Statistical Parsing of Tree Wrapping Grammars

Tatiana Bladier Jakub Waszczuk Laura Kallmeyer
Heinrich Heine University of Düsseldorf
Universitätsstraße 1, 40225 Düsseldorf, Germany
{bladier,waszczuk,kallmeyer}@phil.hhu.de

Abstract

We describe an approach to statistical parsing with Tree-Wrapping Grammars (TWG). TWG is a tree-rewriting formalism which includes the tree-combination operations of substitution, sister-adjunction and tree-wrapping substitution. TWGs can be extracted from constituency treebanks and aim at representing long distance dependencies (LDDs) in a linguistically adequate way. We present a parsing algorithm for TWGs based on neural supertagging and A* parsing. We extract a TWG for English from the treebanks for Role and Reference Grammar and discuss first parsing results with this grammar.

1 Introduction

We present a statistical parsing approach for Tree-Wrapping Grammar (TWG) (Kallmeyer et al., 2013). TWG is a grammar formalism closely related to Tree-Adjoining Grammar (TAG) (Joshi and Schabes, 1997), which was originally developed with regard to the formalization of the typologically oriented Role and Reference Grammar (RRG) (Van Valin and LaPolla, 1997; Van Valin Jr, 2005). TWG allows for, among others, a more linguistically adequate representation of *long distance dependencies (LDDs)* in sentences, such as topicalization or long distance wh-movement. In the present paper we show a grammar extraction algorithm for TWG, propose a TWG parser, and discuss parsing results for the grammar extracted from the RRG treebanks RRGbank and RRGparbank[1] (Bladier et al., 2018).

Similarly to TAG, TWG has the elementary tree combination operations of *substitution* and *sister-adjunction*. Additionally, TWG includes the operation of *tree-wrapping substitution*, which accounts for preserving the connection between the parts of the discontinuous constituents. Operations similar to tree-wrapping substitution were proposed by (Rambow et al., 1995) as *subsertion* in D-Tree Grammars (DTG) and by (Rambow et al., 2001) as *generalized substitution* in D-Tree substitution grammar (DSG). To our best knowledge, no statistical parsing approach was proposed for DTG or DSG. An approach to symbolic parsing for TWGs with edge features was proposed in (Arps et al., 2019). In this work, we propose a statistical parsing approach for TWG and extend the pipeline based on supertagging and A* algorithm (Waszczuk, 2017; Bladier et al., 2019) originally developed for TAG to be applied to TWG.

The contributions of the paper are the following: 1) We present the first approach to statistical parsing for Tree-Wrapping Grammars. 2) We propose an extraction algorithm for TWGs based on the algorithm developed for TAG by (Xia, 1999). 3) We extend and modify the neural A* TAG-parser (Waszczuk, 2017; Kasai et al., 2018; Bladier et al., 2019) to handle the operation of tree-wrapping substitution.

2 Long distance dependencies and wrapping substitution in TWG

TWGs consist of elementary trees which can be combined using the operations a) *substitution* (replacing a leaf node with a tree), b) *sister adjunction* (adding a new daughter to an internal node) and c) *tree-wrapping substitution* (adding a tree with a d(ominance)-edge by substituting the lower part of the d-edge for a leaf node and merging the upper node of the d-edge with the root of the target tree, see Fig. 1). The latter is

[1] https://rrgbank.phil.hhu.de, https://rrgparbank.phil.hhu.de

Proceedings of the 28th International Conference on Computational Linguistics, pages 6759–6766
Barcelona, Spain (Online), December 8-13, 2020

used to capture long distance dependencies (LDDs), see the wh-movement in Fig. 1. Here, the left tree with the *d-edge* (depicted as a dashed edge) gets split; the lower part fills a substitution slot while the upper part merges with the root of the target tree. TWG is more powerful than TAG (Kallmeyer, 2016). The reason is that a) TWG allows for more than one wrapping substitution stretching across specific nodes in the derived tree and b) the two target nodes of a wrapping substitution (the substitution node and the root node) need not come from the same elementary tree, which makes wrapping non-local compared to adjunction in TAG.

