Accurate Parsing with Compact Tree Substitution Grammars: Double-DOP

Federico Sangati
Willem Zuidema

July 27, 2011

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION
University of Amsterdam

Outline

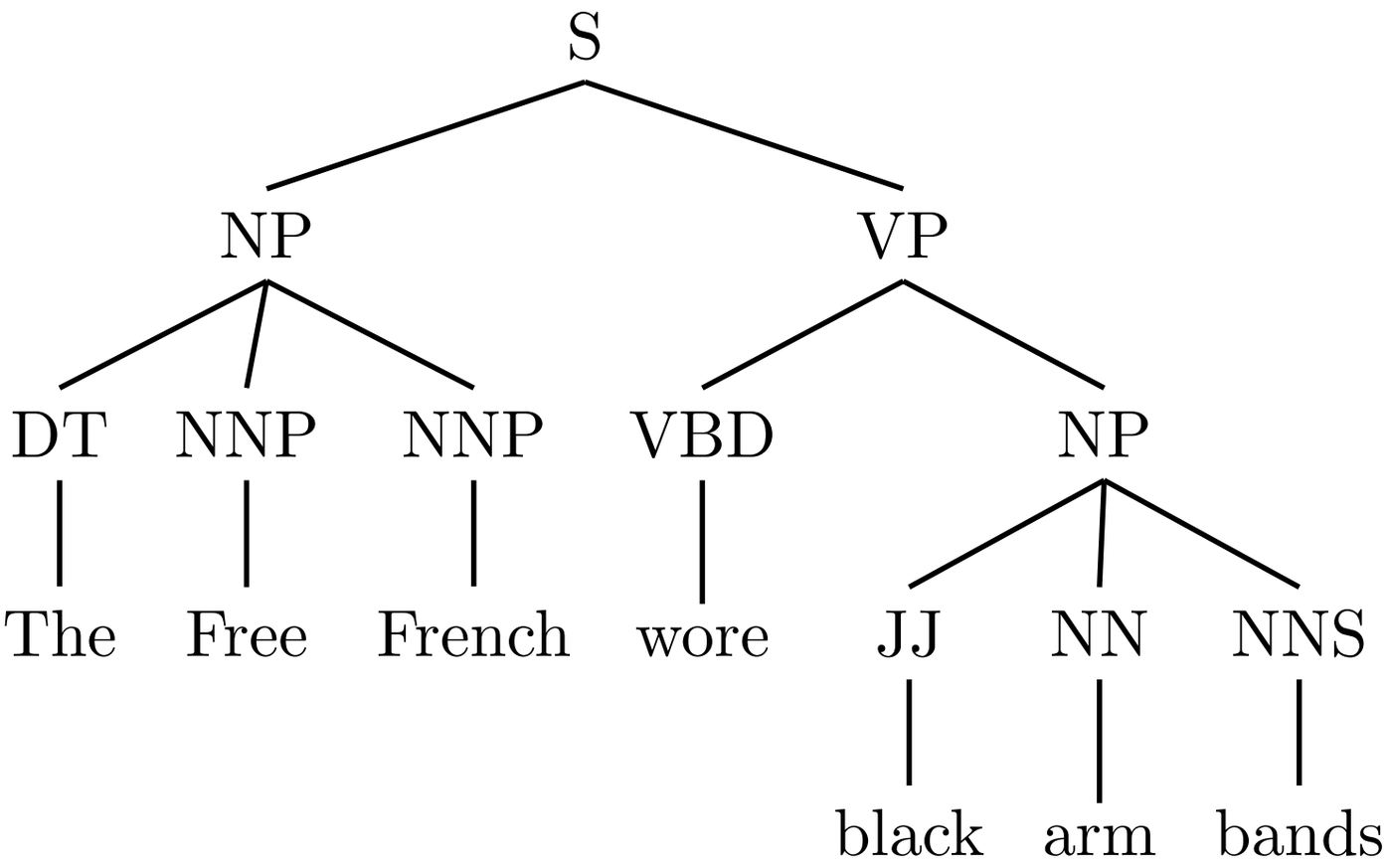
1. DOP (Data Oriented Parsing)
2. Seeking Recurring Fragments
3. Parsing with Double-DOP
4. Conclusions

Data Oriented Parsing (DOP)

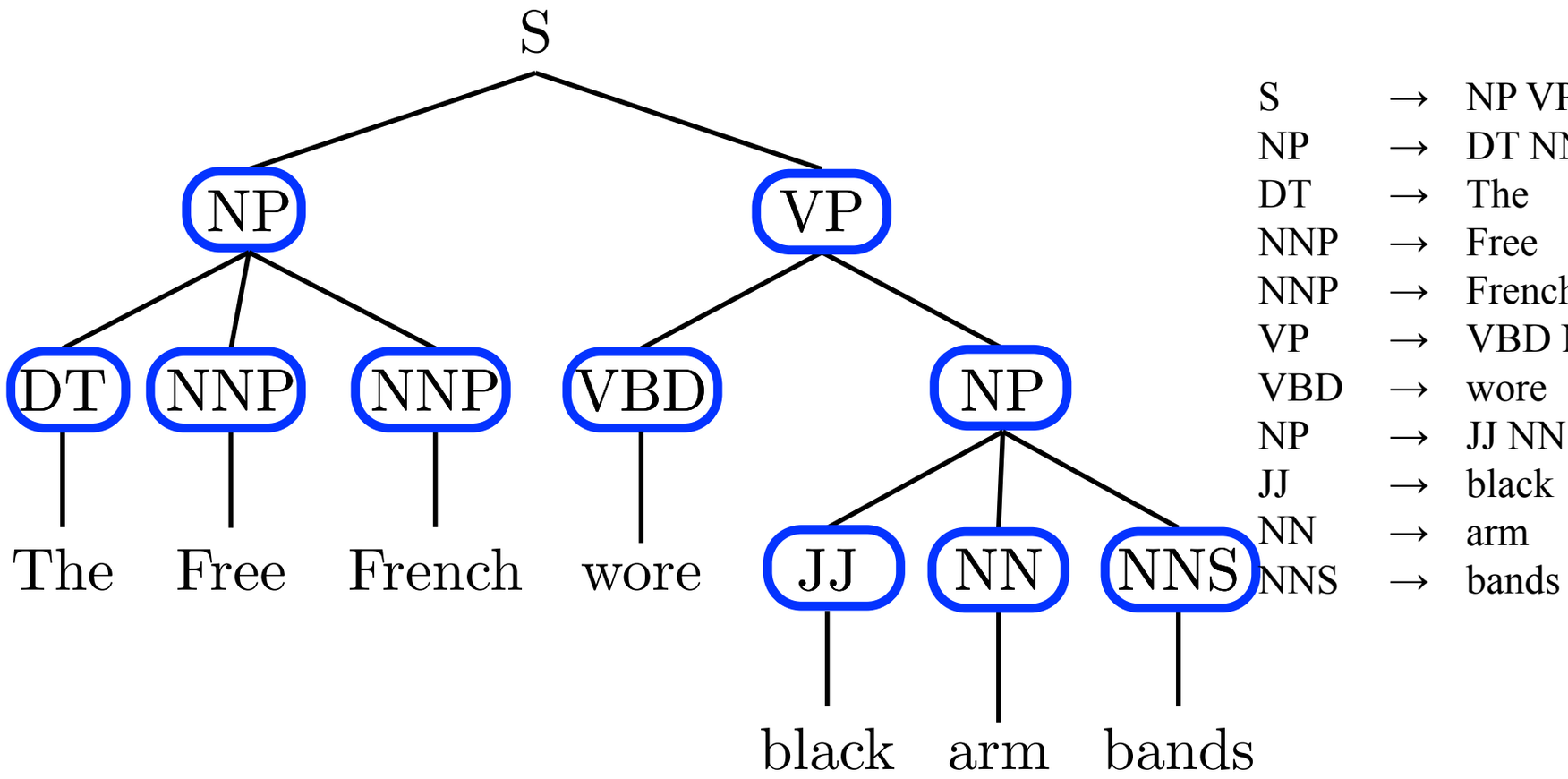
- Conceptualized by Remko Scha (1990)
- Uses arbitrarily large fragments
- Belongs to the family of Tree Substitution Grammars

➡ R. Scha. Language theory and language technology: Competence and Performance. 1990.

Phrase Structures

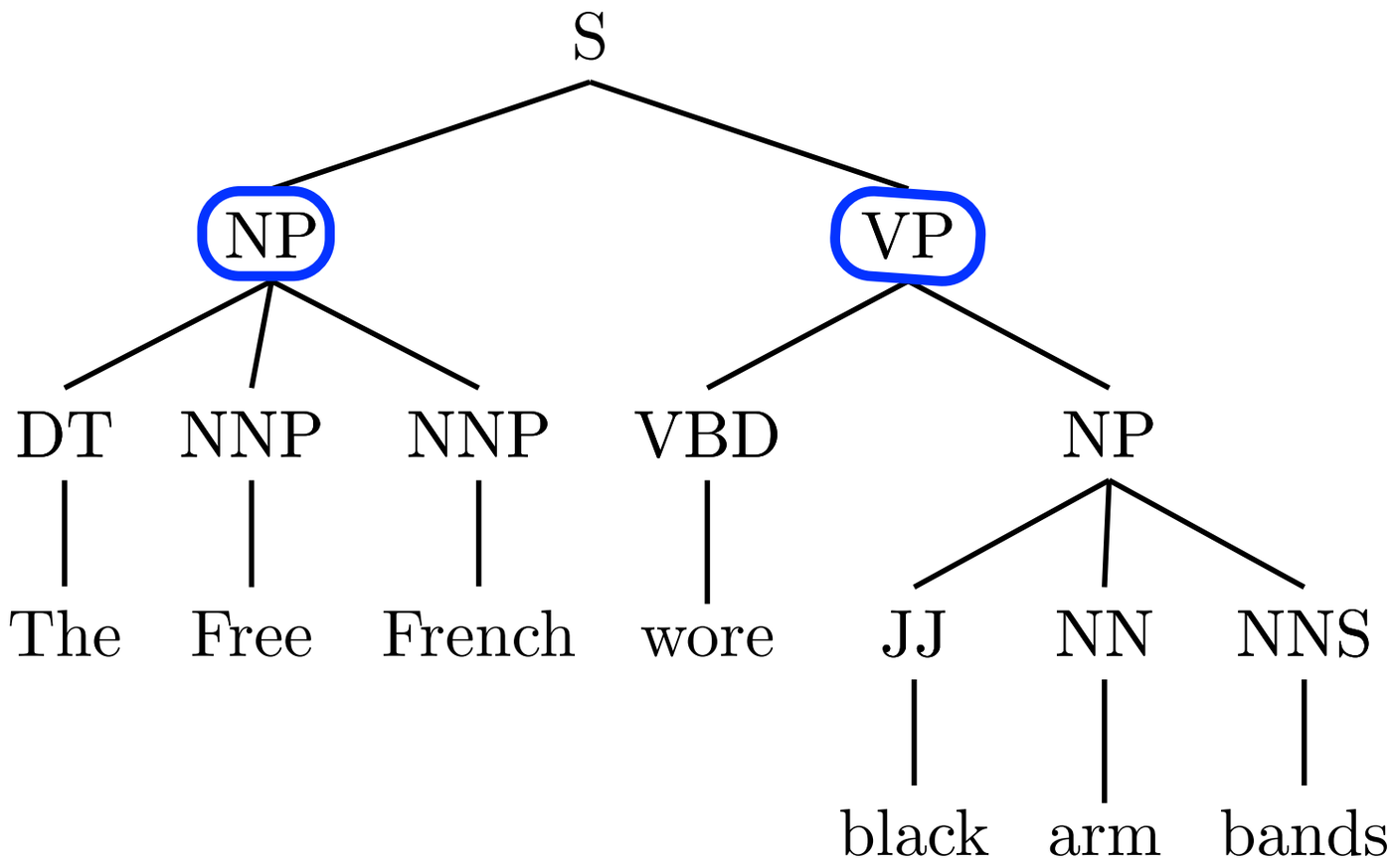


Context Free Grammar (CFG)

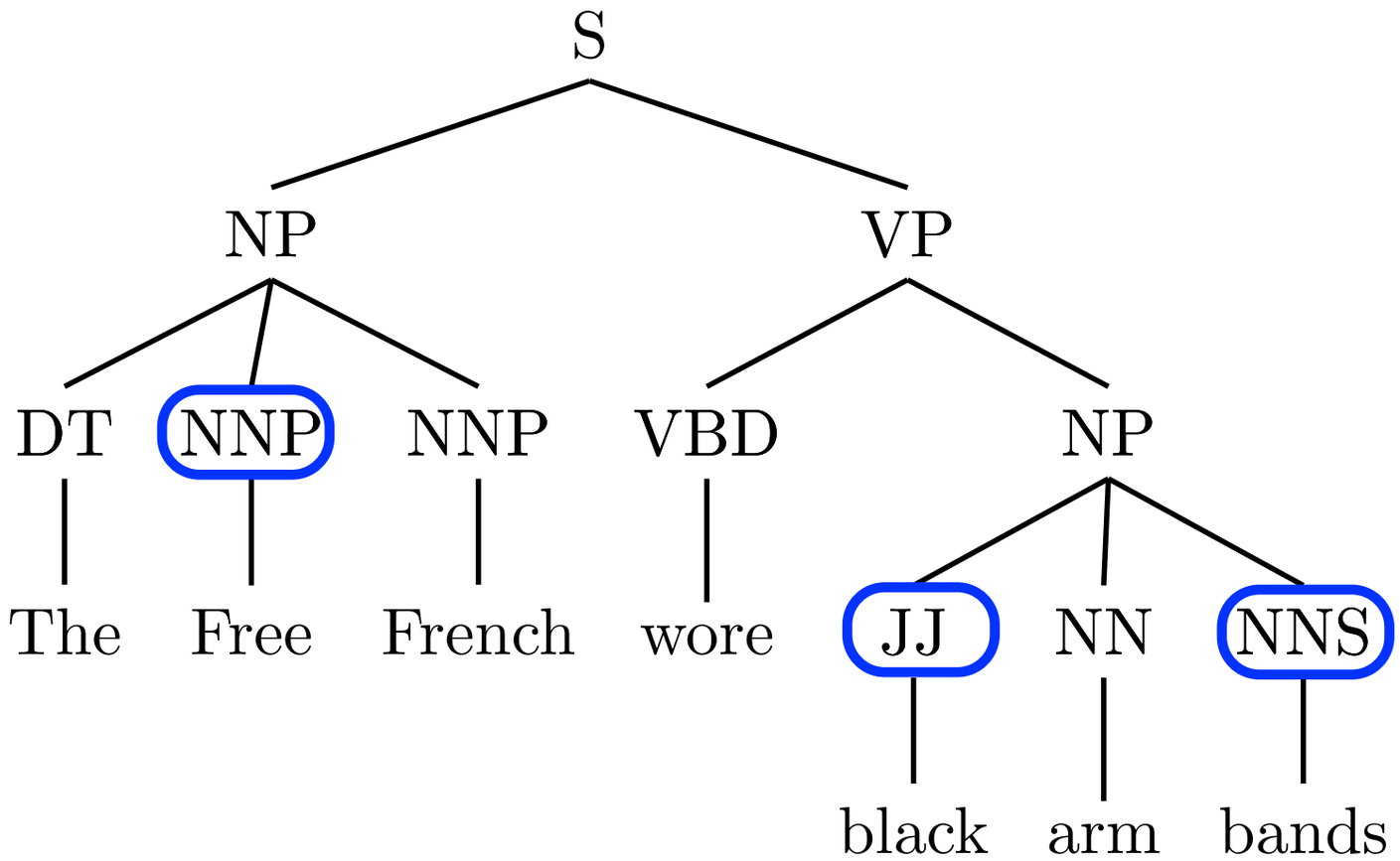


S	→	NP VP
NP	→	DT NNP NNP
DT	→	The
NNP	→	Free
NNP	→	French
VP	→	VBD NP
VBD	→	wore
NP	→	JJ NN NNS
JJ	→	black
NN	→	arm
NNS	→	bands

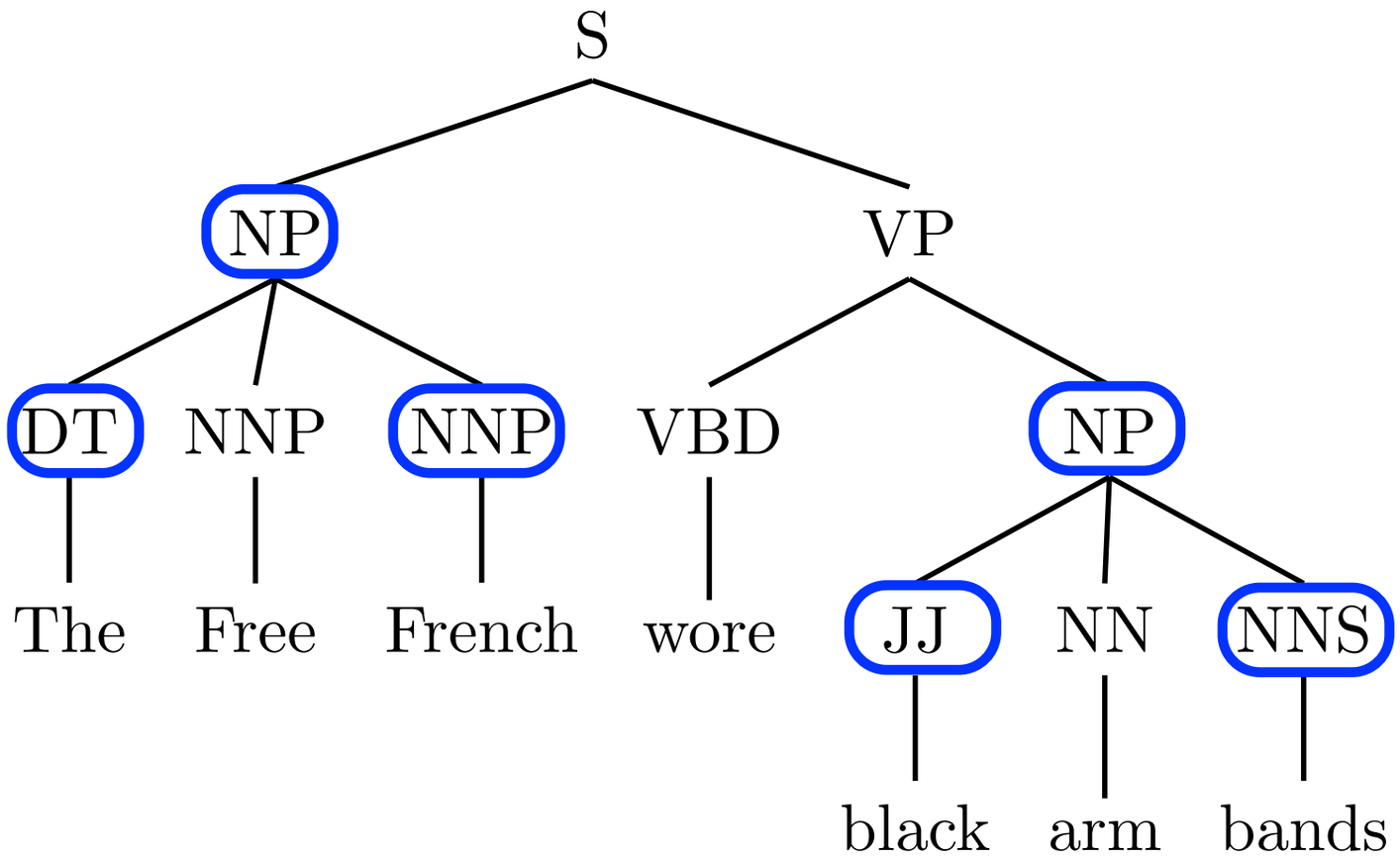
Data Oriented Parsing (DOP)



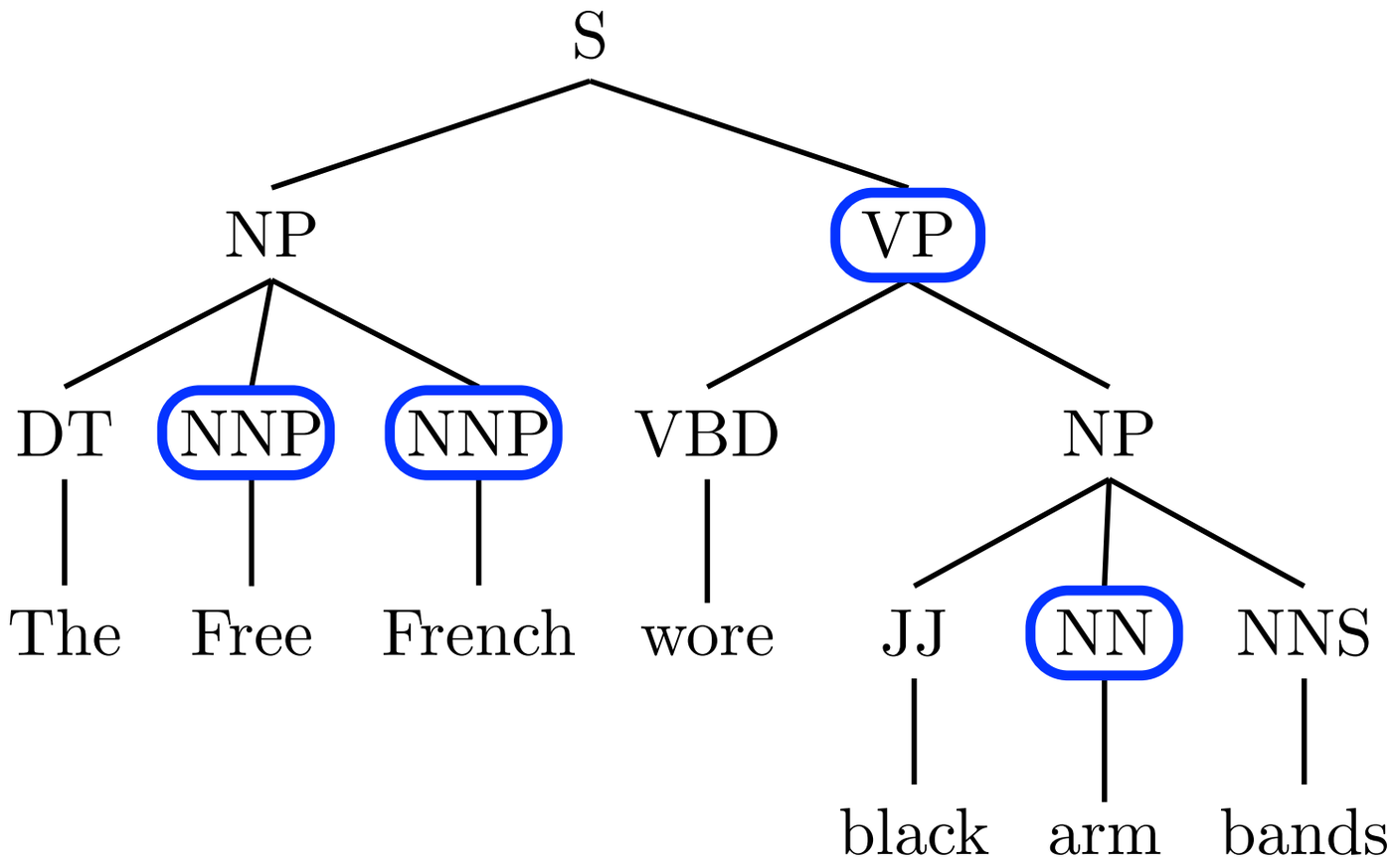
Data Oriented Parsing (DOP)



