

Statistical Parsing

Dr. Demetrios Glinos
University of Central Florida

CAP6640 – Computer Understanding of Natural Language

Today

- Probabilistic Parsing
 - Probabilistic Context-Free Grammars
 - Probabilistic CKY Parsing of PCFGs
 - Ways to Learn PCFG Rule Probabilities
 - Problems with PCFGs
 - Improving PCFGs by Splitting Non-Terminals
 - Probabilistic Lexicalized CFGs
 - Probabilistic CCG Parsing

Probabilistic Parsing

- On average, sentences tend to be syntactically ambiguous
 - attachment ambiguity
 - coordination ambiguity
- CKY parsing can represent such phenomena, but not resolve them
 - can recognize multiple parses for same sentence
- NLP tasks require us to choose the "best" parse
 - semantic analysis
 - summarization
 - question answering
 - machine translation
- As a result, most modern parsers are necessarily probabilistic
 - the "best" parse is the most probable

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

PCFGs

- Probabilistic Context-Free Grammar (PCFG)
 - most commonly used
 - augments basic CFG using probabilities
- Can be trained on treebank grammars
- Can be parsed using a probabilistic version of CKY algorithm
- Can be further extended to handle additional linguistic phenomena
 - subcategorization
 - lexical dependencies

PCFG Formalism

- A PCFG is defined by

N - a set of non-terminal symbols

Σ - a set of terminal symbols (disjoint from N)

R - a set of rules or productions of the form $A \rightarrow \beta [p]$, where

A is a non-terminal

β is a string of symbols from the infinite set $(\Sigma \cup N)^*$

p is a number between 0 and 1 expressing $P(\beta | A)$

and where $\sum_{\beta} P(A \rightarrow \beta) = 1$ for each non-terminal A

S - a designated start symbol

- i.e., a PCFG augments a CFG grammar by adding conditional probabilities for each production

Example: PCFG for L_1

Grammar		Lexicon
$S \rightarrow NP VP$	[.80]	$Det \rightarrow that [.10] \mid a [.30] \mid the [.60]$
$S \rightarrow Aux NP VP$	[.15]	$Noun \rightarrow book [.10] \mid flight [.30]$
$S \rightarrow VP$	[.05]	$\mid meal [.015] \mid money [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flight [.40] \mid dinner [.10]$
$NP \rightarrow Proper-Noun$	[.30]	$Verb \rightarrow book [.30] \mid include [.30]$
$NP \rightarrow Det Nominal$	[.20]	$\mid prefer [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I [.40] \mid she [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me [.15] \mid you [.40]$
$Nominal \rightarrow Nominal Noun$	[.20]	$Proper-Noun \rightarrow Houston [.60]$
$Nominal \rightarrow Nominal PP$	[.05]	$\mid NWA [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does [.60] \mid can [.40]$
$VP \rightarrow Verb NP$	[.20]	$Preposition \rightarrow from [.30] \mid to [.30]$
$VP \rightarrow Verb NP PP$	[.10]	$\mid on [.20] \mid near [.15]$
$VP \rightarrow Verb PP$	[.15]	$\mid through [.05]$
$VP \rightarrow Verb NP NP$	[.05]	
$VP \rightarrow VP PP$	[.15]	
$PP \rightarrow Preposition NP$	[1.0]	

Figure 13.1 A PCFG that is a probabilistic augmentation of the \mathcal{L}_1 miniature English CFG

source: J&M (3d Ed. draft)

Using a PCFG for Disambiguation

- A PCFG assigns a probability to each parse T (i.e., to each derivation)
- We can disambiguate by choosing the highest probability parse
- Probability of particular parse T:

$$P(T, S) = \prod_{i=1}^n P(RHS_i | LHS_i) \text{ where } i \text{ ranges over all rules applied in the tree}$$

Now,

$P(T, S) = P(T) P(S | T)$ from the definition of joint probability

But $P(S | T) = 1$ since the tree includes all words in the sentence

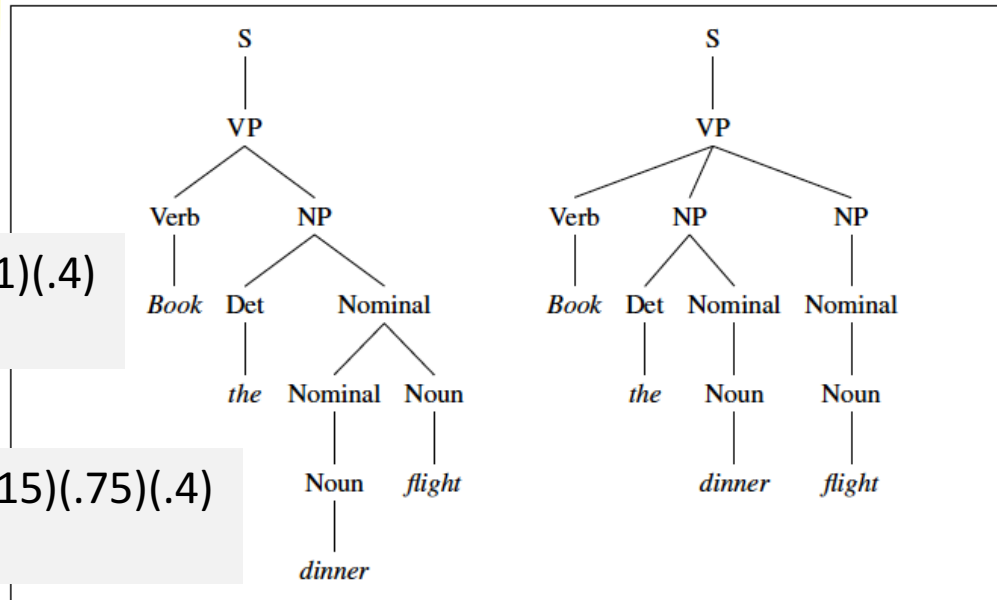
Thus, $P(T, S) = P(T)$

Therefore, the joint probability above is the probability of the parse

Example: Using PCFG to Disambiguate

$$P(T_{\text{left}}) = (.05)(.2)(.3)(.2)(.6)(.2)(.75)(.1)(.4) \\ = 2.2 \times 10^{-6}$$