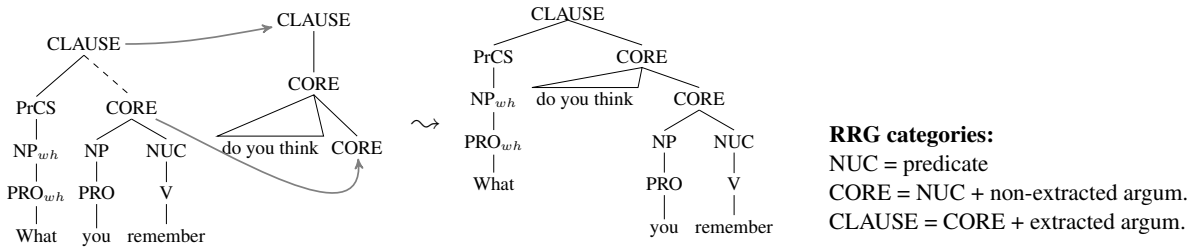

Figure 1: Tree-wrapping substitution for the sentence *"What do you think you remember"* with long-distance wh-movement.

Linguistic phenomena leading to LDD differ across languages. Among LDDs in English are some cases of extraction of a phrase to a non-canonical position with respect to its head, which is typically fronting in English (Candito and Seddah, 2012). We identified the following LDD variants in our data which can be captured with tree-wrapping substitution: long-distance relativization, long-distance wh-movement, and long-distance topicalization, which we discuss in Section 6.[2] LDD cases are rather rare in the data, which is partly due to the RRG analysis of operators such as modals, which do not embed CORE constituents (in contrast to, for example, the analyses in the Penn Treebank). Only 0,11 % of tokens in our experiment data (including punctuation) are dislocated from their canonical position in sentence to form an LDD. This number is on a par with 0,16 % of tokens reported by (Candito and Seddah, 2012) for French data.

3 Statistical Parsing with TWGs

The proposed A* TWG parser[3] is a direct extension of the simpler A* TAG parser described in (Waszczuk, 2017). The parser is specified in terms of weighted deduction rules (Shieber et al., 1995; Nederhof, 2003) and can be also seen as a weighted variant of the symbolic TWG parser (Arps et al., 2019). As in (Bladier et al., 2019), both TWG elementary trees (supertags) and dependency links are weighted, a schema also used in A* CCG parsing (Yoshikawa et al., 2017). These weights come directly from a neural supertagger and dependency parser, similar to the one proposed by (Kasai et al., 2018). Parsing consists then in finding a best-weight derivation among the derivations that can be constructed based on the deduction rules for a given sentence.

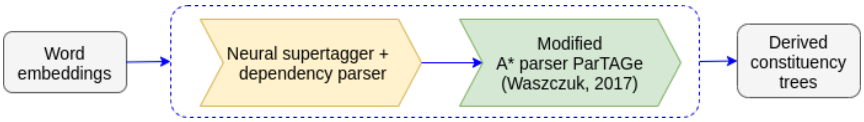

Figure 2: Pipeline of our neural statistical TWG parsing architecture.

The supertagger takes on input a sequence of word embeddings[4] $(x_i)_{i=1}^n$, to which a 2-layer BiLSTM transducer is applied to provide the contextualized word representations $(h_i)_{i=1}^n$, common to all subsequent tasks: POS tagging, TWG supertagging, and dependency parsing. On top of that, we apply two additional

[2] Another potential LDD cases in English are *it-clefts* (for example "*It was the uncertainty that Mr Lorin **feared***"). Although we have not found this LDD variant in our data, our parsing method will work for these cases as well.

[3] The parser, the TWG extraction code and the recipes to reproduce the experiments described in this paper are available at https://github.com/TaniaBladier/Statistical_TWG_Parsing.

[4] In our experiments (see Sec. 5), we used `fastText` (Bojanowski et al., 2016) to obtain the word vector representation.

