Discontinuous Data-Oriented Parsing
using Coarse-to-Fine methods

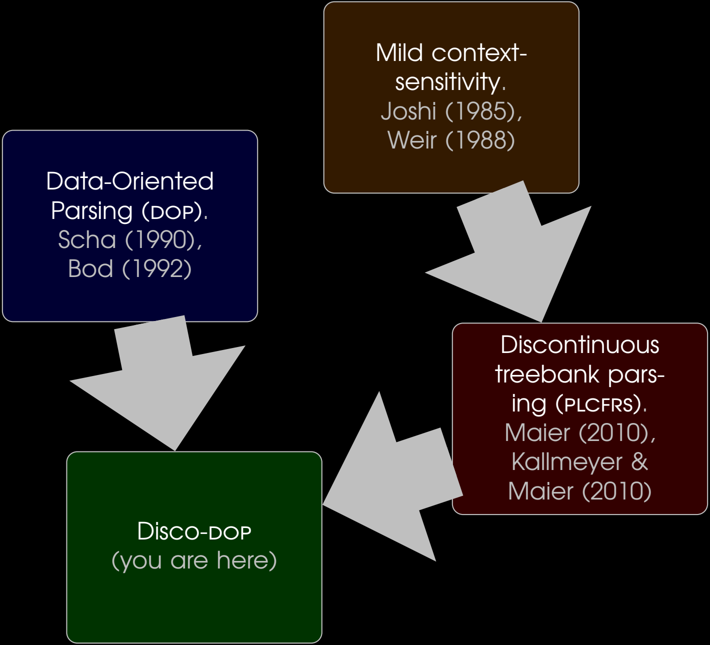
Andreas van Cranenburgh

Huygens ING
Royal Netherlands Academy of Arts and Sciences

Institute for Logic, Language and Computation
University of Amsterdam

November 29, 2012

Overview


Mild context-sensitivity.
Joshi (1985), Weir (1988)
Data-Oriented Parsing (DOP).
Scha (1990), Bod (1992)
Discontinuous treebank parsing (PLCFRS).
Maier (2010), Kallmeyer & Maier (2010)
Disco-DOP
(you are here)

This talk

1. Discontinuity
2. Coarse-to-fine
3. Data-Oriented Parsing

Discontinuity

Discontinuity

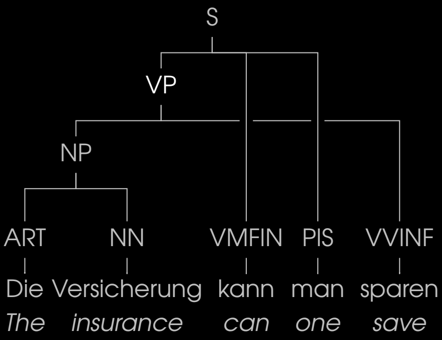


Figure: A discontinuous tree from the Negra corpus.
Translation: *As for the insurance, one can save it.*

Discontinuity

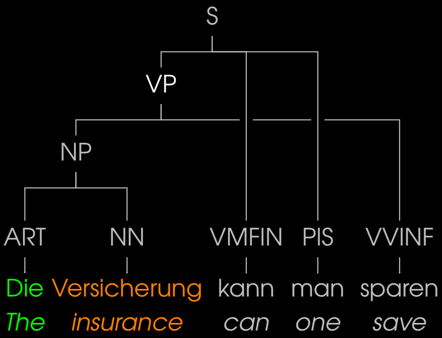


Figure: A discontinuous tree from the Negra corpus.
Translation: *As for the insurance, one can save it.*

Context-Free Grammar (CFG)

$$NP(ab) \rightarrow DT(a)\ NN(b)$$

Discontinuity

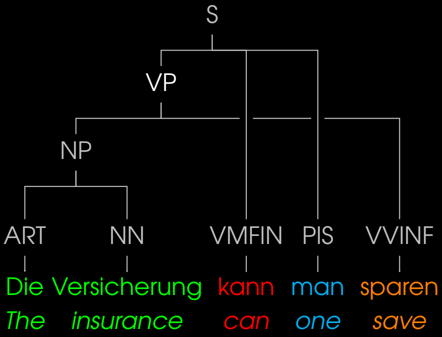


Figure: A discontinuous tree from the Negra corpus.
Translation: *As for the insurance, one can save it.*

Linear Context-Free Rewriting System (LCFRS)

$$S(abcd) \rightarrow VP_2(a, d)\ VMFIN(b)\ PIS(c)$$
$$VP_2(a, b) \rightarrow NP(a)\ VVINF(b)$$

Linear Context-Free Rewriting Systems

LCFRS are a generalization of CFG:
$\Rightarrow$ rewrite tuples, trees or graphs!

