Sequence to Sequence Learning with Neural Networks (1409.3215v3)

General idea is to use one LSTM to read input sequence one timestep at a time to obtain large fixed dimensional vector representation then to use another LSTM to extract the output sequence from those vectors.

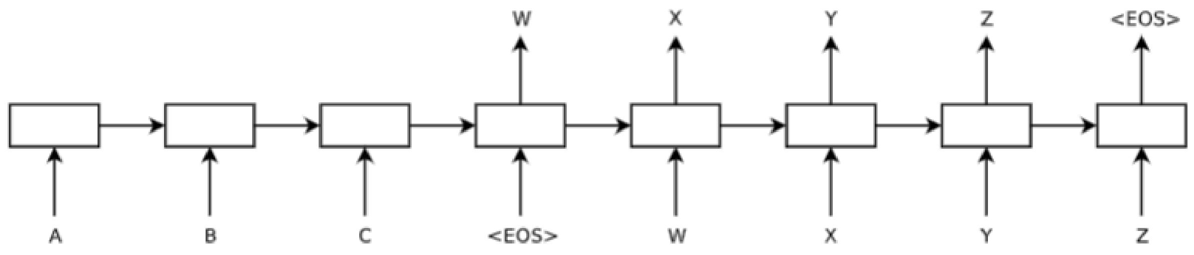

Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence to Sequence Learning with Neural Networks (1409.3215v3), p.2

> The main result of this work is the following. On the WMT'14 English to French translation task, we obtained a BLEU score of 34.81 by directly extracting translations from an ensemble of 5 deep LSTMs (with 384M parameters and 8,000 dimensional state each)

> using a simple left-to-right beamsearch decoder.

Sequence to Sequence Learning with Neural Networks (1409.3215v3), p.2

> Surprisingly, the LSTM did not suffer on very long sentences, despite the recent experience of other researchers with related architectures [26]. We were able to do well on long sentences because we reversed the order of words in the source sentence but not the target sentences in the training and test set. By doing so, we introduced many short term dependencies that made the optimization problem much simpler (see sec. 2 and 3.3). As a result, SGD could learn LSTMs that had no trouble with long sentences. The simple trick of reversing the words in the source sentence is one of the key technical contributions of this work.

The encoder outputs very last hidden state of the input sequence, though it has all the info from the very first time step to the very end but the last token has much more influence during the decoding, which make the optimization very hard but if we feed the encoder the sequence in reverse order then during decoding the first word will be affect the first decoded word and it makes the optimization simpler.

[Sequence to Sequence Learning with Neural Networks (1409.3215v3), p.3]

> The goal of the LSTM is to estimate the conditional probability p(y1, . . . , yT ′ |x1, . . . , xT) where (x1, . . . , xT) is an input sequence and y1, . . . , yT ′ is its corresponding output sequence whose length T ′ may differ from T . The LSTM computes this conditional probability by first obtaining the fixeddimensional representation v of the input sequence (x1, . . . , xT) given by the last hidden state of the LSTM, and then computing the probability of y1, . . . , yT ′ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of x1, . . . , xT :

Logic why reverse the input sequence.

[Sequence to Sequence Learning with Neural Networks (1409.3215v3), p.3]

> we found it extremely valuable to reverse the order of the words of the input sentence. So for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c. This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for SGD to “establish communication” between the input and the output. We found this simple data transformation to greatly improve the performance of the LSTM