SNS COLLEGE OF TECHNOLOGY

(Autonomous)

www.snsgroups.com

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Introduction to Long Short Term Memory

Long Short-Term Memory is an improved version of recurrent neural network designed by Hochreiter & Schmidhuber. LSTM is well-suited for sequence prediction tasks and excels in capturing long-term dependencies. Its applications extend to tasks involving time series and sequences. LSTM's strength lies in its ability to grasp the order dependence crucial for solving intricate problems, such as machine translation and speech recognition. The article provides an in-depth introduction to LSTM, covering the LSTM model, architecture, working principles, and the critical role they play in various applications.Supervised

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies..

The paper defines 3 basic requirements of a recurrent neural network:

- That the system be able to store information for an arbitrary duration.
- That the system be resistant to noise (i.e. fluctuations of the inputs that are random or irrelevant to predicting a correct output).
- That the system parameters be trainable (in reasonable time).

This is a behavior required in complex problem domains like machine translation, speech recognition, and more.LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

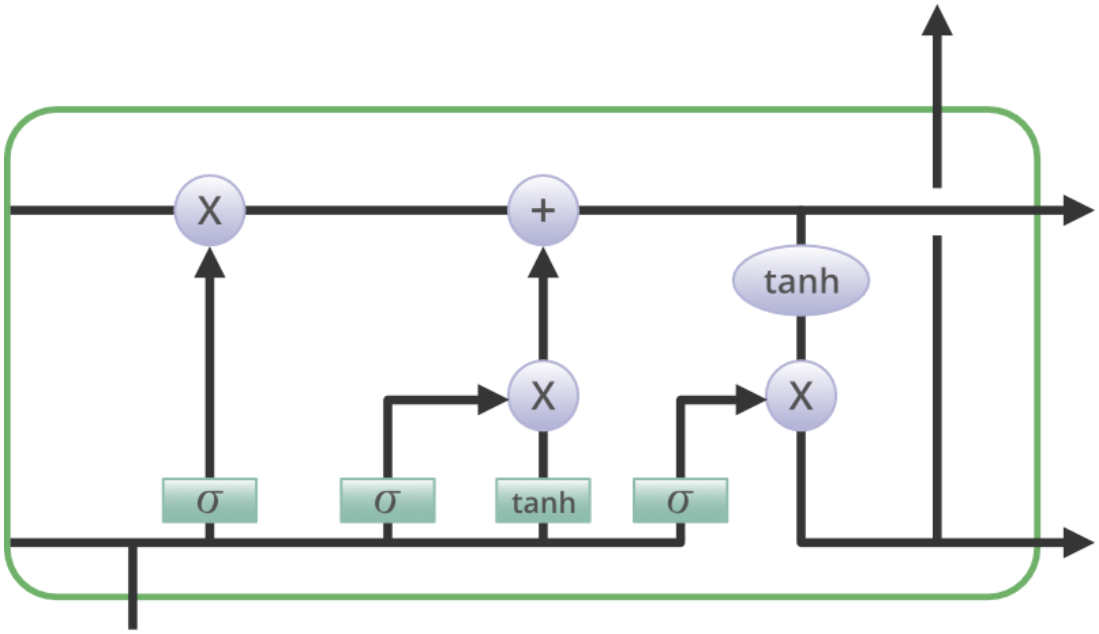
In this post, you will get insight into LSTMs using the words of research scientists that developed the methods and applied them to new and important problems.There are few that are better at clearly and precisely articulating both the promise of LSTMs and how they work than the experts that developed them.

Bidirectional LSTM

Bidirectional LSTM (Bi LSTM/ BLSTM) is recurrent neural network (RNN) that is able to process sequential data in both forward and backward directions. This allows Bi LSTM to learn longer-range dependencies in sequential data than traditional LSTMs, which can only process sequential data in one direction.

Bi LSTMs are made up of two LSTM networks, one that processes the input sequence in the forward direction and one that processes the input sequence in the backward direction. The outputs of the two LSTM networks are then combined to produce the final output. Bi LSTM have been shown to achieve state-of-the-art results on a wide variety of tasks, including machine translation, speech recognition, and text summarization.
LSTMs can be stacked to create deep LSTM networks, which can learn even more complex patterns in sequential data. Each LSTM layer captures different levels of abstraction and temporal dependencies in the input data.

