Data-Oriented Parsing and Discontinuous Constituents

Andreas van Cranenburgh

Huygens ING
Royal Netherlands Academy of Arts and Sciences

Institute for Logic, Language and Computation
University of Amsterdam

March 4, 2014

Amsterdam 2014

This lecture

1. Context & context freedom
2. Parsing with . . .
   - discontinuous constituents:
     Linear Context-Free Rewriting Systems (LCFRS)
   - treebank fragments:
     Data-Oriented Parsing (DOP)
     Tree-Substitution Grammar (TSG)
3. Other non-context-free challenges

Context and Context Freedom

Two meanings for context-free:

1. Rewrite operations are independent from anything not being rewritten.
   Counterexamples: HPSG, LFG, &c.
   model theoretic syntax;
   unification based $\Rightarrow$ exp. time complexity

Context and Context Freedom

Two meanings for context-free:

1. Rewrite operations are independent from anything not being rewritten.
   Counterexamples: HPSG, LFG, &c.
   model theoretic syntax;
   unification based $\Rightarrow$ exp. time complexity
2. CFG: a grammar $\langle V, T, S, P \rangle$ with the above property, such that productions in P are of the form:

$$\alpha \rightarrow \beta_1 \ldots \beta_n$$

   Counterexamples: TAG, CCG, &c.

NB: a formalism can be context-free (1) without being Context-Free (2) ...

The Domain of Locality

Domain of Locality: the information which is available while applying rewrite operations

CFG: parent $\rightarrow$ nonterminals dominating adjacent terminals

LCFRS: parent $\rightarrow$ nonterminals dominating terminals regardless of position in sentence

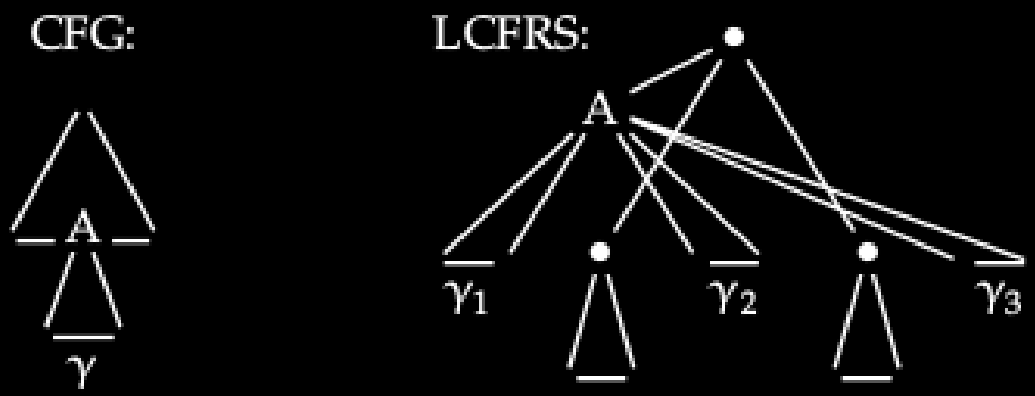


Figure 3
Different domains of locality.

Discontinuous Constituents

Example:

- Why did the chicken cross the road?
- The chicken crossed the road to get to the other side.

Non-local information in PTB: traces

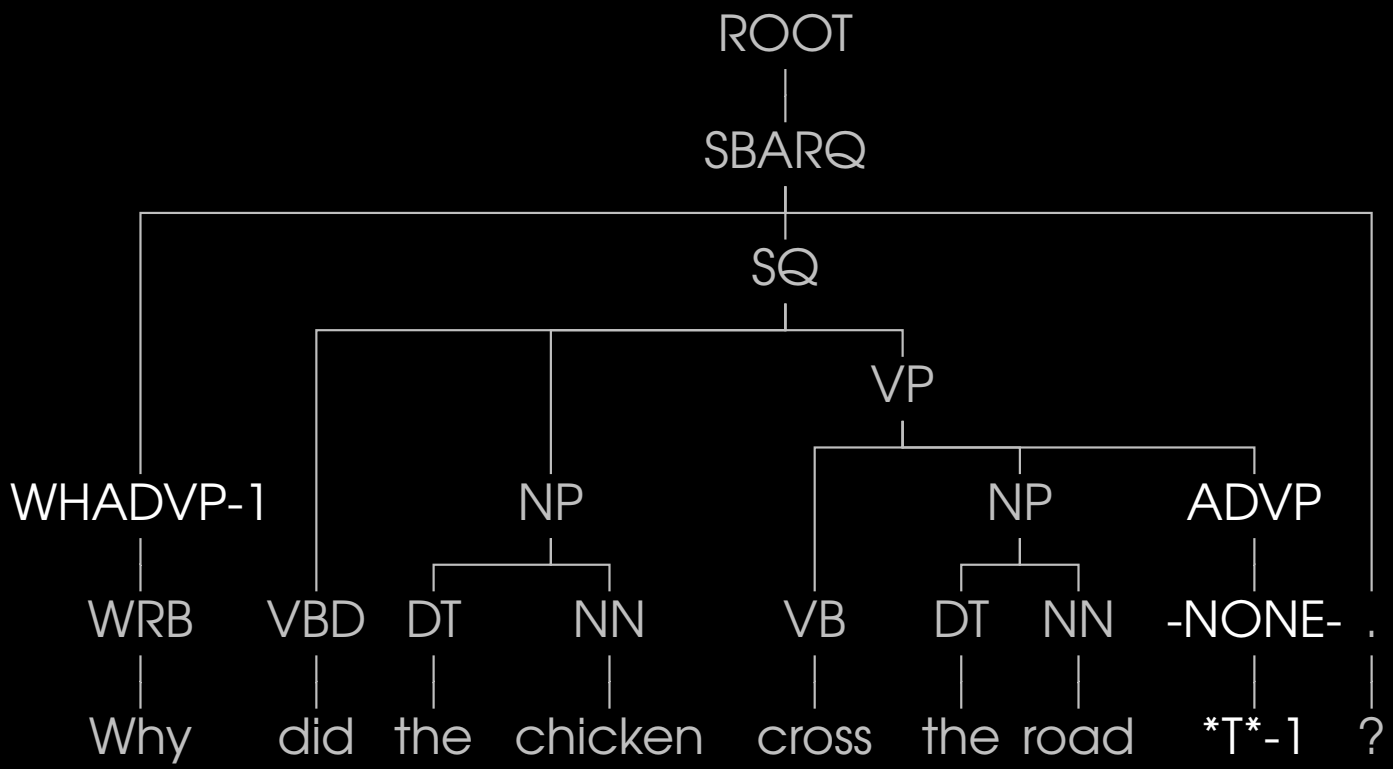


Figure : PTB-style annotation.

