# *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 German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.

# *Why use a neural network to model German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.
- Free-order languages like German contain many grammatical discontinuities.

# *Why use a neural network to model German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.
- Free-order languages like German contain many grammatical discontinuities.
- Discontinuous representations introduce computational complexity but can be more valuable for downstream applications.

## *Why use a neural network to model German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.
- Free-order languages like German contain many grammatical discontinuities.
- Discontinuous representations introduce computational complexity but can be more valuable for downstream applications.
- Grammar-less neural network-based models have continually pushed the state of the art.

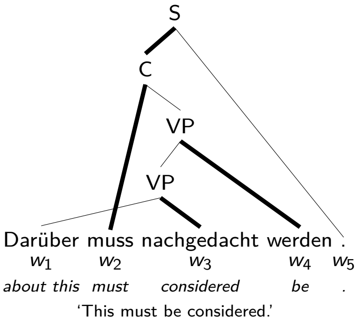
## *Why use a neural network to model German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.
- Free-order languages like German contain many grammatical discontinuities.
- Discontinuous representations introduce computational complexity but can be more valuable for downstream applications.
- Grammar-less neural network-based models have continually pushed the state of the art.
- My model is based on Fernández-González and Gómez-Rodríguez (2022), who propose an architecture based on pointer neural networks in a multi-task setting.

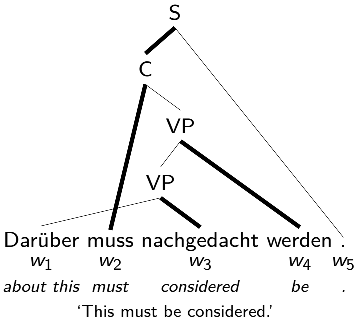
# *Why use a neural network to model German grammar?*

- Constituent trees are a syntactic formalism representing phrasal hierarchy in a sentence.
- Free-order languages like German contain many grammatical discontinuities.
- Discontinuous representations introduce computational complexity but can be more valuable for downstream applications.
- Grammar-less neural network-based models have continually pushed the state of the art.
- My model is based on Fernández-González and Gómez-Rodríguez (2022), who propose an architecture based on pointer neural networks in a multi-task setting.
- I achieve state-of-the-art performance across several metrics.

# *Linguistic preliminaries: constituent trees*



# *Linguistic preliminaries: constituent trees*




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

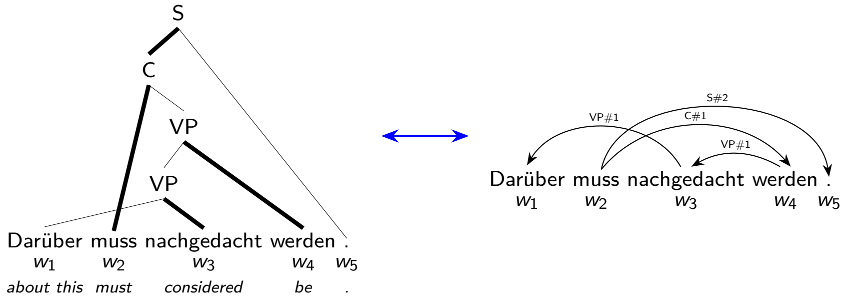
A *dependency tree* is a rooted tree spanning the words in the sentence $(w_i)_{i=1}^{L}$. Each edge is labelled and connects a *head word* (parent) to a *dependency* (child).

# *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 $(w_i)_{i=1}^{L}$.

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *Mathematical formalisation*

- Regressor: sentence $(w_i)_{i=1}^{L}$.
- Regressand: dependency tree, parts of speech and morphology.

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

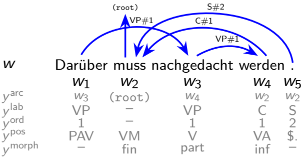
# *Mathematical formalisation*

- 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 $y_i^{\text{arc}} \in w \setminus w_i$.

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *Mathematical formalisation*

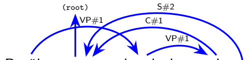
- Regressor: sentence $(w_i)_{i=1}^{L}$.
- Regressand: dependency tree, parts of speech and morphology.
- Bottom-up approach: think of *arcs* going from every child $w_i$ to its parent $y_i^{\text{arc}} \in w \setminus w_i$.




