Text Generation with LSTM

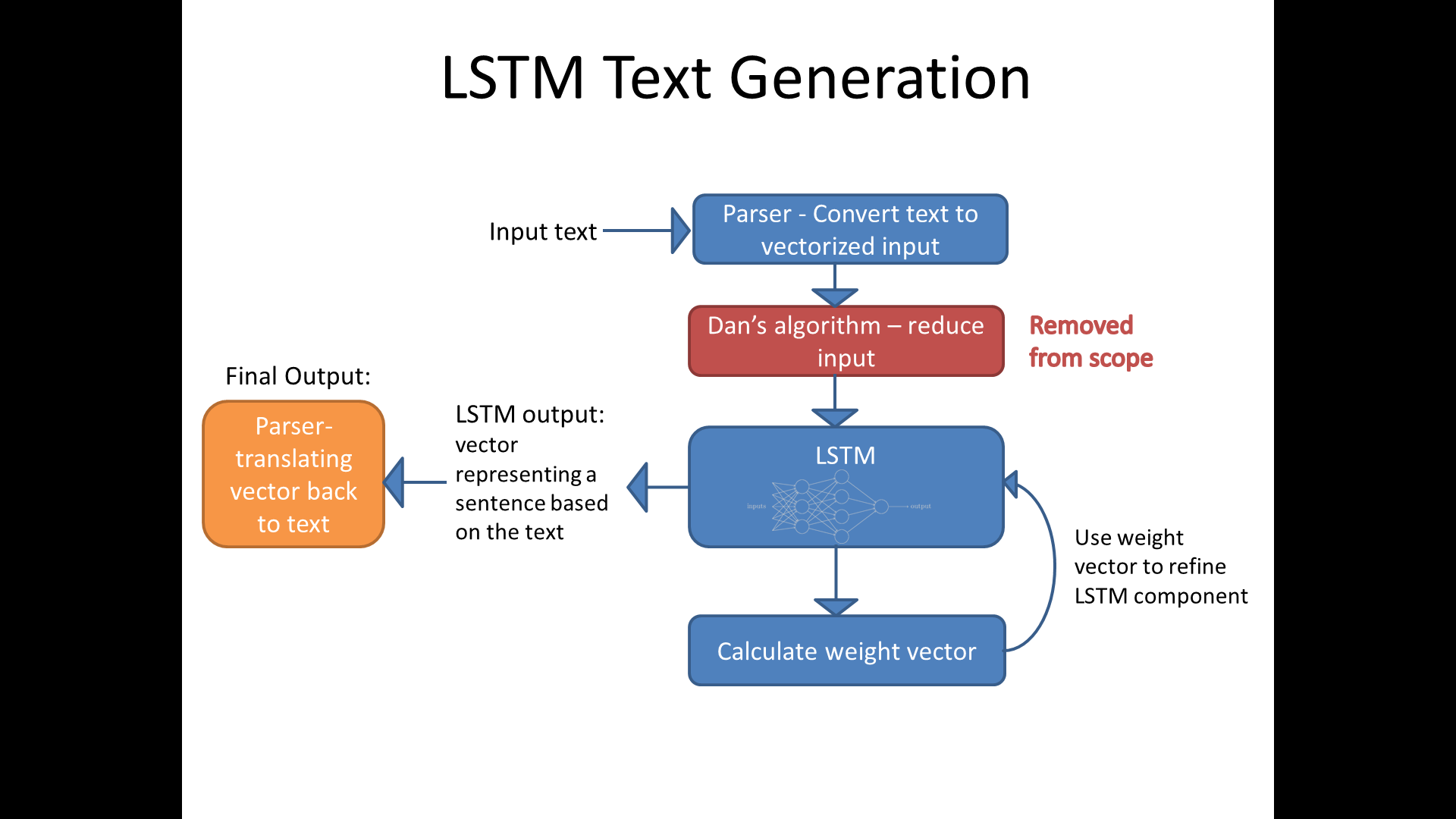
## Overview:

We wanted to create a deep learning system for generating text based on a given input.

For this, we wanted to use the LSTM model- a deep learning multilayer recursive model, which can generate similar output based on a given input.

Our hope was to have the network create english words on its own by learning the patterns between individual characters.

The project model is as follows:



**Parser**: converts the given text character into binary vectors.

**LSTM**: the learning component- based on a given weight vector, and an input, can generate an output in the form of a binary vector.

**Weight vector**: the brains behind the LSTM learning operation- the most time consuming part of the algorithm is to generate the weight vector. Once it's generated though, it's possible to save it to file and reuse it as many time as we'd like.

**Parser**: Converts the binary vector back to text.

## Algorithm steps:

1. We take our text and parse it into a vector of 1's and 0's. The parsing is made by character, so for every character we get a vector that's 26 long, like so:

A = [1,0,0,0,0 , 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,0]

1. We train the LSTM model on the text
   1. For the text, we used 'Alice in Wonderland' in .txt format.
   2. For the model, we used 3 layers of size 512 each. This took about 8 hours to train, creating a weights file.
2. We input the weights file into the model- allowing us to make predictions.
3. Taking a random seed – a sentence – we input it into the model, and it generates an output.

The output is words that are generated character by character, with the hope of simulating English.

## Results

1. Initially, we've attempted our code with a single size 2 layer, for simplicity.

This yielded a lot of blank spaces, and we were unable to get any viable results.

1. Later, we tried a single 128 size layer- this yielded our first result:

**"toe toe toe toe toe.."**

1. Then, a 2 layers with size 256 each:

**"she oue she oue ier she oue ier.."**

Not much, but an improvement over 'toe'

1. When trying 3 layers with size of 512 each, we got the following result:

**"was the was ier and the was ier of the was.."**

This was definitely an improvement over the lower layers- the output contains a lot more English words, but still not as close to fluent English as we'd hoped .

## Conclusion

We assume that the more we increase the size of the network, the more accurate and English-like our results will be. However, on a single machine this process is extremely time consuming (8 hours to train 3 x 512)

We hope that in the future we'll be able to run our code on Amazon cloud- allowing us to produce weights of much more complex networks and getting better results.