Discontinuous trees

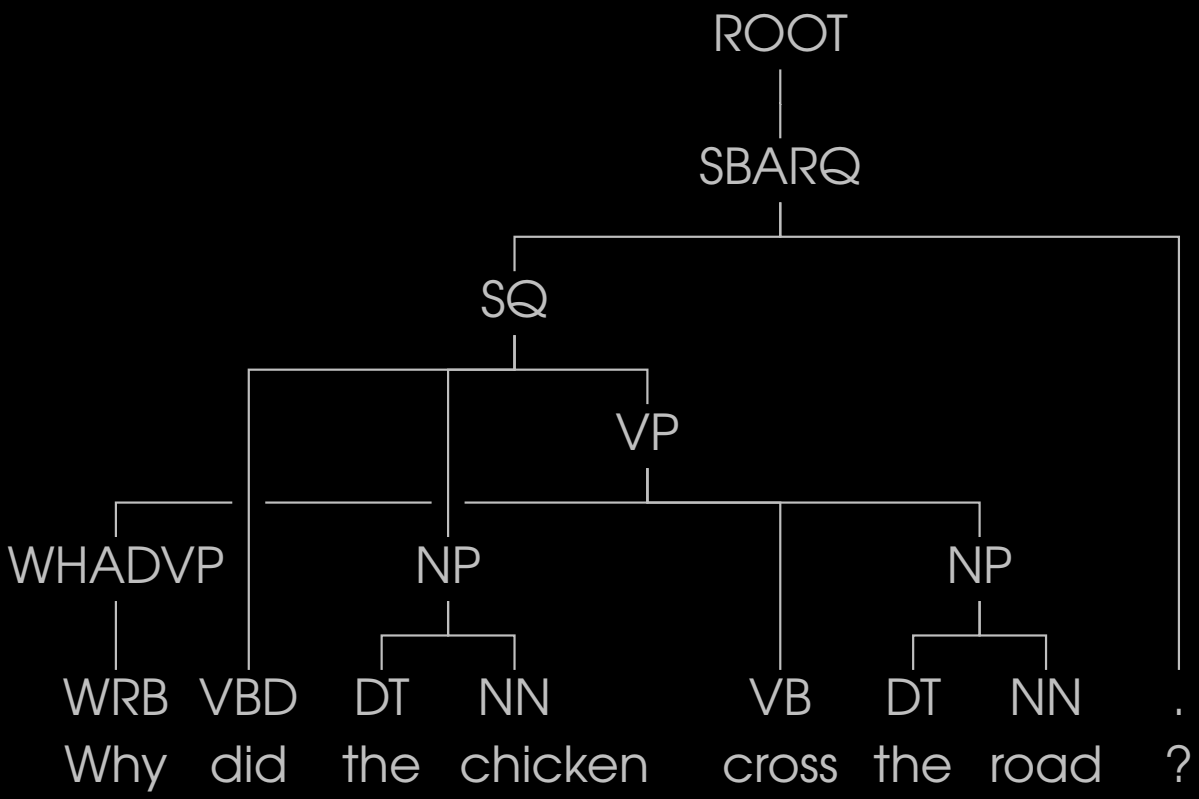


Figure : A tree with a discontinuous constituent.

Discontinuous constituents

Motivation:

- Handle flexible word-order, extraposition, &c.
- Capture argument structure
- Combine information from constituency & dependency structures

(NB: non-projectivity is a subset of discontinuous phenomena)

Discontinuous treebanks

Treebanks with discontinuous constituents:

German/Negra: Skut et al. (1997). An annotation scheme for free word order languages.

Dutch/Alpino: van der Beek (2002). The Alpino dependency treebank.

English/PTB (after conversion): Evang & Kallmeyer (2011). PLCFRS Parsing of English Discontinuous Constituents.

Swedish, Polish, ...

Discontinuous trees

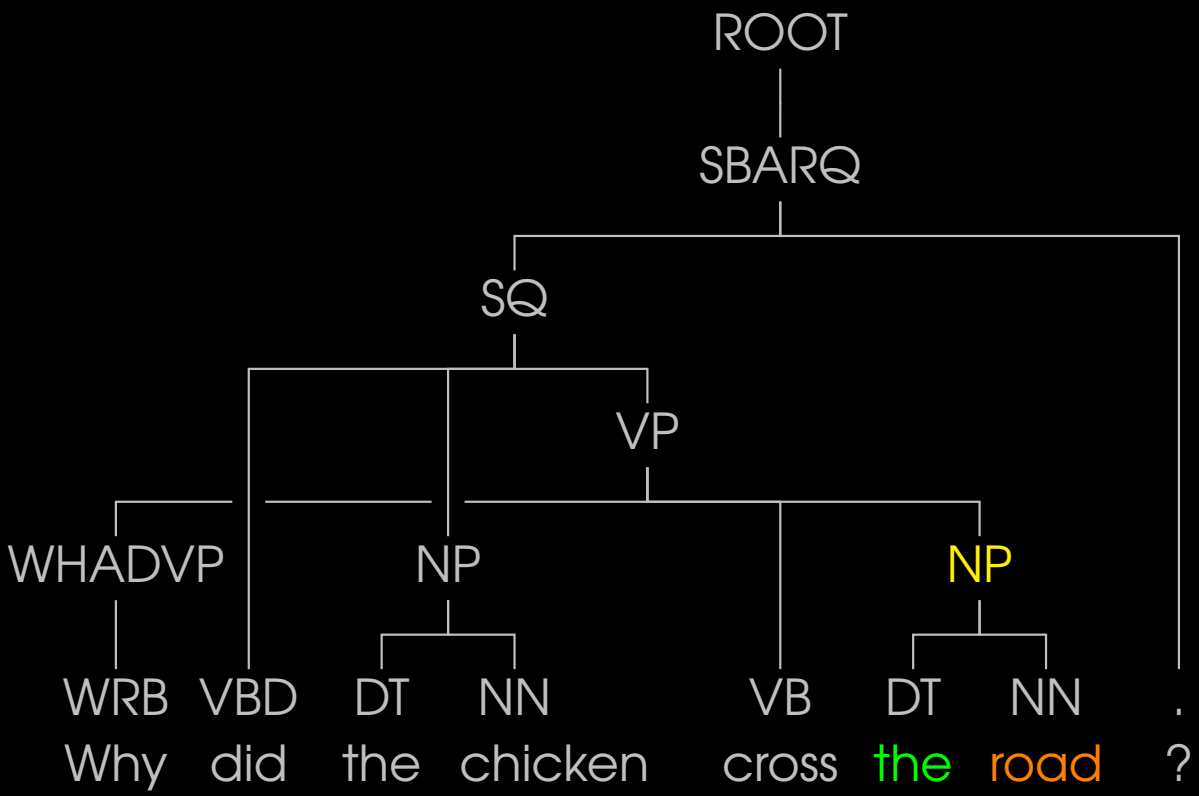


Figure : A tree with a discontinuous constituent.