$$P(T_{\text{right}}) = (.05)(.1)(.3)(.2)(.6)(.75)(.1)(.15)(.75)(.4) \\ = 6.1 \times 10^{-7}$$



Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
Verb	→	book	.30	Nominal	→	Noun	.75
Det	→	the	.60	Verb	→	book	.30
Noun	→	dinner	.10	Det	→	the	.60
Noun	→	flight	.40	Noun	→	dinner	.10
				Noun	→	flight	.40

Figure 13.2 Two parse trees for an ambiguous sentence. The transitive parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the ditransitive parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner'".

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Probabilistic CKY Parsing

- Requirement: PCFG must be in Chomsky Normal Form (CNF)
 - RHS of each rule must expand to either a single terminal or exactly 2 nonterminals, i.e., rules of form $A \rightarrow B C$ or $A \rightarrow w$
- As with CKY, use indices between the words, for example

[0] Book [1] the [2] flight [3] through [4] Houston [5]
- Grammar for example that follows:

$S \rightarrow NP VP$.80	$Det \rightarrow the$.40
$NP \rightarrow Det N$.30	$Det \rightarrow a$.40
$VP \rightarrow V NP$.20	$N \rightarrow meal$.01
$V \rightarrow includes$.05	$N \rightarrow flight$.02

source: J&M (3d Ed. draft), Chap 13

Example: Probabilistic CKY Parsing

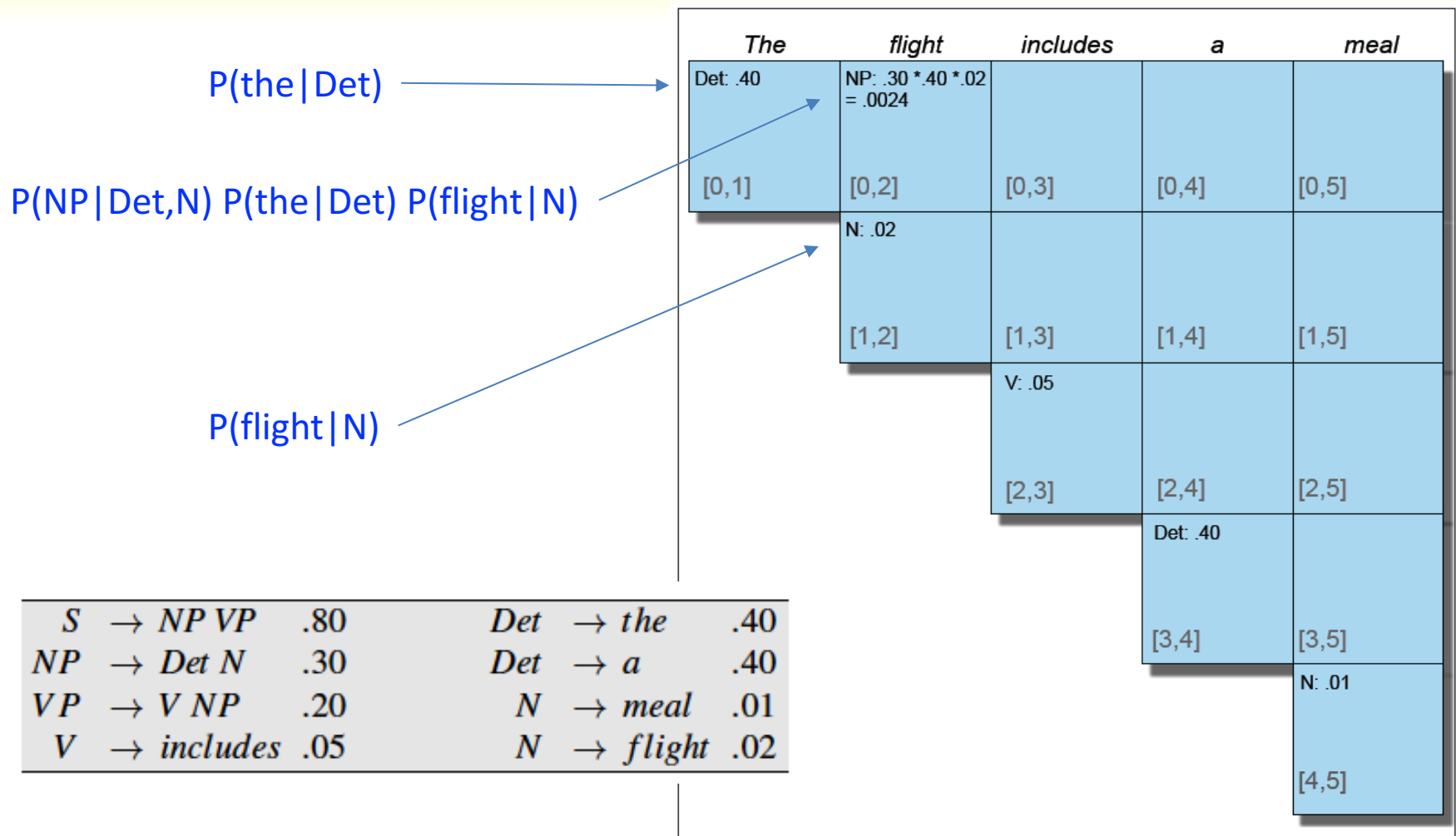


Figure 13.4 The beginning of the probabilistic CKY matrix. Filling out the rest of the chart is left as Exercise 13.4 for the reader.

