Data-oriented Parsing

Jelle Zuidema
ILLC, Universiteit van Amsterdam

Guestlecture Elements of NLP, 2013

Plan

1980s

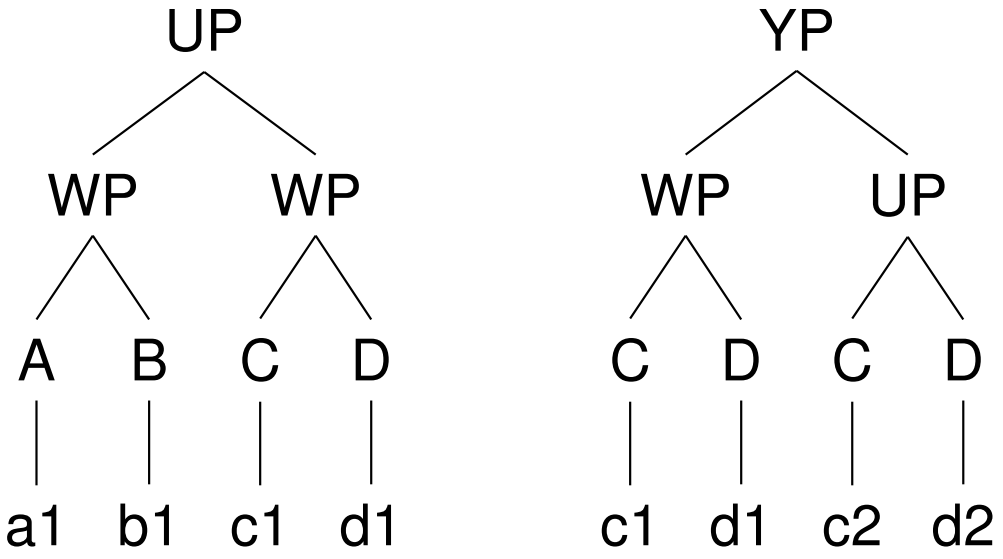
- Algorithmic problem of parsing with context-free grammars considered solved: CKY and Earley algorithms (dynamic programming)
- Handwritten grammars: struggling with coverage and ambiguity
- Theoretical linguistics: natural language syntax necessitates trans-contextfree formalisms (Huybrechts, 1984; Shieber, 1985; Joshi, 1985)
- Language acquisition & comparative linguistics: constructions / multi-word expressions can be building blocks
- Computational linguistics: effective natural language processing (translation, speech recognition, syntax?) requires (corpus) statistics

Scha 1990

Data-oriented Parsing (Scha, 1990)

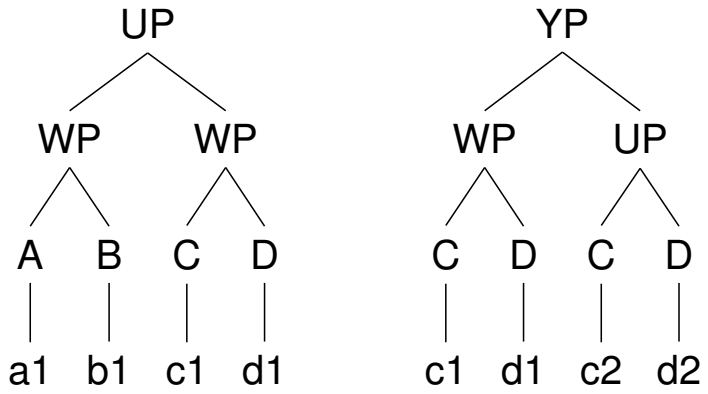
- Use corpus statistics for syntactic disambiguation (Lari & Young, 1990), instead of cleverly selected (semantic, syntactic, pragmatic) features
- Parse with fragments from a corpus (“treebank grammars”, Charniak, 1996), instead of handwritten grammars
- Use fragments of arbitrary size (“all-subtrees approach”, Collins & Duffy, 2002), instead of the minimal contextfree rewrite rules

Scha 1990

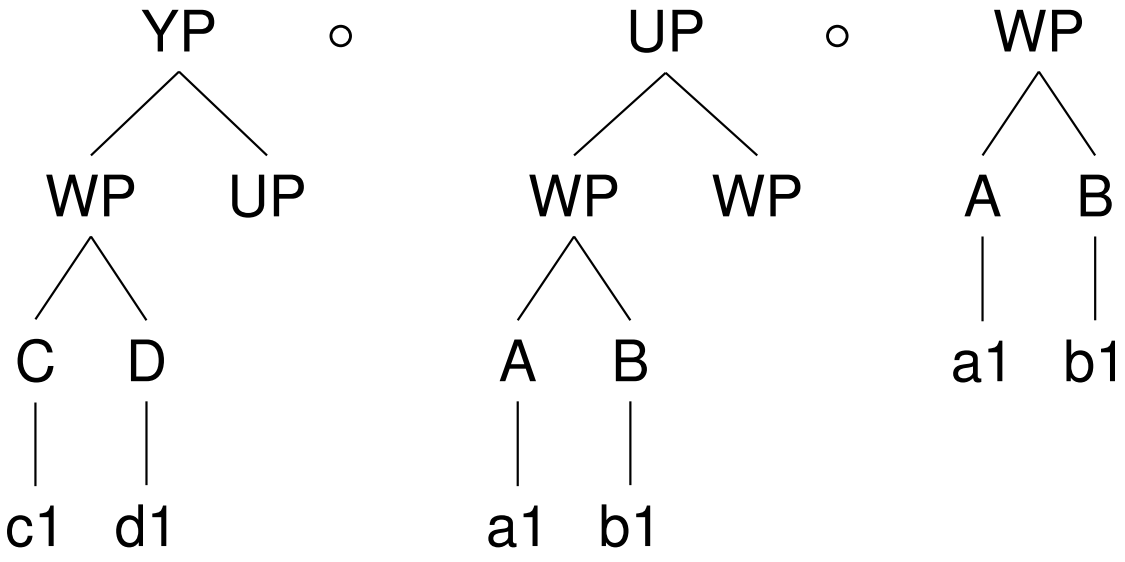


c1 d1 a1 b1 a1 b1

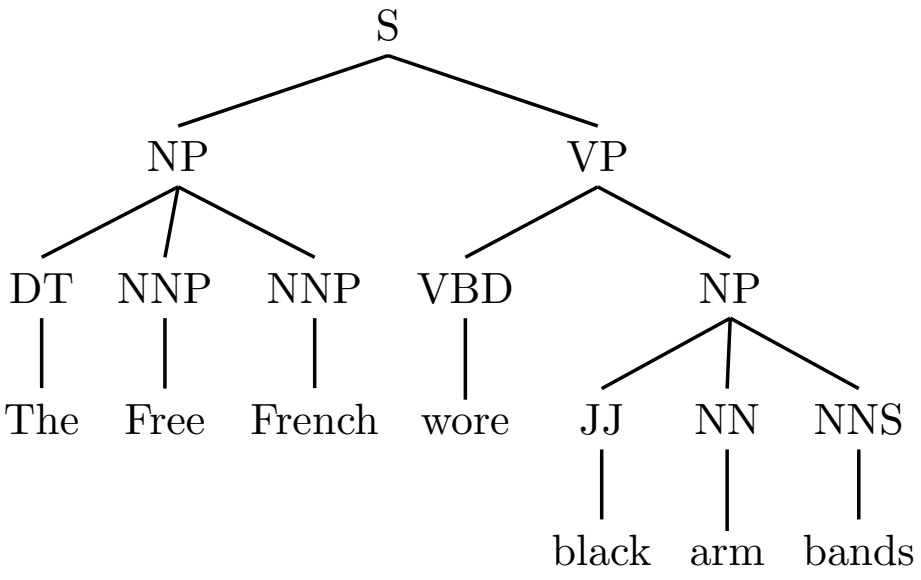
Scha 1990



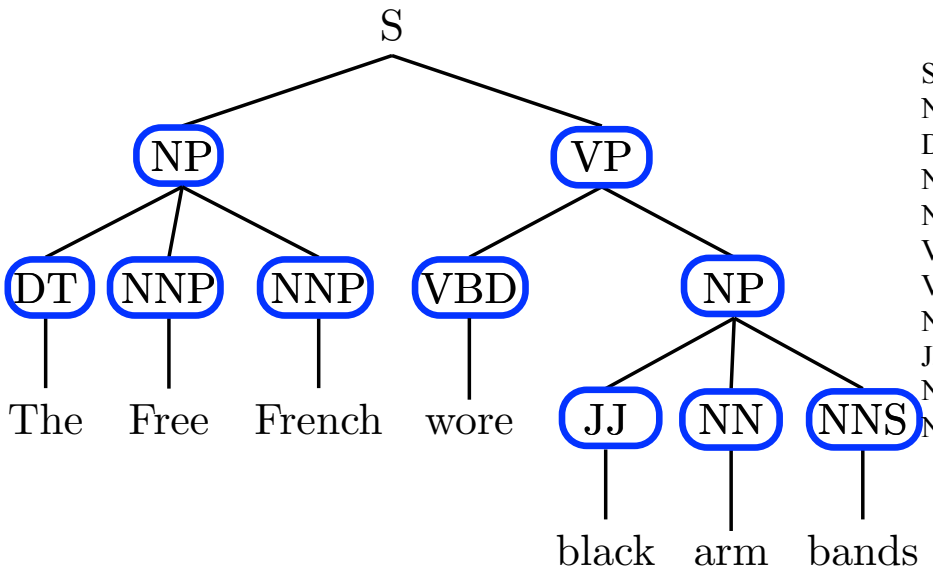
c1 d1 a1 b1 a1 b1



Phrase Structures

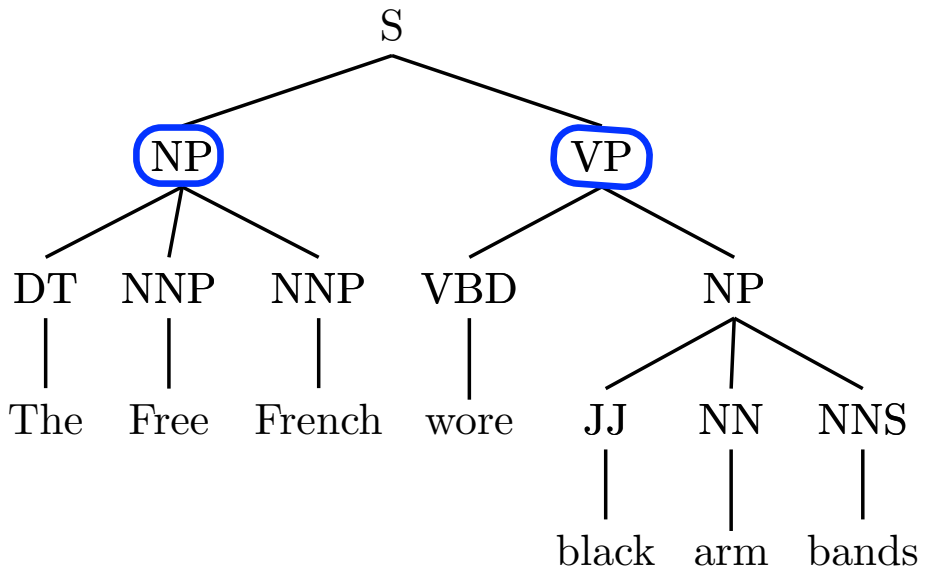


Context Free Grammar (CFG)

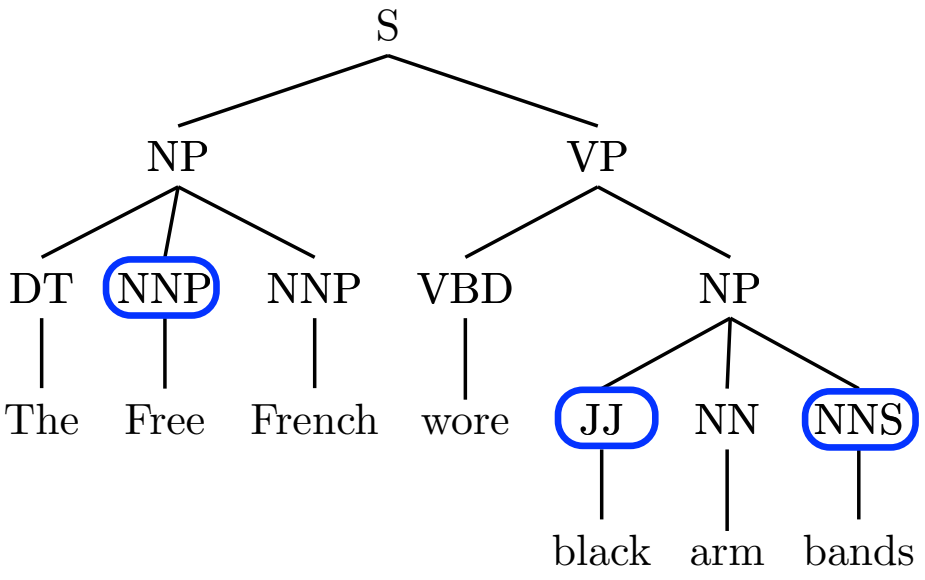


S → NP VP
NP → DT NNP NNP
DT → The
NNP → Free
NNP → French
VP → VBD NP
VBD → wore
NP → JJ NN NNS
JJ → black
NN → arm
NNS → bands

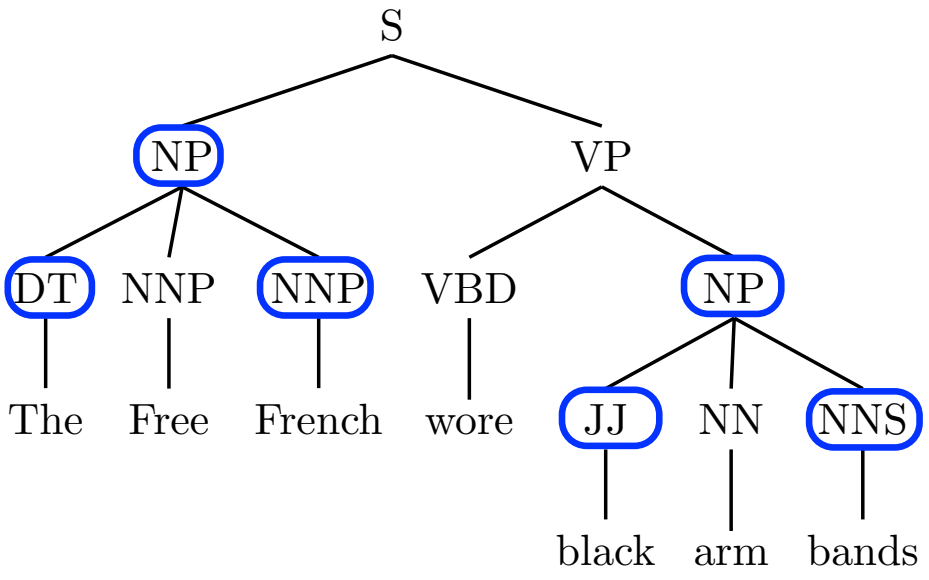
Data Oriented Parsing (DOP)



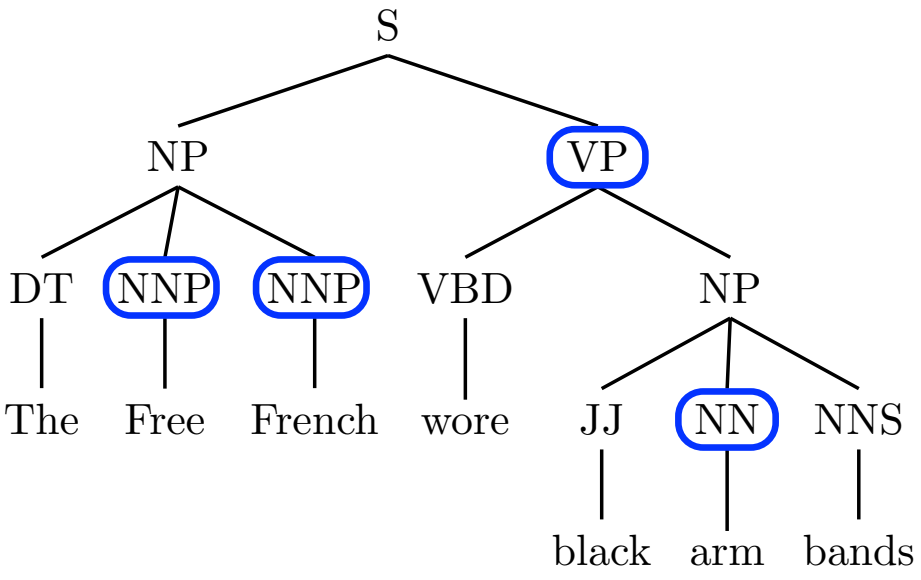
Data Oriented Parsing (DOP)