[1]For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# 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 $y_i^{\text{arc}} \in w \setminus w_i$.



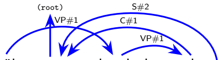

| | | | | | |
|---|---|---|---|---|---|
| $w$ | Darüber | muss | nachgedacht | werden | . |
| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |
| $y^{\text{arc}}$ | $w_3$ | (root) | $w_4$ | $w_2$ | $w_2$ |
| $y^{\text{lab}}$ | VP | – | VP | C | S |
| $y^{\text{ord}}$ | 1 | – | 1 | 1 | 2 |
| $y^{\text{pos}}$ | PAV | VM | V | VA | $. |
| $y^{\text{morph}}$ | – | fin | part | inf | – |

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

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *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 $y_i^{\text{arc}} \in w \setminus w_i$.




| $w$ | Darüber | muss | nachgedacht | werden | . |
|---|---|---|---|---|---|
| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |
| $y^{\text{arc}}$ | $w_3$ | (root) | $w_4$ | $w_2$ | $w_2$ |
| $y^{\text{lab}}$ | VP | – | VP | C | S |
| $y^{\text{ord}}$ | 1 | – | 1 | 1 | 2 |
| $y^{\text{pos}}$ | PAV | VM | V | VA | $. |
| $y^{\text{morph}}$ | – | fin | part | inf | – |

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 $y_{<i} = (y_1, \dots, y_{i-1})$ for each $i = 2, \dots, L$ and $y_{<1} = 0$.

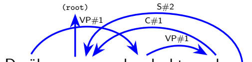
### *Assumption*

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

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *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 $y_i^{\text{arc}} \in w \setminus w_i$.




| | | | | | |
|---|---|---|---|---|---|
| $w$ | Darüber | muss | nachgedacht | werden | . |
| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |
| $y^{\text{arc}}$ | $w_3$ | (root) | $w_4$ | $w_2$ | $w_2$ |
| $y^{\text{lab}}$ | VP | – | VP | C | S |
| $y^{\text{ord}}$ | 1 | – | 1 | 1 | 2 |
| $y^{\text{pos}}$ | PAV | VM | V | VA | $. |
| $y^{\text{morph}}$ | – | fin | part | inf | – |

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

### *Assumption*

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

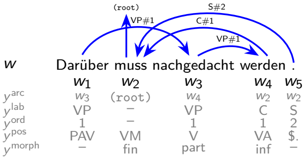
We can decompose the conditional probability of $y$ given $w$:[1]

$$p_w(y) = \prod_{i=1}^{L} p_w(y_i \mid y_{<i})$$

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *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 $y_{<i} = (y_1, \ldots, y_{i-1})$ for each $i = 2, \ldots, L$ and $y_{<1} = 0$.

### *Assumption*

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

We can decompose the conditional probability of $y$ given $w$:[1]

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

[1] For notational simplicity we have written $p_w(\cdot) \equiv p(\cdot \mid w)$ and $p_w(\cdot \mid \cdot) \equiv p(\cdot \mid \cdot, w)$.

# *Sequence-to-sequence setup*

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

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

## *Sequence-to-sequence setup*

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

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

- **WordEmbed** is a simple lookup table.

# *Sequence-to-sequence setup*

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

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

- **WordEmbed** is a simple lookup table.
- **CharEmbed** is implemented using a CNN á la Chiu and Nichols (2016).

# *Sequence-to-sequence setup*

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

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

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

# *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

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

- **WordEmbed** is a simple lookup table.
- **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:

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

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

# *Sequence-to-sequence setup*

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

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

- **WordEmbed** is a simple lookup table.
- **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:

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

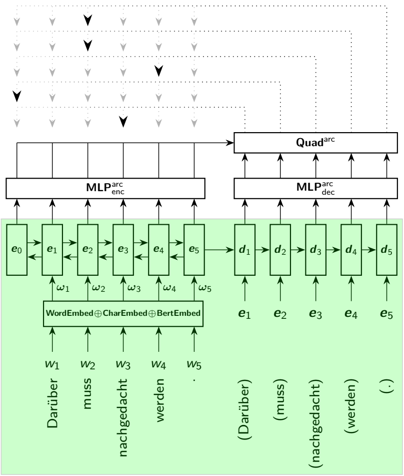
($\boldsymbol{e}_0$ represents the root pseudo-node.)