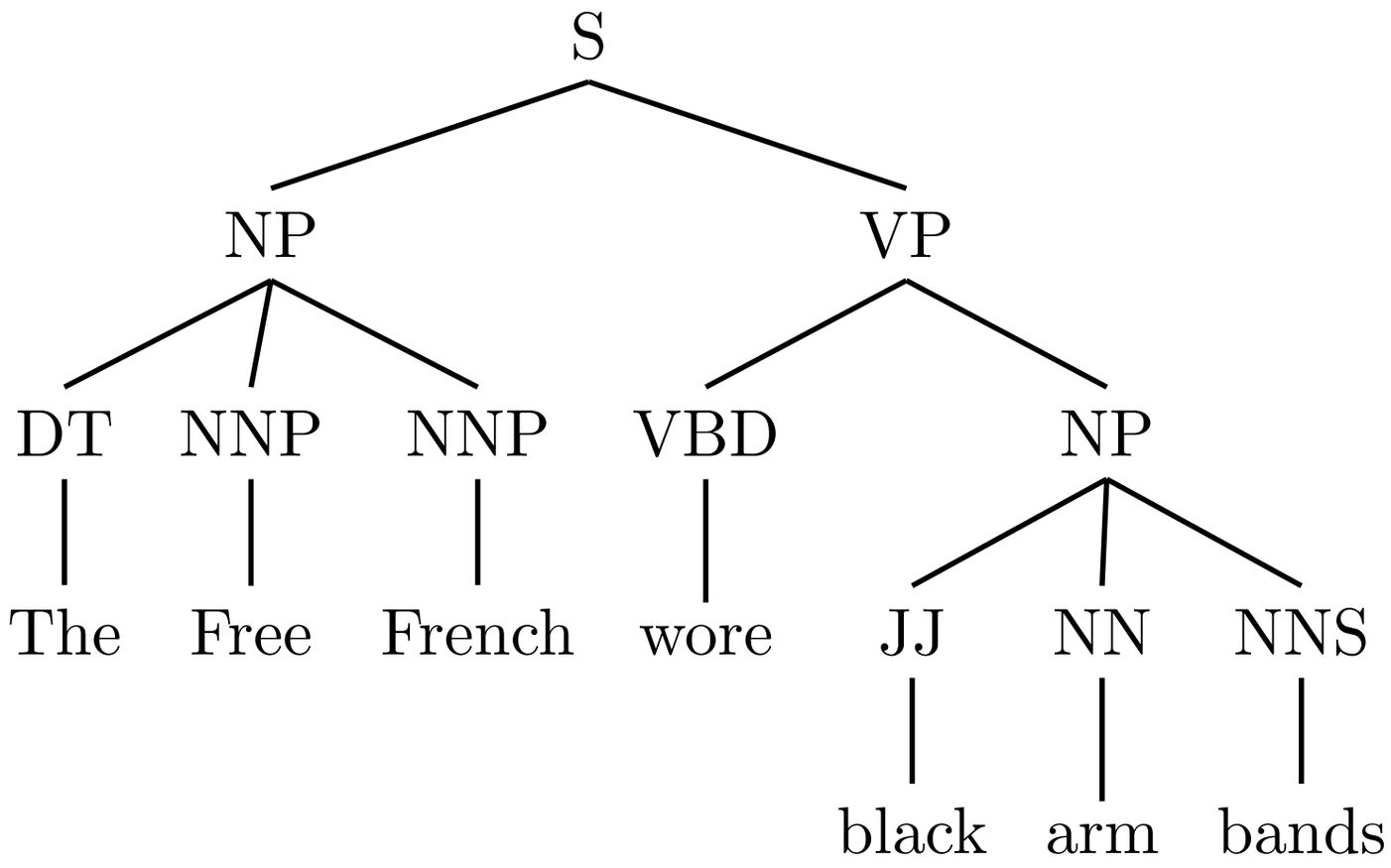
Data Oriented Parsing (DOP)



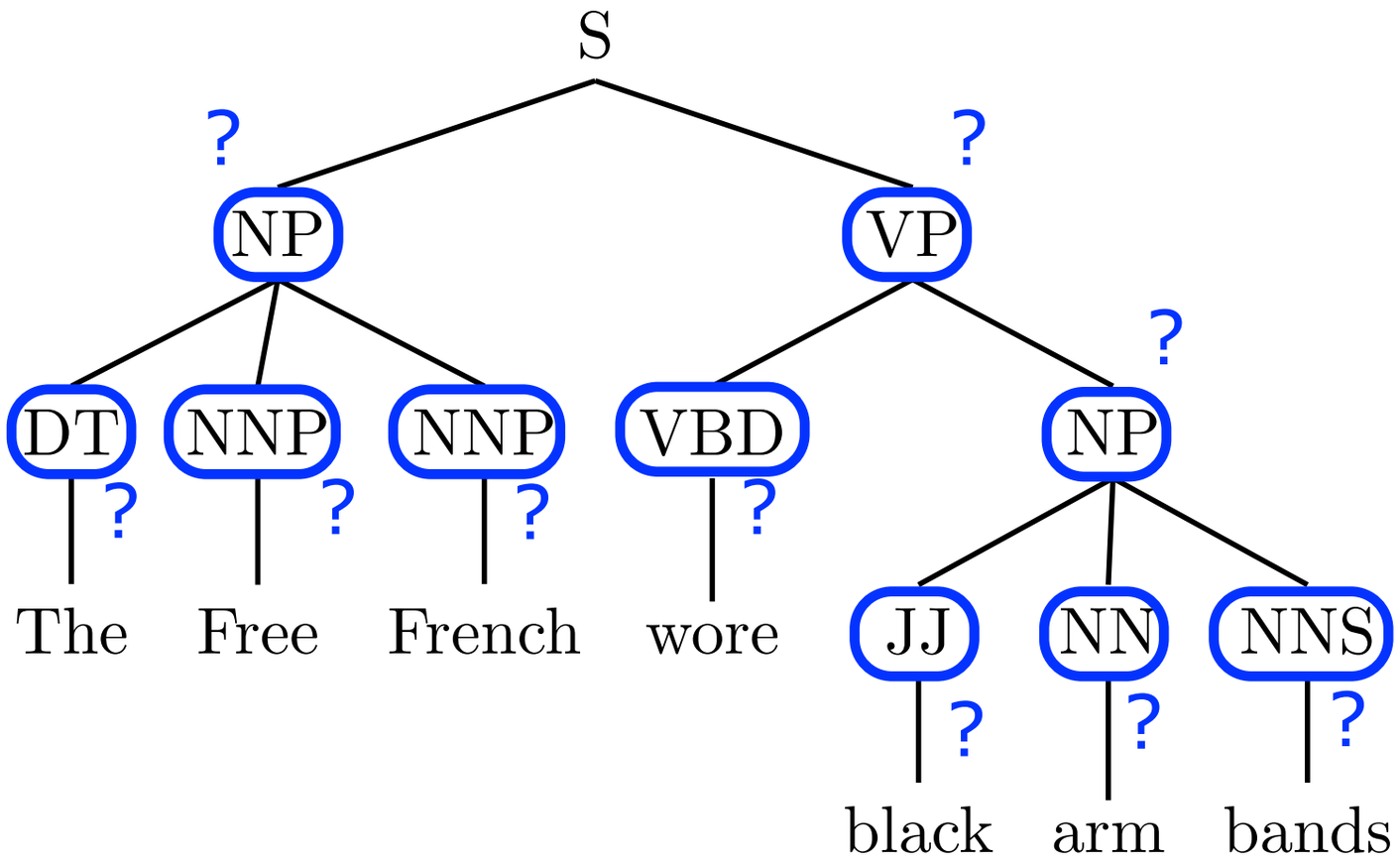
Data Oriented Parsing (DOP)



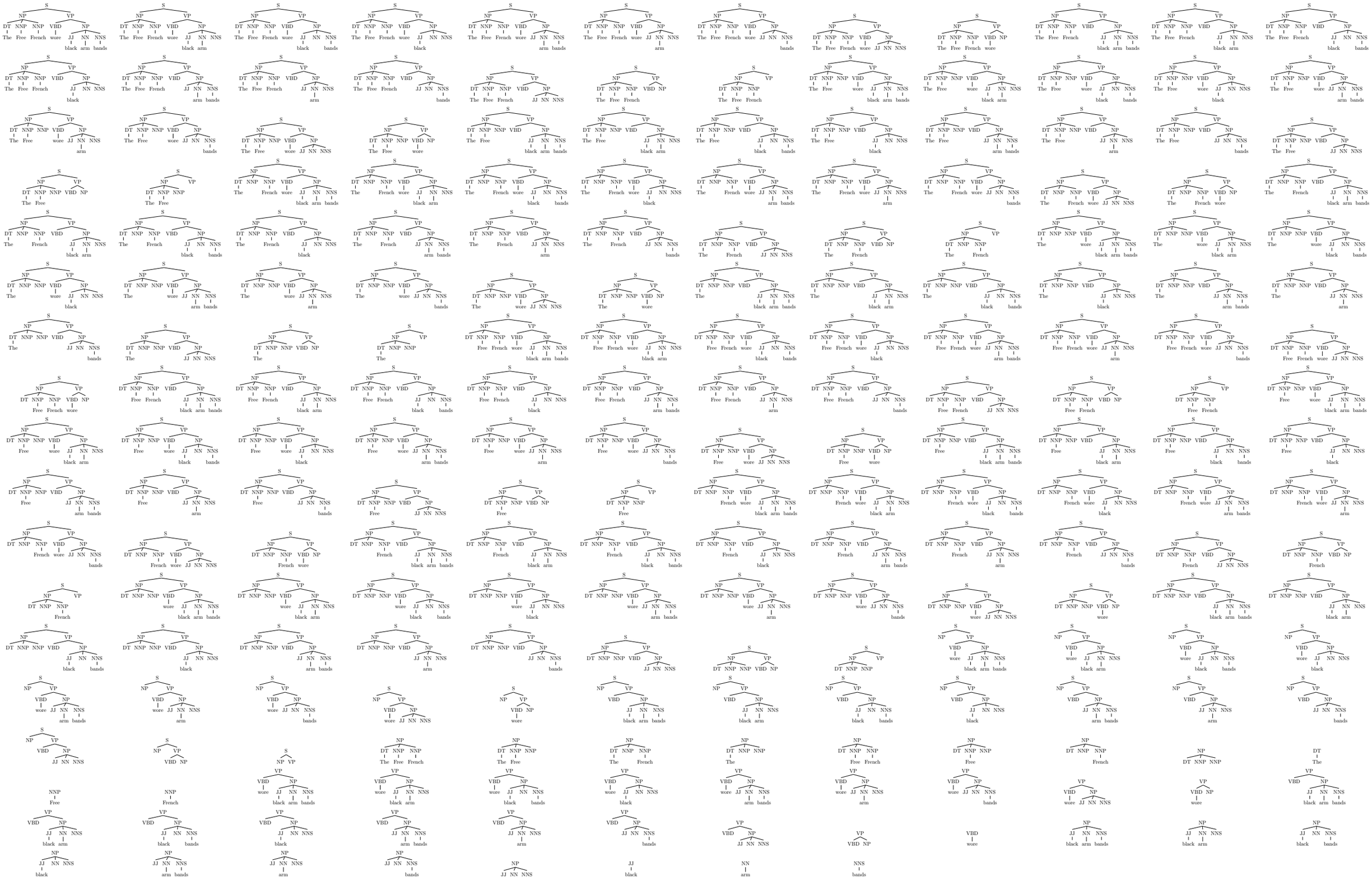
Data Oriented Parsing (DOP)



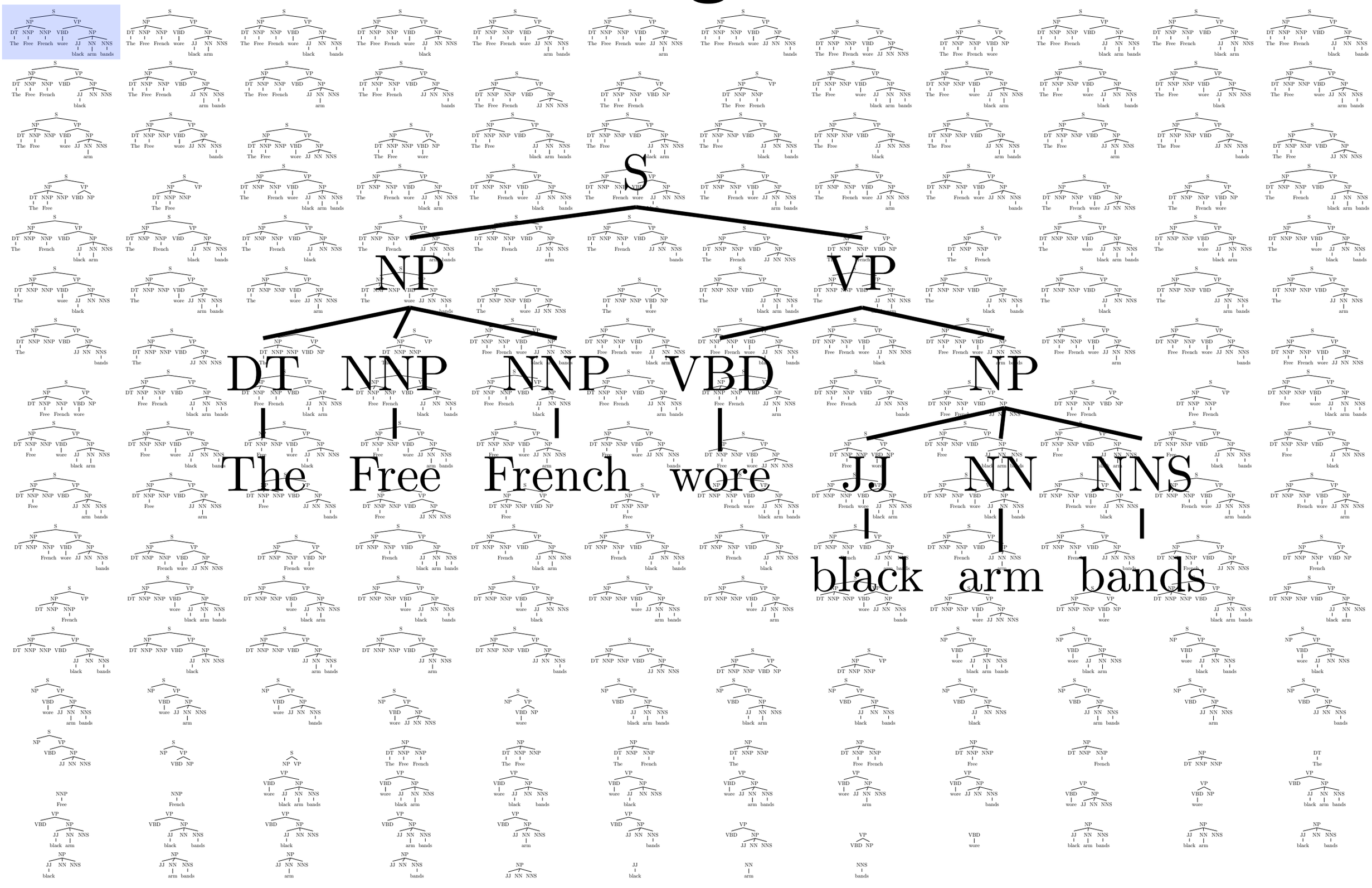
Data Oriented Parsing (DOP)



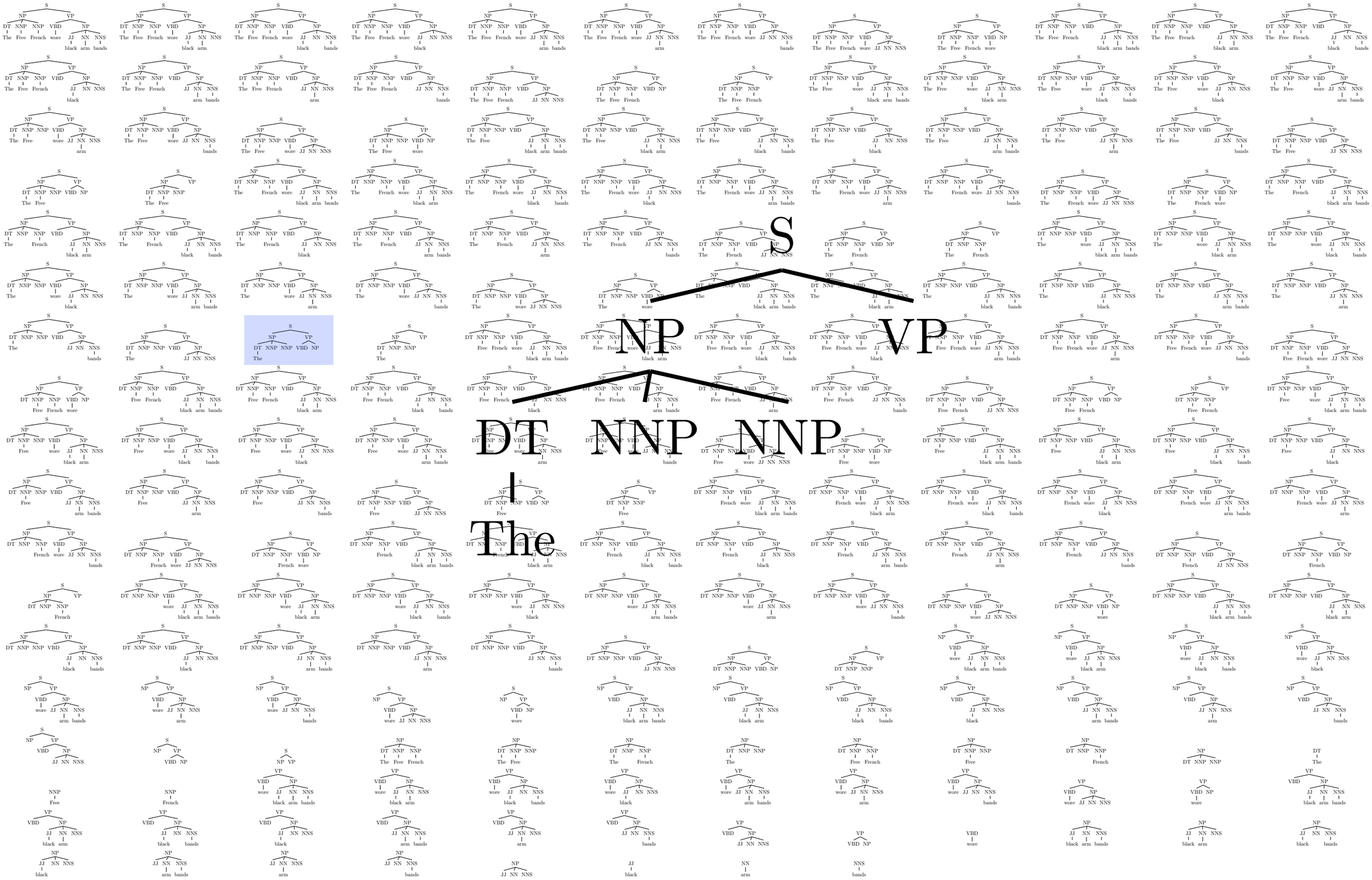
212 fragments



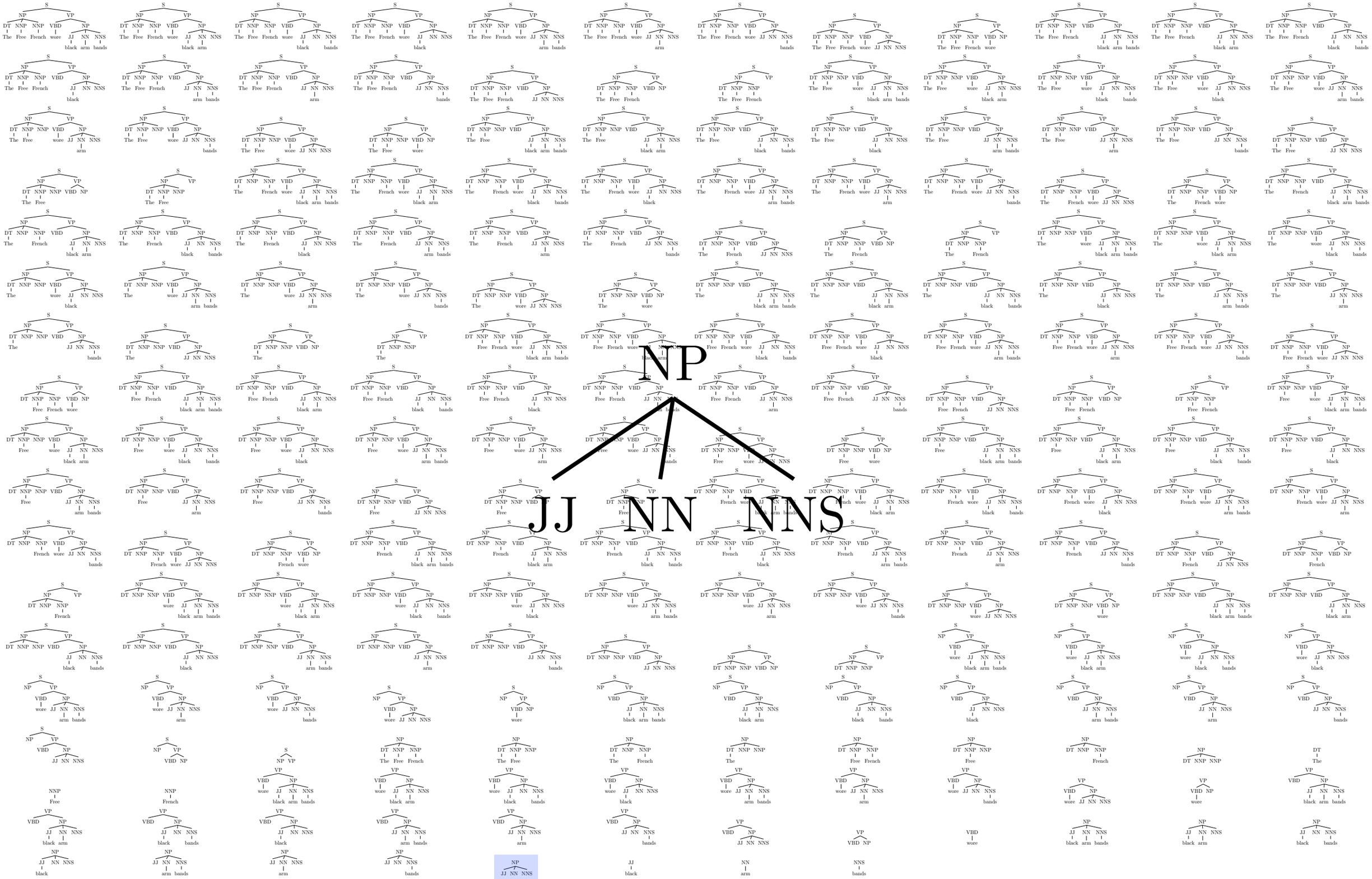
212 fragments



212 fragments



212 fragments



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

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)

