Parsing Headed Constituencies

Katarzyna Krasnowska-Kieraś, Marcin Woliński
Institute of Computer Science, Polish Academy of Science
{k.krasnowska,wolinski}@ipipan.waw.pl

Abstract

In the paper, we present a parsing technique that generates headed constituency trees, which combine information typically contained in constituency and dependency trees. We advocate for using such structures for syntactic representation. The parsing method combines prediction of dependency links with prediction of constituency spines in a 'parsing as tagging' approach and outputs a hybrid structure. An interesting feature is that the method can generate constituency trees with discontinuities. The parser is built on top of a BERT model for the given language and uses a specially crafted classifier for predicting dependency links. With suitable training data the method can be applied to arbitrary language; we report evaluation results for Polish and German.

Keywords: neural syntactic parsing, constituency/dependency structures, discontinuities

1. Introduction

Syntactic parsing is an important NLP task often used in various text processing pipelines. Syntactic structures are usually represented using constituency or dependency trees. Constituency structures give easy access to phrases that make up sentences. Thus, they are preferred in tasks such as nominal phrase extraction, identification of terminology etc. Dependency trees are closer to predicate-argument structures, so they are the preferred representation for more semantic tasks, involving, e.g., the analysis of events and their actors.

Since constituency and dependency structures are used for different tasks, it seems a practical solution to have both available. Our goal in this work is to fulfil one more requirement: the structures have to be consistent with each other.

We plan to create a large parsebank of Polish sentences annotated with both types of structures. The treebank search engine will allow to query both constituency and dependency structures and we want to be sure that the results of these queries are in line with each other. The goal of the present work is to develop a hybrid parsing method which would provide consistent constituency and dependency structures.

In our first experiments (Krasnowska-Kieraś and Woliński, 2023) we concentrated on the method for predicting constituency structure consistent with dependency trees. In this paper, we present a complete parser for headed constituencies[1] and test it on the data in two languages.

[1] http://git.nlp.ipipan.waw.pl/constituency/Parser

2. Headed Constituencies

To introduce the proposed method, let us take a step back and look at the structures commonly used to represent syntax. In dependency trees, the relations between words are in the focus of interest: the verb's need of its complements, the relation between a noun and its adjectival attribute, and so on. The exact choice of relations is a matter of convention, but, wisely chosen, it leads to binding all words of a sentence in the form of a tree (cf. Fig. 1).

Constituency trees model the hierarchical nature of the natural language by showing how longer fragments are composed of shorter constituents (cf. Fig. 2). The parent-child relation corresponds to the child being a sub-span of the parent.

The grammatical features of words can be generalised to constituents. Thus a constituent *nowego domu*[2] 'new house' can be considered to be in the genitive case and in the singular, since these are the features of the nominal form *domu* 'house' shared by the adjectival form *nowego* 'new'.

Moreover, the relations of dependency syntax can be thought of as occurring between constituents: a verbal phrase *pokazali i opisali* 'showed and described' subcategorises for a subject in the nominative (e.g. *dziennikarze* 'the journalists') and an object in the accusative (e.g. *sytuację* 'the situation').

These relations can be used to create a local dependency structure among the children of a given constituent. For the sentence node **S**, the local dependencies can be understood as going from the verbal phrase **VP** (marked as a head) to two nominal phrases (labelled subj and obj) and to a prepositional adjunct. (A technical relation punct is also

[2] We use examples in Polish, since its system of 7 grammatical cases makes the grammatical relations easier to observe than in English.

LREC-COLING 2024, pages 12633–12643

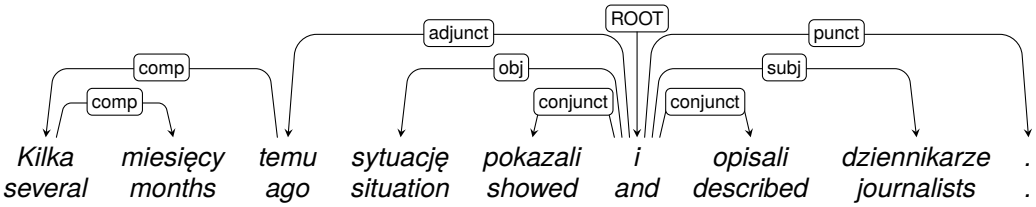

Figure 1: Dependency tree for the sentence: *Kilka miesięcy temu sytuację pokazali i opisali dziennikarze.* 'The journalists exposed and described the situation several months ago.'