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- **Ways to Learn PCFG Rule Probabilities**
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Learning PCFG Probabilities

- We can learn these from a treebank
 - count the number of times an expansion occurs, and then normalize

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} (\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- We can also use a non-probabilistic parser
 - parse a corpus of sentences with the parser, counting each rule expansion
 - keep track of partial counts separately for ambiguous sentences
 - assuming equal rule probabilities, compute probabilities for each such parse, and use them to weight the rule counts
 - re-estimate the rule probabilities, recompute parse probabilities, and iterate until convergence
 - this method is called the "inside-outside" algorithm

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- **Problems with PCFGs**
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Structural Dependencies Between Rules

- Context-free nature of PCFG misses structural dependencies

- Example:

In English, NPs that are syntactic subjects are more likely to be pronouns than are syntactic objects

NP statistics from the
Switchboard corpus

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

Basic PCFG sets probabilities to overall probability in the corpus

$$NP \rightarrow DT \ NN \quad [.28]$$
$$NP \rightarrow PRP \quad [.25]$$

- Parse tree context can be added by using "parent annotation" condition to rule

Lexical Dependencies

- Basic PCFG is not sensitive to the words in a parse tree
 - Words are used in the lexicon, but not in the rules
- But related words can be useful in the rules as well

Example: **Workers dumped sacks into a bin**

Q: Where to attach the PP “into a bin”?

modify “dumped”: $VP \rightarrow VBD \ NP \ PP$

modify “sacks”:
 $VP \rightarrow VBD \ NP$
 $NP \rightarrow NP \ PP$

Parses are compared on next slide

Example: Lexical Dependencies

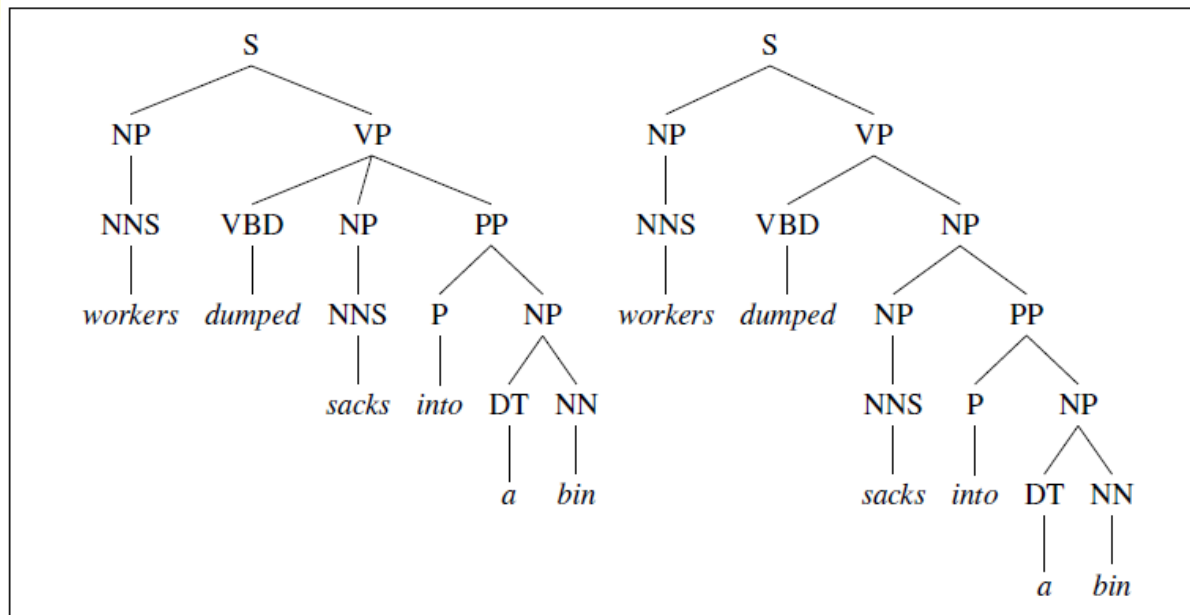


Figure 13.5 Two possible parse trees for a **prepositional phrase attachment ambiguity**. The left parse is the sensible one, in which "into a bin" describes the resulting location of the sacks. In the right incorrect parse, the sacks to be dumped are the ones which are already "into a bin", whatever that might mean.

source: J&M (3d Ed. draft)

- NP attachment is more common in English, which would give incorrect result here
- But if we adjusted probabilities to correct, would get incorrect results in other cases, e.g., "Fishermen caught tons of herring."
- Solution to use the identities of the verbs and prepositions involved
 - e.g., greater "affinity" of "dumped" to "into" than "sacks" to "into"