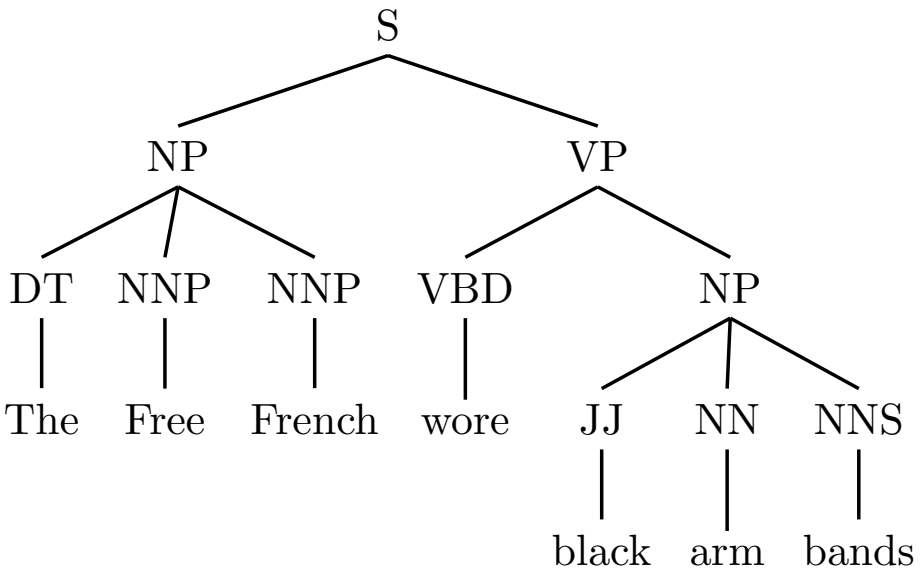
Data Oriented Parsing (DOP)



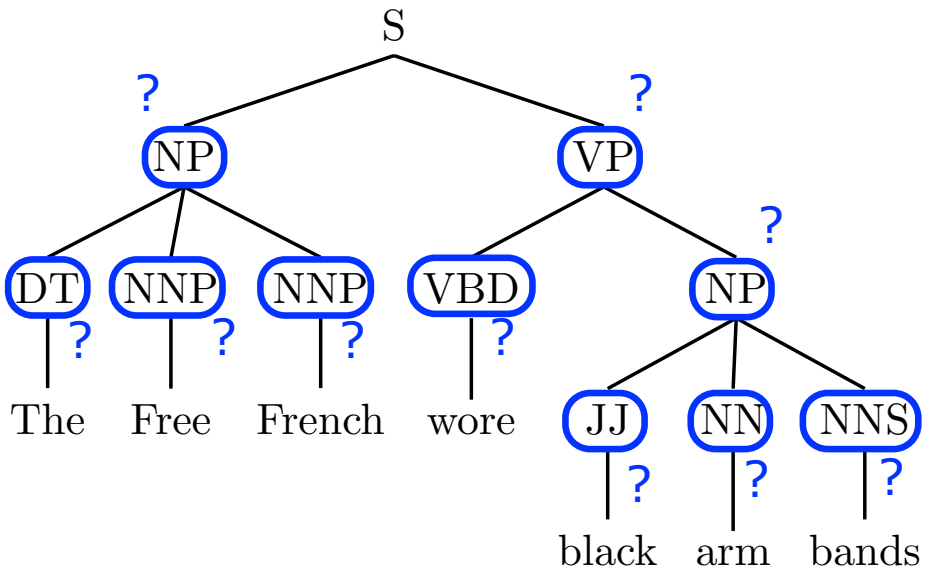
Data Oriented Parsing (DOP)



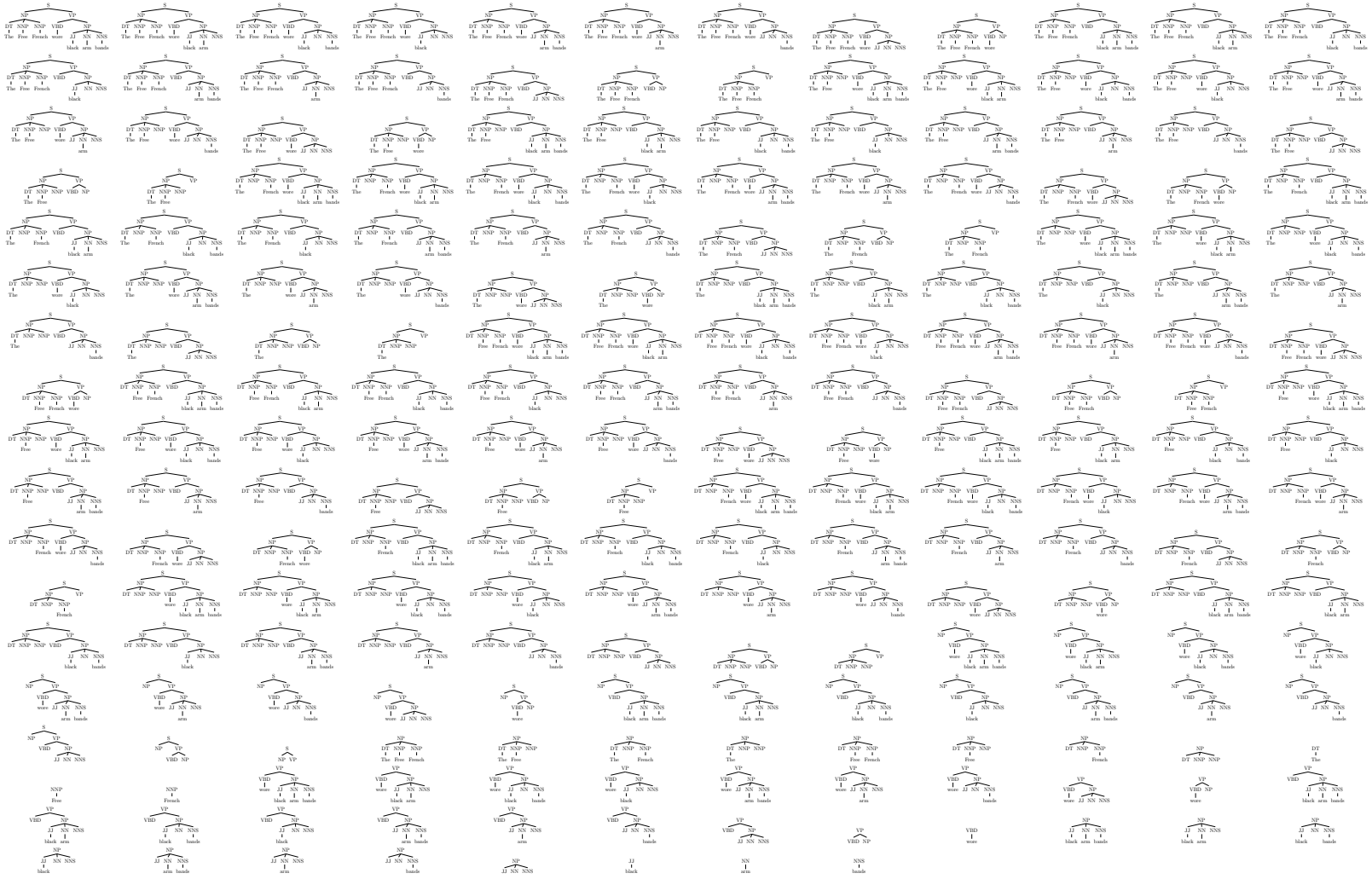
Data Oriented Parsing (DOP)



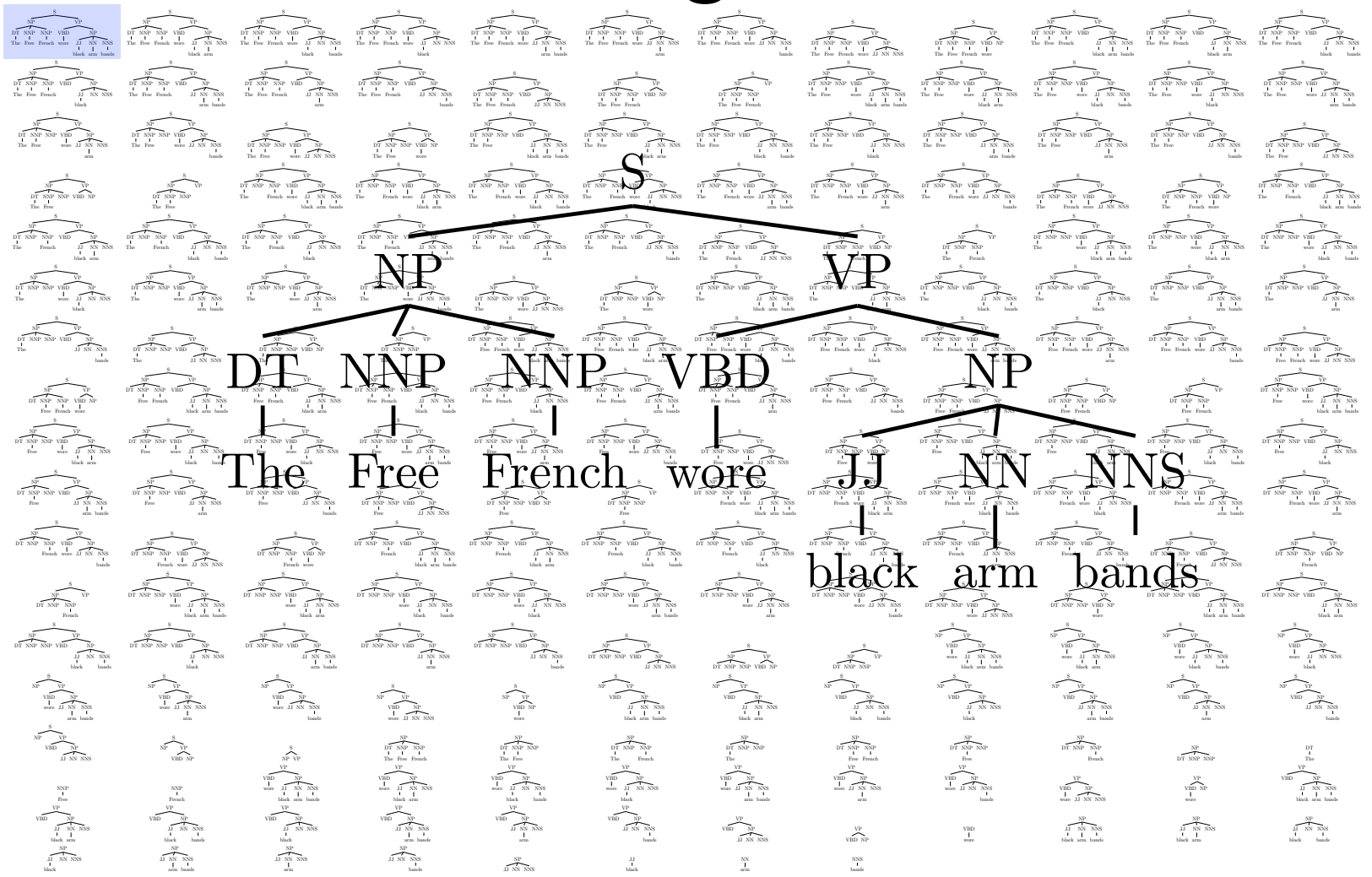
Data Oriented Parsing (DOP)



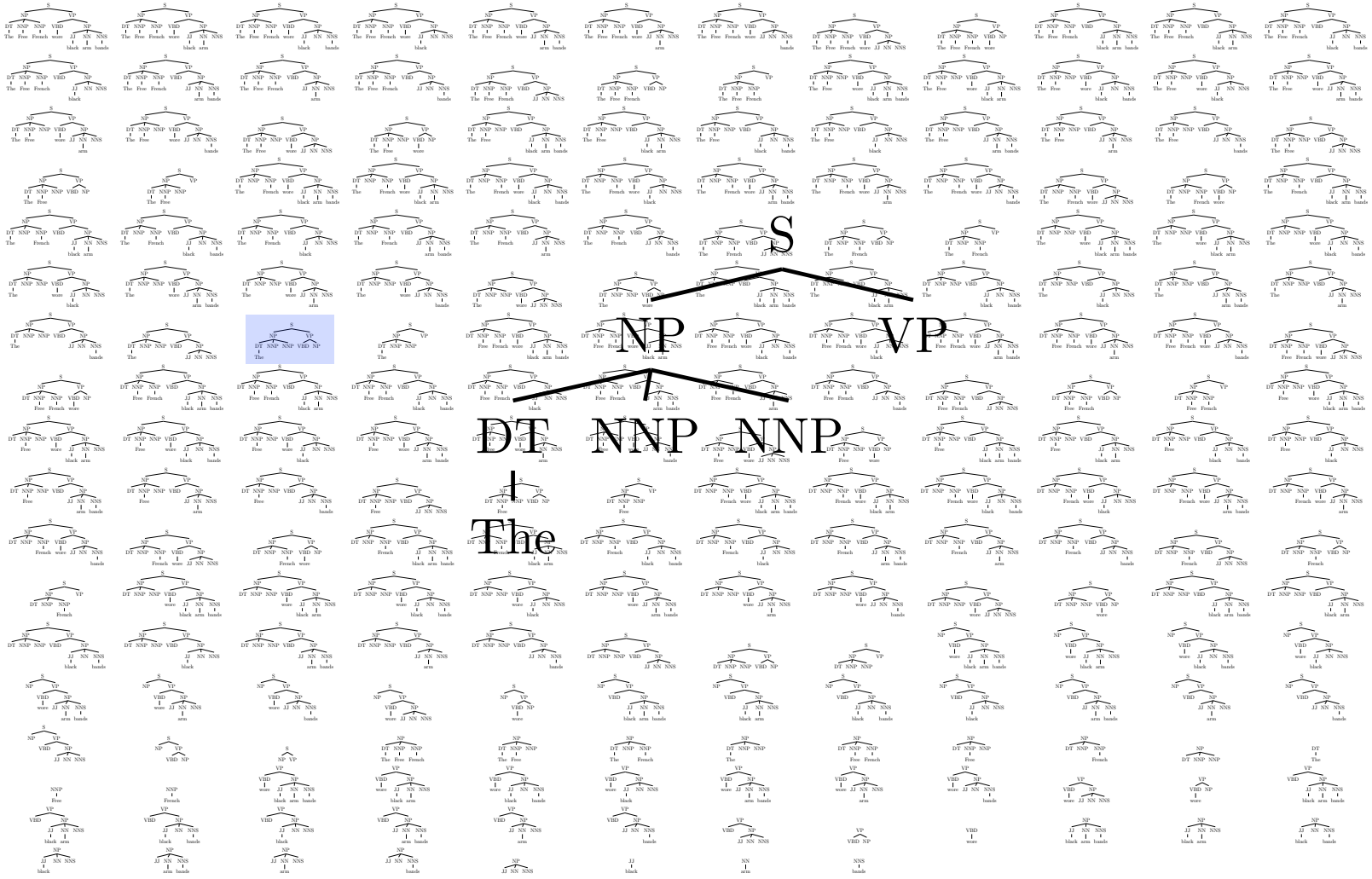
212 fragments



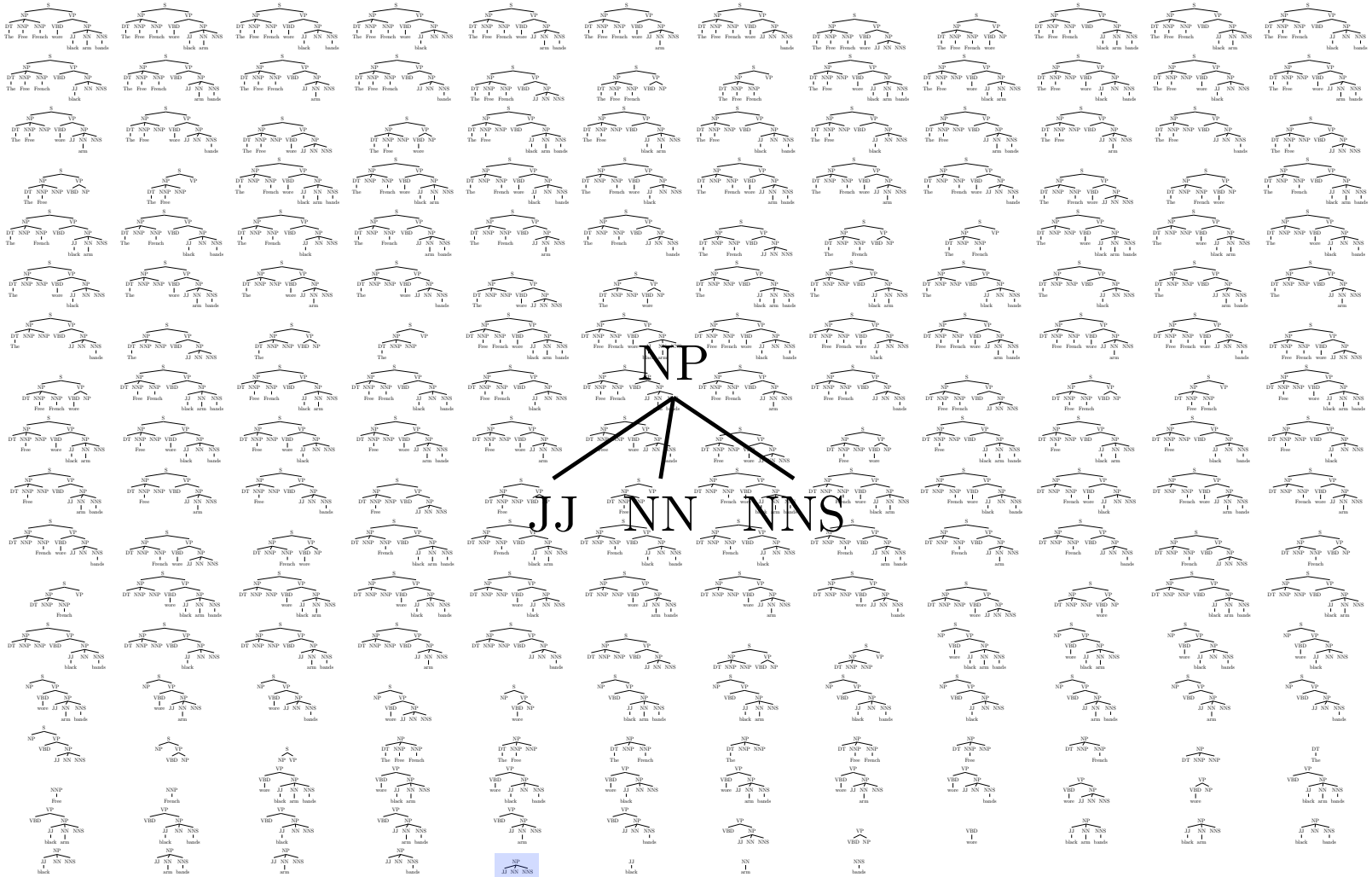
212 fragments



212 fragments



212 fragments



Tree Substitution Grammars

Definition

A **Tree Substition Grammar** (TSG) is a 4-tuple $\langle V_n, V_t, S, T \rangle$, where

