1. Intro to LSTMs

It's time to briefly introduce Long Short Term Memory networks, also known as LSTMs.

2. What are RNNs?

LSTMs are a type of recurrent neural network, RNN for short. A simple RNN is a neural network that can use past predictions in order to infer new ones. This allows us to solve problems where there is a dependence on past inputs.

3. What are LSTMs?

LSTM neurons are pretty complex, they are actually called units or cells. They have an internal state that is passed between units, you can see this as a memory of past steps. A unit receives the internal state, an output from the previous unit, and a new input at time t. Then it updates the state and produces a new output that is returned, as well as passed as an input to the following unit.

4. What are LSTMs?

LSTM units perform several operations. They learn what to ignore, what to keep and to select the most important pieces of past information in order to predict the future. They tend to work better than simple RNNs for most problems.

5. When to use LSTMs?

LSTMs have been used for image captioning, speech to text, text translation, document summarization, text generation, musical composition,and many more.

[1] Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions.

6. LSTMs + Text

Let's go over an example on how to use LSTMs with text data to predict the next word in a sentence!

7. Embeddings

Neural networks can only deal with numbers, not text. We need to transform each unique word into a number. Then these numbers can be used as inputs to an embedding layer.

8. Embeddings

Embedding layers learn to represent words as vectors of a predetermined size. These vectors encode meaning and are used by subsequent layers.

9. Sequence preparation

We first define some text and choose a sequence length. With a sequence length of 3 we will end up feeding our model with two words and it will predict the third one. We split the text into words with the split method. The output looks like this: We then, need to turn these words into consecutive lines of 3 words each. We can loop from seq_len to the number of words + 1 and store each line. The end results look like this:

10. Text preparation in Keras

After that we turn our text sequences into numbers. We import Keras Tokenizer from the preprocessing text module. Instantiate it, fit it on lines, and then turn those lines into numeric sequences. This is how the 3-word lines look now. The tokenizer object stores the word-to-number mapping. There are two dictionaries, the index_word, and the word_index. Here, the index_word is printed, which shows the encoded word for each index. We can use this dictionary to decode our outputs, mapping numbers back to words.

11. Building a LSTM model

Our data is ready to be processed. Now, we are ready to build the LSTM model. We start by importing the Dense, LSTM, and Embedding layers from tensorflow.keras.layers. We then store the vocab_size, since we will use it when defining our layers. The vocab_size is the length of the tokenizer dictionary plus one. The plus one is because we account for 0 as an integer reserved for special characters, as we saw, our dictionary starts at 1, not 0. We add an embedding layer, the input_dim is the vocab_size variable, we will turn our word numbers into 8-dimensional vectors, and need to declare the input_length so that our model understand that two words will be passed simultaneously as a sequence. We end by adding an LSTM layer of 8 units, a hidden layer, and an output layer with softmax and as many outputs as possible words.