Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?

Yu Zhang, Zhenghua Li*, Houquan Zhou, Min Zhang
School of Computer Science and Technology, Soochow University, China
yzhang.cs@outlook.com, hqzhou@stu.suda.edu.cn
{zhli13,minzhang}@suda.edu.cn

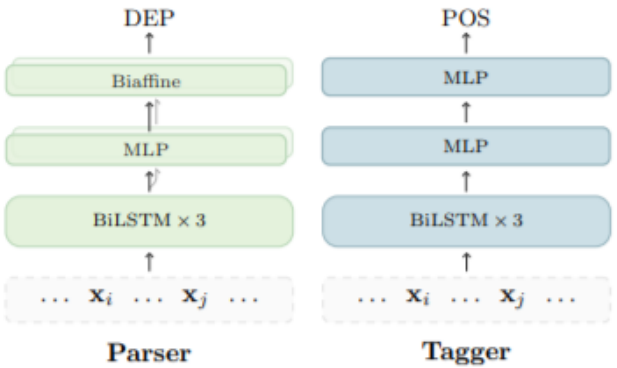


Figure 2: The basic tagging and parsing models.

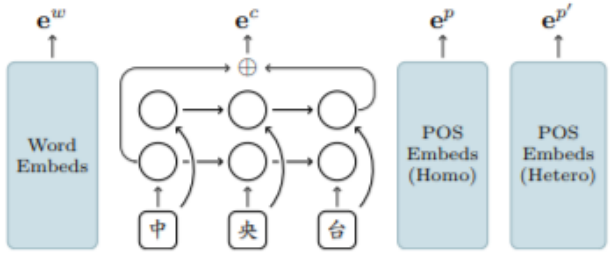
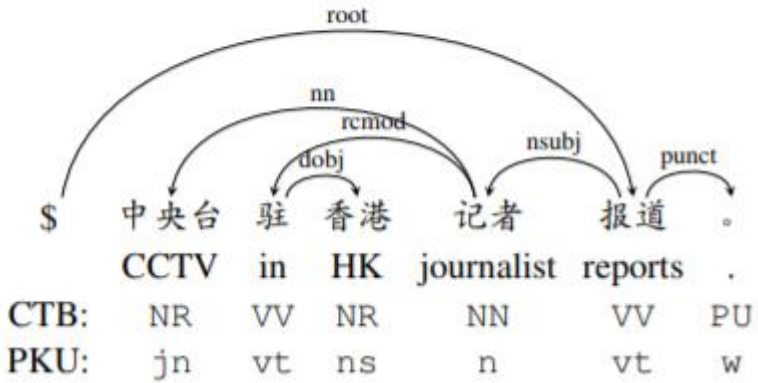


Figure 3: An illustration of the input representation of word "中央台(CCTV)". The homogeneous POS embedding $\mathbf{e}^p$ and heterogeneous POS embedding $\mathbf{e}^{p'}$ are used in the pipeline framework.