Context-Free Grammar (CFG)

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

Discontinuous trees

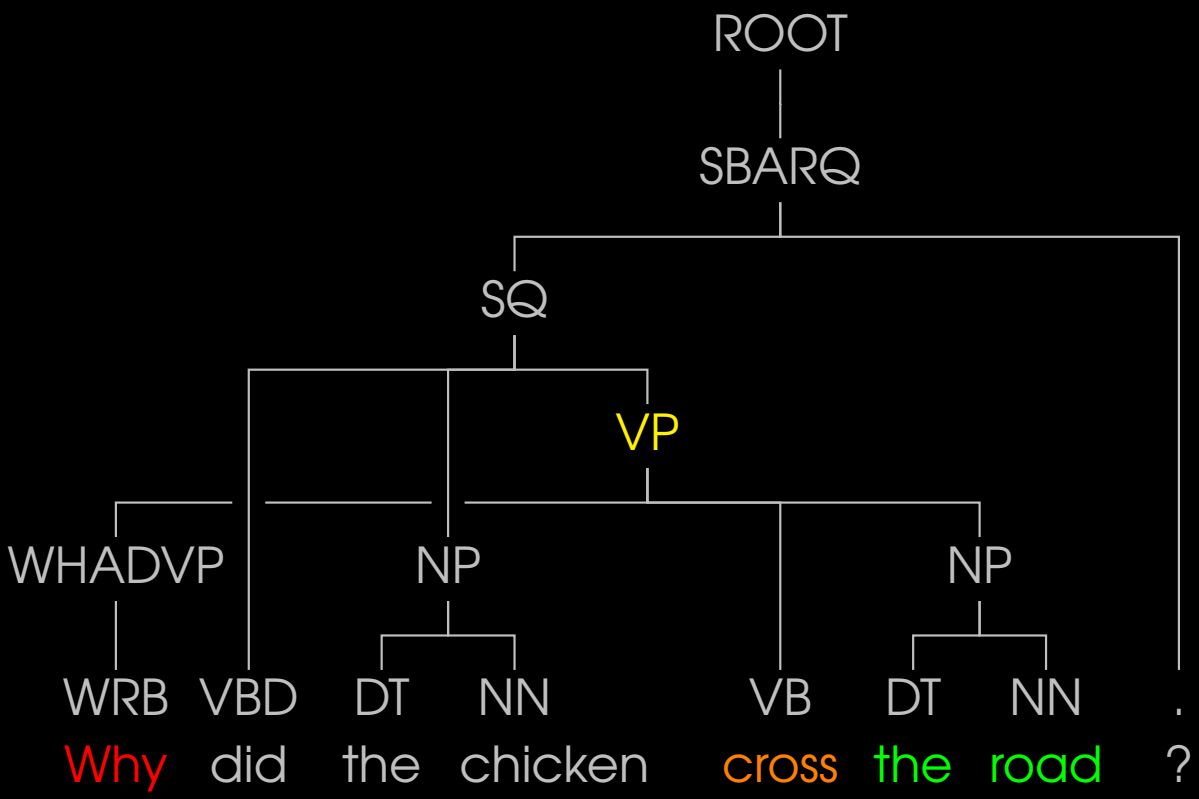


Figure : A tree with a discontinuous constituent.

Linear Context-Free Rewriting System (LCFRS)

$VP_2(a, bc) \rightarrow WHADVP(a)\ VB(b)\ NP(c)$

Discontinuous trees

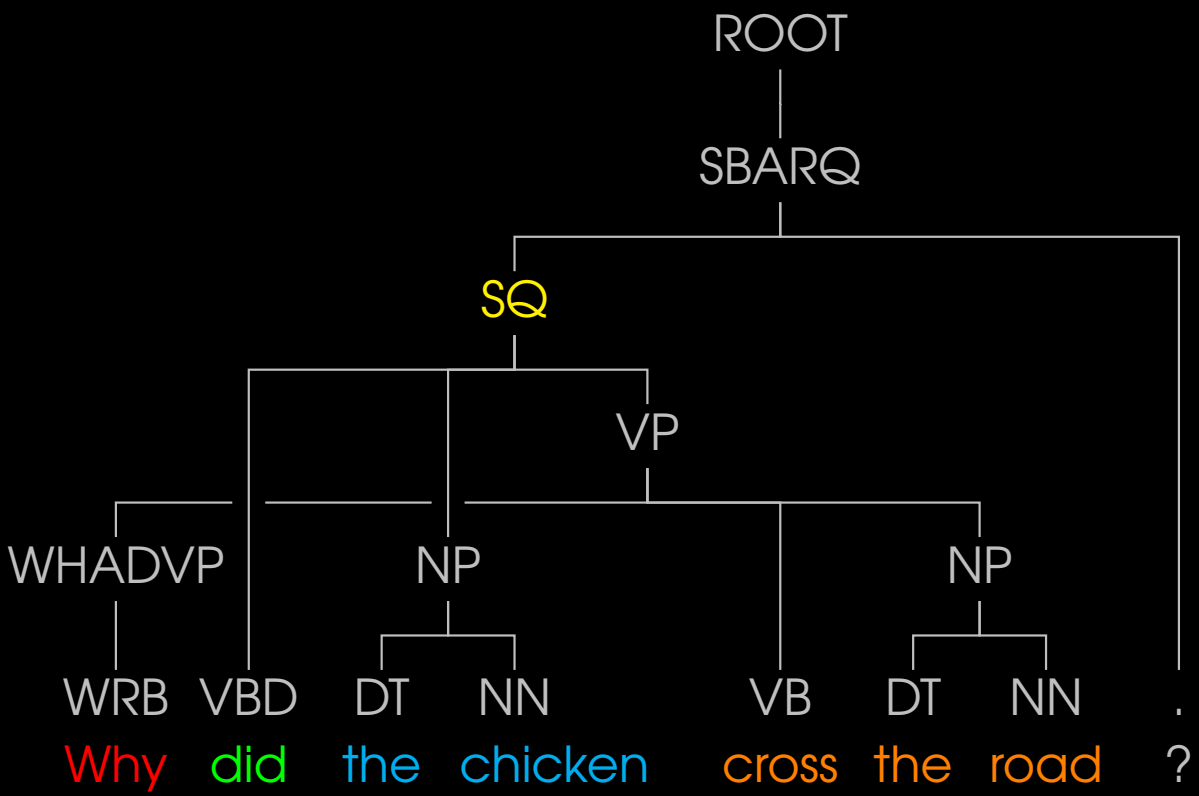


Figure : A tree with a discontinuous constituent.

Linear Context-Free Rewriting System (LCFRS)

$VP_2(a, bc) \rightarrow WHADVP(a)\ VB(b)\ NP(c)$

$SQ(abcd) \rightarrow VBD(b)\ NP(c)\ VP_2(a, d)$

Linear Context-Free Rewriting Systems: I

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: I

LCFRS are a generalization of CFG:
$\Rightarrow$ 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., formal grammar

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

Linear Context-Free Rewriting Systems: II

LCFRS are weakly equivalent to:

- Combinatory Categorial Grammar
- Tree-Adjoining Grammar
- Synchronous Context-Free Grammar
- Multiple Context-Free Grammar
- Minimalist Grammar
- &c. . . .

$\Rightarrow$ LCFRS form a 'lingua franca' formalism.