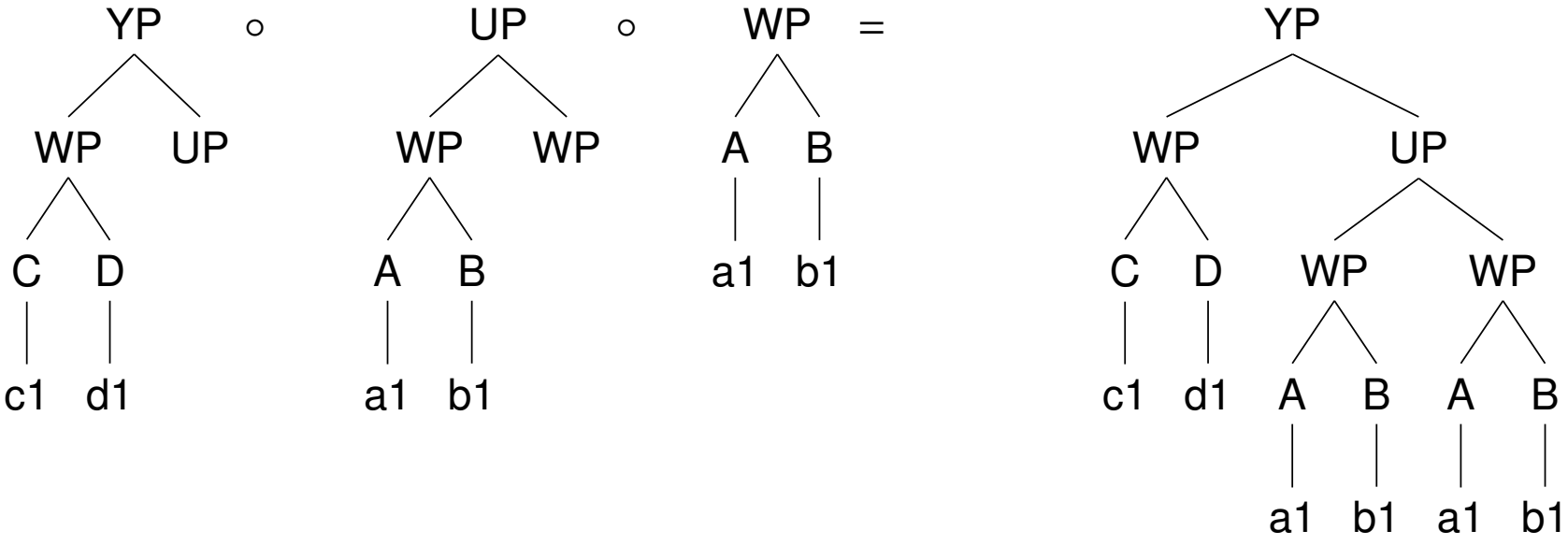
- V_n is the finite set of non-terminal symbols;
- V_t is the finite set of terminal symbols;
- $S \in V_n$ is the start symbol;
- T is a finite set of elementary trees, such that for every $\tau \in T$
  - the unique root node $r(\tau) \in V_n$,
  - the (possibly empty) set of internal nodes $i(\tau) \subseteq V_n$ and
  - the set of leaf nodes $l(\tau) \subseteq V_n \cup V_t$.

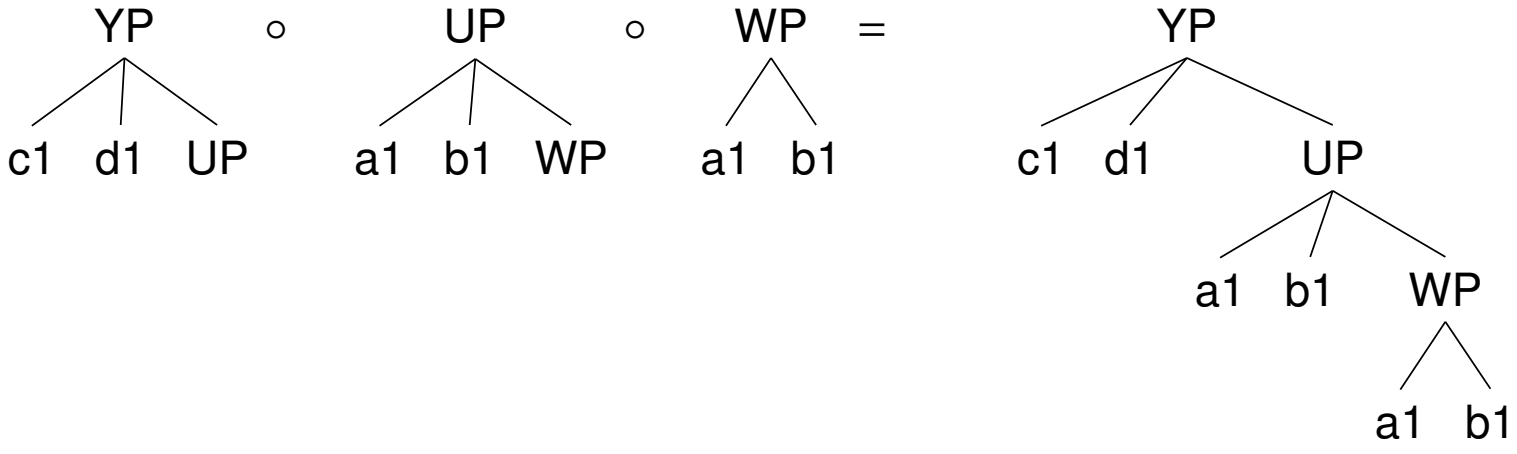
Tree Substitution Grammars (ctd)

Theorem

(Joshi & Schabes, 1991) TSGs have the same weak generative capacity as context-free grammars.

We can replace every elementary tree τ by a context-free rewrite rule that directly rewrites the root to the yield: $r(\tau) \mapsto y(\tau)$.


YP
WP
UP
C
D
c1
d1
UP
WP
WP
A
B
a1
b1
WP
A
B
a1
b1
YP
WP
UP
C
D
WP
WP
c1
d1
A
B
A
B
a1
b1
a1
b1

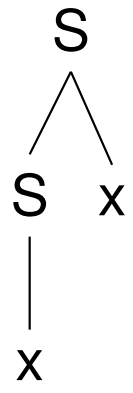

YP
c1
d1
UP
UP
a1
b1
WP
WP
a1
b1
YP
c1
d1
UP
a1
b1
WP
a1
b1

Tree Substitution Grammars (ctd)

Lemma

TSGs can generate more tree languages that CFGs, i.e. the strong generative capacity of TSGs is larger than that of CFGs.

E.g., consider a TSG consisting of a single elementary tree:



This grammar generates only this particular tree, but no possible CFG can generate the exact same tree and no other.

Tree Substitution Grammars (ctd)

- The string language of a TSG *G* can be recognized in a time polynomial (in fact, cubic) in the length of the sentences.
- we can use standard CFG parsing techniques to find all derivations of a given string licensed by *G*.
- however, for finding the set of unique *derived trees* licensed by *G*, we need to collapse derivations that yield the same derived tree.
- if *G* is a treebank grammar consisting only of subtrees of trees in treebank, the set of *derived trees* will always be a subset of the set of parses licensed by the treebank CFG.

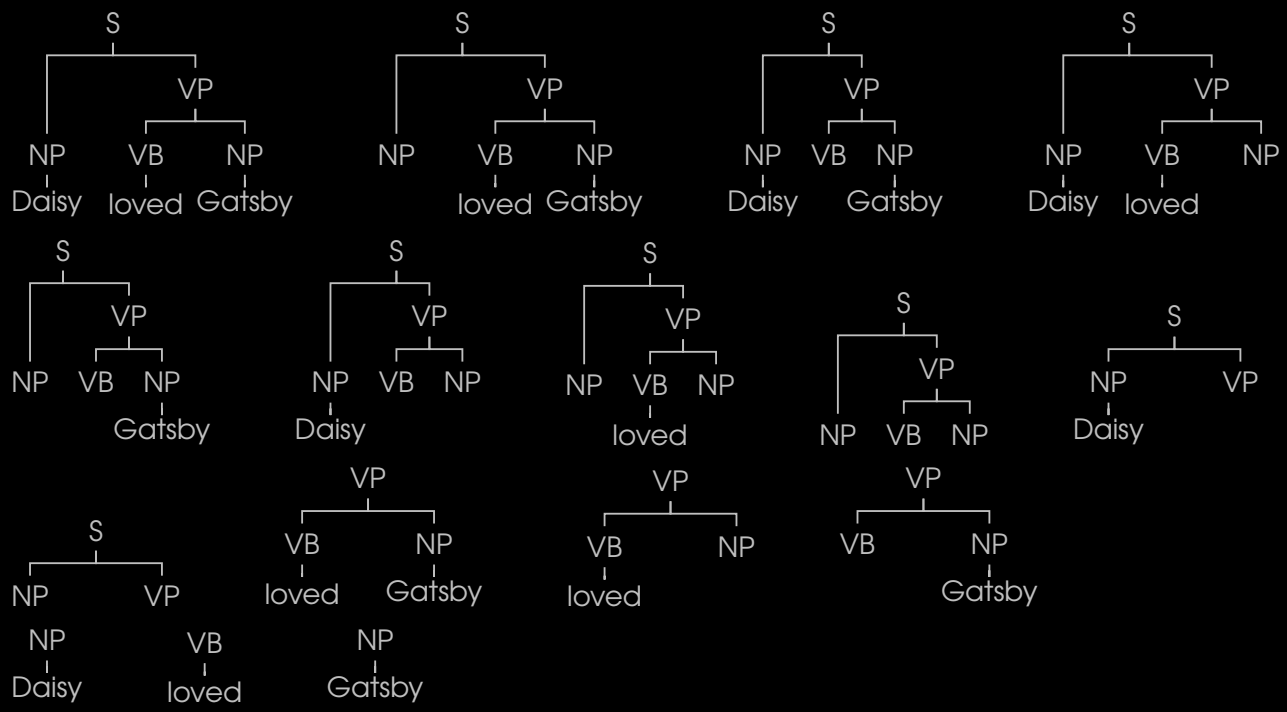
DOP1

- First DOP-implementation presented by Bod (1992)
- Extract all unique subtrees up to depth d from a corpus C: bag of subtrees T
- For every unique subtree t (upto depth d) in C, create an elementary tree τ with weight:

$$w(\tau) = \frac{\text{COUNT}(\tau, C)}{\sum_{\text{ROOT}(\tau)=\text{ROOT}(\tau')} \text{COUNT}(\tau', C)}$$

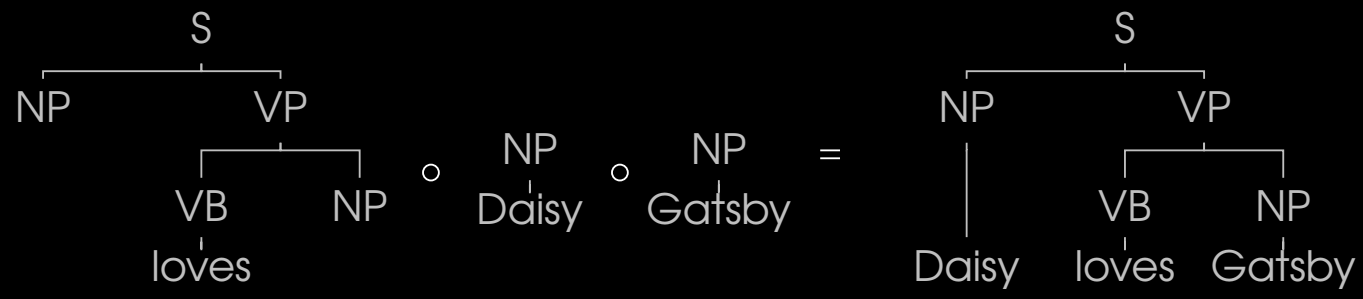
- Probabilistic Tree Substitution Grammar

DOP1 fragments



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

DOP1 derivation



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

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

Probabilistic Tree Substitution Grammar

A PTSG is a 5-tuple $\langle V_n, V_t, S, T, w \rangle$, where

- V_n is the set of non-terminal symbols;
- V_t is the set of terminal symbols; $S \in V_n$ is the start symbol;
- T is a set of elementary trees, such that for every $\tau \in T$ the unique root node $r(\tau) \in V_n$, the (possibly empty) set of internal nodes $i(\tau) \subset V_n$ and the set of leaf nodes $l(\tau) \subset V_n \cup V_t$;
- $w : T \rightarrow [0, 1]$ is a probability (weight) distribution over the elementary trees, such that for any $\tau \in T$, $\sum_{\tau' \in R(\tau)} w(\tau') = 1$, where $R(\tau)$ is the set of elementary trees with the same root label as τ.

The probability of a derivation d is defined as the product of weights of the elementary trees involved:

$$P(d =< \tau_1, \ldots, \tau_n >) = \prod_{i=1}^{n} (w(\tau_i)).$$

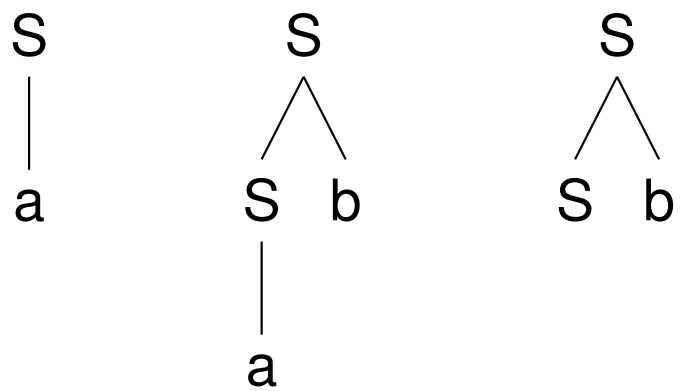
Multiple derivations can yield the same parse tree; the probability of a parse tree p equals the sum of the probabilities of the different derivations that yield that same tree:

$$P(p) = \sum_{d:t(d)=p} (P(d)).$$

Theorem

(Bod, 1998) TSGs are stochastically richer than PCFGs, even for PCFGs that generate the same tree languages.

E.g., consider a PTSG with the following elementary trees:



where all elementary trees assign an equal weight of 1/3.

