Re-thinking Supertags in Linear Context-free Rewriting Systems for Constituency Parsing

Anonymous ACL submission

Abstract

Recently, a supertagging-based approach for parsing discontinuous constituent trees with linear context-free rewriting systems (LCFRS) was introduced. We reformulate their algorithm for the extraction of supertags from treebanks to be more concise. Moreover, we add some extensions that give us control over the extraction process in terms of supertag granularity and which terminal symbols are associated with supertags. Our additions lead to an increase in parsing quality with LCFRS supertagging in all three compared treebanks. The scores are among the state of the art in discontinuous constituent parsing.

1 Introduction

Discontinuous constituency parsing deals with the task to find hierarchies of – possibly non-contiguous – phrases (*constituents*) in a given sentence of natural language and assigns a label (*constituent symbol*) to each phrase. Traditional approaches use grammar formalisms such as linear context-free rewriting systems (LCFRS) to model these hierarchies (Maier and Søgaard, 2008; Kallmeyer and Maier, 2013; van Cranenburgh et al., 2016; Gebhardt, 2020). Statistical parsing with these grammars is remarkably slow and inaccurate by today's standards. But they still find some attraction as both, the grammars and parsing with them, are easily interpretable. More recent parsers use neural classifiers and either leverage the parsing process into a linear task (Coavoux, 2021; Fernández-González and Gómez-Rodríguez, 2020, 2021b,a) or score constituent labels for selected phrases (Corro, 2020; Stanojević and Steedman, 2020).

In supertagging-based parsing (Bangalore and Joshi, 1999), a grammar is accompanied by a classifier that selects and scores a small sample of rules in the grammar. After that, the rules and their scores are interpreted as a weighted grammar which is used for parsing in the usual manner. A recent publication (Ruprecht and Mörbitz, 2021) showed that supertagging improved the quality and speed of parsing with LCFRS significantly, bringing it close to recent discontinuous parsing methods. However, their extraction process for supertags is rather convoluted and uses hard-wired options for, e.g., terminal transportation and binarization.

In Section 3, we will present a formulation of supertags and an extraction algorithm that is easier to grasp than the previous definition. In comparison, it is re-ordered such that LCFRS rules are assembled after it is determined which terminal symbol they are associated with; therefore we will avoid transporting terminal symbols through LCFRS rule derivations. Secondly, the annotations that were introduced to revert the terminal transportation will not be a necessary part of the LCFRS rules but a separate component of each supertag. Both changes give us the opportunity to introduce two new parameters to the extraction process: the *transportation guide* controls which terminal symbol is associated with each part in the constituent tree, and the *nonterminal constructor* is responsible for the granularity of the underlying grammar. We will give some instances for these two parameters. Section 4 describes our experiments with the extraction algorithm for the discontinuous English Penn Treebank (DPTB, Marcus et al., 1994; Evang and Kallmeyer, 2011), and the two German treebanks NeGra (Skut et al., 1998) and Tiger (Brants et al., 2004). It explains how we found viable configurations for the introduced parameters and gives results for parsing with them. The implementation will be published on GitHub.

2 Notation

A *discontinuous constituent tree* is a tuple (ξ, pos, w) as follows: w is a sequence of terminal symbols (*phrase*), pos is a sequence of part-of-

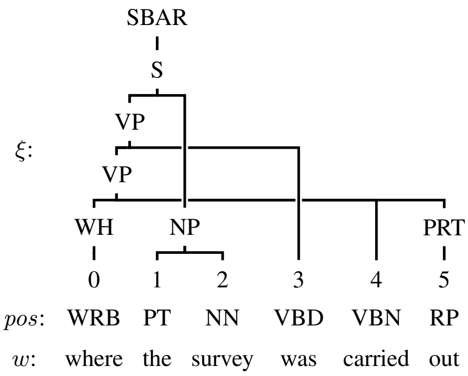

Figure 1: Discontinuous constituent tree for the phrase *where the survey was carried out*. The tree is illustrated with crossing branches, so that the leaves are ordered.

speech (*pos*) symbols with same length as w, and the *constituent structure* ξ is a tree; its inner nodes are *constituent symbols* and its leaves are *phrase positions* $0 \dots |w| - 1$ such that each leaf occurs exactly once in ξ. Figure 1 shows an example.