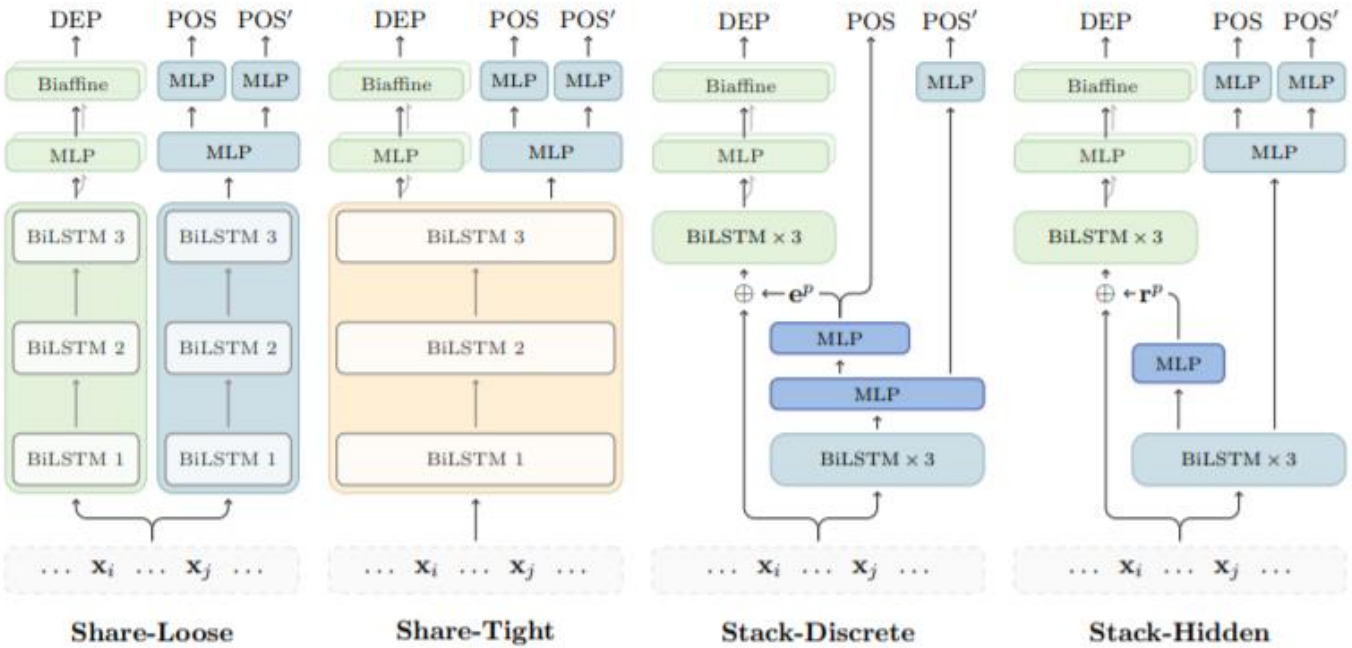


Figure 4: The framework of four variants of the joint model.

	PTB	CoNLL09	CTB7
$\mathbf{e}^p$	87.79	75.94	75.72
$\mathbf{e}^w$	93.42	85.30	84.43
$\mathbf{e}^c$	93.34	84.42	83.73
$\mathbf{e}^w \oplus \mathbf{e}^p$	93.92	85.94	85.12
$\mathbf{e}^w \oplus \mathbf{e}^c$	**93.97**	86.09	85.23
$\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p$	93.88	**86.17**	**85.32**
$\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p \oplus \mathbf{e}^{p\prime}$	-	86.01	85.23

Table 2: Parsing performance (LAS) on the dev data under the pipeline framework. $\mathbf{e}^w \oplus \mathbf{e}^c$ is our basic model.

		PTB	CoNLL09	CTB7
w/o hetero	Share-Loose	93.95	86.28	85.56
	Share-Tight	93.93	86.17	85.56
	Stack-Discrete	93.86	**86.35**	85.49
	Stack-Hidden	**94.09**	86.26	**85.79**
w/ hetero	Share-Loose	-	86.30	85.57
	Share-Tight	-	86.62	85.86
	Stack-Discrete	-	86.46	85.85
	Stack-Hidden	-	**86.69**	**85.88**

Table 3: Parsing performance (LAS) comparison on the dev data for the four joint methods.

		PTB			CoNLL09			CTB7		
		TA	UAS	LAS	TA	UAS	LAS	TA	UAS	LAS
w/o BERT	Andor et al. (2016)	97.44	94.61	92.79	-	84.72	80.85	-	-	-
	Dozat and Manning (2017)	97.3	95.74	94.08	-	88.90	85.38	-	-	-
	Ji et al. (2019)	97.3	95.97	94.31	-	-	-	-	-	-
	Li et al. (2019)	97.3	95.93	94.19	-	88.77	85.58	-	-	-
	Basic ($\mathbf{e}^w \oplus \mathbf{e}^c$)	97.50	95.97	94.34	96.42	89.12	86.00	96.48	88.54	85.34
	Pipeline ($\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p$)	97.50	95.88	94.27	96.42	89.12	85.98	96.48	88.42	85.28
	Joint Stack-Hidden	**97.91**	**96.13**	**94.53**	96.55	89.46	86.44	96.62	88.86	85.88
	Joint Stack-Hidden w/ hetero	-	-	-	**96.66**	**89.85**	**86.85**	**96.71**	**89.21**	**86.22**
w/ BERT	Li et al. (2019)	-	96.67	95.03	-	92.24	89.29	-	-	-
	Basic ($\mathbf{e}^w \oplus \mathbf{e}^c$)	97.42	96.85	95.14	97.29	92.21	89.42	97.22	91.66	88.75
	Joint Stack-Hidden	**97.57**	**96.85**	**95.25**	97.36	92.44	89.68	97.32	91.67	88.84
	Joint Stack-Hidden w/ hetero	-	-	-	**97.39**	**92.46**	**89.76**	**97.40**	**91.81**	**89.04**

Table 4: Final results on the test data.