Proof The only PCFG that generates the same trees, is one that has the rules $S \rightarrow a$ and $S \rightarrow Sb$. It must assign a weight of $1/3$ to the first rule, and hence $2/3$ to the second. But then the PCFG generates the second tree with a probability of $2/3 \times 1/3 = 2/9$, whereas the PTSG generates it with probability $1/3 + 1/3 \times 1/3 = 4/9$. I.e., there is no choice of weights for this PCFG that will generate trees with the same probabilities as the PTSG.

Empirical Results

Air Travel Information System (ATIS) corpus

depth of corpus-subtrees	parse accuracy	
	most probable parse	most probable derivation
1	47%	47%
≤2	68%	56%
≤3	79%	64%
≤4	83%	67%
≤5	84%	67%
≤6	84%	69%
unbounded	85%	69%

- Include all dependencies and “let the statistics decide”
- The “DOP hypothesis”: including larger fragments always improves accuracy

Finding Most Probable Parse is NP-Hard

- the problem of calculating tree probabilities under the PTSG model cannot be solved using standard PCFG techniques.
- If we use PCFG parsing techniques to find all derivations of sentences (analogous to the CFG-conversion) we need to sum the probabilities of exponentially many derivations of each distinct derived tree.

Theorem

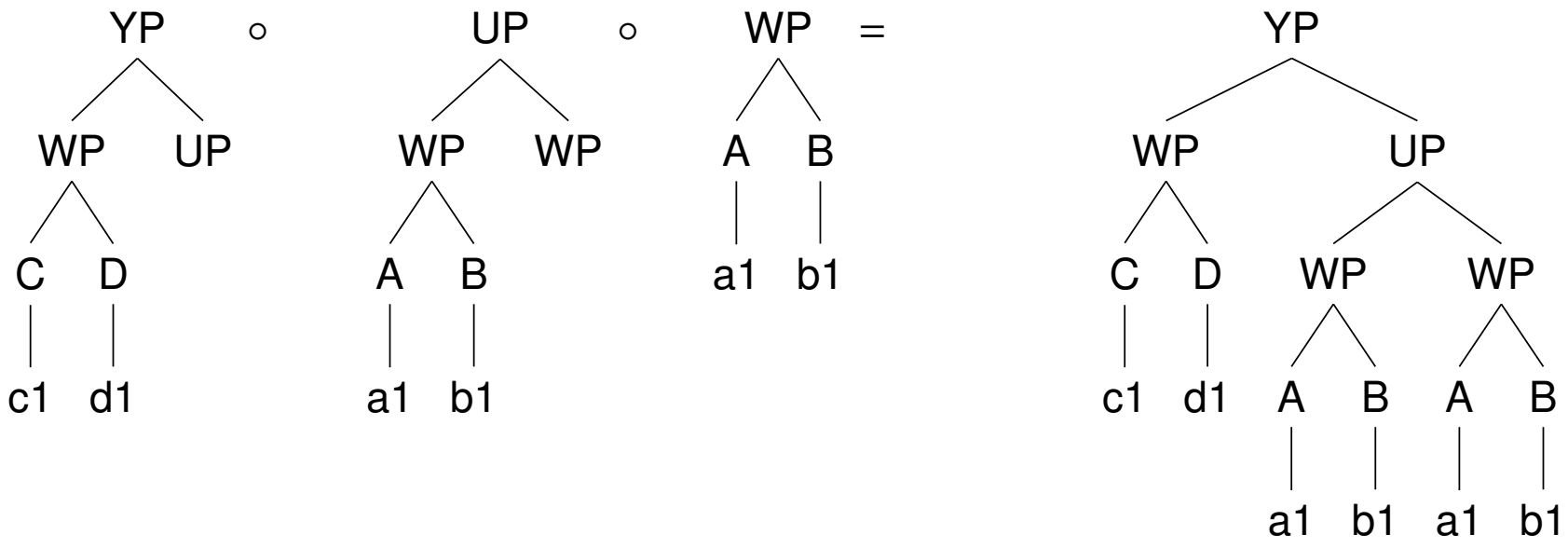
(Sima'an, 1998, 2002) Finding the most probable parse under the unrestricted PTSG model is NP hard.

Solution

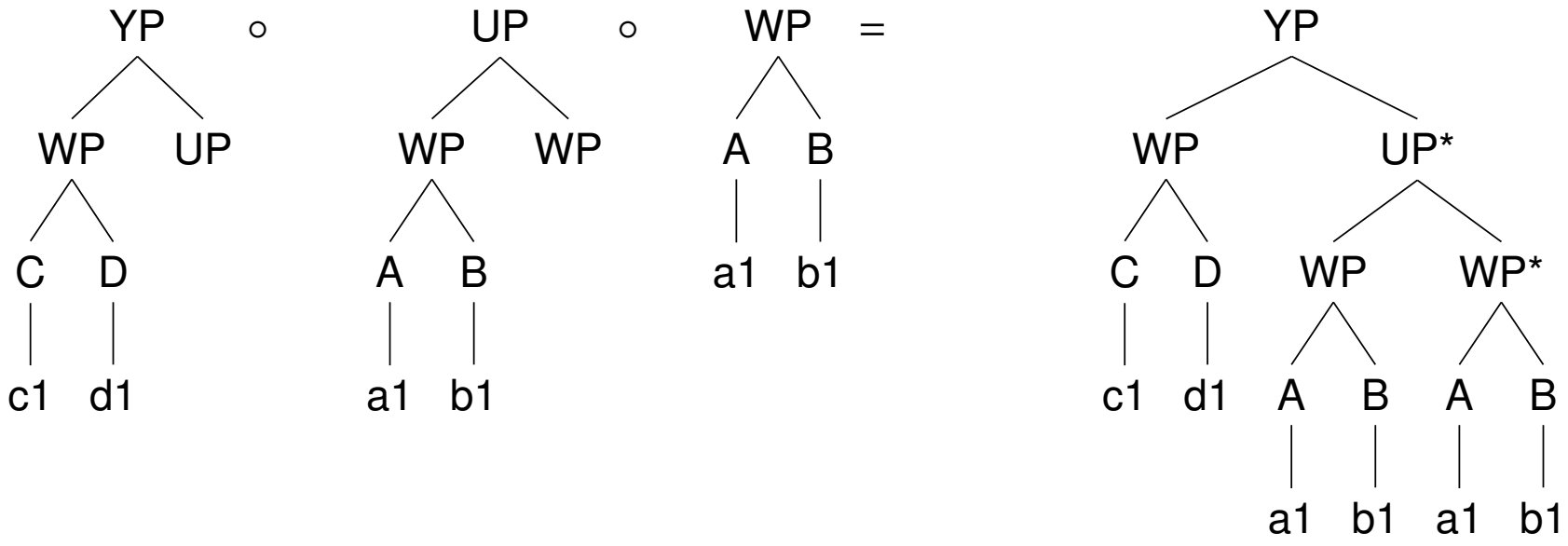
Most current models:

- either approximate the most probable parse based on the 100- or 1000-best derivations;
- or use a different objective, such as the parse with maximum expected number of correct constituents (MCP).

Exponential Growth



Exponential Growth



Exponential Growth

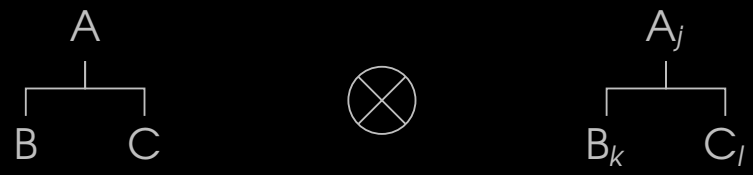
- Number of subtrees of a a tree t equals $2^{i(t)}$
- Total number of subtrees of the Penn WSJ treebank estimated at 10^{48}

Solution

- Principled restriction on subtrees considered: Parsimonious DOP & Double-DOP (Zuidema, 2007; Sangati & Zuidema, 2011)
- Goodman's (1998,2003) reduction to PCFG

Reduction to PCFG

- Treebank refinement: take non-terminal and split according to contexts
- In the limit: each non-terminal becomes a particular occurrence in a tree



$A_j \rightarrow B\ C$	$(1/a_j)$	$A \rightarrow B\ C$	$(1/(a\bar{a}))$
$A_j \rightarrow B_k\ C$	(b_k/a_j)	$A \rightarrow B_k\ C$	$(b_k/(a\bar{a}))$
$A_j \rightarrow B\ C_l$	(c_l/a_j)	$A \rightarrow B\ C_l$	$(c_l/(a\bar{a}))$
$A_j \rightarrow B_k\ C_l$	$(b_k c_l/a_j)$	$A \rightarrow B_k\ C_l$	$(b_k c_l/(a\bar{a}))$

- Polynomial time parsing.
- (Exact) disambiguation still NP-hard.
  Goodman (2003): Efficient parsing of DOP with PCFG-reductions

Bias & Inconsistency

Johnson (2002) showed that the DOP1 estimator is biased and inconsistent:

- Given a treebank of size n sampled from an arbitrary PTSG G
- the mean of weight distribution induced by DOP1 method is not equal to the true distribution G ("bias");
- the DOP1 method is not guaranteed to converge to the true distribution in the limit of $n \rightarrow \infty$ ("inconsistency").

These properties are undesirable from a (frequentist) statistical estimation perspective, but less relevant in practice although they might be diagnostic for a serious empirical problem:

- Because a parse tree of size n has a number of subtrees exponential in n, subtrees from the largest tree(s) in a corpus dominate the probability calculations.

Solution

Alternative estimators

- Equal weights estimate (Goodman, 2003)
- Backoff-DOP (Sima'an & Buratto, 2003)
- DOP* (Zollmann, 2004)
- Push-n-pull (Zuidema, 2007)

Which fragments to extract?

1. All (Goodman reduction, Goodman 1996, Bod 2003, Bansal and Klein 2010)
2. A subset
   - restriction on depth (Bod, 1998)
   - random sample (Bod, 2001)
   - only fragments with 1 word (Sangati and Zuidema, 2009)
   - ....

- R. Bod. Beyond Grammar: An Experience-Based Theory of Language. CSLI, Stanford, CA., 1998.
- R. Bod. A Computational Model Of Language Performance: Data Oriented Parsing. COLING 1992.
- J. Goodman. Efficient algorithms for parsing the DOP model. 1996.
- R. Bod. What is the minimal set of fragments that achieves maximal parse accuracy? ACL 2001.
- R. Bod. An efficient implementation of a new DOP model. EACL 2003.
- W. Zuidema. What are the productive units of natural language grammar?: a DOP approach to the automatic identification of constructions. CoNLL-X 2006
- F. Sangati and W. Zuidema. Unsupervised Methods for Head Assignments. EACL 2009.
- M. Bansal and D. Klein. Simple, accurate parsing with an all-fragments grammar. ACL 2010.

Seeking Recurring Fragments

- Criterion: a syntactic construction is linguistically relevant if there is some empirical evidence about its **reusability** in a representative corpus of language productions.
- Use only the fragments that recur several times in the treebank, i.e. $\tau \mid \exists\ t_i, t_j,\ i \neq j,\ \tau \in t_i \wedge \tau \in t_j$

Fragment Seeker

(Sangati et al., 2010)

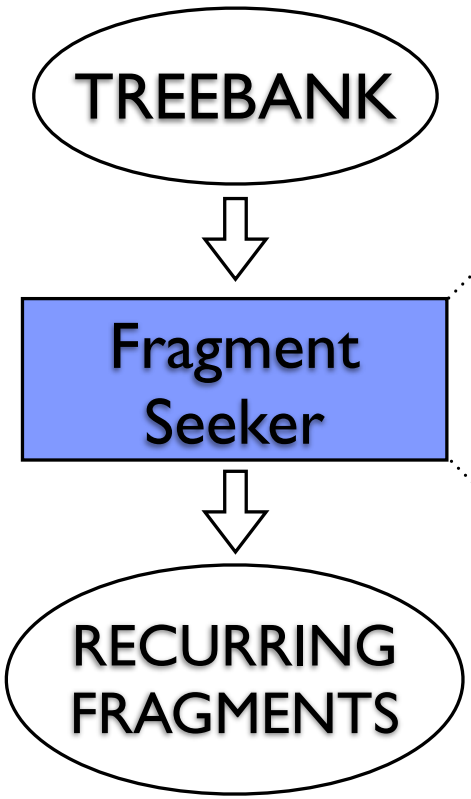
- Based on **Tree Kernels** (Collins and Duffy, 2001; Moschitti 2006)
- Dynamic programming
- Original idea: compute the similarity between two trees as the number of fragments they have in common.
- Current idea: we are not only interested in a number, we want to extract the shared fragments.
- Available at http://staff.science.uva.nl/~fsangati/

➡ F. Sangati and W. Zuidema and R. Bod. Efficiently Extract Recurring Tree Fragments from Large Treebanks. LREC 2010.

➡ M. Collins and N. Duffy. Convolution Kernels for Natural Language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, NIPS, pages 625–632. MIT Press, 2001.

➡ A. Moschitti. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In ECML, pages 318–329, Berlin, Germany, September 2006. Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Proceedings.

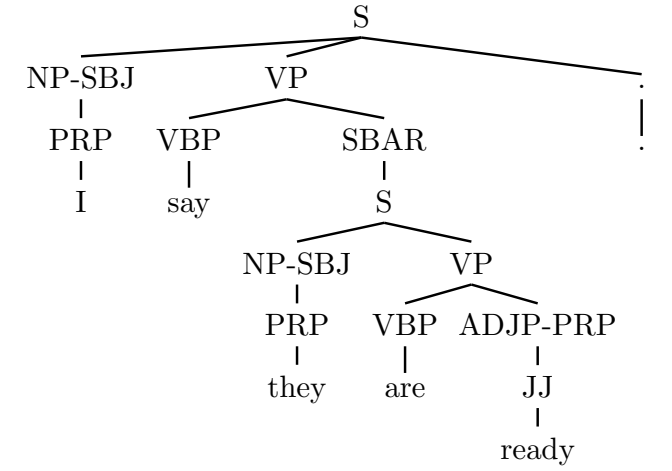
Fragment Seeker

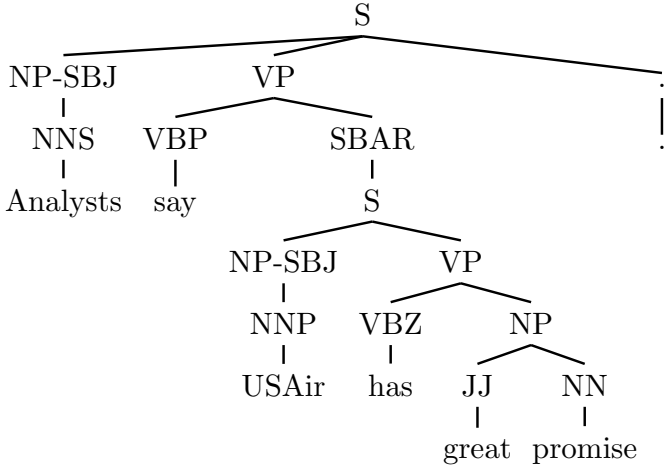


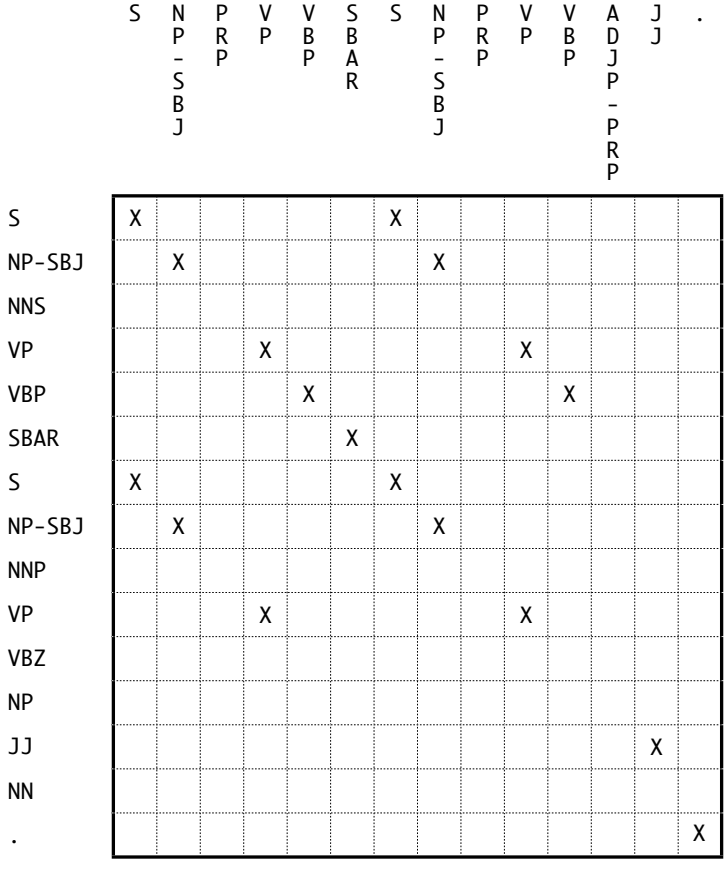
Algorithm: `ExtractFragments(T)`

Input: a corpus `T` of PS trees
Output: a set of fragments and partial fragments
begin
  `FragList`: a set of fragments;
  foreach *tree* $t_i \in T$ **do**
    foreach *tree* $t_j \in T$ *where* $t_i \neq t_j$ **do**
      foreach *node* $N_i \in t_i$ **do**
        foreach *node* $N_j \in t_j$ **do**
          `FragList.addAll(ExtractMaxFragment(`N_i, N_j`))`;
  return `FragList`;

