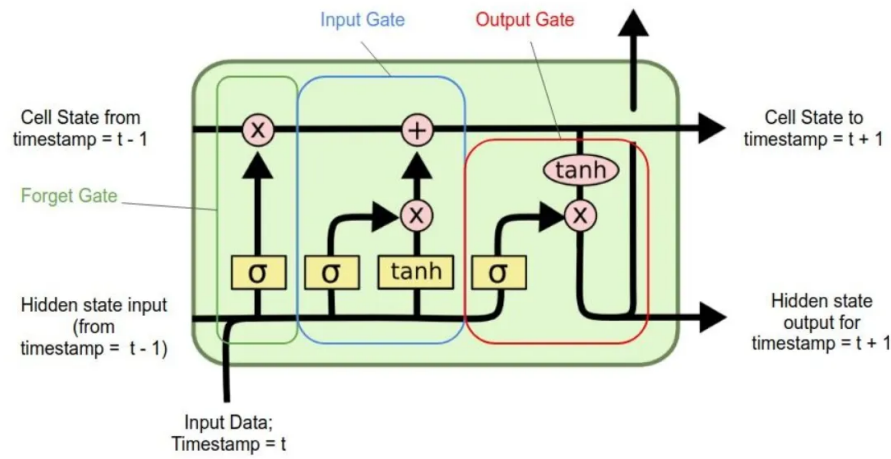
LSM network; Source: [colah's blog](#)



Notations used

LSTMs have 4 different components, namely

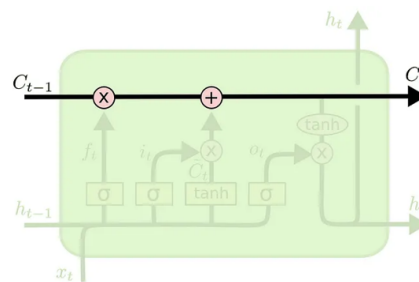
1. Cell state (Memory cell)
2. Forget gate
3. Input gate
4. Output gate

LSTM components; Source: [Turing's blog](#)

Let's understand these components, one by one.

1. Cell State (Memory cell)

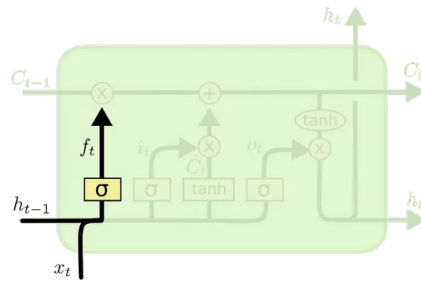
It is the first component of LSTM which runs through the entire LSTM unit. It kind of can be thought of as a conveyor belt.

LSTM Cell State; Source: [colah's blog](#)

This cell state is responsible for remembering and forgetting. This is based on the context of the input. This means that some of the previous information should be remembered while some of them should be forgotten and some of the new information should be added to the memory. The first operation (X) is the pointwise operation which is nothing but multiplying the cell state by an array of $[-1, 0, 1]$. The information multiplied by 0 will be forgotten by the LSTM. Another operation is (+) which is responsible to add some new information to the state.

2. Forget Gate

The forget LSTM gate, as the name suggests, decides what information should be forgotten. A sigmoid layer is used to make this decision. This sigmoid layer is called the "forget gate layer".

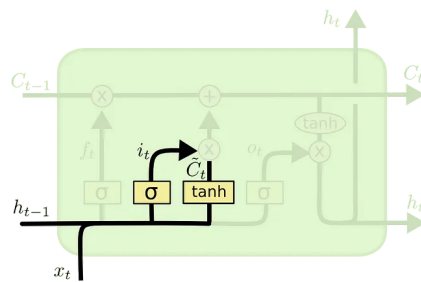
LSTM Forget Gate; Source: [colah's blog](#)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

It does a dot product of $h(t-1)$ and $x(t)$ and with the help of the sigmoid layer, outputs a number between 0 and 1 for each number in the cell state $C(t-1)$. If the output is a '1', it means we will keep it. A '0' means to forget it completely.

3. Input gate

The input gate gives new information to the LSTM and decides if that new information is going to be stored in the cell state.

