On the Music Transformer

Aditya Gomatam

March 22, 2021

1 Introduction

The Music Transformer or Transformer with Relative Self-Attention[1] is an autoregressive sequence model that builds on the Transformer[2] to consider the relative distances between different elements of the sequence rather than / along with their absolute positions in the sequence. This consideration was designed to model music, recognizing that music relies heavily on repetition to construct meaning and maintain long-term structure. The Music Transformer, in my estimation, appears to be the first neural network capable of reasonably capturing any sort of meaning to a piece. Google achieved extremely compelling results that truly demonstrate this capability of the Transformer architecture with their Piano Transformer. This essay presents my thoughts on why the Self-Attention and Relative Self-Attention algorithms are so effective.

2 Self-Attention

Self-Attention is an algorithm that is exactly what it sounds like - having a model pay attention to itself, or rather, it's entire input. How can you achieve this? By trying to find the amount of "attention" that each position in the sequence pays to every other position in the sequence, in deconstructing (or reconstructing) the overall meaning of the sequence. Unlike convolutional or recurrent neural networks, the Transformer is able to select which portions of the input sequence to attend to, in order to best predict the next token.

So, how does it work? Vaswani et. al. 2017 describe Self-Attention as, "mapping a query and a set of key-value pairs to an output"[2]. I did not learn much from that. The way I see it, it's easier to understand by exploring the Attention equation itself.

Let's start with a simplest scenario - suppose we want to compute the attention for a single sequence X of length L of events in a vocabulary. First, we embed the sequence (create learnable variable vector representations for each event in the vocabulary, then replace each element in the sequence with the corresponding vector) to make it of shape (L, d_{model}), where d_{model} is the embedding size or hidden dimension of the Transformer. For Attention to work properly, and for residual connections to be made conveniently, most tensors being handled by the Transformer will have a dimension of shape d_{model}, which is why it is so named.

Now, as we had established before, we want to determine the amount of "attention" each element of X should pay to every other element in X - that is, we want to compute some sort of compatibility of each element with every other element in the sequence. It's easy to see that such information would be best stored in a matrix M such that M_{ij} represents the compatibility of the i^{th} element of the sequence with the j^{th} element of the sequence. So, if we could create representations of the sequence such that they can be mapped onto each other to determine how compatible each element is with every other element, we can achieve this goal.

This is why the first step in the Attention calculation is to create two different representations of the input sequence, under the assumption that the closer each element of one is to the other (i.e., the higher their dot product), the more compatible they are, and the more attention one pays to the other. These two representations are called the *queries* and *keys*. We want a number dependent on the dot product of the i^{th} query with the j^{th} key to measure the compatibility of the j^{th} element with the i^{th}. Notice that this need not be symmetric information - the playing of a certain note may not be as important to a change in volume as the change in volume is to the note.

This is why the first step in the calculation of Attention is to *transform* the input sequence into 3 different representations - the queries, keys, and values:

$$Q = XW^Q + B^Q$$
$$K = XW^K + B^K$$
$$V = XW^V + B^V$$

where the parameter tensors W^i and B^i are learned by backpropagation. In order to preserve the shape of X, i.e., (L, d_{model}), each W^i must be shaped (d_{model}, d_{model}), and each B^i must be shaped $(d_{model},)$.

What this seeks to achieve is to determine how much a query lines up with a key, and then use that much of the value in creating the next representations of the sequence

e create three representations of the sequence - two to determine how compatible each

We create two representations of the sequence, and by the dot product attention, determine how compatible the i element is with the j.

Assuming this information about importance is contained within the sequence itself, it can theoretically be learned by backpropagation.

Turns out, that's fairly easy (in theory) to do. We create a representation of the sequence that holds the importance of each element in the sequence

So, if we were able to create a representation of the sequence that holds the importance of each element j

So, if we were to create a representation of the sequence that holds the importance of all elements in the sequence to themselves

queries the importance of element i in constructing the sequence by dot producing with the importance of element j in the sequence held in the representation of the sequence

queries guesses an initial value for the attention

sdfsdfsdfSo, if we were to create a representation of the sequence that *queried* for the importance of each element in the sequence, matched it with a representation of the sequence that *keyed* the importance of each element in the sequence to itself, and matched it

So, let's compute this compatibility matrix.

A central assumption here is that this information about compatibility is contained within the sequence itself - note on translation

Attention, or more formally, Scaled Dot-Product Attention, is given by:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where Q is the queries tensor, K is the keys tensor and V is the values tensor. For those who don't know what softmax is, it's a function that can convert each vector in a tensor, or each matrix in a tensor, or the tensor itself, into a probability distribution. Look it up.

note: positional information is also important for generalizing the model to arbitrary sequence length note: relative self-attention overfits much quicker than normal self-attention but learns much faster

References

[1] C. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music Transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, 2017.