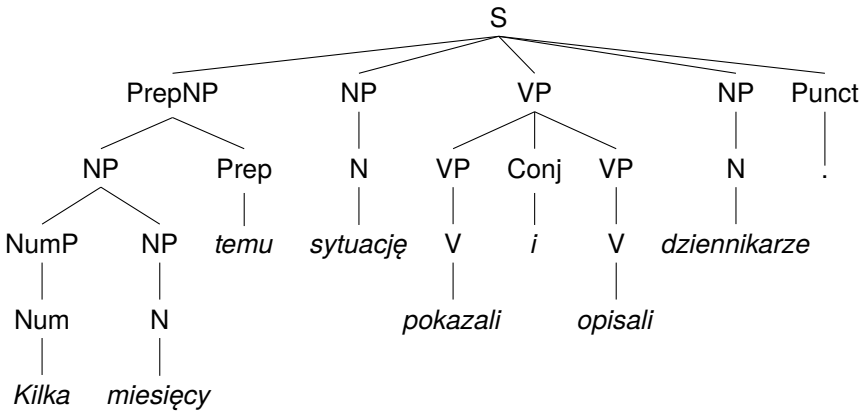

Figure 2: Constituency tree for the same sentence

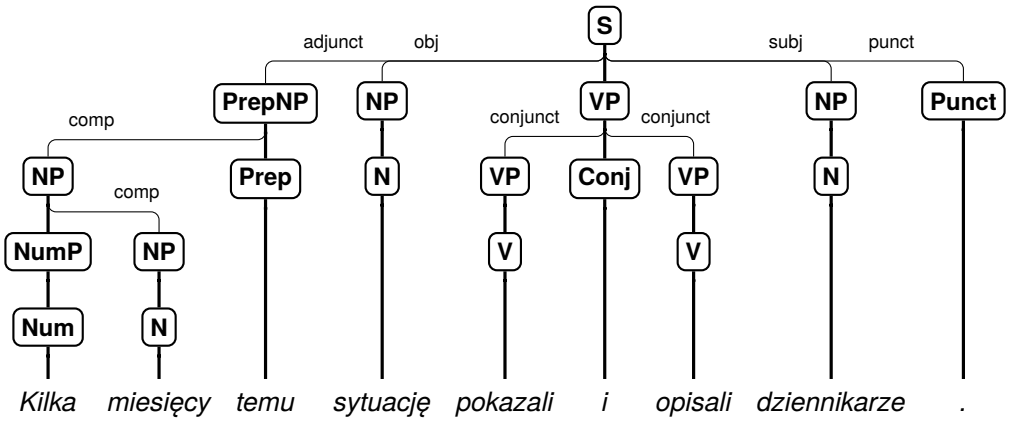

Figure 3: Headed constituency tree for the same sentence. Bold line joins a constituent with its head child.

used to take care of punctuation.)

This procedure leads to the creation of a *headed constituency tree* in which each constituent has exactly one head among its children and each non-head child is labelled with a dependency relation (cf. Fig. 3). The constituency and dependency trees can be rather trivially extracted from this structure. However, we prefer to think of headed constituencies as a model of syntax in its own right, which includes both types of syntactic information.

There is one important catch in these considerations: for the joint structure to be possible, the constituency and dependency trees have to be consistent with each other. This means that each couple of a parent and its non-head child in the constituency tree has to correspond to a dependency edge between head tokens of these two constituents.

Typically, constituency trees model surface syntax, that is they reflect purely grammatical interactions within a sentence (e.g. Marcus et al., 1993; Brants et al., 2004). This is often not true in the case of dependency trees, in particular Universal Dependencies (Nivre et al., 2020), which involve more semantic relations. In contrast, an example of a surface syntax oriented dependency scheme is SUD (Gerdes et al., 2018).

3. The Datasets

The requirement of consistent constituency and dependency trees severely limits the available data. For the present experiment we were able to construct two datasets: Polish and German. It is worth noting that these languages belong to different language groups, which allows for a more comprehensive testing of our method. The sizes of the datasets are reported in Table 1.

3.1. Polish

The Polish dataset is built by merging information from the *Składnica* constituency treebank (Woliński, 2019; Woliński and Hajnicz, 2021; Woliński, 2019; Świdziński and Woliński, 2010) with the Polish Dependency Bank (PDB, Wróblewska, 2022; Wróblewska, 2014).

Składnica consists of surface-syntactic constituency trees which were manually selected among parse forests generated by a rule based parser. The trees include information on the heads (syntactic centres) of constituents. However, dependency labels are not present, so these are not complete headed constituencies as we understand them in this paper.

The starting point for PDB was converting the trees of Składnica to dependency structures. The result was enriched with dependency labels and manually validated. From that moment, the two resources were developed independently. However, PDB retains the surface-syntax character of Składnica, which makes it easy to align the trees.

Składnica does not insist on binary trees. As the authors explain (Woliński, 2019), for Polish, with its free word order and a rich repertoire of verbal complements, it is most natural to treat all these as direct children of the **S** node. In result, the trees are rather "flat" and similar in structure to dependency trees (cf. Fig. 1 and 2). Coordination is the source of most visible difference in these structures: in the dependency tree in Fig. 1 almost all edges fan out from the node for conjunction. The tree in Fig. 2 provides a more readable structure with a separate **VP** node for the coordinated verbal structure which as a whole becomes a constituent of the sentence **S**.

3.2. German

The concept of decorating edges of a constituency tree with relations was used in the TIGER treebank of German (Tiger Corpus Team, 2006; Brants et al., 2004). Unfortunately, their structures are not strictly headed (the relations do not form a dependency tree).