We use the usual notation for *(Gorn-)positions* in the constituent structure, i.e. each position determines exactly one node in ξ. The *set of all inner node positions* in ξ is denoted by $\mathrm{npos}(\xi)$. The *subtree of* ξ *at position* ρ is denoted by $\xi|_\rho$. The *yield* $\mathrm{yd}(\xi)$ is the set of leaves in ξ. The *fanout of (a set of leaves)* L is the smallest number of contiguous subsets of L. For instance, in Fig. 1, the yield of the subtree governed by the upper node labeled by VP is the set $\{\underline{0}, \underline{3, 4, 5}\}$, its fanout is 2. $\xi(\rho)$ denotes the constituent symbol at position ρ.

We briefly cover the notation for *binary lexical LCFRS*. Each rule is either of the form $A \to [w]$ or $A \to [u_1, \dots, u_k](B_1, B_2)$. A is called left-hand side (*lhs*) nonterminal, B_1, B_2 are right-hand side (*rhs*) nonterminals, and w is the rule's lexical symbol (or terminal). Each string $u_1, \dots, u_k$ consists of one lexical symbol and variables $x_1, \dots, x_n$ and $y_1, \dots, y_m$ where each x_i (resp. y_j) refers to a string produced by a first (resp. second) successor. They denote a function composing k strings from the lexical symbol and $n + m$ successor strings.

3 Contributions

A *supertag* is a tuple (r, t, c, p) where

- r is an LCFRS rule, its terminal is a wildcard,
- t is either *None* or an index that indicates from which successor an associated terminal originates,
- c is either *None* or a constituent symbol, and
- p is a pos symbol.

Previously, supertags were introduced as LCFRS rules with certain annotation that did not fit into the usual LCFRS framework but was necessary to convert supertag derivations into constituent trees. Our notation separates the information needed for the conversion from the grammar rule and hence allows us to generalize the extraction.

3.1 Extraction Parameters

Apart from the constituent treebank, our extraction algorithm expects parameters for binarization, a *guide* and a *nonterminal constructor*. The *vanilla* parameters coincide with the existing algorithm.

Binarization. We use the usual *binarization strategies* for constituent structures in parsing, (Kallmeyer and Maier, 2013) with factorization from left to right (*lr*) or head-outward (*ho*). Both strategies, ho and lr, are extended by markovization. The width of the horizontal markovization window is denoted by h, the vertical one by v.

Guide. A *guide* for ξ assigns a leaf to each inner node position. During the extraction, a supertag will be constructed for each inner node and its assigned leaf. Formally, a transportation guide for ξ is an injective function $\mathtt{G}\colon \mathrm{npos}(\xi) \to \mathrm{yd}(\xi)$ such that, for each $\rho \in \mathrm{npos}(\xi)$, the transported leaf $\mathtt{G}(\rho)$ is in $\mathrm{yd}(\xi|_\rho)$. We use the following guides:[1]

- The *vanilla guide* maps each node position either to the leftmost leaf that is a direct successor, or (if not available) to the leftmost leaf in the yield of its right successor. The assignment is determined for each node from top to bottom.
- The *strict guide* maps each node position to the leftmost leaf in the yield of its right successor. Figure 2 shows the leafs assigned to each node in an example constituent structure.

Nonterminals. A *nonterminal constructor* computes the nonterminals for the grammar rule included in each supertag. Here, we suppose that the lhs nonterminal for the position ρ in ξ is computed from the constituent symbol $\xi(\rho)$, the set of leaves $\mathrm{yd}(\xi|_\rho)$ and the set of leaves L assigned by $\mathtt{G}$ to the ancestors of ρ as follows:

- *vanilla* – The nonterminal consists of the symbol $\xi(\rho)$, the fanout $\mathrm{fo}(\mathrm{yd}(\xi|_\rho))$, a marker if L contains any leaf in $\mathrm{yd}(\xi|_\rho)$ and, if yes, the difference in fanout $\mathrm{fo}(\mathrm{yd}(\xi|_\rho) \setminus L) - \mathrm{fo}(\mathrm{yd}(\xi|_\rho))$.