2-layer BiLSTM transducers in order to obtain the supertag- and dependency-specific word representations:

$$(h_1^{(sup)}, \ldots, h_n^{(sup)}) = \texttt{BiLSTM}_s(h_1, \ldots, h_n) \quad (1)$$

$$(h_1^{(dep)}, \ldots, h_n^{(dep)}) = \texttt{BiLSTM}_d(h_1, \ldots, h_n) \quad (2)$$

The supertag-specific representations are used to predict both supertags and POS tags (POS tagging is a purely auxiliary task, since POS tags are fully determined by the supertags):

$$\Pr(sup(i)) = \texttt{softmax}(\texttt{Linear}_s(h_i^{(sup)})) \quad (3)$$

$$\Pr(pos(i)) = \texttt{softmax}(\texttt{Linear}_p(h_i^{(sup)})) \quad (4)$$

Finally, the dependency parsing component is based on biaffine scoring (Dozat and Manning, 2017), in which the head and dependent representations are obtained by applying two feed-forward networks to the dependency-specific word representations, $\texttt{hd}_i = \texttt{FF}_{hd}(h_i^{(dep)})$ and $\texttt{dp}_i = \texttt{FF}_{dp}(h_i^{(dep)})$. The score of word j becoming the head of word i is then defined as:

$$\phi(i, j) = \texttt{dp}_i^T M \ \texttt{hd}_j + b^T \texttt{hd}_j, \quad (5)$$

where M is a matrix and b is a bias vector.[5]

Extending the TAG parser to TWG involved adapting the weighted deduction rules to handle wrapping substitution as well as updating the corresponding implementation with appropriate index structures to speed up querying the chart. The A* heuristic is practically unchanged and it is both admissible (by construction) and monotonic (checked at run time), which guarantees that the first derivation found by the parser is the one with the best weight. The scheme of our parsing architecture is shown in Fig. 2. In Appendix A we provide details on modifications we have applied to the A* parser to handle the tree-wrapping substitution.

4 TWG extraction

To extract a TWG from RRGbank and RRGparbank, we adapt the top-down grammar extraction algorithm developed by (Xia, 1999) for TAG. While inital and sister-adjoining trees can be extracted following this algorithm, we added a new procedure to extract d-edge trees for wrapping substitution operation. Extraction of initial and sister-adjoining elementary trees requires manually defined percolation tables for marking head and modifier nodes. In order to extract d-edge elementary trees for LDDs, dependent constituents need to be marked prior to TWG extraction. In RRGbank and RRGparbank the constituents belonging to LDDs are indicated with features PRED-ID and NUC-ID and an index. These indicated parts alongside with the mother node are extracted to form a single tree with a *dominance link (d-edge)* (see for instance the elementary tree for "*What to say*" in Fig. 3). The remaining nodes plus the duplicated mother node and a substitution slot form the target tree, for example the tree for "*I'm trying*" in Fig. 3. Please find a more detailed formal description of our extraction algorithm along with a link to the percolation tables in Appendix B.

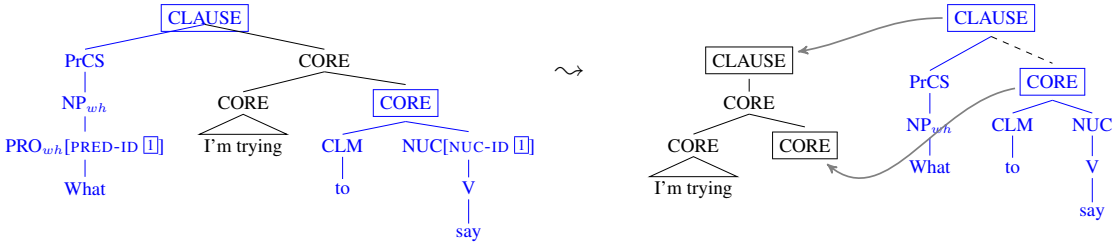

Figure 3: Extraction of a target tree and an elementary tree with a *dominance edge* (marked with dotted line). The nodes with PRED-ID and NUC-ID in the left tree identify the components of the LDD.

[5]The head representation $\texttt{hd}_0$ of the dummy root node is a parameter in this architecture.

5 Data and Parsing Experiments

We have taken the gold and silver data from RRGbank and the English part of RRGparbank[6]. The data is split in a train and a test set. We have taken 10 % of the sentences from the train set to create a dev. set. Thus, our train, dev. and test sets include 4960, 551, and 2145 trees, respectively. There are 46 constituents with LDDs in the train set, 5 in the dev. set and 27 in the test set. We extracted a TWG from this data and present in Table 1 statistics on the elementary tree templates (*supertags*) in the TWG.

Parameters	**TWG**
Supertags	3125
Supertags occuring once	1858
Avg. sentence length	10.97
Sentences	7656
# initial trees	1527
# sister-adjoining trees	1549
# d-edge trees	49

Table 1: Statistics on the extracted TWG.