Vijay-Shanker, Weir, Joshi (1987): Structural descriptions produced by various grammar formalisms

Linear Context-Free Rewriting Systems

LCFRS are a generalization of CFG:
⇒ rewrite tuples, trees or graphs!

linear each variable on the left occurs once on the right & vice versa

context-free apply productions based on what they rewrite

rewriting system i.e., grammar

Vijay-Shanker, Weir, Joshi (1987): Structural descriptions produced by various grammar formalisms

Linear Context-Free Rewriting Systems

LCFRS are a generalization of CFG:
⇒ rewrite tuples, trees or graphs!

linear each variable on the left occurs once on the right & vice versa

context-free apply productions based on what they rewrite

rewriting system i.e., grammar

Rules can be read off from treebank,
relative frequencies give probabilistic LCFRS (PLCFRS)

Vijay-Shanker, Weir, Joshi (1987): Structural descriptions produced by various grammar formalisms

Linear Context-Free Rewriting Systems

- Can be parsed with tabular parsing algorithm
- Parsing a binarized LCFRS has complexity

$$\mathcal{O}(|w|^{3\varphi})$$

where φ is the maximum number of components covered by a non-terminal (fan-out).

Kallmeyer & Maier (2010). Data-driven parsing with probabilistic linear context-free rewriting systems.

Linear Context-Free Rewriting Systems

- Can be parsed with tabular parsing algorithm
- Parsing a binarized LCFRS has complexity

  $$\mathcal{O}(|w|^{3\varphi})$$

  where φ is the maximum number of components covered by a non-terminal (fan-out).
- Agenda-based probalistic parser for LCFRS (Kallmeyer & Maier 2010)
- Our parser builds an exhaustive chart, because we need the k-best derivations

Kallmeyer & Maier (2010). Data-driven parsing with probabilistic linear context-free rewriting systems.

Punctuation

Problem: Punctuation in Negra causes spurious discontinuity.

van Cranenburgh (2012), Efficient parsing with linear context-free rewriting systems. EACL.

Punctuation

Problem: Punctuation in Negra causes spurious discontinuity.
Solution:

1. Attach punctuation to highest constituent with neighbor on its right
2. Parentheses & quotation marks as low as possible around same constituent

van Cranenburgh (2012), Efficient parsing with linear context-free rewriting systems. EACL.

Punctuation

Problem: Punctuation in Negra causes spurious discontinuity.

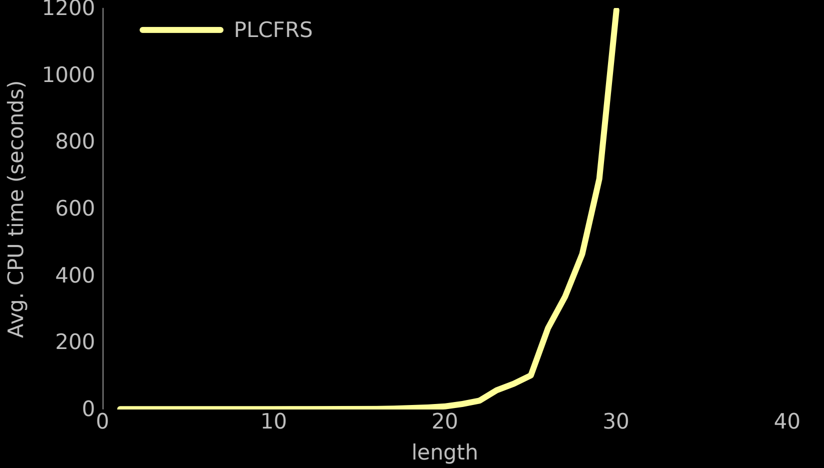
Solution:

1. Attach punctuation to highest constituent with neighbor on its right
2. Parentheses & quotation marks as low as possible around same constituent

Result: original and binarized treebank have same fan-out $\varphi = 4$; complexity $O(n^9)$

van Cranenburgh (2012), Efficient parsing with linear context-free rewriting systems. EACL.

