Constituency Parsing with Spines and Attachments[⋆]

Katarzyna Krasnowska-Kieraś[0000−0002−7052−0568] and Marcin Woliński[0000−0002−7498−1484]

Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
`http://zil.ipipan.waw.pl {k.krasnowska,m.wolinski}@ipipan.waw.pl`

Abstract. We propose a hybrid representation of syntactic structures, combining constituency and dependency information. The headed constituency trees that we use offer the advantages of both those approaches to representing syntactic relations within a sentence, with a focus on consistency between them. Based on this representation, we introduce a new constituency parsing technique capable of handling discontinuous structures. The presented approach is centred around head paths in the constituency tree that we refer to as spines and the attachments between them. Our architecture leverages a dependency parser and a large BERT model and achieves 95.96% F1 score on a dataset where ≈10% of trees contain discontinuities.

Keywords: Constituency parsing · Headed constituencies · Syntactic structures.

1 Introduction

Syntactic parsing is an important NLP task often used in various text processing pipelines. In this context, syntactic structures are usually represented with 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. This is the task of hybrid parsing. Our goal, however, is to fulfil one more requirement: the structures have to be consistent with each other. Two tokens should be connected with a dependency relation in the dependency tree if and only if there exists a constituent in the constituency tree whose yield contains both tokens.

We plan to create a large parsebank of Polish annotated with both types of structures. The treebank search engine will allow to query both constituency and

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

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 provide a hybrid parsing method which would provide consistent constituency and dependency structures.

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*[1] '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 **S** node, dependencies go from the verbal phrase **VP** (the head) to two nominal phrases (labelled subj and obj) and to a prepositional adjunct. (A technical relation punct is used to take care of punctuation.)

This procedure leads to the creation of a *headed constituency tree* in which each constituent has a head among its children and each non-head child is connected with the head child using a labelled 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. 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).

[1] We use examples in Polish, since its system of 7 grammatical cases makes the grammatical relations more easily visible than in English.

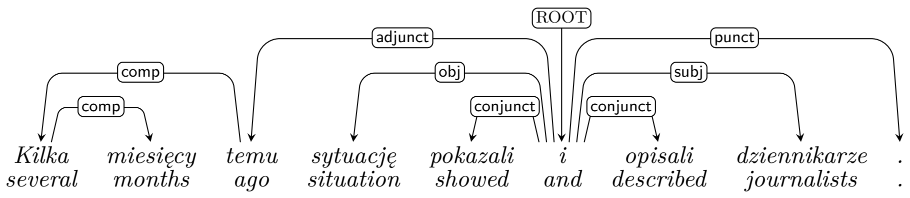

Fig. 1. Dependency tree for the sentence: *Kilka miesięcy temu sytuację pokazali i opisali dziennikarze.* 'The journalists exposed and described the situation several months ago.'

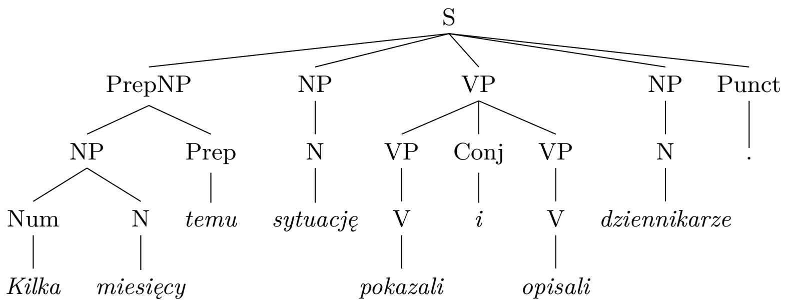

Fig. 2. Constituency tree for the same sentence.