We compare the parsing results with the parser DiscoDOP (van Cranenburgh and Bod, 2013) which is based on the discontinuous data-oriented parsing model. We also compare our results with the state-of-the-art transition-based parser Discoparset (Coavoux and Cohen, 2019). We evaluated[7] the overall performance of the parsers and also analyzed how well all three systems predict LDDs (see Tables 2 and 3). Unrelated to LDDs, the treebanks contain crossing branches (e.g., for operators and modifiers). Prior to TWG extraction, we decross these while keeping track of the transformation in order to be able to reverse it. For parsing with DiscoDOP and Discoparset, we added crossing branches for all LDDs. To evaluate LDD prediction with DiscoDOP and Discoparset we counted how many crossing branches were established in parsed trees. For ParTAGe we counted the LDD predictions as correct whenever the predicted supertags and dependencies indicated that the long distance element would be substituted to the elementary tree of the corresponding predicate. We counted partially correct LDDs in both parsing architectures as correctly predicted as long as the connection between the predicate and the fronted element was predicted.

	DiscoDOP		**Discoparset**		**ParTAGe**		**ParTAGe** gold POS	
	dev	test	dev	test	dev	test	dev	test
Unlabeled Attachment Score	–	–	–	–	87.74	87.67	85.13	84.64
Supertagging accuracy	–	–	–	–	74.25	75.81	77.50	77.52
POS-tagging accuracy	92.02	93.25	$94.24_{(+3.04)}$	$94.92_{(+2.69)}$	**94.63**	**95.07**	100.00	100.00
Exactly matching parses	29.04	32.87	$28.68_{(+8.89)}$	$28.30_{(+13.19)}$	**36.12**	**38.32**	38.64	38.73
Labeled F1	79.26	80.96	$83.57_{(+6.83)}$	$84.56_{(+6.39)}$	**85.26**	**85.26**	85.54	85.84
# sentences with parses	551	2145	551	2145	551	2145	546(−5)	2120(−25)

Table 2: Parsing results compared with DiscoDOP (van Cranenburgh et al., 2016) and Discoparset (Coavoux and Cohen, 2019). In case of Discoparset, the numbers in subscript represent the relative gain provided by BERT (Devlin et al., 2019) used in neither DiscoDOP nor ParTAGe experiments.

6 Error analysis for LDD prediction

We evaluated the performance of our parsing architecture with regard to the labeled F1-score and we also focused on prediction of the LDDs (see Tables 2 and 3). The results show that ParTAGe predicted the LDDs in the test data more accurately than the compared parsers. Please note that LDDs are generally rare in the corpus data and that we also had only about 5000 sentences in the training data.

Predicted LDDs	**DiscoDOP**	**Discoparset**	**ParTAGe**	
	test	test	test	test (gold POS)
# true positives	13	14	22	18
# false positives	7	0	0	0
# false negatives	14	13	5	9

Table 3: Prediction of LDDs on test data.

Some mistakes resulted from the wrong prediction of a POS tag which

[6] *Gold annotated data* means that data were annotated and approved by at least two annotators of RRGbank or RRGparbank and *silver data* means an annotation by one linguist.

[7] We use the evaluation parameters distributed together with DiscoDOP for constituency parsing evaluation. Our evaluation file is available at https://github.com/TaniaBladier/Statistical_TWG_Parsing/blob/main/experiments/eval.prm.

leads to the parser confusing an LDD constituent with a construction without LDD. For example, in (1), the word *"is"* should have POS tag V, but the parsing system erroneously labels it as AUX (= auxiliary) and thus interprets the wh-element as a predicate. In order to check our assumption about POS tags as a source of error, we have run an experiment in which we presented the parser with gold POS tags. Although this additional information helped to rule out the LDD errors in (1), restriction of the available supertags introduced new errors in LDD predictions (see Table 3) and also was the reason why some sentences could not be parsed (as shown in Table 2).

(1) a. *What is one* ***to think*** *of all this?* (*is* tagged AUX instead of V)
b. *[. . .] which he told her* ***to place*** *on her tongue* (*which* tagged CLM instead of PRO-REL)

In some cases where the relative or wh phrase of the LDD is an adjunct, as in (2), the parser incorrectly attaches it higher, taking it to be a modifier of the embedding verb.

