Self-Aware LSTM-Based Agents

# Long Short Term Memory (LSTM) Neural Networks

LSTM neural networks are among the most successful recurrent neural network architectures for processing series data, and we give a brief description of them here, closely following (Olah, 2015) from which we borrow our explanatory figures in this section and which we briefly summarize here.

Traditional neural networks are designed to have a fixed input vector **x** and a fixed output vector **h** which depends entirely on the input **x**, and such networks define a function **h**=*f(***x***)*. These neural networks have no memory, and **h** does not depend at all on previous inputs, only on the current input **x**.

A recurrent neural network, by contrast, has an output that depends not only on the current input but also on all previous inputs. Let (**x0,x1,…,xt**) denote a time series of inputs, and let (**h0,h1,…,ht**) denote the corresponding time series of outputs. In a recurrent neural network, we have:

A screenshot of a computer

Description automatically generated with low confidence

Figure : A recurrent neural network

Here, we observe that **ht** is a function not only of **xt** but of all previous inputs as well. Hence, recurrent neural networks have memory, unlike traditional neural networks.

One particularly successful type of recurrent neural network is the LSTM. The basic architecture of the LSTM is shown here.

A screenshot of a video game

Description automatically generated with medium confidence

Figure : LSTM Illustration

We will now explain the internal operation of the LSTM cell in the middle of Figure 2: LSTM Illustration.

The cell state **Ct** is propagated in each computational step as shown in Figure 3: Propagation of Cell State.

A picture containing text, clock

Description automatically generated

Figure : Propagation of Cell State

Here, the state must pass through two key stages: a forgetting stage followed by an updating stage. The forgetting stage multiplies elements of the previous cell state **Ct-1** by numbers between 0 (completely forget that vector element) and 1 (remember that vector element perfectly). The updating stage adds new information to the cell state. The computation of the forgetting stage is shown in Figure 4: Forgetting Stage Computation

Diagram

Description automatically generated

Figure : Forgetting Stage Computation

Once this stage is passed, we must update the cell state. The computation for the update is shown in Figure 5: Update Computation.

Diagram

Description automatically generated

Figure : Update Computation

Combining the above we see that the new cell state is computed as follows:

Diagram

Description automatically generated with medium confidence

Figure : Computing New Cell State

Once the new cell state has been computed, it is necessary to compute the new output **ht**.

A picture containing text, clock

Description automatically generated

Figure : Computing the Output of the LSTM Cell

# Sequence-to-Sequence LSTMs

Sequence-to-Sequence LSTMs have been successfully used for many complex tasks, including language translation (Bengio, 2014) . A description of Sequence-to-sequence LSTMs can be found in (Le, 2014), (Chollet, 2017), and (Bengio, 2014), and our implementation code is a modified version of code from (Chollet, 2017). The description of sequence-to-sequence LSTMs given here is a summary of (Chollet, 2017).

As shown in Figure 8: Sequence to Sequence LSTM, there are two LSTM networks involved. The LSTM encoder accepts an input sequence and generates a final internal state (**h**, **c**) as described in Section 0. The LSTM decoder takes (**h**, **c**) as inputs to its cells along with the [START] token and generates an output sequence of characters in the target language until the [STOP] token is reached. Hence, the meaning of the original English sentence is captured in (**h**, **c**) by the LSTM encoder, and this meaning is then converted into an output sequence in the French language by the LSTM decoder. The decoder LSTM cell is presented, initially, with inputs (**h**, **c**) capturing cell state and linguistic meaning from the encoder, and this is the initial state passed to the cell. The initial input is the [START] token. The decoder’s output is fed back into the input, generating the French phrase one character at a time until the network generates a [STOP] token.

Diagram

Description automatically generated

Figure : Sequence to Sequence LSTM

The process of training such a network is known as teacher-forcing. In teacher forcing which is illustrated in Figure 9: Training a Sequence-to-Sequence LSTM, the input sequence begins with the [START] token and continues one character at a time with the desired French phrase. The desired output sequence is the desired target phrase.

Diagram

Description automatically generated

Figure : Training a Sequence-to-Sequence LSTM

The teacher forcing method has been used with success to training language translation LSTM encoder-decoder networks, and this is the fundamental architecture used in our present work.

# References

Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved from colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/