		UAS	LAS			UAS	LAS
NN	→ NN	91.73	89.69	NR	→ NR	91.73	86.96
	→ VV	67.25	44.98		→ NN	86.39	83.67
	→ NR	90.43	86.96	JJ	→ JJ	95.40	94.33
	→ JJ	91.96	20.54		→ NN	92.82	14.92
VV	→ VV	85.92	84.12	DEG	→ DEG	96.75	95.91
	→ NN	65.60	40.07		→ DEC	92.06	26.56
	→ VA	84.75	83.05	DEC	→ DEC	94.28	92.39
	→ AD	55.32	25.53		→ DEG	96.88	22.22

Table 5: The impact of specific POS tagging error patterns on parsing.

Discontinuous Constituent Parsing with Pointer Networks

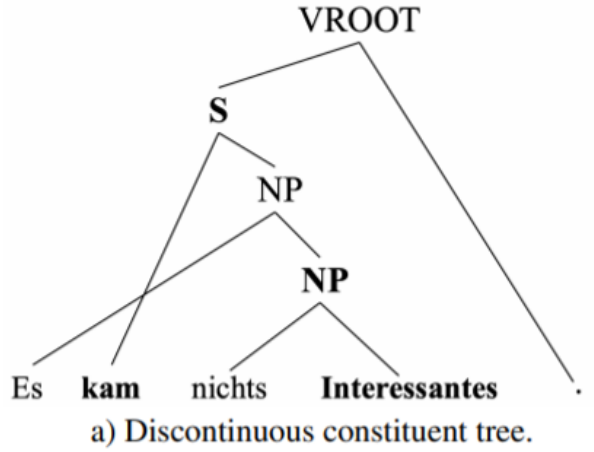
Daniel Fernández-González and Carlos Gómez-Rodríguez
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Group
Depto. de Ciencias de la Computación y Tecnologías de la Información
Elviña, 15071 A Coruña, Spain
d.fgonzalez@udc.es, carlos.gomez@udc.es

Introduction

- One of the most complex syntactic representations used in computational linguistics and NLP are discontinuous constituent trees, crucial for representing all grammatical phenomena of languages such as German.
- Recent advances in dependency parsing have shown that Pointer Networks excel in efficiently parsing syntactic relations between words in a sentence. This kind of sequence-to-sequence models achieve outstanding accuracies in building non-projective dependency trees, but its potential has not been proved yet on a more difficult task.
- We propose a novel neural network architecture that, by means of Pointer Networks, is able to generate the most accurate discontinuous constituent representations to date, even without the need of Part-of-Speech tagging information. To do so, we internally model discontinuous constituent structures as augmented non-projective dependency structures. The proposed approach achieves state-of-the-art results on the two widely-used NEGRA and TIGER benchmarks, outperforming previous work by a wide margin.

Previous Work

Linear Context-Free Rewriting Systems.

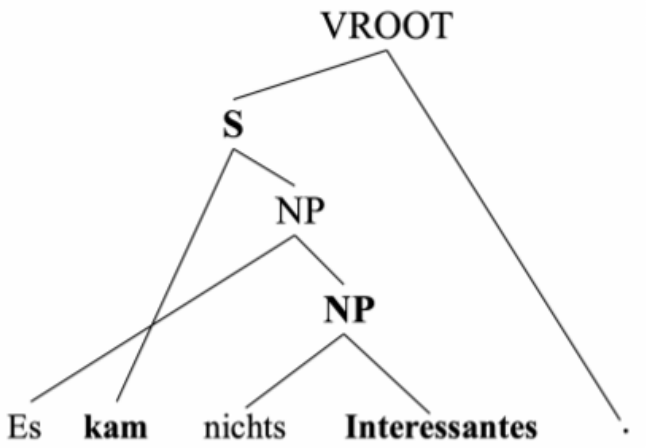


a) Discontinuous constituent tree.

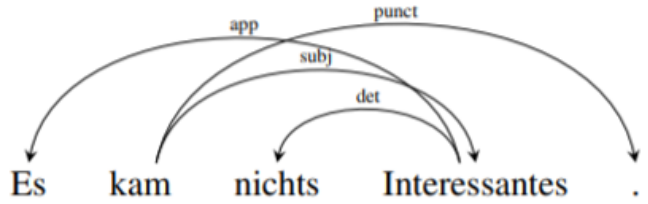
Transition-based parsers with transitions and data structures that are able to deal with discontinuity.

Represent discontinuous formalisms as dependencies with augmented information. Use an extra module that, at a post-processing step after the parsing, labels each dependency to increase final accuracy. As a result, parsing and labelling tasks are learned and performed in a two-stage procedure.

Modelling Discontinuous Constituent Trees Preliminaries



a) Discontinuous constituent tree.

