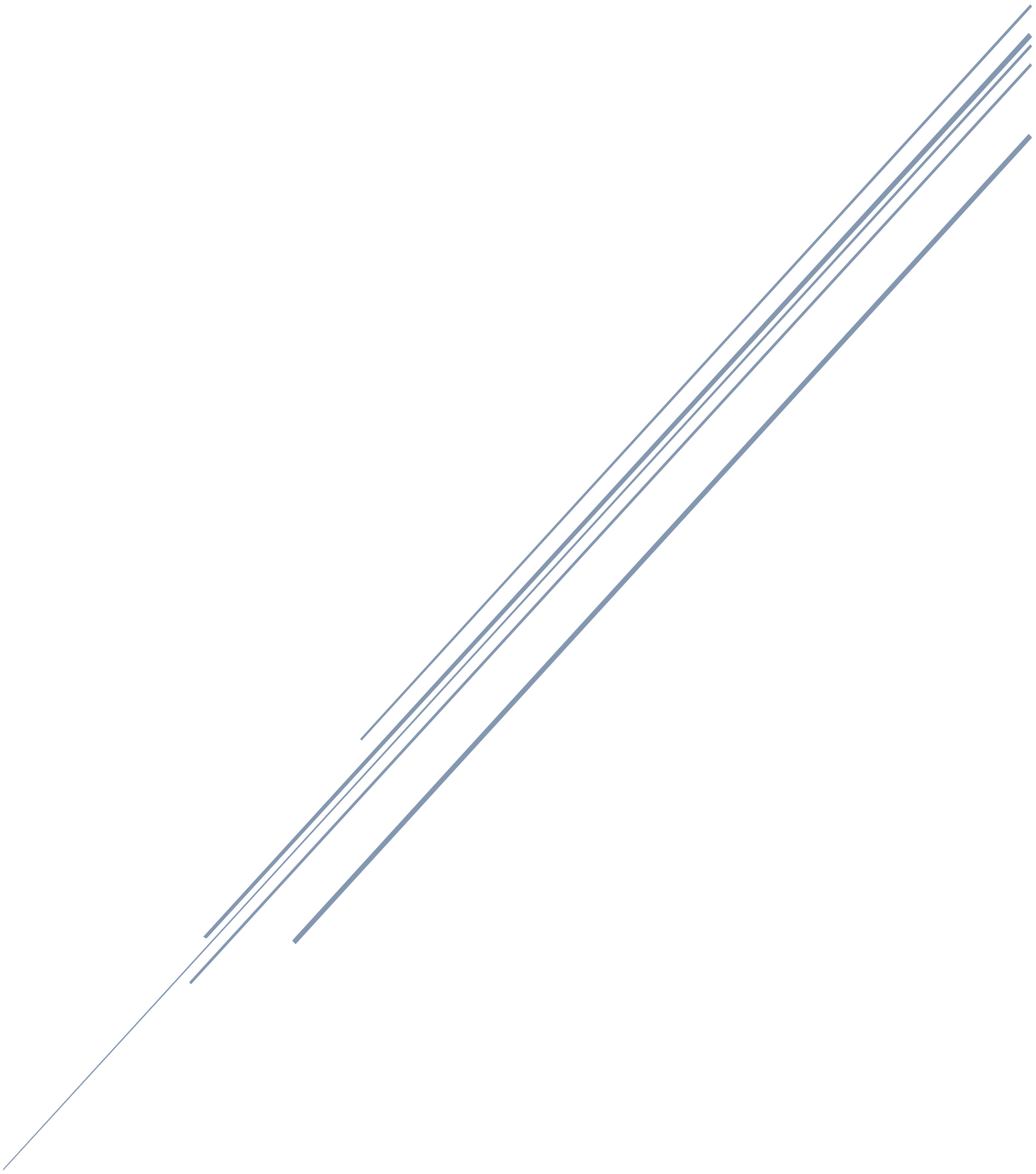
RECURRENT NEURAL NETWORKS (RNN)

