CS114 Lecture 13b

Probabilistic Parsing

March 12, 2013

Professor Meteer

Thanks for Jurafsky & Martin & Prof. Pustejovksy for slides

Why NLP is difficult: Newspaper headlines

- Ban on Nude Dancing on Governor's Desk
- Iraqi Head Seeks Arms
- Juvenile Court to Try Shooting Defendant
- Teacher Strikes Idle Kids
- Stolen Painting Found by Tree
- Local High School Dropouts Cut in Half
- Red Tape Holds Up New Bridges
- Clinton Wins on Budget, but More Lies Ahead
- Hospitals Are Sued by 7 Foot Doctors
- Kids Make Nutritious Snacks

Probabilistic CFGs

- The probabilistic model
  - Assigning probabilities to parse trees
- Getting the probabilities for the model
- Parsing with probabilities
  - Slight modification to dynamic programming approach
  - Task is to find the max probability tree for an input

Probability Model

- Attach probabilities to grammar rules
- The expansions for a given non-terminal sum to 1

  VP -> Verb .55

  VP -> Verb NP .40

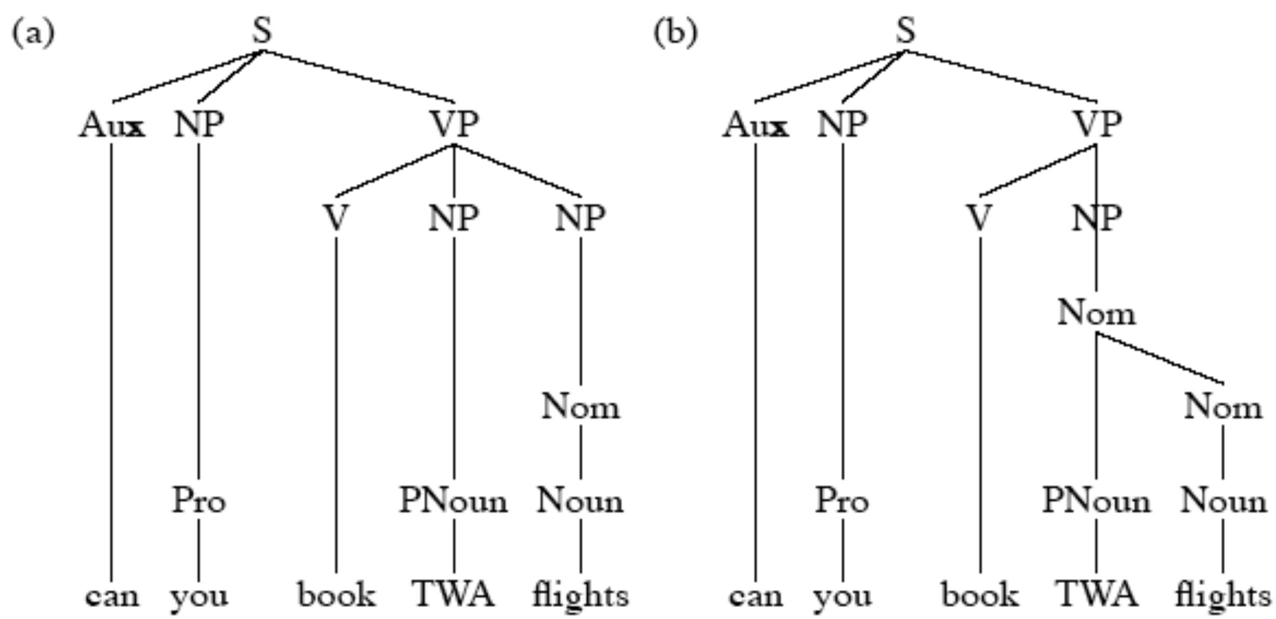
  VP -> Verb NP NP .05

  – Read this as P(Specific rule | LHS)

PCFG

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

PCFG



Rules			P	Rules			P
S	→	Aux NP VP	.15	S	→	Aux NP VP	.15
NP	→	Pro	.40	NP	→	Pro	.40
VP	→	V NP NP	.05	VP	→	V NP	.40
NP	→	Nom	.05	NP	→	Nom	.05
NP	→	PNoun	.35	Nom	→	PNoun Nom	.05
Nom	→	Noun	.75	Nom	→	Noun	.75
Aux	→	Can	.40	Aux	→	Can	.40
NP	→	Pro	.40	NP	→	Pro	.40
Pro	→	you	.40	Pro	→	you	.40
Verb	→	book	.30	Verb	→	book	.30
PNoun	→	TWA	.40	Pnoun	→	TWA	.40
Noun	→	flights	.50	Noun	→	flights	.50

Probability Model (1)

- A derivation (tree) consists of the set of grammar rules that are in the tree
- The probability of a tree is just the product of the probabilities of the rules in the derivation.

Probability model

$$P(T,S) = \prod_{n \in T} p(r_n)$$

- P(T,S) = P(T)P(S|T) = P(T); since P(S|T)=1

$$\begin{aligned} P(T_l) &= .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.5 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P(T_r) &= .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.7 \times 10^{-6} \end{aligned}$$

Probability Model (1.1)

- The probability of a word sequence P(S) is the probability of its tree in the unambiguous case.
- It’s the sum of the probabilities of the trees in the ambiguous case.

