Parsing Beyond Context-Free Grammars: Discontinuous Data-Oriented Parsing

Laura Kallmeyer & Tatiana Bladier
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2018

Overview

Discontinuous Data-oriented Parsing: Idea

- Discontinuous Data-Oriented Parsing:
  - ⇒ Statistical parsing approach.
  - ⇒ A synthesis of discontinuous grammars and Data-Oriented Parsing (DOP) [vC$^+$16].
  - ⇒ Based on the PLCFRS parsing algorithm described in [KM13].
- Data-Oriented Parsing:
  - ⇒ conceptually simple and general account of the interpretation of new utterances given a corpus of previous utterances [Sch90].
- Instead of LCFRSs, a formalism of *Discontinuous Tree-Substitution Grammar* (DTSG) is used.

Data-oriented Parsing: Principles [Sch90, vC+16]

- Language users process sentences based on fragments from previous language experience.
  - $\Rightarrow$ DOP considers statistical properties of actual language use
- Grammar is implicit in the treebank itself.
  - $\Rightarrow$ The treebank **is** the grammar
  - $\Rightarrow$ probabilities are derived from the frequencies of all connected tree fragments in the treebank.

"*. . . in analysing new input [the system] tries to find the most probable way to reconstruct this input from fragments that are already contained in the corpus.*"

Data-oriented Parsing: Principles [Sch90]

- All possible fragments from the treebank trees can be used to derive new sentences [VCSS11].
- A memory bias: "[T]he number of constructions that is used to re-construct the sentence in order to recognize it must be as small as possible."
- A probabilistic bias: "More frequent constructions are to be preferred above less frequent ones." [vC+16]

Data-oriented Parsing: Principles [Sch90]

- DOP trees $\Rightarrow$ fragments (+frequencies)
- fragments $\Rightarrow$ arbitrarily sized chunks from the corpus

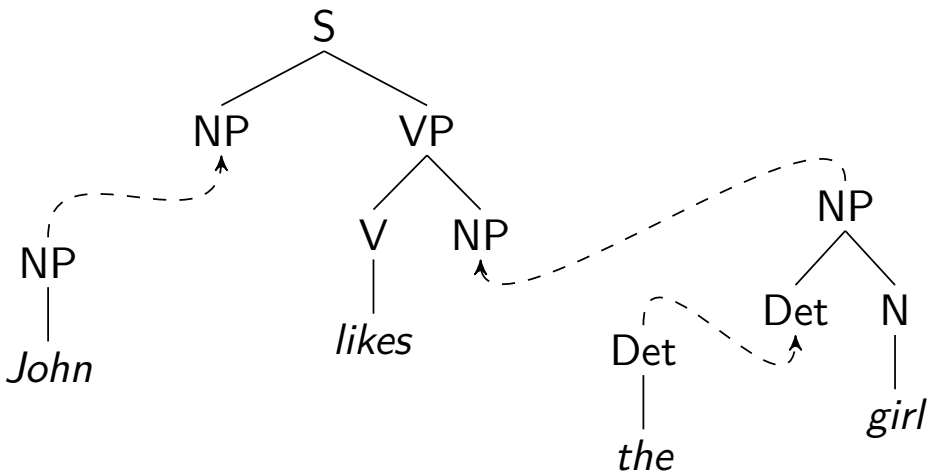
- instead of manually writing a grammar or refining probabilities

  $\Rightarrow$ consider all possible fragments from treebank
  and "let the statistics decide" [vC$^+$16]

- The DOP model for discontinuous parsing is based on **Discontinuous Tree-Substitution Grammar (DTSG).**

Probabilistic Tree-Substitution Grammar (1)

- A Tree Substitution Grammar (TSG) is a special case of TAG
  - ⇒ TAG without adjunction
- TSG has the same context-free property as CFG, but:
  - ⇒ non-terminals can rewrite as entire tree fragments, not just immediate children;
  - ⇒ TSG captures more structural relations than (P)CFG.
- Weakly equivalent to CFG;



$S \rightarrow NP\ (VP\ (V\ likes)\ NP)$
$NP \rightarrow Det\ (N\ girl)$
$NP \rightarrow John$
$Det \rightarrow the$