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

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

# *The pointer network: quadratic attention mechanism*

*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

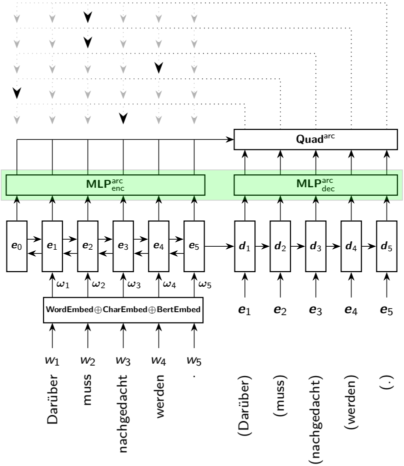


# *The pointer network: quadratic attention mechanism*

*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

- Obtain dimension-reduced representations:

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



# *The pointer network: quadratic attention mechanism*

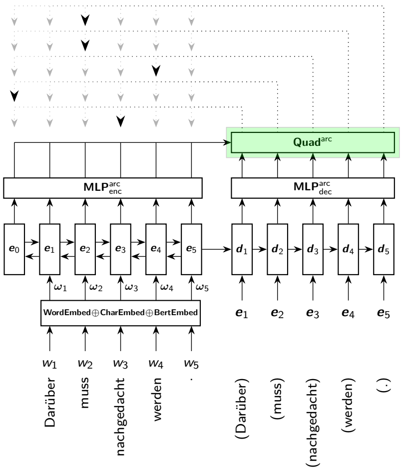
*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

- Obtain dimension-reduced representations:

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

- Obtain *latent features* $\boldsymbol{v}^{\text{arc}}_{i,j}$:

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



# The pointer network: quadratic attention mechanism

*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

- Obtain dimension-reduced representations:

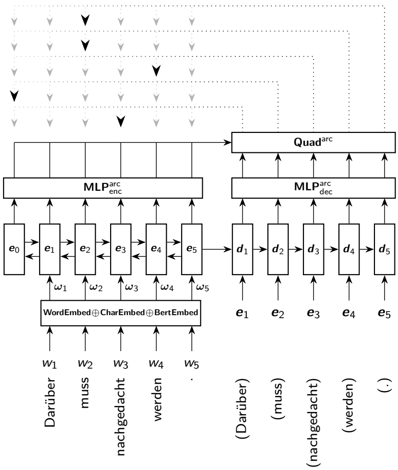
$$\boldsymbol{e}^{\text{arc}} = \textbf{MLP}^{\text{arc}}_{\text{enc}}(\boldsymbol{e}); \quad \boldsymbol{d}^{\text{arc}} = \textbf{MLP}^{\text{arc}}_{\text{dec}}(\boldsymbol{d}).$$

- Obtain *latent features* $\boldsymbol{v}^{\text{arc}}_{i,j}$:

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

- Obtain *attention logits* $s^{\text{arc}}_{i,j}$:

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



*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

- Obtain dimension-reduced representations:

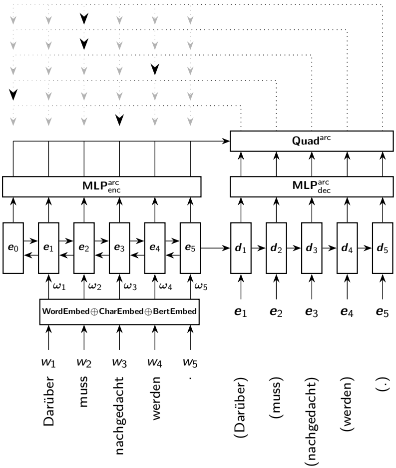
$$\boldsymbol{e}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{enc}}(\boldsymbol{e}); \quad \boldsymbol{d}^{\text{arc}} = \mathbf{MLP}^{\text{arc}}_{\text{dec}}(\boldsymbol{d}).$$