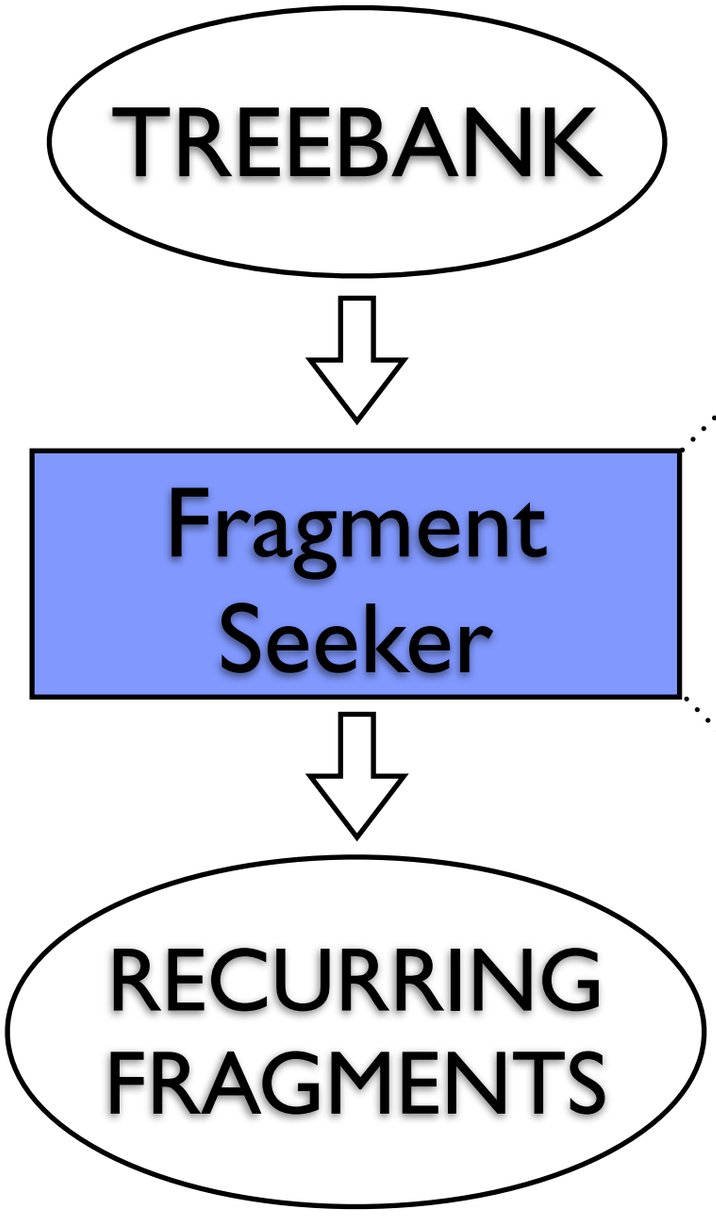
- Based on **Tree Kernels** (Collins and Duffy, 2001; Moschitti 2006)
- Dynamic programming
- Original idea: compute the similarity between two trees as the number of fragments they have in common.
- Current idea: we are not only interested in a number, we want to extract the shared fragments.
- Available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

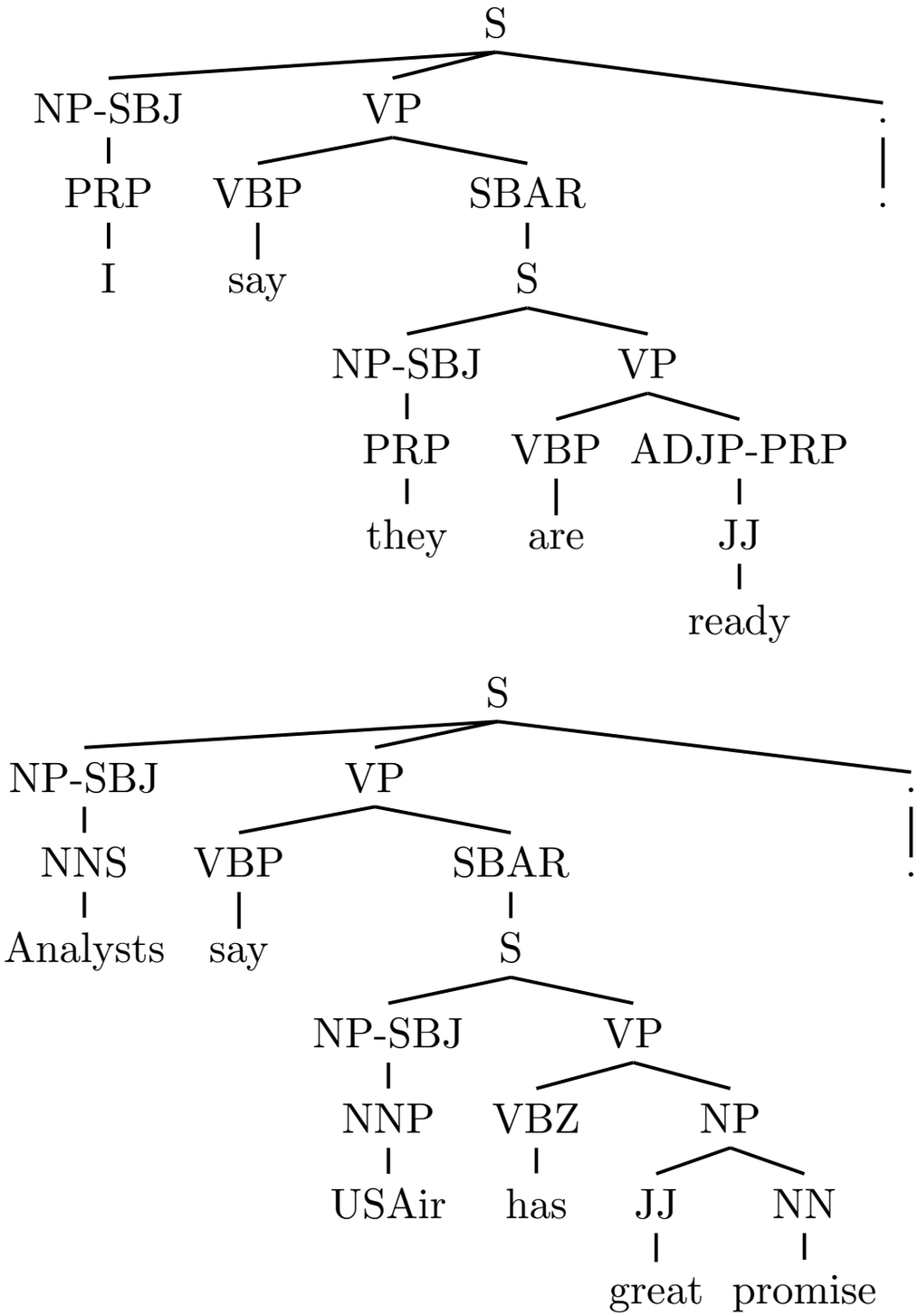


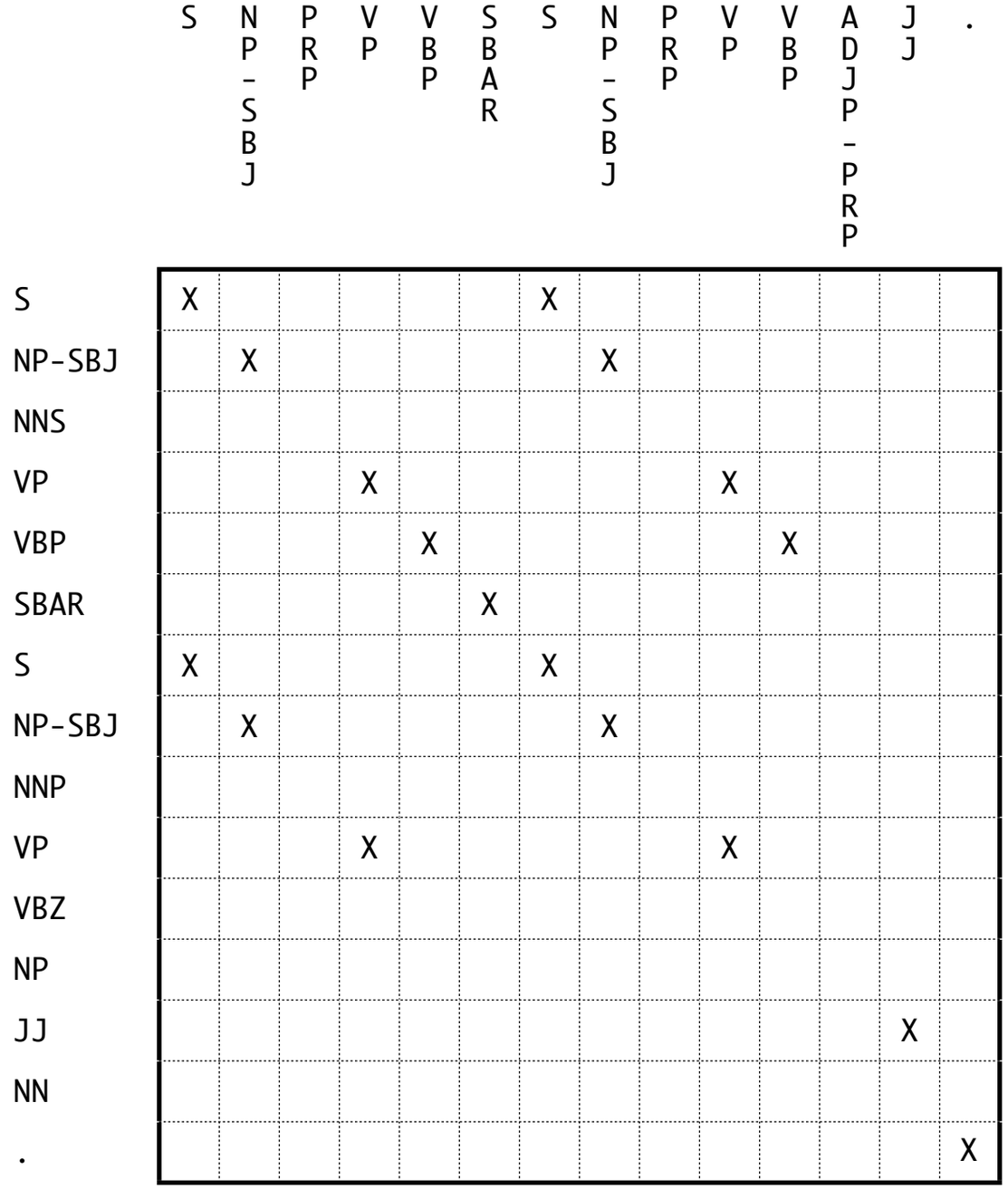
Algorithm: `ExtractFragments(T)`

Input: a corpus `T` of PS trees
Output: a set of fragments and partial fragments
begin

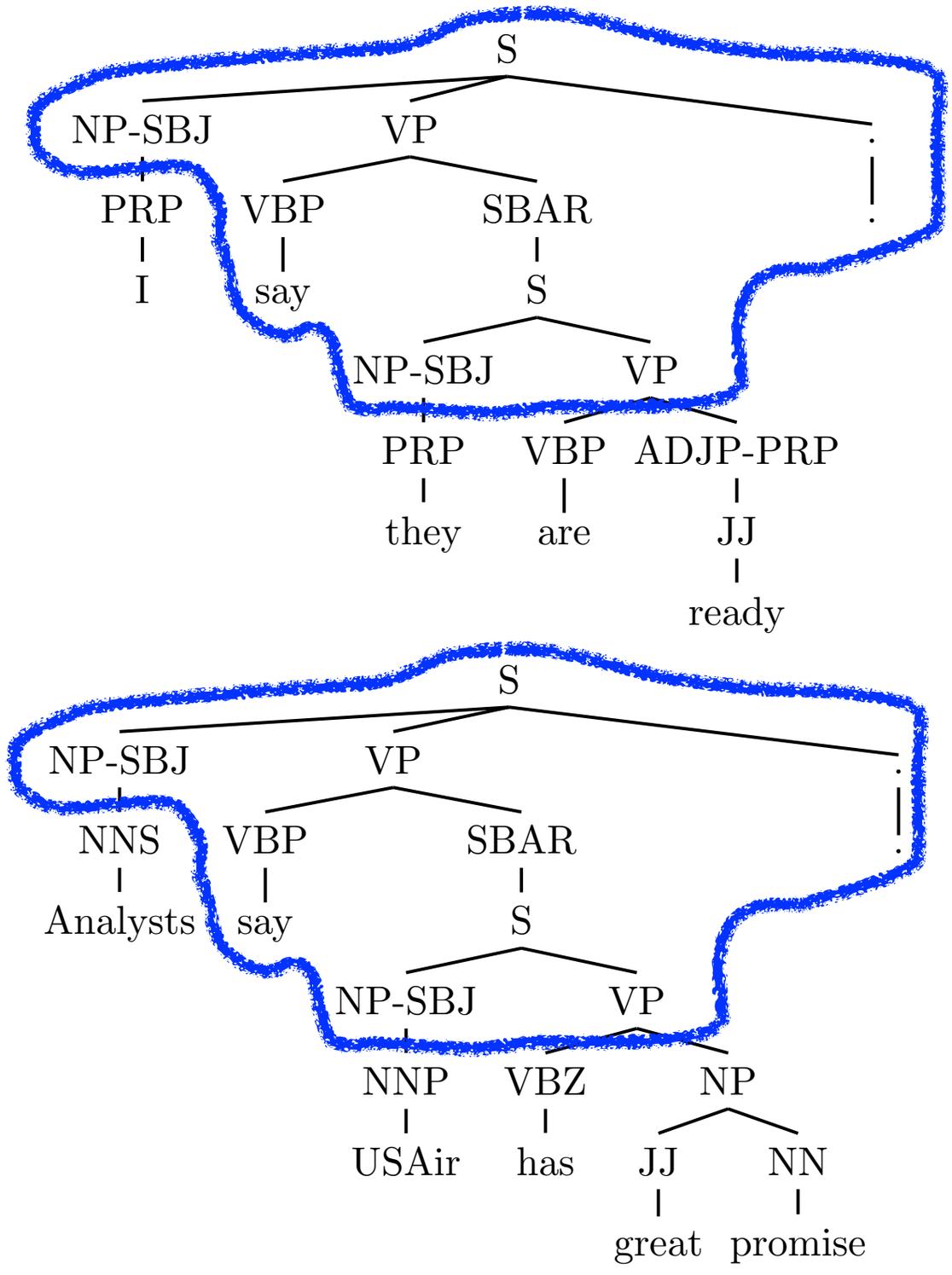
```
FragList: a set of fragments;
foreach tree t_i ∈ T do
    foreach tree t_j ∈ T where t_i ≠ t_j do
        foreach node N_i ∈ t_i do
            foreach node N_j ∈ t_j do
                FragList.addAll(ExtractMaxFragment(N_i, N_j));
return FragList;
```