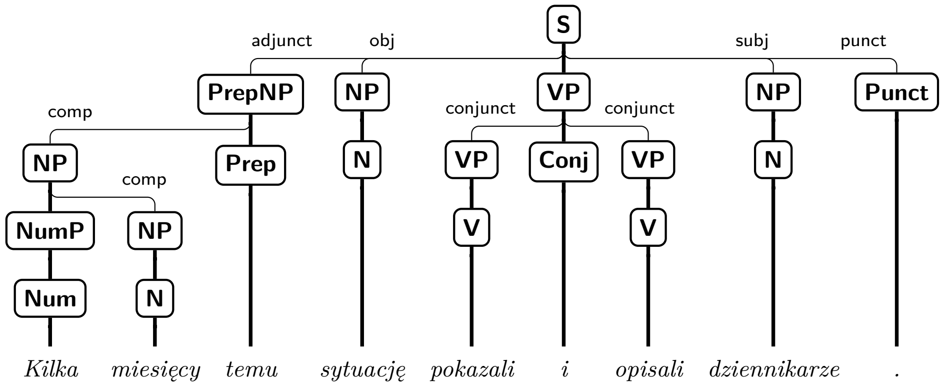

Fig. 3. Headed constituency tree for the same sentence. Bold line joins a constituent with its head child.

3 The Dataset

To perform the experiments we need a constituency treebank consistent with a dependency one. For that reason we chose to work with a Polish dataset built on the *Składnica* constituency treebank[2] (Woliński and Hajnicz, 2021; Woliński, 2019; Świdziński and Woliński, 2010) and the Polish Dependency Bank[3] (PDB Wróblewska, 2014).

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

The size of the dataset, which we were able to create by merging information from Składnica and PDB, is reported in Table 1.[4]

4 Proposed 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*. 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

[2] `http://zil.ipipan.waw.pl/Sk%C5%82adnica`

[3] `http://zil.ipipan.waw.pl/PDB`

[4] See `http://git.nlp.ipipan.waw.pl/constituency/spines-attachments` for the code and dataset.

Table 1. The sizes of the Polish dataset used in this work

		continuous	discontinuous	total
train	trees	15903	1756 (9.9%)	**17659**
	tokens	239531	40515 (14.5%)	280046
	avg. tokens/tree	15.06	23.07	15.86
validation	trees	1980	231 (10.4%)	**2211**
	tokens	29531	5034 (14.6%)	34565
	avg. tokens/tree	14.91	21.79	15.63
test	trees	1990	215 (9.8%)	**2205**
	tokens	28529	4815 (14.4%)	33344
	avg. tokens/tree	14.34	22.40	15.12

attached to a verbal spine. Spines with a conjunction as a base are more context dependent: the higher nodes depend 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 with no preterminal node). Suppose that token t_i is the dependency head of another token t_j, and their corresponding spines are $s_i = n_{i,1} \to ... \to n_{i,k}$ and $s_j = n_{j,1} \to ... \to n_{j,l}$. 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** → **Num**, s_2 = **N**, s_3 = **PrepNP** → **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

Constituency parsing can be decomposed into (1) determining the spines for each token of the input and (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).

Part (2a) corresponds exactly to determining the dependency structure between tokens. If we use a dependency parser for this part, the resulting constituency trees will be consistent with dependency trees produced by this parser.[5]

A very desirable trait of the proposed method is that if the task (2a) is performed by a dependency parser capable of parsing discontinuous constructions, the resulting constituency parser becomes immune to the problem of discontinuity. Morover, with this technique the other typical problem in 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 *jeszcze trzy tygodnie* 'three more weeks' in the tree shown in Fig. 4 reflects its hierarchical structure $(\textit{jeszcze}(\textit{trzy tygodnie})_{\mathsf{NP}})_{\mathsf{NP}}$, with $\mathsf{Part} \rightarrow$ *jeszcze* 'more' and $\mathsf{NP} \rightarrow \mathsf{N} \rightarrow$ *tygodnie* 'weeks' attached to $\mathsf{NP}_2 \rightarrow \mathsf{NP}_1 \rightarrow \mathsf{NumP} \rightarrow \mathsf{Num} \rightarrow$ *trzy* 'three' as non-head children of NP_2 and NP_1 respectively.[6] Note that such $\mathsf{X}_2 \rightarrow \mathsf{X}_1$ sequence appearing along a spine means that X_2 must have at least one non-head child apart from its head child X_1, since there are no $\mathsf{X} \rightarrow \mathsf{X}$ unary branchings in our trees. Moreover, the spines resulting from our trees do not contain $\mathsf{X} \rightarrow \mathsf{Y} \rightarrow \ldots \rightarrow \mathsf{X}$ sequences (where $\mathsf{X} \neq \mathsf{Y}$).