But . . .



PLCFRS
Avg. CPU time (seconds)
0
200
400
600
800
1000
1200
0
10
20
30
40
length

Coarse-to-fine

Taming parsing complexity

Context summary estimates

Given a new item NP[2, 4], consult table giving (lower bound on) cost to construct parse with that item

- precomputed total order on possible items

Problem: table is large & expensive to compute for LCFRS

Kallmeyer & Maier (2010). Data-driven parsing with probabilistic linear context-free rewriting systems.

Taming parsing complexity

Coarse-to-fine parsing

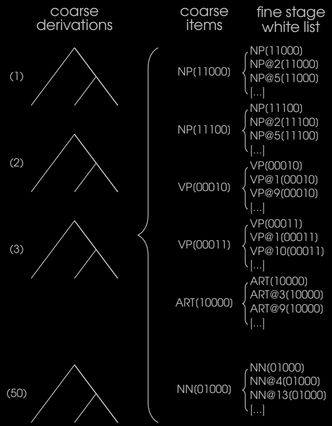
- Given grammars $G_1 \ldots G_n$,
  where G_m is an approximation of G_{m+1}
- Approximation: e.g., each label of G_m is mapped to multiple labels of G_{m+1}
- parse sentence with $G_1 \ldots G_n$,
  pruning any item not on whitelist derived from parsing w/previous grammar

Result: can parse very large grammars, e.g.:

1. lexicalized grammars
2. latent variable / split-merge grammars
3. data-oriented parsing grammars (next part)

Charniak et al. (2006), Multi-level coarse-to-fine parsing

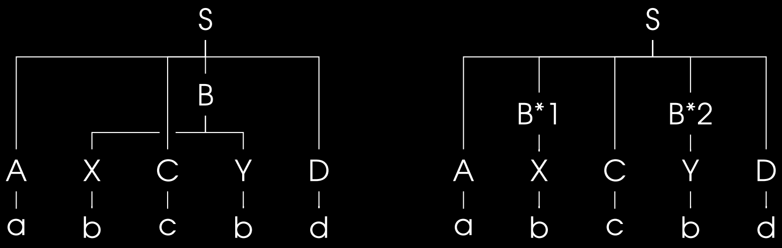
Coarse-to-fine



k-best G_m derivations
help prune G_{m+1} derivations.

van Cranenburgh, Scha, Sangati (2011), Discontinuous Data-Oriented Parsing: A mildly context-sensitive all-fragments grammar

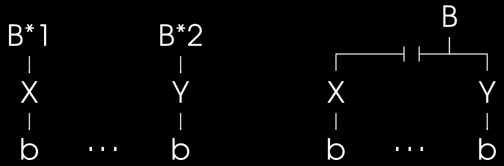
PCFG approximation of PLCFRS



- Transformation is reversible
- Increased independence assumption:
  ⇒ every component is a new node
- Language is a superset of original PLCFRS
  ⇒ coarser, overgenerating PCFG (‘split-PCFG’)

Boyd (2007). Discontinuity revisited.

Coarse-to-fine from PCFG to PLCFRS



- For a discontinuous item, look up multiple items from PCFG chart (`splitprune')
- e.g. $\{\ NP^{*}1 : [1,2],\ \ NP^{*}2 : [4,5]\ \} \Rightarrow NP_2 : [1,2;\ 4,5]$

Barthélemy et al. (2001) Guided parsing of range concatenation languages.
van Cranenburgh (2012), Efficient parsing with LCFRS

Evaluation with discontinuous constituents

- Labeled bracketings; e.g., NP : $[1, 2;\ 6, 8]$
- COLLINS.prm: discount ROOT node & punctuation (Collins, 1997)

  Makes a big difference!
  Negra dev set $\leq$ 25 words, PLCFRS:

Discounted	72.45 % F1
Not discounted	76.28 % F1

Results w/coarse-to-fine

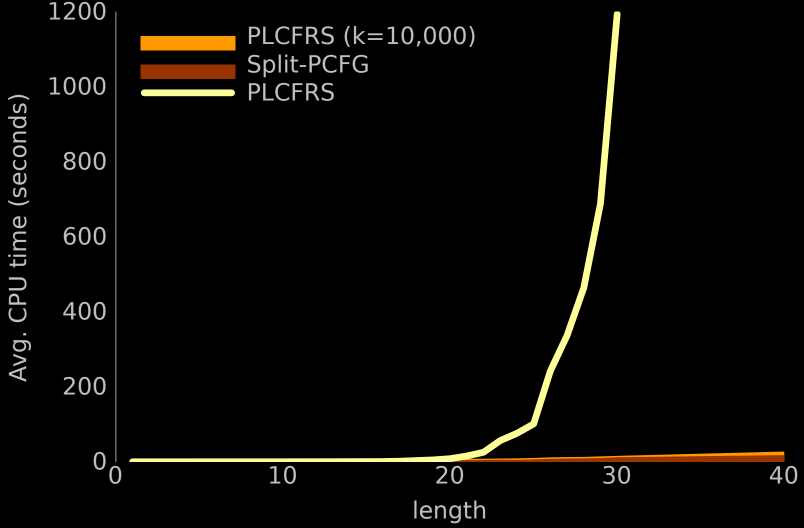
	words	F1 %	disc. brack.	time
PLCFRS	≤ 30	71.17	255	18h28m
Split-PCFG	≤ 30	71.29	162	
PLCFRS ($k = 10,000$)	≤ 30	70.91	250	1h22m
PLCFRS ($k = \infty$)	≤ 30	71.17	255	18h49m

Results w/coarse-to-fine

	words	F1 %	disc. brack.	time
PLCFRS	≤ 30	71.17	255	18h28m
Split-PCFG	≤ 30	71.29	162	
PLCFRS ($k = 10,000$)	≤ 30	70.91	250	1h22m
PLCFRS ($k = \infty$)	≤ 30	71.17	255	18h49m
Split-PCFG	∞	64.65	76	
PLCFRS ($k = 10,000$)	∞	64.94	267	

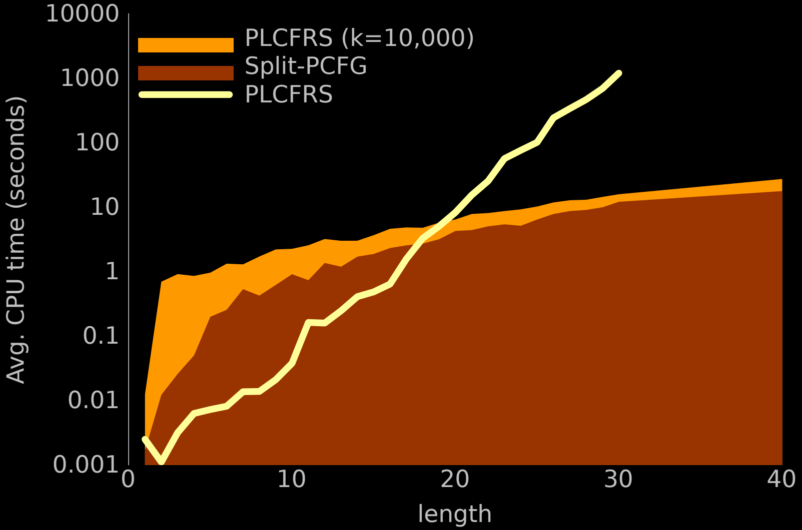
Table: Negra dev set, gold tags

Efficiency



Avg. CPU time (seconds)
1200
1000
800
600
400
200
0
0
10
20
30
40
length
PLCFRS (k=10,000)
Split-PCFG
PLCFRS

Efficiency (y-axis with log-scale)



Data-Oriented Parsing

Origin: Scha (1990)

- Cognitive research program
  Apply principles of gestalt perception to language
- Performance rather than competence model
- "(Takes) into account the statistical properties of actual language use."

Scha (1990): Language theory and language technology; competence and performance

Origin: Scha (1990)

- Cognitive research program
  Apply principles of gestalt perception to language
- Performance rather than competence model
- "(Takes) into account the statistical properties of actual language use."
- Rely on previous experience to process novel sentences,
  $\Rightarrow$ the treebank **is** the grammar.
  "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."

Scha (1990): Language theory and language technology; competence and performance

DOP principles (Scha 1990)

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."