Module 3



v-cardona

Deep Learning with Tensorflow

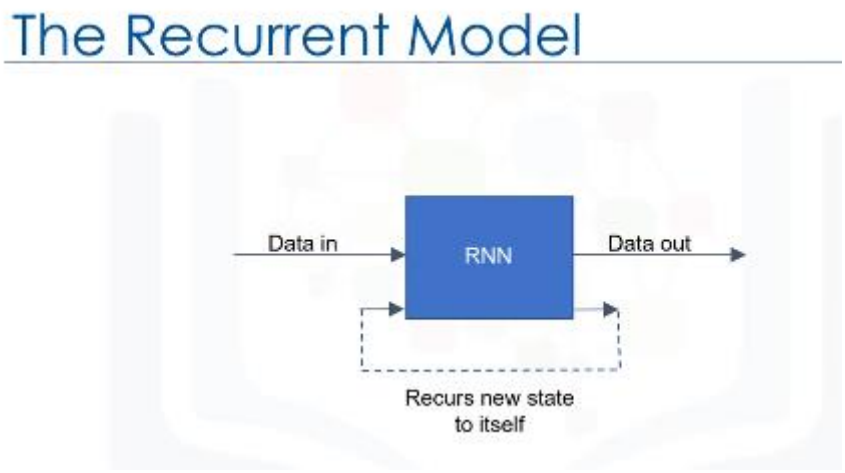
Contenido

The sequential problem

Sequential data. Each input to the neural network depends on the previous data, this is not handle on the traditional networks.

The RNN model

A Recurrent neural network, or RNN for short, is a great tool for modeling sequential data. The RNN is able to remember the analysis that was done up to a given point by maintaining a state, or a context, so to speak. You can think of the "state" as the "memory" of RNN, which captures information about what's been previously calculated. This "state" recurs back into the net with each new input, which is where the network gets its name.

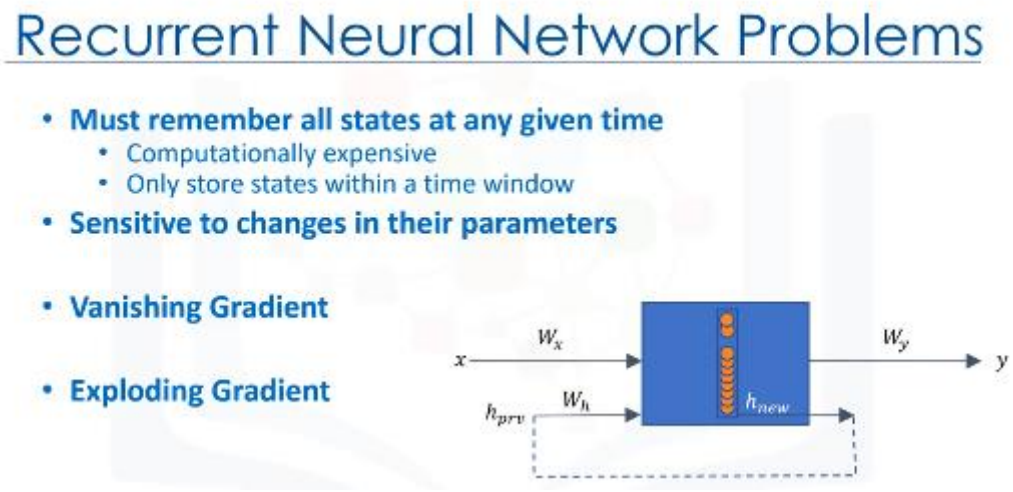


In the hidden layer, two values will be calculated: First, the new or updated state, is to be used for the next data point in the sequence. Second, the output of the network will be computed.

Applications: speech recognition, image captioning, market price,…

It can be multiples outputs, like a sequence of words, it is sometimes called one-to-many (on image input and many words output).

This model presents several problems. One issue is that the network needs to keep track of the states at any given time. There could be many units of data, or many time steps, so this becomes computationally expensive. Another issue is that Recurrent Neural Networks are extremely sensitive to changes in their parameters. As a result, gradient descent optimizers may struggle to train the net. Also, the net may suffer from the "Vanishing Gradient" problem, where the gradient drops to nearly zero and training slows to a halt. Finally, it may also suffer from the "Exploding Gradient", where the gradient grows exponentially off to infinity.