These trees were later converted to dependency structures (Falenska et al., 2020). In the preparation of the dataset, we follow these dependencies to determine heads for particular constituents. We also reattach punctuation characters in the same way (in original TIGER all punctuation is attached to a virtual root node). However, we use the representation of coordinated structures with the conjunction (or a comma) as the head, as this leads to dependency structure better harmonized with constituencies.

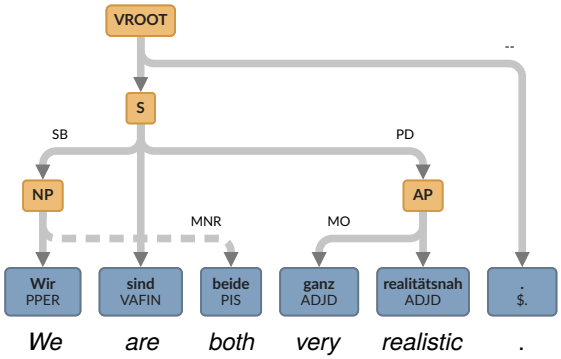

Figure 4: Tree with a discontinuity from the German dataset

The constituency trees of TIGER are somewhat similar to Składnica (cf. Fig. 4), e.g., they attach all dependents of a verb in one go. However, they use smaller number of internal nodes, in result having a larger average number of children per node. They also sport a larger number of discontinuities (Table 1).

Out of 50,472 structures in TIGER, we reject 4,207 degenerate ones. A vast majority of them were rather trivial, e.g. "sentences" like *dpa tb pe dpa* (4 tokens, all marked as roots) or 876 single-token sentences.

4. Parsing Technique

4.1. Spines

If a headed constituency tree is visualised as in Fig. 3 – with each node centred over its head constituent – an interesting structure becomes visible. The sequences of syntactic units having the same token as their centre form vertical clusters which we call *spines*.[3] The spines are quite intuitive: in a subordinate construction, the grammatical features of a constituent take source (mainly) in its head. Thus all nodes of a spine having a noun as its base represent nominal constructions of various levels of complication. When the nominal construction is, e.g., required by a verb, the nominal spine does not grow higher, but gets attached to a verbal spine. Spines with a conjunction as a base are more context dependent: the higher nodes depend

[3] It seems that the first to use this term in this meaning is Carreras et al. (2008).

		Polish			German		
		discontinuous		total	discontinuous		total
train	trees	1,756	(9.9%)	**17,659**	12,201	(32.3%)	**37725**
	tokens	40,515	(14.5%)	280,046	295,196	(41.6%)	709 074
	avg token/tree	23.07		15.86	24.19		18.80
	avg tree height/length	0.21		0.34	0.11		0.15
validation	trees	231	(10.4%)	**2,211**	1,326	(30.8%)	**4,312**
	tokens	5,034	(14.6%)	34,565	31,983	(40.2%)	79,588
	avg token/tree	21.79		15.63	24.12		18.46
	avg tree height/length	0.22		0.36	0.11		0.15
test	trees	215	(9.8%)	**2,205**	1,267	(30.0%)	**4,228**
	tokens	4,815	(14.4%)	33 344	29,982	(39.3%)	76,315
	avg token/tree	22.40		15.12	23.66		18.05
	avg tree height/length	0.21		0.37	0.11		0.16

Table 1: Dataset sizes for Polish and German

	Polish		German	
length	occurrences in the data			
0	3155	0.9%	459724	53.2%
1	70133	20.2%	344606	39.8%
2	148987	42.8%	56857	6.6%
3	104049	29.9%	3665	0.4%
4	21619	6.2%	118	<0.1%
5	12	<0.1%	7	<0.1%
avg. length		2.21		0.54
spine types		110		299

Table 2: Spine lengths and the number of spine types in the Polish and German data

on constituents being coordinated by the conjunction. In both cases the height of a spine is rather limited. It depends on the way modifiers are attached in a given grammar/treebank.

Prediction of a spine for a token can be seen as generalised part of speech tagging. Nodes low in the spine depend mainly on the base token, higher nodes include more contextual information. We think that it is an interesting model in view of the fact that nowadays a large language model is usually used as an encoder for parsing. It seems reasonable to attach the prediction of a constituent to the token in its centre. The mechanism of attention used in current models should be able to provide the necessary data about the token and its context.

More formally, a spine is a maximum path following head edges between the nodes. Each spine ends with some token of the sentence and each token of the sentence is the end of exactly one spine (we assume empty spines for tokens which are not the head of any node). Suppose that token t_i is the dependency head of another token t_j, and their corresponding spines are $s_i = n_{i,1} \rightarrow ... \rightarrow n_{i,k}$ and $s_j = n_{j,1} \rightarrow ... \rightarrow n_{j,l}$. This means that $n_{j,1}$ (the topmost nonterminal of s_j) is a non-head child of some nonterminal along s_i.