(2) *And why do you imagine that we* ***bring*** *people to this place?*

Cases where the embedding verb also has a strong tendency to take a wh-element as argument sometimes get parsed incorrectly: In (3), *which* is analysed as an argument of *said*.

(3) *[. . .] slip of paper which they said* ***was the bill***

7 Conclusions and Outlook

We have presented a statistical parsing algorithm for parsing Tree-Wrapping Grammar - a grammar formalism inspired by TAG which aims at linguistically better representations of long distance dependencies. The LDDs in TWG are represented in a single elementary tree called *d-edge tree* which is combined with the target tree using *tree-wrapping substitution*. This operation allows to simultaneously put both parts of a discontinuous constituent to the corresponding slots of the target tree. We have extracted a TWG for English from two RRG treebanks and have compared our parsing experiments with the parser DiscoDOP based on the DOP parsing model and with the transition-based parser Discoparset. We have evaluated our parser on prediction of LDDs and could achieve more accurate results than the compared parsers.

In our future work we plan to explore TWG extraction and parsing for different languages, since the linguistic phenomena leading to LDDs vary across the languages. In particular, we have already started to work on extraction of TWGs for German and French. We plan to apply our TWG extraction and parsing algorithm to other constituency treebanks, for example French Treebank (Abeillé et al., 2003). We also plan to implement a slightly extended version of tree wrapping substitution which would allow to place the parts of discontinuous constituents in various slots between the nodes of the target tree.

Acknowledgements

We thank three anonymous reviewers for their insightful comments, as well as Rainer Osswald and Robin Möllemann for their help with collecting the experimental data and fruitful discussions. This work was carried out as a part of the research project TREEGRASP (treegrasp.phil.hhu.de) funded by a Consolidator Grant of the European Research Council (ERC).

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

David Arps, Tatiana Bladier, and Laura Kallmeyer. 2019. Chart-based RRG parsing for automatically extracted and hand-crafted RRG grammars. University at Buffalo, Role and Reference Grammar RRG Conference 2019.

Tatiana Bladier, Andreas van Cranenburgh, Kilian Evang, Laura Kallmeyer, Robin Möllemann, and Rainer Osswald. 2018. RRGbank: a Role and Reference Grammar Corpus of Syntactic Structures Extracted from the Penn Treebank. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018),*

December 13–14, 2018, Oslo University, Norway, number 155, pages 5–16. Linköping University Electronic Press.

Tatiana Bladier, Jakub Waszczuk, Laura Kallmeyer, and Jörg Janke. 2019. From partial neural graph-based LTAG parsing towards full parsing. *Computational Linguistics in the Netherlands Journal*, 9:3–26, Dec.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Marie Candito and Djamé Seddah. 2012. Effectively long-distance dependencies in French: Annotation and parsing evaluation.

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, June. 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, June. Association for Computational Linguistics.

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

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Laura Kallmeyer, Rainer Osswald, and Robert D. Van Valin, Jr. 2013. Tree Wrapping for Role and Reference Grammar. In G. Morrill and M.-J. Nederhof, editors, *Formal Grammar 2012/2013*, volume 8036 of *LNCS*, pages 175–190. Springer.

Laura Kallmeyer. 2016. On the mild context-sensitivity of k-Tree Wrapping Grammar. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar: 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers. FG 2016, Italy, August 2016, Proceedings*, number 9804 in Lecture Notes in Computer Science, pages 77–93, Berlin. Springer.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1181–1194, New Orleans, Louisiana, June. Association for Computational Linguistics.

Mark-Jan Nederhof. 2003. Squibs and Discussions: Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics*, 29(1).

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of ACL*.

Owen Rambow, K Vijay-Shanker, and David Weir. 2001. D-tree substitution grammars. *Computational Linguistics*, 27(1):87–121.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate dop model. In *Proceedings of the International Conference on Parsing Technologies (IWPT 2013)*. Citeseer.

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.

Robert D. Van Valin, Jr. and Randy LaPolla. 1997. *Syntax: Structure, meaning and function*. Cambridge University Press.

Robert D. Van Valin Jr. 2005. *Exploring the syntax-semantics interface*. Cambridge University Press.