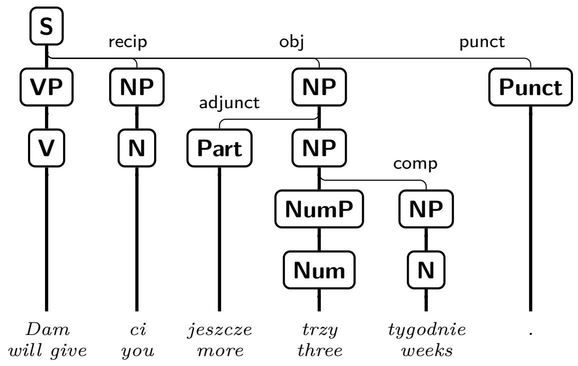

Fig. 4. A tree with an $\mathsf{NP} \rightarrow \mathsf{NP}$ edge along a spine: 'I will give you three more weeks.'

The attachment information consists of the category of the node a spine attaches to and its number 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.

[5] 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.

[6] We use the lower subscript NP_i to differentiate between two different NP nodes, and not to introduce a separate category NP_i.

Table 2. The tree of Fig. 4 encoded with dependency relations, spines and attachments

token	head ID	deprel	spine	attachment
Dam	0	root	**S** → **VP** → **V**	**root**
ci	1	recip	**NP** → **N**	**S**-1
jeszcze	4	adjunct	**Part**	**NP**-2
trzy	1	obj	**NP** → **NP** → **NumP** → **Num**	**S**-1
tygodnie	4	comp	**NP** → **N**	**NP**-1
.	1	punct	**Punct**	**S**-1

As an example, consider the tree from Fig. 4 and its representation shown in Table 2. The spines for *ci* 'you', *trzy* 'three' and the final punctuation are all attached as children of the same (and only) **S** node along the 'root' spine for *Dam* '(I) will give'. Thererefore, they all have the same **S**-1 attachment. The spine for *trzy* contains a sequence of two **NP** nodes. The spine for its one dependent, *tygodnie*, is attached to the first **NP** from the bottom (**NP**-1) and the spine for the other dependent *jeszcze* is attached to the second **NP** from the bottom (**NP**-2). The first two columns of the table contain the dependency relations between tokens following a CoNLL(-U)-like convention.

We also note that, considering the above observations about **X** → **X** head edges, we can alternatively use a 'compressed' representation of spines where any sequence of **X** nodes is collapsed into one **X** node (in practice, there are only sequences of two such nodes in our data). Such modification does not result in any information loss, since any repetitions of same-labeled nodes along spines are retained in the attachment representation. In the example from Table 2, we could represent the spine for *trzy* as **NP** → **NumP** → **Num**, and the information that there are in fact two **NP** nodes is carried by the **NP**-2 attachment of *jeszcze*. Such variant of representation reduces the number of different spines. There are 110 distinct spines in our data, producing 70 distinct compressed spines.

Table 3 presents spines ending with **N** and **V** preterminals as an example. The total number of occurrences in our data, as well as the percentage of occurrences among spines with given preterminal, are given for each spine type. In the case of **N** spines, there are 3 types corresponding to types of phrase built around a nominal token: noun phrase with modifiers (hence the repetition of the **NP** node) nested as a non-head in the sentence structure, a nested noun phrase without modifiers, and a noun phrase serving as the syntactic centre of an utterance without a predicate. Fig. 5 shows example tree contexts for four **V** spines: main predicate (*wymienia*), relative clause (*zorganizował*), coordinated sentences (*Było*, *płonęły*) and coordinated verbal phrases (*fascynowały*, *przerażały*).