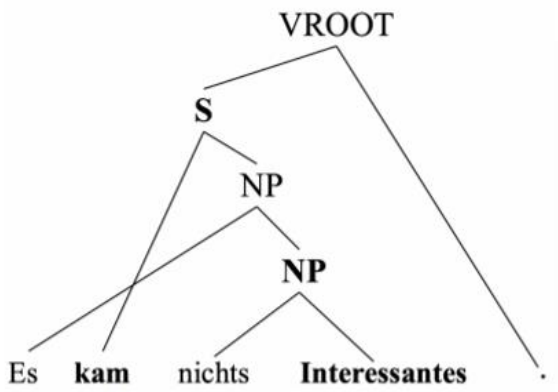


c) Labelled dependency tree.

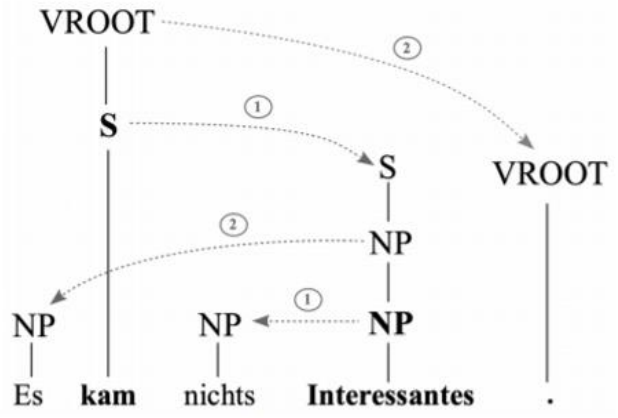
It can be noticed from these definitions that, in order to represent the same syntactic phenomenon as described in a discontinuous constituent tree, we will need to use a non-projective dependency structure in order to handle discontinuities. However, a constituent tree is able to provide information that cannot be represented in a regular dependency tree (Kahane and Mazziotta 2015).

Constituents as Augmented Dependencies

A constituent tree with m words can be decomposed into a set of m spines (Carreras, Collins, and Koo 2008), one per word, as shown in Figure 1(b) for the discontinuous tree in Figure 1(a). These spines and their interaction to finally build a constituent tree can be represented in an augmented dependency tree as described by (Fernández-González and Martins 2015).

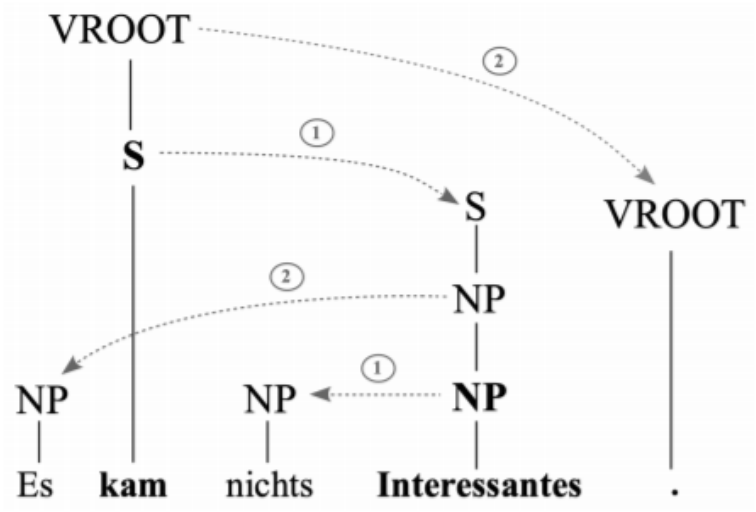


a) Discontinuous constituent tree.

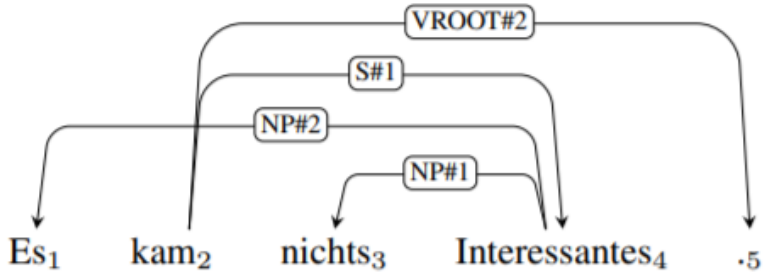


b) Spines interaction.