Complexity of LCFRS parsing

- LCFRS are Mildly Context-Sensitive grammar formalisms.
- Parsing an LCFRS has polynomial time complexity:

$$\mathcal{O}(n^{r\varphi})$$

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

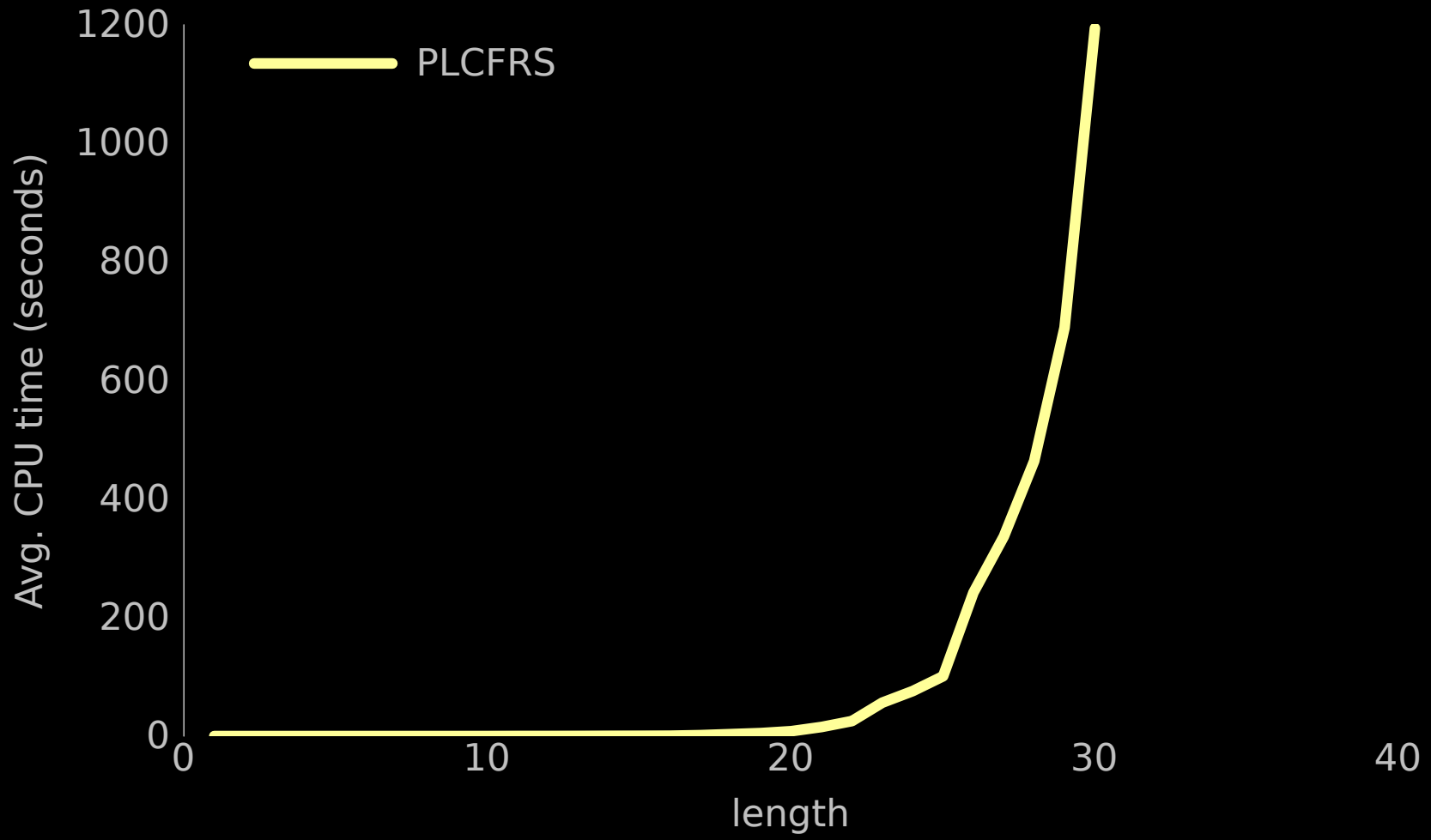
  $\Rightarrow$ infinite 2D hierarchy of languages between context-free and context-sensitive.
- Both CFG & LCFRS $\in$ LOGCFL

Parsing with LCFRS

- An LCFRS can be parsed with tabular parsing algorithm (similar to CKY):
- Agenda-based probabilistic parser for LCFRS (Kallmeyer & Maier 2010);
  extended to produce *k*-best derivations
- Rules can be read off from treebank,
  relative frequencies give probabilistic LCFRS (PLCFRS)

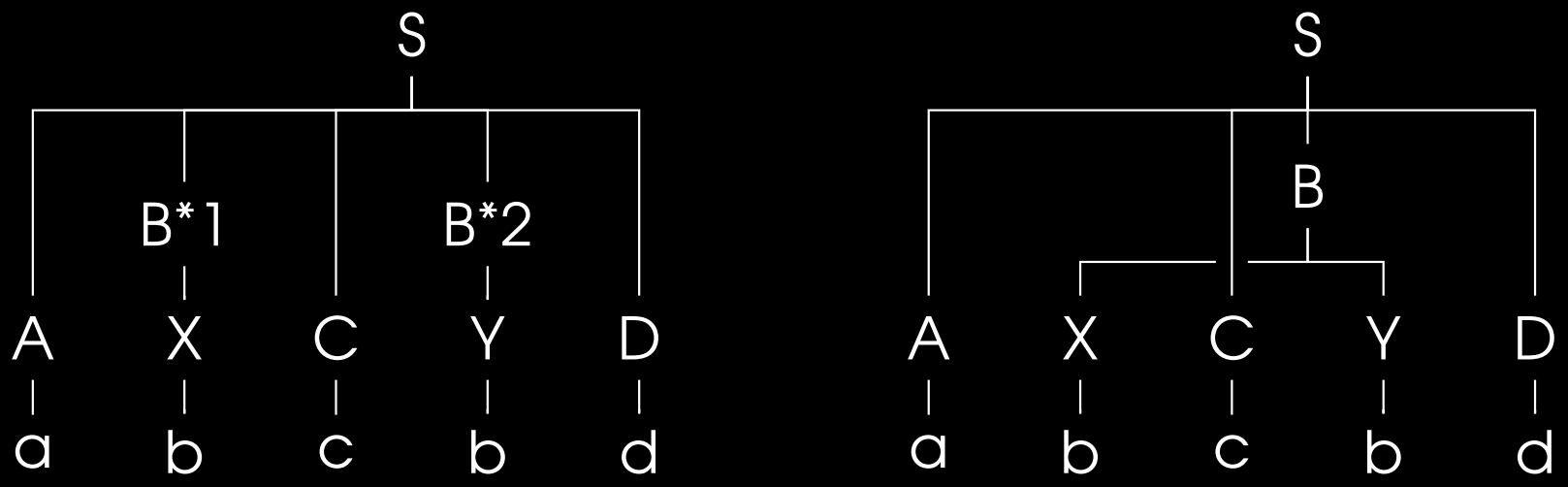
Kallmeyer & Maier (2010). Data-driven parsing with probabilistic linear context-free rewriting systems.

But . . .