5 Parser Architecture

In our approach to constituency parsing, we leverage an already available, well performing dependency parser and combine its analyses with our own constituency component predicting spines and their attachments.

Table 3. Nominal and verbal spines.

spine	occurrences in data	
NP→**NP**→**N**	57595	52.84%
NP→**N**	50864	46.66%
ROOT→**NP**→**NP**→**N**	548	0.50%
ROOT→**S**→**VP**→**V**	16406	36.36%
VP→**VP**→**V**	12536	27.78%
S→**VP**→**V**	12037	26.68%
CP→**S**→**VP**→**V**	2407	5.33%
VP→**V**	1725	3.82%
V	7	0.02%
S→**S**→**VP**→**V**	3	0.01%
CP→**S**→**S**→**VP**→**V**	1	<0.01%
ROOT→**S**→**VP**→**VP**→**V**	1	<0.01%
S→**VP**→**VP**→**V**	1	<0.01%

As the dependency component, we use the best available dependency parser for Polish which is COMBO[7] (Klimaszewski and Wróblewska, 2021). The published pre-trained models for COMBO operate on UD and are therefore incompatible with the PDB dependency annotation scheme used in our data. Therefore, we trained a dedicated COMBO model on the train portion of our data. The model achieved 96.3% UAS/93.2% LAS on training data and 94.9% UAS/89.8% LAS on the validation portion of our data.

For the constituency component, we used a modified version of the Huggingface Transformers[8] `TFBertForTokenClassification` architecture. The architecture is a simple dense layer classifier added on top of a pre-trained BERT model, predicting a label for each input token. For each token, the classifier is applied to the contextualised vector representation of the token produced by the BERT model, and predicts a vector of logits for all possible classification labels. The whole architecture is then trained, with the classifier being fitted from scratch, and the weights of the BERT model fine-tuned to the specific task. Our modification consisted in adding multiple dense layer classifiers, allowing us to jointly train predictors for several objectives, while fine-tuning the BERT model to all of them. As the BERT model, we use HerBert[9] (Mroczkowski et al., 2021).

The constituency component's model tested in this work involves 3 classifiers, predicting for each token:

– its spine,[10]
– the category of its spine's attachment node,

[7] `https://wiki.clarin-pl.eu/pl/nlpws/services/COMBO`
[8] `https://huggingface.co/docs/transformers`
[9] `https://huggingface.co/allegro/herbert-large-cased`
[10] Since the number of distinct spines is fairly limited, we decided to treat them as atomic labels.