Scha (1990): Language theory and language technology; competence and performance

Contrast: Treebank Grammars

Treebank grammar

trees ⇒ productions (+frequencies)

Manning & Schütze (1999, ch. 11): Foundations of Statistical NLP

Contrast: Treebank Grammars

Treebank grammar

trees $\Rightarrow$ productions (+frequencies)

Strong independence assumptions.

The probability of a subtree does not depend on …

Place invariance: …where in the string the words it dominates are (...)

Context-free: …words not dominated by the subtree.

Ancestor-free: …nodes in the derivation outside the subtree.

Manning & Schütze (1999, ch. 11): Foundations of Statistical NLP

Data-Oriented Parsing

Data-Oriented Parsing (DOP)

- trees ⇒ fragments (+frequencies)
- fragments are arbitrarily sized chunks from the corpus

Bod (1992): A computational model of language performance

Data-Oriented Parsing

Data-Oriented Parsing (DOP)

trees ⇒ fragments (+frequencies)

fragments are arbitrarily sized chunks from the corpus

⇒ instead of manually writing a grammar or refining probabilities . . .

consider all possible fragments from treebank

. . . and “let the statistics decide”

Bod (1992): A computational model of language performance

Definition of a DOP model

Fragments: what are the units on which the model operates?

Operations: what operations can be performed to combine or alter fragments?

Estimation: how will the probability of performing operations on particular fragments be determined?

Disambiguation: how will the most appropriate parse tree be selected among candidates?

Initial implementation: DOP1 (Bod, 1992)

- DOP as probabilistic tree-substitution grammar (TSG) for parsing phrase-structures
- Strongly equivalent to treebank PCFG (given all depth-1 fragments)
- …but more stochastic power due to probabilities of fragments

TSG is a versatile formalism:

- Parsing (Bayesian SR-TSG is current best result on WSJ!)
- Extraction of Multi-Word Expressions
- Grammaticality judgments
- Authorship classification

Bod (1992): A computational model of language performance

Definition of DOP1 (Bod, 1992)

Fragments: Connected subsets of phrase-structure trees, where each node either has all children in common with the original tree, or none (substitution site)

Operations: Left-most substitution
e.g., a fragment with an NP-slot can be combined with an NP fragment.

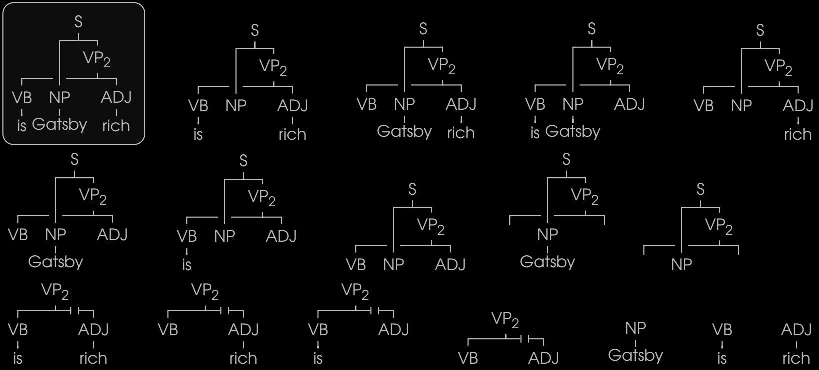
Estimation: Relative frequency of fragments (like rules in PCFG)

Disambiguation:

MPD: Most Probable single Derivation
MPP: Most Probable Parse
shortest derivation: Minimize no. of operations

Bod (1992): A computational model of language performance

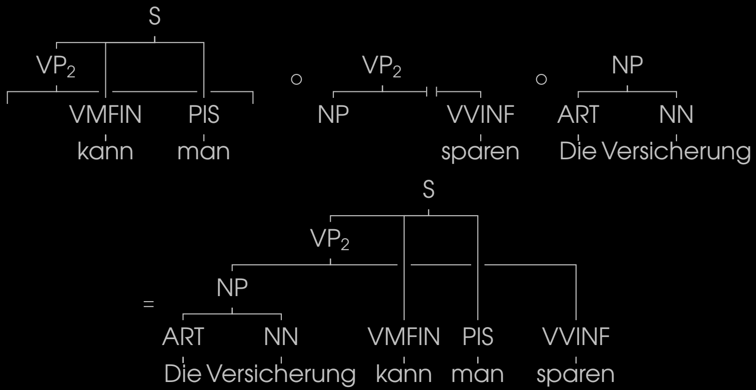
DOP fragments



$$P(f) = \frac{\text{count}(f)}{\sum_{f' \in F} \text{count}(f')} \text{ where } F = \{ f' \mid root(f') = root(f) \}$$