Negra dev. set, gold tags

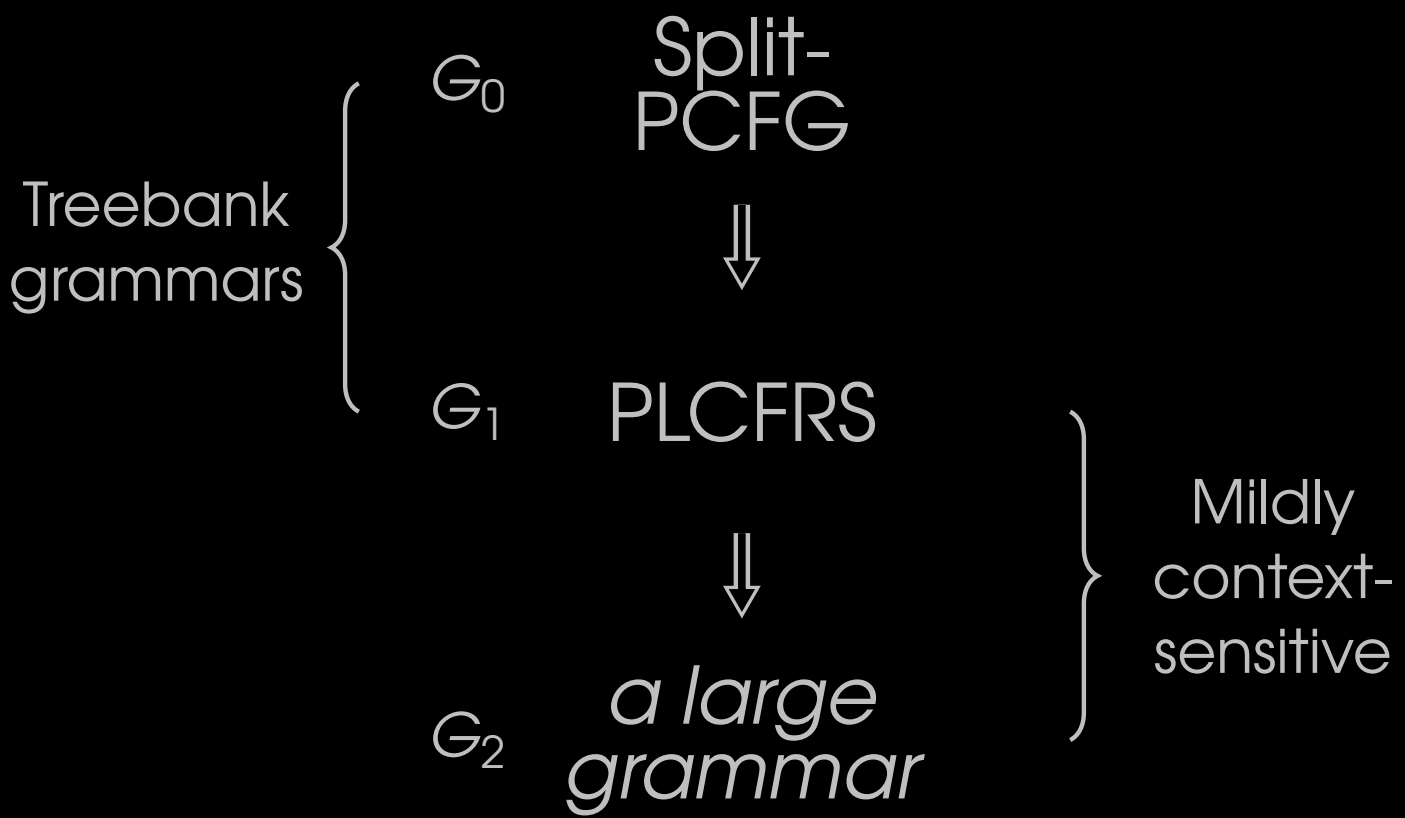
PCFG approximation of PLCFRS



- Transformation is reversible
- Increased independence assumption:
  $\Rightarrow$ every component is a new node
- Language is a superset of original PLCFRS
  $\Rightarrow$ coarser, overgenerating PCFG (`split-PCFG')

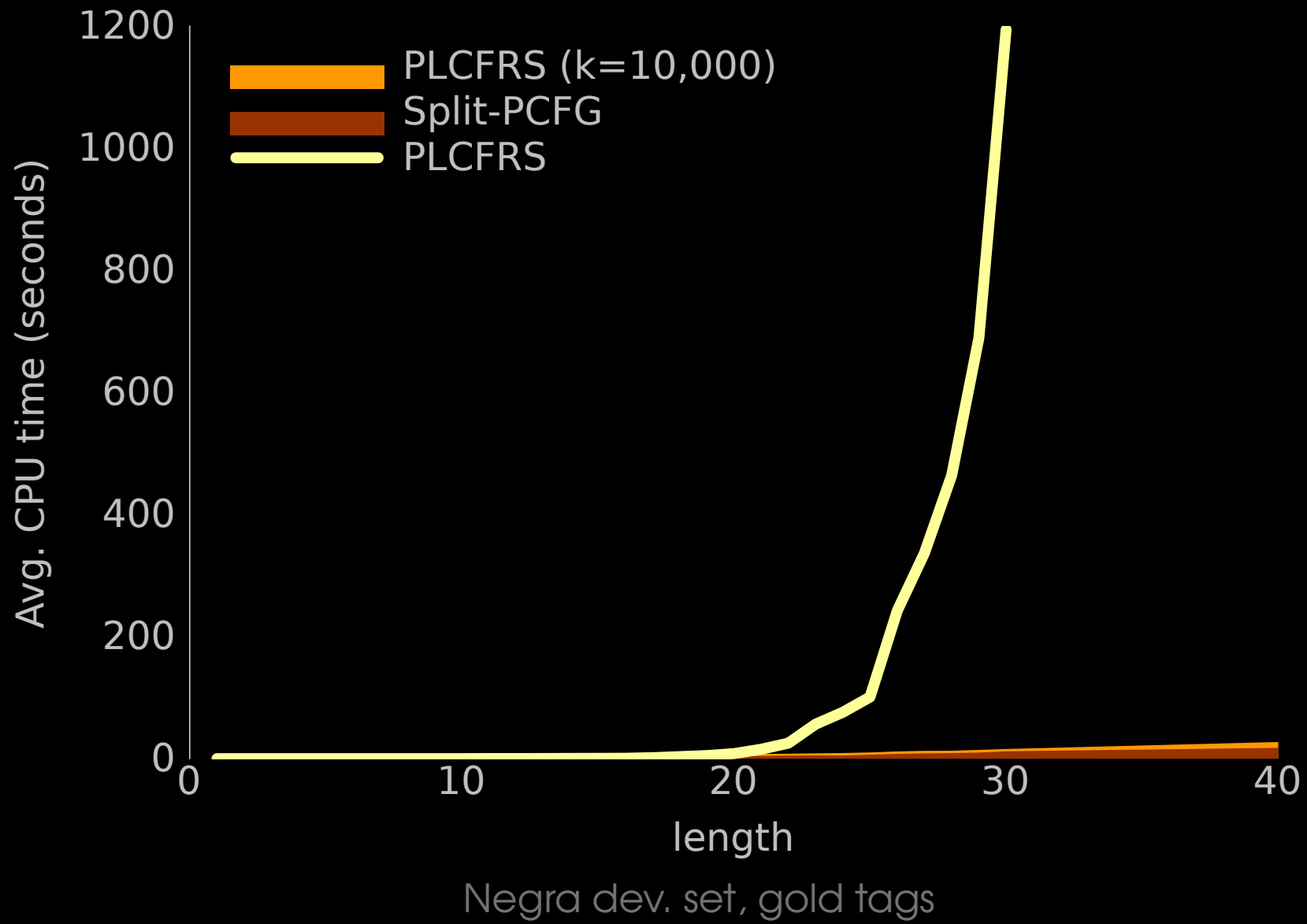
Boyd (2007). Discontinuity revisited.