Example: Coordination Ambiguity

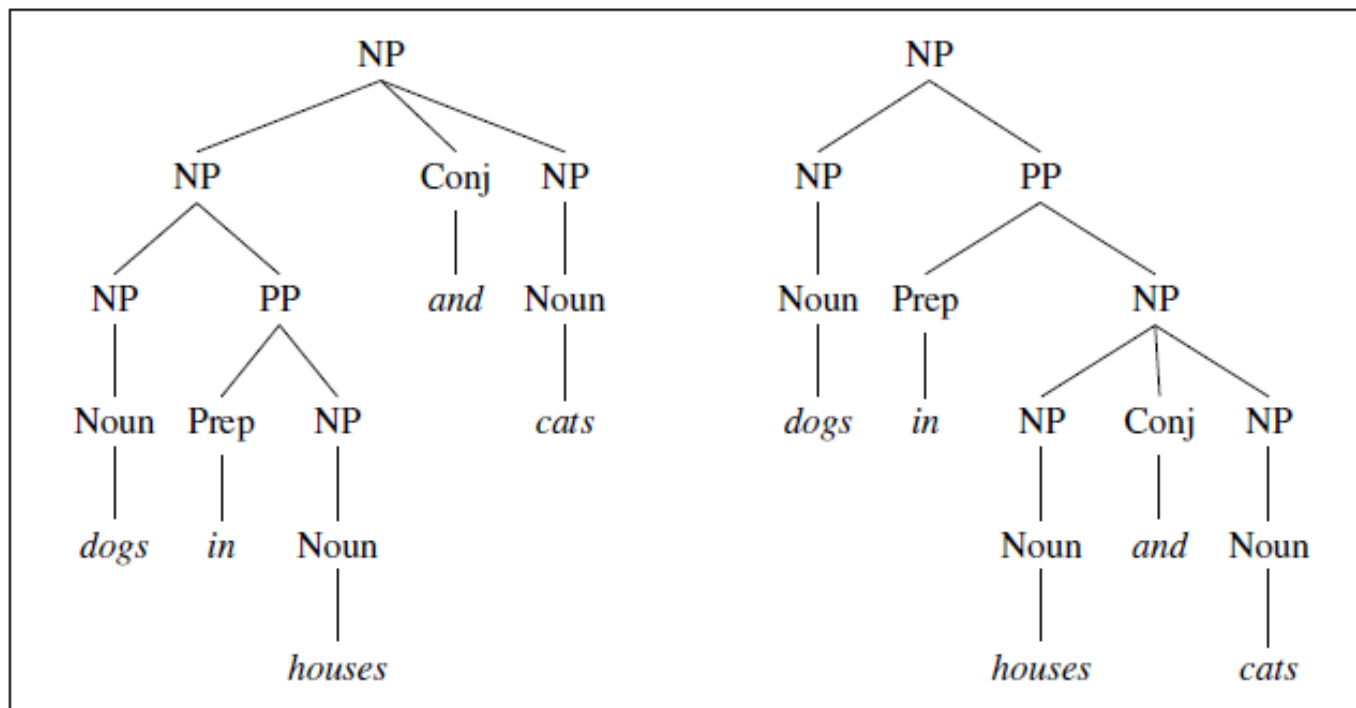


Figure 13.7 An instance of coordination ambiguity. Although the left structure is intuitively the correct one, a PCFG will assign them identical probabilities since both structures use exactly the same set of rules. After Collins (1999).

source: J&M (3d Ed. draft)

Note: a PCFG would assign same probability to both parses, since they contain exactly the same rule expansions

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Splitting Non-Terminals

- Basic idea: split non-terminals to have separate forms for different structural situations
 - e.g., two forms for NP: NP_{subject} and NP_{object}
 - allows different distributions for their rule expansions
e.g., different probabilities for $NP_{\text{subject}} \rightarrow PRP$ and $NP_{\text{object}} \rightarrow PRP$

Q: But how do we know which form to apply?

A: We don't, but we can take advantage of the fact that an NP subject is usually the child of an S node, while an NP object is usually the child of a VP node

Thus, we can use *parent annotations* to distinguish the cases:

NP^S for subject NPs

NP^VP for object VPs

Example: Parent Annotations

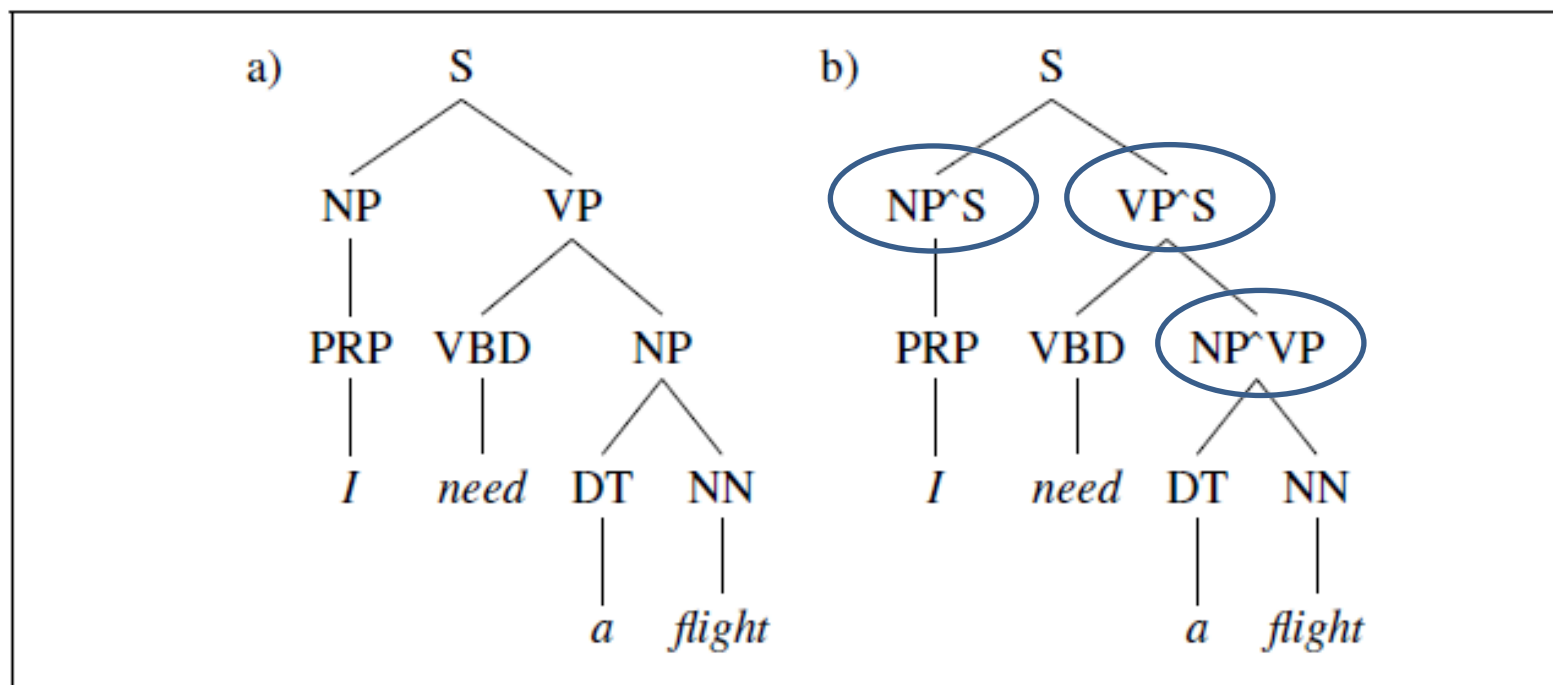


Figure 13.8 A standard PCFG parse tree (a) and one which has parent annotation on the nodes which aren't pre-terminal (b). All the non-terminal nodes (except the pre-terminal part-of-speech nodes) in parse (b) have been annotated with the identity of their parent.

source: J&M (3d Ed. draft) [annotated]

Splitting Pre-Terminal POS Nodes

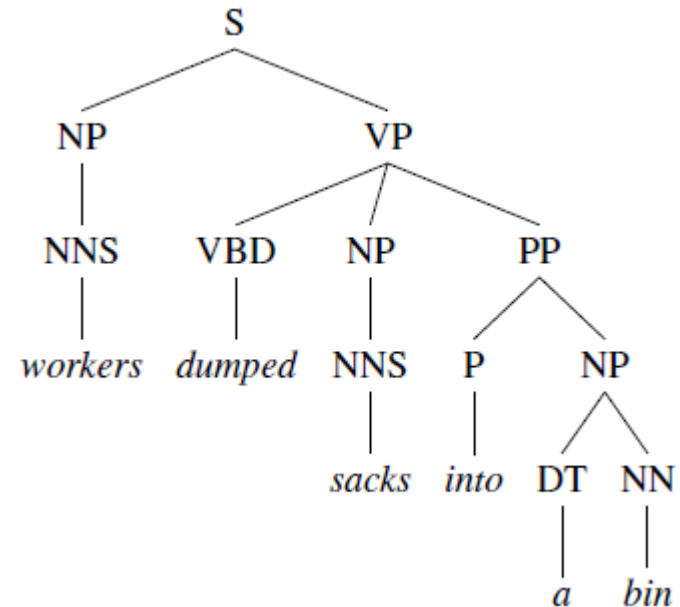
- Pre-terminal POS nodes also exhibit different distributional characteristics
 - e.g., standard POS tag for adverb is RB, but
 - Most common adverbs with ADVP parents are “also” and “now”
 - Most common adverbs with VP parents are “not” and “n’t”
 - Most common adverbs with NP parents are “only” and “just”
 - Can expect to improve PCFG with: RB^{ADVP} , RB^{VP} , and RB^{NP}
- Similarly, standard Penn Treebank IN tag includes
 - Subordinating conjunctions (e.g., while, as, if)
 - Complementizers (e.g., that, for)
 - Prepositions (e.g., of, in, from)
- But parent annotation is not always sufficient
 - Can hand-write rules that take into account other features
 - e.g., that/IN more likely a complementizer; as/IN more likely a subordinating conjunction

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Lexicalized PCFGs

- Lexicalized rules
 - modifying the probabilistic model instead of modifying the grammar rules to account for specific words (lexicals)
 - Charniak parser (1997)
 - Collins parser (1999)
- Background: lexicalized grammar
 - Rules augmented to identify head daughter non-terminal
 - Non-terminals are annotated with associated headwords
 - Also common to annotate with POS tag for headword



source: J&M (3d Ed. draft)

Fig. 13.5 [excerpt]

e.g., $VP(dumped, VBD) \rightarrow VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)$

The Problem with Lexicalized PCFGs

- To obtain the MLE for

$$VP(dumped, VBD) \rightarrow VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)$$

we must compute

$$\frac{Count(VP(dumped, VBD) \rightarrow VBD(dumped, VBD) NP(sacks, NNS) PP(into, P))}{Count(VP(dumped, VBD))}$$

- But this is so specific that we just cannot get enough (maybe not even any) data in a corpus
- We must make some additional statistical independence assumptions so we can estimate these values

The Collins Parser

- Intuition: consider RHS of each rule as consisting of a head non-terminal, plus non-terminals to the left and right of it, i.e.,

$$LHS \rightarrow L_n L_{n-1} \dots L_1 H R_1 R_2 \dots R_m$$

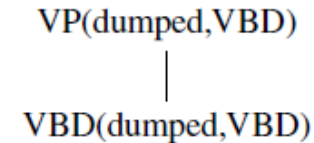
Where each symbol (L,R, or H) is a complex symbol representing the category, plus the head and the head tag, e.g. VP(dumped,VP) or NP(sacks,NNS)

- Collins Model 1 simplification:
 - Do not compute the MLE for the entire rule directly
 - Instead, first generate the head
 - Then generate the Ls and Rs around it, one at a time, from the inside out
 - Each generation step will have its own probability
 - Also add special STOP non-terminal so we know when to stop generating on a side, so we are generating a rule augmented, for example, as:

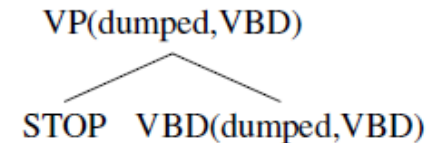
$$P(VP(dumped,VBD) \rightarrow STOP \ VBD(dumped,VBD) \ NP(sacks,NNS) \ PP(into,P) \ STOP)$$

Example: Generation of Augmented Rule

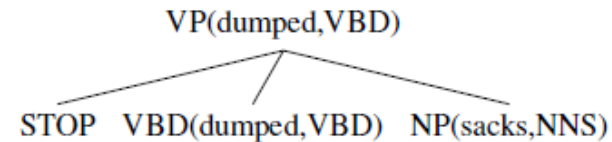
1. Generate the head VBD(dumped,VBD) with probability
 $P(H|LHS) = P(VBD(dumped,VBD) | VP(dumped,VBD))$



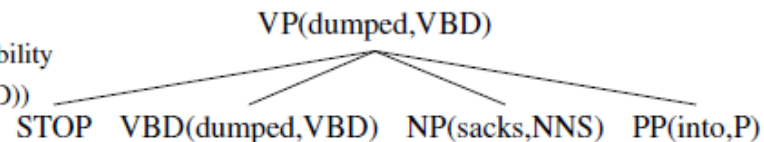
2. Generate the left dependent (which is STOP, since there isn't one) with probability
 $P(STOP | VP(dumped,VBD) VBD(dumped,VBD))$



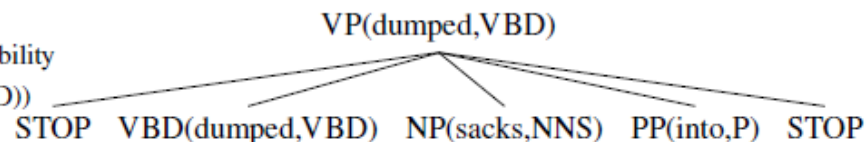
3. Generate right dependent NP(sacks,NNS) with probability
 $P_r(NP(sacks,NNS) | VP(dumped,VBD), VBD(dumped,VBD))$



4. Generate the right dependent PP(into,P) with probability
 $P_r(PP(into,P) | VP(dumped,VBD), VBD(dumped,VBD))$



- 5) Generate the right dependent STOP with probability
 $P_r(STOP | VP(dumped,VBD), VBD(dumped,VBD))$



Example: Generation of Augmented Rule

Using Collins's Model 1 simplification, we estimate the probability of

$$P(VP(dumped, VBD) \rightarrow STOP \ VBD(dumped, VBD) \ NP(sacks, NNS) \ PP(into, P) \ STOP)$$

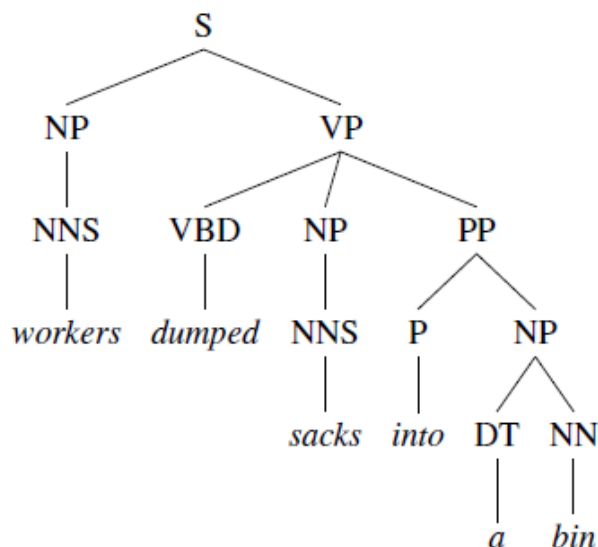
by computing

$$P_H(VBD|VP, dumped) \times P_L(STOP|VP, VBD, dumped)$$

$$\times P_R(NP(sacks, NNS)|VP, VBD, dumped)$$

$$\times P_R(PP(into, P)|VP, VBD, dumped)$$

$$\times P_R(STOP|VP, VBD, dumped)$$



Estimating Values

- Much less data is needed for estimating the values used in the Collins parser
- Example

$$P_R(NP(sacks, NNS)|VP, VBD, dumped)$$

$$= \frac{\text{Count}(VP(dumped, VBD) \text{ with } NNS(sacks) \text{ as a daughter somewhere on right})}{\text{Count}(VP(dumped, VBD))}$$

- Parsing algorithm: a probabilistic extension of CKY parsing

Additional Features in Collins Parser

- Collins Model 1
 - also incorporates a "distance" feature in the probability calculations
 - are there any words between the head and L/R word in the calculation?
 - is there a verb between the head and L/R word in the calculation?
- Collins Model 2
 - adds features for verb subcategorization frames
 - distinguishes arguments from adjuncts
- Smoothing
 - Collins models use interpolated smoothing of backoff probabilities that successively ignore the headword, its tag, and the parent node's tag
- Unknowns
 - Collins model uses UNKNOWN word token for unknowns and words occurring fewer than 6 times in the training set

Today

- Probabilistic Parsing
- Probabilistic Context-Free Grammars
- Probabilistic CKY Parsing of PCFGs
- Ways to Learn PCFG Rule Probabilities
- Problems with PCFGs
- Improving PCFGs by Splitting Non-Terminals
- Probabilistic Lexicalized CFGs
- Probabilistic CCG Parsing

Recall: Categorical Grammar Rules



- Categorical grammar rules specify how functions and their arguments combine
- Only 2 rule templates for all categorical grammars

$$\begin{array}{ll} X/Y \ Y \Rightarrow X & \leftarrow \text{forward function application} \\ Y \ X \backslash Y \Rightarrow X & \leftarrow \text{backward function application} \end{array}$$

- Example

Given: *United* and *Miami* are both simple NPs
serves is transitive with category $(S \backslash NP) / NP$

We analyze:

<i>United</i>	<i>serves</i>	<i>Miami</i>
NP	$(S \backslash NP) / NP$	NP
		
	$S \backslash NP$	
		
S		

Reprise: Combinatory Categorical Grammar

- Basic categorial grammar no more expressive than traditional CFG
 - grammatical facts just basically pushed down into the lexicon
- CCG is more expressive by including operations on functions

- Composition operations

$$X/Y \quad Y/Z \Rightarrow X/Z$$

$$Y \setminus Z \quad X \setminus Y \Rightarrow X \setminus Z$$

- Type raising operations

- elevates simple categories to the status of functions

$$X \Rightarrow T / (T \setminus X)$$

$$X \Rightarrow T \setminus (T / X)$$

- where T can be any atomic or functional category already in the grammar

Ambiguity in CCG

- Since CCG rules are all either binary or unary, a bottom-up tabular approach like CKY should work
- However, CCG rule structure typically results in an explosion of lexical categories for each word encountered in sequence
- This explosion of possibilities is managed by using **supertagging** to assess and exploit the most likely lexical categories that are possible for each word
- Supertagging approaches
 - maximum entropy
 - A* search

Supertags

- Supertags
 - tags highly lexicalized grammar frameworks
 - encode much of the derivation for a sentence
 - sometimes referred to as "almost parsing"
- Treebanks provide
 - overall set of lexical categories
 - typically use ~425 high-frequency categories out of >1,000 in CCGbank
 - compare to the 45 categories in Penn Treebank
 - allowable category assignments for each word in the lexicon

MaxEnt Approach

- Standard supervised machine learning approach
- Basic MEMM still has too many errors in most probable parse to be of practical use
- Solution is to return a probability distribution over the possible supertags for each word in the input

- Example:

United	serves	Denver
$N/N: 0.4$	$(S \backslash NP)/NP: 0.8$	$NP: 0.9$
$NP: 0.3$	$N: 0.1$	$N/N: 0.05$
$S/S: 0.1$
$S \backslash S: .05$		
...		

Note: ellipsis denotes all remaining possible supertags for each word

source: J&M (3d Ed. draft) Ch. 13

- We can't use Viterbi here, since it returns only the most probable partial sequence
- Use instead the related "forward-backward" algorithm (which we have not discussed)
- Deep learning using recurrent neural networks (RNNs) have also met with success

A* Search

- A* is a heuristic search algorithm
 - uses an agenda of partial solutions ordered by a cost function
 - lowest cost so far used to select a partial solution for further expansion
 - cost function $f(n)$ value is sum of 2 components
 - $g(n)$ = actual cost so far
 - $h(n)$ = heuristic estimate of cost from current state to solution
- If heuristic function does not overestimate the actual future cost (i.e., it is "admissible"), then A* finds an optimal solution
- For parsing, search states correspond to edges for completed constituents
- A* for parsing
 - phrase structure parsing (Klein and Manning, 2003)
 - CCG parsing (Lewis and Steedman, 2014)

A* for CCG Parsing

- Initialize agenda and parse table
 - for each word in input
 - along with f-costs
- main loop: remove lowest cost edge from agenda
 - if it is solution, done
 - else, generate new states based on CCG rules, assign costs, and add to agenda
- loop until solution found or agenda empty (= failed parse)

```

function CCG-ASTAR-PARSE(words) returns table or failure

    supertags ← SUPERTAGGER(words)
    for i ← from 1 to LENGTH(words) do
        for all {A | (words[i], A, score) ∈ supertags}
            edge ← MAKEEDGE(i − 1, i, A, score)
            table ← INSERTEDGE(table, edge)
            agenda ← INSERTEDGE(agenda, edge)

    loop do
        if EMPTY?(agenda) return failure
        current ← POP(agenda)
        if COMPLETEDPARSE?(current) return table
        table ← INSERTEDGE(chart, edge)
        for each rule in APPLICABLERULES(edge) do
            successor ← APPLY(rule, edge)
            if successor not ∈ agenda or chart
                agenda ← INSERTEDGE(agenda, successor)
            else if successor ∈ agenda with higher cost
                agenda ← REPLACEEDGE(agenda, successor)
    
```

Figure 13.11 A*-based CCG parsing.

source: J&M (3d Ed. draft)