Note: discontinuous frontier nodes mark destination of components

DOP derivation



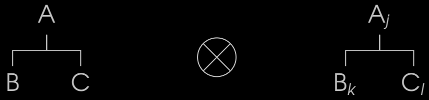
$$P(d) = P(f_1 \circ \cdots \circ f_n) = \prod_{f \in d} p(f)$$

$$P(t) = P(d_1) + \cdots + P(d_n) = \sum_{d \in D(t)} \prod_{f \in d} p(f)$$

DOP implementation issues

- Exponential number of **fragments** due to all-fragments assumption
- can restrict number of fragments by depth or frontier nodes &c.,
  $\Rightarrow$ **but:** not data-oriented!
- Exponential number of **derivations**
  - Makes finding MPP NP-hard.
  - Can approximate with random or n-best derivations.

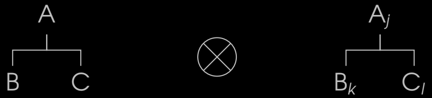
DOP reduction



$$
\begin{array}{llll}
A_j(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/a_j) & A(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/a) \\
A_j(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/a_j) & A(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/a) \\
A_j(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/a_j) & A(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/a) \\
A_j(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/a_j) & A(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/a)
\end{array}
$$

Goodman (2003): Efficient parsing of DOP with PCFG-reductions

DOP reduction



$$
\begin{array}{llll}
A_j(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/a_j) & A(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (1/a) \\
A_j(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/a_j) & A(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C(\vec{\alpha_C}) & (b_k/a) \\
A_j(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/a_j) & A(\vec{\alpha}) \rightarrow B(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (c_l/a) \\
A_j(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/a_j) & A(\vec{\alpha}) \rightarrow B_k(\vec{\alpha_B})\ C_l(\vec{\alpha_C}) & (b_k c_l/a)
\end{array}
$$

- Polynomial time parsing with **all** fragments!
- **But:** probabilities are distributed over 8 rules per node in the treebank
- need to sum many derivations to approximate parse probability

Goodman (2003): Efficient parsing of DOP with PCFG-reductions

Results w/DOP reduction

CTF stage	F1 %
Split-PCFG	66.81
PLCFRS	67.26
DOP reduction	74.27

(Negra dev set $\leq$ 40 words, gold tags)

van Cranenburgh (2012), Efficient parsing with linear context-free rewriting systems.

Double-DOP

- Extract fragments that occur at least twice in treebank
- Number of fragments is small enough to parse with directly

Sangati & Zuidema (2011). Accurate parsing w/compact TSGs: Double-DOP

Double-DOP

- Extract fragments that occur at least twice in treebank
- Number of fragments is small enough to parse with directly
- Fragments mapped to unique rules, relative frequencies as probabilities
  - Remove internal nodes,
    leaves root node, substitution sites & terminals
    $X \rightarrow X_1 \ldots X_n$
- Reconstruct derivations after parsing

Sangati & Zuidema (2011). Accurate parsing w/compact TSGs: Double-DOP

Double-DOP

- Extract fragments that occur at least twice in treebank
- Number of fragments is small enough to parse with directly
- Fragments mapped to unique rules, relative frequencies as probabilities
  - Remove internal nodes, leaves root node, substitution sites & terminals $X \rightarrow X_1 \dots X_n$
- Reconstruct derivations after parsing

Contrast: Bayesian TSGs w/Dirichlet Priors (Cohn et al., 2009, Post & Gildea, 2009)

Sangati & Zuidema (2011). Accurate parsing w/compact TSGs: Double-DOP

Extract recurring fragments in linear average time

Tree kernel: find similarities in trees of treebank

- Worst case: need to compare every node to all other nodes in treebank
- Speed up fragment extraction by sorting nodes of trees:
  $\Rightarrow$ Aligns potentially equal nodes, allowing us to skip the rest! (Moschitti 2006)
- Figure out fragments from list of matching nodes

Moschitti (2006): Making tree kernels practical for natural language learning
van Cranenburgh (2012), Extracting tree fragments in linear average time

Extract recurring fragments in linear average time

Implementation	CPU	Time Wall clock	fragments
Sangati (2012):			
Quadratic tree kernel, WSJ	160	10h00m	1,023,092
van Cranenburgh (2012):			
Fast tree kernel, WSJ	2.3	0h09m	1,023,880
Fast tree kernel, Negra	0.8	0h04m	370,081

Wall clock time is when using 16 cores.