[1] As shown in Appendix A, we experimented with three more guides that did not perform as well as the ones shown here. Among them is the *modifier guide* that is defined using head relations in the constituent tree.

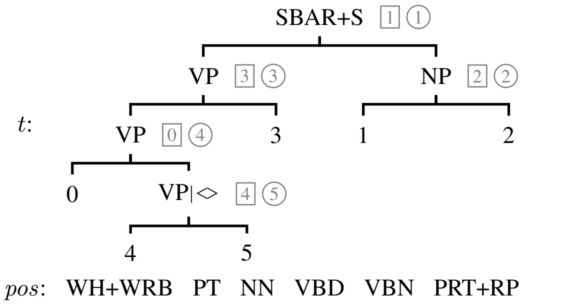

Figure 2: Constituent structure and pos symbols from Fig. 1 after binarization ($v = 1$, $h = 0$; ho and lr binarization coincide) and merging unary nodes; the symbol "VP|<>" resulted from binarization; merged symbols are joined by "+". The values assigned by the vanilla (rectangle) and strict (circle) guides are shown next to each inner node.

• *classic* – The nonterminal consists of the first symbol[2] in $\xi(\rho)$ (including the binarization markers) and the fanout $\mathrm{fo}(\mathrm{yd}(\xi|_\rho) \setminus L)$. This omits markers used to revert the extraction and is more similar to usual grammars from treebanks (Maier and Søgaard, 2008).

• *coarse* – Like the classic nonterminals, but we replace the constituent and pos symbols occurring in the treebank by their first letter. This is a rough approximation of nonterminals in coarse-to-fine parsing (Charniak et al., 2006) that does not need a specific clustering for each treebank.

3.2 Extraction algorithm.

The extraction is documented in the following three steps, we denote the chosen guide by G and the nonterminal constructor by NT. We give examples for the steps 2 and 3 that refer to (the root position in) the constituent structure shown in Fig. 2 using the vanilla guide (squares next to the nodes in the figure) and coarse nonterminals.

1. The constituent structure is binarized according to the chosen strategy. Unary nodes are merged with child nodes or, if their child is a leaf, with the pos symbol at the leaf's position. After this step, there are $|w| - 1$ inner nodes in the constituent structure; there is exactly one leaf i not in the image of G. In the following steps, we construct a supertag for each inner node in the binary constituent structure (step 2) and an additional one for the leaf i (step 3). An example is shown in Figs. 1 and 2.

2. For each node in the constituency structure, a supertag (r, t, c, p) is constructed as follows:

(r) The LCFRS rule is assembled in the usual manner from $\mathtt{G}(\rho)$ as lexical symbol (which is later replaced by the wildcard "_") and all leaves below ρ except those that are assigned by G to ρ or its ancestors. NT produces the lhs nonterminal for ρ, the rhs nonterminals are adopted from the children. In our example, r is assembled from the lexical symbol 1, the left successor's leaves are $\{\underline{0}_{(x_1)}, \underline{3,4,5}_{(x_2)}\}$ and the right one's are $\{\underline{2}_{(y_1)}\}$; hence $r = \mathrm{S}_1 \rightarrow [x_1 \,\_\, y_1 x_2](\mathrm{V}_2, \mathrm{N}_1)$.

(t) If the leaf $\mathtt{G}(\rho)$ is a direct child of ρ, then t is *None*. Otherwise, it is the index among the children where $\mathtt{G}(\rho)$ is located. In our example, 1 is not a child of the root, it is in its second successor; therefore $t = 2$.

(c) If $\xi(\rho)$ was introduced during binarization, then c is None, else it is $\xi(\rho)$. In our example, $c =$ SBAR+S.

(p) p is the pos symbol at $\mathtt{G}(\rho)$ in *pos*. In our example $p =$ PT.