The LSTM model

A specific type of RNN that solve some problems like maintain a strong gradient over many time steps, so you can train long sequences. An LSTM unit in Recurrent Neural Networks is composed of four main elements. The "memory cell" and three logistic "gates":

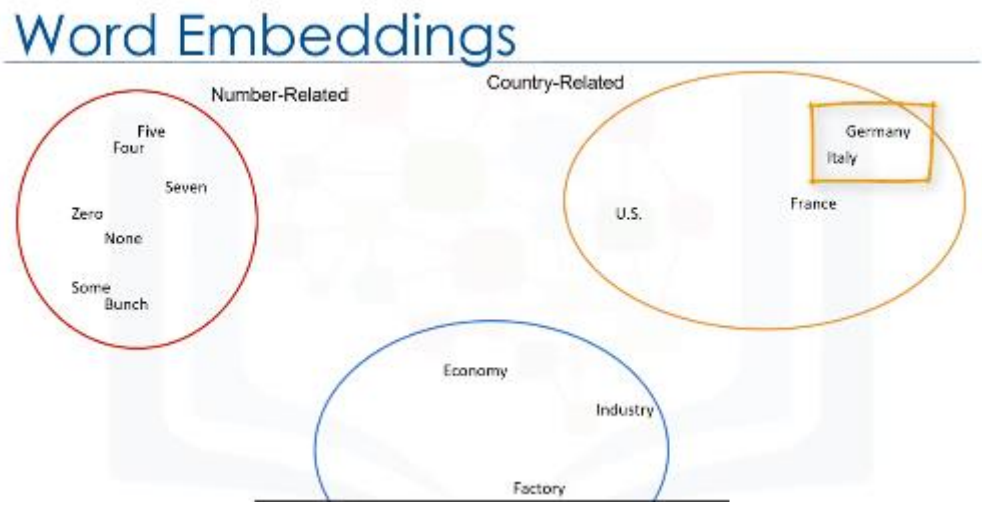
1. The "memory cell" is responsible for holding data.
2. The Write Gate is responsible for writing data into the memory cell.
3. The Read Gate reads data from the memory cell and sends that data back to the recurrent network.
4. The Forget Gate maintains or deletes data from the information cell, or in other words, determines how much old information to forget.

By manipulating these gates, a Recurrent Network is able to remember what it needs in a sequence and simply forget what is no longer useful.

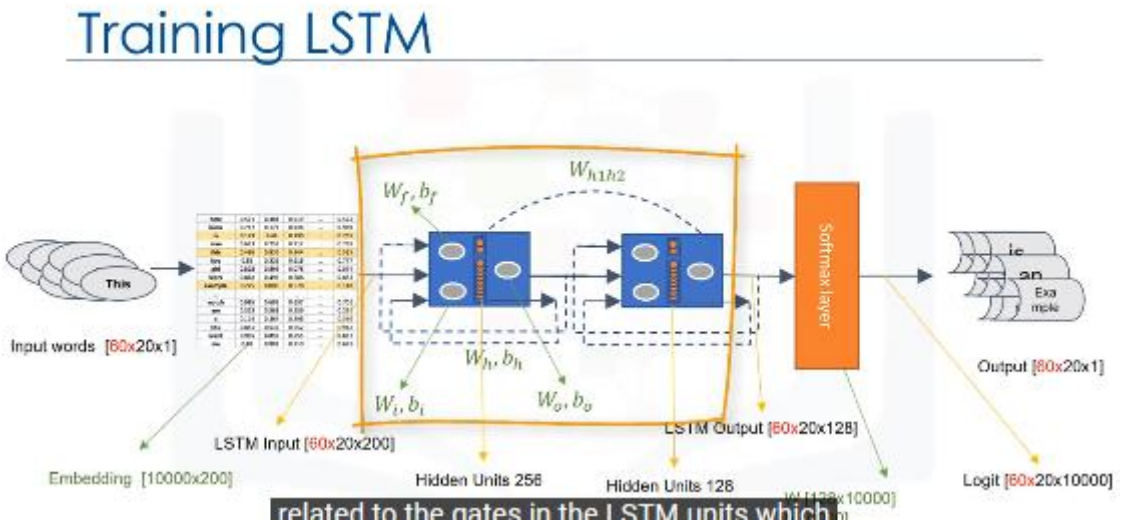
Applying RNNs to language modelling

An interesting way to process words is through a structure known as a Word Embedding. A word embedding is an n-dimensional vector of real numbers for each word. The vector is typically large, for example, 200 length.

After training the RNN, if we visualize the words based on their embedding vectors, the words are grouped together either because they're synonyms, or they're used in similar places within a sentence.



Only one word in each time step is fed into the network, and one word would be the output.



The weights keep updating based on the error of the network-in-training. First, the embedding matrix will be updated in each iteration. Second, there are bunches of weight matrices related to the gates in the LSTM units which will be changed. Finally, the weights related to the Softmax layer, which somehow plays the decoding role for encoded words in the embedding layer.

Lower Perplexity means that the model is more certain about its prediction.