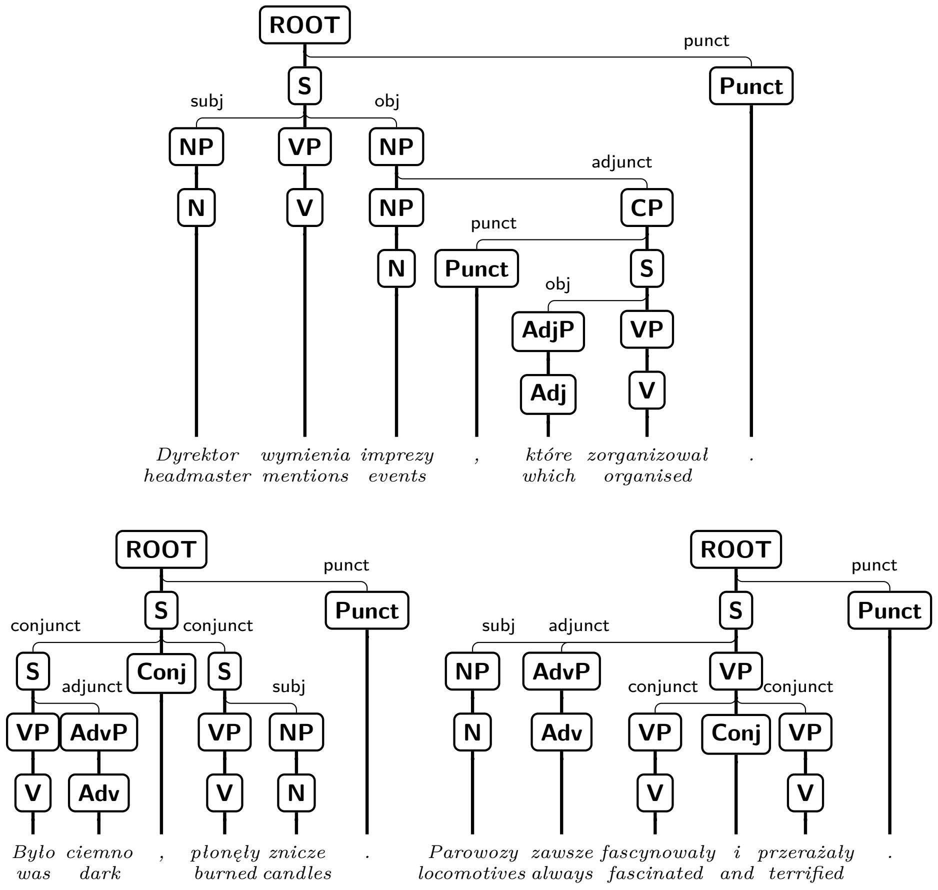

Fig. 5. Examples of verbal spines: 'The headmaster mentions events he organised.', 'It was dark, candles were burning.', 'Steam locomotives have always fascinated and terrified people.'

– the number of its spine's attachment node counting from the bottom.

The BERT-style models perform their own tokenisation as a preprocessing step. This tokenisation often splits text words into smaller segments, according to the specific models' tokeniser vocabulary. As a result, for one input sentence token, several BERT vectors can be produced and passed to the final classifiers, leading to multiple (possibly incoherent) predictions associated with the same token. Therefore, we use the common technique of masking the BERT outputs for all but one segment of each token when calculating the loss function. In our experiments, we chose to only leave the first segment unmasked.

Given the outputs of the dependency and constituency components of our parsing architecture, reconstructing a headed constituency tree is straightforward. First, in order to ensure a basic well-formedness of the produced structure, we remove any **ROOT** nodes from spines recognised as non-root by the dependency component, and add a **ROOT** node on top of the root token's spine if missing. Then we 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 number). If there are not enough consecutive nodes of given category along the head's spine, we insert them on top of the existing ones. If there are no nodes of the requested category, we use the head spine's topmost node as a fallback attachment site. Finally, we collapse any unary **X** $\rightarrow$ **X** branches that could result from the above procedure (e.g. if there was an **X**-2 attachment along the spine, but no **X**-1 attachment).

6 Related Work

The proposed method is in a close relation to the attempts at hybrid dependency/constituency parsing. In particular the structure we propose is similar to what Zhou and Zhao (2019); Zhou et al. (2020) call "simplified HPSG". However the authors do not assume full compatibility of the structures, which leads to the necessity of artificial nodes which accommodate for the discrepancies.

There is a substantial line of work devoted to acommodating transition-based constituency parsing to discontinuities, e.g. Coavoux and Cohen (2019). 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.

As for continuous constituency parsing, Kitaev et al. (2019) quote 96.36% as F1 score of their neural chart-based parser for Polish. This figure is to some extent suitable for comparison with our experiments since it is calculated on the data of (Seddah et al., 2013) which is in fact an old version of Składnica (including short sentences and simpler grammatical constructions).

Our idea to gather all constituents with the same token being the centre, forming a spine, is, to the best of our knowledge, new. 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). The concept of decorating edges of a constituency tree with relations was present already in the TIGER treebank of German (Brants et al., 2004), but their structures were not strictly headed (the relations did not form a dependency tree).