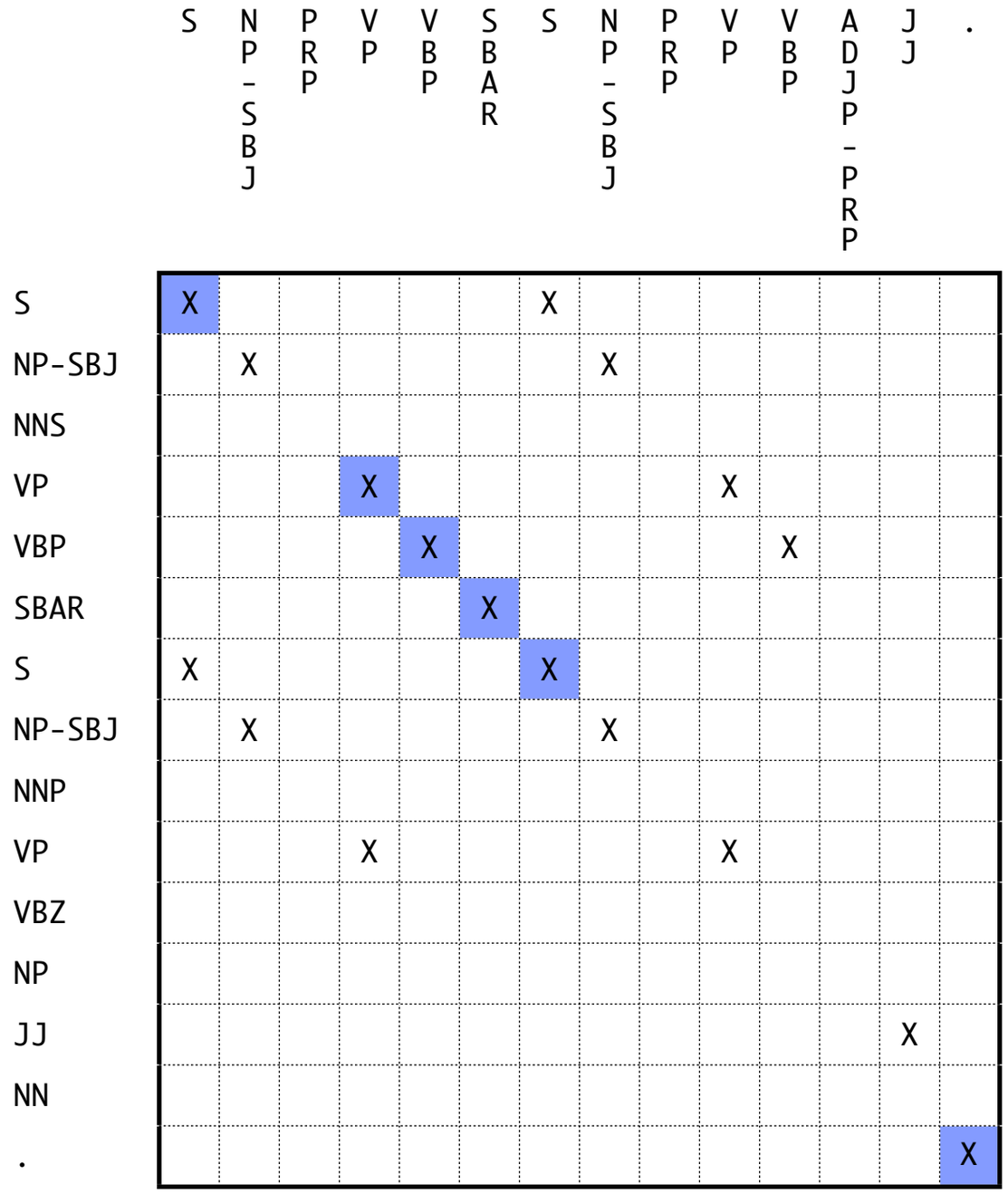
Fragment Seeker



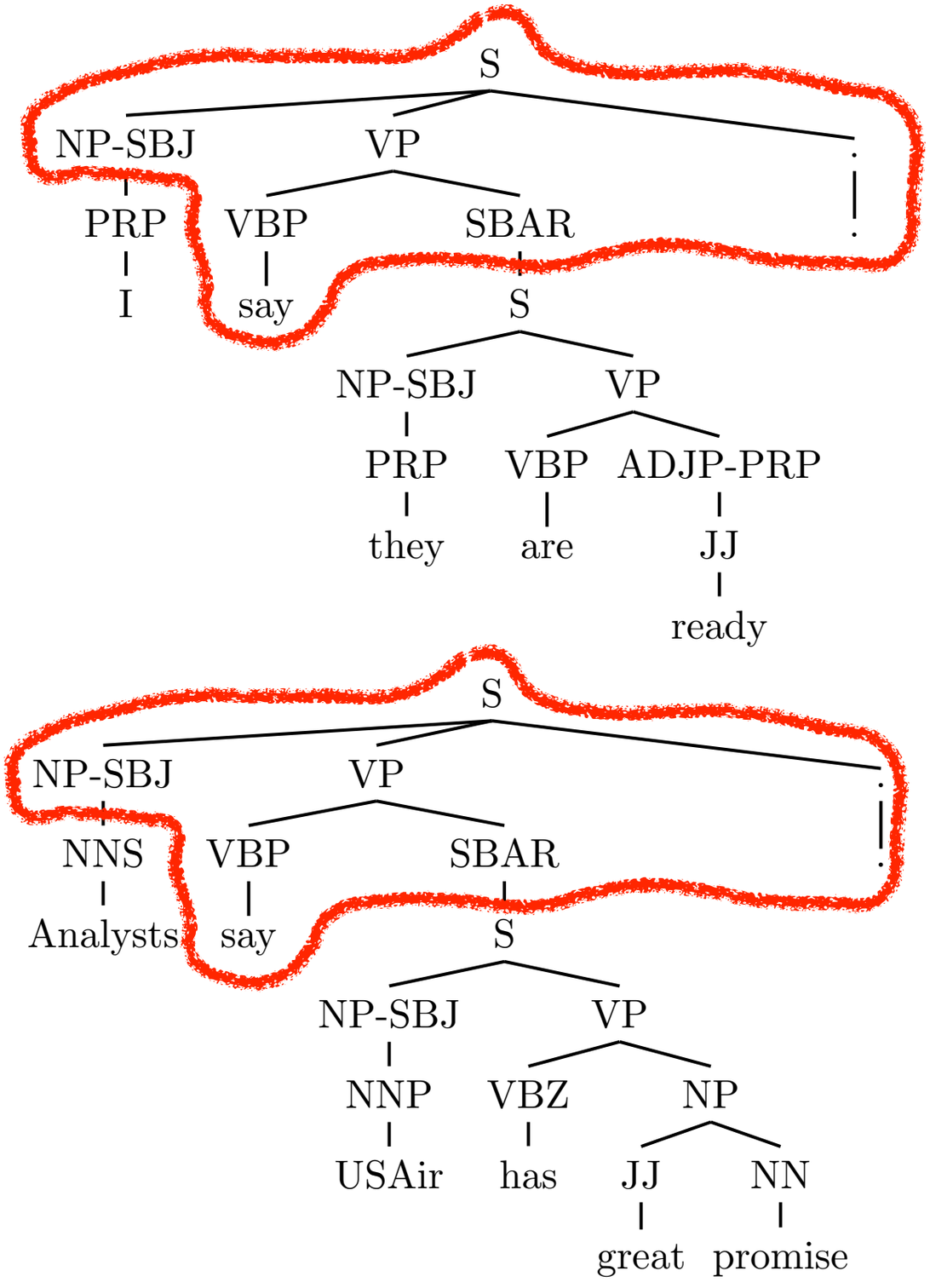


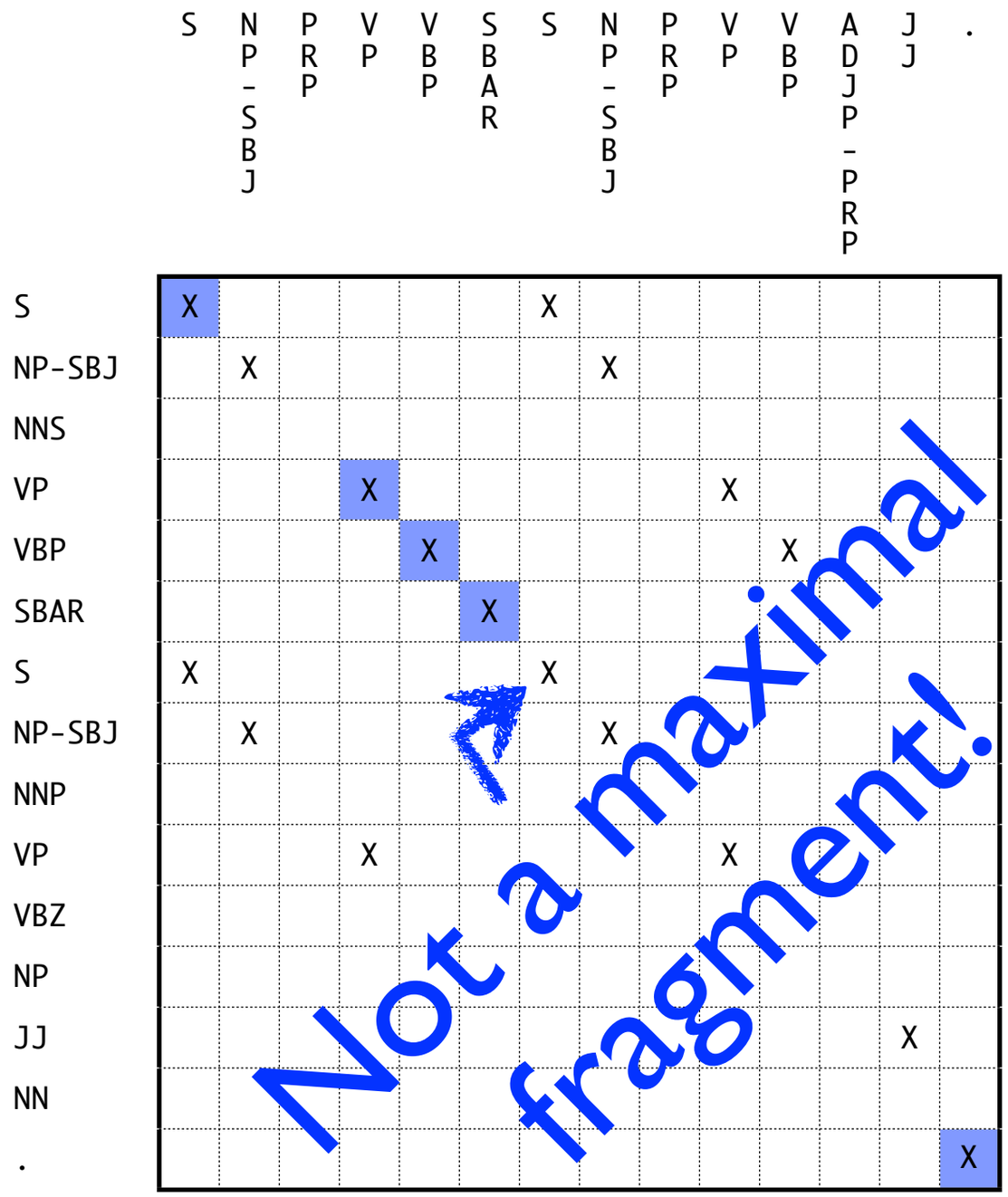
Fragment Seeker





Fragment Seeker

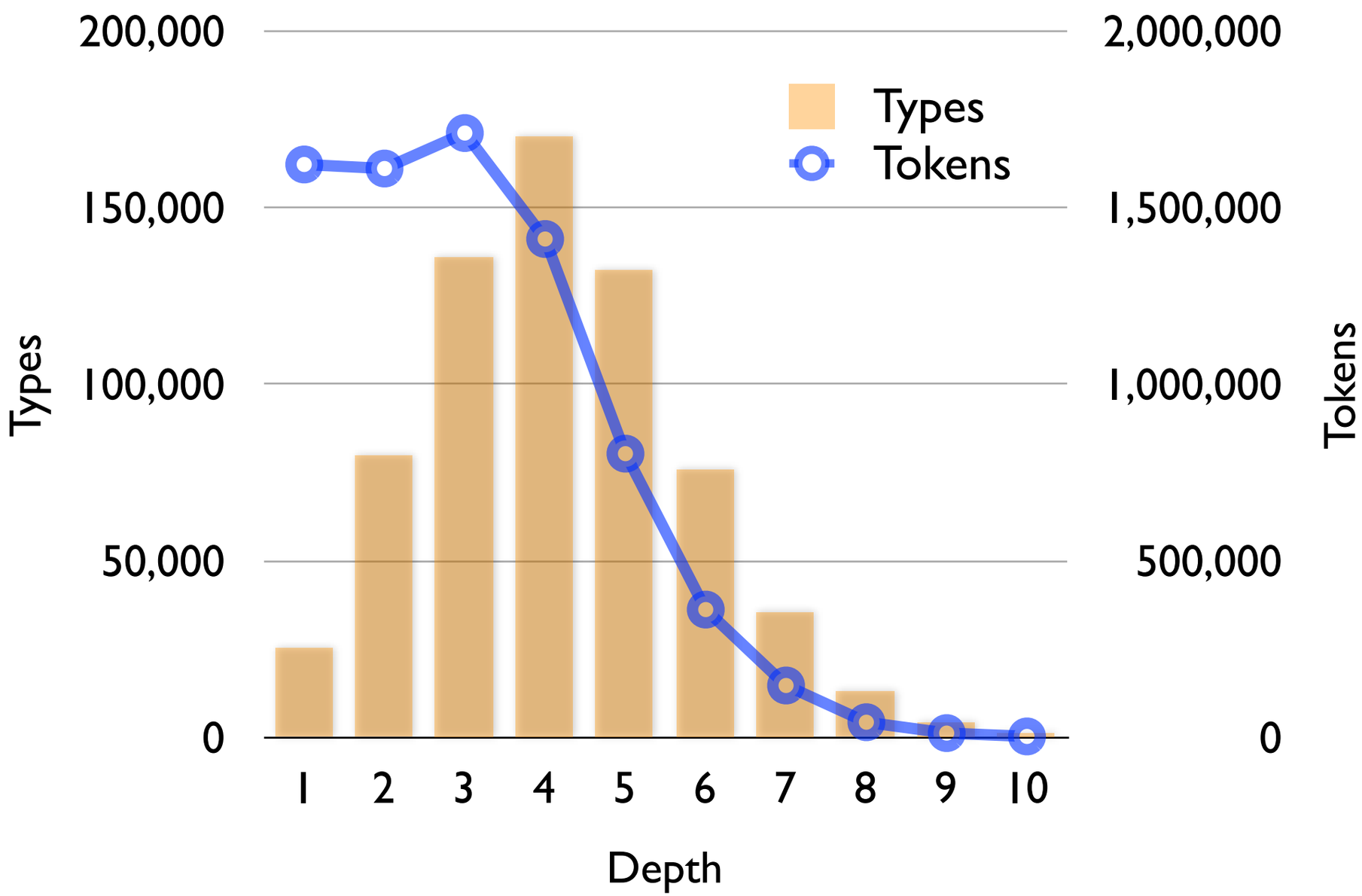




Quantitative Analysis

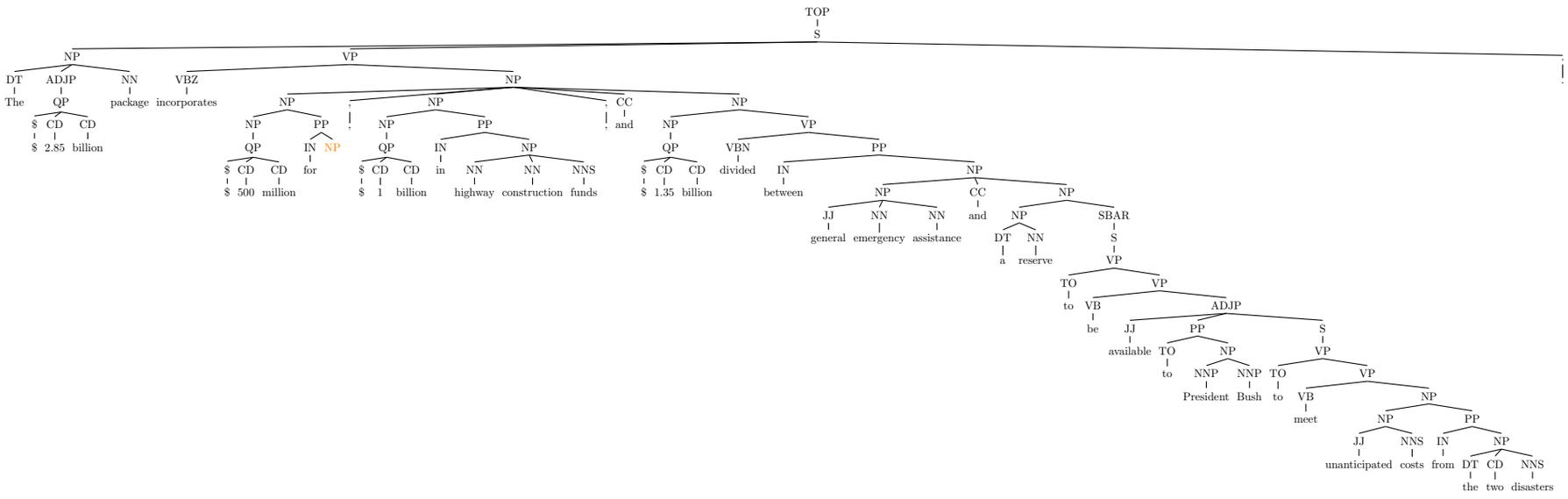
- **Treebank**: WSJ 02-21 (39,832 sentences)
- Recurring fragments types: 674,747

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



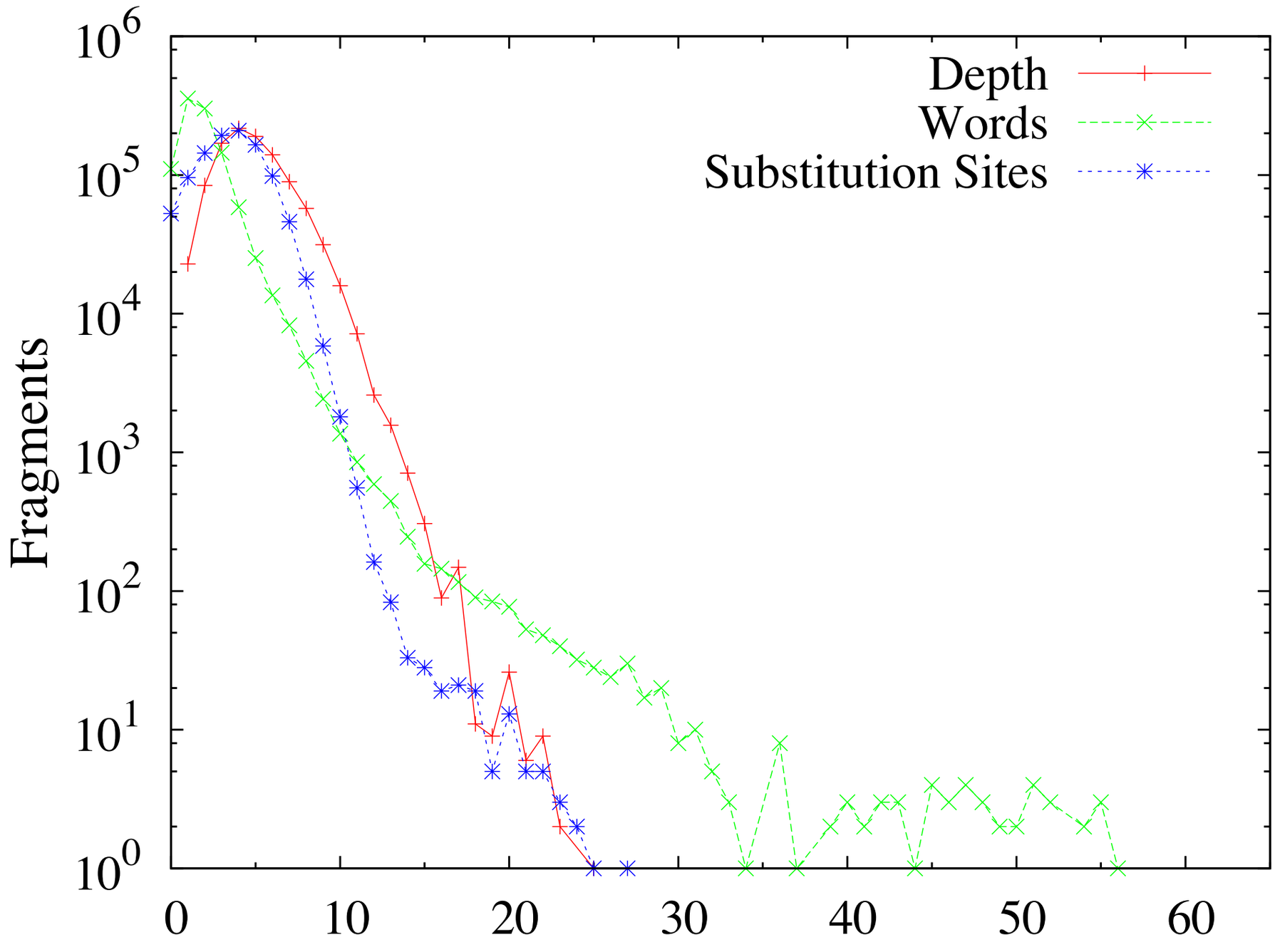
Very Big Fragments

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 .

Fragments Features

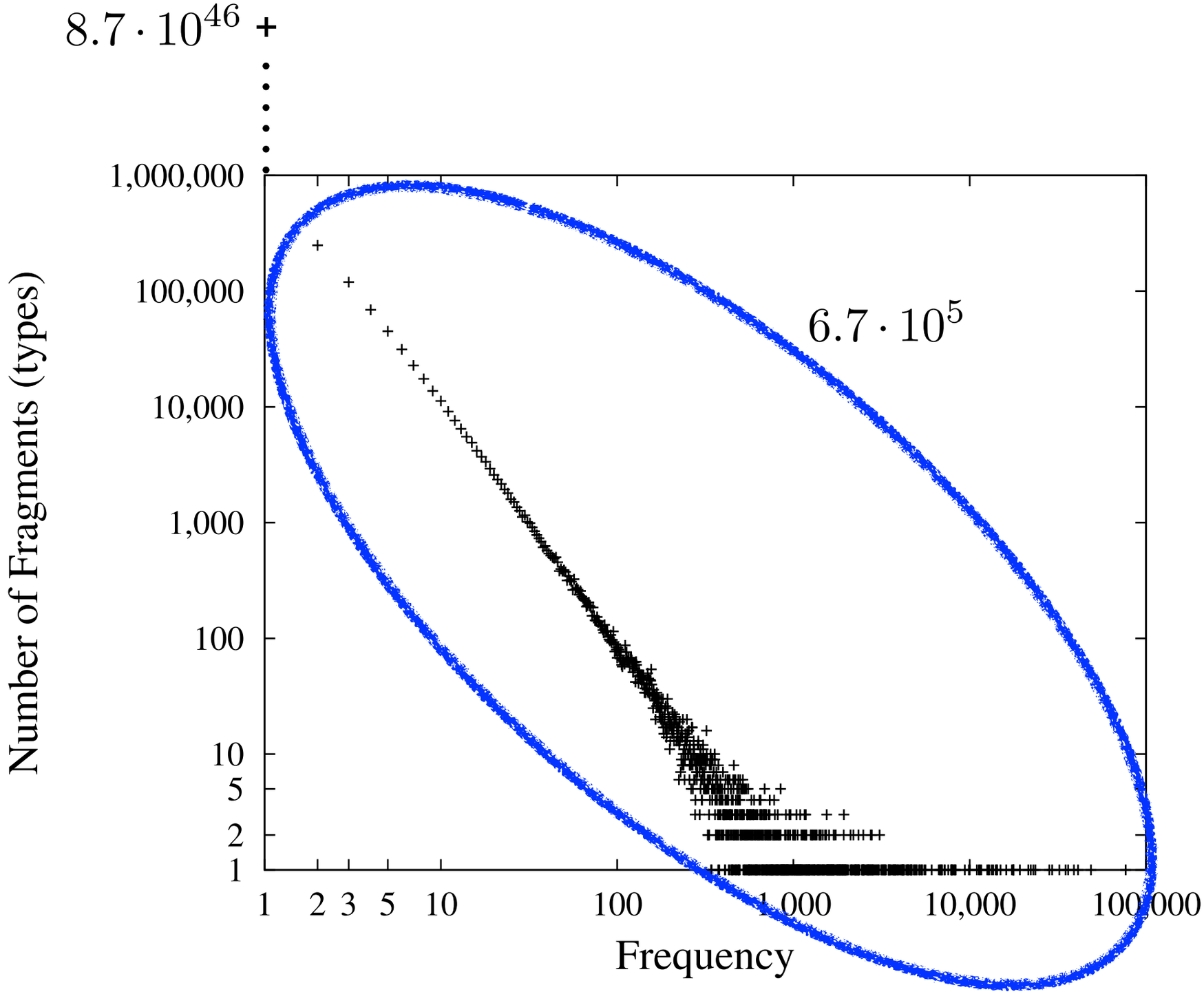


Possible Applications

- Corpus Analysis
- Treebank correction
- Argument Adjunct Distinction
- Parsing
- ...