Constituents as Augmented Dependencies

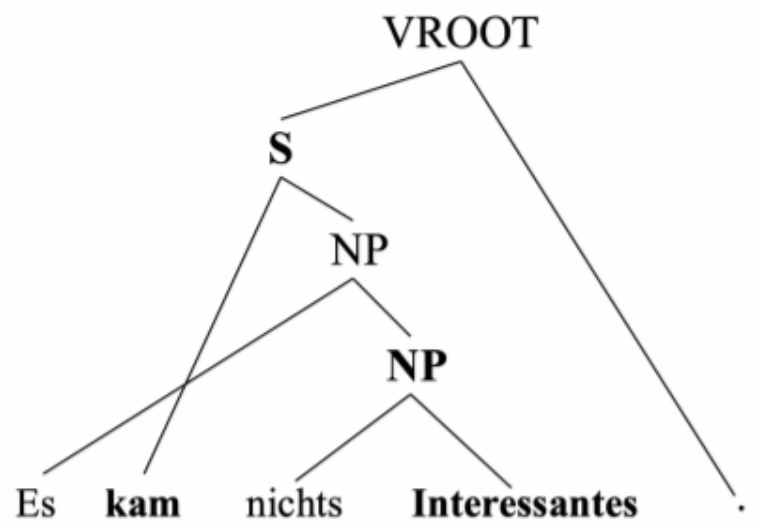


b) Spines interaction.

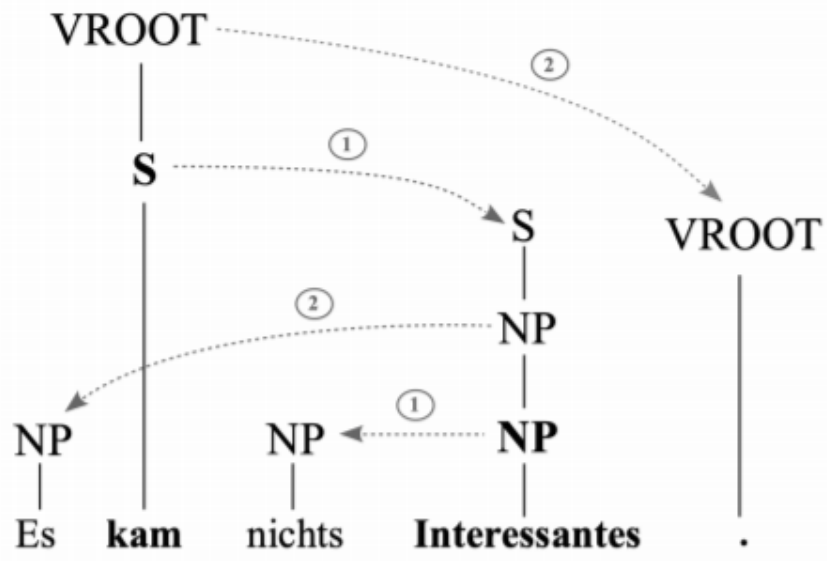


d) Augmented dependency tree.

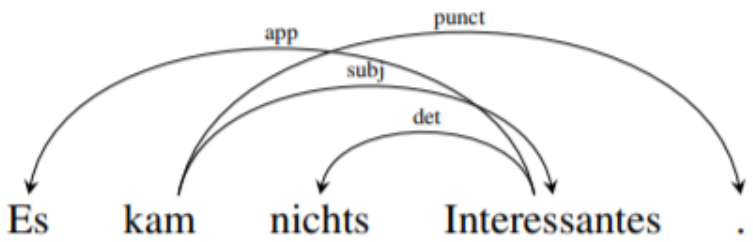
Additionally, to represent the original phrase structure it is also necessary to save some vital information into arc labels: the non-terminal symbol X plus an index that indicates the order in which spines are attached. This index will be crucial in those cases where more than one constituent share the same head word (and, therefore, the same head spine), but they are at a different level in the original tree and, therefore, should be created in a different hierarchical order. The final dependency arcs will have the form $(w_h, w_i, X\#p)$, where p is the attachment order. For instance, constituent $(NP, \{nichts, Interessantes\}, Interessantes)$ in Figure 1(a), is encoded as the augmented dependency arc $(Interessantes, nichts, NP\#1)$ in Figure 1(d), and constituent $(NP, \{Es, nichts, Interessantes\}, Interessantes)$ is represented with $(Interessantes, Es, NP\#2)$. Both constituents share the same head spine anchored to word $Interessantes$, but they are attached in a different level.



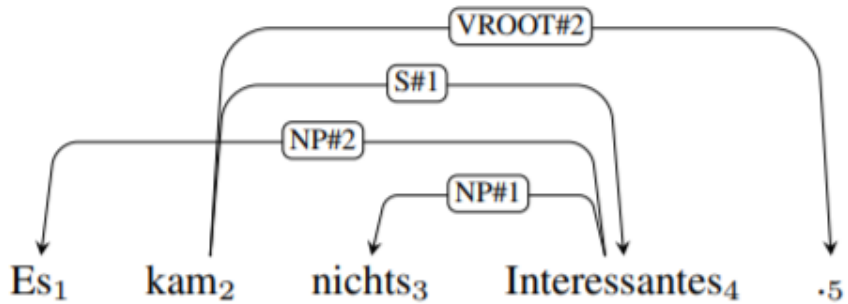
a) Discontinuous constituent tree.



b) Spines interaction.



c) Labelled dependency tree.



d) Augmented dependency tree.