7 Evaluation

We evaluate our parser in terms of precision (P), recall (R) and F1 metrics over constituents in three variants. The evaluation units are the (possibly discontinuous) yields of all nonterminals of a tree paired with (depending on the metric variant):

– constituents: the nonterminal's syntactic category,
– headed constituents: the nonterminal's syntactic category and information whether the nonterminal is a head node,

Table 4. Results on validation and test data.

	validation data			test data		
	precision	recall	F1	precision	recall	F1
non-compressed						
bracketing	96.40%	96.25%	96.32%	96.56%	96.45%	**96.51%**
constituents	96.46%	96.33%	96.39%	96.64%	96.53%	**96.59%**
headed constit.	95.70%	95.93%	95.81%	95.82%	96.09%	**95.96%**
compressed						
bracketing	96.28%	96.23%	96.26%	96.41%	96.48%	**96.44%**
constituents	96.40%	96.25%	96.32%	96.57%	96.48%	**96.52%**
headed constit.	95.81%	95.67%	95.74%	95.96%	95.88%	**95.92%**

– bracketing only: no additional information (i.e. a constituent needs only to be recognised, regardless of its assigned syntactic category).[11]

All the models for the constituency component were trained using Adam optimiser with learning rate of $2 \cdot 10^{-5}$ and categorical cross-entropy loss summed over all classifiers. The best model was selected using average accuracy on validation data across classifiers. The training was stopped when no better accuracy was achieved for 4 epochs.[12] We trained two variants of the model, using non-compressed and compressed spines. We show the results in Table 4. The difference in performance of both models is negligible, therefore in further analysis we concentrate on the non-compressed version which achieved 95.96% headed constituents F1-score on test portion of our dataset.

In Table 5, we examine how the constituency component of our architecture performs when it comes to predicting spines for particular tokens. The overall accuracy of spine prediction of test data was 97.13%. We calculate aggregate precision, recall and F1[13] on test data for spines ending with particular preterminals in order to check which kinds of spines are more difficult for the parser. We are not surprised by the **Conj** spines turning out to be the hardest to correctly assign since, in our annotation scheme, they are associated with coordination. One could expect a (near) 100% figure for punctuation which should be very easy for the model to learn. However, some occurences of punctuation are annotated as

[11] For the bracketings metric, each span is counted only one time, e.g. for the tree in Fig. 4, $(Dam)_{\mathsf{VP}}$ and $(Dam)_{\mathsf{V}}$ are treated as the same span (Dam) etc.

[12] We noticed that when validation data loss was used for early stopping and model selection, the accuracies on validation data still exhibited a growing tendency.

[13] Let TP_l, FP_l, FN_l denote the number of true positives, false positives and false negatives respectively for label l in evaluation data. For a set of labels S, we calculate the aggregate precision P_S as $(\sum_{l \in S} TP_l)/(\sum_{l \in S} TP_l + FP_l)$, i.e. the proportion of correctly predicted labels from S to all predicted labels from S. The aggregate recall R_S is $(\sum_{l \in S} TP_l)/(\sum_{l \in S} TP_l + FN_l)$, i.e. the proportion of correctly predicted labels from S to all gold labels from S. The aggregate $\mathrm{F1}_S$ is the harmonic mean of P_S and R_S.

Table 5. Aggregate precision, recall and F1 for spine types.

spine	precision	recall	F1	occurrences
... → **Punct**	99.03%	99.36%	99.20%	5158
... → **Prep**	98.31%	98.39%	98.35%	3488
∅	98.78%	97.88%	98.33%	330
... → **Adj**	98.00%	98.41%	98.20%	4079
... → **Part**	98.69%	97.42%	98.05%	1627
... → **N**	98.07%	97.95%	98.01%	10435
... → **Comp**	97.08%	97.51%	97.29%	682
... → **V**	95.47%	95.45%	95.46%	4398
... → **Adv**	95.11%	95.74%	95.42%	1056
... → **Num**	90.87%	90.47%	90.67%	451
... → **Conj**	85.43%	84.68%	85.05%	1364

conjunctions in our data, so their classification is not entirely trivial. We also note that **V** spines are, as a whole, predicted with worse results than the **N** ones.