For example, in the tree shown in Fig. 3, the tokens $t_1 =$ *Kilka* 'several', $t_2 =$ *miesięcy* 'months' and $t_3 =$ *temu* 'ago' have following spines respectively: $s_1 =$ **NP** $\rightarrow$ **Num**, $s_2 =$ **N**, $s_3 =$ **PrepNP** $\rightarrow$ **Prep**. t_2 is a dependent of t_1, and the topmost (and only) node of s_2 (**N**) is a non-head child of the **NP** node of s_1; t_1 is a dependent of t_3, and the topmost node of s_1 (**NP**) is a non-head child of the **PrepNP** node of s_3.

4.2. Spine Based Parsing

Parsing of headed constituencies can be decomposed into (1) determining the spines for each token of the input (2) determining the way their top nodes are attached to other spines. For the latter part, it is necessary to determine which spine is attached to which other spine (2a) and through which node of the head spine (2b). Finally, (3) dependency labels have to be introduced on respective edges.

Tasks (2a) and (3) correspond exactly to building a dependency tree. If we use a dependency parser for this part, the resulting constituency trees will be consistent with dependency trees produced by this parser.[4]

A very desirable trait of the proposed method is that if the task (2a) is performed by a dependency parser capable of handling non-projective structures, the resulting constituency parser becomes immune to the problem of discontinuity. Moreover, with this technique the other typical problem in

[4] In other words, the tasks (1) and (2b) can be seen as converting the dependency structure to constituencies. Note, however, that the constituency trees are more detailed, so this process adds information.

#	token	head	deprel	spine	attachment
1	*Jak*	4	adjunct_mod	**AdvP** → **Adv**	**VP**-2
2	*można*	0	root	**ROOT** → **S** → **VP** → **V**	—
3	*sobie*	4	refl	**NP** → **N**	**VP**-2
4	*radzić*	2	comp_inf	**VP** → **VP** → **V**	**S**-1
5	*?*	2	punct	**Punct**	**ROOT**-1

Table 3: The tree of Fig. 5 encoded with dependency heads and relations, spines and their attachments

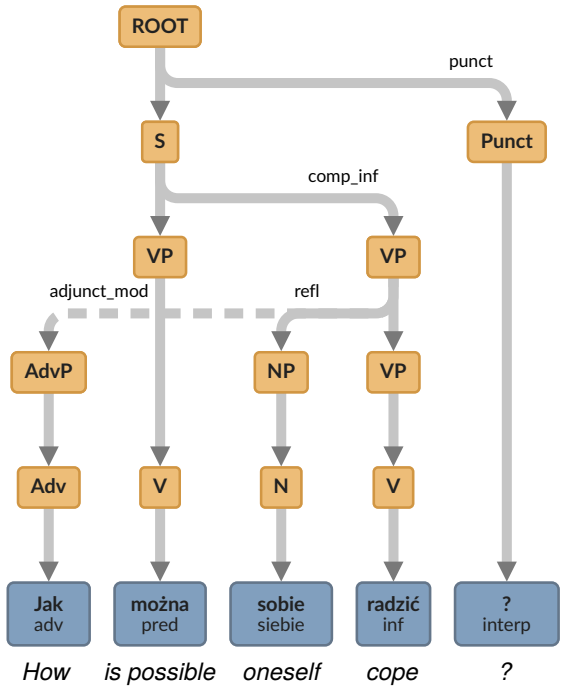

Figure 5: A tree with a discontinuity and a **VP** → **VP** edge along a spine: 'How can one cope?'

constructing constituency parsers, that of unary branches, does not arise at all. Unary branches in the tree are parts of some spines and they get predicted as such.

One issue that needs addressing is that a spine may contain a sequence of consecutive nodes bearing the same syntactic category. For example, the representation of the phrase *Jak sobie radzić* 'how to cope' in the tree shown in Fig. 5 reflects its hierarchical structure $((Jak)_{\mathbf{AdvP}}(sobie)_{\mathbf{NP}}(radzić)_{\mathbf{VP}})_{\mathbf{VP}}$, with **AdvP** → **Adv** → *Jak* 'how' and **NP** → **N** → *sobie* 'oneself' attached to $\mathbf{VP}_2 \rightarrow \mathbf{VP}_1 \rightarrow \mathbf{V} \rightarrow$ *radzić* 'to cope' as non-head children of $\mathbf{VP}_2$.[5]

The attachment information consists of the category of the node a spine attaches to and its level in the sequence of identical nodes in its spine, counting from the bottom. Together with the dependencies between tokens, such representation of spines and attachments allows us to encode the complete headed constituency tree.

As an example, consider the tree from Fig. 5 and its representation shown in Table 3. The spines for *radzić* 'to cope' and the final punctuation are attached as children of the only **S** and **ROOT** nodes respectively along the 'root' spine for *można* 'is possible'. Thererefore, they both have **S**-1 and **ROOT**-1 attachments. The spine for *radzić* contains a sequence of two **VP** nodes. The spines for its two dependents, *Jak* 'how' and *sobie* 'oneself', are attached to the second **VP** from the bottom (**VP**-2). The 'head' and 'deprel' columns of the table contain the dependency relations between tokens following a CoNLL(-U)-like convention.

