Our tensor notation is discussed in Section **??** of Bayesuvius.
ℓ = maximum number of words in a sentence segment. $\alpha \in [\ell]$, $\ell \sim 100$
L = number of words in vocabulary, $\beta \in [L]$, $L >> \ell$
$d = d_{\underline{q}} = d_{\underline{k}} = d_{\underline{v}} = 64$, hidden dimension per head, $\delta \in [d]$.
$n_{\underline{h}} = 8$, number of heads, $\nu \in [n_{\underline{h}}]$
$D = n_{\underline{h}} d = 8(64) = 512$, hidden dimension for all heads, $\Delta \in [D]$
$\Lambda = 6$, number of layers in plate (a.k.a., stack), $\lambda \in [\Lambda]$
reshaping

$$T^{\nu,\delta} \to T^{\Delta} \quad \left(T^{[n_{\underline{h}}],[d]} \to T^{[D]}\right) \tag{1}$$

$$T^{\Delta} \to T^{\nu,\delta} \quad \left(T^{[D]} \to T^{[n_{\underline{h}}],[d]}\right) \tag{2}$$

concatenation

$$T^{[n]} = (T^0, T^1, \ldots, T^{n-1}) = (T^\nu)_{\nu \in [n]} \tag{3}$$

Hadamard product (element-wise, entry-wise multiplication)

$$T^{[n]} * S^{[n]} = (T^\nu S^\nu)_{\nu \in [n]} \tag{4}$$

Matrix multiplication ($T^{[n]} = T^{[n],[1]}$ is a column vector)

$$(T^{[n]})^T S^{[n]} = \text{scalar} \tag{5}$$

$$T^{[a],[b]} S^{[b],[c]} = \left[\sum_{\beta \in [b]} T^{\alpha,\beta} S^{\beta,\gamma}\right]_{\alpha \in [a], \gamma \in [c]} \tag{6}$$

$$x^{[L],[\ell]} = \text{one hot columns} = [\delta(\beta, \beta(\alpha))]_{\beta \in [L], \alpha \in [\ell]} \tag{7}$$

$$e^{\delta,\alpha} = \sum_\beta E^{\delta,\beta} x^{\beta,\alpha} \quad \left(e^{[d],[\ell]} = E^{[d],[L]} x^{[L],[\ell]}\right) \tag{8}$$

$$Q^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{q}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(Q^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]} E^{[d],[\ell]}\right) \tag{9}$$

$$K^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{k}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(K^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} E^{[d],[\ell]}\right) \tag{10}$$

$$V^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{v}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(V^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]} E^{[d],[\ell]}\right) \tag{11}$$

$$B^{\nu,\alpha,\alpha'} = \frac{1}{\sqrt{d}} \sum_\delta Q^{\nu,\delta,\alpha} K^{\nu,\delta,\alpha'} \quad \left(B^{[n_h],[\ell],[\ell]} = \left[\frac{1}{\sqrt{d}} (Q^{\nu,[d],[\ell]})^T K^{\nu,[d],[\ell]}\right]_{\nu \in [n_{\underline{h}}]}\right) \tag{12}$$

$$A^{[n_{\underline{h}}],[d],[\ell]} = \left[\sum_\alpha V^{\nu,[d],\alpha} \underbrace{\text{softmax}(B^{\nu,\alpha,[\ell]})}_{P^{\nu,\alpha,[\ell]}}\right]_{\nu\in[n_{\underline{h}}]} \quad (13)$$

$$= \left[V^{\nu,[d],[\ell]} P^{\nu,[\ell],[\ell]}\right]_{\nu\in[n_{\underline{h}}]} \quad (14)$$

$$\sum_{\alpha\in[\ell]} P^{[n],[l],\alpha} = 1 \quad (15)$$

$$A^{[n_{\underline{h}}],[d],[\ell]} \to A^{[D],[\ell]} \quad (16)$$

- **Positional Enbedding (a.k.a. encoding) Matrix**

  $E_{pos}^{[d],[\ell]}$

$$E_{pos}^{\delta,\beta} = \begin{cases} \sin\left(\frac{\beta}{10^{4\delta/d}}\right) = \sin(2\pi\frac{\beta}{\lambda(\delta)}) & \text{if } \delta \text{ is even} \\ \cos\left(\frac{\beta}{10^{4(\delta-1)/d}}\right) = \cos(2\pi\frac{\beta}{\lambda(\delta)}) & \text{if } \delta \text{ is odd} \end{cases} \quad (17)$$

  $E_{pos}^{\delta,\beta}$ changes in phase by $\pi/2$ every time δ changes by 1. Its wavelength λ is independent of β, but increases rapidly with δ, from $\lambda(\delta = 0) = 2\pi * 1$ to $\lambda(\delta = d) = 2\pi * 10^4$.

  Total Enbedding equals initial enbedding plus positional enbedding:$E = E_0 + E_{pos}$

  The purpose of positional enbedding is to take $x^{\beta,\alpha}$ to $e^{\delta,\alpha} = \sum_\beta E_{pos}^{\delta,\beta} x^{\beta,\alpha}$ where $e^{\delta,\alpha}$ changes quickly as δ (position) changes.

- **ReLU**

  For a tensor T of arbitrary shape

$$ReLU(T) = (T)_+ = max(0, T) \quad (18)$$

  max element-wise

- **Feed Forward neural net**

$$F(x^{[1],[\ell]}) = ReLU(x^{[1],[\ell]} W_1^{[\ell],[d]} + b_1^{[1],[d]}) W_2^{[d],[\ell]} + b_1^{[1],[\ell]} \quad (19)$$

$$F(x^{[\ell]}) = W_2^{[\ell],[d]} ReLU(W_1^{[d],[\ell]} x^{[\ell]} + b_1^{[d]}) + b_1^{[\ell]} \quad (20)$$

- **Softmax**

  softmax() takes a vector and returns a vector of probabilities of the same length

  $$x^{[n]} \to P^{[n]} \tag{21}$$

  where

  $$P^{\alpha} = \frac{\exp(x^{\alpha})}{\sum_{\alpha \in [n]} \exp(x^{\alpha})} \quad \left(P^{[n]} = \frac{\exp(x^{[n]})}{\parallel \exp(x^{[n]}) \parallel_0} \right) \tag{22}$$

  For example,

  $$(1, 0, 0) \to (e, 1, 1)/norm \tag{23}$$

  $$(10, 0, 0) \to (e^{10}, 1, 1)/norm \approx (1, 0, 0) \tag{24}$$

  For any $a \in \mathbb{R}$,

  $$(a, a, a) \to (1, 1, 1)/3 \tag{25}$$

- **Skip Connection (Add & Normalize)**

  A **skip connection** is when you split the input to a **filter** into two streams, one stream goes through the filter, the other doesn't. The one that doesn't is then merged with the output of the filter via a **add & normalize** node. The reason for making skip connections is that the signal exiting a filter is usually full of jumps and kinks. By merging that filter output with some of the filter input, one smooths out the filter output to some degree. This makes back-propagation differentiation better behaved.

  The filter might be a Multi-Head Attention or a Feed Forward NN.

  Add & Normalize just means $(A + B)/norm$ where A and B are the two input signals and "norm" is some norm of $A + B$ (for instance, $\parallel A + B \parallel_2$).

  Normalization keeps the signal from growing too big and saturating the signal entering components upstream. Normalization can also involve subtracting the mean $\langle X \rangle$ of the signal X so as to get a signal $X - \langle X \rangle$ that has zero mean.

- **Redundancy**

  For better results, the Decoder contains a pair of multi-head attentions in series, and Λ of those pairs in parallel.