LSTM Input gate; Source: [colah's blog](#)

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

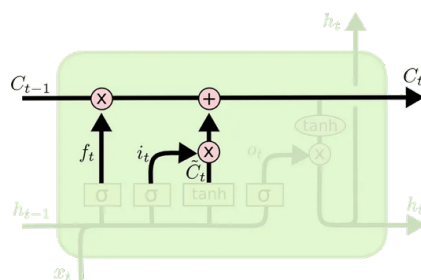
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

This has 3 parts-

1. A *sigmoid* layer decides the values to be updated. This layer is called the "input gate layer"
2. A *tanh* activation function layer creates a vector of new candidate values, $\tilde{C}(t)$, that could be added to the state.
3. Then we combine these 2 outputs, $i(t) * \tilde{C}(t)$, and update the cell state.

Top highlight

The new cell state $C(t)$ is obtained by adding the output from forget and input gates.

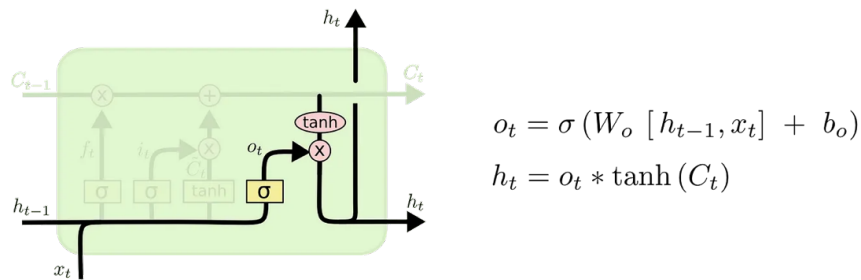


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM new Cell state; Source: [colah's blog](#)

4. Output gate

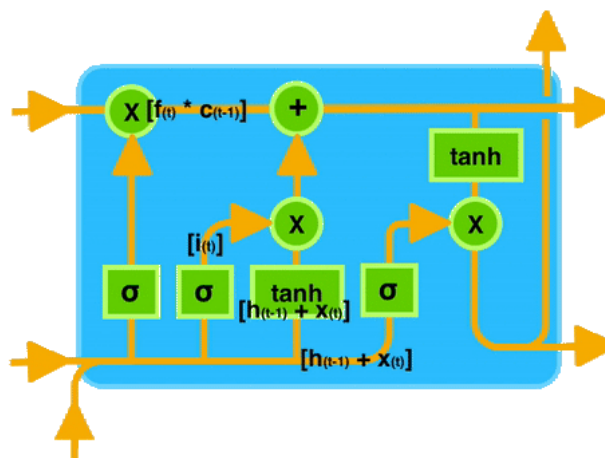
The output of the LSTM unit depends on the new cell state.

LSTM Output gate; Source: [colah's blog](#)

First, a sigmoid layer decides what parts of the cell state we're going to output. Then, a *tanh* layer is used on the cell state to squash the values between -1 and 1, which is finally multiplied by the sigmoid gate output.

LSTM in action

Now that we have understood the architecture and the components of LSTM, let's see it in action.



LSTM in action; Source: Medium article

Conclusion

As mentioned in the article, LSTMs can hold information longer by forgetting and remembering information. This is achieved by 4 components — a cell state and 3 gates. It also combats the vanishing gradient problem, which was a limitation with RNNs. This gives LSTMs an edge over vanilla RNNs. We also understood the architecture and working of LSTMs.

If you like it, please leave a 🍌.

Feedback/suggestions are always welcome.

NLP

Lstm

Machine Learning

Artificial Intelligence

AI



Published in Analytics Vidhya

72K Followers · Last published 2 days ago

Analytics Vidhya is a community of Generative AI and Data Science professionals. We are building the next-gen data science ecosystem
<https://www.analyticsvidhya.com>

Follow



Written by Archit Saxena

63 Followers · 14 Following

Senior Data Scientist

Follow

Responses (3)



What are your thoughts?