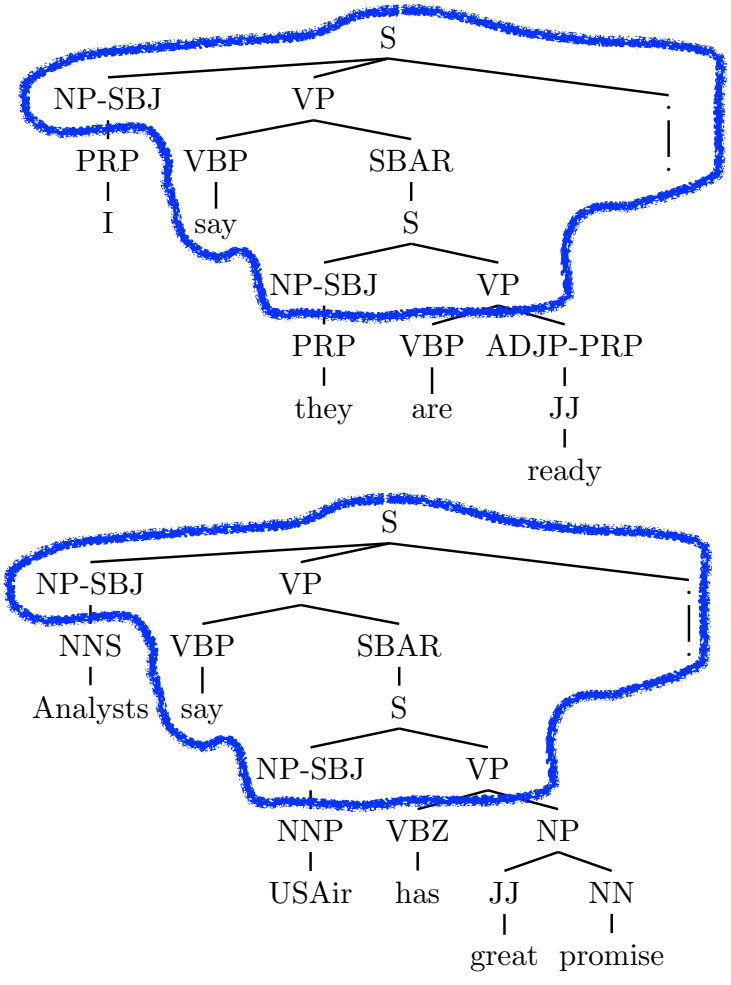
Fragment Seeker

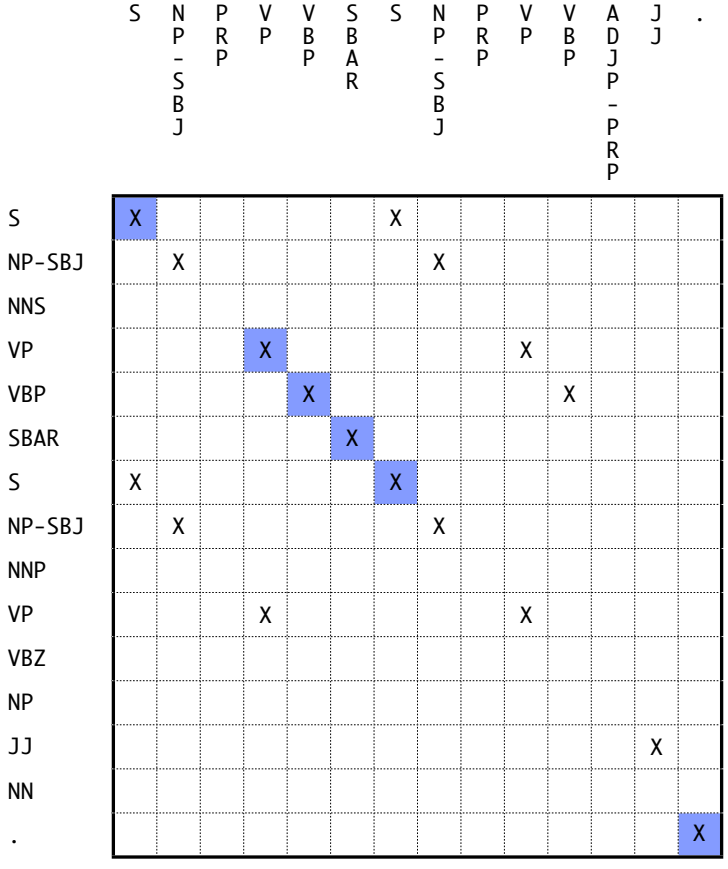




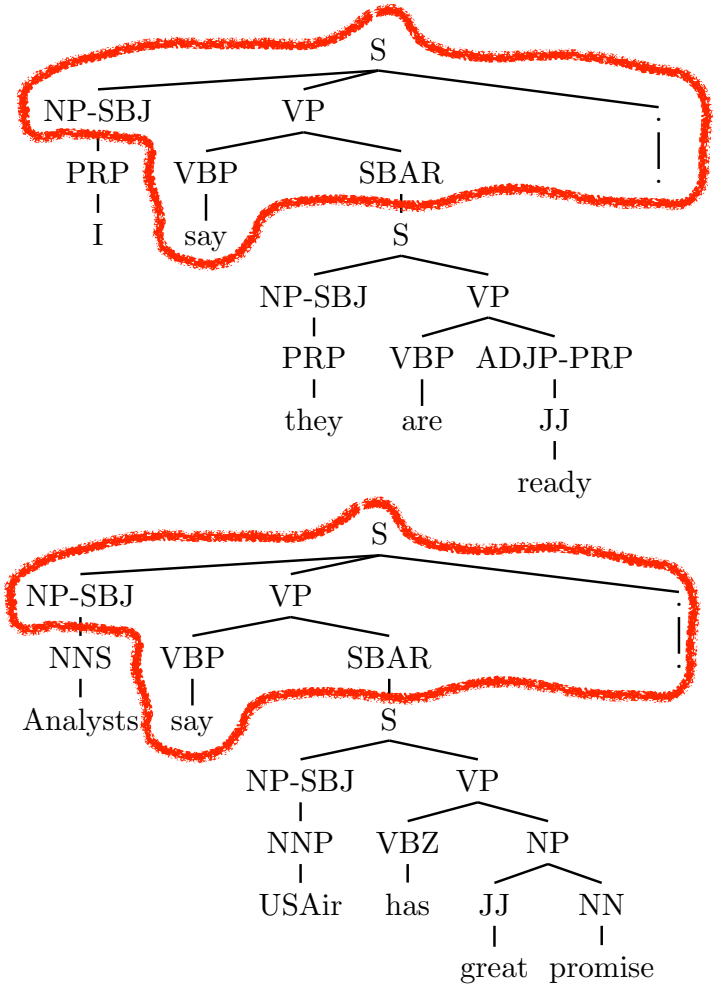


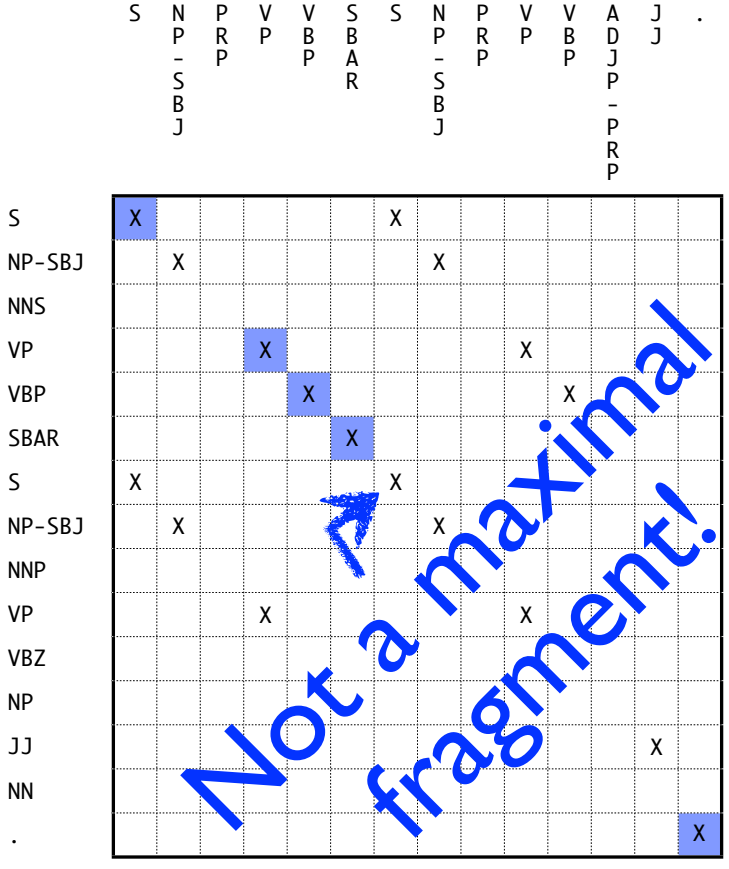
Fragment Seeker





Fragment Seeker

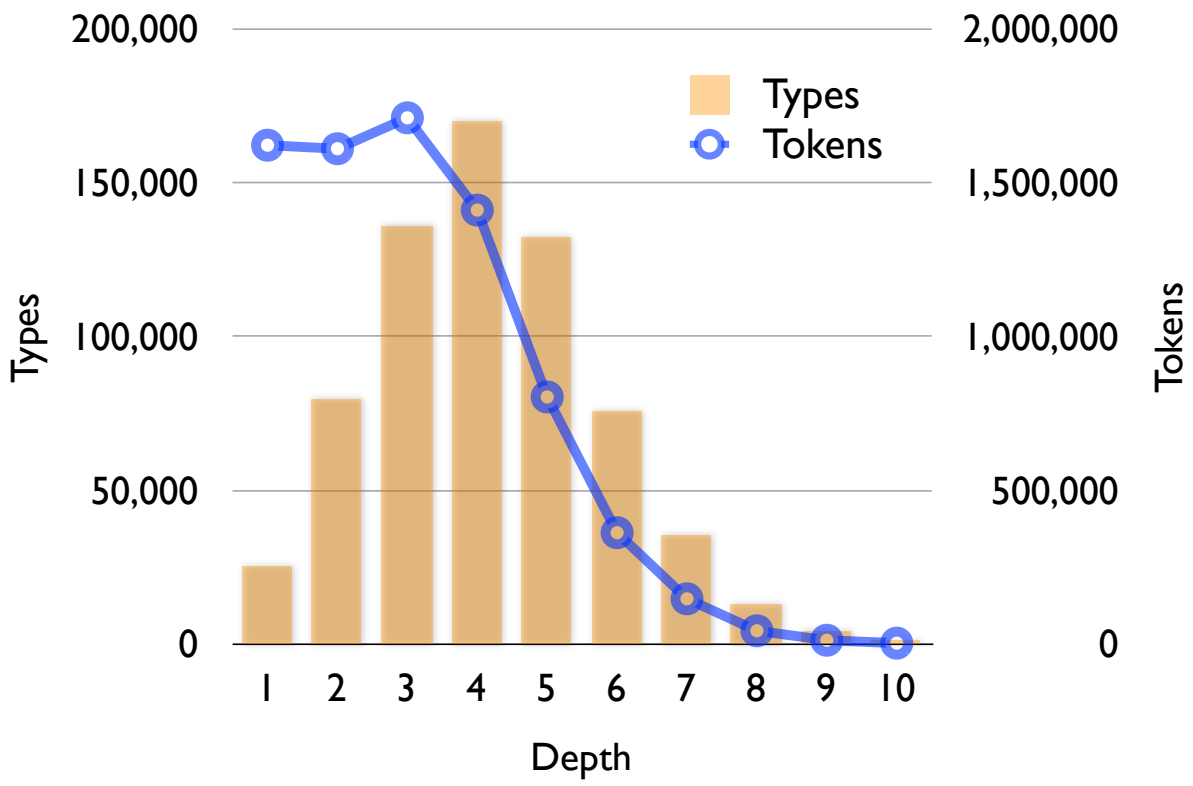




Quantitative Analysis

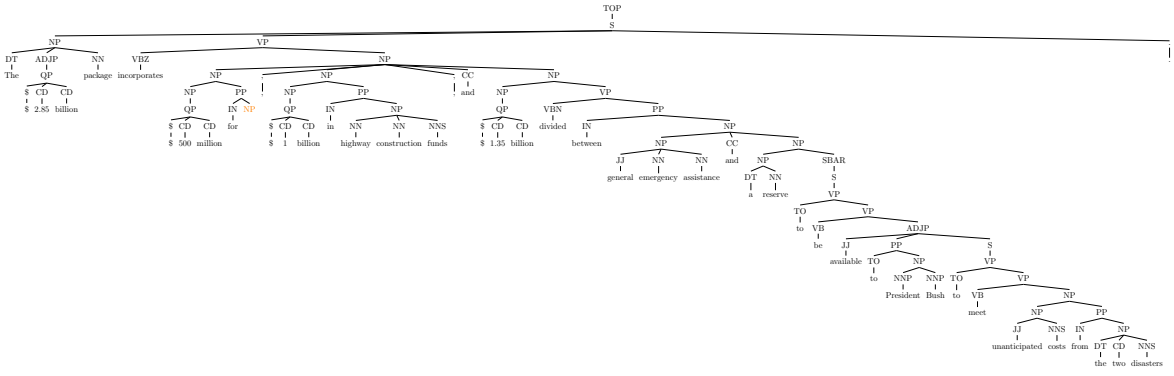
- **Treebank**: WSJ 02-21 (39,832 sentences)
- Recurring fragments types: 674,747

Depth	Types	Tokens
1	25,378	1,622,713
2	79,870	1,611,257
3	136,031	1,711,712
4	170,201	1,411,724
5	132,393	804,099
6	75,872	362,732
7	35,585	147,830
8	13,071	43,864
9	4,343	13,056
10	1,313	3,639
11	478	1,213
12	111	260
13	57	135
14	22	51
15	7	17
16	8	18
17	3	7
18	1	2
19	2	4
21	1	2
Total	674,747	7,734,335



Very Big Fragments

Depth	Types	Tokens
1	25,378	1,622,713
2	79,870	1,611,257
3	136,031	1,711,712
4	170,201	1,411,724
5	132,393	804,099
6	75,872	362,732
7	35,585	147,830
8	13,071	43,864
9	4,343	13,056
10	1,313	3,639
11	478	1,213
12	111	260
13	57	135
14	22	51
15	7	17
16	8	18
17	3	7
18	1	2
19	2	4
21	1	2
Total	674,747	7,734,335



The $ 2.85 billion package incorporates $ 500 million for NP , $ 1 billion in highway construction funds , and $ 1.35 billion divided between general emergency assistance and a reserve to be available to President Bush to meet unanticipated costs from the two disasters .

Parsing Experiments

1. Preprocess TB: unknown words, binarization, smoothing → *Berkeley* (Petrov, 2009)
2. Extracting Recurring Fragments from Treebank
3. Add unseen CFG rules
4. Estimate frequencies of fragments
5. Convert Fragments to CFG rules
6. Parse (obtain *1,000* most probable derivations) → *BITPAR* (Schmid, 2004)
7. Convert back the CFG rules to fragments
8. Post process trees (unknown words, binarization)
9. Maximize Objective (MPD, MPP, MCP)

➡ S. Petrov. Coarse-to-Fine Natural Language Processing. PhD thesis, University of California at Bekeley, 2009.

➡ H. Schmid. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In Proceedings of Coling 2004

Unknown Words

- Every word in the train and in the test occurring less than 5 times in the training set is replaced by a set of features.

