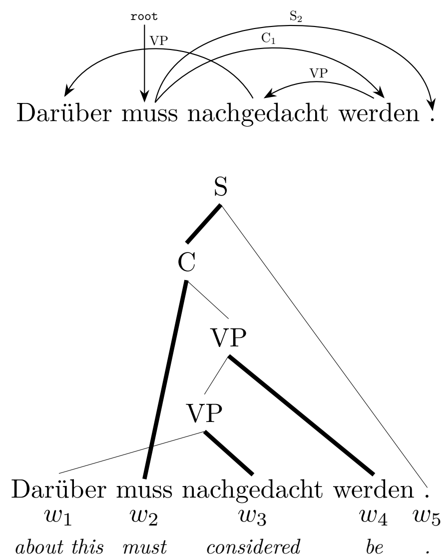


Figure 1.1: Example of a discontinuous constituent tree for the German sentence *'Darüber muss nachgedacht werden.'* ('this must be considered.'). Bold lines indicate head words.

1 Preliminaries: Constituency Trees

2 Encoder-Decoder Setup

Let $N = \{1, \dots, n\}$ be an index set. Given an input sentence $w = (w_i)_{i \in N}$ we generate a sequence of *embeddings* $\boldsymbol{\omega} = (\boldsymbol{\omega}_i)_{i \in N}$ where

$$\boldsymbol{\omega}_i = \mathbf{WordEmbed}(w_i) \oplus \mathbf{CharEmbed}(w_i) \oplus \mathbf{BertEmbed}(w_i)$$

Encoder: feed embeddings through a multi-layer bi-directional LSTM with skip-connections and dropout:

$$\mathbf{e} = (\mathbf{e}_i)_{i=0,\dots,n} = \mathbf{BiLSTM}(\boldsymbol{\omega})$$

Decoder: feed embeddings through a single-layer uni-directional LSTM with dropout:

$$\mathbf{d} = (\mathbf{d}_i)_{i=1,\dots,n} = \mathbf{LSTM}(\boldsymbol{\omega})$$

3 Bi-affine Attention Mechanism

Goal: given dependency (child) w_j, choose the most probable head (parent) e_i. We feed $\mathbf{e}$ and $\mathbf{d}$ through MLPs to produce sequences $(\mathbf{e}^{\mathrm{arc}}, \mathbf{d}^{\mathrm{arc}})$ of dimension-reduced vectors. These are fed into a bi-affine layer which produces latent features $\mathbf{v}^{\mathrm{arc}}$ that are then fed into an attention layer, resulting in logits corresponding to strength of an arc.

$$\mathbf{e}^{\mathrm{arc}} = \mathbf{MLP}^{\mathrm{arc}}_{\mathrm{enc}}(\mathbf{e}); \qquad \mathbf{d}^{\mathrm{arc}} = \mathbf{MLP}^{\mathrm{arc}}_{\mathrm{dec}}(\mathbf{d}) \tag{3.1}$$

$$\mathbf{v}^{\mathrm{arc}}_{i,j} = \mathbf{BiAff}^{\mathrm{arc}}(\mathbf{e}^{\mathrm{arc}}_i, \mathbf{d}^{\mathrm{arc}}_j) \tag{3.2}$$

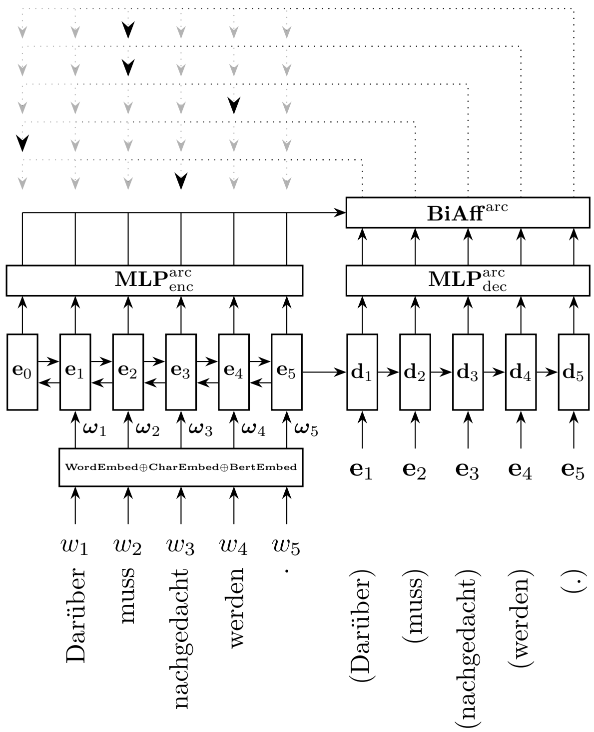
$$:= \mathbf{e}^{\mathrm{arc}\,\mathsf{T}}_i \mathbf{U}^{\mathrm{arc}}_{\text{h-d}} \mathbf{d}^{\mathrm{arc}}_i + \mathbf{e}^{\mathrm{arc}\,\mathsf{T}}_i \mathbf{U}^{\mathrm{arc}}_{\text{h-h}} \mathbf{e}^{\mathrm{arc}}_i + \mathbf{d}^{\mathrm{arc}\,\mathsf{T}}_i \mathbf{U}^{\mathrm{arc}}_{\text{d-d}} \mathbf{d}^{\mathrm{arc}}_i \tag{3.3}$$