3. For the leaf i, we create the supertag $(\text{L-}A \rightarrow [\_], \text{None}, \text{None}, pos(i))$ where A is the nonterminal produced by NT for the parent of i and "L-" marks this supertag for an unassigned leaf. In our example, the leaf $i = 5$ is not in the image of G, it yields the supertag $(\text{L-V|<>}_1 \rightarrow [\_], \text{None}, \text{None}, \text{PRT+RP})$.

Parsing. For parsing, a small sample of supertags is predicted for each position in the input phrase. The wildcard in the predicted tags is replaced by the input positions and the sequence of input positions is parsed using these lexical LCFRS rules in the usual manner. The resulting tree of supertags is transformed into a constituent structure by adopting the constituent symbol c from each supertag (if not None), unmerging unary nodes and transporting the associated terminal according to the index t from top to bottom.

4 Experiments

Our experiments are conducted with the usual train/dev/test splits[3] for the three discontinuous

[2] After merging unary nodes in step 1 of the extraction, each node may consist of multiple constituent symbols.

[3] We use the split for Negra by Dubey and Keller (2003), for Tiger by Seddah et al. (2013), and the standard split for DPTB (sections 2–21 for training, 22 for development, 23 for testing). In evaluation, we use the implementation for

Table 1: Our results on test sets compared to other published parsers for discontinuous constituents. Type gives a rough classification of the parsing approach in the following concepts: G – statistical grammar-based, GS – grammar-based with supertagging, C – grammarless chart-based, T – transition-based, N – untraditional neural approaches. *bert-b* and *bert-L* are language specific bert-base and bert-large models.

Type	Model	pretrained embeddings	Negra F1	Negra F1-d	Negra sent/s	Tiger F1	Tiger F1-d	Tiger sent/s	DPTB F1	DPTB F1-d	DPTB sent/s
G	van Cranenburgh et al., 2016	–	76.8	–	2	78.2	–	1	87.0	–	< 1
	Gebhardt, 2020	–	81.7	43.5	–	77.7	40.7	–	–	–	–
GS	ours	(bert-b)	91.8	74.6	120	89.7	72.6	105	94.4	82.0	81
	ours	(bert-L)	**93.9**	**79.1**	88	**91.6**	**75.4**	77	94.9	82.4	64
	Ruprecht and Mörbitz, 2021	(bert-b)	90.9	72.6	68	88.3	69.0	60	93.3	80.5	57
C	Corro, 2020	(bert-b)	91.6	66.1	–	90.0	62.1	–	94.8	68.9	–
T	Coavoux, 2021	(bert-b)	91.7	73.3	–	90.2	72.9	–	95.0	82.5	–
N	Fernández-G., Gómez-R., 2020	(bert-b)	91.0	76.6	–	90.0	62.6	–	–	–	–
	Fernández-G., Gómez-R., 2021a	(bert-b)	90.0	65.9	275	88.5	63.0	238	94.0	72.9	231
	Fernández-G., Gómez-R., 2021a	(bert-L)	92.0	67.9	216	90.5	68.1	207	95.1	74.1	193
	Fernández-G., Gómez-R., 2021b	(bert-L)	89.1	67.1	–	88.5	67.8	–	**95.5**	**82.9**	–

constituent treebanks DPTB, NeGra and Tiger. For each treebank, we select parameters for the extraction using an incomplete grid search as described in the following paragraph. Each model is trained (cf. parameters in Appendix B) to predict pos symbols separately from the other supertag components (cf. Appendix C). We use the top 10 (DPTB, Tiger) and top 15 (NeGra) predicted supertags during parsing (cf. Appendix E). For the final models, we fine-tune bert-base and bert-large (Devlin et al., 2019; Chan et al., 2020, gbert-large for German data) models with the selected final parameter configurations for 20 epochs and report the parsing scores and speed in Table 1.