5. Parser Architecture

In our first experiments with this method (Krasnowska-Kieraś and Woliński, 2023), we used an external dependency parser for tasks (2a) and (3) mentioned earlier. This of course led to using separate instances of language models for dependency and constituency. In the present work, we build a complete parser in the form of multiple classifiers attached to the output of a single language model.

The input sentence is run through a BERT model which computes contextualised vector representations for individual tokens. Those representations are, in turn, passed to layers predicting particular aspects of the syntactic structure (see Fig. 6). At training time, those layers are optimised jointly, with the BERT model being fine-tuned to all of them at once. Therefore, there is no direct interaction between individual classifiers, only an indirect one through the shared BERT model.

In Krasnowska-Kieraś and Woliński (2023) we show that spine types are not very numerous in the Polish data – 110 items – and can be represented as atomic labels. There are more spine types for German (see Table 2) but their number remains in the same order of magnitude. In our architecture, spines, attachment points (nonterminal type and its height) and dependency relations are predicted each by a single dense layer operating on BERT output vectors (the layers are shown symbolically as SPINES, ATT.NONT., ATT.HEIGHT and DEPRELS in Fig. 6).

[5] We use the lower subscript $\mathbf{VP}_i$ to differentiate between two different **VP** nodes, and not to introduce a separate category $\mathbf{VP}_i$.

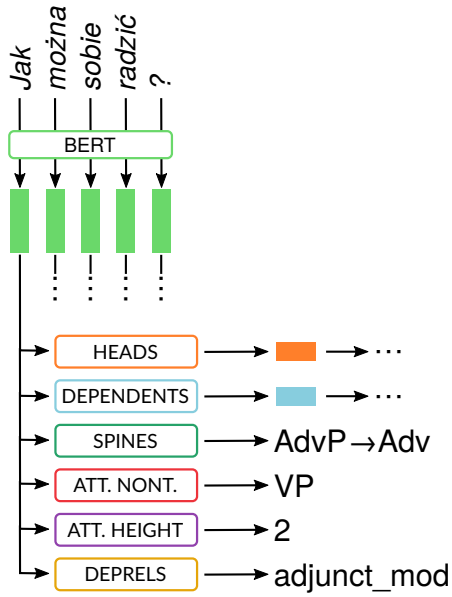

Figure 6: The model architecture

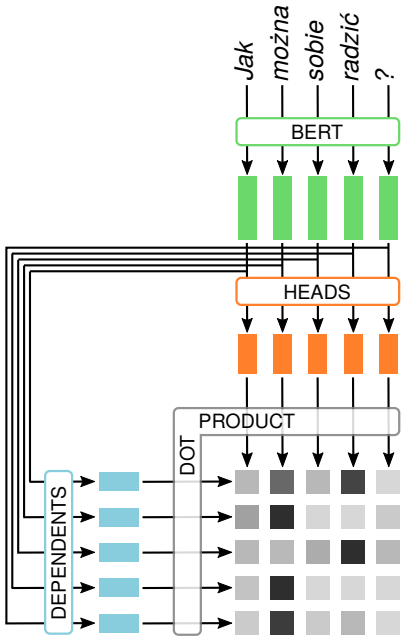

Figure 7: How the adjacency matrix is predicted

For prediction of dependency edges, we use a classifier based on that used in COMBO (Klimaszewski and Wróblewska, 2021) which in turn utilises an idea related to that of Dozat and Manning (2017).[6] The classifier, shown schematically in Fig. 7, consists in two dense layers (HEADS and DEPENDENTS, also indicated in Fig. 6) producing intermediate representations for all tokens. The dot-product of the two layers is then calculated, resulting in an $n \times n$ adjacency matrix for a sequence of n tokens. The i-th row of the matrix represents logits for particular tokens being the dependency head of the i-th token.[7]

We implement our parser using a version of the Huggingface Transformers[8] `TFBertForTokenClassification` architecture modified for multiple classifiers on top of single BERT model. We use HerBert[9] (Mroczkowski et al., 2021) for Polish and German BERT[10] for German.

BERT-style models perform their own tokenisation as a preprocessing step, often splitting text words into smaller subtokens. Therefore, we use the common technique of masking the BERT outputs for all but one (in our experiments: first) subtoken of each word when calculating the loss function. At parsing time, we assign the label predicted for the first subtoken to the whole word.

A headed constituency tree is reconstructed from the model's separate outputs as follows:

1. Apply row-wise softmax to the adjacency matrix output and use the Maximum Spanning Tree algorithm to obtain a dependency tree.[11]
2. Build spines on top of all tokens according to the model's predictions.
3. Attach each non-root token's spine to its dependency head's spine at the node pointed to by the two predicted attachment labels (category and level).
4. Label the resulting non-spine edges with predicted dependency labels.