For a more detailed look into that last osbervation, we also report, in Table 6 precision, recall and F1-score on test data for individual nominal and verbal spines (as shown in Table 3) that appear in the test data portion. For **N** spines, we note that the results for two most common cases (modified vs non-modified noun) the parser performs comparably well. The results for the spine corresponding to noun-centered utterances are visibly lower, but it should be noted that they are two orders of magnitude less frequent in the data, which probably makes learning them substantially harder. As far as the **V** spines are concerned, the results seem to reflect the diversity of syntactic contexts in which particular spines may appear. The **ROOT**-dominated spine corresponds to the main predicate, a **CP** has a specific context, and they are both recognised with a relatively high F1. Meanwhile, the **S**- and **VP**-dominated spines may be associated with a variety of syntactic phenomena: coordination, subordinate infinitival phrase, subordinate clause.

We also performed an experiment comparing our architecture with one of the state-of-the-art constituency parsers, the Berkeley Neural Parser[14] (Benepar, Kitaev et al. 2019). Since Benepar only operates on continuous constituency structures, we trained and evaluated it on the continuous portions of our data, and compared the results with an instance of our architecture trained on the same data (for both the dependency and spine-attachment components). Moreover, as Benepar does not have an explicit mechanism for handling syntactic heads, we represented head nodes by prepending a special character to their labels, thus

[14] `https://parser.kitaev.io/`

Table 6. Precision, recall and F1 for selected spines.

spine	precision	recall	F1	occurrences
NP → **NP** → **N**	98.51%	97.89%	98.20%	5458
NP → **N**	97.62%	98.17%	97.89%	4926
ROOT → **NP** → **NP** → **N**	95.45%	82.35%	88.42%	51
ROOT → **S** → **VP** → **V**	98.09%	99.10%	98.59%	1661
CP → **S** → **VP** → **V**	96.68%	97.14%	96.91%	210
VP → **VP** → **V**	91.89%	95.36%	93.60%	1165
S → **VP** → **V**	95.69%	91.18%	93.38%	1168
VP → **V**	92.47%	88.66%	90.53%	194

Table 7. Comparison with Benepar.

	validation data			**test data**		
	precision	recall	F1	precision	recall	**F1**
non-compressed, continuous only						
headed	96.05%	96.29%	96.17%	96.13%	96.38%	**96.26%**
Benepar, continuous only						
headed	97.44%	97.43%	97.44%	97.74%	97.75%	**97.75%**

creating separate labels for head and non-head constituents of each type and roughly doubling the number of possible labels. The results are presented in Table 7. Our architecture performs worse (by 1.5 pp F1-measure) than Benepar on this data. Nevertheless, we find this result encouraging. We believe that, given the inherent ability of our approach to handle discontinuous structures, it is worthwhile to aim at improving its general performance in future work.

8 Conclusions

In the paper, we have proposed a syntactic structure of headed constituencies and a method for parsing such structures.

We think that this structure, merging constituency and dependency information, is a handy model of syntax. 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.

The performed experiments show that with present neural models it is possible to express constituency parsing in terms of detecting spines and their attachments. For us, the most important feature of this method is that it generates constituency structures consistent with the dependencies. Another important aspect is the ability to process discontinuous structures, which are hard for many constituency parsers. The method was tested on Polish due to availability of the data, but the approach is not in any way specific to the language in question. Taking into the account that BERT-type models were successfully trained for various languages, we have reasons to believe that the proposed method would work comparably for other languages.

The achieved results of around 96% F1 measure for a dataset with discontinuities show that the proposed method is well in the state-of-the-art zone. Our result for continuous trees is lower than that of Berkeley Neural Parser, which shows that there is room for improvement, which we intend to explore. In this work, we used an external parser as the source of dependency edges. An interesting direction of future work would be to check whether integrating both parts within a joint model could provide some synergy in learning.

References

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.

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.

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.

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.

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.

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.