Seq2eq in Tensorflwo: <https://github.com/tensorflow/nmt#introduction>

Seq2Seq with attention in Keras: <https://medium.com/@jbetker/implementing-seq2seq-with-attention-in-keras-63565c8e498c>

**Visualizing a Neural Machine Translation Model**

* A neural machine translation model takes an input of a sequence of words and outputs a sequence of words
* Input -> Sequence 2 Sequence model -> Output
* What does the model look like under the hood?
  + Consists of encoder and decoder
  + Encoder takes the input which it converts to a vector called the **context**
  + Encoder processes entire input sequence then sends the **context** to the decoder
    - **Context** is actually just the hidden state from the last timestep of the RNN the encoder processes
  + Decoder produces output sequence item by item
* **IMPORTANT CLARIFICATION POINT IN VISUALIZATIONS**
  + Models will often show a single encoder or decoder entity that processes multiple elements
  + In reality, what is happening is that there are copies of encoders/decoders which process each element in a series of elements (e.g. each word in a series of sentence)
  + Graphic models often don’t show this, but the result from one element (i.e. hidden state) is passed into an RNN for the second element (at the second timestep) which is just a “copy of the encoder” with different date.
  + These keeps going on..
* Encoders and decoders are just models – typically RNNs (for machine translation)
* \*TIP – size of the context vector is the number of hidden units in the RNN
* Words needs to be converted into a vector before it can be fed into the RNN
* **So what’s the deal with attention?**
  + Turns out that the context vector was a bottleneck for long sentences (i.e the model would have to wait on decoding anything until the context vector was generated from the entire input being processed)
  + Solution? Attention
  + Technique which allows model to get relevant information in bits
  + Model can focus on different parts of speech as needed
* How does an attention model differ from a Seq2Seq model?
  + Passes all of the hidden states instead of final hidden state (=more data)
  + Does some different stuff with decoder:
    - Scores each hidden state then
    - Softmaxes these scores (turns them into a probability distribution 0,1)
    - Multiplies each vector by softmax(score) to amplify important bits of information and down out low scores
* How does this come together in the decoder stage?
  + Input is still the context vector (of the embedding token) and an initial decoder state
  + RNN is run and generates a preliminary hidden state and an output (discarded)
  + Attention step – calculate the context vector for this timestep and concatenate this with the preliminary hidden state from the step above
  + Pass this vector (hidden state + context vector for time step) into a **feedforward neural network** (trained in conjunction with the model)
  + Output of the feedforward neural network indicates output word for this time step
* These are how attention sequence 2 sequence models are used for machine translation, but they are also just as important in speech recognition and text summarization