Four years ago, I came across a post on *Ars Technica* unveiling to the public a short science fiction film, called Sunspring. The 9-minute movie directed by Oscar Sharp and starring Thomas Middleditch was presented at the Sci-Fi-London film festival’s *48h Challenge*. The premise of the Challenge is to produce a film containing a set of prompts (for example, props or dialogue lines), from scratch and in less than two days! However, none of these things actually caught my attention at the time. Instead, it was film-writer Benjamin that enticed me to click on the link. Benjamin is an AI bot, or specifically, a recurrent neural network called long short-term memory (LSTM) which was developed by New York University AI researcher Ross Goodwin.

LSTM is an artificial recurrent neural network (RNN) architecture. This type of architecture has feedback connections, which gives it the ability to process sequences of data and makes it useful for video, speech or text recognition applications (more on this below!). In the case of Sunspring, Goodwin trained his algorithm with dozens of .txt files of sci-fi screenplays that were available online at the time. Once “turned on”, the algorithm looked to find patterns and tendencies in the sequences of letters, words and phrases from the training dataset. Over time, the algorithm learned to reproduce the structure of a screenplay, with its associated stage directions and character lines. Once it was deemed ‘*ready*’ for the Challenge, Benjamin was fed with the following prompts from the Sci-fi London film festival as input:

* TITLE: Sunspring
* DIALOGUE: “It may never be forgiven, but that is just too bad.”
* PROP & ACTION: A character pulls a book from a shelf, flips through it and puts its back.
* OPTIONAL SCIENCE IDEA: In a future with mass unemployment, young people are forced to sell blood. … gave it a cup of really hot tea…

The result? Well, other than boastfully naming *itself* Benjamin and providing hilarious stage directions such as “He is standing in the stars and sitting on the floor”, it *did* produce a script. I invite you to have a look at the short-film to judge for yourself! Here’s the link to the original post:

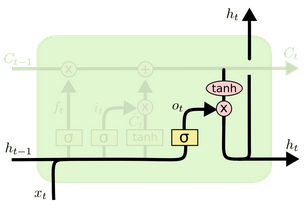
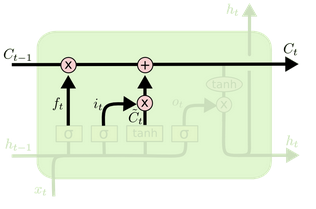
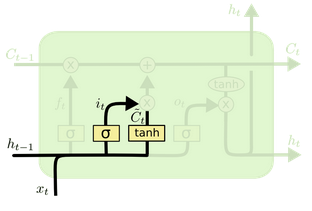
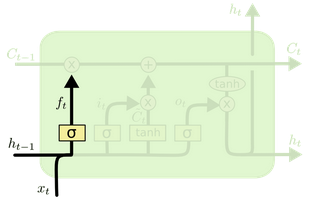
<https://arstechnica.com/gaming/2016/06/an-ai-wrote-this-movie-and-its-strangely-moving/>

If you watched the film, you will have noticed that most dialogue lines make logical *sense* (or at least syntactic sense) when considered independently. This observation is a feature of RNNs in that they use information about a previous prediction to make its current prediction, providing the algorithm with ‘short-term memory’. For example, for a given sentence “*This blog post is about*…”, an RNN will use the information it has for the word ‘about’ to make a prediction on the next word (probably a noun, not a verb, etc.). However, such short-term memory may not give predictions accurately when context is necessary. For example, consider the following text:

“*This blog post is about Sunspring, a short-film presented at the* the Sci-Fi-London film festival’s *48h Challenge. The …*”.

Here, our RNN may not consider the context of films or film festival in order to predict the next word. Depending on the natural language processing (NLP) technique used and the vocabulary of the algorithm, it might just as well predict the word *horse* (!), which would fit syntactically but not contextually. This is known as the long-term dependencies problem.

LSTM, which is a sub-set of RNNs, addresses the issue by incorporating ‘memories’ from further back and weighing them into the next prediction. LSTM has four neural network layers (instead of one in a standard RNN) which interact together in a special way. The adjacent diagram illustrates the concept of LSTM, where



1

2 3

5

4

: is the cell state vector, which incorporates all the *kept* information from previous steps,

: is the input vector,

: is the output vector, and

: is the current time step.

Yellow boxes the neural network layers with their activation functions,

: sigmoid function, and

: hyperbolic tangent function.

Finally, pink circles are junction functions,

X: times junction, and

+: plus junction.

The LSTM algorithm goes through the following steps:

1) Forget gate layer: this layer looks at the previous output and current input to determine what information to keep or forget from the previous cell state .

2) Input gate layer: this layer decides which values from should be updated into the cell state vector.

3) Cell state candidate layer: this layer creates a vector of all new candidate values from for our cell state vector.

4) The above steps are then applied onto the old cell state to update it into a new cell state vector .

5) Output layer: this layer produces a new output from and a filtered .

As mentioned above, the algorithm first needs to learn sequentially the interactions between words, sentences, paragraphs before moving on to the daunting task of capturing the entire structure of a script. It would appear that Benjamin did a good job of learning up to the level of sentences or neighboring dialogue exchanges. And although it was able to construct a full script, it was not capable of capturing the underlying coherence within each script from the training data. The result is a script which lacks meaning…

<https://en.wikipedia.org/wiki/Sunspring>

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

<https://en.wikipedia.org/wiki/Long_short-term_memory>