Stat Quest - Attention for NNs

Problem w/ basic en-dec is that LSTM unrolls sentance into a single context vec

this is fine for short phrases

but is input vocab of 1000's words & possibility of longer, more complex sentance

Here, early words in the LSTM can be forgotten

early words might be v important for the context of the sentance

Recall, RNNs had a similar issue w/ remembering things long back

Main idea of LSTM was to solve through seperate path for long & short term memories

but
~~even~~ even w/ seperate paths, if the paths are long words can get lost

Attention adds a bunch of new paths from the encoder to the decoder

Decoder can directly Access Steps from the encoder

this video covers a combination of Seq2Seq combined w/ Attention

encoder-Decoder

↳ this is all a precursor to transformers

how does attention connect the inputs to each step of the decoder

this route is not simple

there are no rules about how Attention should be added to an encoder-Decoder model

↳ only conventions

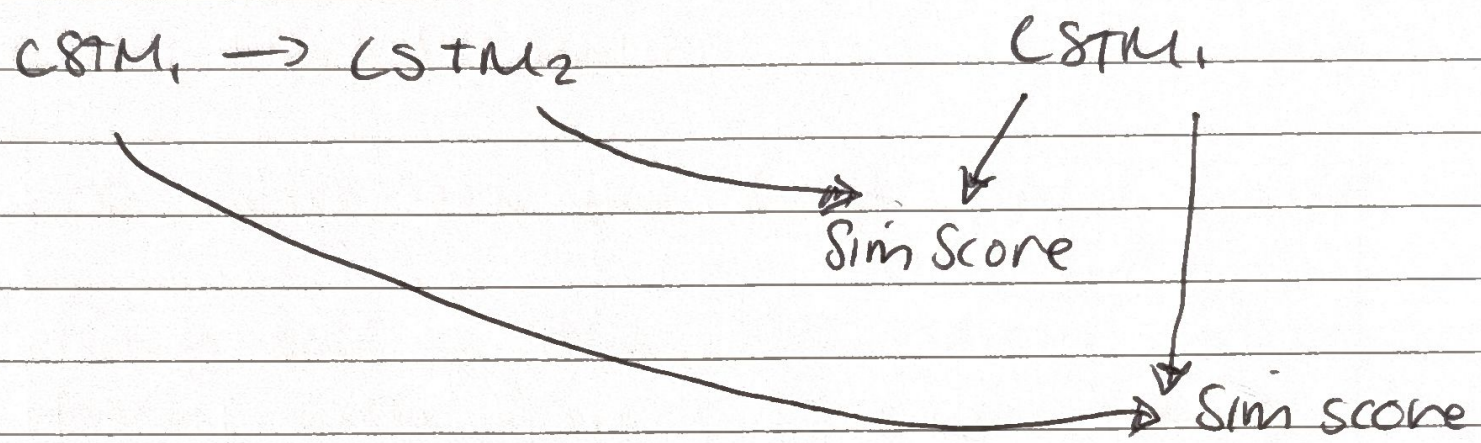
Possible Steps

1. Determin how similar the outputs from the encoder LSTMS @ each step to the outputs of the decoder LSTM

   Similarity Score between "words" (Seq)

   Attention Algo have DIFF ways of doing this but <u>cosine similarity</u> is common

Cosine Similarity =
Calcs Sim between two seqs of nums
between ~~0~~ -1 & 1



hSTM, output from Decode gets a sim score w/ each step of the Encode LSTMS
→ Plugged into Cosine formula

Cosine Sim can be replaces w/ a dot product to improve computation speed w/ similar Affect

how are the scores used?
Higher score = More influence on Decoder Step

Run Score through ||Softmax||
→ recall Soft Max = bet 0 & 1 + All add upto 1
→ Result = % of encode word to use in influence

- Scale input words by their Softmax

Sum all scaled inputs into single val

↳ these are the Attention values for the decode LSTM, input & token/word

to determin the first pred word

- Plug all of the Attention values from each layer into a fully connected layer w/ Softmax

repeat until EOS is predicted
1. unroll LSTM
2. Comp Attention
3. Fully connect

w/ Attention in encoder - Decoder

↳ the encoder is much the same

but decoder, each step has access to individual encodings for each input word

→ use SOFTMAX & similar score to determ what % of each encoded word should be use to help predict the next output word

A finding of attention is that we no longer need the CBTMS

↳ but this is an advancement shown in transformers