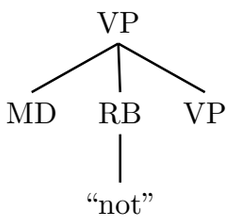
Fragments Distribution

Zipf's law

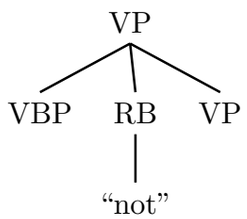


Qualitative Analysis

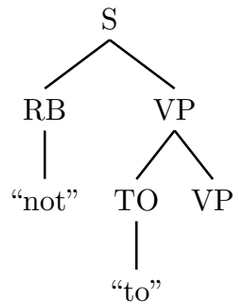
1299



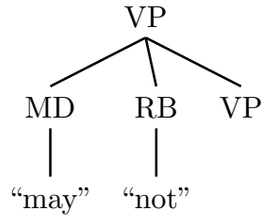
182



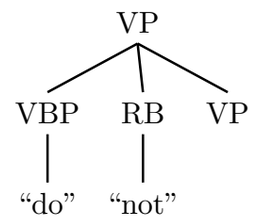
108



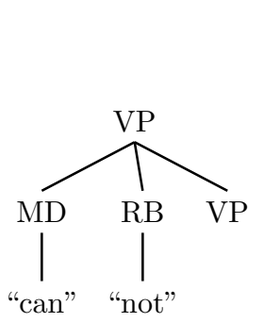
66



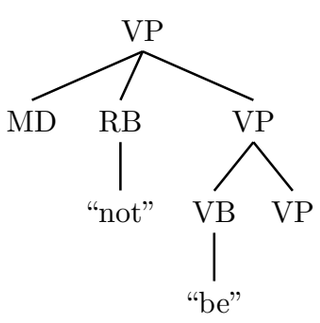
49



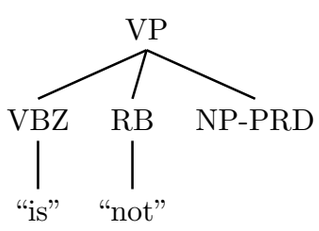
40



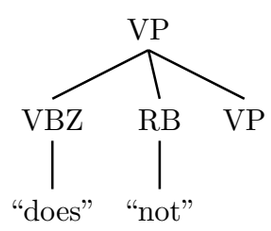
40



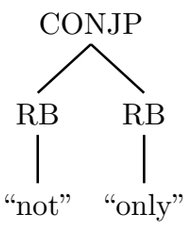
36



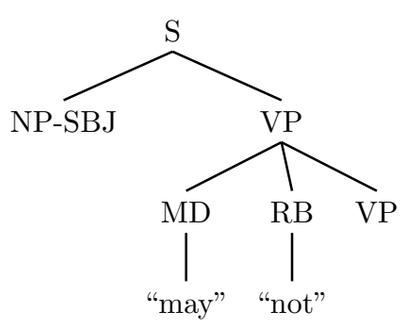
32



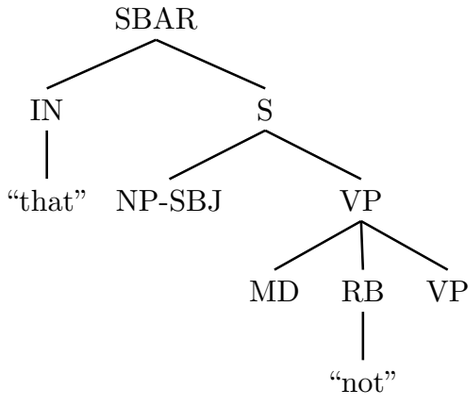
32



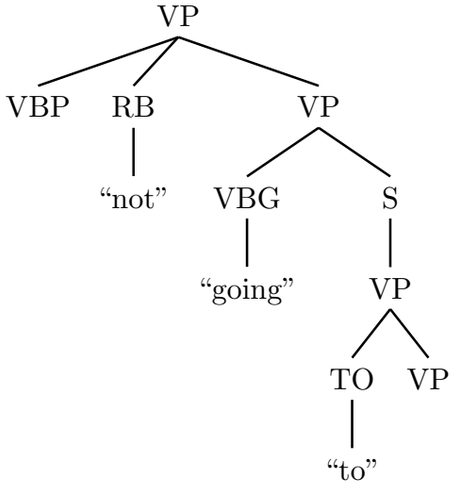
31



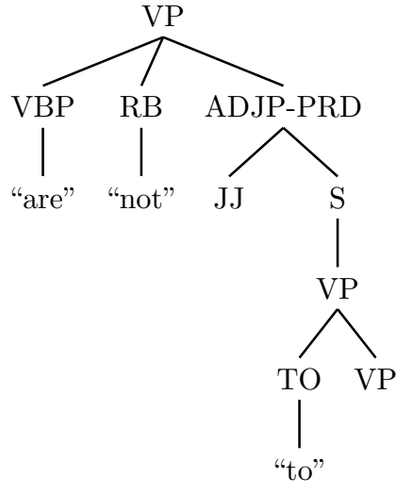
25



20



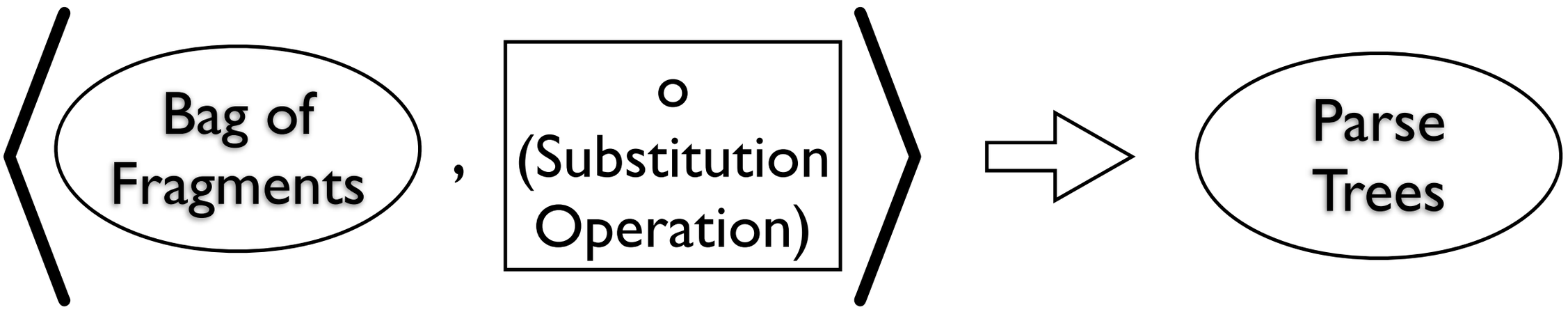
14

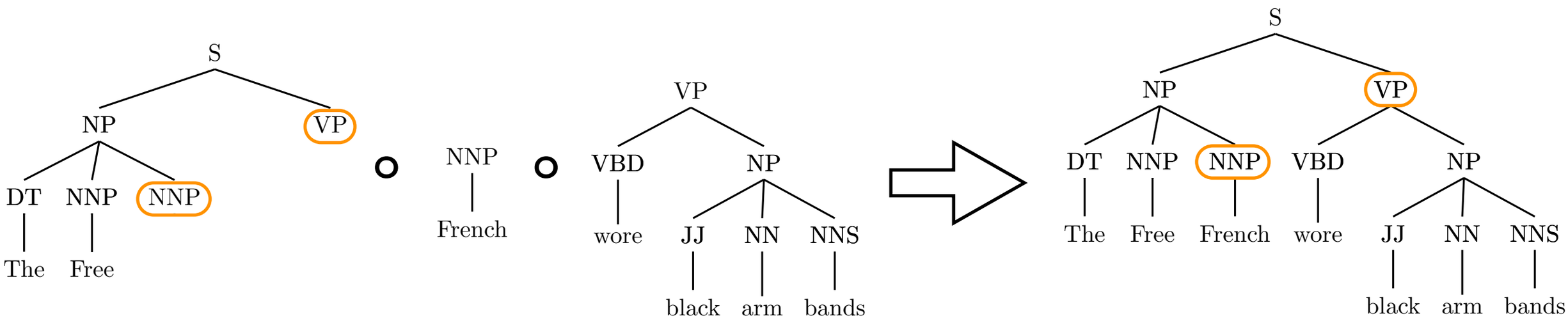


3

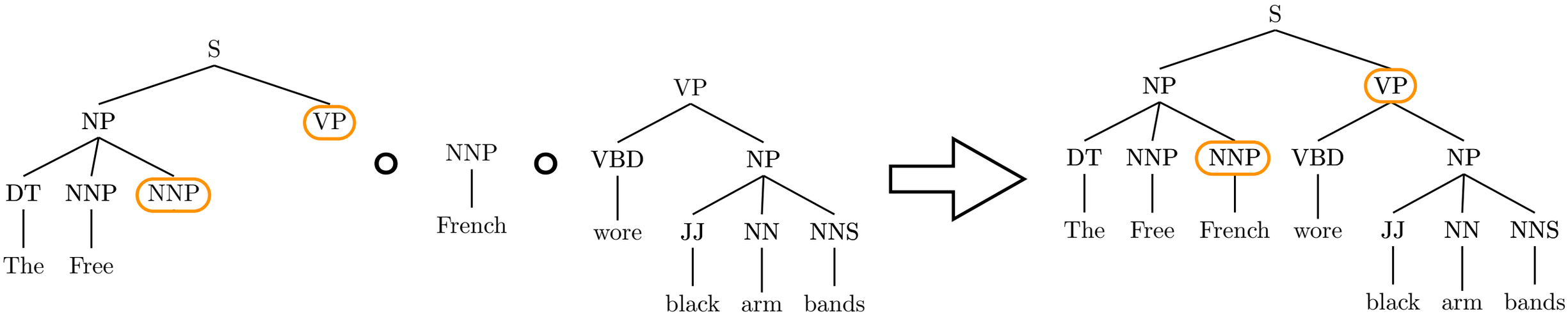
Parsing with Double-DOP

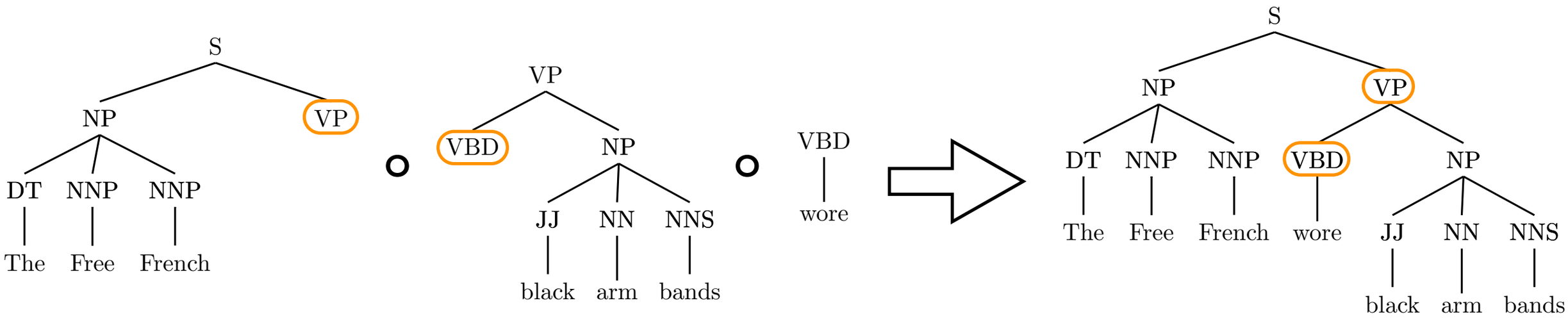
Generating Trees with DOP



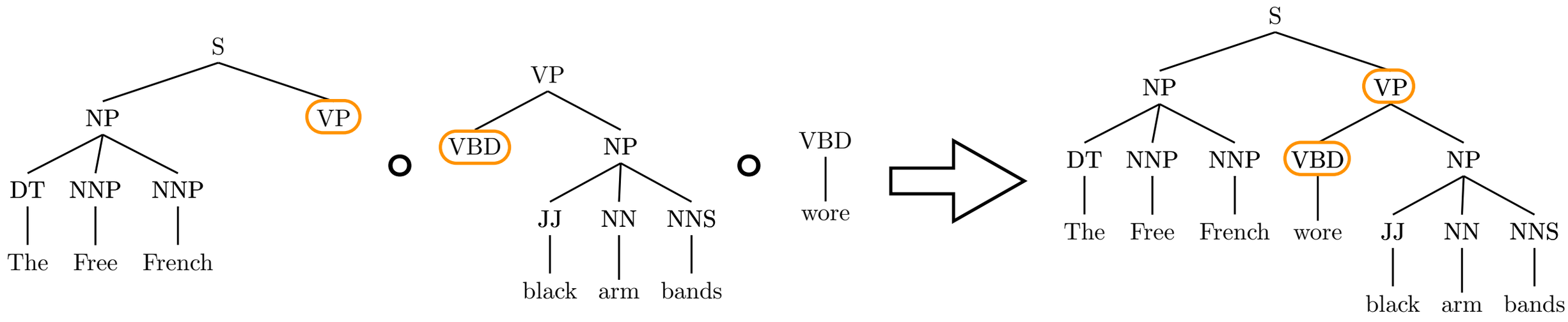


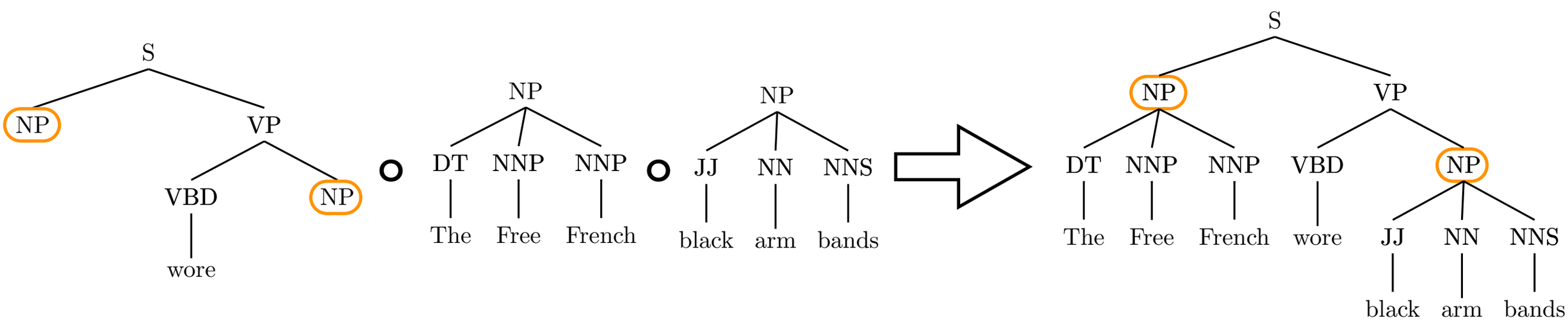
Generating Trees with DOP





Generating Trees with DOP





Probabilistic DOP models

τ : extracted fragment

$f(\tau)$: frequency assigned to fragment τ

$F(\tau)$: relative frequency of a fragment

$$F(\tau) = \frac{f(\tau)}{\sum_{\substack{\tau':root(\tau')\\ =root(\tau)}} f(\tau')}$$

d_t : a derivation, i.e. $\langle \tau_1, \tau_2, \ldots, \tau_n \rangle$ such that $\tau_1 \circ \tau_2 \circ \ldots \circ \tau_n \Rightarrow t$

$P(d_t)$: probability of a derivation generating t

$$P(d_t) = \prod_{\tau_i \in d} F(\tau_i)$$ → MPD

$P(t)$: probability of a tree t

$$P(t) = \sum_{d_j \in \delta(t)} P(d_j) = \sum_{d_j \in \delta(t)} \prod_{\tau_i \in d} F(\tau_i)$$ → MPP

Probabilistic DOP models

What probability model do we choose?

1. Which estimate do we choose (RFE, uniform, ...)?
2. Which objective we want to maximize (MPD, MPP, MCP)?
   - Which metrics we want to target (F1, Exact Match)?

- R. Bod. A Computational Model Of Language Performance: Data Oriented Parsing. COLING 1992.
- J. Goodman. Efficient algorithms for parsing the DOP model. 1996.
- R. Bonnema, P. Buying, and R. Scha. A New Probability Model for Data Oriented Parsing. 1999.
- R. Bod. What is the minimal set of fragments that achieves maximal parse accuracy? ACL, 2001.
- M. Johnson. The DOP estimation method is biased and inconsistent. CL, 2002.
- K. Sima'an and L. Buratto. Backoff parameter estimation for the DOP model. ECML, 2003.
- W. Zuidema. What are the productive units of natural language grammar?: a DOP approach to the automatic identification of constructions. CoNLL-X '06
- W. Zuidema. Parsimonious Data-Oriented Parsing. EMNLP-CoNLL2007.
- M. Post and D. Gildea. Bayesian learning of a tree substitution grammar. ACL-IJCNLP 2009.
- M. Bansal and D. Klein. Simple, accurate parsing with an all-fragments grammar. ACL 2010.

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.

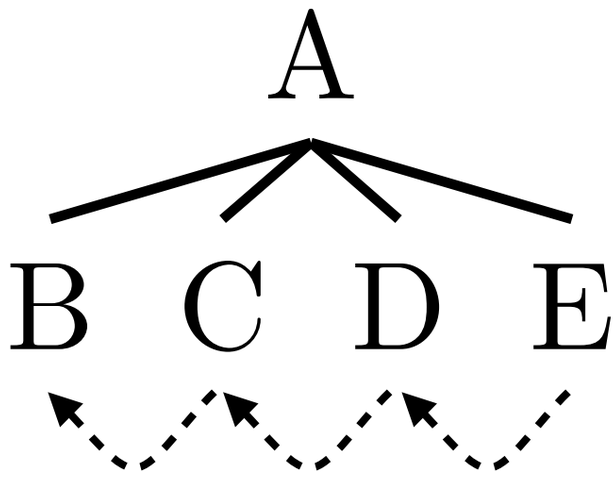
- Feature Set :
  1. `suffix`
  2. `isFirstWord`
  3. `isCapitalized`
  4. `hasDash`
  5. `hasForwardSlash`
  6. `hasDigit`
  7. `hasAlpha`

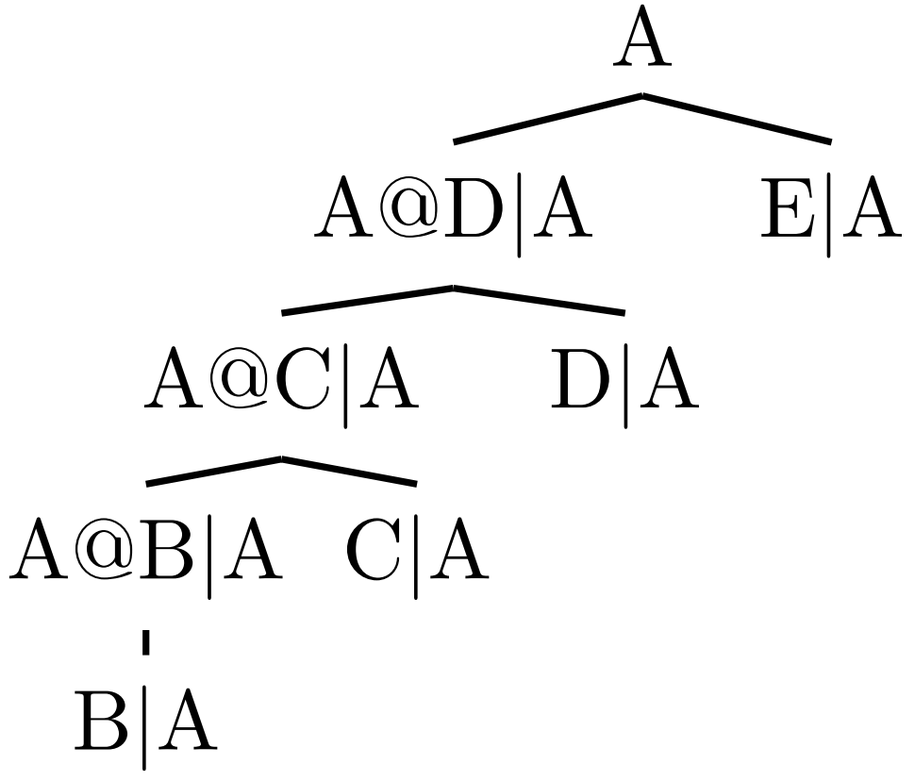
Lex Smoothing :

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

Left Binarization

(P=1,V=1)

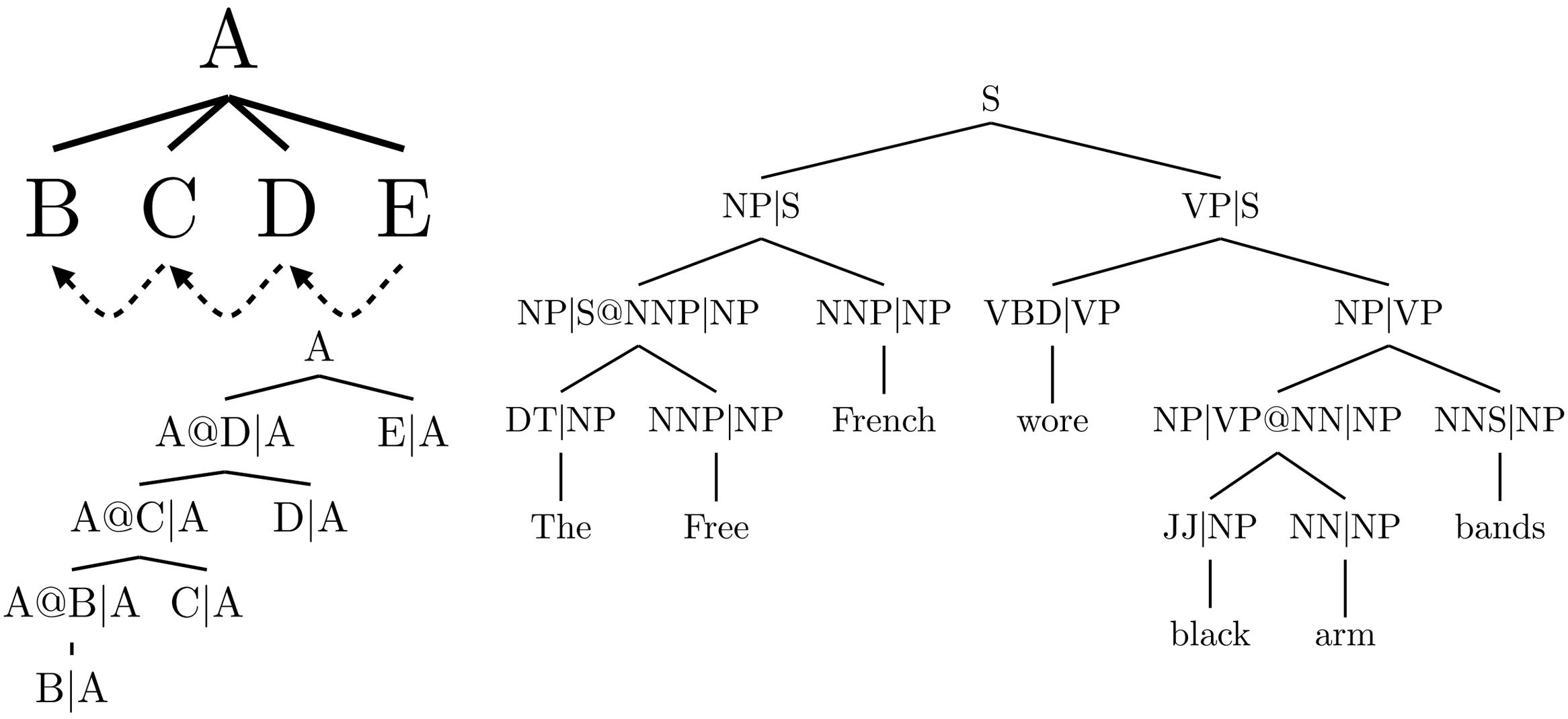




- K. Sima'an. Tree-gram parsing lexical dependencies and structural relations. ACL 2000.
- D. Klein and C. D. Manning. Accurate unlexicalized parsing. ACL 2003.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic cfg with latent annotations. ACL 2005.
- M. Bansal and D. Klein. Simple, accurate parsing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