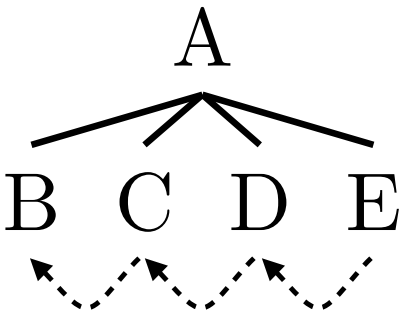
- Feature Set :
  1. `suffix`
  2. `isFirstWord`
  3. `isCapitalized`
  4. `hasDash`
  5. `hasForwardSlash`
  6. `hasDigit`
  7. `hasAlpha`

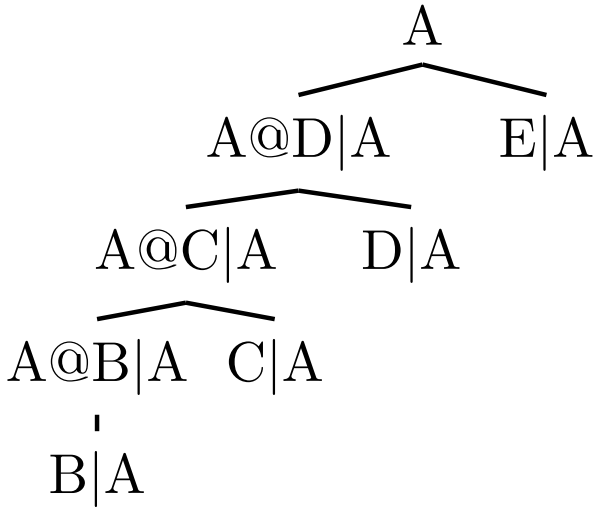
Lex Smoothing :

Low counts ($\varepsilon = 0.01$) to open-class $\langle$word, PoS-tag$\rangle$ pairs not encountered in the training corpus.

Left Binarization

(P=1,V=1)

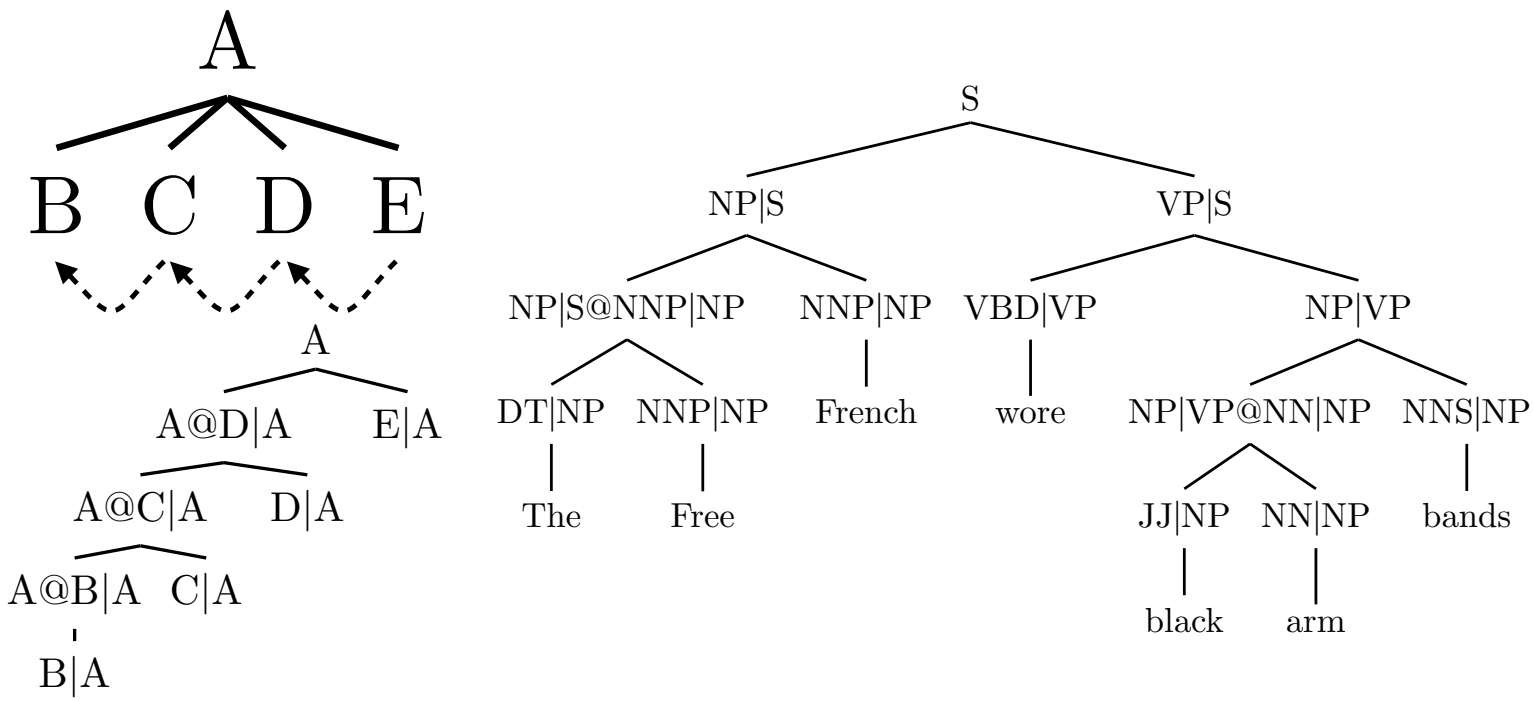




- K. Sima'an. Tree-gram parsing lexical dependencies and structural relations. ACL 2000.
- D. Klein and C. D. Manning. Accurate unlexicalized parsing. ACL 2003.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic cfg with latent annotations. ACL 2005.
- M. Bansal and D. Klein. Simple, accurate parsing with an all-fragments grammar. ACL 2010.

Left Binarization

(P=1,V=1)

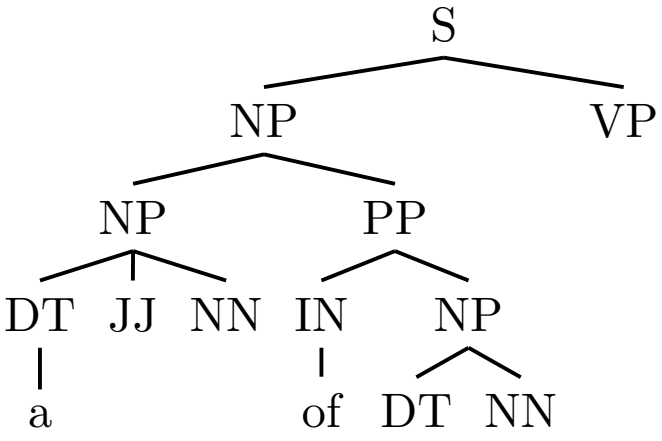


- K. Sima'an. Tree-gram parsing lexical dependencies and structural relations. ACL 2000.
- D. Klein and C. D. Manning. Accurate unlexicalized parsing. ACL 2003.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic cfg with latent annotations. ACL 2005.
- M. Bansal and D. Klein. Simple, accurate parsing with an all-fragments grammar. ACL 2010.

Fragment Extraction

- Preprocessed Treebank: WSJ 02-21 (39,832 sent.)
- Recurring fragments: 1,029,342
- Additional Unseen CFG rules: 17,768 (total 40,613)
- Additional smoothing [unseen ⟨word, PoS-tag⟩ pairs] : 398,445
- Total CFG rules in the final grammar: 1,476,941

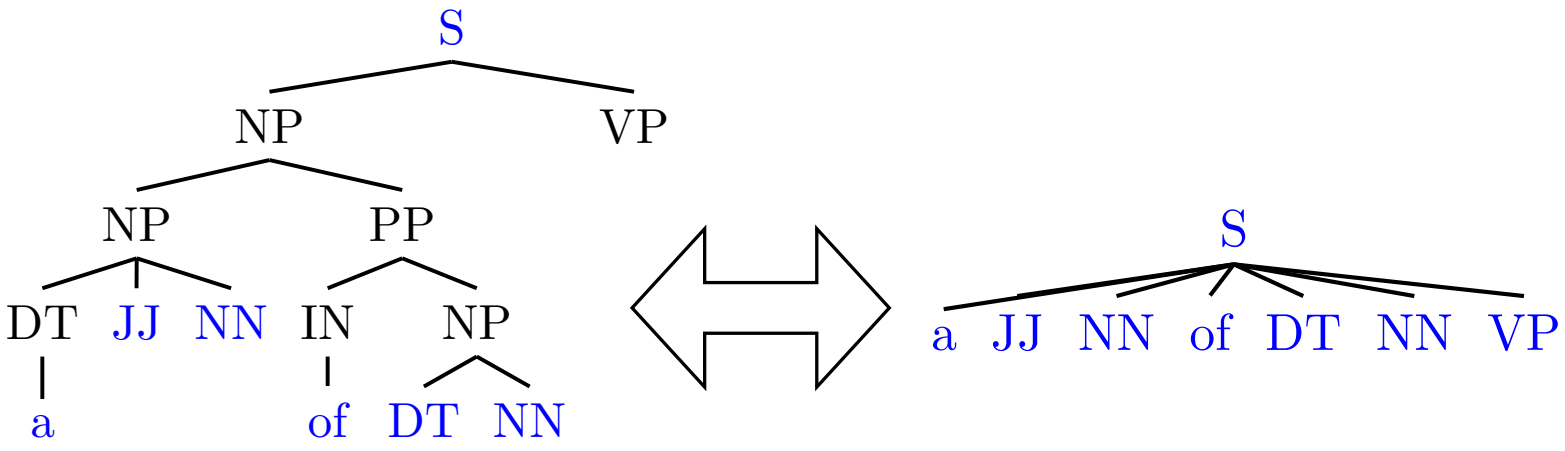
From fragments to CFG rules



f = 11

A significant portion of the order will be placed...

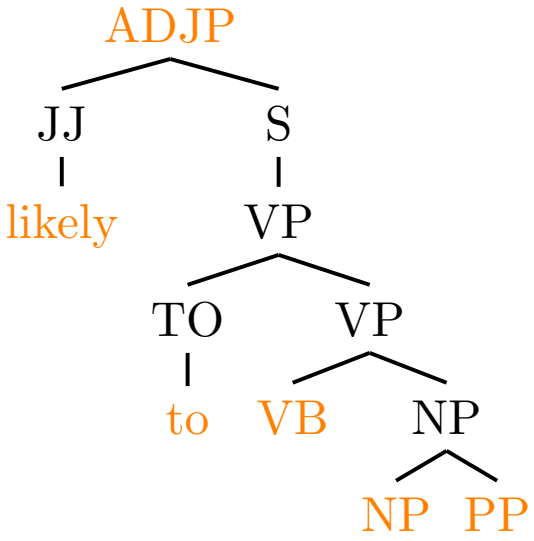
From fragments to CFG rules



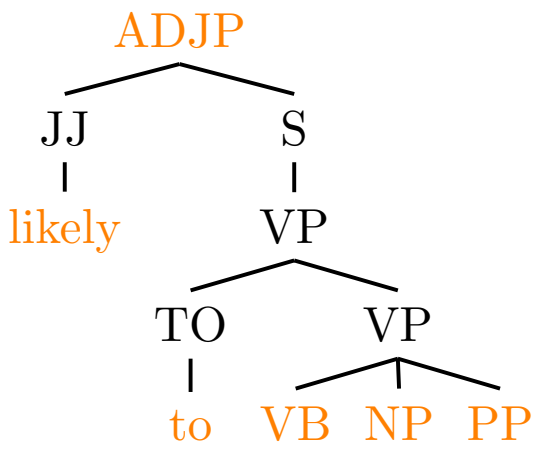
f = 11

A significant portion of the order will be placed...

Ambiguous Fragments

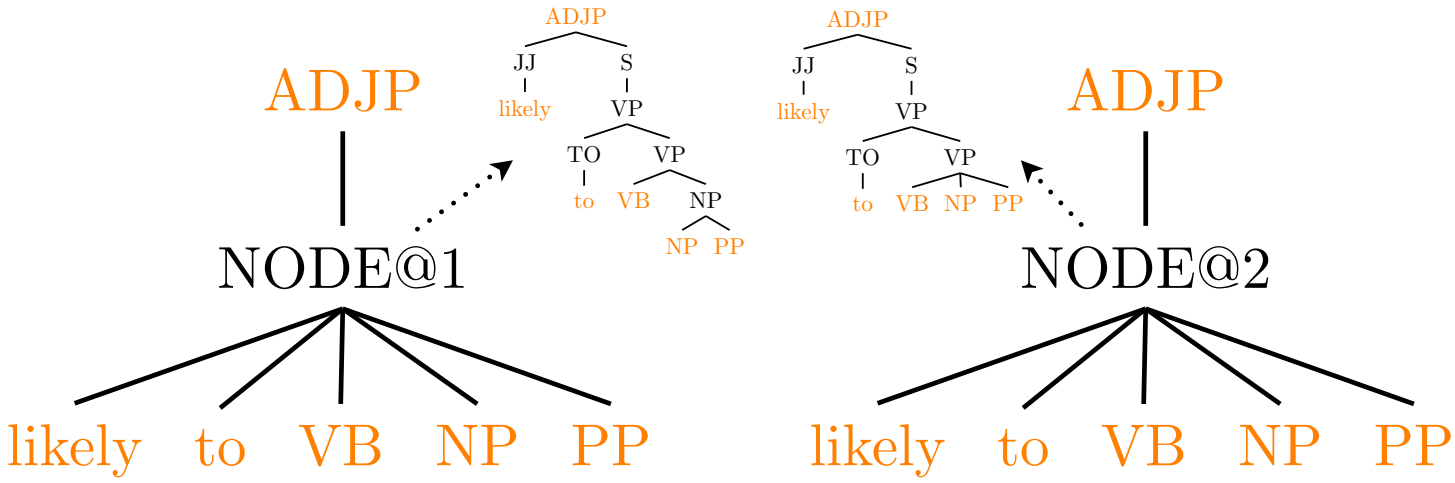


f=13
e.g. Likely to trigger an opposition from people



f =11
e.g. Likely to need help in the meantime

Ambiguous Fragments



f=13
e.g. Likely to trigger an opposition from people

f =11
e.g. Likely to need help in the meantime

Probability Estimates

- RFE: Relative Frequency Estimate on the actual counts of frags.
- EWE: Equal Weights Estimate (Goodman, 2003)

$$w_{\text{EWE}}(f) = \sum_{t \in TB} \frac{count(f,t)}{|\{f' \in t\}|}$$

$$p_{\text{EWE}}(f) = \frac{w_{\text{EWE}}(f)}{\sum_{f' \in F_{root(f)}} w_{\text{EWE}}(f')}$$

➡ J. Goodman. Efficient parsing of DOP with PCFG-reductions. In Data-Oriented Parsing. University of Chicago Press, Chicago, IL, USA, 2003.

- MLE: Maximum Likelihood Estimate

$$\hat{p} = \arg\max_p \text{ Likelihood}(treebank, p)$$

$$\text{Likelihood}(\text{treebank}, p) = \prod_{t \in \text{treebank}} P_p(t)$$

$$= \prod_{t \in \text{treebank}} \sum_{d \in \delta(t)} \prod_{\tau \in d} p(\tau)$$

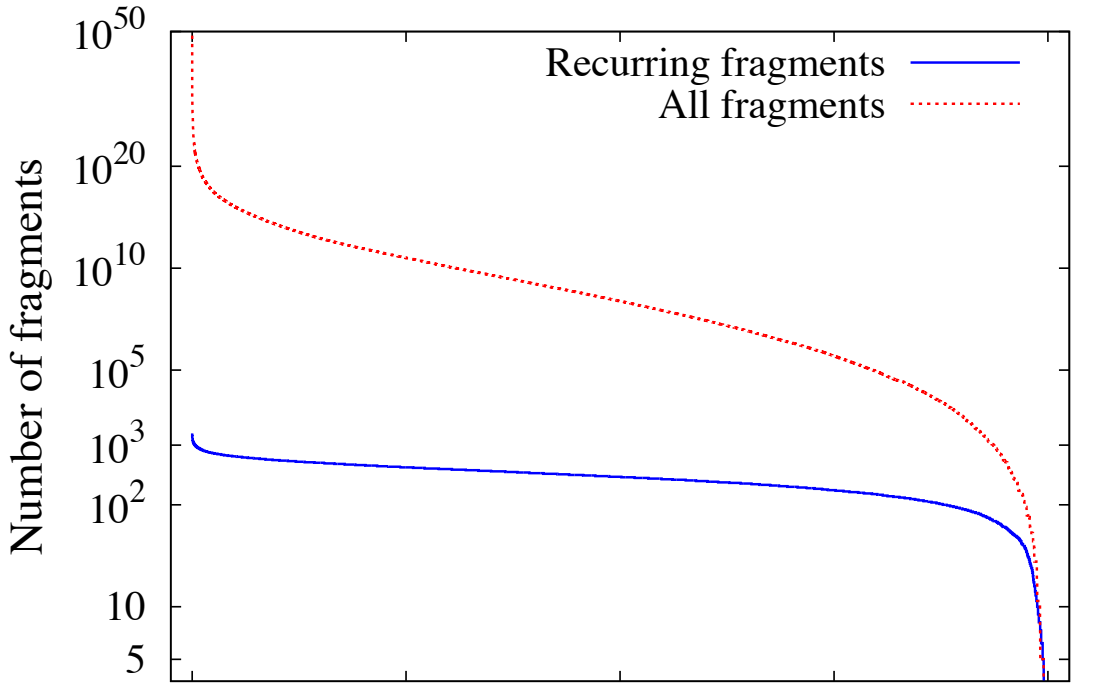
Probability Estimates Evaluation

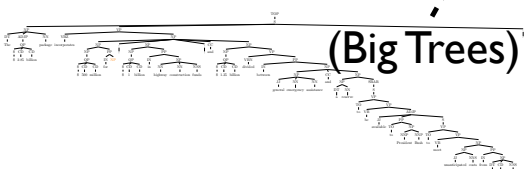
(development set)

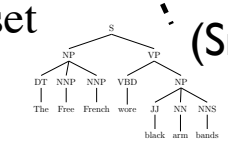
Estimate	**F1**
Rel Freq. Est. (RFE)	**87.2**
Equal Weights Est. (EWE)	86.8
Max Likelihood (ML)	86.6

Why RFE works well?