Step 3 of the procedure above may involve some discrepancies between the dependency and spine/attachment components of the model output. In such cases we employ simple heuristic rules to assure well-formedness of the resulting structure:

- Ensure that a spine contains a **ROOT** only as its topmost node and iff it's assigned to the dependency root.
- Introduce additional nodes along spines whenever the attachment prediction calls for it (e.g. a **VP**-2 attachment to **VP** → **V** requires expanding the spine to **VP** → **VP** → **V**).

[6] The main difference between our approach and that implemented in COMBO is that we directly use vectors produced by fine-tuned BERT, while COMBO takes token representations from frozen (not fine-tuned) language models and passes them through a dedicated layer to obtain task-specific vectors. This possibly contributes to the improvement over COMBO's performance as reported in Section 7.

[7] We assume that the root token is its own head for the purpose of adjacency matrix prediction.

[8] https://huggingface.co/docs/transformers

[9] https://huggingface.co/allegro/herbert-large-cased

[10] https://huggingface.co/bert-base-german-cased

[11] This component situates our system among the graph based approaches to parsing.

- Attach to the topmost node if no node of required category is found along the spine (e.g. attach to **VP** if **NP**-1 attachment to **VP** → **V** spine is predicted by the model).

- If the models' predictions call for attaching to an empty spine, attach to the nearest dependency ancestor with a non-empty spine instead.

6. Related Work

The relation between dependency and constituency structures has been explored in various works both from the theoretical and practical, parsing perspective. In particular, several methods for hybrid or joint parsing of such structures have been proposed. For example the model of Fernández-González and Gómez-Rodríguez (2022) includes two separate decoders for the two types of trees. In result, no structural dependence between the two structures is imposed. (On the other hand, this work is related to our approach in that its constituency component leverages dependency parsing).

A more closely coupled prediction of both structures can be found in the work of Zhou and Zhao (2019); Zhou et al. (2020), who call their joint structure "simplified HPSG". However, the authors do not assume full compatibility of the predicted structures, which leads to introducing additional technical devices to accommodate for the discrepancies.

Constituency structures with marked heads can be traced back to lexicalised formalisms like PCFGs (Collins, 1997) and LTAGs (Carreras et al., 2008).

The idea of the spine as a linguistic unit coincides with the notion of the projection path in X-bar theory (Jackendoff, 1977), although the annotations in our training data do not have to conform to the X-bar assumptions.

An example of a technique centered around the notion of the syntactic spine are the transition-based parsers that construct the spines in a stepwise manner (constituency node by constituency node) (Ballesteros and Carreras, 2015, 2017).

To the best of our knowledge, the present work is the first to treat spines as atomic values in parsing. In result, spines can be used in a graph-based parser. The idea to predict spines at given token positions is similar to the conception of 'parsing as tagging' (e.g. Gómez-Rodríguez and Vilares, 2018). It can also be related, albeit more loosely, to the 'parsing as supertagging' approach popular in the 90s (e.g. Bangalore and Joshi, 1999).

There is a substantial line of work devoted to handling discontinuities in constituency parsing, including the following. Fernández-González and Martins (2015) present an approach that is very related to ours in that it leverages dependency parsing for (also discontinuous) constituency parsing. Coavoux and Cohen (2019) acommodate the transition-based approach by replacing the standard stack with a random-access set. Corro (2020) proposes a chart parser with a neural component that operates on discontinuous spans restricted to a single gap. Fernández-González and Gómez-Rodríguez (2021) train an additional component, i.e. a pointer network that reorders tokens of a possibly discontinuous sentence so that a continuous parser can be leveraged.

7. Experiments and evaluation

Models for both languages were trained using Adam optimiser with a learning rate of $1 \cdot 10^{-5}$ and categorical cross-entropy loss summed over all classifiers. The best model was selected using average accuracy on validation data across classifiers. The training patience was set to 4 epochs (i.e. training was stopped when no improvement in accuracy was achieved for 4 epochs).

Table 4 presents our parser's performance on Polish and German data, assessed by the following metrics:

- **UAS** and **LAS**: Unlabeled/Labeled Attachment Score, as typically used for dependency parsing,
- **brackets** F1:[12] correctly predicted constituents (token spans only, possibly discontinuous),
- **constituents** F1: correctly predicted constituents with correct category of their dominating nonterminal,
- **headed constituents** F1: correctly predicted constituents with correct category of their dominating nonterminal and correct decision whether the nonterminal is a head,
- **edges** F1: correctly predicted constituency parent-child edges. In order to calculate this measure, we encode each edge using the categories of its beginning and end node, the indices of the two nodes' spines, and the dependency label along the edge (if present).[13]

For brackets and constituents metrics, for our two main models, we also provide metric values calculated on discontinuous constituents only.

There is a visible gap between the brackets and constituents F1-scores for Polish and German data (ca. 4–5pp). We also notice that the annotation schemes adopted in the respective treebanks make those measures somewhat less comparable across

[12] F1 is the harmonic mean of precision and recall.