Left Binarization

(P=1,V=1)

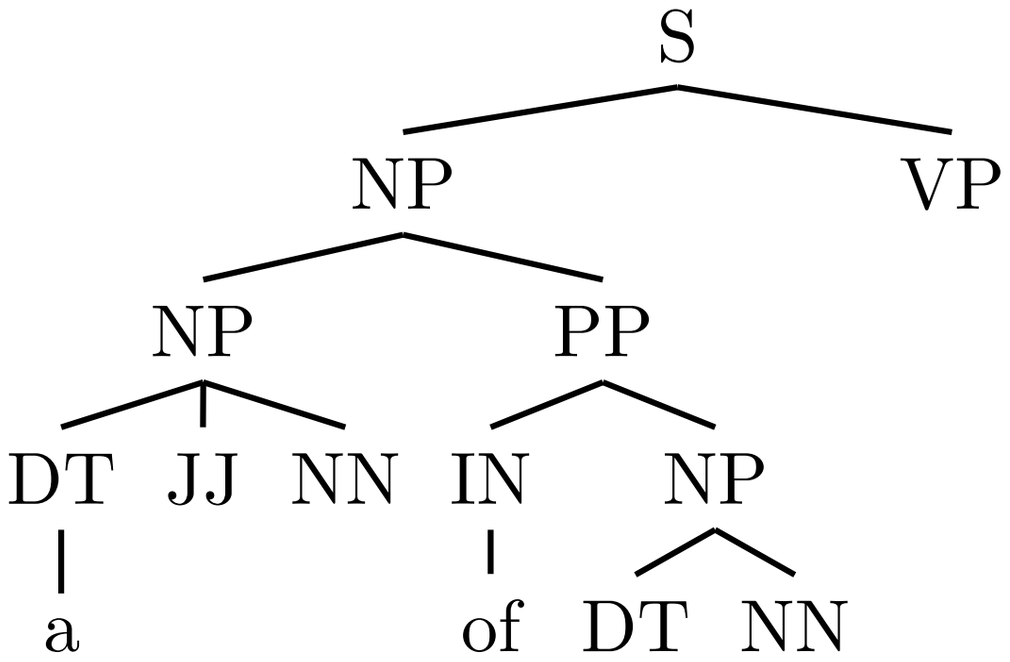


- K. Sima'an. Tree-gram parsing lexical dependencies and structural relations. ACL 2000.
- D. Klein and C. D. Manning. Accurate unlexicalized parsing. ACL 2003.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic cfg with latent annotations. ACL 2005.
- M. Bansal and D. Klein. Simple, accurate parsing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

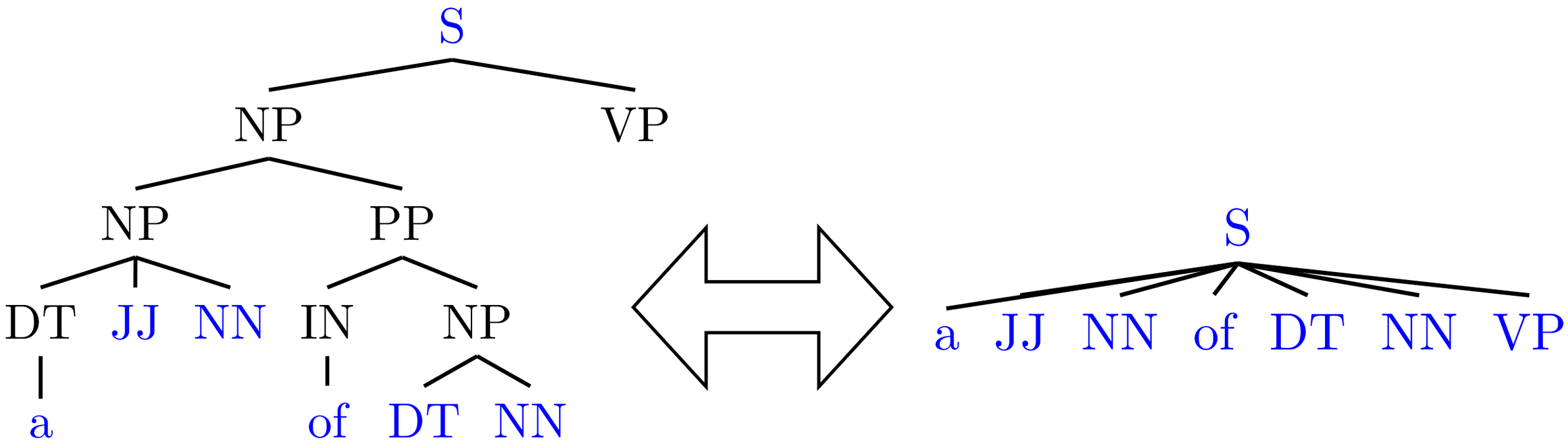
From fragments to CFG rules



f = 11

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

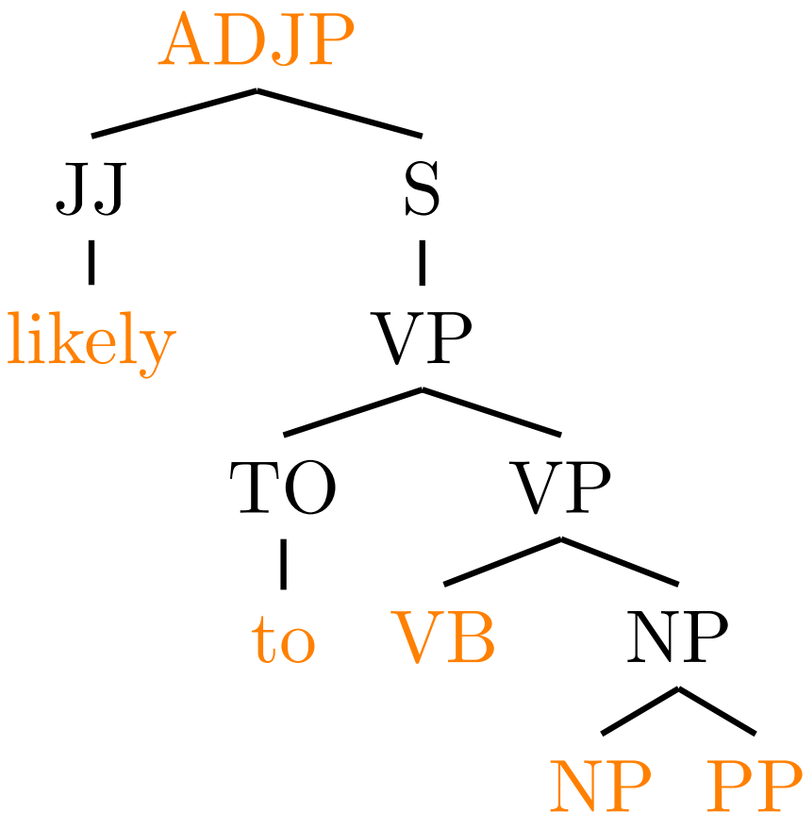
From fragments to CFG rules



f = 11

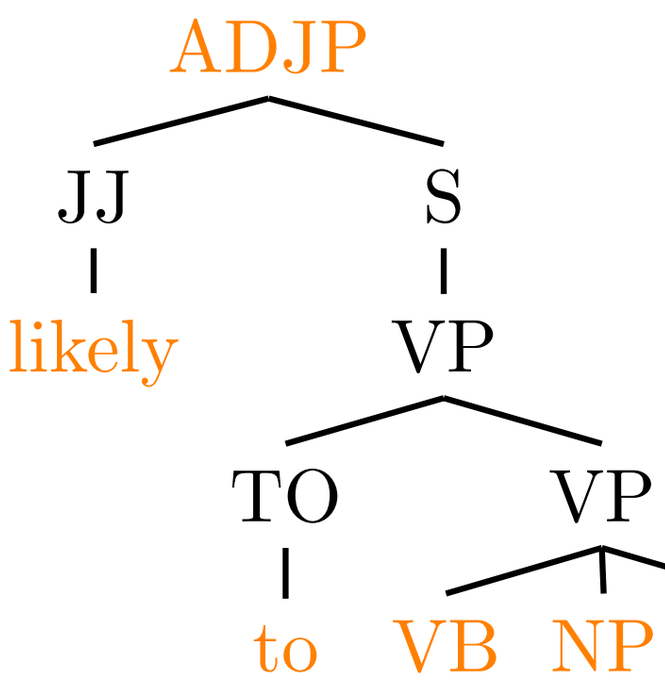
A significant portion of the order will be placed...

Ambiguous Fragments



f=13

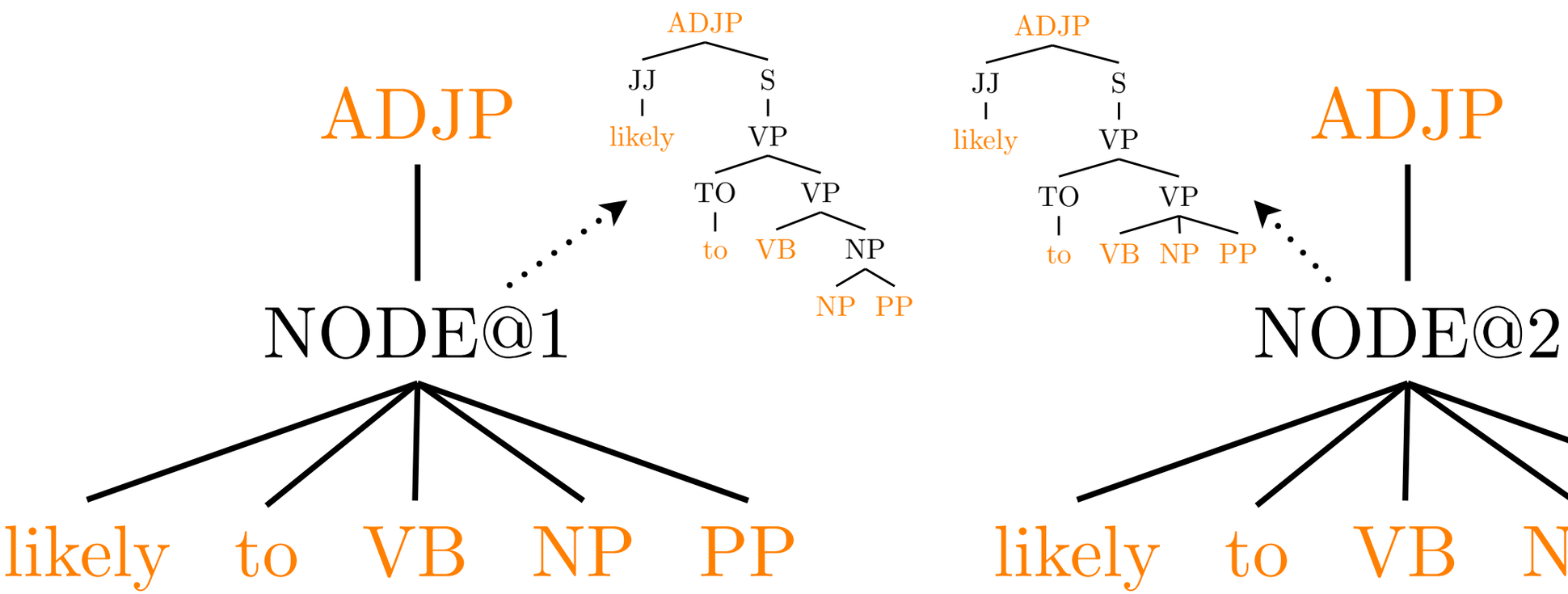
e.g. Likely to trigger an opposition from people



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)

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

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

➡ 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

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

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

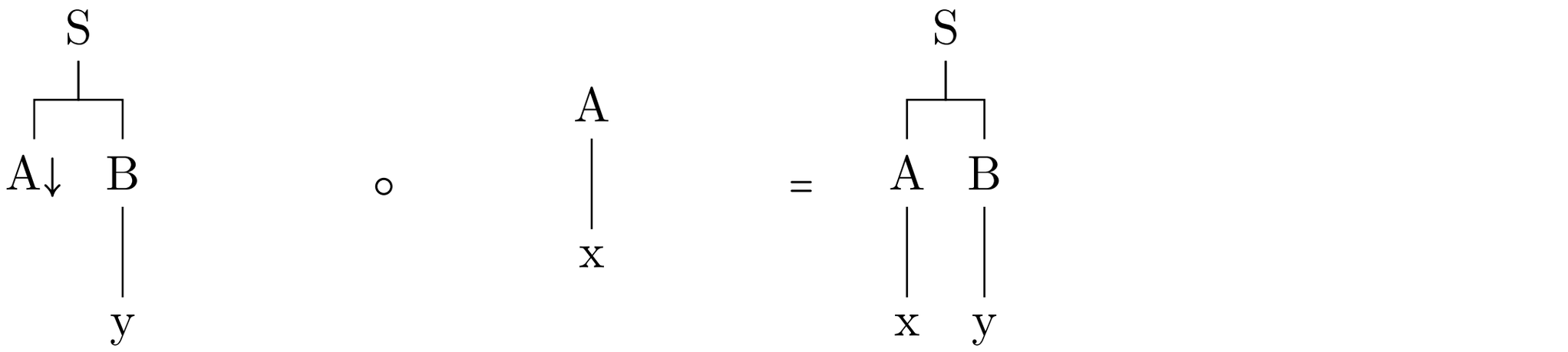
$$= \prod_{t \in \mathrm{treebank}} \sum_{d \in \delta(t)} \prod_{\tau \in d} p(\tau)$$

Re-estimate Fragments Frequencies with Inside-Outside (EM)

- Originally used to estimate CFG rules probabilities to maximize the likelihood of training sentences.
- Here we use it to estimate Fragments probabilities to maximize the likelihood of training tree structures.
- The same idea!
  - Multiple CFG derivations for each sentence.
  - Multiple DOP derivations for each tree.

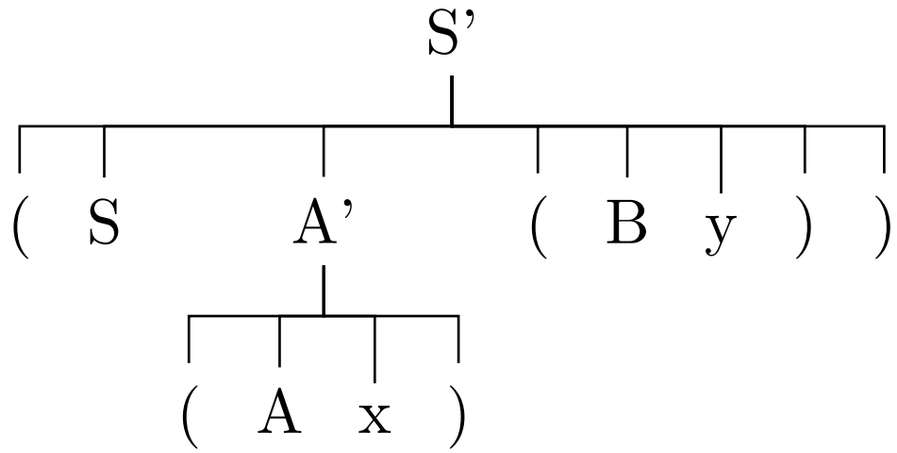
➡ K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. Computer Speech and Language, 4:35–56, 1990.

Expectation Maximization

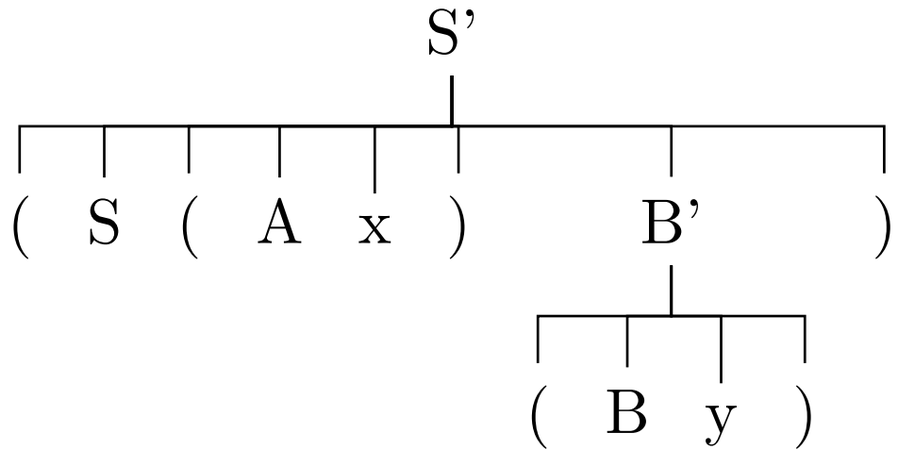


(S A↓ (B y)) ∘ (A x) = (S (A x) (B y))

S’ → (S A’ (B y)) ∘ A’ → (A x) =



S’ → (S (A x) B’) ∘ B’ → (B y) =



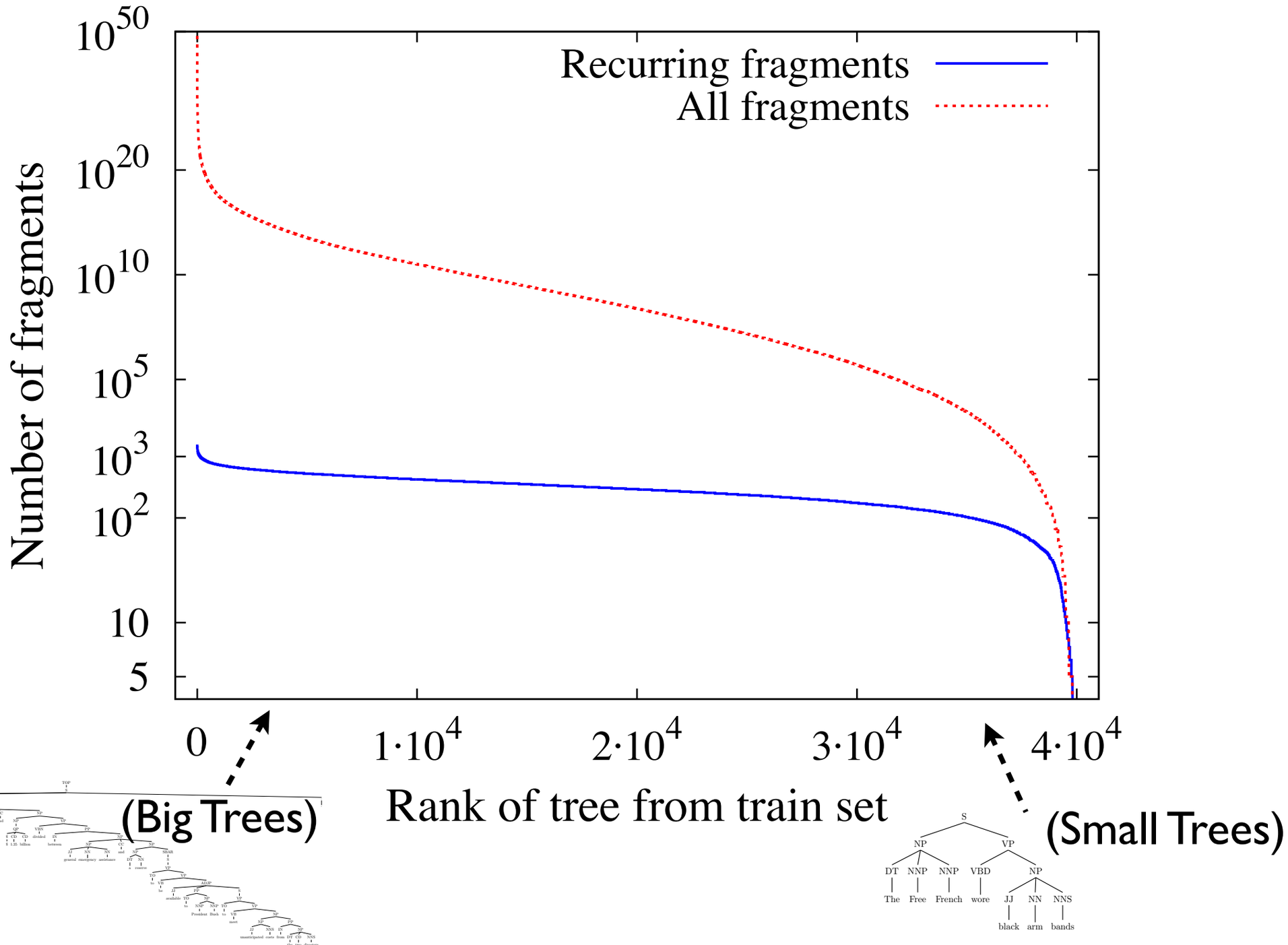
Probability Estimates Evaluation

(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

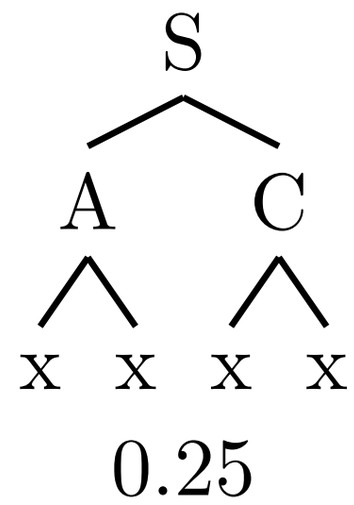
If we are interested in F1 better try to select the parse tree that is most likely to optimize this metric.