Probabilistic Tree-Substitution Grammar (2)

Definition 1 (Tree Substitution Grammar)

A Tree Substitution Grammar (TSG) is a tuple $G = \langle N, T, S, P \rangle$ such that

- T and N are disjoint alphabets, the terminals and nonterminals,
- $S \in N$ is the start symbol,
- P the productions take the form of elementary trees – tree fragments of depth ≥ 2,
- internal nodes are labelled with non-terminals and each leaf is labelled with either a *terminal* or a *non-terminal*,

  $\Rightarrow$ non-terminal leaves are called *frontier non-terminals*.
- A *derivation* creates a tree by starting with the root symbol and rewriting (substituting) it with an elementary tree, then continuing to rewrite frontier non-terminals with elementary trees until there are no remaining frontier non-terminals [CGB09].

Probabilistic Tree-Substitution Grammar (3)

- A Probabilistic Tree Substitution Grammar (PTSG), like a PCFG, assigns a probability to each rule in the grammar.
- The probability of a derivation is defined as the product of the probabilities of its component elementary trees:

$$p(\mathbf{e}) = \prod_{x \rightarrow e \in \mathbf{e}} p(e|x)$$

- where
  - $e = (e1, e2, \dots)$ is a sequence of elementary trees used for the derivation,
  - $x = root(e)$ is the root symbol of e,
  - $p(e|x)$ is the probability of generating e given its root symbol x.
- As in a PCFG, e is generated conditionally independent of all others given x.

Discontinuous Tree-Substitution Grammar (1)

- A *Discontinuous Tree Substitution Grammar* (TSG) provides a generalization of CFG that operates with larger chunks than just single grammar productions.
- A DTSG is a tuple $\langle N, T, V, S, \vartheta \rangle$ where N and T are disjoint sets of non-terminal and terminal symbols, V is a finite set of variables, S denotes the start non-terminal, and ϑ is a finite set of elementary trees.

  $\rightarrow$ For all trees in ϑ it holds that for each non-terminal, there is a unique fan-out.
- A probabilistic DTSG (PDTSG) is a tuple $\langle N, T, V, S, \vartheta, P \rangle$, where
  - $\rightarrow$ P is the function which assigns each elementary tree t a probability value $0 < P(t) \leq 1$ such that
  - $\rightarrow$ for every non-terminal $A \in N$, the probabilities of all elementary trees whose root node is labeled A sum to 1.

[vC+16]

Discontinuous Tree-Substitution Grammar (2)

Definition 2 (A discontinuous tree)

- a rooted, unordered tree;
- each node consists of a label and a yield, where a yield is a tuple of strings composed of lexical items;
- Given a node X,
  - the yield of X is composed of the terminals in the yields of the children of X;
  - the yield of each child of X is a subsequence of the yield of X;
  - the yields of siblings do not overlap.

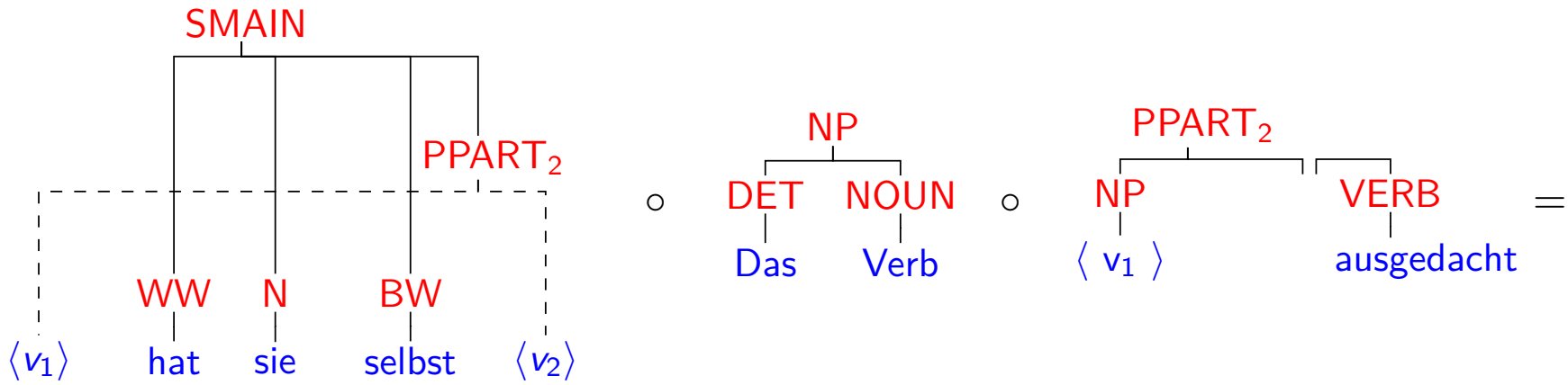
[vCB13]