[13] For example, in the tree from Fig. 4, the set of edges is: <**VROOT**, **S**, 2, 2, _>, <**S**, *sind*, 2, 2, _>, <**S**, **NP**, 2, 1, SB>, <**NP**, *Wir*, 1, 1, _>, <**NP**, *beide*, 1, 3, MNR>, <**S**, **AP**, 2, 5, PD>, <**AP**, *ganz*, 5, 4, MO>, <**AP**, *realitätsnah*, 5, 4, M)>, <**VROOT**, ., 2, 6, -->.

lang.	model	train	test	UAS	LAS	brackets	constit.	h. constit.	edges
PL	**our**	OUR	OUR	**96.29%**	**91.04%**	97.80%	97.77%	97.25%	95.89%
		discontinuous				53.04%	52.15%	51.75%	
	COMBO	OUR	OUR	94.63%	89.33%	—	—	—	—
	our	UD 2.9	UD 2.9	**96.53%**	**94.94%**	—	—	—	—
⋆	COMBO	UD 2.9	UD 2.9	95.60%	93.93%	—	—	—	—
	our	OUR-C	OUR-C	96.39%	91.12%	98.03%	97.93%	97.46%	96.02%
	BeNePar	OUR-C	OUR-C	—	—	**98.20%**	**98.10%**	—	—
⋆	BeNePar	SPMRL	SPMRL	—	—	—	97.15%	—	—
DE	**our**	OUR	OUR	96.21%	94.96%	93.77%	92.58%	92.16%	94.80%
		discontinuous				74.84%	73.88%	73.86%	
	our	OUR-C	OUR-C	96.38%	95.00%	94.43%	93.11%	92.64%	94.65%
	BeNePar	OUR-C	OUR-C	—	—	**95.59%**	**94.41%**	—	—
⋆	BeNePar	SPMRL	SPMRL	—	—	—	92.10%	—	—

Table 4: Evaluation results. Rows marked with ⋆ contain results reported by authors of respective parsers.

the two languages. The Tiger constituents have a much flatter structure than the Składnica ones.[14] This means that a local error in a tree predicted by a parser can potentially affect a greater fraction of Tiger constituents, leading to a higher penalty in F1-score. This is why we also use the more fine-grained edges measure. In line with our expectations, the absolute difference across edges F1-scores is lower and amounts to ca. 1pp.

It is not straightforward to compare our results directly with other parsers since, to our knowledge, there were no experiments with parsing headed constituencies for Polish nor German. Therefore, we conducted some experiments concentrating on either dependency or (non-headed) constituency performance of our models in comparison with two state-of-the art parsing systems: COMBO (best available dependency system for Polish, Klimaszewski and Wróblewska, 2021) and Berkeley Neural Parser (BeNePar, constituency, Kitaev et al., 2019). The additional experiments, together with relevant results reported by the authors of both parsers, are also listed in Table 4.[15]

For the purpose of comparison with COMBO, we trained and tested COMBO on the dependency component of our Polish dataset. In addition, we trained and tested the dependency classifiers (heads and dependency relations) of our model on Polish UD 2.9 data, which has a different dependency annotation scheme, to provide comparison with the results reported by Klimaszewski and Wróblewska (2021). Our model outperforms COMBO on both datasets.

Comparison with BeNePar is less straightforward since the SPMRL data (Seddah et al., 2013) on which it was tested by its authors does not contain trees with annotations suitable for our parser.[16] Instead, in order to achieve a meaningful comparison, we trained and evaluated both our model and BeNePar on subsets of our respective dataset splits containing continuous trees only (OUR-C). Nonetheless, we include the results[17] given in Kitaev et al. (2019), since their test data is very similar to OUR-C: the Polish dataset in SPMRL is an older continuous version of Składnica (including short sentences and a limited set of grammatical constructions), and the German set is a continuous version of Tiger. Reassuringly, these figures turn out to be similar to ours.

Our model performs worse than BeNePar (ca. 1.3 pp on German data and ca. 0.2 pp on Polish data). Nevertheless, we consider those results encouraging, especially taking into account the fact that our models are able to analyse discontinuous structures.

[14] Compare, for example, the trees from Figures 4 and 5: similarly short sentences have trees with 4 (German) and 12 (Polish) nonterminals. Also note the different distributions of spine lengths in Table 2.

[15] The empty cells in the table correspond to situations where either the evaluated system does not perform dependency/constituency parsing or the test data does not contain dependency/constituency annotations.

[16] The SPMRL data contains both constituency and dependency structures for the same sets of sentences, but they are not fully consistent with each other and therefore do not easily provide headed constituencies as required by our approach.

[17] Note that they use the SPMRL EVALB evaluation which is different from ours in that it, e.g., excludes some labels.

8. Limitations

One limitation of our approach to parsing lies in data availability. The method requires headed constituencies: a specific form of syntactic annotation that is not readily available for many languages. In result, the method has only been tested on two languages. The parser architecture does not include any language-dependent elements and various BERT-style models, either single- or multi-language, are available. Thus, the method can be reasonably expected to work with any language. However, since we do not have an exact picture, it may be that for some structurally very different language the results would be different.