Figure 1: Augmented dependency representation of a sentence taken from NEGRA corpus in discontinuous constituent format, as well as the spine interaction required for its reconstruction. Head nodes of each constituent are marked in bold.

Neural Network Architecture

$$\mathbf{x}_i = \mathbf{e}_i^c \oplus \mathbf{e}_i^w \oplus \mathbf{e}_i^p$$

$$\mathbf{h}_i = \mathbf{h}_{li} \oplus \mathbf{h}_{ri} = \mathbf{BiLSTM}(\mathbf{x}_i)$$

$$\mathbf{r}_i = \mathbf{h}_{i-1} + \mathbf{h}_i + \mathbf{h}_{i+1}$$

$$\mathbf{s}_t = \mathbf{LSTM}(\mathbf{r}_i)$$

$$\mathbf{v}_j^t = \mathbf{score}(\mathbf{s}_t, \mathbf{h}_j)$$

$$\mathbf{a}^t = \mathbf{softmax}(\mathbf{v}^t)$$

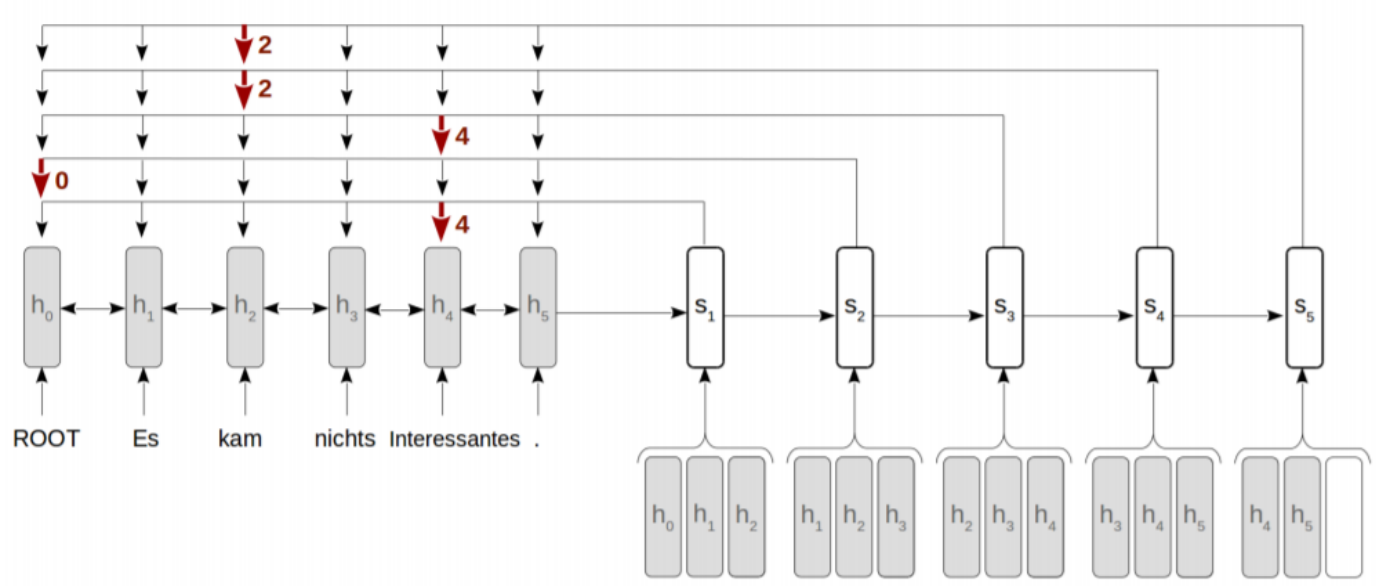
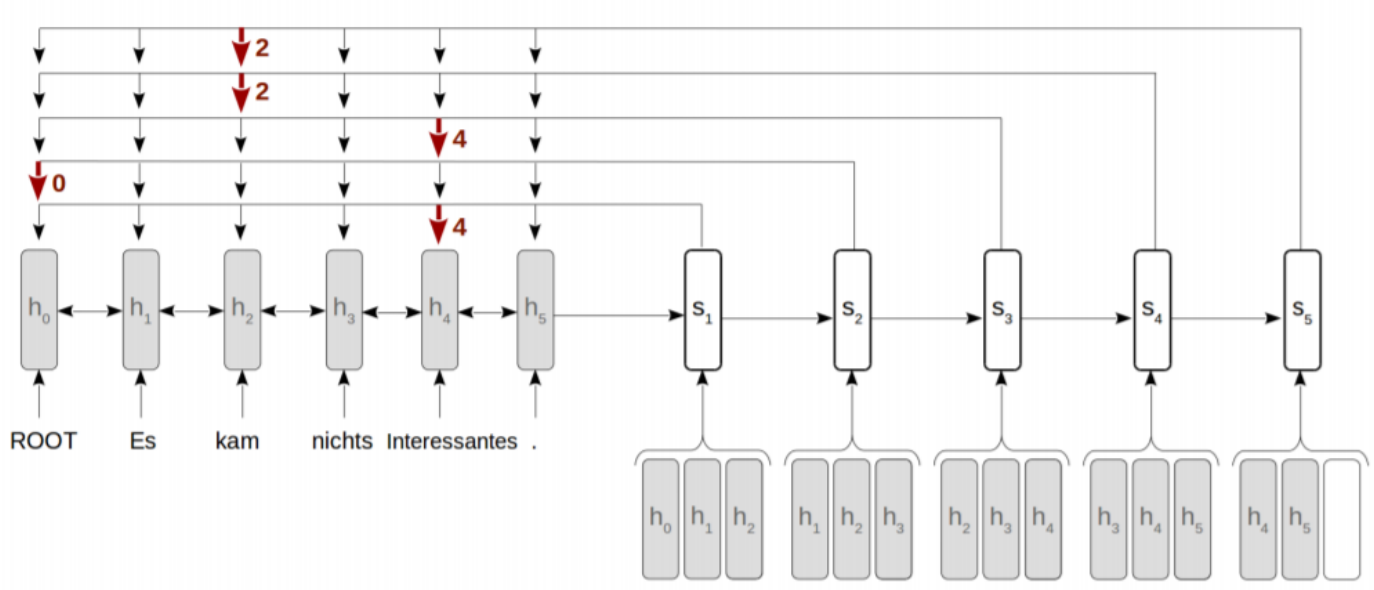
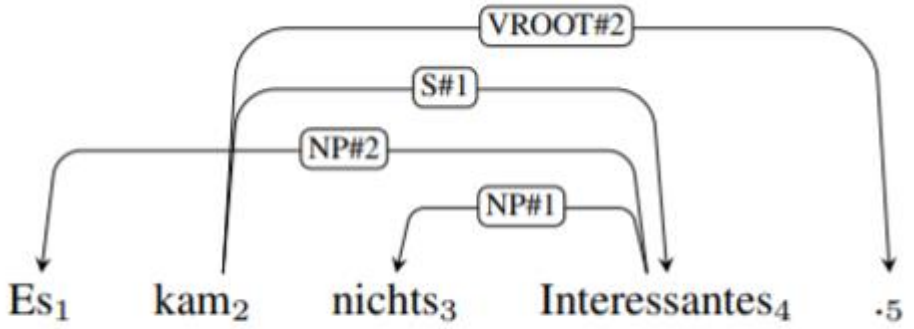