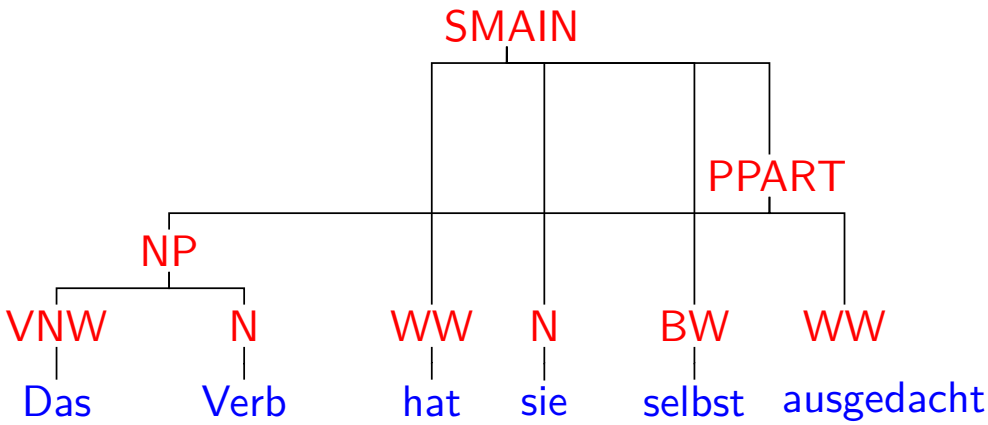
Discontinuous Tree-Substitution Grammar (3)

- Elementary trees in DTSG are combined via **left-most substitution**.
- The left-most substitution $A \circ B$ is defined iff the label of the left-most substitution site of A equals the label of the root node of B.
- The result of $A \circ B$ equals a copy of the tree A with B substituted for the left-most substitution site of A.
- In the yield argument of A, each variable terminal is replaced with the corresponding component of one or more contiguous terminals from B.
  - $\Rightarrow$ For example, given $yield(A) = \langle l_1 v_2 l_4 \rangle$, and $yield(B) = \langle l_2 l_3 \rangle$ where l_n is a lexical terminal and v_n a variable, $yield(A \circ B) = \langle l_1 l_2 l_3 l_4 \rangle$

[vC+16]

Discontinuous Tree-Substitution Grammar (4)