Architecture and Working of LSTM:



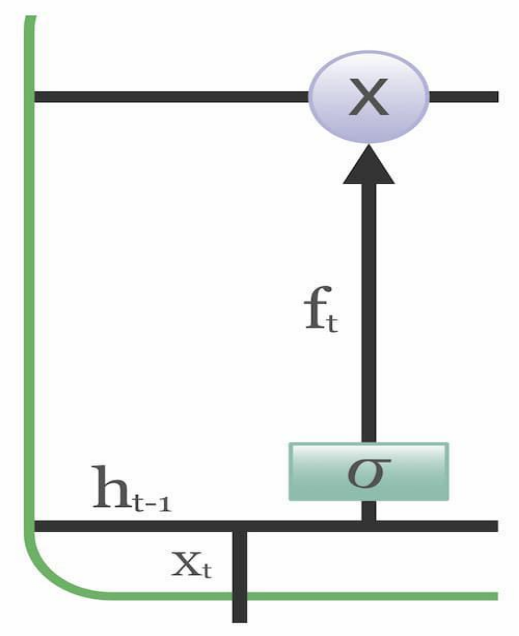
LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.

Forget Gate

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *xt* (input at the particular time) and *ht-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use. The equation for the forget gate is:

where:

- W_f represents the weight matrix associated with the forget gate.
- [h_t-1, x_t] denotes the concatenation of the current input and the previous hidden state.
- b_f is the bias with the forget gate.
- σ is the sigmoid activation function.



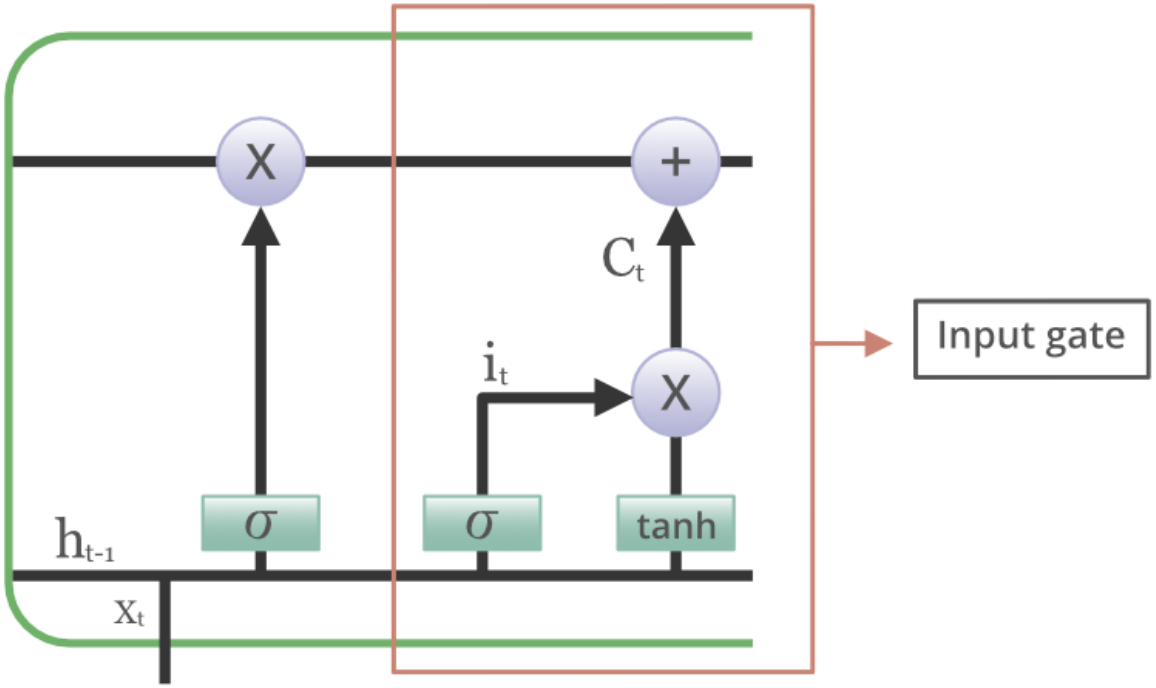
Input gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *ht-1* and *xt*. . Then, a vector is created using *tanh* function that gives an output from -1 to +1, which contains all the possible values from ht-1 and *xt*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

We multiply the previous state by ft, disregarding the information we had previously chosen to ignore. Next, we include it∗Ct. This represents the updated candidate values, adjusted for the amount that we chose to update each state value.

where

- ⊙ denotes element-wise multiplication
- tanh is tanh activation function



Output gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs *ht-1* and *xt*. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is: