Probabilistic Parsing: Evaluation & Improvement

Ling571
Deep Processing Techniques for NLP
January 25, 2016

Roadmap

- Probabilistic Parsing:
  - Parser training
  - Evaluation: Parseval
  - Limitations of PCFGs
  - Improvements on PCFGs
    - Parent annotation
    - Lexicalization

Probabilistic Parser Development Paradigm

- Training:
  - (Large) Set of sentences with associated parses (Treebank)
    - E.g., Wall Street Journal section of Penn Treebank, sec 2-21
      - 39,830 sentences
    - Used to estimate rule probabilities
- Development (dev):
  - (Small) Set of sentences with associated parses (WSJ, 22)
    - Used to tune/verify parser; check for overfitting, etc.
- Test:
  - (Small-med) Set of sentences w/parses (WSJ, 23)
    - 2416 sentences
  - Held out, used for final evaluation

Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'
  - Partial credit:
    - Constituents in output match those in reference
      - Same start point, end point, non-terminal symbol

Parseval

- How can we compute parse score from constituents?
- Multiple measures:
  - Labeled recall (LR):
    - # of correct constituents in hyp. parse
    - # of constituents in reference parse
  - Labeled precision (LP):
    - # of correct constituents in hyp. parse
    - # of total constituents in hyp. parse

Parseval (cont'd)

- F-measure:
  - Combines precision and recall

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2(P + R)}$$

  - F1-measure: $\beta = 1$ $$F_1 = \frac{2PR}{(P + R)}$$

- Crossing-brackets:
  - # of constituents where reference parse has bracketing ((A B) C) and hyp. has (A (B C))

Precision and Recall

- Gold standard
  - (S (NP (A a)) (VP (B b) (NP (C c)) (PP (D d))))
- Hypothesis
  - (S (NP (A a)) (VP (B b) (NP (C c) (PP (D d)))))
- G: S(0,4) NP(0,1) VP (1,4) NP (2,3) PP(3,4)
- H: S(0,4) NP(0,1) VP (1,4) NP (2,4) PP(3,4)
- LP: 4/5
- LR: 4/5
- F1: 4/5

State-of-the-Art Parsing

- Parsers trained/tested on *Wall Street Journal* PTB
  - LR: 90%+;
  - LP: 90%+;
  - Crossing brackets: 1%
- Standard implementation of Parseval: **evalb**

Evaluation Issues

- Constituents?
  - Other grammar formalisms
    - LFG, Dependency structure, ..
  - Require conversion to PTB format

  - Extrinsic evaluation
    - How well does this match semantics, etc?

Issues with PCFGs

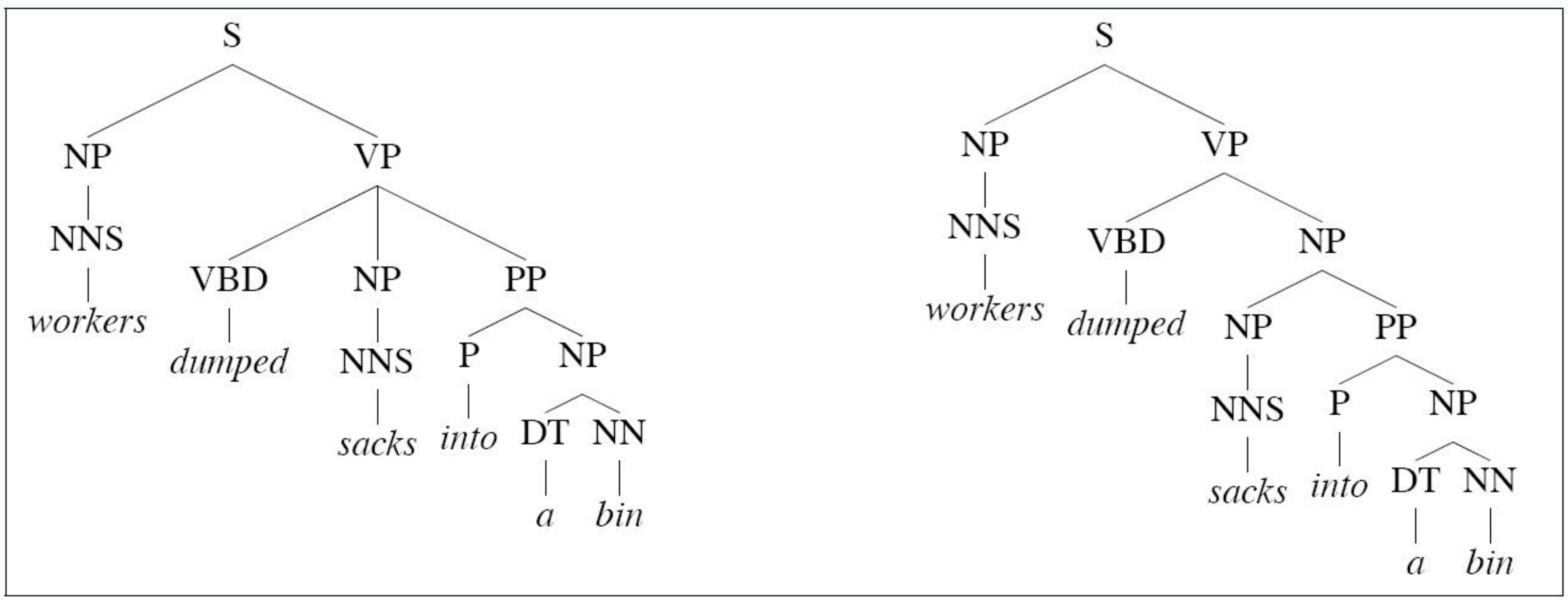
- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

	Pronoun	Non-pronoun
Subject	91%	9%
Object	34%	66%

    - In Treebank: roughly equi-probable
  - How can we handle this?
    - Condition on Subj/Obj with parent annotation

Issues with PCFGs

- Insufficient lexical conditioning
  - Present in pre-terminal rules
- Are there cases where other rules should be conditioned on words?



Different verbs & prepositions have different attachment preferences

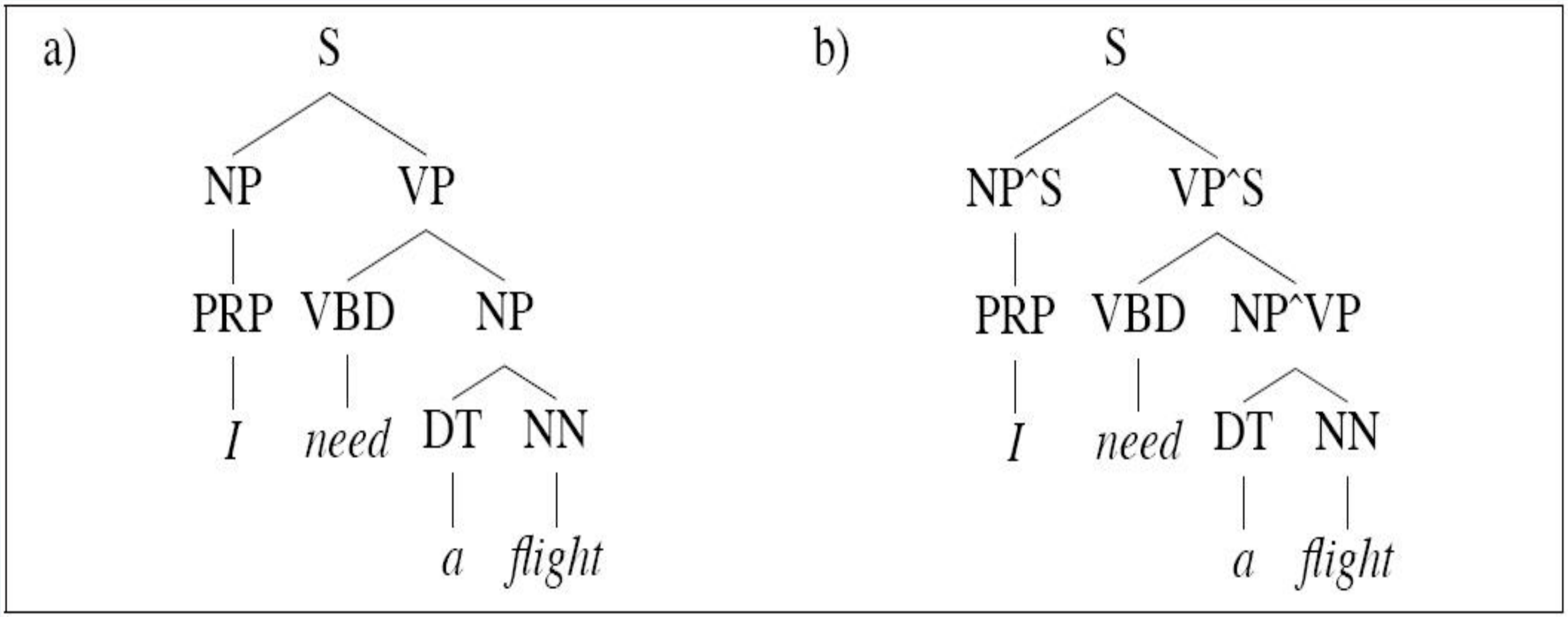
Parser Issues

- PCFGs make many (unwarranted) independence assumptions
  - Structural Dependency
    - NP → Pronoun: much more likely in subject position
  - Lexical Dependency
    - Verb subcategorization
    - Coordination ambiguity

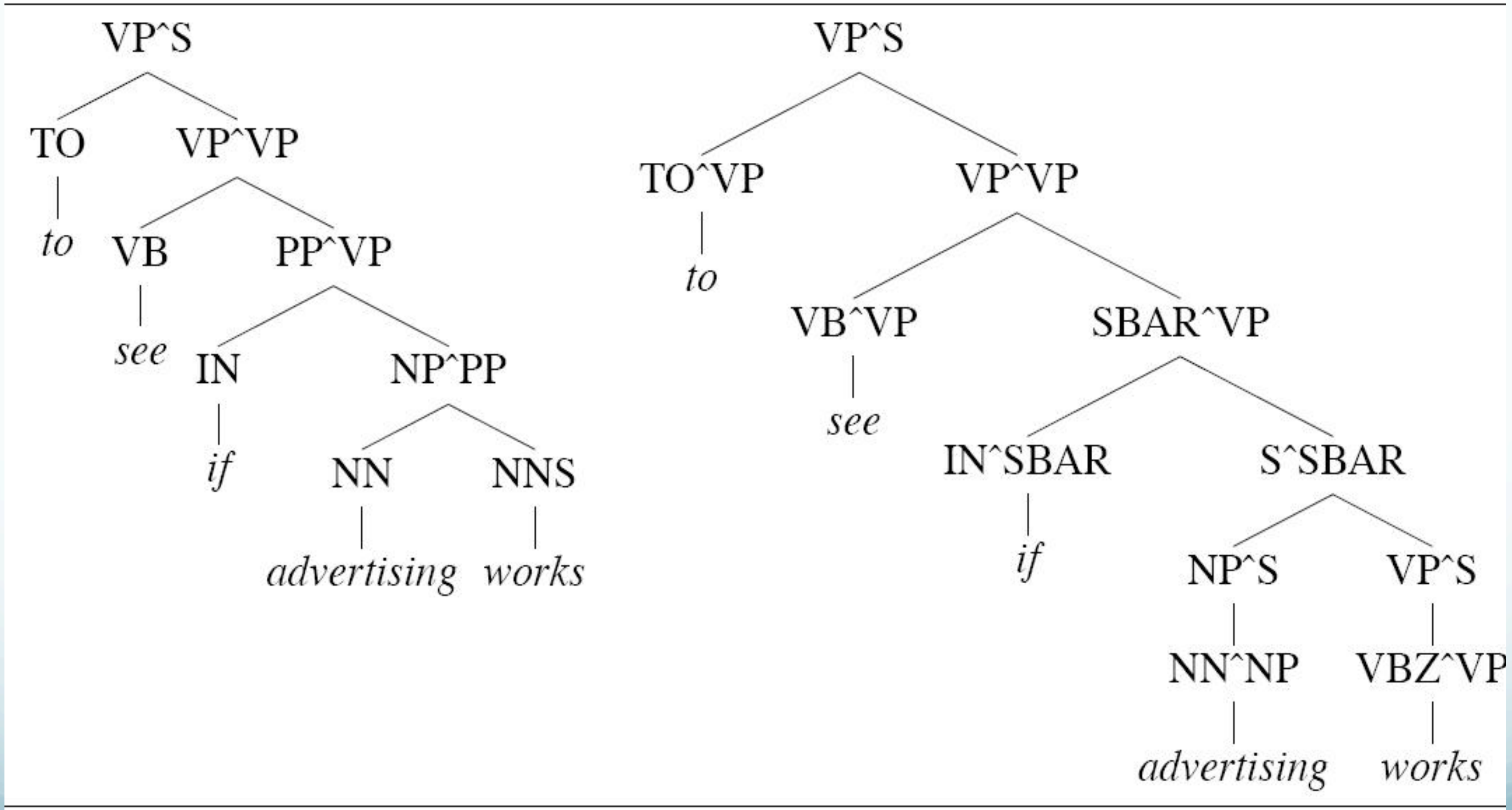
Improving PCFGs: Structural Dependencies

- How can we capture Subject/Object asymmetry?
  - E.g., $NP_{subj} \rightarrow$ Pron vs $NP_{Obj} \rightarrow$ Pron
- Parent annotation:
  - Annotate each node with parent in parse tree
    - E.g., NP^S vs NP^VP
    - Also annotate pre-terminals:
      - RB^ADVP vs RB^VP
      - IN^SBAR vs IN^PP
- Can also split rules on other conditions

Parent Annotation



Parent Annotation: Pre-terminals



Parent Annotation

- Advantages:
  - Captures structural dependency in grammars

- Disadvantages:
  - Increases number of rules in grammar
  - Decreases amount of training per rule

    - Strategies to search for optimal # of rules

Improving PCFGs: Lexical Dependencies

- Lexicalized rules:
  - Best known parsers: Collins, Charniak parsers
  - Each non-terminal annotated with its lexical head
    - E.g. verb with verb phrase, noun with noun phrase
  - Each rule must identify RHS element as head
    - Heads propagate up tree
  - Conceptually like adding 1 rule per head value

    - VP(dumped) → VBD(dumped)NP(sacks)PP(into)
    - VP(dumped) → VBD(dumped)NP(cats)PP(into)

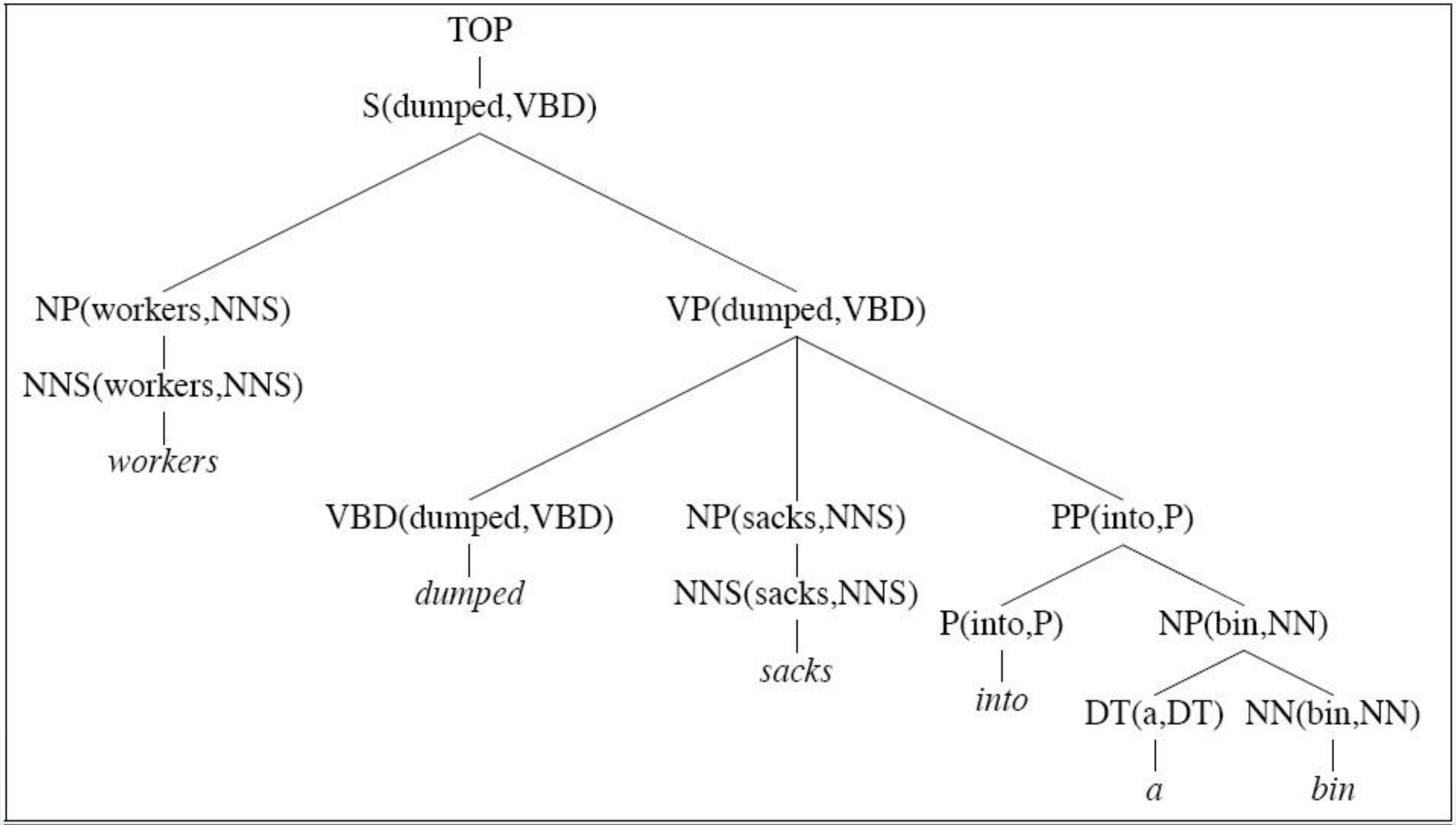
Lexicalized PCFGs

- Also, add head tag to non-terminals
  - Head tag: Part-of-speech tag of head word
    - VP(dumped) → VBD(dumped)NP(sacks)PP(into)
    - VP(dumped,VBD) → VBD(dumped,VBD)NP(sacks,NNS)PP(into,IN)
- Two types of rules:
  - Lexical rules: pre-terminal → word
    - Deterministic, probability 1
  - Internal rules: all other expansions
    - Must estimate probabilities

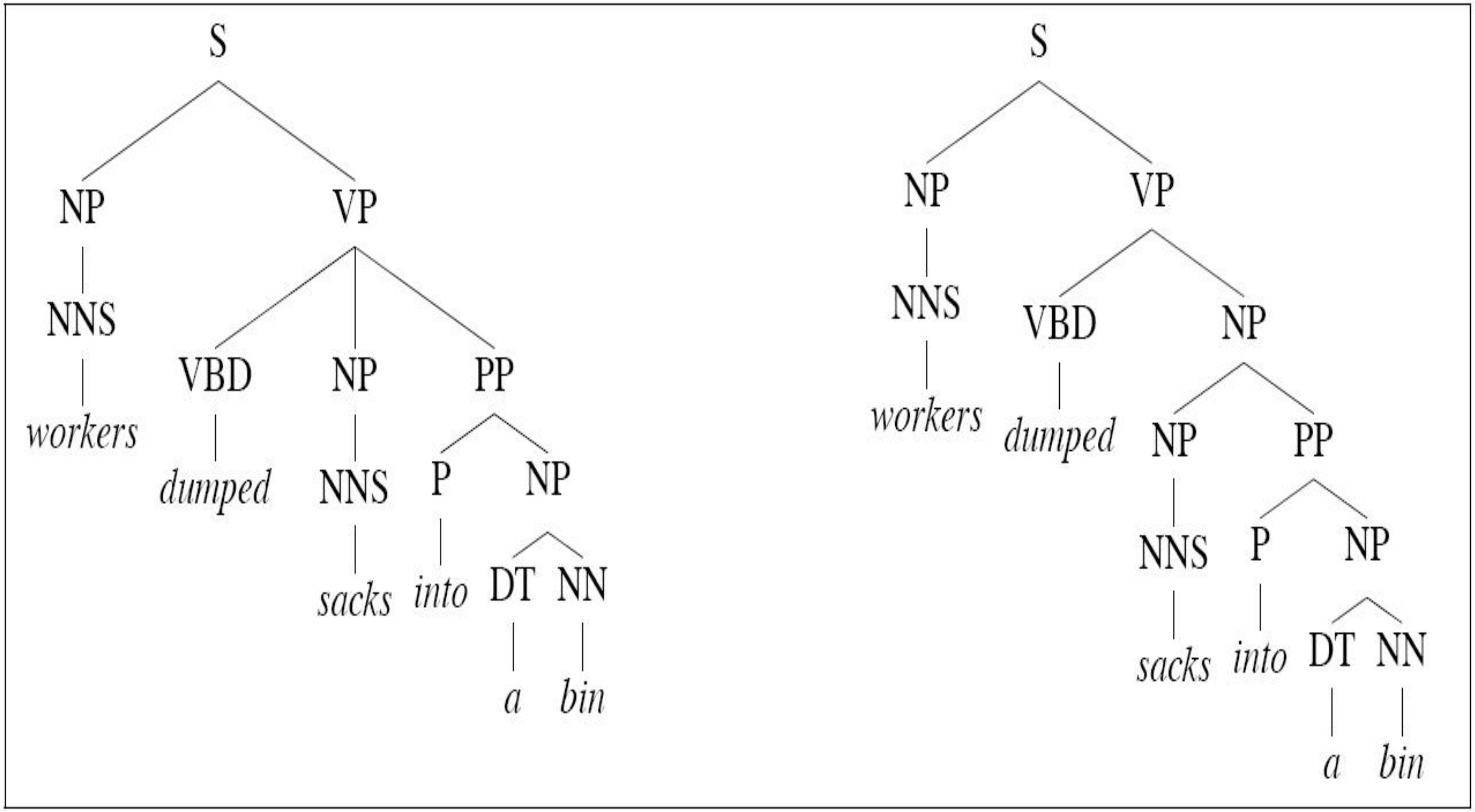
PLCFGs

- Issue: Too many rules
  - No way to find corpus with enough examples
- (Partial) Solution: Independence assumed
  - Condition rule on
    - Category of LHS, head
  - Condition head on
    - Category of LHS and parent's head

$$P(T,S) = \prod_{n \in T} p(r(n) \mid n, h(n)) * p(h(n) \mid n, h(m(n)))$$


TOP
S(dumped,VBD)
NP(workers,NNS)
NNS(workers,NNS)
workers
VP(dumped,VBD)
VBD(dumped,VBD)
dumped
NP(sacks,NNS)
NNS(sacks,NNS)
sacks
PP(into,P)
P(into,P)
into
NP(bin,NN)
DT(a,DT)
a
NN(bin,NN)
bin

Disambiguation Example



Disambiguation Example

$$P(VP \rightarrow VBD\,NP\,PP \mid VP, dumped) = \frac{C(VP(dumped) \rightarrow VBD\,NP\,PP)}{\sum_{\beta} C(VP(dumped) \rightarrow \beta)} = 6/9 = 0.67$$

$$p(VP \rightarrow VBD\,NP \mid VP, dumped) = \frac{C(VP(dumped) \rightarrow VBD\,NP)}{\sum_{\beta} C(VP(dumped) \rightarrow \beta)} = 0/9 = 0$$