Double-DOP vs DOP1







Maximizing Objectives

- MPD: most probable derivation (Viterbi-best)
- MPP: approximate most probable parse tree
  - Get the 1,000 most probable derivations of the sentence
  - Sum up the probability of those generating the same tree
  - Obtain the parse tree with max probability

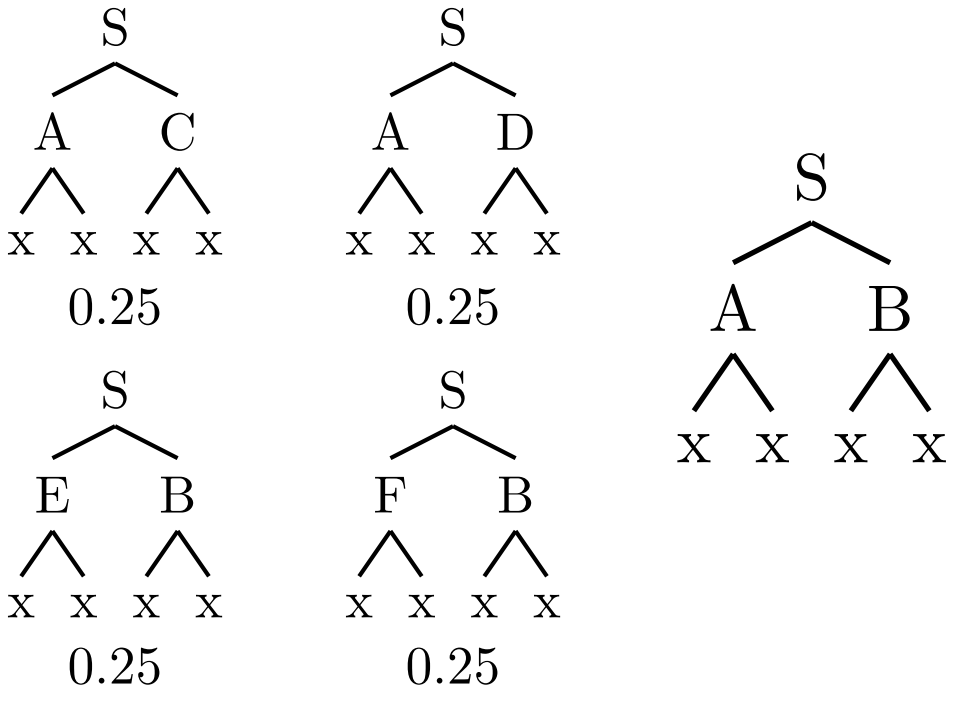
If we are interested in F1 better try to select the parse tree that is most likely to optimize this metric.

- MCP: maximum constituent parse (Goodman, 1996)

Maximum Constituent Parse

Binary Case: max(Recall) = max(Precision)

S	→	A C	0.25
S	→	A D	0.25
S	→	E B	0.25
S	→	F B	0.25
A	→	x x	1.0
B	→	x x	1.0
C	→	x x	1.0
D	→	x x	1.0
E	→	x x	1.0
F	→	x x	1.0

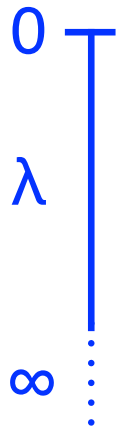


➡ J. Goodman. Parsing algorithms and metrics. ACL 1996.

Maximum Constituent Parse

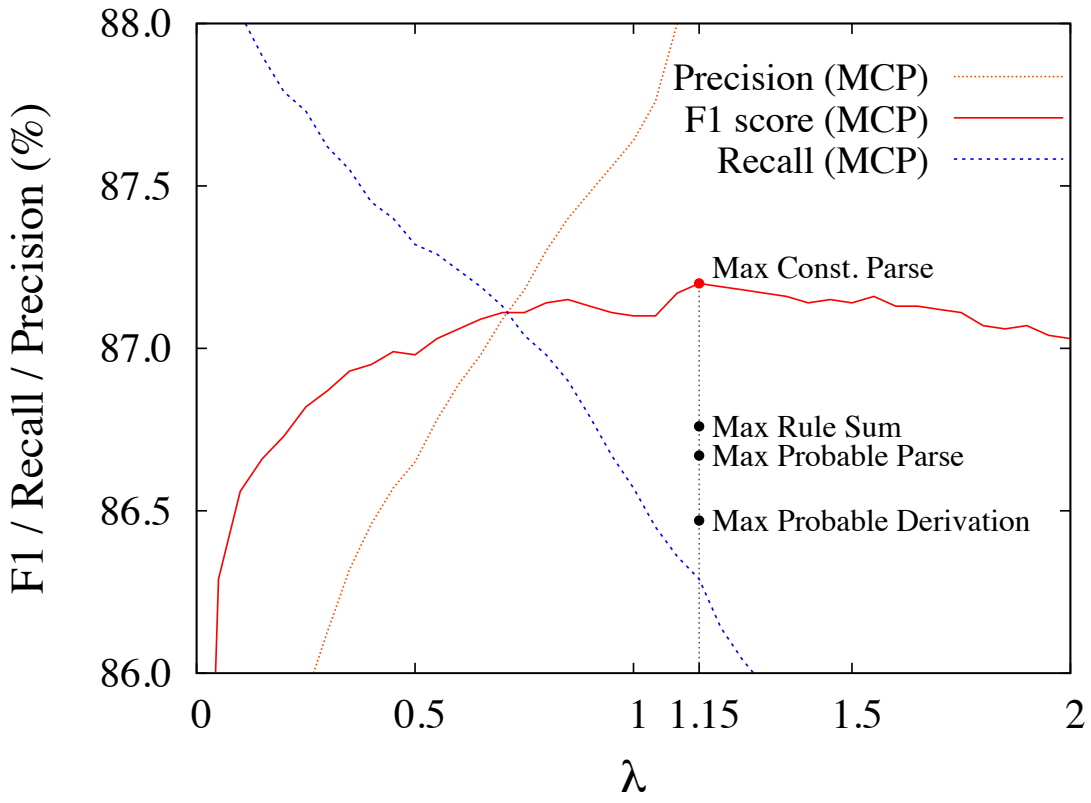
n-ary branching case: $\max(\text{Recall}) \neq \max(\text{Precision})$

- **Max Recall**
  - # correct constituents / gold constituents
  - risk: get as many correct constituents as possible
  - prefers binary rules
- **Max Precision**
  - # correct constituents / guessed constituents
  - play safe: get as few correct constituents as possible
  - prefers flat rules



➡ J. Goodman. Parsing algorithms and metrics. ACL 1996.

Max Objectives Evaluation
(development set)



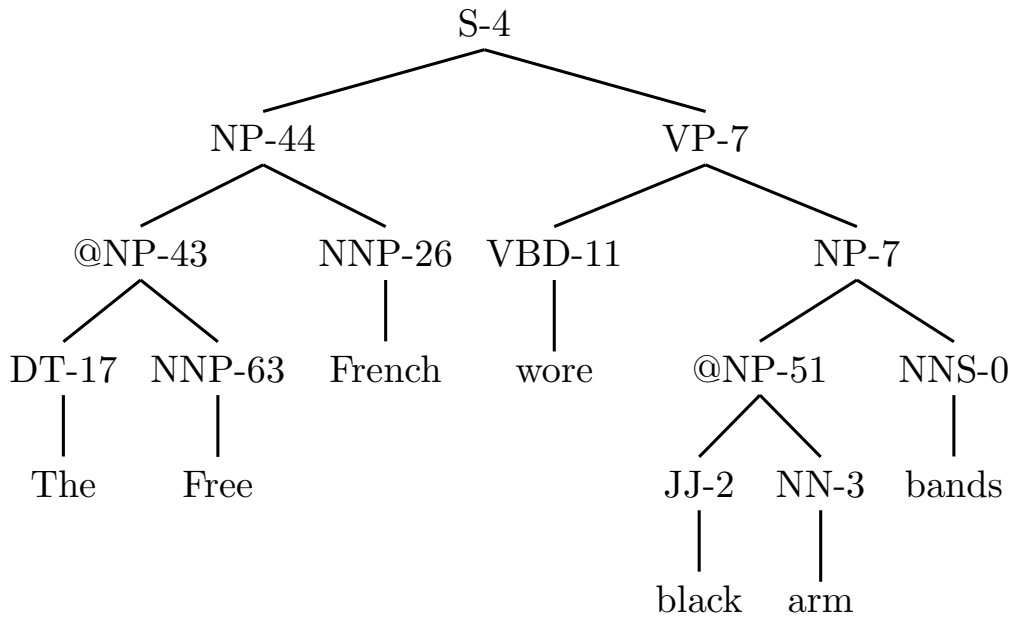
Parsing Results

(WSJ test set)

Parsing Model	test (≤ 40) F1	test (≤ 40) EX	test (all) F1	test (all) EX
PCFG Baseline				
PCFG (H=1, P=1)	77.6	17.2	76.5	15.9
PCFG (H=1, P=1) Lex smooth.	78.5	17.2	77.4	16.0
FRAGMENT-BASED PARSERS				
Zuidema (2007)*	83.8	26.9	-	-
Cohn et al. (2010) MRS	85.4	27.2	84.7	25.8
Post and Gildea (2009)	82.6	-	-	-
Bansal and Klein (2010) MCP	88.5	33.0	87.6	30.8
Bansal and Klein (2010) MCP + Additional Refinement	**88.7**	**33.8**	**88.1**	**31.7**
THIS PAPER				
Double-DOP	87.7	33.1	86.8	31.0
Double-DOP Lex smooth.	**87.9**	**33.7**	**87.0**	**31.5**
REFINEMENT-BASED PARSERS				
Collins (1999)	88.6	-	88.2	-
Petrov and Klein (2007)	**90.6**	**39.1**	**90.1**	**37.1**

Berkeley State Splitting (Sp)

6 levels of refinements

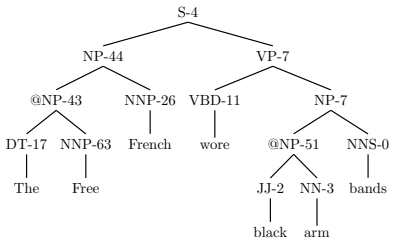


Category	Words
DT-17	The, A
NNP-26	Commission, Committee, Association, Department, Museum, Revolution, Office, ...
VBD-11	sent, exercised, received, retained, completed, followed, made, ...
JJ-2	ceramic, young, daily, imperial, full, ...

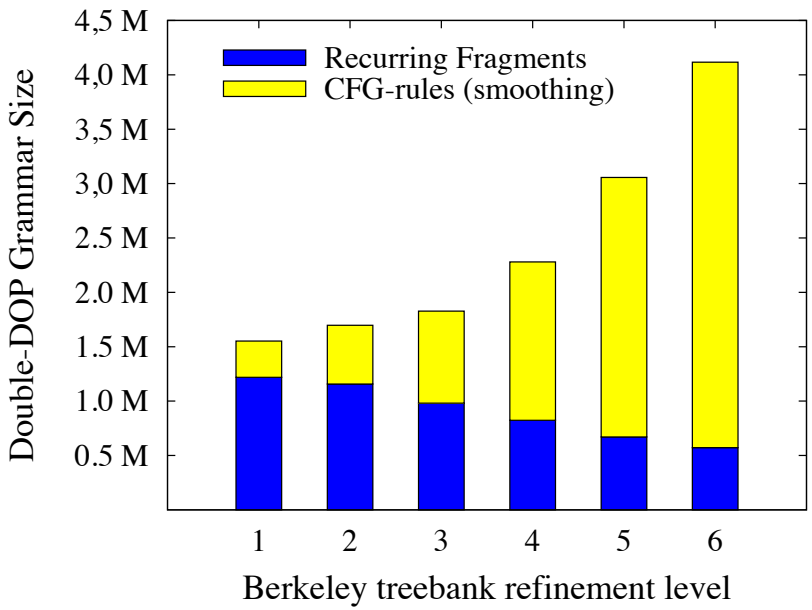
➡ S. Petrov. Coarse-to-Fine Natural Language Processing. PhD thesis, University of California at Bekeley, 2009.

Berkeley State Splitting (Sp)

Evaluating Double-DOP on the 6 levels

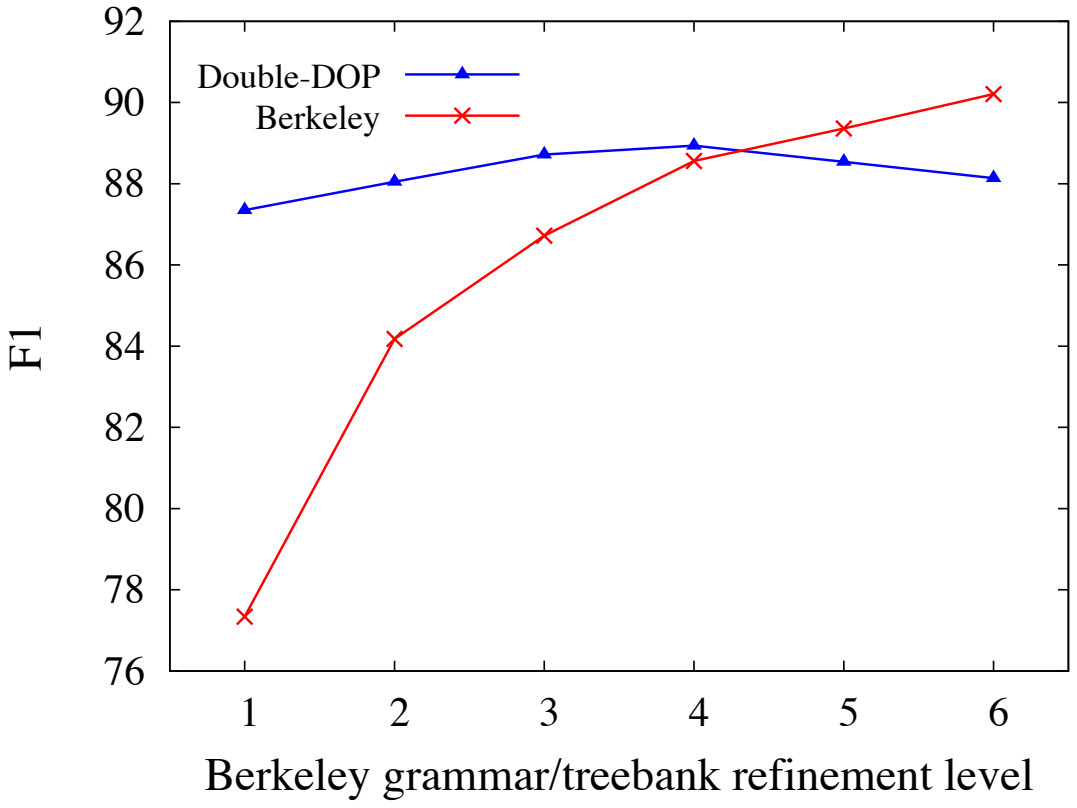


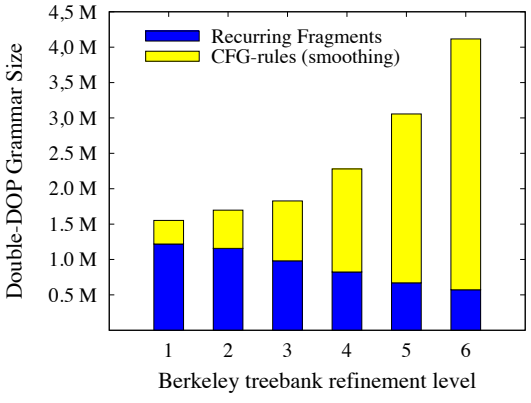
Category	Words
DT-17	The, A
NNP-26	Commission, Committee, Association, Department, Museum, Revolution, Office, ...
VBD-11	sent, exercised, received, retained, completed, followed, made, ...
JJ-2	ceramic, young, daily, imperial, full, ...