Sangati (2012): Decomposing and Regenerating Syntactic Trees
van Cranenburgh (2012), Extracting tree fragments in linear average time

Results w/Double-DOP

	F1 %
DOP reduction	74.27
Double-DOP	

(Negra dev set $\leq$ 40 words, gold tags)

Results w/Double-DOP

	F1 %
DOP reduction	74.27
Double-DOP	76.58

(Negra dev set $\leq$ 40 words, gold tags)

Explicit representation of recurring fragments with Double-DOP leads to better sample of derivations than parsing with all fragments

Results w/Double-DOP

	k=50 F1 %	k=5000 F1 %
DOP reduction	74.27	73.45
Double-DOP	76.58	

Results w/Double-DOP

	k=50 F1 %	k=5000 F1 %
DOP reduction	74.27	73.45
Double-DOP	76.58	78.52

(Negra dev set $\leq$ 40 words, gold tags)

$\Rightarrow$ For Double-DOP, performance does not deterioriate with expanded search space.

Part-of-speech tagging

	tags	F1 %
Gold tags	100	78.52
Stanford tagger	96.92	

Part-of-speech tagging

	tags	F1 %
Gold tags	100	78.52
Stanford tagger	96.92	74.78

Part-of-speech tagging

	tags	F1 %
Gold tags	100	78.52
Stanford tagger	96.92	74.78

Non-discontinuous work on Negra $\leq$ 40 words:

Without discontinuity	tags	F1 %
Sangati (2012)	94.75	76.5
Petrov & Klein (2008)		81.5

Part-of-speech tagging

With discontinuity	tags	F1 %
Gold tags	100	78.52
Stanford tagger	96.92	74.78

Non-discontinuous work on Negra $\leq$ 40 words:

Without discontinuity	tags	F1 %
Sangati (2012)	94.75	76.5
Petrov & Klein (2008)		81.5

NB: with the last two results, discontinuities have been removed from both training & test sets, so scores are not directly comparable.

Part-of-speech tagging

With discontinuity	tags	F1 %	exact	UAS
Gold tags	100	78.52	41.44	88.62
Stanford tagger	96.92	74.78	37.03	85.0

Non-discontinuous work on Negra $\leq$ 40 words:

Without discontinuity	tags	F1 %	exact	UAS
Sangati (2012)	94.75	76.5	34.59	? 82.63
Petrov & Klein (2008)		81.5	45.2	

NB: with the last two results, discontinuities have been removed from both training & test sets, so scores are not directly comparable.

Do we need LCFRS for discontinuity?

Split-PCFG	Split-PCFG
⇓	
PLCFRS	⇓
⇓	
PLCFRS Double-DOP	Split-Double-DOP
78.52 % F1	
41.44 % EX	
88.62 UAS	

Do we need LCFRS for discontinuity?

Split-PCFG	Split-PCFG
⇓	
PLCFRS	⇓
⇓	
PLCFRS Double-DOP	Split-Double-DOP
78.52 % F1	78.25 % F1
41.44 % EX	41.44 % EX
88.62 UAS	88.27 UAS

Answer: No!

Fragments can capture discontinuous contexts

Conclusions

- Coarse-to-fine is indispensable for large grammars & complex formalisms

Conclusions

- Coarse-to-fine is indispensable for large grammars & complex formalisms
- All fragments vs. selected fragments

Conclusions

- Coarse-to-fine is indispensable for large grammars & complex formalisms
- All fragments vs. selected fragments
  - Explicit representation of recurring fragments with Double-DOP leads to better sample of derivations than parsing with all fragments

Conclusions

- Coarse-to-fine is indispensable for large grammars & complex formalisms
- All fragments vs. selected fragments
  - Explicit representation of recurring fragments with Double-DOP leads to better sample of derivations than parsing with all fragments
- Not necessary to parse beyond CFG!
  $\Rightarrow$ Increase amount of context through fragments / labels

Conclusions

- Coarse-to-fine is indispensable for large grammars & complex formalisms
- All fragments vs. selected fragments
  - Explicit representation of recurring fragments with Double-DOP leads to better sample of derivations than parsing with all fragments
- Not necessary to parse beyond CFG!
  $\Rightarrow$ Increase amount of context through fragments / labels
  - LCFRS could be exploited for other things than discontinuity: adjunction, synchronous parsing, ...