Jakub Waszczuk. 2017. *Leveraging MWEs in practical TAG parsing: towards the best of the two worlds*. Ph.D. thesis.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. *CoRR*, abs/1704.06936.

Appendix A. Specification of the TWG parser

AX:	$\frac{}{(0,\emptyset) : (N \rightarrow \bullet\alpha, i, i, [])}$	$i \in \{0,\ldots,n-1\}$ $N \rightarrow \alpha$ is a rule
SC:	$\frac{(w,m) : (N \rightarrow \alpha \bullet M \beta, i, j, \Gamma)}{(w,m) : (N \rightarrow \alpha M \bullet \beta, i, j{+}1, \Gamma)}$	$\ell(M) = s_{j+1}$
DE:	$\frac{(w,m) : (N \rightarrow \alpha \bullet, i, j, \Gamma)}{(w,m) : (N, i, j, \Gamma, ws?)}$	$ws?{=}yes \iff dnode(N)$
CS:	$\frac{(w_1,m_1) : (N \rightarrow \alpha \bullet M \beta, i, j, \Gamma_1) \quad (w_2,m_2) : (M, j, k, \Gamma_2, no)}{(w_1{+}w_2, m_1 \oplus m_2) : (N \rightarrow \alpha M \bullet \beta, i, k, \Gamma_1 \oplus \Gamma_2)}$	
SU:	$\frac{(w_1,m_1) : (N \rightarrow \alpha \bullet M \beta, i, j, \Gamma_1) \quad (w_2,m_2) : (R, j, k, \Gamma_2, no)}{(w_1{+}w_2{+}\omega(R,N), m_1 \oplus m_2) : (N \rightarrow \alpha M \bullet \beta, i, k, \Gamma_1 \oplus \Gamma_2)}$	$leaf(M)$ $root(R) \wedge \neg sister(R)$ $\ell(M){=}\ell(R)$
SA:	$\frac{(w_1,m_1) : (N \rightarrow \alpha \bullet \beta, i, j, \Gamma_1) \quad (w_2,m_2) : (M, j, k, \Gamma_2, no)}{(w_1{+}w_2{+}\omega(M,N), m_1 \oplus m_2) : (N \rightarrow \alpha \bullet \beta, i, k, \Gamma_1 \oplus \Gamma_2)}$	$\ell(M){=}\ell(N) \wedge sister(M)$ $\neg sister(N)$
PW:	$\frac{(w_1,m_1) : (N \rightarrow \alpha \bullet M \beta, i, j, \Gamma_1) \quad (w_2,m_2) : (D, j, k, \Gamma_2, yes)}{(w_1, m_1[j \Rightarrow w_2{+}sum(m_2){+}A(D)]) : (N \rightarrow \alpha M \bullet \beta, i, k, \Gamma_1 \oplus [(j,k,\ell(D))])}$	$leaf(M)$ $\ell(M){=}\ell(D)$
CW:	$\frac{(w_1,m_1) : (R, i, j, \Gamma_1 \oplus [(f_1,f_2,y)] \oplus \Gamma_2, ws?) \quad (w_2,m_2) : (D, f_1, f_2, \Gamma_3, yes)}{(w_1{+}w_2{+}\omega(R,D), m_1[f_1 \Rightarrow \bot] \oplus m_2) : (D, i, j, \Gamma_1 \oplus \Gamma_3 \oplus \Gamma_2, no)}$	$root(R) \wedge y{=}\ell(D)$ $\ell(parent(D)){=}\ell(R)$ $\neg sister(R)$

Table 4: Weighted deduction rules of the TWG parser

Our TWG parser is specified in terms of weighted deduction rules (Shieber et al., 1995; Nederhof, 2003). Each deduction rule (see Table 4) takes the form of a set of antecedent items, presented above the horizontal line, from which the consequent item (below the horizontal line) can be deduced, provided that the corresponding conditions (on the right) are satisfied. The specification of the TWG parser consists of 8 deduction rules which constitute a blend of the TAG parser (Bladier et al., 2019) with the symbolic TWG parser (Arps et al., 2019). Here, we assume familiarity with both these parsers and limit ourselves to explaining the features specific to the statistical TWG parser.