Parameter selection. We conducted a parameter search in two steps to select satisfactory configurations. The first step was to exclude underperforming combinations of nonterminal constructors and guides, and the second one to select a final combination with binarization parameters. Each step is a grid search and for each configuration in the grid, we fine-tuned a bert-base model for 5 epochs using the supertags extracted from the training set and evaluated using the dev set of the treebank. In this search, we found the following configurations for our final models:

(DPTB) strict guide, classic nonterminals with lr binarization where $h = 0$ and $v = 2$,

(NeGra) strict guide, classic nonterminals with lr binarization where $h = 0$ and $v = 1$,

(Tiger) strict guide, coarse nonterminals with lr binarization where $h = 0$ and $v = 1$.

This process is documented for the NeGra treebank in Appendix D in very detail as an example.

5 Conclusion

We generalized the extraction of supertags from treebanks by introducing parameters for previously fixed parts of the construction. The parameters allow us to control the parts of the constituent tree that is associated with a terminal for each supertag (guide) as well as the granularity of the grammar rules (nonterminal constructor). At the same time, the extraction process was re-ordered so that its description is less convoluted while retaining the same functionality.

The introduced guide and nontermninal constructors performed better than the vanilla variants. Specifically, we observe the following: While the highly ambiguous grammar extracted from DPTB benefits from finer nonterminal granularity with greater markovization window, the large and more specific grammar extracted from Tiger improved with coarser granularity; the grammar for NeGra lies somewhere in between.

Compared to the previous implementation of supertagging with LCFRS, we could improve the parsing quality across all three discontinuous treebanks. The improvements close the gap between the quality of parses with LCFRS supertagging and the most recent discontinuous parsing strategies. In case of the two German treebanks, we could even surpass them, most notably in terms of the F1-score for discontinuous constituents.

F-scores by van Cranenburgh et al. (2016) with default parameters in `proper.prm`.

References

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-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on language and computation*, 2(4):597–620.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R Shrivaths, Jeremy Moore, Michael Pozar, et al. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 168–175.

Maximin Coavoux. 2021. BERT-proof syntactic structures: Investigating errors in discontinuous constituency parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3259–3272, Online. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan. Association for Computational Linguistics.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Multitask pointer network for multi-representational parsing.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021a. Discontinuous grammar as a foreign language.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021b. Reducing discontinuous to continuous parsing with pointer network reordering.

Kilian Gebhardt. 2020. Advances in using grammars with latent annotations for discontinuous parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 91–97, Online. Association for Computational Linguistics.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th conference on Formal Grammar*, pages 61–76, Hamburg, Germany. CSLI Publications.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119, Plainsboro, New Jersey. Association for Computational Linguistics.

Thomas Ruprecht and Richard Mörbitz. 2021. Supertagging-based parsing with linear context-free rewriting systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online. Association for Computational Linguistics.

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.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *Proceedings* *of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany.

Miloš Stanojević and Mark Steedman. 2020. Span-based LCFRS-2 parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 111–121, Online. Association for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

A Additional Guides

Additionally to the two guides described in Section 3, we experimented with the following:

- The *modifier guide* maps each position to its modifier's lexical head. In constituent trees, a *lexical head* is the critical phrase position for a syntactic category in a phrase, i.e. the leaf that determines the constituent symbol at a node. For each node, we call each direct child that does not contain the node's lexical head a *modifier*. This guide requires that the constituents structures are binarized head-outward. This ensures that the head of each binarized node is attached to the bottom-most introduced node; in ho binarized trees, each inner node has exactly one modifier.
- The *least transportation guide* maps as few as possible positions to leafs that are not a direct child. The guide is determined for each position from bottom to the top and selects the nearest (and leftmost, if ambiguous) leaf for each position.
- The *shortest transportation guide* maps each position in the constituent structure to a leaf that is as near as possible. The guide is determined for each positions top to bottom and, similar to the least transportation guide, selects the nearest (and leftmost, if ambiguous) leaf for each position. But, when searching for the nearest leaf, we exclude a subtree if a leaf in it was selected previously.