Both tested datasets are based on constituency schemes where a node may have an arbitrary number of children. This results in rather short spines. A treebank comprising binary trees would have longer and potentially more numerous spines, which might cause a problem with handling them as atomic labels. In that case the method of predicting spines could be changed to recurrently generate individual nodes within spines, which seems possible with current neural architectures. However, we haven't explored this path yet.

9. Conclusions

We have proposed a method for parsing headed constituencies, which are structures combining constituency and dependency information. The structure exploits strengths of both representations. In particular constituents provide natural representation for coordinated structures (which are problematic in dependency trees), but the dependencies between tokens (and constituents) are also available in the structure.

In the present paper, we describe a complete parser, generating headed constituency trees. We introduce an architecture combining the ideas of spines and attachments and 'parsing as tagging', leveraging COMBO-like adjacency matrix prediction and BERT fine-tuning. To the best of our knowledge, we present the first graph-based parser based on spines. We show that, thanks to the power of BERT models, the spines can be successfully predicted as atomic values. An important feature of the proposed method is the ability to analyse discontinuous structures.

Our test results show that the architecture generates state-of-the-art results for both constituency and dependency trees. In particular, we improve SOTA for Polish dependency parsing by 1 pp (UAS and LAS).

10. Acknowledgements

Work supported by POIR.04.02.00-00-D006/20-00 national grant (Digital Research Infrastructure for the Arts and Humanities DARIAH-PL).

11. Bibliographical References

Miguel Ballesteros and Xavier Carreras. 2015. Transition-based spinal parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 289–299, Beijing, China. Association for Computational Linguistics.

Miguel Ballesteros and Xavier Carreras. 2017. Arc-standard spinal parsing with stack-LSTMs. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 115–121, Pisa, Italy. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Journal of Language and Computation*, 2:597–620.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16, Manchester, England. Coling 2008 Organizing Committee.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 204–217, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Caio Corro. 2020. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from $O(n^6)$ down to $O(n^3)$. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2753–2764, Online. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing.

Agnieszka Falenska, Zoltán Czesznak, Kerstin Jung, Moritz Völkel, Wolfgang Seeker, and Jonas Kuhn. 2020. GRAIN-S: Manually annotated syntax for German interviews. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5169–5177, Marseille, France. European Language Resources Association.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021. Reducing discontinuous to continuous parsing with pointer network reordering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10570–10578, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *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)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2022. Multitask Pointer Network for multi-representational parsing. *Knowledge-Based Systems*, 236:107760.

Kilian Gebhardt, Mark-Jan Nederhof, and Heiko Vogler. 2017. Hybrid Grammars for Parsing of Discontinuous Phrase Structures and Non-Projective Dependency Structures. *Computational Linguistics*, 43(3):465–520.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Universal Dependencies Workshop 2018*, Brussels, Belgium.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Ray Jackendoff. 1977. *X-bar Syntax: A Study of Phrase Structure*. Number 2 in Linguistic Inquiry Monographs. MIT Press.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Mateusz Klimaszewski and Alina Wróblewska. 2021. COMBO: State-of-the-art morphosyntactic analysis. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 50–62, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Katarzyna Krasnowska-Kieraś and Marcin Woliński. 2023. Constituency parsing with spines and attachments. In *Computational Science – ICCS 2023. 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, Proceedings, Part I*, volume 14073 of *Lecture Notes in Computer Science*, pages 191–205, Cham. Springer Nature Switzerland.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Robert Mroczkowski, Piotr Rybak, Alina Wróblewska, and Ireneusz Gawlik. 2021. HerBERT: Efficiently pretrained transformer-based language model for Polish. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajic, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis M. Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. *CoRR*, abs/2004.10643.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared

task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Marcin Woliński. 2019. *Automatyczna analiza składnikowa języka polskiego*. Wydawnictwa Uniwersytetu Warszawskiego, Warsaw.

Marcin Woliński and Elżbieta Hajnicz. 2021. Składnica: a constituency treebank of Polish harmonised with the Walenty valency dictionary. *Language Resources and Evaluation*, 55:209–239.

Alina Wróblewska. 2014. *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Junru Zhou, Zuchao Li, and Hai Zhao. 2020. Parsing all: Syntax and semantics, dependencies and spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4438–4449, Online. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*, number 6231 in Lecture Notes in Artificial Intelligence, pages 197–204, Heidelberg. Springer-Verlag.

12. Language Resource References

Tiger Corpus Team. 2006. *TIGER Corpus*. Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung. PID https://www.ims.uni-stuttgart.de/en/research/resources/corpora/tiger.

Woliński, Marcin. 2019. *Składnica*. Institute of Computer Science, Polish Academy of Sciences. PID http://zil.ipipan.waw.pl/Składnica.

Wróblewska, Alina. 2022. *Polish Dependency Bank*. Institute of Computer Science, Polish Academy of Sciences. PID http://zil.ipipan.waw.pl/PDB.