Weights. A pair (w, m) is assigned to each chart item via deduction rules, where w is the inside weight, i.e., the weight of the inside derivation, and m is a map assigning weights to the individual gaps in the corresponding gap list Γ. Since each gap in Γ can be uniquely identified by its starting position, we use the starting positions as keys in m. The need to use a map (dictionary) data structure instead of a single scalar value, as in the TAG parser, stems from the `CW` rule (*complete wrapping*), in which the calculation of the resulting weight map requires removing the weight corresponding to the gap (f_1, f_2, y).

We use $\emptyset$ to denote an empty map, $m[x \Rightarrow y]$ to denote m with y assigned to x, $m[x \Rightarrow \bot]$ to denote m with x removed from the set of keys (together with the corresponding value), and $sum(m)$ to denote the sum of values (weights) in the map m. We also re-use the concatenation operator $\oplus$ to represent map union. Whenever map union is used ($m_1 \oplus m_2$), the sets of keys of the two map arguments (m_1 and m_2) are guaranteed to be disjoint (an invariant which can be proved by induction over the deduction rules).

Heuristic. Given a chart item $\eta = (x, i, j, \Gamma)$ with the corresponding weights (w, m), the TWG A* heuristic (which provides a lower-bound estimate on the cost of parsing the remaining part of the sentence) is a straightforward generalization of the TAG A* heuristic used by (Bladier et al., 2019). In particular, it accounts for the total minimal cost of scanning each word outside the span (i, j), as well as the words remaining in the gaps in Γ. Thus, in constrast with the TAG heuristic, since there can be many gaps in Γ, the sum of the weights in the map m has to be accounted for.

Appendix B. TWG extraction algorithm

1. **Decross tree branches.** First, for local discontinuous constituents (for instance NUCs consisting of a verb and a particle in German), we split the constituent into two components (e.g., NUC1 and NUC2), both attached to the mother of the original discontinuous node.

   Second, if a tree τ still has crossing branches, the tree is traversed top-down from left to right and among its subtrees those trees are identified whose root labels contain one of the following strings: OP-, -PERI, -TNS, CDP, or VOC. For each such subtree γ in question with r being its root, we choose the highest node v below the next left[8] sibling of r such that the rightmost leaf dominated by v immediately precedes the leftmost leaf dominated by r. If r and v are not yet siblings, γ is reattached to the parent of v. If the subtree in question has no left siblings, it is reattached to the right in a corresponding way. After this step, it should be checked if the tree τ still contains crossing branches. If yes, the process of decrossing branches is continued by applying the steps above to the next subtree in question.

2. **Extract LDDs.** Then we traverse each tree τ in a top-down left-to-right fashion and check for each subtree of τ whether it contains the following special markings for LDDs in its root label: PREDID=, NUCID= or REF=. The indexes identify the parts of the NLD which belong together. In case of an LDD, the parts of the minimal subtree which contain both parts of the LDD are extracted within a single tree with a *d-edge* (see the multicomponent NUC and CORE in Figure 3). The substitution site and the mother node are added to the remaining subtree in order to mark the nodes on which the wrapping substitution takes place (see Figure 3).

   After this step, an empty agenda is created and the extracted tree chunks and the pruned tree τ with the remaining nodes are placed into the agenda.

3. **Extract initial and sister-adjoining trees.** If no agenda with tree chunks was created in the previous step, an empty agenda is created in this step and the entire tree τ is placed into it. Each tree chunk in the agenda is traversed and the percolation tables[9] are used to decide for each subtree $\tau_1 \ldots \tau_n$ in the tree chunk whether it is a head, a complement or a modifier with respect to its parent. Initial trees for identified complements and sister-adjoining trees for identified modifiers are extracted recursively in the top-down fashion until each elementary tree has exactly one anchor site.

   Initial trees are extracted as follows: If a node of a subtree is identified as a complement, it is removed from the parent tree and the parent node is marked as a substitution slot. In order to extract sister-adjoining trees for identified modifier subtrees, the parent node of the subtree is copied and added as the new root node of the elementary tree with a special marking * on the root label.

[8] A node v_1 is left to another node v_2 if the leftmost leaf dominated by v_1 is left of the leftmost leaf dominated by v_2.

[9] Please find the code for our TWG extraction algorithm along with the percolation tables for head and modifier distinction in this repository: https://github.com/TaniaBladier/Statistical_TWG_Parsing.