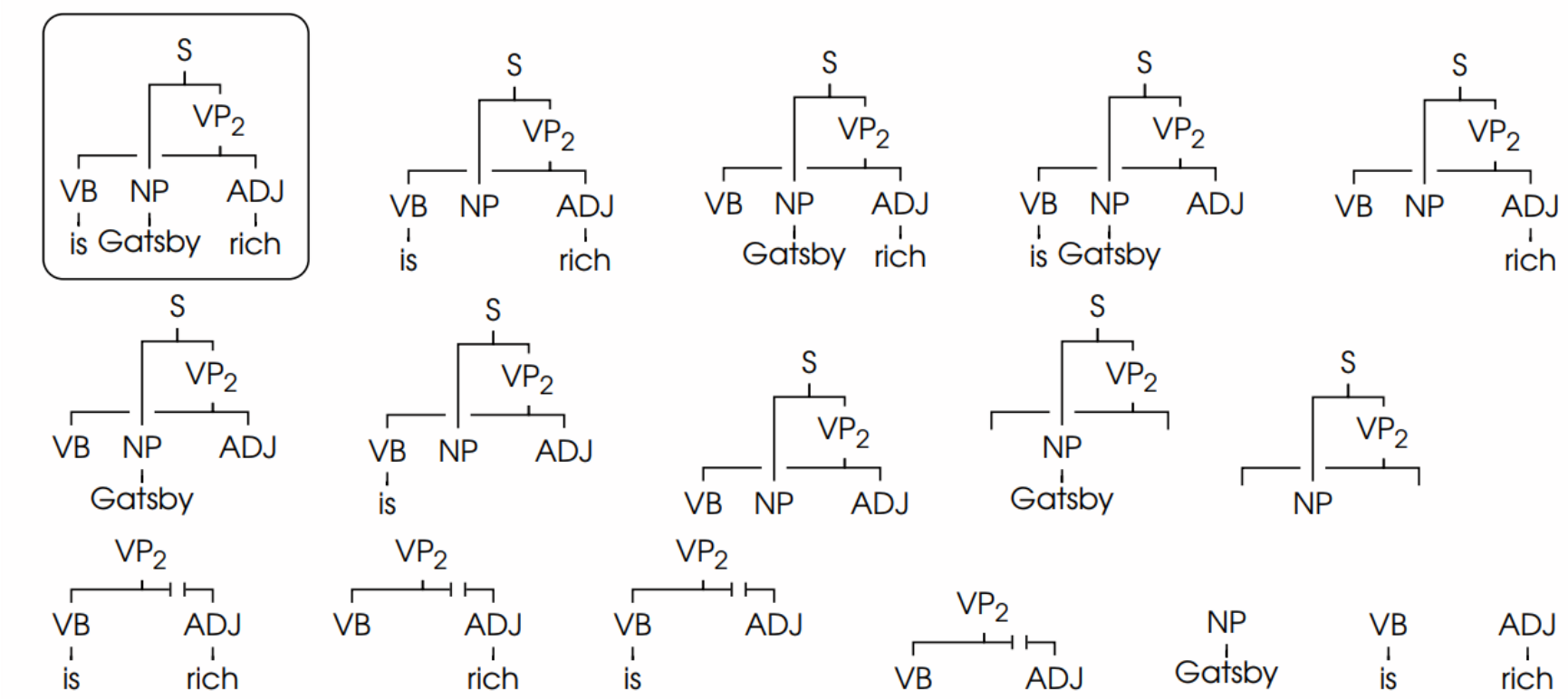


[vC+16]

Inducing DTSG from the treebank

The weight of a fragment is its relative frequency in the training data:

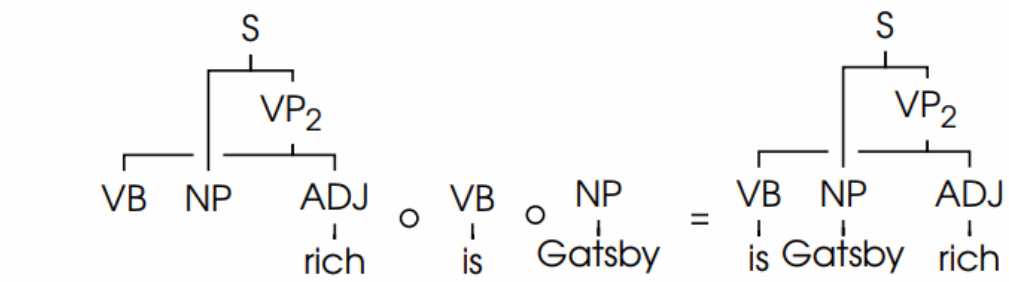
$$\mathbf{P(f)} = \frac{count(f)}{\sum_{f' \in F} count(f')} \quad \textbf{where } F = \{f' | root(f') = root(f)\}$$



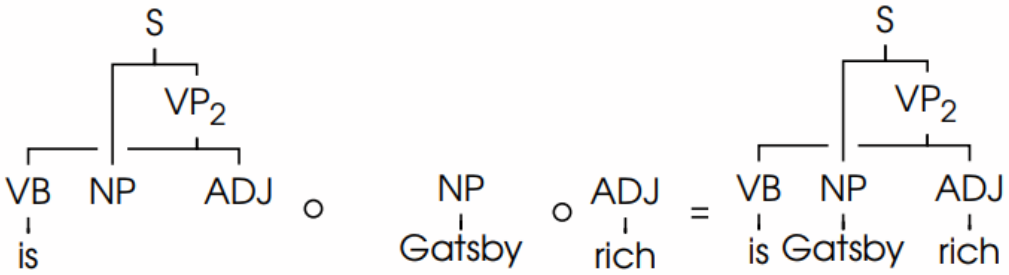
Source: http://andreasvc.github.io/gothenburgtalk.pdf

