A Quick Summary: Attention Is All You Need

Original Paper: https://arxiv.org/abs/1706.03762

24th Feb 2019

1 Ideas:

The dominant model paradigms in NLP before this paper was presented were those of RNNs, CNNs, in addition to adding some form of an attention and/or memory mechanism to the model.

In this paper, the authors show that it is possible to dispense with everything except the attention mechanism(s) and still achieve exemplary results, while taking significantly less time to train.

The model presented in this paper consists of two parts, the encoder and the decoder. The architecture is known as a **transformer**.

(a) Encoder: The input to the encoder is usually a sentence (but theoretically it can be a bunch of sentences too). The sentence has an embedding X, which is an $n \times m$ matrix, where n is the number of words, and m is the dimension of the word embedding. The output is fed into the decoder.

(b) Decoder: At every timestep (training and testing), the decoder takes as input:

  i the output from the encoder

  ii the outputs from the previous timestep of the decoder. And outputs the next word in the (translation) sequence.

2 Explanations:

For simplicity, I will work with the simplest version of the transformer.

(a) When the encoder receives an embedding X:

  i The encoder adds a positional encoding (to preserve sequential information) to get X_1

  ii Multiply X_1 by three matrices (which will be trained) K, Q, V (Keys, Queries, Values) to obtain X_K, X_Q, X_V.

  iii Perform a "Scaled Dot-Product Attention":

$$Z = \text{Attention}(X_Q, X_K, X_V) = \text{softmax}(\frac{X_Q X_K^T}{\sqrt{d_k}})X_V$$

  The result of this would be a matrix of dimension $m \times k$, where k is the number of columns in the matrices K, Q, V. One way to think of the matrix Z is as follows: Each word pays a certain amount of attention to every other word, and now each word is represented by a linear combination of the other words, with the weighting determined by the amount of attention it gives to every other word.

  iv Z is passed through a feed forward network to obtain L.

  v L is then passed to the decoder.

(b) Decoder:

i Before L is passed on to the decoder, it is multiplied by matrices K', V' (note the absence of Q' - explained shortly). The resultant matrices $X_{K'}$ and $X_{V'}$ are then passed to the decoder.

ii As stated before, the decoder also takes in the (partial) translation that we have so far. This is encoded as a query $Y_{Q'}$.

iii Now, as in the encoder, we perform a Scaled Dot-Product Attention on $X_{K'}$, $X_{V'}$, and $Y_{Q'}$, except now this occurs in the decoder:

$$Z' = \text{Attention}(Y_{Q'}, X_{K'}, X_{V'}) = \text{softmax}(\frac{Y_{Q'}X_{K'}^T}{\sqrt{d_k}})X_{V'}$$

Similarly to that in the encoder, one can interpret Z' to be a representation of each translated word we have so far, which is a linear combination of the the representations of the input sentence.

iv We then pass this into another feedforward layer and softmax to obtain the output probabilities.

3 Model:

1. Refer to notes (b). I don't think there is a more comprehensive explanation.

4 Results:

(a) Surpasses all previously published models and ensembles at a fraction of the training costs.

5 Notes:

(a) In my opinion this is truly a game changer - it reduces training costs dramatically and is able to outperform the usual models.

(b) The best online resource that I believe explains this best in detail can be found at http://jalammar.github.io/illustrated-transformer/.