B Training Parameters

All models were trained using the parameter listed in the following table, during parameter search for 5 epochs, the final models for 20 epochs.

parameter	value
embeddings	last 4 layers of bert-base/bert-large
architecture	single feed forward layer
loss	cross entropy
learning rate	$5 \cdot 10^{-5}$
weight decay	10^{-2}
dropout	10^{-1}
optimizer	AdamW
batchsize	32

C Joint Prediction

In supertags as defined by Ruprecht and Mörbitz (2021), grammar rules r, constituent symbols c and indices t were all stored in the grammar rules, the pos symbols were predicted separately. Here, we investigate if there are advantages in predicting other subsets of supertag components jointly while the others are determined independently. We use the following restrictions/terminology:

- A subset of supertag components, called *core*, that always includes the grammar rule is predicted jointly as a tuple. During parsing, we use the k best *core* predictions for each input position.
- Each other component is predicted separately (via a separate feed forward layer, but the same embedding). During parsing, we only use the best prediction for each position in the input.

Table 2: Number of extracted core supertags, parsing scores (F1, F1-d), number of parse fails (↯) and prediction accuracy (acc.) for varying core supertags. Supertag components are abbreviated (grammar rule, transport index, constituent, pos).

core	no. core tags	F1	F1-d	↯	acc. core	c	t	p
g	2078	88.0	62.1	7	86.8	94.4	95.7	98.8
gt	2294	88.8	68.9	8	86.7	94.3	—	98.7
gc	2151	89.0	62.8	2	86.5	—	95.4	98.8
gp	7695	86.9	62.1	15	84.5	94.1	95.5	96.3
gtc	2368	90.6	70.7	1	86.9	—	—	98.7
gtp	8207	87.4	64.1	18	84.6	94.3	—	96.1
gcp	7784	87.6	60.3	10	84.2	—	95.7	96.4
gtcp	8295	88.7	66.4	12	84.0	—	—	96.8

To remain overview, we consider in this experiment only the set of supertags extracted with the vanilla guide, classic nonterminals and lr binarization with $v = 1$ and $h = 0$. We suggest that the results shown in Table 2 are clear enough to omit experiments with other extraction parameters or other treebanks. From the table we observe that, when pos symbols are excluded from the core,

- the number of core supertags is significantly smaller and they can be predicted more accurately,
- the quality of pos tags after parsing is significantly better, and
- there are less parse fails.

The quality of the predictions and the quality of parse trees benefits from the other components included in the core supertags. Hence, we suggest that the separation of pos tags from the other components of the core supertags is the best option and will continue our experiments with that separation.

D Parameter Selection

This section documents the selection of the nonterminal constructor, guide and binarization parameters for the NeGra treebank.

Guides and Nonterminal Constructors. In the first step, we extract supertags for each combination of nonterminal constructors and guides. Binarization is fixed to lr (except for the modifier guide which requires ho) with $h = 0$ and $v = 1$.

Table 3: Number of supertags extracted from NeGra. Rows distinguish nonterminal constructors, columns distinguish guides.

	vanilla	strict	least	shortest	modifier
vanilla	3265	2773	4677	4236	4528
classic	2367	2228	3544	3611	3587
coarse	1754	1677	2823	2837	2933

Table 3 shows the size of the extracted grammars in terms of the number of supertags. We can clearly see that both parameters determine the size of the grammar; of course that behavior was expected for the nonterminal constructors. Significantly less supertags are extracted using the strict and vanilla guides than in the three other guides.

For each combination, a classifier was trained (fine-tuned bert-base, pos symbols were predicted separately). Table 5 shows the accuracy of tag predictions and the quality of parses using the 10 best predicted supertags per phrase position. The strict guide takes a clear lead in terms of the parsing scores as well as the prediction accuracy. Looking at the parse fails, both, the vanilla and strict guides, seem to have a clear advantage over the other guides. We continue the search restricted to the strict guide and all three nonterminal constructors.

Binarization. We extract supertags using different configurations for binarization: ho and lr binarization, with horizontal markovization context $h \in \{0, 1\}$ and vertical markovization context $v \in \{0, 1, 2\}$. Table 6 shows the parsing scores for supertags extracted using all those combinations. Markovization contexts $h > 0$ and $v > 1$ do not seem to give us an advantage in this setting, it is clearly disadvantageous with vanilla and classic nonterminals. The impact of greater contexts is significantly less with coarse nonterminals. However, it does not benefit from higher values either. We select the final configuration for NeGra via the highest F1-score in the table (and the previous restrictions).