Respond



AhmedEhab

Jan 13, 2024 (edited)



I noticed an issue when calculating the new cell state, it should be equal to previous cell state multiplied by the forget gate value $(f_t + (i_t * c_t))$

which you declared correctly, but the problem is that the output of the forget gate f_t should... [more](#)



1



1 reply

[Reply](#)



Nishant Raj

Mar 26, 2024



I have Noticed a Slight Mistake herein the contest of Calculating $f(t)$, You have mentioned that it does a dot product of $h(t-1)$ and $x(t)$. But i think that it concatenates $h(t-1), x(t)$ and Then it does a dot product with weight metric and the resultant concatenated vector in order to calculate the final $f(t)$.



[Reply](#)



Salvatore Raieli

Jan 18, 2023



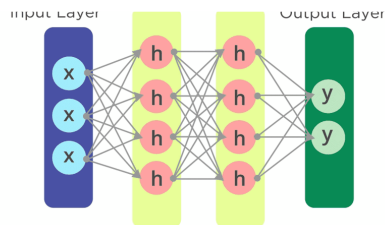
very interesting article!



1 reply

[Reply](#)

More from Archit Saxena and Analytics Vidhya




 Archit Saxena

Recurrent Neural Networks (RNNs)

A recurrent neural network (RNN) is a type of artificial neural network that takes the output

Jan 3, 2023  74

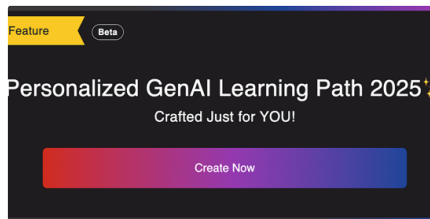


 In Analytics Vidhya by Harikrishnan N B

Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric

Dec 11, 2019  1.2K  6



 In Analytics Vidhya by Himanshi

Introducing the Personalized GenAI Learning Path!

Exciting news! Analytics Vidhya has just launched a new feature: Personalized GenAI

Jan 15  13



 Archit Saxena

Attention is All You Need: Demystifying the Transformer

Unveiling the Revolutionary Transformer Architecture in NLP

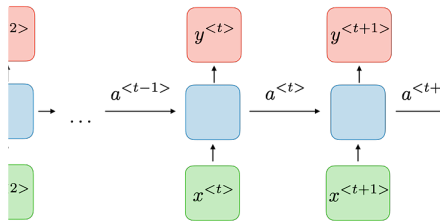
Feb 12, 2024  105




See all from Archit Saxena

See all from Analytics Vidhya

Recommended from Medium




 Rishabh Singh

Recurrent Neural Network (RNN)

In the world of machine learning and deep learning, we've made significant strides in

Nov 7, 2024  10



 Sharmasravan

Building a GRU-Based Sentiment Classifier with TensorFlow and

In this blog, we'll walk through implementing a Gated Recurrent Unit (GRU) neural network

Oct 13, 2024



Lists



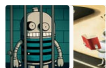
Generative AI Recommended Reading

52 stories · 1651 saves



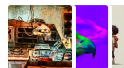
Natural Language Processing

1931 stories · 1585 saves



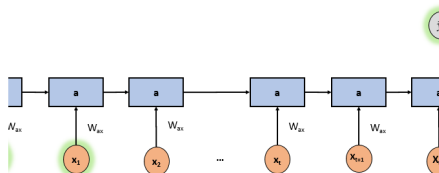
AI Regulation

6 stories · 687 saves



What is ChatGPT?

9 stories · 508 saves

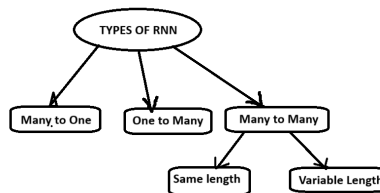


Samina Amin

Understanding Recurrent Neural Networks (RNN)—Step-by-Step

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to

Oct 14, 2024 148



Abhishek Jain

Types of RNN : Many to Many, One to Many and Many to One

1. Many to One:

Sep 28, 2024 20 1



Tiya Vaj

GRU vs LSTM

How GRU Handles Memory

Oct 3, 2024 1



Hugman Sangkeun Jung

Understanding Backpropagation and Vanishing Gradients

A Comprehensive Guide to Backpropagation and the Vanishing Gradient Problem

Nov 14, 2024 1



See more recommendations