Multitask Pointer Network for Discontinuous Constituent Parsing

An application to the German language

James Yu
jby21@cam.ac.uk

Faculty of Economics
University of Cambridge

15th January 2024

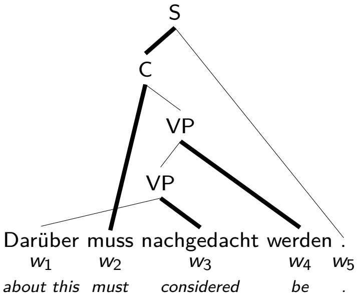
Why use a neural network to model grammar?

Linguistic preliminaries: constituent trees

Let $w = (w_1, \ldots, w_N)$ be a sentence.

Definition

- A *constituent tree* is a rooted tree whose leaves are the words $(w_i)_{i=1}^N$ and internal nodes are constituents satisfying some constraints.
- A *constituent* is a triple $(Z, \mathcal{Y}, h)$ containing, respectively, its label, yield, and lexical head.
- A constituent is *discontinuous* if its yield is not contiguous.

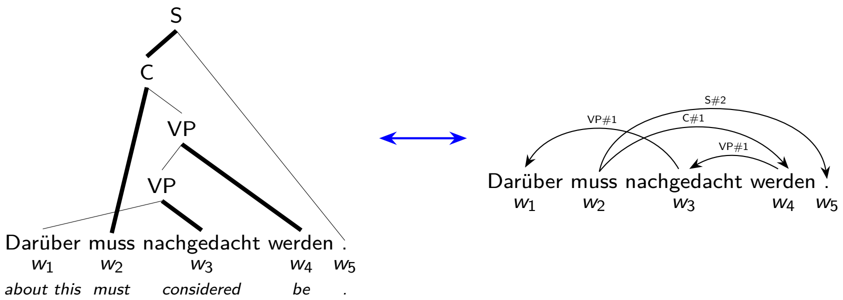


Reduction to dependency parsing

Definition

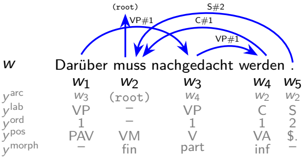
A *dependency tree* is a rooted tree spanning the words in the sentence $(w_i)_{i=1}^N$. Each edge is labelled and connects a *head word* (parent) to a *dependency* (child).

Fernández-González and Martins (2015) show that constituent trees are isomorphic to dependency trees in which the edges contain information about constituent labels and attachment order.



Mathematical formalisation

- Regressor: sentence $(w_i)_{i=1}^N$.
- Regressand: dependency tree, parts of speech and morphology.
- Bottom-up approach: think of *arcs* going from every child w_i to its parent $y_i^{\text{arc}} \in w \setminus w_i$.



Denote the regressand by $y = (y_i)_{i=1}^N$ where $y_i = (y_i^{\text{arc}}, y_i^{\text{lab}}, y_i^{\text{ord}}, y_i^{\text{pos}}, y_i^{\text{morph}})$ and with mild abuse of notation let $y_{<i} = (y_1, \dots, y_i)$ for each $i = 2, \dots, N$ and $y_{<1} = 0$.

Assumption

For each $i = 1, \dots, N$, the random variables $y_i^{\text{lab}}, y_i^{\text{ord}}, y_i^{\text{pos}}$ and y_i^{morph} are mutually independent conditional on y_i^{arc}, $y_{<i}$ and w.