DOP1 fragments and derivation

- A derivation is defined as a sequence of fragments combined through the left-most substitution.
- *most probable derivation P(d)* = maximized probability of an individual derivation (one sequence of fragments leading to a complete analysis)
- *most probable parse P(p)* = maximized sum of probabilities leading to the same derivation (e.g. P(p) = 0.2 + 0.3 = 0.5)



$P(d) = 0.2$

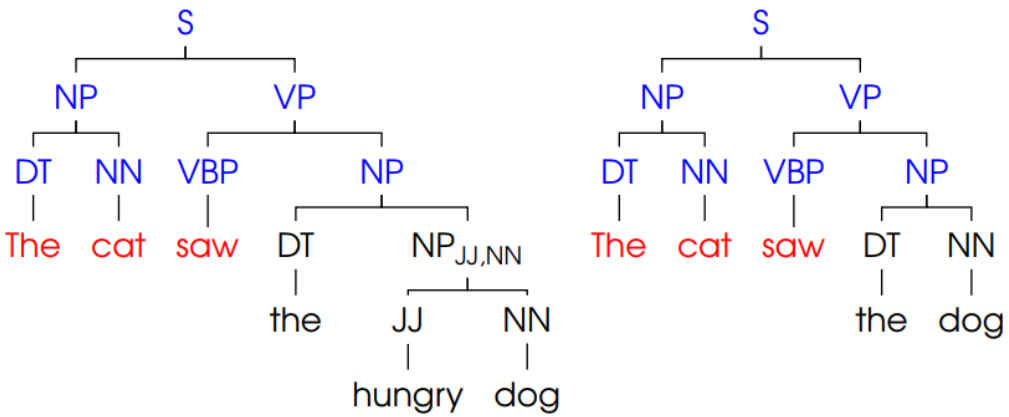


$P(d) = 0.3$

Source: http://andreasvc.github.io/gothenburgtalk.pdf

Double-DOP: Extraction of recurrent fragments

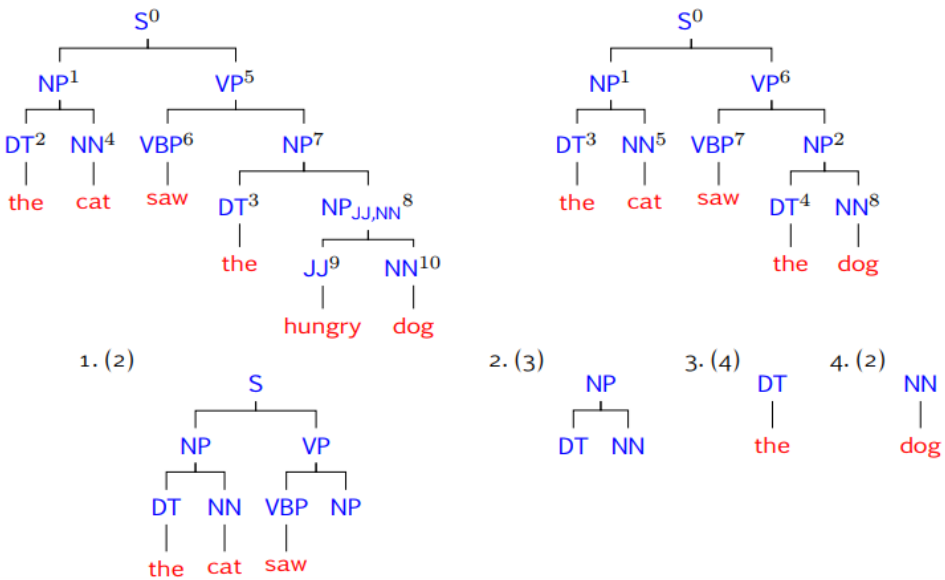
- DOP1 works with all possible fragments of a treebank.
- representing all possible fragments of a treebank is not feasible:
  - the number of fragments is exponential in terms of the number of nodes.
    $\Rightarrow$ a subset of fragments should be defined.
- Double DOP (DOP2) restricts the fragment set to recurring fragments [SZ11]
  - fragments that occur in at least two different contexts;
  - for every pair of trees, extract maximal overlapping fragments;
  - the number of fragments is small enough to parse directly;
  - fragments are extracted using the tree kernel method [VC12].