- MCP: maximum constituent parse (Goodman, 1996)

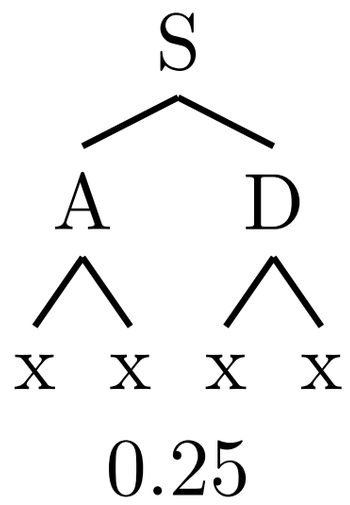
Maximum Constituent Parse

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

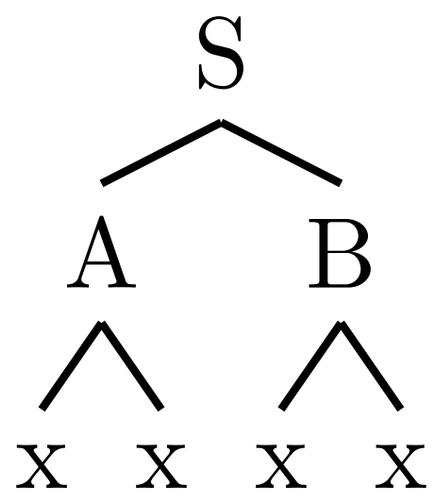
S	→	A C	0.25
S	→	A D	0.25
S	→	E B	0.25
S	→	F B	0.25
A	→	x x	1.0
B	→	x x	1.0
C	→	x x	1.0
D	→	x x	1.0
E	→	x x	1.0
F	→	x x	1.0

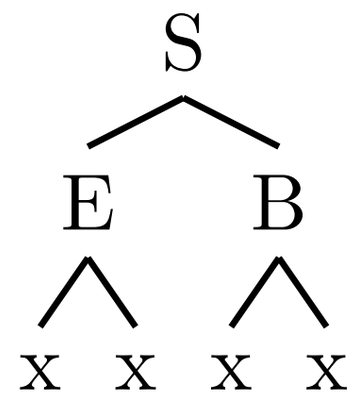


0.25

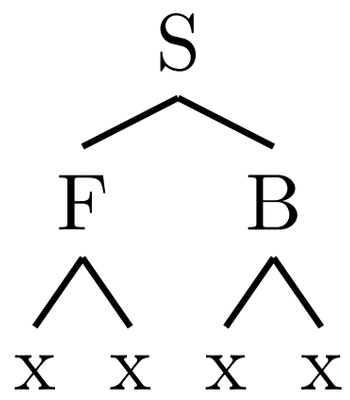


0.25





0.25



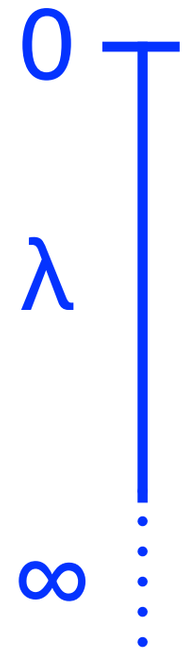
0.25

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

Maximum Constituent Parse

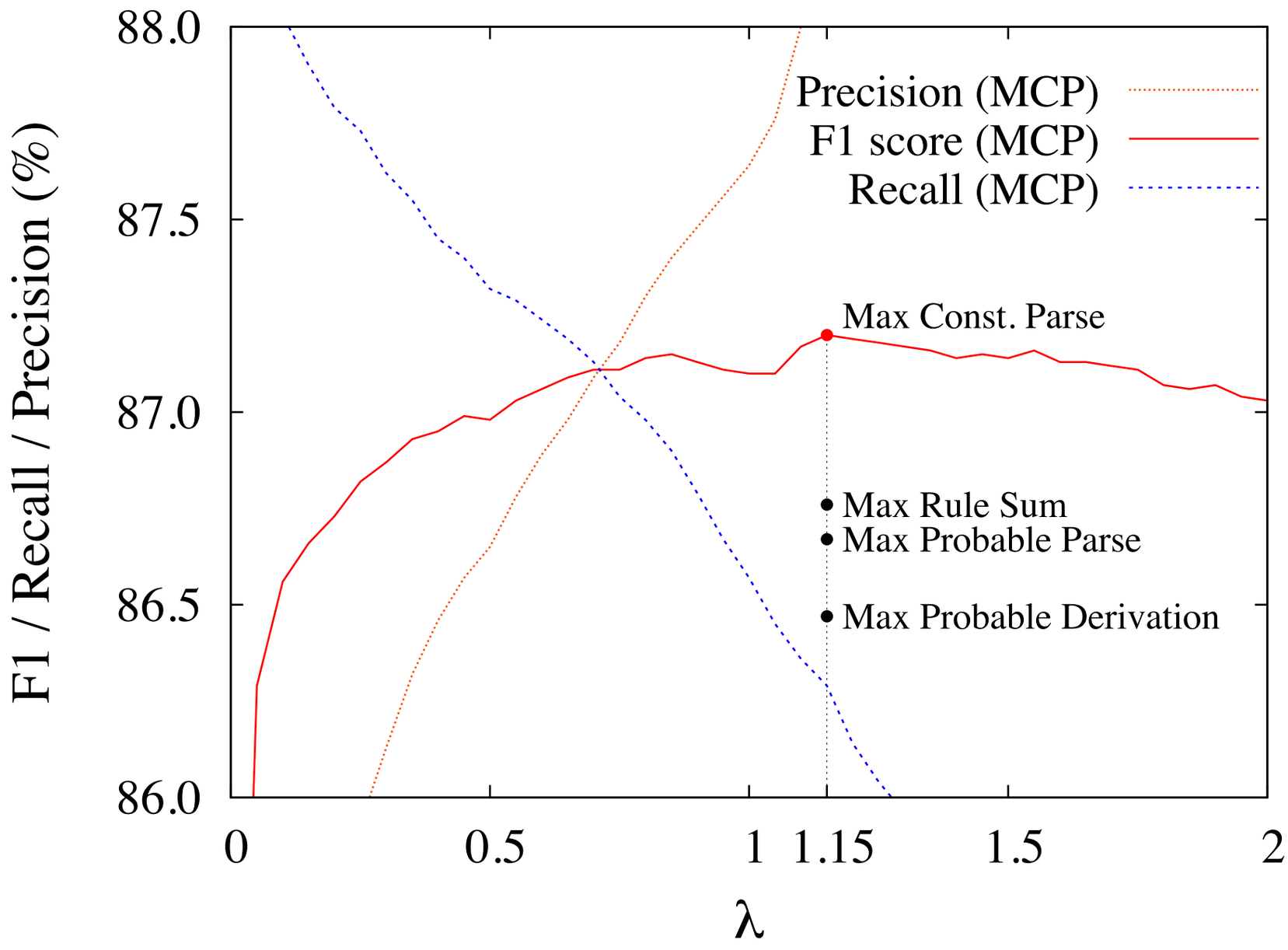
n-ary branching case: $\max(\text{Recall}) \neq \max(\text{Precision})$

- **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)

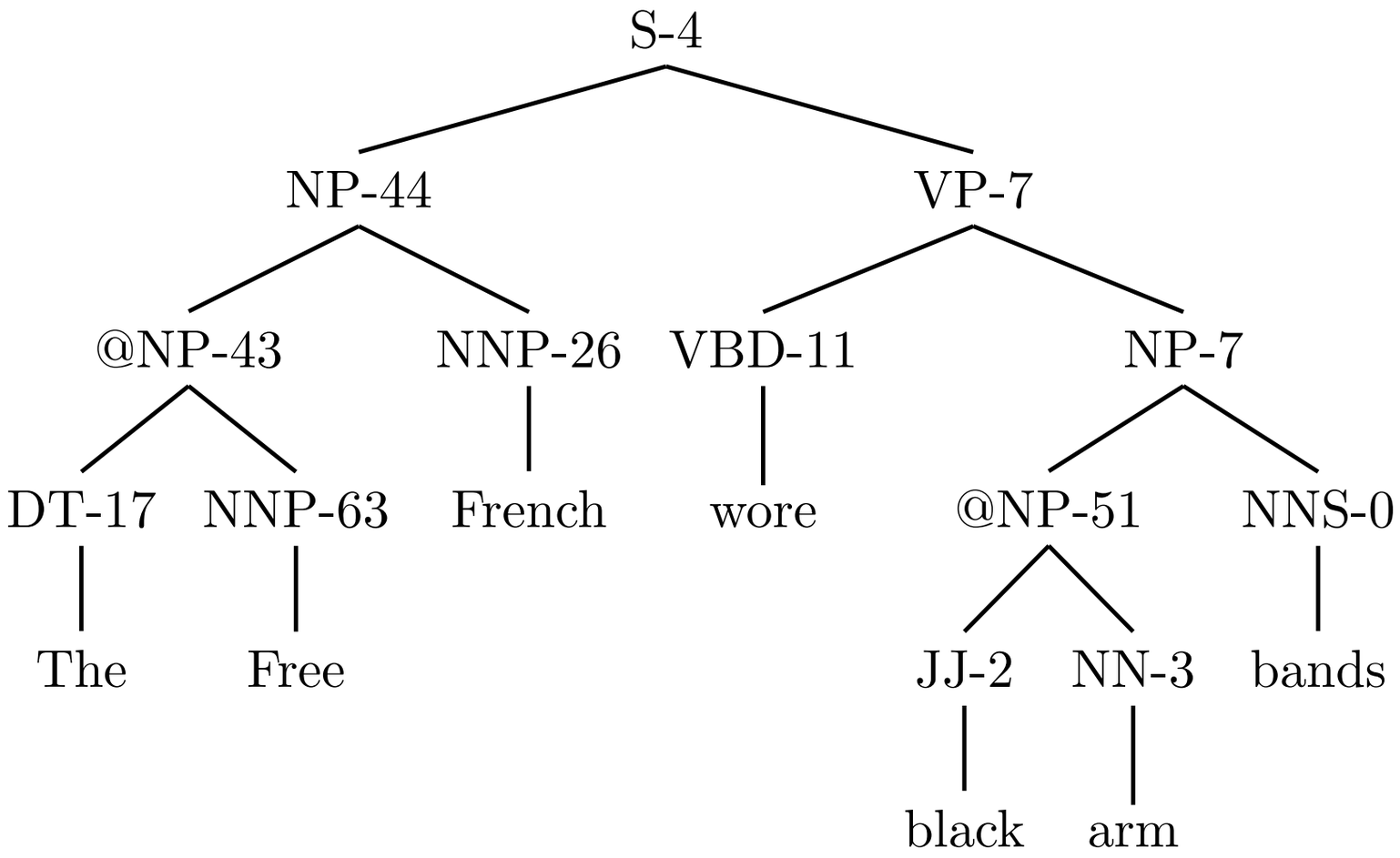


Parsing Results
(WSJ test set)

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

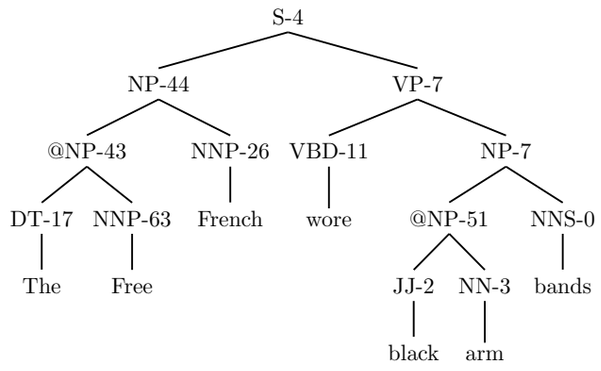


Category	Words
DT-17	The, A
NNP-26	Commission, Committee, Association, Department, Museum, Revolution, Office, ...
VBD-11	sent, exercised, received, retained, completed, followed, made, ...
JJ-2	ceramic, young, daily, imperial, full, ...

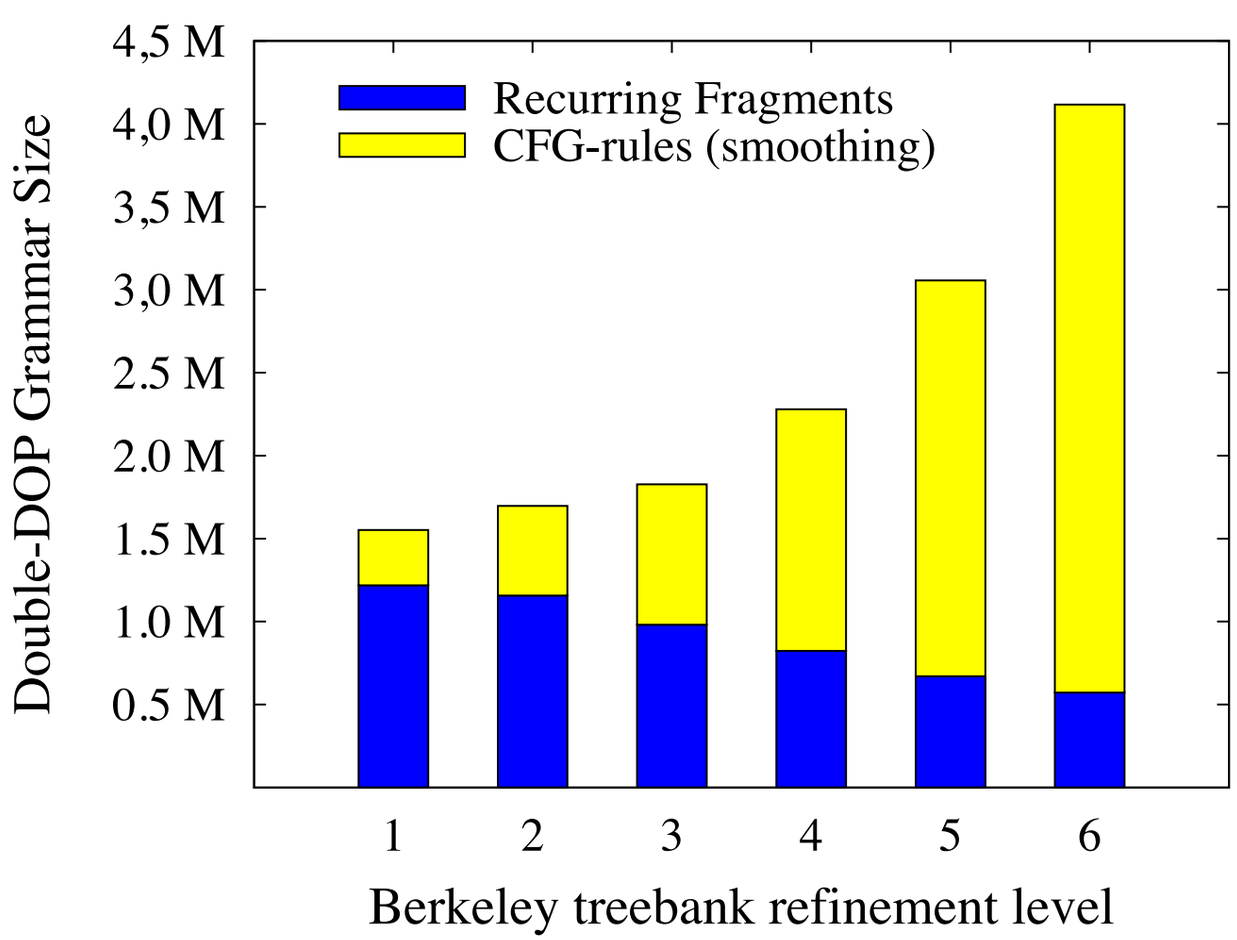
➡ S. Petrov. Coarse-to-Fine Natural Language Processing. PhD thesis, University of California at Bekeley, 2009.

Berkeley State Splitting (Sp)

Evaluating Double-DOP on the 6 levels

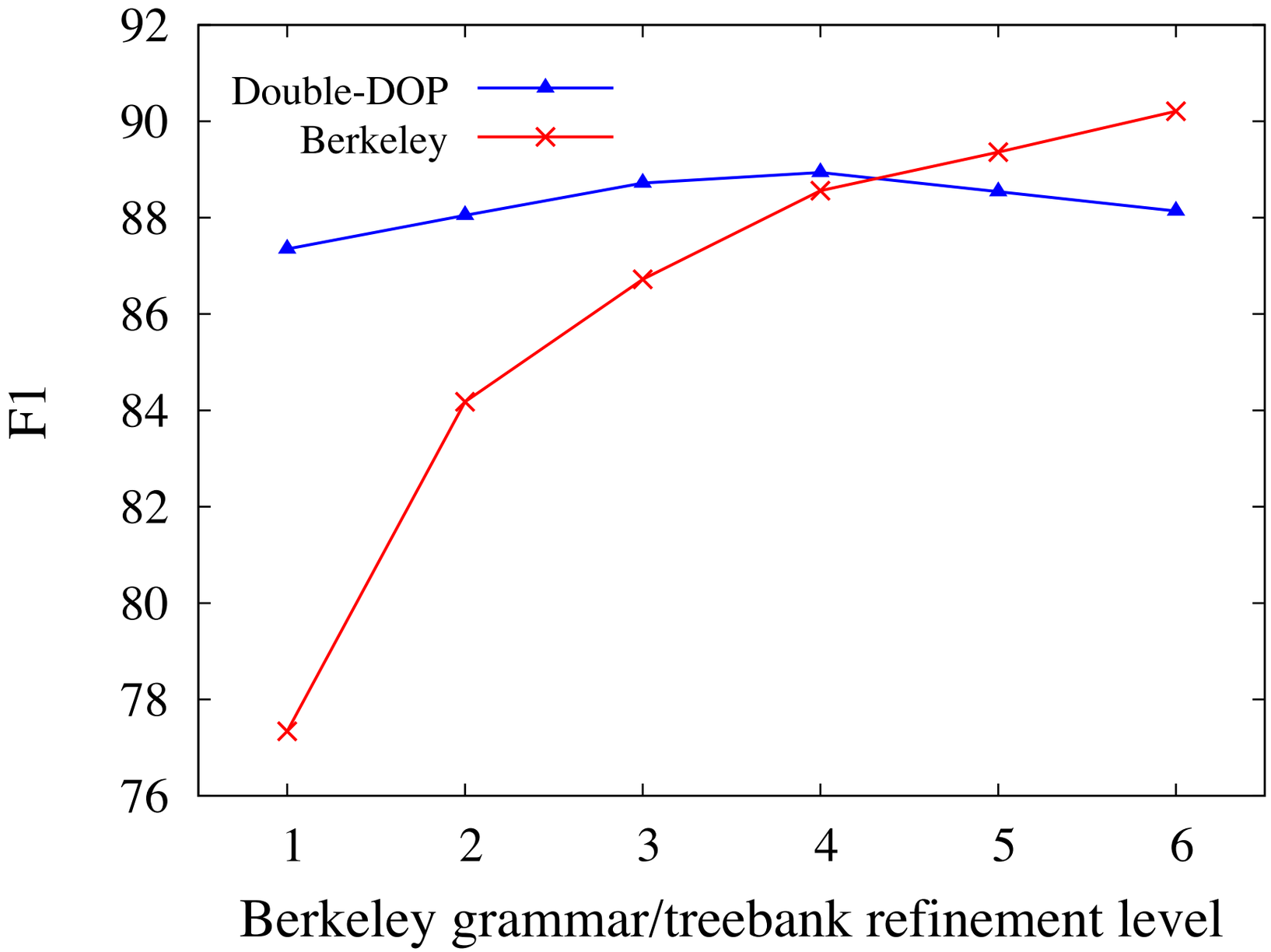


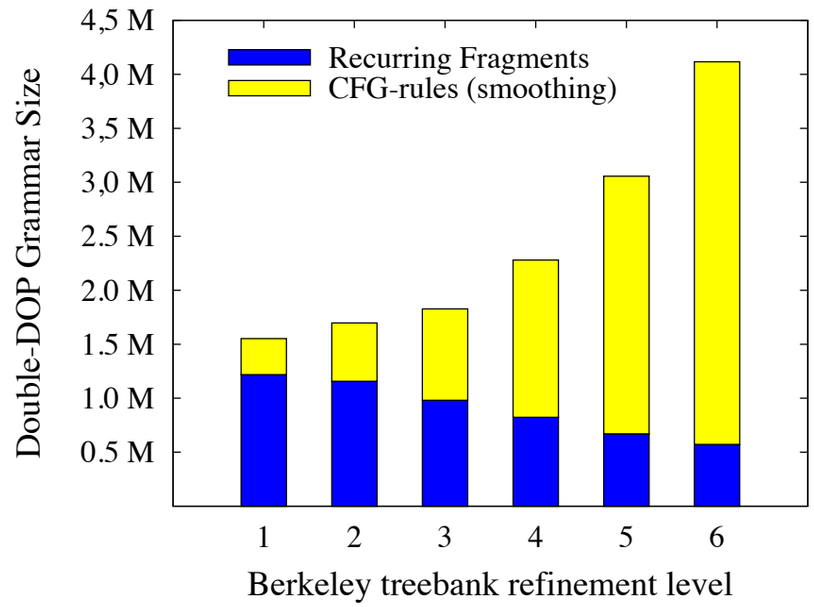
Category	Words
DT-17	The, A
NNP-26	Commission, Committee, Association, Department, Museum, Revolution, Office, ...
VBD-11	sent, exercised, received, retained, completed, followed, made, ...
JJ-2	ceramic, young, daily, imperial, full, ...