We can decompose the conditional probability of y given w:

$$\begin{aligned}
p_w(y) &= \prod_{i=1}^N p_w(y_i \mid y_{<i}) \\
&= \prod_{i=1}^N \Big\{ p_w(y_i^{\text{arc}} \mid y_{<i}) p_w(y_i^{\text{lab}} \mid y_i^{\text{arc}}, y_{<i}) \\
&\qquad \cdot p_w(y_i^{\text{ord}} \mid y_i^{\text{arc}}, y_{<i}) p_w(y_i^{\text{pos}} \mid y_i^{\text{arc}}, y_{<i}) p_w(y_i^{\text{morph}} \mid y_i^{\text{arc}}, y_{<i}) \Big\}.
\end{aligned}$$

Encoder-decoder setup

Given an input sentence $w = (w_i)_{i=1}^{N}$ we generate *embeddings* $\boldsymbol{\omega} = (\omega_i)_{i=1}^{N}$, where

$$\omega_i = \textbf{WordEmbed}(w_i) \oplus \textbf{CharEmbed}(w_i) \oplus \textbf{BertEmbed}(w_i).$$

- **CharEmbed** is implemented using a CNN á la Chiu and Nichols (2016).
- BERT model pre-trained on German text by Chan et al. (2020).

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

$$\mathbf{e} = (\mathbf{e}_i)_{i=0,\ldots,n} = \textbf{BiLSTM}(\omega).$$

($\mathbf{e}_0$ represents the root pseudo-node.)

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

$$\mathbf{d} = (\mathbf{d}_i)_{i=1,\ldots,n} = \textbf{LSTM}(\omega).$$

The pointer network: bi-affine attention mechanism

Taking inspiration from Dozat and Manning (2016) and Vinyals et al. (2015).

- Obtain dimension-reduced representations:

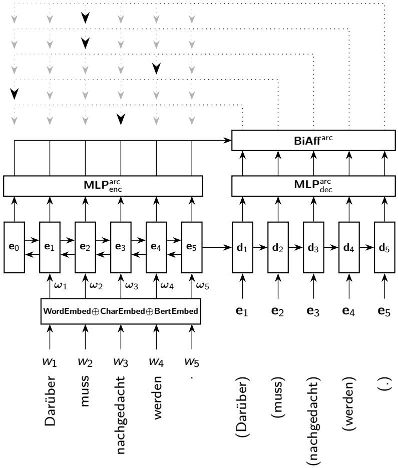
$$\mathbf{e}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{enc}}(\mathbf{e}); \quad \mathbf{d}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{dec}}(\mathbf{d}).$$

- Obtain *latent features* $\mathbf{v}^{\text{arc}}$:

$$\begin{aligned}\mathbf{v}^{\text{arc}}_{i,j} &= \mathbf{BiAff}^{\text{arc}}(\mathbf{e}^{\text{arc}}_i, \mathbf{d}^{\text{arc}}_j) \\ &:= \mathbf{e}^{\text{arc}\,\mathsf{T}}_i \mathbf{U}^{\text{arc}}_{\text{h-d}} \mathbf{d}^{\text{arc}}_i \\ &\quad + \mathbf{e}^{\text{arc}\,\mathsf{T}}_i \mathbf{U}^{\text{arc}}_{\text{h-h}} \mathbf{e}^{\text{arc}}_i + \mathbf{d}^{\text{arc}\,\mathsf{T}}_i \mathbf{U}^{\text{arc}}_{\text{d-d}} \mathbf{d}^{\text{arc}}_i \\ &\quad + U^{\text{arc}}_{\text{h}} \mathbf{e}^{\text{arc}}_i + U^{\text{arc}}_{\text{d}} \mathbf{d}^{\text{arc}}_i + \mathbf{u}^{\text{arc}}_{\text{bias}}.\end{aligned}$$

- Obtain *attention logits*:

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



Fixing child w_j, the vector $\mathbf{softmax}(s^{\text{arc}}_{:,j})$ can be interpreted as an estimated probability distribution over potential parents.

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

Chan, B., Möller, T., Pietsch, M., & Soni, T. (2020). *German BERT (bert-base-german-cased)*. Retrieved September 1, 2023, from https://huggingface.co/bert-base-german-cased

Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the association for computational linguistics*, *4*, 357–370.

Dozat, T., & Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing. *International Conference on Learning Representations*.

Fernández-González, D., & Martins, A. F. (2015). Parsing as reduction. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1523–1533.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, *28*.