Figure 2: Pointer Network architecture and decoding steps to output the dependency tree in Figure 1(d).

$$\mathbf{v}_j^t = \mathbf{s}_t^T W \mathbf{h}_j + \mathbf{U}^T \mathbf{s}_t + \mathbf{V}^T \mathbf{h}_j + \mathbf{b}$$

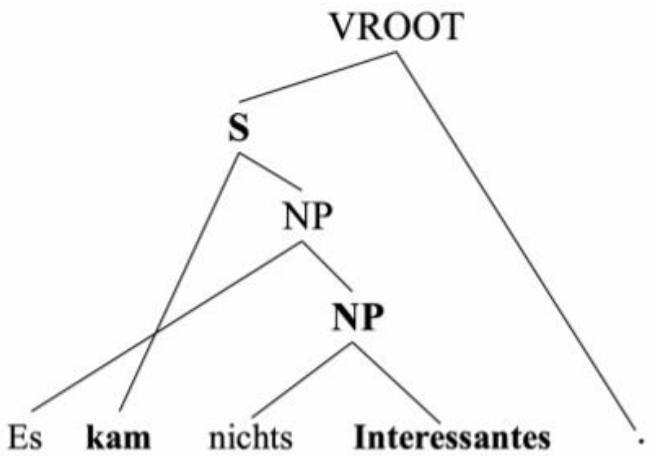
$$\mathbf{s}_{tj}^l = \mathbf{s}_t^T W_l \mathbf{h}_j + \mathbf{U}_l^T \mathbf{s}_t + \mathbf{V}_l^T \mathbf{h}_j + \mathbf{b}_l$$