Berkeley State Splitting (Sp)

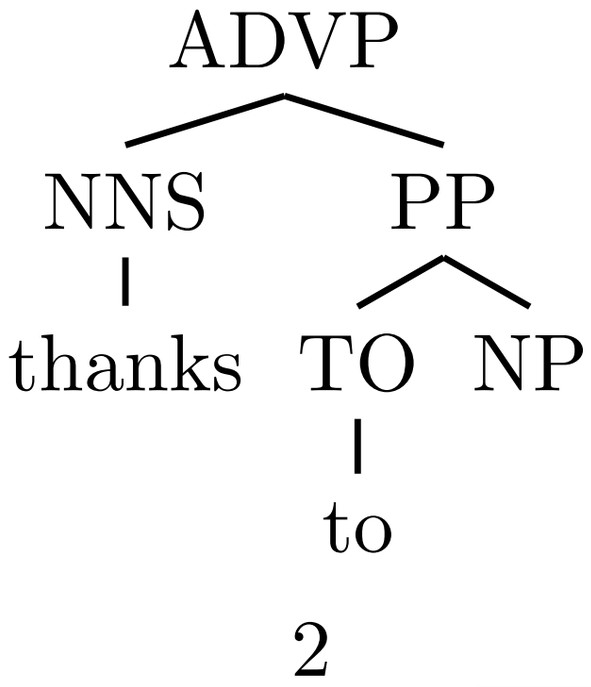
Evaluating Double-DOP on the 6 levels





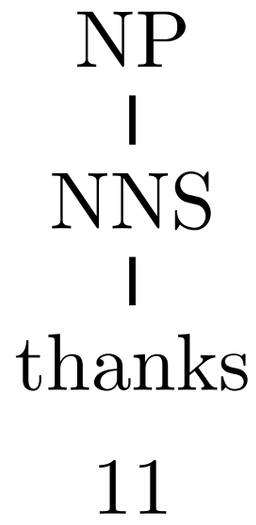
Conclusions

- 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



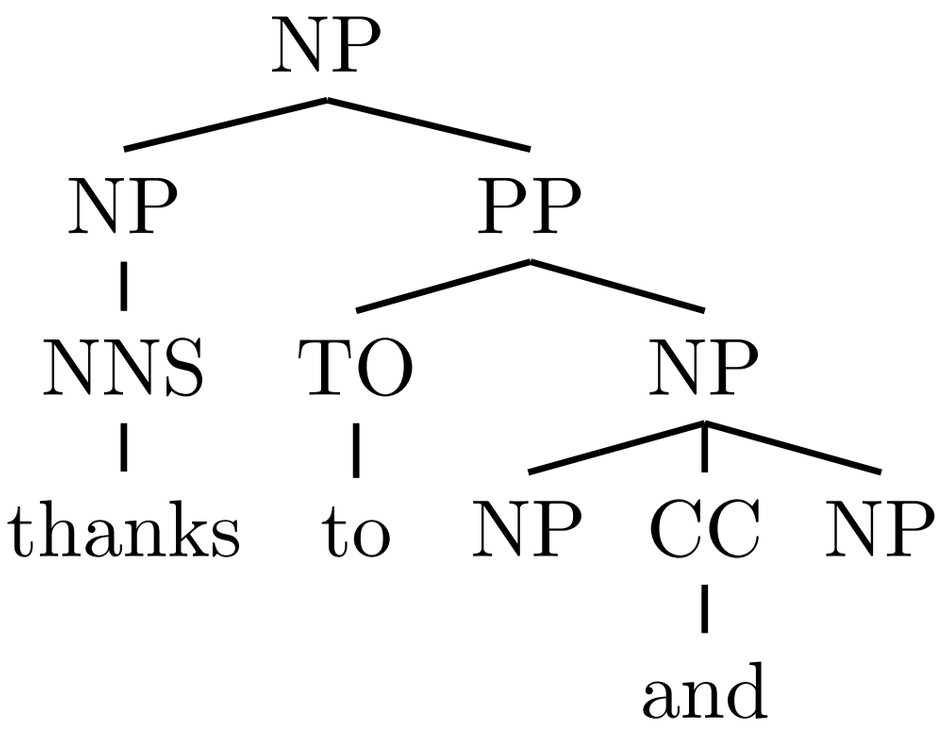
2

15

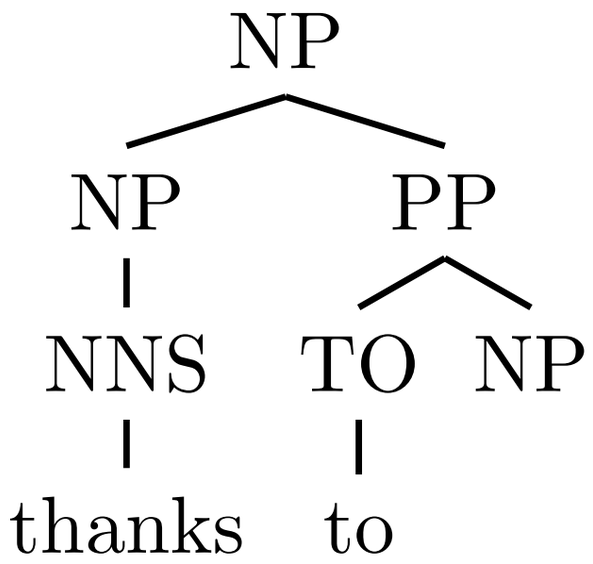


11

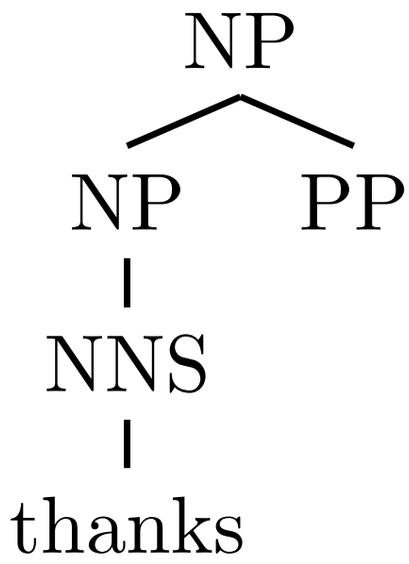
Thank you!



2

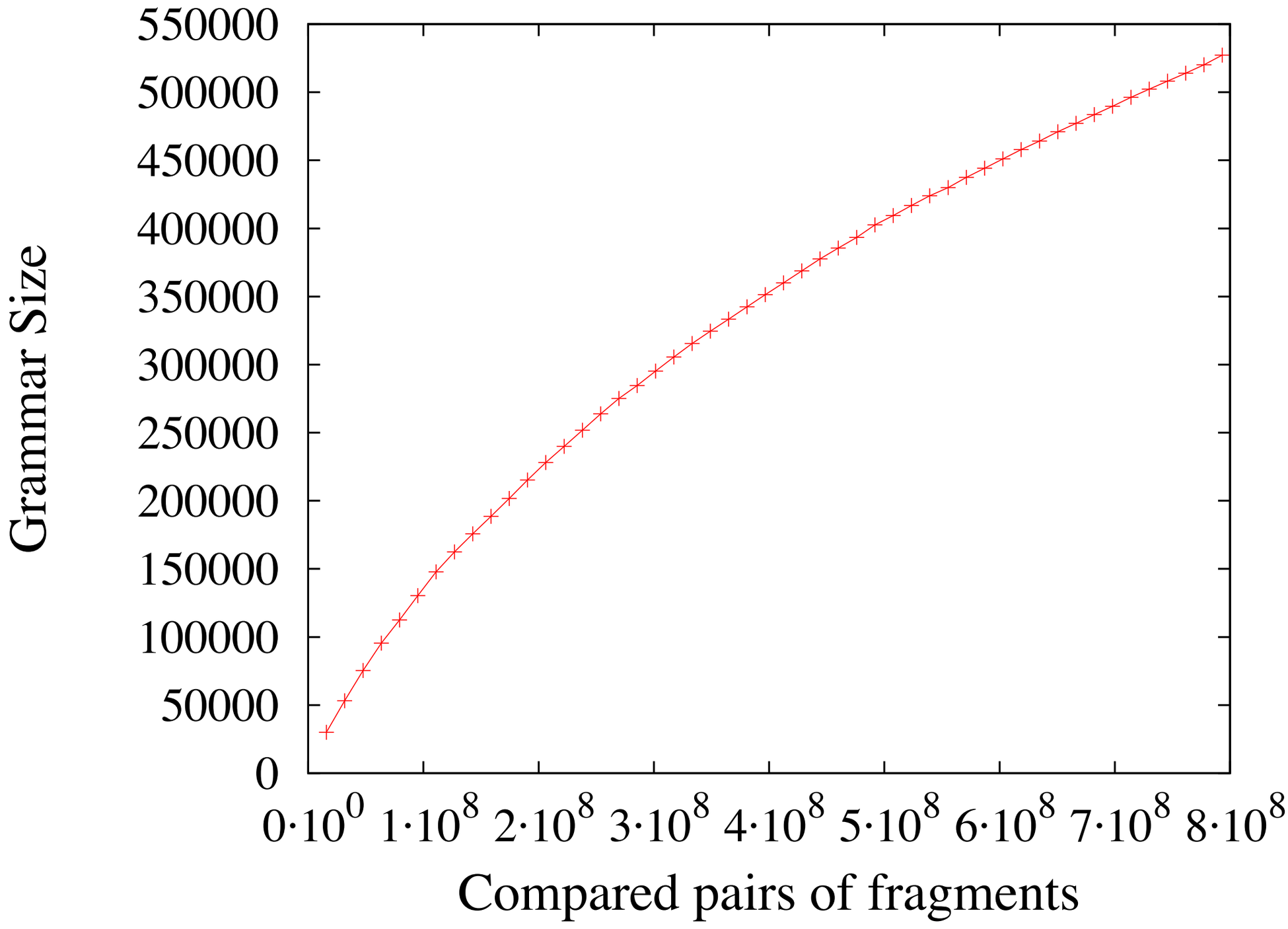


7

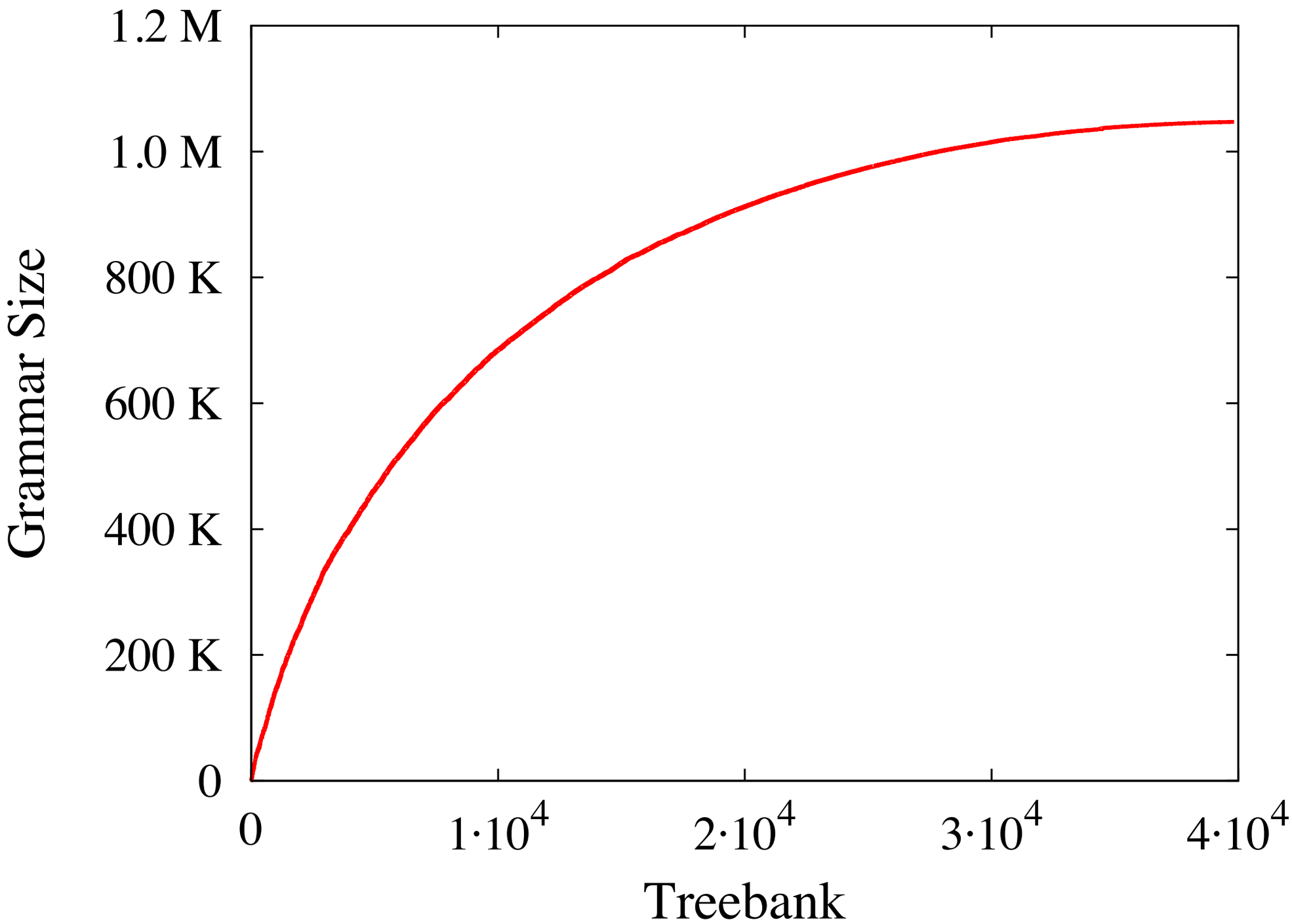


9

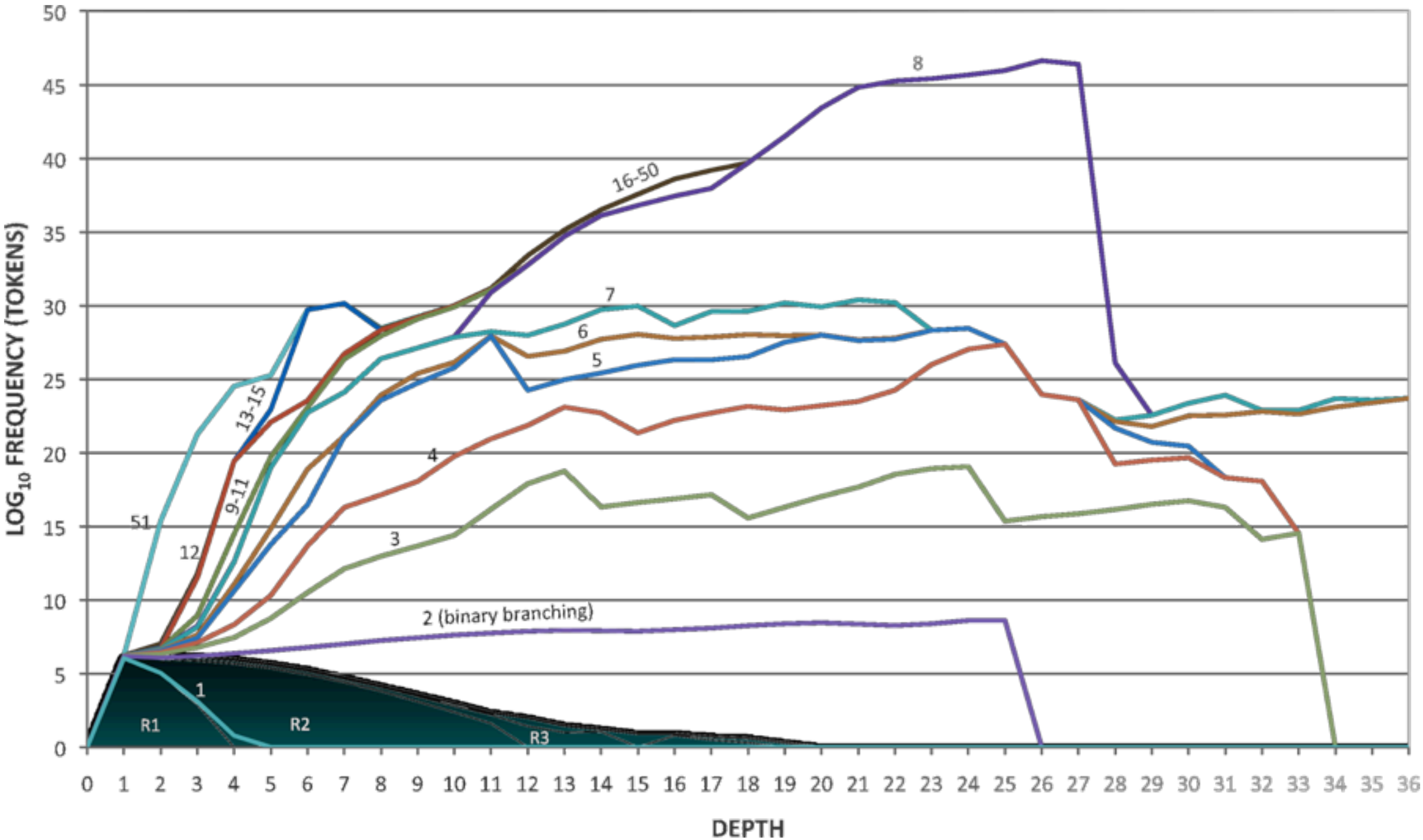
Grammar Size



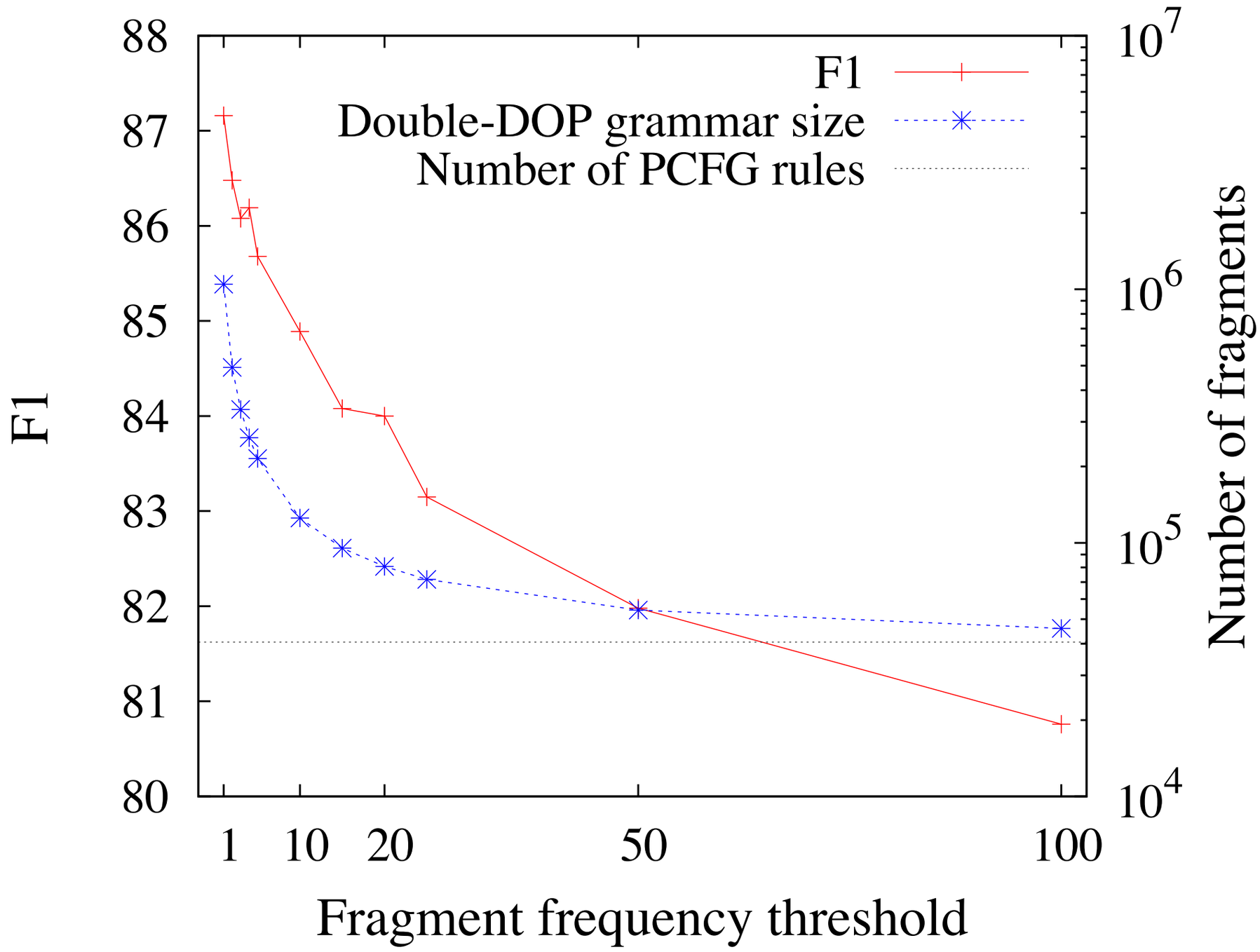
Grammar Size



Depth and Branching



Restricting Fragments
based on their frequencies



Results

Category label	% in gold	F1 Berkeley	F1 Double-DOP
NP	41.42	**91.4**	89.5
VP	20.46	**90.6**	88.6
S	13.38	**90.7**	87.6
PP	12.82	**85.5**	84.1
SBAR	3.47	**86.0**	82.1
ADVP	3.36	**82.4**	81.0
ADJP	2.32	**68.0**	67.3
QP	0.98	82.8	**84.6**
WHNP	0.88	**94.5**	92.0
WHADVP	0.33	**92.8**	91.9
PRN	0.32	**83.0**	77.9
NX	0.29	**9.50**	7.70
SINV	0.28	**90.3**	88.1
SQ	0.14	**82.1**	79.3
FRAG	0.10	26.4	**34.3**
SBARQ	0.09	84.2	**88.2**
X	0.06	72.0	**83.3**
NAC	0.06	54.6	**88.0**
WHPP	0.06	**91.7**	44.4
CONJP	0.04	55.6	**66.7**
LST	0.03	**61.5**	33.3
UCP	0.03	30.8	**50.0**
INTJ	0.02	44.4	**57.1**