Berkeley State Splitting (Sp)

Evaluating Double-DOP on the 6 levels





Conclusions

- Recurring Fragments (Fragment Seeker)
  - ⋆ Versatile for different applications
  - ⋆ Easy to extend to other representations
- Double-DOP
  - ⋆ Good results with parsing
  - ⋆ Explicit fragments (complementary to other approaches)
  - ⋆ Software publicly available

Word-order

- DOP has mostly been evaluated on English, which is highly configurational.
- Word-order variation presents a problem:
  Have fragment for "a b", but this is useless to parse variant "b a".
- Fundamental problem: allowing all possible-orders results in $O(n!)$ permutations.
- Idea: recognize variants from treebank; derive rules.
- But: no direct evidence for this.
  $\Rightarrow$ alternation vs. change in meaning.

DOP for dependency structures?

- Most DOP models are based on constituency structures (or LFG, HPSG)
- Could there be a Data-Oriented Dependency model?
- Dependency structures are labelled, directed graphs.
- All nodes are terminals; i.e., no phrases.

Discontinuous constituents

Definition of discontinuity: A **discontinuous constituent** is a group of words that form a constituent while being non-contiguous

Discontinuous phenomena:

- Cross-serial dependencies
- Extraposition: topicalization, wh-extraction
- Word-order freedom: scrambling

Example

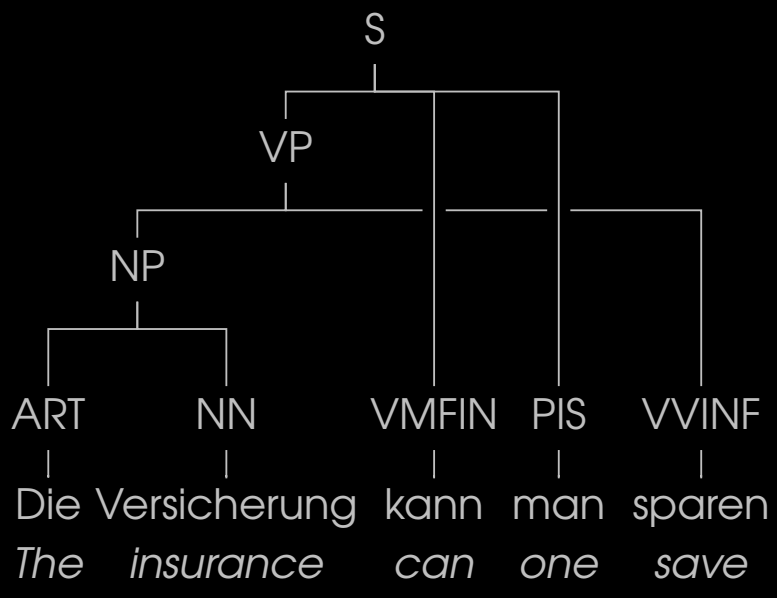


Figure: A discontinuous tree from the Negra corpus.
Translation: *As for the insurance, one can save it.*

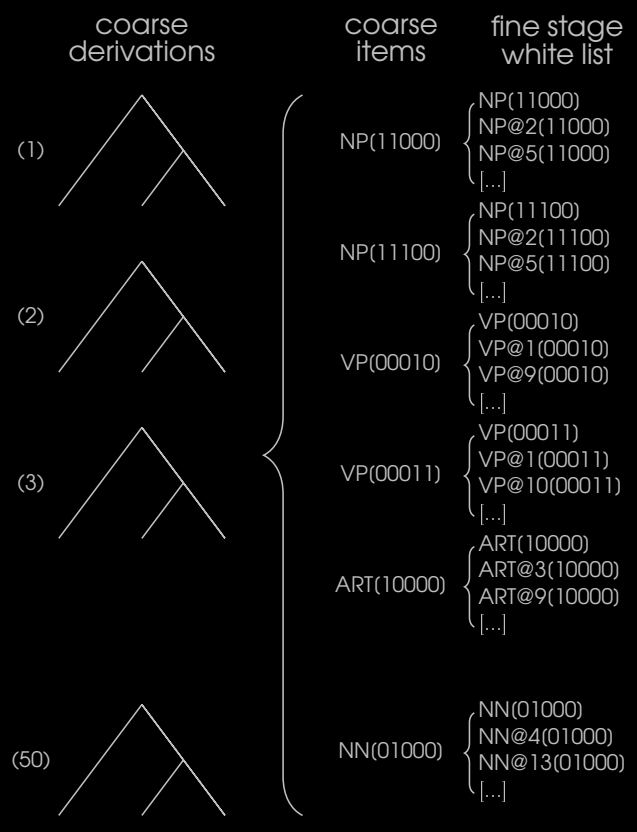
Discontinuous constituents: Disco-DOP (2011)

Disco-DOP:

- Mildly-context sensitive grammar (LCFRS) as treebank grammar
- Encode Goodman reduction in it
- Parse using coarse-to-fine

van Cranenburgh (2011), MSc thesis

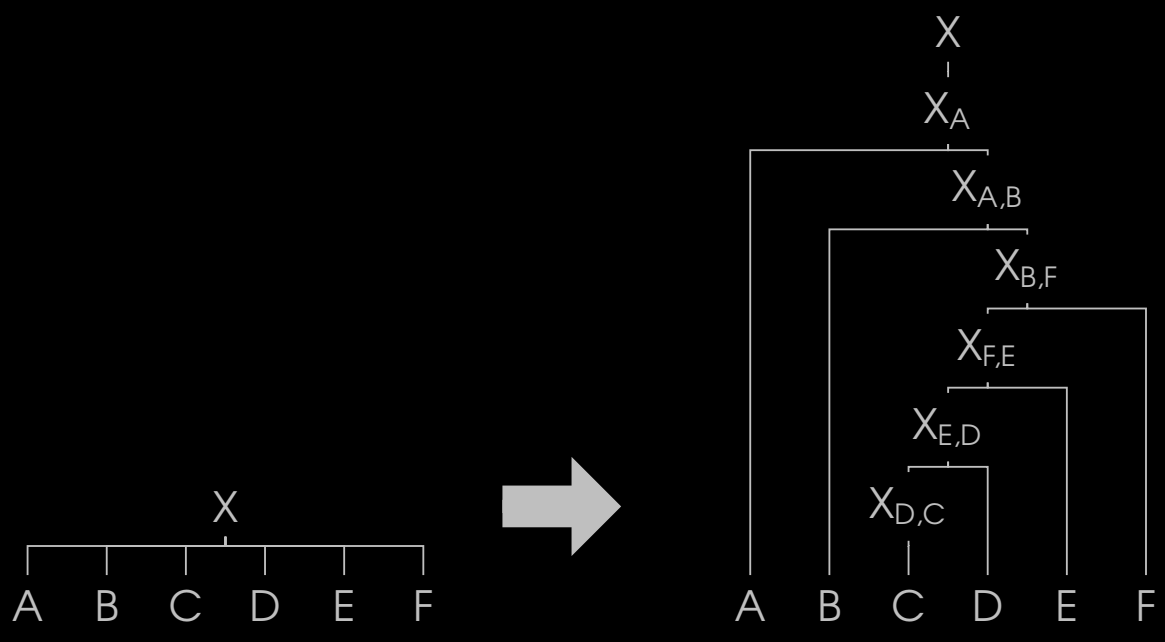
k-best Coarse-to-fine parsing



Prune DOP derivations with k-best PLCFRS derivations.

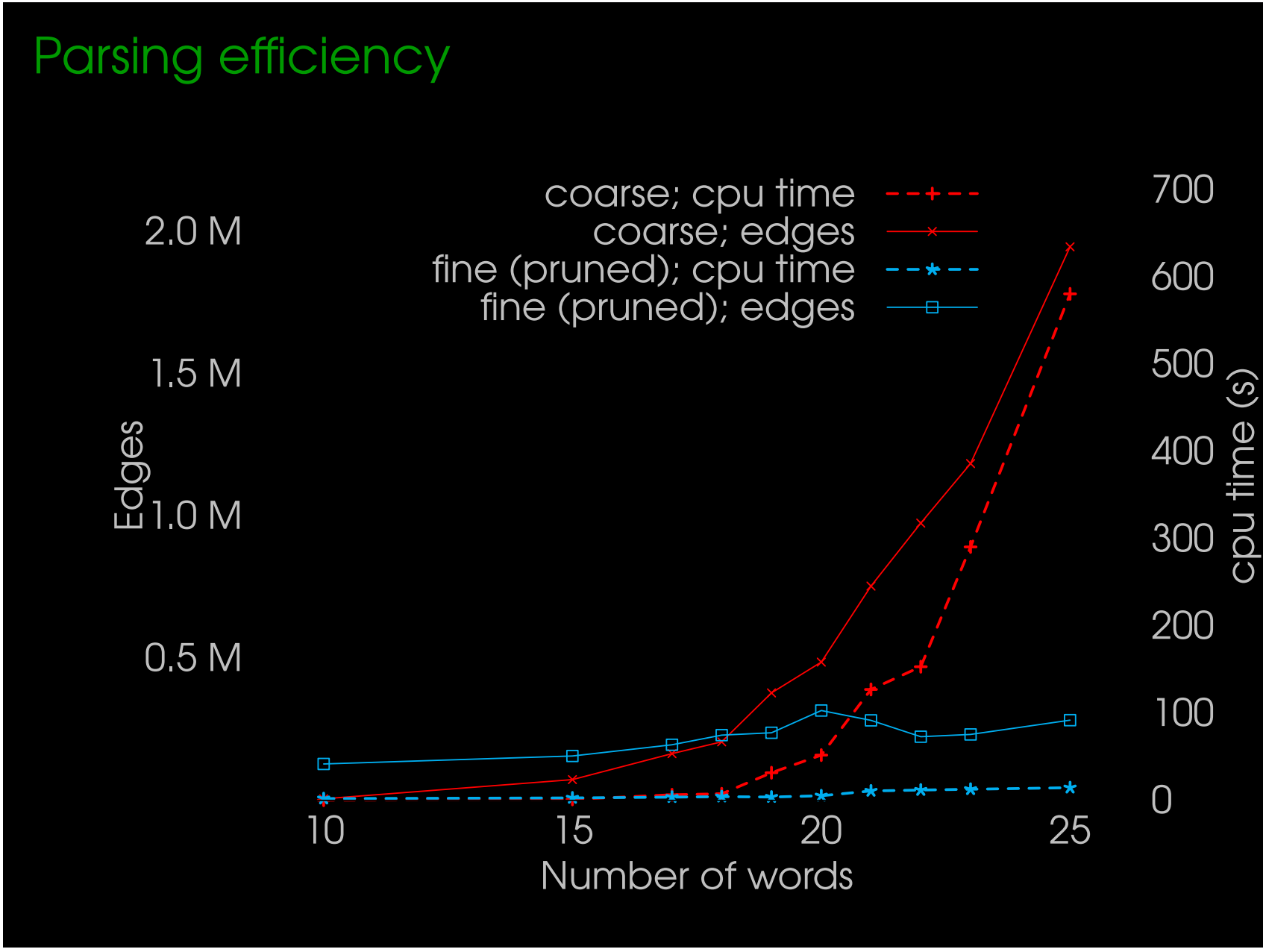
Binarization

- mark heads of constituents
- head-outward binarization (parse head first)
- no parent annotation: $v = 1$
- horizontal Markovization: $h \in \{1, 2, \infty\}$



Klein & Manning (2003): Accurate unlexicalized parsing.

Parsing efficiency



Evaluation

NEGRA	words	F_1	EX	COV.
DPSG Pla2004*	$\leq$ 15	73.16	39.0	96.04
PLCFRS KaMa2010†	$\leq$ 15	81.27	-	-
DISCO-DOP v=1, h=1	$\leq$ 15	**84.56**	54.68	99.90
PLCFRS KaMa2010†	$\leq$ 25	73.25	-	99.45
PLCFRS v=1, h=2	$\leq$ 25	75.98	36.79	98.90
DISCO-DOP v=1, h=2	$\leq$ 25	**78.81**	39.60	98.90
PLCFRS Mai2010‡	$\leq$ 30	71.52	-	97.00
PLCFRS v=1, h=∞	$\leq$ 30	72.34	31.27	96.59
DISCO-DOP v=1, h=∞	$\leq$ 30	**73.98**	34.96	96.59
DISCO-DOP CFG-CTF, v=1 h=1	$\leq$ 40	**74.27**	34.26	100.0

Table: Discontinuous parsing on the Negra corpus.
Function tags discarded; Gold POS tags given to parser.

Source code: `http://github.com/andreasvc/disco-dop`

*Plaehn (2004). †Kallmeyer & Maier (2010). ‡Maier (2010).

Extract recurring fragments in avg linear time (WIP)

- We can speed up fragment extraction by sorting nodes of trees:
- $\Rightarrow$ Aligns potentially equal nodes, allowing us to skip the rest! (Moschitti 2006)

	Time (hr:min)		
Implementation	CPU	Wall clock	# fragments
Quadratic (Sangati, 2012)	160	10:00	1,023,092
Quadratic (my impl.)	93	6:15	1,032,568
Fast (my impl.)	2.3	0:09	1,023,880

Table: Performance comparison. Wall clock time is when using 16 cores.

Moschitti (2006): Making tree kernels practical for natural language learning

Authorship attribution with fragments (accepted)

- Parse known texts with off-the-shelf parser
- Classify author of unknown text by counting common fragments w/known texts
- Author corpus with maximal fragment overlap is probably the author.

20 sents	trigrams	fragments	combined
Conrad	89.00	91.00	95.00
Hemingway	77.00	58.00	78.00
Huxley	74.74	66.32	76.84
Salinger	95.00	91.00	98.00
Tolstoy	84.00	82.00	92.00
average:	84.04	77.78	**88.08**
100 sents avg	97.98	92.93	**98.99**

Table: Accuracy in % for authorship attribution of literary texts.

Credits

Slides borrowed from:

- Andreas van Cranenburgh
- Federico Sangati

Conclusions

- Data-oriented parsing was one of the first proposals for modern, wide-coverage parsing - radical at the time, but the main innovations have become standard in the field;
- Work on generative, data-oriented grammars continues, with competitive results on English and other languages (but less attention internationally than around 2000)
- Remains one of the few approaches to syntactic structure that combines engineering success with aspirations as a cognitive model

Bod, R. (1992). A computational model of language performance: Data oriented parsing. In: *Proceedings COLING'92 (Nantes, France)*, pp. 855–859. Morristown, NJ: Association for Computational Linguistics.

Charniak, E. (1996). Tree-bank grammars. Tech. rep.

Collins, M. & Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: *ACL 2002*.

Huybrechts, R. (1984). The weak inadequacy of context-free phrase structure grammars. In: *Van Periferie naar Kern* (de Haan, G., Trommelen, M. & Zonneveld, W., eds.). Foris.

Johnson, M. (2002). The DOP estimation method is biased and inconsistent. *Computational Linguistics* **28**, 71–76.

Joshi, A. K. (1985). How much context-sensitivity is required to provide reasonable structural descriptions: Tree-adjoining grammars. In: *Natural Language Parsing: Psycholinguistic, Computational and Theoretical Perspectives* (Dowty, D., Karttunen, L. & Zwicky, A., eds.), pp. 206–350. New York: Cambridge University Press.

Lari, K. & Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* **4**, 35–56.

Sangati, F. & Zuidema, W. (2011). Accurate parsing with compact tree-substitution grammars: Double-dop. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 84–95. Association for Computational Linguistics.

Scha, R. (1990). Taaltheorie en taaltechnologie; competence en performance. In: *Computertoepassingen in de Neerlandistiek* (de Kort, R. & Leerdam, G., eds.), pp. 7–22. Almere, the Netherlands: LVVN. English translation at http://iaaa.nl/rs/LeerdamE.html.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy* **8**, 333–343.

Sima'an, K. & Buratto, L. (2003). Backoff parameter estimation for the DOP model. In: *Proceedings of the 14th European Conference on Machine Learning (ECML'03, Cavtat-Dubrovnik, Croatia*, no. 2837 in Lecture Notes in Artificial Intelligence, pp. 373–384. Berlin, Germany: Springer Verlag.

Zollmann, A. (2004). *A Consistent and Efficient Estimator for the Data-Oriented Parsing Model*. Master's thesis, Institute for Logic, Language and Computation, University of Amsterdam, the Netherlands. http://www-2.cs.cmu.edu/~zollmann/.

Zuidema, W. (2007). Parsimonious Data-Oriented Parsing. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 551–560.