$$+ U^{\mathrm{arc}}_{\mathrm{h}} \mathbf{e}^{\mathrm{arc}}_i + U^{\mathrm{arc}}_{\mathrm{d}} \mathbf{d}^{\mathrm{arc}}_i + \mathbf{u}^{\mathrm{arc}}_{\mathrm{bias}} \tag{3.4}$$

$$s^{\mathrm{arc}}_{i,j} = \mathbf{u}^{\mathrm{arc}\,\mathsf{T}}_{\mathrm{agg}} \tanh(\mathbf{v}^{\mathrm{arc}}_{i,j}) \tag{3.5}$$

Fixing dependency j, the vector $\mathbf{softmax}(\mathbf{s}_{:,j})$ can be interpreted as an estimated probability distribution over potential heads.

$$\hat{p}^{\mathrm{arc}}(w_i \mid y_{<j}, w) = \mathbf{softmax}(\mathbf{s}^{\mathrm{arc}}_{:,j})_i.$$

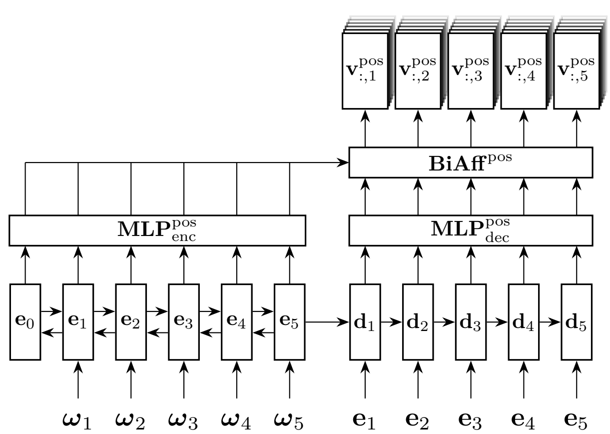


4 Bi-affine Classifier for Attachment Order, POS and Morphology

Attachment order, POS and morphologies are predicted via a classification. We use a bi-affine classifier which allows us to model probabilities *conditional* on arcs, and thus use structural cues in addition to encoder/decoder states to better capture the complexity of the language. The encoder and decoder are *shared* across the tasks.

For example suppose we would like to predict the part of speech $c \in \mathcal{C}$ for word w_j conditional on its parent being w_i.

$$
\begin{aligned}
\mathbf{e}^{\text{pos}} &= \mathbf{MLP}^{\text{pos}}_{\text{enc}}(\mathbf{e}); \qquad \mathbf{d}^{\text{pos}} = \mathbf{MLP}^{\text{pos}}_{\text{dec}}(\mathbf{d}) && (4.1)\\
\mathbf{v}^{\text{pos}}_{i,j} &= \mathbf{BiAff}^{\text{pos}}(\mathbf{e}^{\text{pos}}_i, \mathbf{d}^{\text{pos}}_j) && (4.2)\\
&:= \mathbf{e}_i^{\text{pos}\mathsf{T}} \mathbf{U}^{\text{pos}}_{\text{h-d}} \mathbf{d}^{\text{pos}}_i + \mathbf{e}_i^{\text{pos}\mathsf{T}} \mathbf{U}^{\text{pos}}_{\text{h-h}} \mathbf{e}^{\text{pos}}_i + \mathbf{d}_i^{\text{pos}\mathsf{T}} \mathbf{U}^{\text{pos}}_{\text{d-d}} \mathbf{d}^{\text{pos}}_i && (4.3)\\
&\quad + U^{\text{pos}}_{\text{h}} \mathbf{e}^{\text{pos}}_i + U^{\text{pos}}_{\text{d}} \mathbf{d}^{\text{pos}}_i + \mathbf{u}^{\text{pos}}_{\text{bias}} && (4.4)\\
\hat{p}^{\text{pos}}(c \mid w_i; y_{<j}, w) &= \mathbf{softmax}(\mathbf{v}^{\text{pos}}_{i,j})_c && (4.5)
\end{aligned}
$$



Hi there