Source: http://andreasvc.github.io/gothenburgtalk.pdf

Double-DOP: Extraction of recurrent fragments (2)

- The goal is to extract maximal common fragments of the trees and to count their frequencies.
- Algorithm for extraction of tree fragments: **Fast Tree Kernel** [MPB08];
  - $\Rightarrow$ the problem of finding common productions = finding the intersection of two multisets that have been sorted in advance [vC+16].
- The input of the function fast-tree-kernel is a pair of trees $\langle a, b \rangle$, represented as a list of nodes sorted by their productions.
- The output is a boolean matrix where the bit at (n, m) is set iff the nodes at those indices in the respective trees have the same production.



	S0	NP1	NP2	DT3	DT4	NN5	VP6	VBP7	NN8
S0	1								
NP1		1	1,2						
DT2				1,3	3				
DT3				3	3				
NN 4						4			
VP 5							1		
VBP6								1	
NP 7									
NP8JJ,NN									
JJ 9									
NN 10									1

Grammar transformations [vCSB16] (1)

- CFG, LCFRS, and DTSG can be seen as natural extensions of each other.

  $\Rightarrow$ transformations from one grammar to another can be defined that help to make parsing more efficient;

  $\Rightarrow$ productions and labels in transformed grammars map back to the original grammar.

Grammar transformations [vCSB16] (2)

- LCFRS to (Split)-CFG:
  - $\Rightarrow$ discontinuous constituents in LCFRS are split into several non-terminals,
  - $\Rightarrow$ each new non-terminal covers a single contiguous component of the yield of the discontinuous constituent.
  - $\Rightarrow$ Given a discontinuous non-terminal X_n in the original treebank, the new non-terminals will be labelled X_n^{*m}, with m the index of the component, so that $1 \leq m \leq n$.

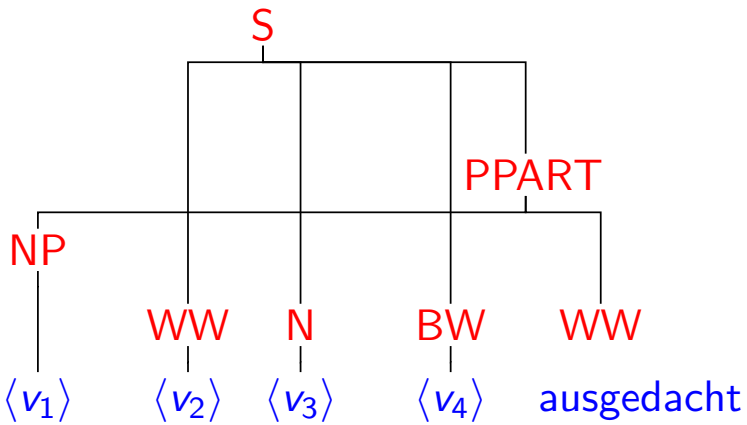
LCFRS productions	**CFG approximation**
$S(abc) \rightarrow NP(b)VP_2(a,c)$	$S \rightarrow VP_2^{*1}\ NP\ VP_2^{*2}$
$VP_2(a,b) \rightarrow VB(a)PRT(b)$	$VP_2^{*1} \rightarrow VB$
	$VP_2^{*2} \rightarrow PRT$

Grammar transformations [vCSB16] (3)

- DTSG to CFG:

  $\Rightarrow$ Due to the design of the Disco-DOP parser, it is desirable to have grammar productions in binarized form, and to separate phrasal and lexical productions.

