# AttentionTransformerGraph

February 2021

## 1 Notes on Attention

In this section, we study a bit of attention mechanism. This section will be rewritten after we have covered a sufficient number of papers to establish a good understanding of the attention mechanisms which seem to be may.

### 1.1 Hierarchical Attention [4]

Hierarchical Attention is described in [4] which consists two levels of attention, a **word** attention and **sentence** attention.

#### 1.1.1 Word Attention

In [4], the **Word Attention** is described as follows. Some analysis is borrowed from the notes [1]

Consider a document $\mathcal{D}$ consists of some **sentences** $S_i$ i.e. $\mathcal{D} = \{S_i\}_{i=1}^{d}$. In which each $S_i$ is a sentence consists of $T_i$ number of words $w$ i.e. $S_i = \{w_j^i\}_{j=1}^{T_i}$. (Note it can be seen that the length of each sentence can be arbitrary such that $T_i \neq T_j : \forall i \neq j \in \mathcal{D}$; thus some paper mentioned the use of *padding* to enforce $T_i = T_j$).

The process of calculating the *word attention* of a given word $w_j^i$ is given as follows in sequential order

1. Embed the word,
2. MLP, with a trainable parameters $W_w$
3. softmax, with a trainable vector $v$

It is worth highlighting the dimensions of the inputs and outputs which we will cover at each particular step.

An embedding $f : X \to Y$ in which $X$ is a dictionary of words and typically $Y \in R^K$, for example a common choice of $f$ is BERT [3] hence $K = 768$. We obtain the embedding of a single word $w_j^i$

$$h_j^i = f(w_j^i) \tag{1}$$

Next, a transformation step is performed through a MLP layer defined by

$$u_j^i = \tanh\left(W_w h_j^i + b_w\right) \tag{2}$$

Since $h_j^i \in R^K$, and assuming that $W_w \in R^{L\times K}$ thus $u_j^i$ is a vector of dimension $L$. It seems the dimension of $W_w$ is not covered in the original paper but some sources seem to use $L = K$ (but in general it makes sense that $L \leq K$ so it is not projecting to **higher** dimensional spaces). [1] looks at this from a geometric perspective in this sense this step transform the embedded vector (or annotated words as the author called it), the resulting vector/space is then put through tanh and we might wonder why tanh? (I have no clue here, if we use the tanh in its exponential form it is clear that it is continous thus we can imagine the resulting space is mapped towards a different space, how this new space looks like i donot really know, needs more reading; or maybe it does not have anything to do with geometric was purely from technical details of neural net).

So far we got $u_j^i \in R^k$, the next step is calculate a softmax using a vector $u_w \in R^k$ which is trainable (now I regret using the raised $i$ notation)

$$\alpha_j^i = \frac{(u_j^i)^T u_w}{\sum_{j\in T_i}(u_j^i)^T u_w} \tag{3}$$

so $\alpha_j^i \in [0, 1]$ should be a scalar. The original paper refers to $u_w$ as **word level context vector**, and $\alpha_t^i$ as **normalized importance weight** because $\sum_{j\in T_i} \alpha_j^i = 1$. Thus one can see this step determines a *weight* of a word by

# 2 Activation Functions

This section spends some time study abit about some of the popular activation functions just for the fun and curioustiy of it

## 2.1 tanh

tanh is a popular activation function commonly seen in many attention as can be seen from 1. tanh belongs to the family of hyperbolic functions imagine trigometry but defined on hyperbol instead of circles, thus similarly to tan = sin / cos we have tanh = sinh / cosh. We are given

$$\sinh = \frac{1 - e^{-2x}}{2e^{-x}}$$
$$\cosh = \frac{1 + e^{-2x}}{2e^{-x}}$$

and hence

$$\tanh(x) = \frac{sinh}{cosh} = \frac{1 - e^{-2x}}{1 + e^{-2x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4}$$

in this form it can be easily seen that the function is continuous.

One cool way to write tanh is as follows (collected from [2]). Using Weierstrass factorization for cosh as

$$\cosh(x) = \prod_{k=1}^{\infty} \Big(1 + \frac{4x^2}{(2k-1)^2\pi^2}\Big)$$

taking *log* of both sides

$$\begin{aligned}\log\cosh(x) &= \log\Big\{\prod_{k=1}^{\infty}\Big(1 + \frac{4x^2}{(2k-1)^2\pi^2}\Big)\Big\} \\ &= \sum_{k=1}^{\infty} \log\Big(1 + \frac{4x^2}{(2k-1)^2\pi^2}\Big)\end{aligned}$$

differentiate both side with respect to $x$

$$\begin{aligned}\tanh(x) &= \sum_{k=1}^{\infty} \frac{1}{1 + \frac{4x^2}{(2k-1)^2\pi^2}} \frac{8x}{(2k-1)^2\pi^2} \\ &= \sum_{k=1}^{\infty}\end{aligned}$$

# References


[1] [].

[2] [].

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.