E Predictions per Position

After training the final models with bert-base, we pick a suitable value for k, i.e. the number of tags per position that is considered during parsing. The dev set is parsed with $k \in \{5, 10, 15, 25, 40\}$, Table 4 shows the results. We suggest that there is only one case where a value $k \neq 10$ shows improvements in quality that justifies the given decrease in speed, that is $k = 15$ for parsing NeGra. For both other treebanks, we continue with $k = 10$.

Table 4: Parsing scores (F1, F1-d), number of parse fails (↯) and speed (sent/s) in NeGra for varying amounts of supertags considered during parsing (k).

k	NeGra				Tiger				DPTB			
	F1	F1-d	↯	sent/s	F1	F1-d	↯	sent/s	F1	F1-d	↯	sent/s
5	90.7	73.1	10	148	92.9	76.1	110	133	93.6	85.9	34	85
10	91.1	73.8	1	125	93.1	76.9	10	130	94.7	88.0	7	61
15	91.2	74.4	0	109	93.1	76.8	0	115	94.8	88.1	3	50
25	91.3	74.4	0	72	93.1	76.5	0	83	94.9	88.3	2	52
40	91.3	74.9	0	65	93.1	76.6	0	65	94.9	88.6	0	35

Table 5: Parsing scores (F1, F1-d), number of parse fails (↯) and supertag prediction accuracy (acc.) in NeGra for combinations of nonterminal constructors (rows) and guides (columns).

	vanilla				strict				least				shortest				modifier			
	F1	F1-d	↯	acc.	F1	F1-d	↯	acc.	F1	F1-d	↯	acc.	F1	F1-d	↯	acc.	F1	F1-d	↯	acc.
vanilla	89.7	68.7	7	85.9	90.9	72.1	2	88.2	86.9	65.5	41	78.9	87.4	61.7	30	78.9	89.1	69.6	27	87.1
classic	90.5	70.5	1	84.6	91.2	71.9	2	89.0	88.8	68.7	16	82.6	87.8	58.7	3	78.8	90.4	70.6	5	87.3
coarse	89.9	69.9	0	85.7	90.8	70.3	1	88.8	88.3	66.6	9	82.5	87.7	59.1	3	79.6	90.1	70.8	9	87.8

Table 6: Parsing scores (F1, F1-d) and number of parse fails (↯) in NeGra using supertags extracted with different configurations for binarization (rows distinguish lr and ho, columns distinguish values for h and v) and nonterminal constructors (rows).

		$h = 0$									$h = 1$								
		$v = 0$			$v = 1$			$v = 2$			$v = 0$			$v = 1$			$v = 2$		
		F1	F1-d	↯	F1	F1-d	↯	F1	F1-d	↯	F1	F1-d	↯	F1	F1-d	↯	F1	F1-d	↯
vanilla	rl	90.8	72.5	2	90.9	72.1	2	89.9	70.5	24	89.3	68.1	11	87.5	68.6	35	85.0	58.3	73
	ho	84.6	65.8	0	89.8	70.3	6	88.6	66.2	26	88.2	66.0	17	86.2	63.3	52	81.1	55.4	122
classic	rl	90.9	69.9	0	91.2	71.9	2	89.9	70.7	14	89.8	69.9	6	88.5	67.7	23	86.1	65.8	65
	ho	84.2	62.2	0	90.9	71.4	1	88.5	67.1	20	89.2	69.5	9	87.6	67.7	42	83.9	63.6	89
coarse	rl	90.5	70.5	0	90.8	70.3	1	90.1	70.8	3	90.4	69.6	0	90.0	69.2	5	89.5	71.7	13
	ho	84.2	62.8	0	90.5	71.1	0	89.7	70.6	0	89.6	68.7	1	89.9	69.3	4	89.2	69.4	11