Coarse-to-fine pipeline



prune parsing with G_{m+1} by only considering items in k-best G_m derivations.

With coarse-to-fine



Negra dev. set, gold tags

Data-Oriented Parsing

Treebank grammar

trees $\Rightarrow$ productions + rel. frequencies

$\Rightarrow$ problematic independence assumptions

Data-Oriented Parsing (DOP)

trees $\Rightarrow$ fragments + rel. frequencies

fragments are arbitrarily sized chunks from the corpus

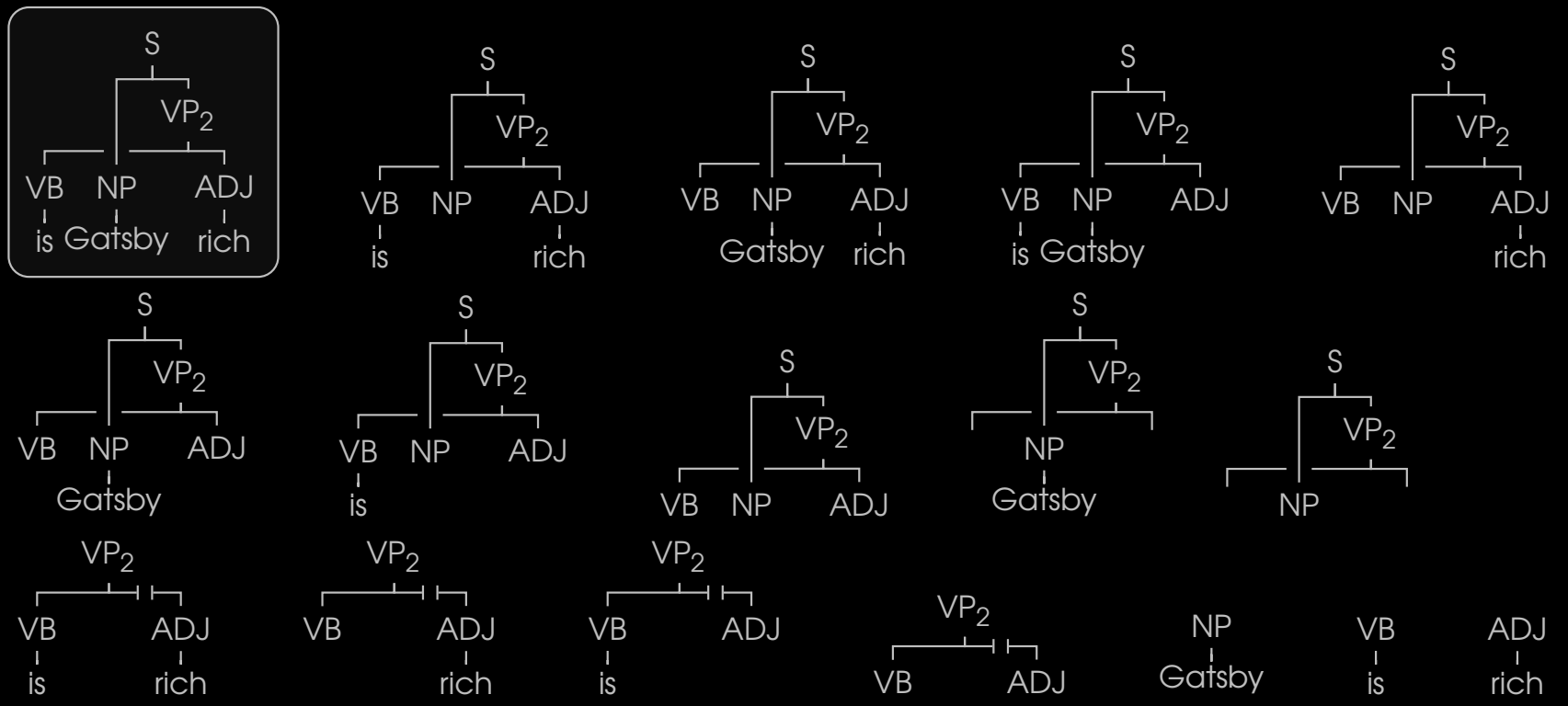
consider all possible fragments from treebank

. . . and "let the statistics decide"

Scha (1990): Lang. theory and lang. tech.; competence and performance

Bod (1992): A computational model of language performance

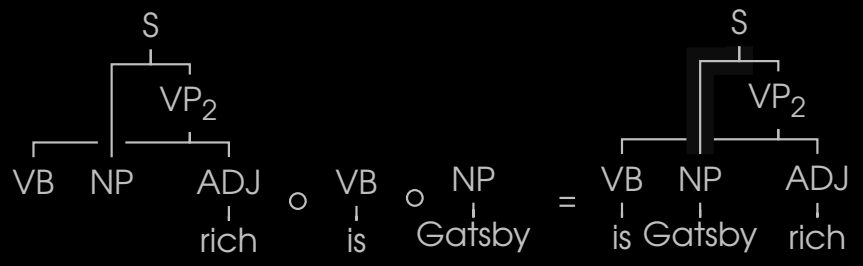
DOP fragments



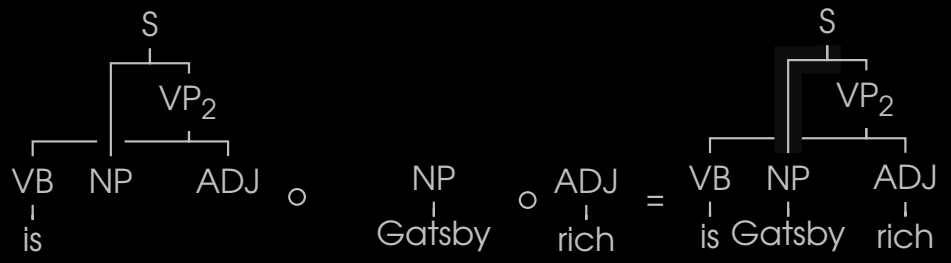
$$P(f) = \frac{\text{count}(f)}{\sum_{f' \in F} \text{count}(f')} \text{ where } F = \{ f' \mid root(f') = root(f) \}$$

Note: discontinuous frontier non-terminals mark destination of components

DOP derivation



$P(d) = 0.2$



$P(d) = 0.3$

Derivations for this tree $P(t) = 0.5$

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

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

Tree-Substitution Grammar

This DOP model (Bod 1992) is based on
Tree-Substitution Grammar (TSG):

- Weakly equivalent to CFG; typically strongly equivalent as well; advantage is in stochastic power of Probabilistic TSG.
- Same Context-Free property as CFG, but multiple productions applied at once;
  $\Rightarrow$ captures more structural relations than PCFG.
- CFG backbone can be replaced with LCFRS to get Discontinuous Tree-Substitution Grammar ($\text{PTSG}_{\text{LCFRS}}$).

DOP implementation issues

Exponential number of fragments
due to all-fragments assumption

- Can use DOP reduction (Goodman 2003);
  weight of fragments spread over many productions
- Can restrict number of fragments
  by depth or frontier nodes &c.,
  $\Rightarrow$ but: not data-oriented!

