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

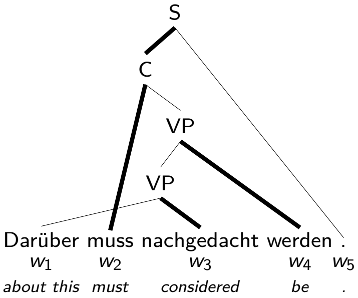
15th January 2024

Why use a neural network to model grammar?

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

Definition

- A *constituent tree* is a rooted tree whose leaves are the words $(w_i)_{i=1}^{L}$ 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.

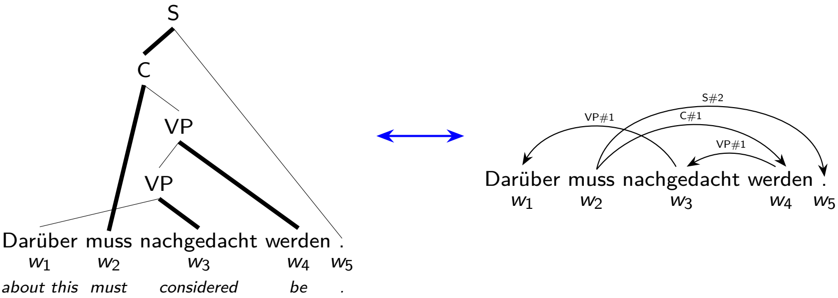


Reduction to dependency parsing

Definition

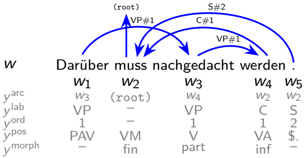
A *dependency tree* is a rooted tree spanning the words in the sentence $(w_i)_{i=1}^{L}$. Each edge is labelled and connects a *head word* (parent) to a *dependency* (child).

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 $(w_i)_{i=1}^{L}$.
- 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 $y = (y_i)_{i=1}^{L}$ 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 $y_{<i} = (y_1, \ldots, y_i)$ for each $i = 2, \ldots, L$ and $y_{<1} = 0$.

Assumption

For each $i = 1, \ldots, L$, 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:

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

Sequence-to-sequence setup

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

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

- **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:

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

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

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

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

The pointer network: bi-affine attention mechanism

Drawing on Dozat and Manning (2016) and Vinyals et al. (2015).

- Obtain dimension-reduced representations:

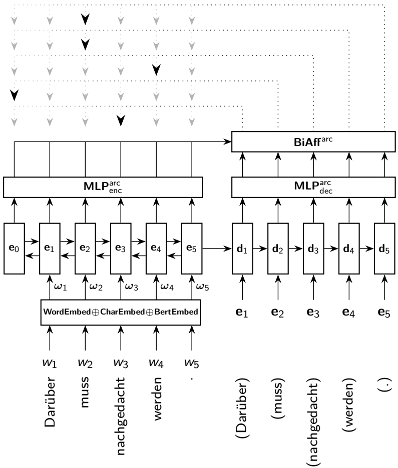
$$\mathbf{e}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{enc}}(\mathbf{e}); \quad \mathbf{d}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{dec}}(\mathbf{d}).$$

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

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

- Obtain *attention logits*:

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



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

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

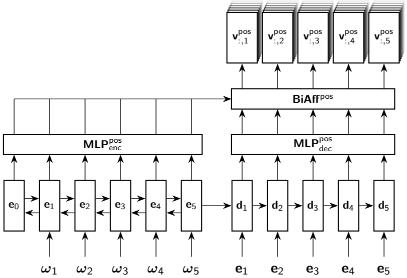
Part-of-speech and morphology: quadratic classifier

Drawing on Dozat and Manning (2016).

- Encoder and decoder are shared across tasks. The remaining network is separately trained.
- Classification via a bi-affine architecture allows modelling of class probabilities conditional on arcs.

Example: part-of-speech classification.