- Obtain *latent features* $\boldsymbol{v}^{\text{arc}}_{i,j}$:

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

- Obtain *attention logits* $s^{\text{arc}}_{i,j}$:

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




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

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

# *The pointer network: quadratic attention mechanism*

*Building on Dozat and Manning (2016)'s bi-affine mechanism and drawing from Vinyals et al. (2015).*

- Obtain dimension-reduced representations:

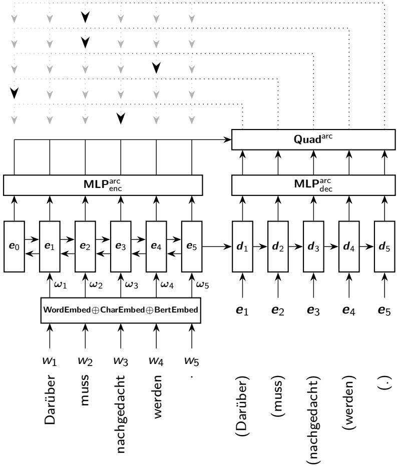
$$\boldsymbol{e}^{\text{arc}} = \textbf{MLP}^{\text{arc}}_{\text{enc}}(\boldsymbol{e}); \quad \boldsymbol{d}^{\text{arc}} = \textbf{MLP}^{\text{arc}}_{\text{dec}}(\boldsymbol{d}).$$

- Obtain *latent features* $\boldsymbol{v}^{\text{arc}}_{i,j}$:

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

- Obtain *attention logits* $s^{\text{arc}}_{i,j}$:

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



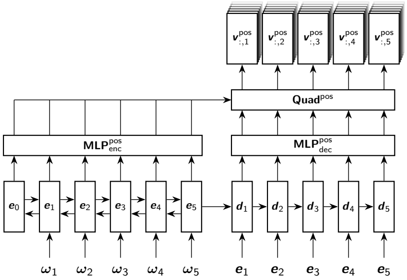

- Fixing child $w_j$, the vector $\textbf{softmax}(\boldsymbol{s}^{\text{arc}}_{:,j})$ can be interpreted as a probability distribution over potential parents:

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

# *Classification tasks: quadratic classifier*

*Building on Dozat and Manning (2016).*

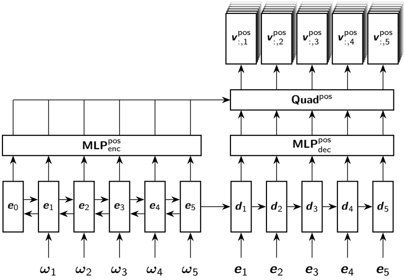
- Encoder and decoder are shared across tasks. The remaining network is separately trained.



# *Classification tasks: quadratic classifier*

*Building on Dozat and Manning (2016).*

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



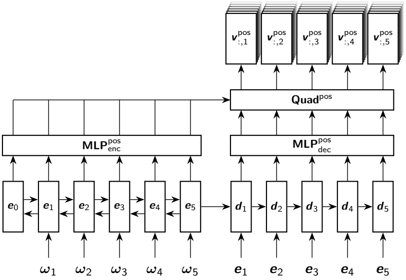
## *Classification tasks: quadratic classifier*

*Building on Dozat and Manning (2016).*

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

**Example:** part-of-speech classification.

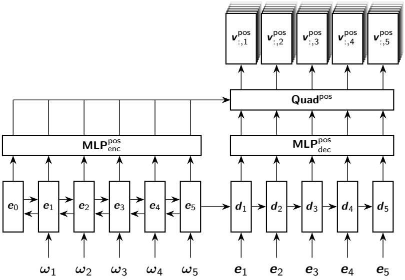
$$\boldsymbol{e}^{\text{pos}} = \mathbf{MLP}^{\text{pos}}_{\text{enc}}(\boldsymbol{e}); \quad \boldsymbol{d}^{\text{pos}} = \mathbf{MLP}^{\text{pos}}_{\text{dec}}(\boldsymbol{d}).$$



# *Classification tasks: quadratic classifier*

*Building on Dozat and Manning (2016).*

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




**Example:** part-of-speech classification.

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

- Obtain class *logits* $\boldsymbol{v}^{\text{pos}}_{i,j}$:

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

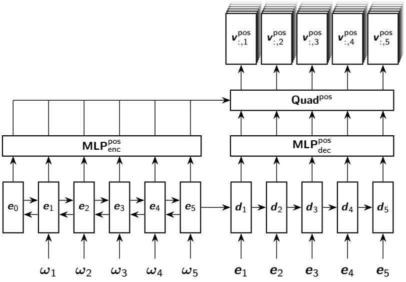
# *Classification tasks: quadratic classifier*

*Building on Dozat and Manning (2016).*

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

**Example:** part-of-speech classification.

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




- Obtain class *logits* $\boldsymbol{v}^{\text{pos}}_{i,j}$:

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

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

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

# *Inference and Training*

## *Training*

We find $\hat{p}$ using *SGD* with *Nesterov momentum*, finding a suitably low *cross-entropy* between the model $p$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

# *Inference and Training*

## *Training*

We find $\hat{p}$ using *SGD* with *Nesterov momentum*, finding a suitably low *cross-entropy* between the model $p$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

$$-\sum_{n=1}^{N} \log \prod_{i=1}^{L_n} \Big\{ p_w(y_i^{n\mathrm{arc}} \mid y_{<i}^n) p_w(y_i^{n\mathrm{lab}} \mid y_i^{n\mathrm{arc}}, y_{<i}^n) p_w(y_i^{n\mathrm{ord}} \mid y_i^{n\mathrm{arc}}, y_{<i}^n) \cdot p_w(y_i^{n\mathrm{pos}} \mid y_i^{n\mathrm{arc}}, y_{<i}^n) p_w(y_i^{n\mathrm{morph}} \mid y_i^{n\mathrm{arc}}, y_{<i}^n)$$

# Inference and Training

## Training

We find $\hat{p}$ using *SGD* with *Nesterov momentum*, finding a suitably low *cross-entropy* between the model $p$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

$$-\sum_{n=1}^{N} \log \prod_{i=1}^{L_n} \Big\{ p_w(y_i^{n\text{arc}} \mid y_{<i}^n) p_w(y_i^{n\text{lab}} \mid y_i^{n\text{arc}}, y_{<i}^n) p_w(y_i^{n\text{ord}} \mid y_i^{n\text{arc}}, y_{<i}^n) \\ \cdot\, p_w(y_i^{n\text{pos}} \mid y_i^{n\text{arc}}, y_{<i}^n) p_w(y_i^{n\text{morph}} \mid y_i^{n\text{arc}}, y_{<i}^n)$$

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

# Inference and Training

## Training

We find $\hat{p}$ using *SGD* with *Nesterov momentum*, finding a suitably low *cross-entropy* between the model $p$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

$$\begin{aligned}
&-\sum_{n=1}^{N} \log \prod_{i=1}^{L_n} \Big\{ p_w(y_i^{n\text{arc}} \mid y_{<i}^n) p_w(y_i^{n\text{lab}} \mid y_i^{n\text{arc}}, y_{<i}^n) p_w(y_i^{n\text{ord}} \mid y_i^{n\text{arc}}, y_{<i}^n) \\
&\qquad\qquad \cdot p_w(y_i^{n\text{pos}} \mid y_i^{n\text{arc}}, y_{<i}^n) p_w(y_i^{n\text{morph}} \mid y_i^{n\text{arc}}, y_{<i}^n) \\
&= \min_p \Big\{ \text{loss}^{\text{arc}} + \text{loss}^{\text{lab}} + \text{loss}^{\text{ord}} + \text{loss}^{\text{pos}} + \text{loss}^{\text{morph}} \Big\}.
\end{aligned}$$

## Inference

Given $w = (w_i)_{i=1}^L$, estimate $y = (y_i)_{i=1}^L$ by maximising the estimated conditional probability:

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

# *Inference and Training*

## *Training*

We find $\hat{p}$ using *SGD* with *Nesterov momentum*, finding a suitably low *cross-entropy* between the model $p$ and the empirical distribution present in the dataset $(w^n, y^n)_{n=1}^N$:

$$-\sum_{n=1}^{N} \log \prod_{i=1}^{L_n} \begin{Bmatrix} p_w(y_i^{n\,\text{arc}} \mid y_{<i}^n) p_w(y_i^{n\,\text{lab}} \mid y_i^{n\,\text{arc}}, y_{<i}^n) p_w(y_i^{n\,\text{ord}} \mid y_i^{n\,\text{arc}}, y_{<i}^n) \\ \cdot\, p_w(y_i^{n\,\text{pos}} \mid y_i^{n\,\text{arc}}, y_{<i}^n) p_w(y_i^{n\,\text{morph}} \mid y_i^{n\,\text{arc}}, y_{<i}^n) \end{Bmatrix}$$

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

## *Inference*

Given $w = (w_i)_{i=1}^L$, estimate $y = (y_i)_{i=1}^L$ by maximising the estimated conditional probability:

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

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

# *Model evaluation*

The TIGER treebank is a widely-used corpus of $\sim$50 000 constituent trees. Its main textual basis is the *Frankfurter Rundschau*.

# *Model evaluation*

The TIGER treebank is a widely-used corpus of ~50 000 constituent 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 evaluation*

The TIGER treebank is a widely-used corpus of $\sim$50 000 constituent 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 %.
- `PARSEVAL` metrics initially proposed by Black et al. (1991):

# *Model evaluation*

The TIGER treebank is a widely-used corpus of ∼50 000 constituent 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 %.
- `PARSEVAL` metrics initially proposed by Black et al. (1991):

$$P = \frac{\#\text{ of correct constituents in prediction}}{\#\text{ of total constituents in prediction}};$$

## *Model evaluation*

The TIGER treebank is a widely-used corpus of $\sim$50 000 constituent 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 %.
- `PARSEVAL` metrics initially proposed by Black et al. (1991):

$$P = \frac{\#\text{ of correct constituents in prediction}}{\#\text{ of total constituents in prediction}}; \quad R = \frac{\#\text{ of correct constituents in prediction}}{\#\text{ of total constituents in reference}}.$$

# *Model evaluation*

The TIGER treebank is a widely-used corpus of $\sim$50 000 constituent 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 %.
- `PARSEVAL` metrics initially proposed by Black et al. (1991):

$$P = \frac{\text{\# of correct constituents in prediction}}{\text{\# of total constituents in prediction}}; \quad R = \frac{\text{\# of correct constituents in prediction}}{\text{\# of total constituents in reference}}.$$

| **Model** | **F1** | **Disc. F1** |
|---|---|---|
| Coavoux et al. (2019) | 82.7 | 55.9 |
| Corro (2020) | 90.0 | 62.1 |
| F.-González & G.-Rodríguez (2022) | 89.8 | 71.0 |
| Chen & Komachi (2023) | 89.6 | 70.9 |
| **This work** | **90.59** | **84.74** |

*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 evaluation*

The TIGER treebank is a widely-used corpus of $\sim$50 000 constituent 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 %.
- `PARSEVAL` metrics initially proposed by Black et al. (1991):

$$P = \frac{\text{\# of correct constituents in prediction}}{\text{\# of total constituents in prediction}}; \quad R = \frac{\text{\# of correct constituents in prediction}}{\text{\# of total constituents in reference}}.$$

| **Model** | **F1** | **Disc. F1** |
|---|---|---|
| Coavoux et al. (2019) | 82.7 | 55.9 |
| Corro (2020) | 90.0 | 62.1 |
| F.-González & G.-Rodríguez (2022) | 89.8 | 71.0 |
| Chen & Komachi (2023) | 89.6 | 70.9 |
| **This work** | **90.59** | **84.74** |

*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) |
|---|---|---|
| Müller et al. (2013) | 98.20 | 98.27 |
| Schnabel & Schütze (2014) | 97.50 | 97.76 |
| Kondratyuk et al. (2018) | 98.58 | 98.97 |
| **This work** | **99.21** | **99.60** |

*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`.