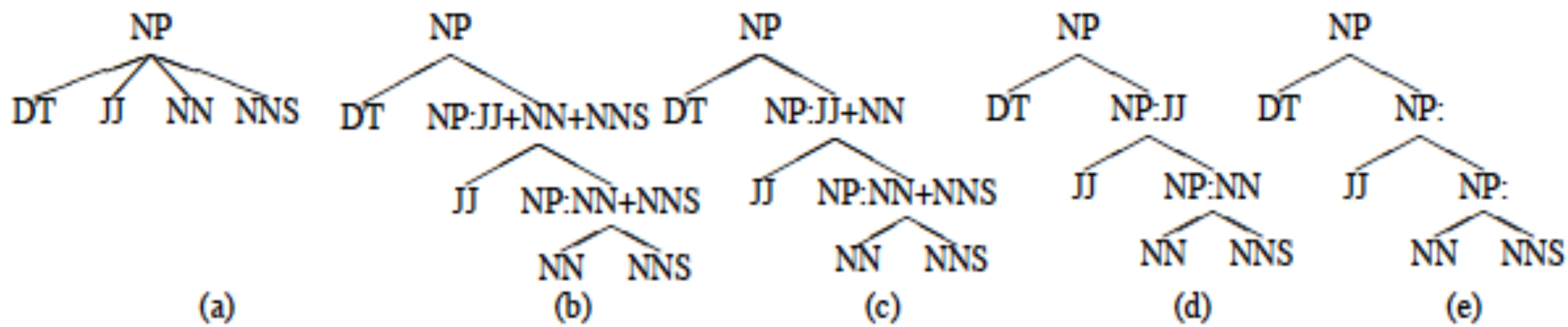
$$p(\mathrm{into} \mid PP, dumped) = \frac{C(X(dumped) \rightarrow \ldots PP(\mathrm{into})..)}{\sum_{\beta} C(X(dumped) \rightarrow \ldots PP\ldots)} = 2/9 = 0.22$$

$$p(\mathrm{into} \mid PP, sacks) = \frac{C(X(sacks) \rightarrow \ldots PP(\mathrm{into})\ldots)}{\sum_{\beta} C(X(sacks) \rightarrow \ldots PP\ldots)} = 0/0$$

CNF Factorization & Markovization

- CNF factorization:
  - Converts n-ary branching to binary branching
  - Can maintain information about original structure
    - Neighborhood history and parent
  - Issue:
    - Potentially explosive
      - If keep all context: 72 → 10K non-terminals!!!
  - How much context should we keep?
    - What Markov order?

Different Markov Orders



Markovization & Costs

(Mohri & Roark 2006)

PCFG	Time (s)	Words/s	$\lvert V \rvert$	$\lvert P \rvert$	LR	LP	F
Right-factored	4848	6.7	10105	23220	69.2	73.8	71.5
Right-factored, Markov order-2	1302	24.9	2492	11659	68.8	73.8	71.3
Right-factored, Markov order-1	445	72.7	564	6354	68.0	73.0	70.5
Right-factored, Markov order-0	206	157.1	99	3803	61.2	65.5	63.3
Parent-annotated, Right-factored, Markov order-2	7510	4.3	5876	22444	76.2	78.3	77.2

Improving PCFGs: Tradeoffs

- Tensions:
  - Increase accuracy:
    - Increase specificity
      - E.g. Lexicalizing, Parent annotation, Markovization, etc
  - Increases grammar
    - Increases processing times
    - Increases training data requirements
- How can we balance?

Efficiency

- PCKY is $|G|n^3$
  - Grammar can be huge
  - Grammar can be extremely ambiguous
    - 100s of analyses not unusual, esp. for long sentences
- However, only care about best parses
  - Others can be pretty bad
- Can we use this to improve efficiency?

Beam Thresholding

- Inspired by beam search algorithm
- Assume low probability partial parses unlikely to yield high probability overall
  - Keep only top k most probably partial parses
    - Retain only k choices per cell
      - For large grammars, could be 50 or 100
      - For small grammars, 5 or 10

Heuristic Filtering

- Intuition: Some rules/partial parses are unlikely to end up in best parse. Don't store those in table.

- Exclusions:
  - Low frequency: exclude singleton productions
  - Low probability: exclude constituents x s.t. $p(x) < 10^{-200}$
  - Low relative probability:
    - Exclude x if there exists y s.t. $p(y) > 100 * p(x)$