Forward pass through model

Take the example sentence *"The dog fetched the stick."*, tokenize into [[CLS], "The", "dog", "fetch", "##ed", "the", "stick", ".", [SEP]]. Input vector $\mathbf{X} = \mathbf{E}_{\text{token}} + \mathbf{E}_{\text{segment}} + \mathbf{E}_{\text{position}} \in \mathbb{R}^{10\times768}$.

Output of layer 1 is $U_1 = U_1(\mathbf{X}) \in \mathbb{R}^{10\times768}$, output of layer k with $k \in [2\ldots12]$ is $U_k = U_k(U_{k-1})$.

Calculate a weighted sum r_j for token j across all layers $i \in [1\ldots12]$ as follows:

$$r_j = \eta \sum_{i=1}^{12} U_{i,j} \cdot \text{softmax}(\lambda)_i \tag{1}$$

with η a trainable scalar and λ a vector of trainable scalar mixing weights. Tokens [CLS] and [sep] are not used. In case of subword tokenization, only the first subtoken of a word is used.

Next, r_j is passed through separate MLPs with 768 hidden dimensions and ELU non-linear activation:

$$\begin{aligned}
H_{\text{arc-head},j} &= \text{ELU}(W_{\text{arc-head}}\, r_j + b_{\text{arc-head}}) \\
H_{\text{arc-dep.}j} &= \text{ELU}(W_{\text{arc-dep}}\, r_j + b_{\text{arc-dep}}) \\
H_{\text{tag-head},j} &= \text{ELU}(W_{\text{tag-head}}\, r_j + b_{\text{tag-head}}) \\
H_{\text{tag-dep},j} &= \text{ELU}(W_{\text{tag-dep}}\, r_j + b_{\text{tag-dep}})
\end{aligned} \tag{2}$$

These are then used to score all possible dependency arcs:

$$\begin{aligned}
\mathcal{S}_{\text{arc}} &= H_{\text{arc-head}} \mathbf{W}_{\text{arc}} H_{\text{arc-dep}}^{\top} + \mathbf{b}_{\text{arc}} \\
\mathcal{S}_{\text{dep}} &= H_{\text{dep-head}} \mathbf{W}_{\text{dep}} H_{\text{dep-dep}}^{\top} + \mathbf{b}_{\text{dep}}
\end{aligned} \tag{3}$$

Then the Chu-Liu/Egmonds algorithm is used to obtain a valid dependency tree:

1. For each node $j \in \{1, \ldots, n-1\}$, select the head:

$$h_j = \arg \max_{i\in\{0,\ldots,n-1\}\setminus\{j\}} \mathcal{S}_{\text{arc}}[i, j]$$

2. Let $\mathcal{T} = \{(h_j, j) \mid j = 1, \ldots, n-1\}$ be the set of selected arcs.
3. If $\mathcal{T}$ forms a valid tree (i.e., no cycles), return $\mathcal{T}$.
4. Otherwise, for each cycle $C \subseteq \mathcal{T}$:
   (a) Contract the cycle C into a single supernode v_C.
   (b) For each edge (i, j) where $i \notin C$ and $j \in C$, define adjusted score:

$$\tilde{\mathcal{S}}_{\text{arc}}[i, v_C] = \mathcal{S}_{\text{arc}}[i, j] - \mathcal{S}_{\text{arc}}[h_j, j] + \max_{k\in C} \mathcal{S}_{\text{arc}}[h_k, k]$$

   (c) Re-run the algorithm recursively on the contracted graph.
   (d) Expand the cycle C, replacing v_C with the original nodes and recovering the incoming arc to the cycle that preserves the maximal score.
5. Return the resulting tree $\mathcal{T}$ with maximum total arc score.