# *Appendix: Mathematical justification for quadratic classifier*

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

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

# Appendix: Mathematical justification for quadratic classifier

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

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

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

$$\log p(c \mid \boldsymbol{e}_i, \boldsymbol{d}_j) = k_c - \frac{1}{2}\Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)^\mathsf{T} A_c \Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)$$

# *Appendix: Mathematical justification for quadratic classifier*

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

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

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

$$\begin{aligned}
\log p(c \mid \boldsymbol{e}_i, \boldsymbol{d}_j) &= k_c - \frac{1}{2}\Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)^\mathsf{T} A_c \Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big) \\
= - \, \boldsymbol{e}_i^\mathsf{T} Q_c^\mathsf{T} \boldsymbol{d}_j - \frac{1}{2}\boldsymbol{e}_i^\mathsf{T} A_c \boldsymbol{e}_i - \frac{1}{2}\boldsymbol{d}_j^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \boldsymbol{d}_j &+ \Big(\boldsymbol{\mu}_c^\mathsf{T} A_c + \phi_c^\mathsf{T} Q_c\Big)\boldsymbol{e}_i + (\phi_c^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} + \boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T})\boldsymbol{d}_j \\
&\qquad - \boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T} \phi_c - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} A_c \boldsymbol{\mu}_c \\
&\qquad - \frac{1}{2}\phi^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \phi + k_c \\
&\qquad + \log p(c, \mathbf{d}_j) - \log p(\mathbf{e}_i, \mathbf{d}_j)
\end{aligned}$$

# *Appendix: Mathematical justification for quadratic classifier*

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

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

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

$$\begin{aligned}
\log p(c \mid \boldsymbol{e}_i, \boldsymbol{d}_j) &= k_c - \frac{1}{2}\Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)^\mathsf{T} A_c \Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big) \\
= -\, \boldsymbol{e}_i^\mathsf{T} Q_c^\mathsf{T} \boldsymbol{d}_j - \frac{1}{2}\boldsymbol{e}_i^\mathsf{T} A_c \boldsymbol{e}_i - \frac{1}{2}\boldsymbol{d}_j^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \boldsymbol{d}_j &+ \Big(\boldsymbol{\mu}_c^\mathsf{T} A_c + \phi_c^\mathsf{T} Q_c\Big)\boldsymbol{e}_i + (\phi_c^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} + \boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T})\boldsymbol{d}_j \\
&\qquad - \boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T} \phi_c - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} A_c \boldsymbol{\mu}_c \\
&\qquad - \frac{1}{2}\phi^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \phi + k_c \\
&\qquad + \log p(c, \mathbf{d}_j) - \log p(\mathbf{e}_i, \mathbf{d}_j) \\
= c\text{th row of } \boldsymbol{e}_i^\mathsf{T} \mathbf{U}_{\text{h-d}} \boldsymbol{d}_j + \boldsymbol{e}_i^\mathsf{T} \mathbf{U}_{\text{h-h}} \boldsymbol{e}_i + \boldsymbol{d}_j^\mathsf{T} \mathbf{U}_{\text{d-d}} \boldsymbol{d}_j &+ U_{\text{h}} \boldsymbol{e}_i + U_{\text{d}} \boldsymbol{d}_j + \boldsymbol{u}_{\text{bias}}.
\end{aligned}$$

# *Appendix: Mathematical justification for quadratic classifier*

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

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

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

$$\log p(c \mid \boldsymbol{e}_i, \boldsymbol{d}_j) = k_c - \frac{1}{2}\Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)^\mathsf{T} A_c \Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^\mathsf{T}(\boldsymbol{d}_j - \phi_c)\Big)$$

$$= -\boxed{\boldsymbol{e}_i^\mathsf{T} Q_c^\mathsf{T} \boldsymbol{d}_j} - \boxed{\frac{1}{2}\boldsymbol{e}_i^\mathsf{T} A_c \boldsymbol{e}_i} - \boxed{\frac{1}{2}\boldsymbol{d}_j^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \boldsymbol{d}_j} + \boxed{\Big(\boldsymbol{\mu}_c^\mathsf{T} A_c + \phi_c^\mathsf{T} Q_c\Big)\boldsymbol{e}_i} + \boxed{(\phi_c^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} + \boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T})\boldsymbol{d}_j}$$