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



- Obtain class *logits* $\mathbf{v}^{\text{pos}}$:

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

Fixing child w_j, the vector $\mathbf{softmax}(\mathbf{v}^{\text{pos}}_{i,j})$ can be interpreted as an estimated probability distribution over its parts-of-speech conditional on having an arc to w_i:

$$\hat{p}^{\text{pos}}(c \mid w_i, y_{<j}, w) = \mathbf{softmax}(\mathbf{v}^{\text{pos}}_{i,j})_c.$$

Inference and Training

Inference

Given $w = (w_i)_{i=1}^L$, estimate $y = (y_i)_{i=1}^L$ via *maximum likelihood*:

$$\hat{y} = \arg\max_y \left\{ \prod_{i=1}^{L} \left\{ \hat{p}_w(y_i^{\text{arc}} \mid y_{<i}) \hat{p}_w(y_i^{\text{lab}} \mid y_i^{\text{arc}}, y_{<i}) \cdot \hat{p}_w(y_i^{\text{ord}} \mid y_i^{\text{arc}}, y_{<i}) \hat{p}_w(y_i^{\text{pos}} \mid y_i^{\text{arc}}, y_{<i}) \hat{p}_w(y_i^{\text{morph}} \mid y_i^{\text{arc}}, y_{<i}) \right\} \right\}.$$

The feasible region is very large, so the maximisation is approximated via *beam search*.

Training

Minimise the *cross-entropy* between $\hat{p}$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

$$\min_{\hat{p}} \left\{ -\sum_{n=1}^{N} \log \prod_{i=1}^{L_n} \left\{ \hat{p}_w(y_i^{n\text{arc}} \mid y_{<i}^n) \hat{p}_w(y_i^{n\text{lab}} \mid y_i^{n\text{arc}}, y_{<i}^n) \cdot \hat{p}_w(y_i^{n\text{ord}} \mid y_i^{n\text{arc}}, y_{<i}^n) \hat{p}_w(y_i^{n\text{pos}} \mid y_i^{n\text{arc}}, y_{<i}^n) \hat{p}_w(y_i^{n\text{morph}} \mid y_i^{n\text{arc}}, y_{<i}^n) \right\} \right\}$$

$$= \min_{\hat{p}} \left\{ \text{loss}^{\text{arc}} + \text{loss}^{\text{lab}} + \text{loss}^{\text{ord}} + \text{loss}^{\text{pos}} + \text{loss}^{\text{morph}} \right\}.$$

This is approximated via *SGD* with *Nesterov momentum*.

Performance metrics

The TIGER treebank is a widely-used corpus of ∼50 000 constituency trees. Its main textual basis is the *Frankfurter Rundschau*.

- 97 % of the dataset is usable as training examples, with train/dev/test split of 80/10/10 %.

Model	**F1**	**Disc. F1**
F.-González & G.-Rodríguez (2022)	89.8	71.0
Corro (2020)	**90.0**	62.1
Chen & Komachi (2023)	89.6	70.9
This work	89.5	**82.2**

Table: Comparison of overall F1-score (%) and F1-score measured only on discontinuous constituents (disc. F1). Calculated using `disco-dop` (Van Cranenburgh et al., 2016) as standard practice. All models configured with BERT.

Model	`pos`	`morph` (avr)
Kondratyuk et al. (2018)	98.58	98.97
Müller et al. (2013)	98.20	98.27
Schnabel & Schütze (2014)	97.50	97.76
This work	**99.16**	**99.54**

Table: Comparison of part-of-speech (`pos`) and morphology accuracies (%). Morphology accuracies are the average of accuracies for `case`, `degree`, `gender`, `mood`, `number`, `person` and `tense`.

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

Chen, Z., & Komachi, M. (2023). Discontinuous combinatory constituency parsing. *Transactions of the Association for Computational Linguistics*, *11*, 267–283.

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

Corro, C. (2020). Span-based discontinuous constituency parsing: A family of exact chart-based algorithms with time complexities from o (n^ 6) down to o (n^ 3). *Empirical Methods in Natural Language Processing*.

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

Eger, S., Gleim, R., & Mehler, A. (2016). Lemmatization and morphological tagging in german and latin: A comparison and a survey of the state-of-the-art. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1507–1513.

Fernández-González, D., & Gómez-Rodriguez, C. (2022). Multitask pointer network for multi-representational parsing. *Knowledge-Based Systems*, *236*, 107760.

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.