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

Double-DOP

- Extract fragments that occur at least twice in treebank
- For every pair of trees, extract maximal overlapping fragments
- Can be extracted in linear average time
- Number of fragments is small enough to parse with directly

Sangati & Zuidema (2011). Accurate parsing w/compact TSGs: Double-DOP
van Cranenburgh (2012). Extracting tree fragments in linear average time

From fragments to grammar

- 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

Preprocessing

- Remove function labels
- Binarize w/markovization (h=1, v=1)
- Simple unknown word model
  - Rare words replaced by features (model 4 from Stanford parser)
  - Reserve probability mass for unseen (tag, word) pairs

Results w/Double-DOP

	F1 %
DOP reduction	74.3
Double-DOP	

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

Results w/Double-DOP

	F1 %
DOP reduction	74.3
Double-DOP	76.3

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

Also: parsing 3$\times$ faster, grammar 3$\times$ smaller

Results w/Double-DOP

	k=50 F1 %	k=5000 F1 %
DOP reduction	74.3	73.5
Double-DOP	76.3	

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

What if we reduce pruning?

Results w/Double-DOP

	k=50 F1 %	k=5000 F1 %
DOP reduction	74.3	73.5
Double-DOP	76.3	77.7

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

What if we reduce pruning?
$\Rightarrow$ For Double-DOP, performance does not deterioriate with expanded search space.

Parsing results: test sets

Parser, treebank	F1	EX
GERMAN		
HaNi2008, Tiger	75.3	32.6
CrBo2013, Tiger	**78.8**	**40.8**
ENGLISH		
SaZu2011, wsj	87.9	33.7
EvKa2011, disc. wsj	79.0	
CrBo2013, disc. wsj	**85.6**	31.3
DUTCH		
CrBo2013, Lassy	77.0	35.2

CrBo: van Cranenburgh & Bod (2013);
HaNi: Hall & Nivre (2008); SaZu: Sangati & Zuidema (2011).

Can DOP handle discontuinity without LCFRS?

Split-PCFG

⇓

PLCFRS

⇓

PLCFRS Double-DOP

77.7 % F1

41.5 % EX

Split-PCFG

⇓

Split-Double-DOP

Can DOP handle discontuinity without LCFRS?

Split-PCFG

⇓

PLCFRS

⇓

PLCFRS Double-DOP

77.7 % F1

41.5 % EX

Split-PCFG

⇓

Split-Double-DOP

78.1 % F1

42.0 % EX

Answer: Yes!

Fragments can capture discontinuous contexts

Summary

Formally,	$\mathrm{CFG} = \mathrm{TSG}_{\mathrm{CFG}} \subset \mathrm{LCFRS}$
Stochastically,	$\mathrm{PCFG} \subset \mathrm{PTSG}_{\mathrm{CFG}}$
Empirically,	$\mathrm{PTSG}_{\mathrm{LCFRS}} \approx \mathrm{PTSG}_{\mathrm{CFG}}$

Limitations of parsing with (approximated) LCFRS

- Just as with CFG, an LCFRS production strictly covers a single configuration of constituents.
- Consider:
  A man who was seeking a unicorn walked into the room.
  vs.
  A man walked into the room who was seeking a unicorn.
- Discontinuous constituents are relatively rare
  $\Rightarrow$ sparse data problem

Unsupervised discontinuous parsing?

Problems:

- A sentence with n words has $O(n^2)$ possible continuous constituents. The same sentence may have $O(2^n)$ discontinuous constituents.

Unsupervised discontinuous parsing?

Problems:

- A sentence with n words has $O(n^2)$ possible continuous constituents.
  The same sentence may have $O(2^n)$ discontinuous constituents.
- For continuous constituents, adjacent co-occurrence is the key heuristic to determine constituency.
  Is there a surface heuristic for discontinuous constituency?
  $\Rightarrow$ Morphology in case of morphologically-rich languages.

Unsupervised discontinuous parsing

Rule-based: e.g., Yoshinaka (2010), Polynomial-time identification of multiple context-free languages from positive data and membership queries

Unsupervised discontinuous parsing

Rule-based: e.g., Yoshinaka (2010), Polynomial-time identification of multiple context-free languages from positive data and membership queries

Statistical: Stanford parser jointly parsers with a PCFG and a dependency model.
Same technique could be applied to induce a PCFG and a non-projective dependency model.

Other non-context-free challenges: supervised

Adjunction: More compact grammar and less sparsity when adjuncts can be inserted with an operation of the grammar formalism; e.g., Tree-Insertion Grammar.

Other non-context-free challenges: supervised

Adjunction: More compact grammar and less sparsity when adjuncts can be inserted with an operation of the grammar formalism; e.g., Tree-Insertion Grammar.

Grammatical functions: Most models reconstruct functions after parsing using machine learning

Other non-context-free challenges: supervised

Adjunction: More compact grammar and less sparsity when adjuncts can be inserted with an operation of the grammar formalism; e.g., Tree-Insertion Grammar.

Grammatical functions: Most models reconstruct functions after parsing using machine learning

Multiple parents: e.g., *John$_1$ walks and (John$_1$) talks to Mary*

Other non-context-free challenges: unsupervised

Free word-order: Separate surface from canonical structure by reordering.

But: Information on canonical order is not in treebanks.

Some order variation is pragmatic, some affects syntax/semantics.

Other non-context-free challenges: unsupervised

Free word-order: Separate surface from canonical structure by reordering.
But: Information on canonical order is not in treebanks.
Some order variation is pragmatic, some affects syntax/semantics.

Condition on lexical relations as opposed to just structural relations.
But: which relations are relevant?

Other non-context-free challenges: unsupervised

Free word-order: Separate surface from canonical structure by reordering.
But: Information on canonical order is not in treebanks.
Some order variation is pragmatic, some affects syntax/semantics.

Condition on lexical relations as opposed to just structural relations.
But: which relations are relevant?