$$\boxed{\begin{aligned} &-\boldsymbol{\mu}_c^\mathsf{T} Q_c^\mathsf{T} \phi_c - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} A_c \boldsymbol{\mu}_c \\ &-\frac{1}{2}\phi^\mathsf{T} Q_c A_c^{-1} Q_c^\mathsf{T} \phi + k_c \\ &+ \log p(c, \mathbf{d}_j) - \log p(\mathbf{e}_i, \mathbf{d}_j) \end{aligned}}$$

$$= c\text{th row of } \boldsymbol{e}_i^\mathsf{T} \mathbf{U}_{\text{h-d}} \boldsymbol{d}_j + \boldsymbol{e}_i^\mathsf{T} \mathbf{U}_{\text{h-h}} \boldsymbol{e}_i + \boldsymbol{d}_j^\mathsf{T} \mathbf{U}_{\text{d-d}} \boldsymbol{d}_j + U_\text{h} \boldsymbol{e}_i + U_\text{d} \boldsymbol{d}_j + \boldsymbol{u}_\text{bias}.$$

# *Appendix: Mathematical justification for quadratic classifier*

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

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

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

$$\log p(c \mid \boldsymbol{e}_i, \boldsymbol{d}_j) = k_c - \frac{1}{2}\Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^{\mathsf{T}} (\boldsymbol{d}_j - \phi_c)\Big)^{\mathsf{T}} A_c \Big((\boldsymbol{e}_i - \boldsymbol{\mu}_c) + A_c^{-1} Q_c^{\mathsf{T}} (\boldsymbol{d}_j - \phi_c)\Big)$$

$$= - \boldsymbol{e}_i^{\mathsf{T}} Q_c^{\mathsf{T}} \boldsymbol{d}_j - \frac{1}{2} \boldsymbol{e}_i^{\mathsf{T}} A_c \boldsymbol{e}_i - \frac{1}{2} \boldsymbol{d}_j^{\mathsf{T}} Q_c A_c^{-1} Q_c^{\mathsf{T}} \boldsymbol{d}_j + \Big(\boldsymbol{\mu}_c^{\mathsf{T}} A_c + \phi_c^{\mathsf{T}} Q_c\Big) \boldsymbol{e}_i + (\phi_c^{\mathsf{T}} Q_c A_c^{-1} Q_c^{\mathsf{T}} + \boldsymbol{\mu}_c^{\mathsf{T}} Q_c^{\mathsf{T}}) \boldsymbol{d}_j$$

$$- \boldsymbol{\mu}_c^{\mathsf{T}} Q_c^{\mathsf{T}} \phi_c - \frac{1}{2} \boldsymbol{\mu}_c^{\mathsf{T}} A_c \boldsymbol{\mu}_c - \frac{1}{2} \phi^{\mathsf{T}} Q_c A_c^{-1} Q_c^{\mathsf{T}} \phi + k_c + \log p(c, \mathbf{d}_j) - \log p(\mathbf{e}_i, \mathbf{d}_j)$$

$$= c\text{th row of } \boldsymbol{e}_i^{\mathsf{T}} \mathbf{U}_{\text{h-d}} \boldsymbol{d}_j + \boldsymbol{e}_i^{\mathsf{T}} \mathbf{U}_{\text{h-h}} \boldsymbol{e}_i + \boldsymbol{d}_j^{\mathsf{T}} \mathbf{U}_{\text{d-d}} \boldsymbol{d}_j + U_{\text{h}} \boldsymbol{e}_i + U_{\text{d}} \boldsymbol{d}_j + \boldsymbol{u}_{\text{bias}}.$$

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

$$v_{i,j}^c = k^c + \left\| W_h^c \boldsymbol{e}_i + W_d^c \boldsymbol{d}_j + \boldsymbol{w}^c \right\|_2^2.$$

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J. L., et al. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

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.

Coavoux, M., Crabbé, B., & Cohen, S. B. (2019). Unlexicalized transition-based discontinuous constituency parsing. *Transactions of the Association for Computational Linguistics*, *7*, 73–89.

Corro, C. (2020). Span-based discontinuous constituency parsing: A family of exact chart-based algorithms with time complexities from o (nˆ 6) down to o (nˆ 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*.

# Bibliography II


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.

# *Bibliography III*


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*.