Thank you to the reviewers for their helpful comments.

Reviewer #3

Regarding affine relationships between positions: we agree that a transformer's layers do not produce simple affine transformations. However, each layer's query and key transforms are affine. In the original transformer, the sinusoidal positional encodings can index a relative position in the sequence using a linear transformation: say position pos is represented by a positional encoding PE_{pos}, then positional encoding of $pos + k$ can be reconstructed using a transformation matrix A_{+k}, so $PE_{pos+k} = A_{+k}PE_{pos}$, allowing query and key transformations to model relative postitions. (This is explained in more detail in our response to Reviewer #4.) This lets the model easily attend to relative positions along the sequence.

In our paper, we want our tree positional encoding scheme to share this property, so that the model can easily attend to relative positions in the tree. Each individual move up or down the tree can be captured with an affine transformation, and a series of moves can be represented as the composition of their affine transformations. We will expand Figure 1 to include example positional encodings and the matrices that transform them, generally review Section 3 carefully.

Overall, we'll make a careful pass of this section of the paper to improve readability. We'll also provide further background about each of the datasets considered.

Reviewer #4

Regarding lines 72-74, which claim that the calculations of any element of a sequence are independent of the order presented, we can cite the original transformer paper (*Attention is all you need*), which says this in section 3.5: "Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks." The XLNet paper (https://arxiv.org/pdf/1906.08237.pdf) also exploits this property: holes can be introduced in the middle of the sequence, because order is captured using only positional encodings.

Regarding line 88, which claims that relationships between sequential positional encodings are modeled by affine transformations, we can again cite the original transformer paper: "We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}." The original paper defines positional encodings as follows:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i\ /\ d_{model}}\right); PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i\ /\ d_{model}}\right)$$

Relative movements in position encodings can leverage the following identities:

$$cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta); sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$$

The transformation that attends to a position offset by k is a block diagonal matrix, using the identities above to shift each position accordingly.

Regarding lines 107-108 and 117-119 (the tree transformations and their preservation of movements up to depth k) we'll spell this out in more detail. In effect, our positional encoding acts like a stack, where a downward movement pushes on a particular child position to the end of the stack, and an upward movement pops off the last position from the stack. Because our stack has fixed dimension, once we push too many elements onto the stack, we lose information about the root.

Regarding "lack of richness": we did find that parameter-free approach did not work as well in practice. Richness is perhaps a poor term here, as the parameters don't add extra computational power over the initial projection matrix. Rather, the parameters induce a bias during training time by correlating nodes at the same depth. We believe this bias helps the model incorporate the kind of information seen in the example heatmaps included in our figures. We will rewrite this section to explain this intuition better in the revised draft.

We'll improve the notation around lines 111 and 112 to clarify that the D_i and U operations can act upon any node, and we will expand Figure 1 to include an example of positional encodings that result from applying these operations. Please note that in Equation 1, D_i is not defined as a function of x – rather, the left-hand side is the matrix-vector product of D_i and x.

Although we did not evaluate the positional encodings in any tree-to-sequence setting, one can use tree positional encodings on the encoder side and the original sequential positional encodings on the decoder side to represemt tree-to-sequence mappings.

Thank you for your suggestions on citing uncited figures and reformatting citations. We will incorporate these into the revised draft.