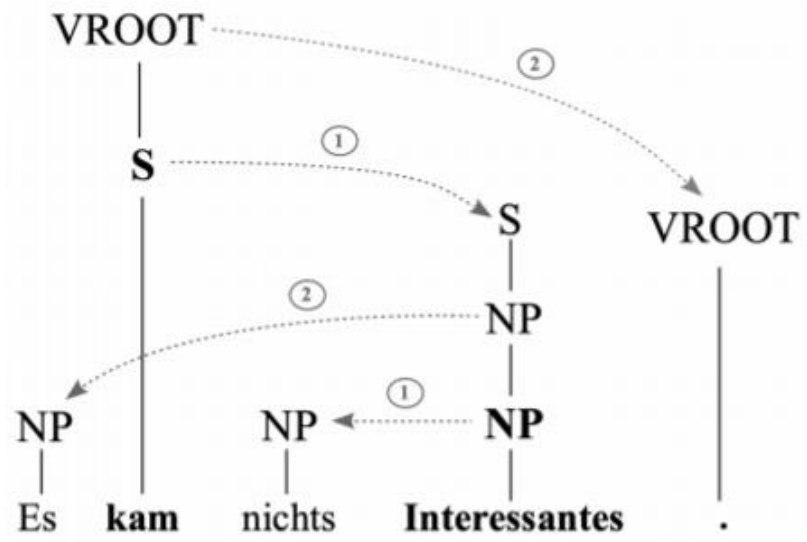


d) Augmented dependency tree.

Figure 2: Pointer Network architecture and decoding steps to output the dependency tree in Figure 1(d).



a) Discontinuous constituent tree.



b) Spines interaction.

Multitask Learning

Our goal is to provide a fully-parsed non-projective dependency tree in a single stage, while the parsing and the labelling should be individually undertaken due to the large amount of labels resulting from the formalism by (Fernández-González and Martins 2015). We follow a multitask learning strategy (Caruana 1997) to achieve that: a single neural architecture is trained for more than one task, sharing a common representation and benefiting from each other.

In particular, the labeler shares the same encoder as the parser, providing a common encoder hidden state representation for both components. As stated by (Kiperwasser and Goldberg 2016), training the BiLSTM-CNN encoder to correctly predict dependency labels significantly improves unlabelled parsing accuracy and vice versa. This is specially crucial in our approach where a constituent structure is jointly encoded in dependency arcs and labels, and a wrong label can lead to a completely different phrase-structure tree after the recovering.

$$P_\theta(y|x) = \prod_{i=1}^{n} P_\theta(a_i|a_{<i}, x) = \prod_{i=1}^{n} P_\theta(w_h|w_i, a_{<i}, x)$$

$$\mathcal{L}_{arc} = -logP_\theta(w_h|w_i, a_{<i}, x)$$
$$\mathcal{L}_{label} = -logP_\theta(l|w_h, w_i)$$

Finally, we train a joint model by summing the losses $\mathcal{L}_{arc}$ and $\mathcal{L}_{label}$ prior to computing the gradients. In that way, model parameters are learned to minimize the sum of the cross-entropy loss objectives over the whole corpus.

Parser	NEGRA F1	NEGRA Disc. F1	TIGER F1	TIGER Disc. F1
Predicted POS tags				
Fernández-González and Martins (2015)	77.0		77.3	
van Cranenburgh, Scha, and Bod (2016), $\leq$40	74.8			
Versley (2016)			79.5	
Stanojević and G. Alhama (2017)			77.0	
Coavoux and Crabbé (2017)			79.3	
Gebhardt (2018)			75.1	
Coavoux, Crabbé, and Cohen (2019)	83.2	54.6	82.7	55.9
Coavoux and Cohen (2019)	83.2	56.3	82.5	55.9
This work	**85.4**	**58.8**	**85.3**	**59.1**
Gold POS tags				
Maier (2015)	77.0	19.8	74.7	18.8
Fernández-González and Martins (2015)	80.5		80.6	
Maier and Lichte (2016)			76.5	
Corro, Le Roux, and Lacroix (2017)			81.6	
Stanojević and G. Alhama (2017)	82.9		81.6	
Coavoux and Crabbé (2017)	82.2	50.0	81.6	49.2
This work	**86.1**	**59.9**	**86.3**	**60.7**
This work				
w/o POS tags	85.7	58.6	85.7	60.4
w/o POS tags, w/o pre-trained word embs.	83.7	54.7	84.6	57.9

Table 2: Accuracy comparison of state-of-the-art discontinuous constituent parsers on NEGRA and TIGER.