Exploit sentence context; e.g., current topic, other discourse effects

THE END

Codes: http://github.com/andreasvc/disco-dop

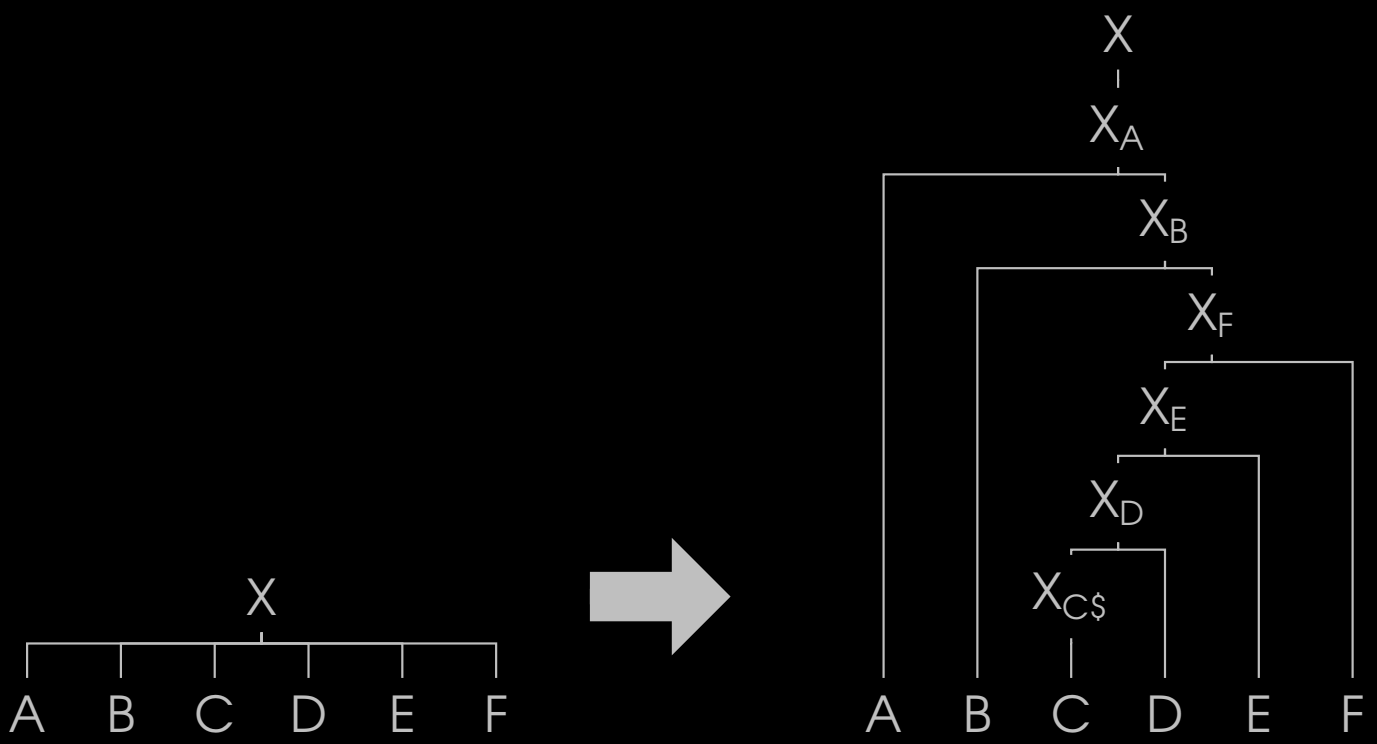
Papers: http://staff.science.uva.nl/~acranenb

Wait . . . there's more

BACKUP SLIDES

Binarization

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



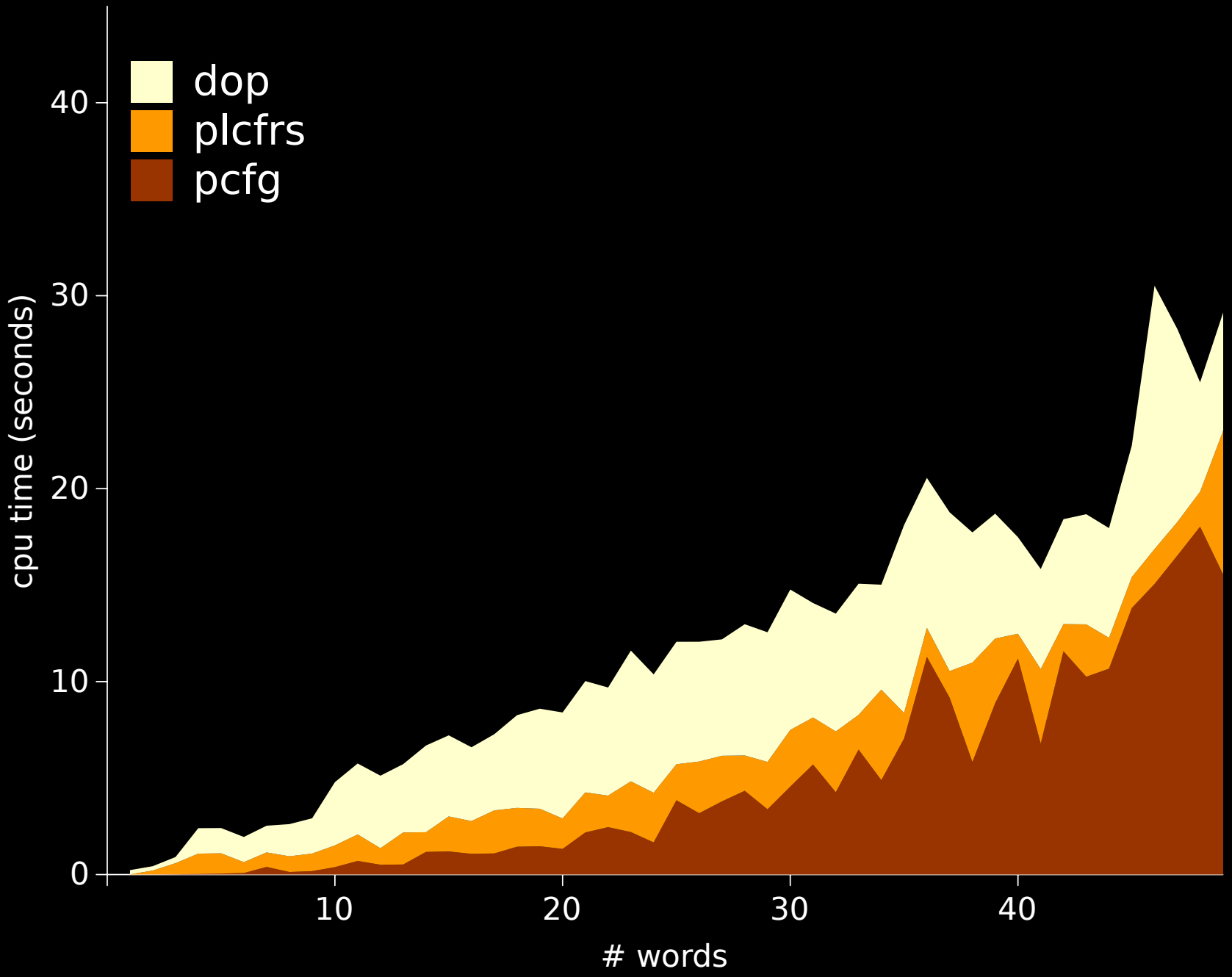
Klein & Manning (2003): Accurate unlexicalized parsing.

Implementation details

- Cython: combines best of both worlds
  C speed, Python convenience.
- Where it matters, manual memory management & layout;
- e.g., grammar rules & edges compactly packed in arrays of `structs`.
- 14k lines of code; FWIW:

Collins parser	C	3k	(!?)
bitpar	C++	6k	
Berkeley parser	Java	58k	
Charniak & Johnson parser	C++	62k	
Stanford parser	Java	151k	

Efficiency (Negra dev set)



Parser setup

```
traincorpus='wsj02-21.export',
testcorpus='wsj24.export',
corpusdir='../../dptb',
stages=[
    dict(
        name='pcfg', mode='pcfg',
        split=True, markorigin=True,
    ),
    dict(
        name='plcfrs', mode='plcfrs',
        prune=True, splitprune=True, k=10000,
    ),
    dict(
        name='dop', mode='plcfrs',
        prune=True, k=5000,
        dop=True, usedoubledop=True, m=10000,
        estimator='dop1', objective='mpp',
    ),
],
[...]
```

Web-based interface