Cost Values

- For generic PCFG
 - probability of a derivation (parse tree) is the product of the probabilities of the rules that make up the tree
- For a CCG derivation
 - probability of derivation is the probability of the supertags assigned to the words in the derivation
 - i.e., we ignore the rules (a simplifying assumption)

$$P(D, S) = P(T, S)$$

$$= \prod_{i=1}^n P(t_i | s_i)$$

where

D = derivation

S = input sentence

T = supertag sequence

Heuristic Functions

Now, for traditional A*, we wish to have a function that we **minimize**

So, we will use negative log probabilities

Thus

$$P(D, S) = P(T, S)$$

$$= \sum_{i=1}^n -\log P(t_i | s_i)$$

f -cost of edge n

$g(n)$ = actual cost of span so far (inside span), using equation above

$h(n)$ = each remaining word in the outside span will be assigned its most probable supertag (an admissible heuristic)

f -cost of edge

Consider: f -cost of "serves" in "United serves Denver"

$$f(w_{i,j}, t_{i,j}) = g(w_{i,j}) + h(w_{i,j})$$

$$= \sum_{k=i}^j -\log P(t_k | w_k) \quad \leftarrow \text{actual cost of this edge}$$

"serves"

$$+ \sum_{k=1}^{i-1} \max_{t \in \text{tags}} (-\log P(t | w_k)) \quad \leftarrow \text{heuristic for words before}$$

"United"

$$+ \sum_{k=j+1}^N \max_{t \in \text{tags}} (-\log P(t | w_k)) \quad \leftarrow \text{heuristic for words after}$$

"Denver"

Example: "United serves Denver"

- Best first edge is "United"
- goal state is recognized as solution when it is popped from agenda
- effectiveness of A* seen by observing that states in white were not examined before solution found
 - compare to PCKY approach, in which all possible states are examined before top right entry

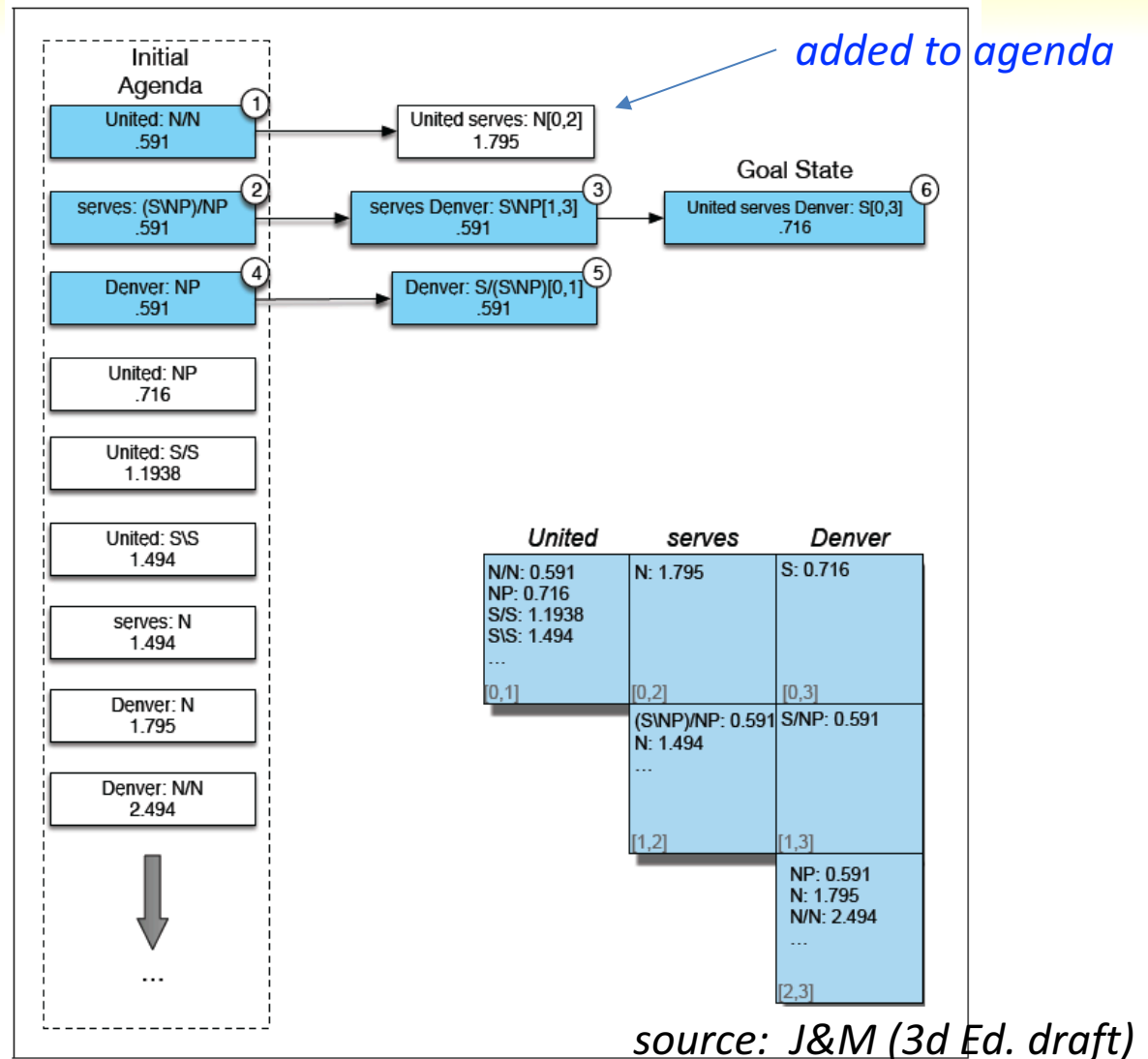


Figure 13.12 Example of an A* search for the example "United serves Denver". The circled numbers on the white boxes indicate the order in which the states are popped from the agenda. The costs in each state are based on f-costs using negative \log_{10} probabilities.