References

Andreas van Cranenburgh, Remko Scha, and Federico Sangati (2011), Discontinuous Data-Oriented Parsing: A mildly context-sensitive all-fragments grammar, Proceedings of SPMRL workshop. October 2011, Dublin, Ireland.

Andreas van Cranenburgh (2012), Efficient parsing with linear context-free rewriting systems. Proceedings of EACL, April 2012, Avignon, France.

Andreas van Cranenburgh (2012), Extracting tree fragments in linear average time. ILLC technical report. http://dare.uva.nl/en/record/421534

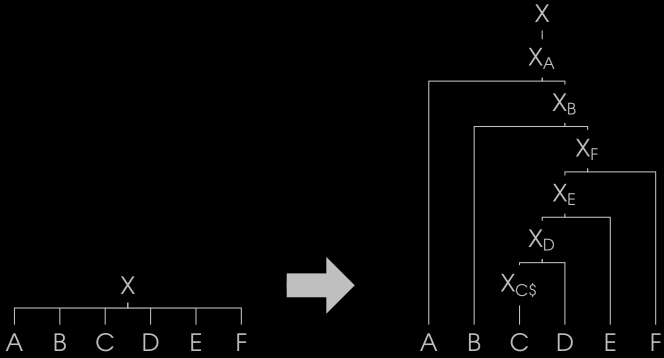
Slides, code: `http://staff.science.uva.nl/~acranenb`

Wait . . . there's more

BACKUP SLIDES

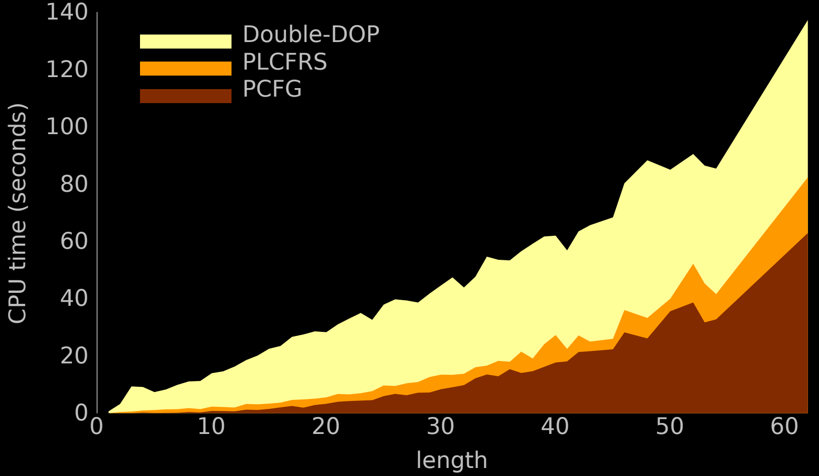
Binarization

- mark heads of constituents
- head-outward binarization (parse head first)
- no parent annotation: $v = 1$
- horizontal Markovization: $h = 1$



Klein & Manning (2003): Accurate unlexicalized parsing.

Efficiency


Double-DOP
PLCFRS
PCFG
CPU time (seconds)
length
0
20
40
60
80
100
120
140
0
10
20
30
40
50
60

Breakdown by category

label	% gold	Precision	Recall	F_1
NP	29.97	74.91	74.58	74.75
PP	26.05	78.72	78.59	78.65
S	18.71	88.96	88.12	88.54
VP	10.28	61.85	61.49	61.67
AP	4.06	72.53	72.53	72.53
CNP	3.33	62.50	66.99	64.67
MPN	2.44	95.12	97.50	96.30
VZ	1.19	100.00	100.00	100.00
AVP	0.92	45.16	50.91	47.86
CS	0.89	66.67	45.45	54.05

Table: Breakdown of F-scores of the 10 most frequent categories, for Double Disco-DOP on Negra development set up to 40 words.