DTSG tree



CFG approximation

$$S(ab) \rightarrow S^1(a)WW(b)$$
$$S^1(ab) \rightarrow S^2(a)BW(b)$$
$$S^2(ab) \rightarrow S^3(a)N(b)$$
$$S^3(ab) \rightarrow NP(a)WW^2(b)$$
$$WW^2(ausgedacht) \rightarrow \epsilon$$

Disco-DOP parser

- https://github.com/andreasvc/disco-dop
- Agenda-based parser for PLCFRS based on the algorithm described in [KM13] (we discussed this parsing algorithm last week).
- Coarse-to-fine pruning:
  - a technique to speed up parsing by exploiting the information that can be gained from parsing with simpler, coarser grammars;
    $\Rightarrow$ grammar pruning with some probabilistic threshold;
  - constituents that do not contribute to a full parse tree are ruled out, which greatly reduces the number of edges that need to be explored

Disco-DOP parser: coarse-to-fine pipeline

- Coarse-to-fine pruning is done in order to tame the complexity of LCFRS and DOP.
- The parsing is done in three stages [vCB13]:
  - Split-PCFG: A PCFG approximation of the discontinuous treebank grammar; rewrites spans of discontinuous constituents independently
  - PLCFRS: The discontinuous treebank grammar; rewrites discontinuous constituents in a single operation
  - The discontinuous DOP grammar (DTSG): tree fragments instead of individual productions from treebank
- Pruning is achieved by limiting the second and third stages to the labeled spans occurring in the k-best derivations of the previous stage.
- The initial values for k are 10,000 and 50, for the PLCFRS and DOP grammar respectively.

Disco-DOP parser: Evaluation

Parser	F1	EX	func
GERMAN: Tiger			
Dep: HaNi2008	75.3	32.6	
2DOP: Cr et al	**78.2**	**40.0**	93.5
Dep: FeMa2015	82.6	45.9	
ENGLISH: wsj			
PLCFRS: EvKa2011	79.0		
2DOP: Cr et al, wsj	**87.0**	34.4	86.3
2DOP: SaZu2011, no disc.	87.9	33.7	
DUTCH: Lassy			
2DOP: Cr et al	76.6	34.0	92.8

HaNi: Hall & Nivre (2008); FeMa: Fernández-González & Martins (2015);
SaZu: Sangati & Zuidema (2011); EvKa: Evang & Kallmeyer (2011);
Cr et al: van Cranenburgh, Scha, Bod (submitted).

Source: http://andreasvc.github.io/gothenburgtalk.pdf

References I

[CGB09] Trevor Cohn, Sharon Goldwater, and Phil Blunsom.
Inducing compact but accurate tree-substitution grammars.
In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 548–556, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[KM13] Laura Kallmeyer and Wolfgang Maier.
Data-driven parsing using probabilistic linear context-free rewriting systems.
Computational Linguistics, 39(1):87–119, 2013.

[MPB08] Alessandro Moschitti, Daniele Pighin, and Roberto Basili.
Tree kernels for semantic role labeling.
Computational Linguistics, 34(2):193–224, 2008.

[Sch90] R Scha.
Language theory and language technology; competence and performance (in dutch). in de kort, q. and leerdam, g., editors.
Computertoepassingen in de Neerlandistiek, 1990.

[SZ11] Federico Sangati and Willem Zuidema.
Accurate parsing with compact tree-substitution grammars: Double-dop.
In *Proceedings of the conference on empirical methods in natural language processing*, pages 84–95. Association for Computational Linguistics, 2011.

[VC12] Andreas Van Cranenburgh.
Extracting tree fragments in linear average time.
Technical report, Technical Report PP-2012-18, FNWI/FGw: Institute for Logic, Language and Computation (ILLC). http://dare. uva. nl/en/record/421534, 2012.

References II

[vC+16] AW van Cranenburgh et al.
Rich statistical parsing and literary language.
2016.

[vCB13] Andreas van Cranenburgh and Rens Bod.
Discontinuous parsing with an efficient and accurate dop model.
In *Proceedings of the International Conference on Parsing Technologies (IWPT 2013)*. Citeseer, 2013.

[vCSB16] Andreas van Cranenburgh, Remko Scha, and Rens Bod.
Data-oriented parsing with discontinuous constituents and function tags.
Journal of Language Modelling, 4(1):57–111, 2016.

[VCSS11] Andreas Van Cranenburgh, Remko Scha, and Federico Sangati.
Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar.
In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 34–44. Association for Computational Linguistics, 2011.