Gleim, R., Eger, S., Mehler, A., Uslu, T., Hemati, W., Lücking, A., Henlein, A., Kahlsdorf, S., & Hoenen, A. (2019). A practitioner's view: A survey and comparison of lemmatization and morphological tagging in german and latin. *Journal of Language Modelling*, *7*(1), 1–52.

Kondratyuk, D., Gavenčiak, T., Straka, M., & Hajič, J. (2018). Lemmatag: Jointly tagging and lemmatizing for morphologically-rich languages with brnns. *arXiv preprint arXiv:1808.03703*.

Müller, T., Schmid, H., & Schütze, H. (2013). Efficient higher-order crfs for morphological tagging. *Proceedings of the 2013 conference on empirical methods in natural language processing*, 322–332.

Schnabel, T., & Schütze, H. (2014). FLORS: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, *2*, 15–26.

Van Cranenburgh, A., Scha, R., & Bod, R. (2016). Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, *4*(1), 57–111.

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

Appendix: Mathematical justification of bi-affine classifier

Notation has been simplified for presentational clarity. Take a single class c and suppose

$$\begin{bmatrix}\mathbf{e}_i\\ \mathbf{d}_j\end{bmatrix} \Bigg|\, c \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_c\\ \phi_c\end{bmatrix}, \begin{bmatrix}\mathbf{A}_c & \mathbf{Q}_c^{\mathsf{T}}\\ \mathbf{Q}_c & \mathbf{B}_c\end{bmatrix}^{-1}\right).$$

The conditional log-likelihood of c is the following affine quadratic form:

$$\begin{aligned}
v_{i,j}^c = \log p(\mathbf{e}_i \mid c, \mathbf{d}_j) &= k^c - \frac{1}{2}((\mathbf{e}_i - \boldsymbol{\mu}_c) - \mathbf{A}_c^{-1}\mathbf{Q}_c^{\mathsf{T}}(\mathbf{d}_j - \phi))^{\mathsf{T}}\mathbf{A}_c(\mathbf{e}_i - \boldsymbol{\mu}_c) - \mathbf{A}_c^{-1}\mathbf{Q}_c^{\mathsf{T}}(\mathbf{d}_j - \phi) \\
&= \boxed{\mathbf{e}_i^{\mathsf{T}}\mathbf{Q}_c^{\mathsf{T}}\mathbf{d}_j} - \boxed{\frac{1}{2}\mathbf{e}_i^{\mathsf{T}}\mathbf{A}_c\mathbf{e}_i} - \boxed{\frac{1}{2}\mathbf{d}_j^{\mathsf{T}}\mathbf{Q}_c\mathbf{A}_c^{-1}\mathbf{Q}_c^{\mathsf{T}}\mathbf{d}_j} + \boxed{\left(\boldsymbol{\mu}^{\mathsf{T}}\mathbf{A}_c - \phi^{\mathsf{T}}\mathbf{Q}_c\right)\mathbf{e}_i} + \boxed{(\phi^{\mathsf{T}}\mathbf{Q}_c\mathbf{A}_c^{-1}\mathbf{Q}_c^{\mathsf{T}} - \boldsymbol{\mu}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}})\mathbf{d}_j} \\
&\quad + \boxed{\boldsymbol{\mu}^{\mathsf{T}}\mathbf{Q}_c^{\mathsf{T}}\phi - \frac{1}{2}\phi^{\mathsf{T}}\mathbf{Q}_c\mathbf{A}_c^{-1}\mathbf{Q}^{\mathsf{T}}\phi + k_c} \\
&= c\text{th row of } \mathbf{e}_i^{\mathsf{T}}\mathbf{U}_{\text{h-d}}\mathbf{d}_j + \mathbf{e}_i^{\mathsf{T}}\mathbf{U}_{\text{h-h}}\mathbf{e}_i + \mathbf{d}_j^{\mathsf{T}}\mathbf{U}_{\text{d-d}}\mathbf{d}_j + U_{\text{h}}\mathbf{e}_i + U_{\text{d}}\mathbf{d}_j + \mathbf{u}_{\text{bias}}
\end{aligned}$$

Memory-efficient implementation: under conditional normality, the affine quadratic transformation can be computed as

$$v_{i,j}^c = k^c + \left\| W_h^c\mathbf{e}_i + W_d^c\mathbf{d}_j + \mathbf{w}^c \right\|_2$$