Getting the Probabilities

- From an annotated database (a treebank)
  - So for example, to get the probability for a particular VP rule just count all the times the rule is used and divide by the number of VPs overall.

TreeBanks

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Probabilistic Grammar Assumptions

- We're assuming that there is a grammar to be used to parse with.
- We're assuming the existence of a large robust dictionary with parts of speech
- We're assuming the ability to parse (i.e. a parser)
- Given all that… we can parse probabilistically

Typical Approach

- Bottom-up (CKY) dynamic programming approach
- Assign probabilities to constituents as they are completed and placed in the table
- Use the max probability for each constituent going up

What's that last bullet mean?

- Say we're talking about a final part of a parse
  - $S\text{->}_0NP_iVP_j$

  The probability of the S is…

  P(S->NP VP)*P(NP)*P(VP)

  The green stuff is already known. We're doing bottom-up parsing

Max

- I said the P(NP) is known.
- What if there are multiple NPs for the span of text in question (0 to i)?
- Take the max (where?)

Problems with PCFGs

- The probability model we're using is just based on the rules in the derivation…
  - Doesn't use the words in any real way
  - Doesn't take into account where in the derivation a rule is used

Solution

- Add lexical dependencies to the scheme…
  - Infiltrate the predilections of particular words into the probabilities in the derivation
  - I.e. Condition the rule probabilities on the actual words

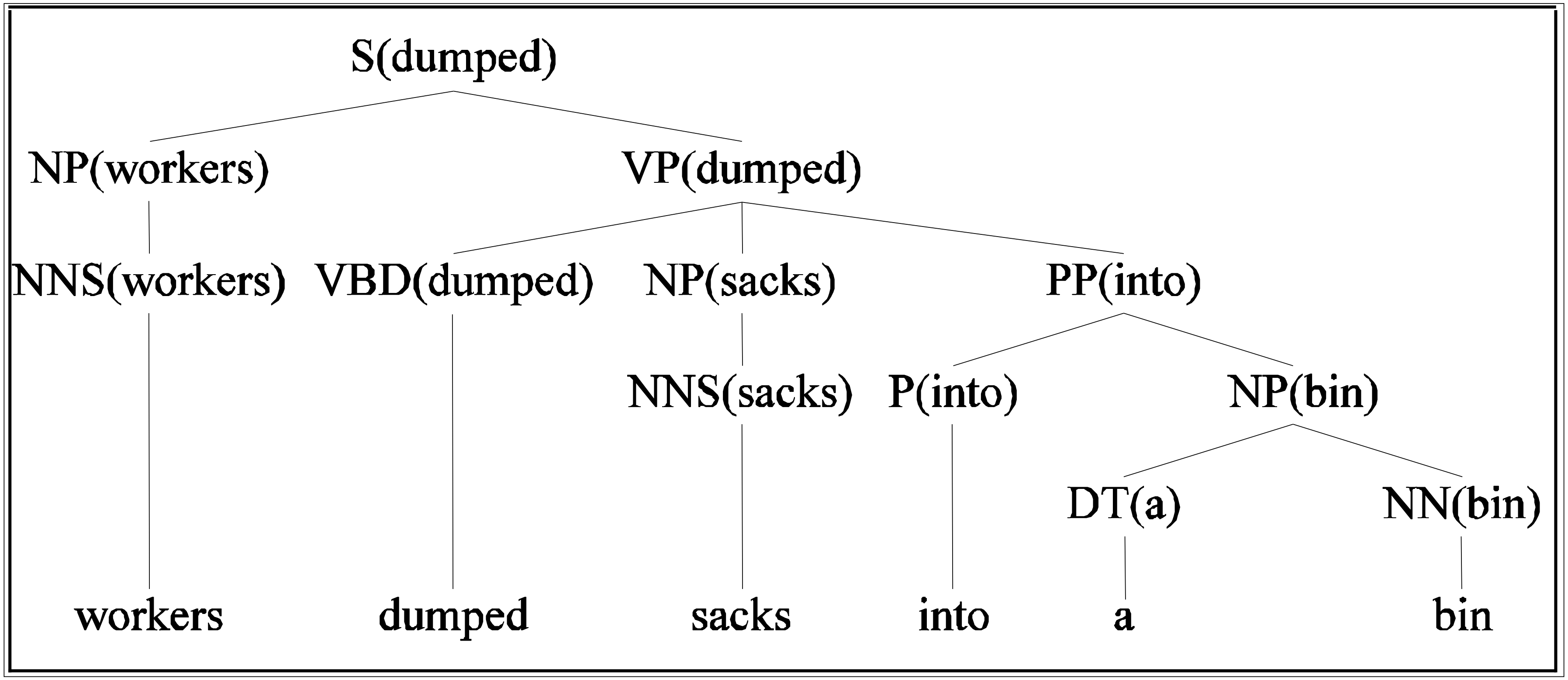
Heads

- To do that we're going to make use of the notion of the head of a phrase
  - The head of an NP is its noun
  - The head of a VP is its verb
  - The head of a PP is its preposition

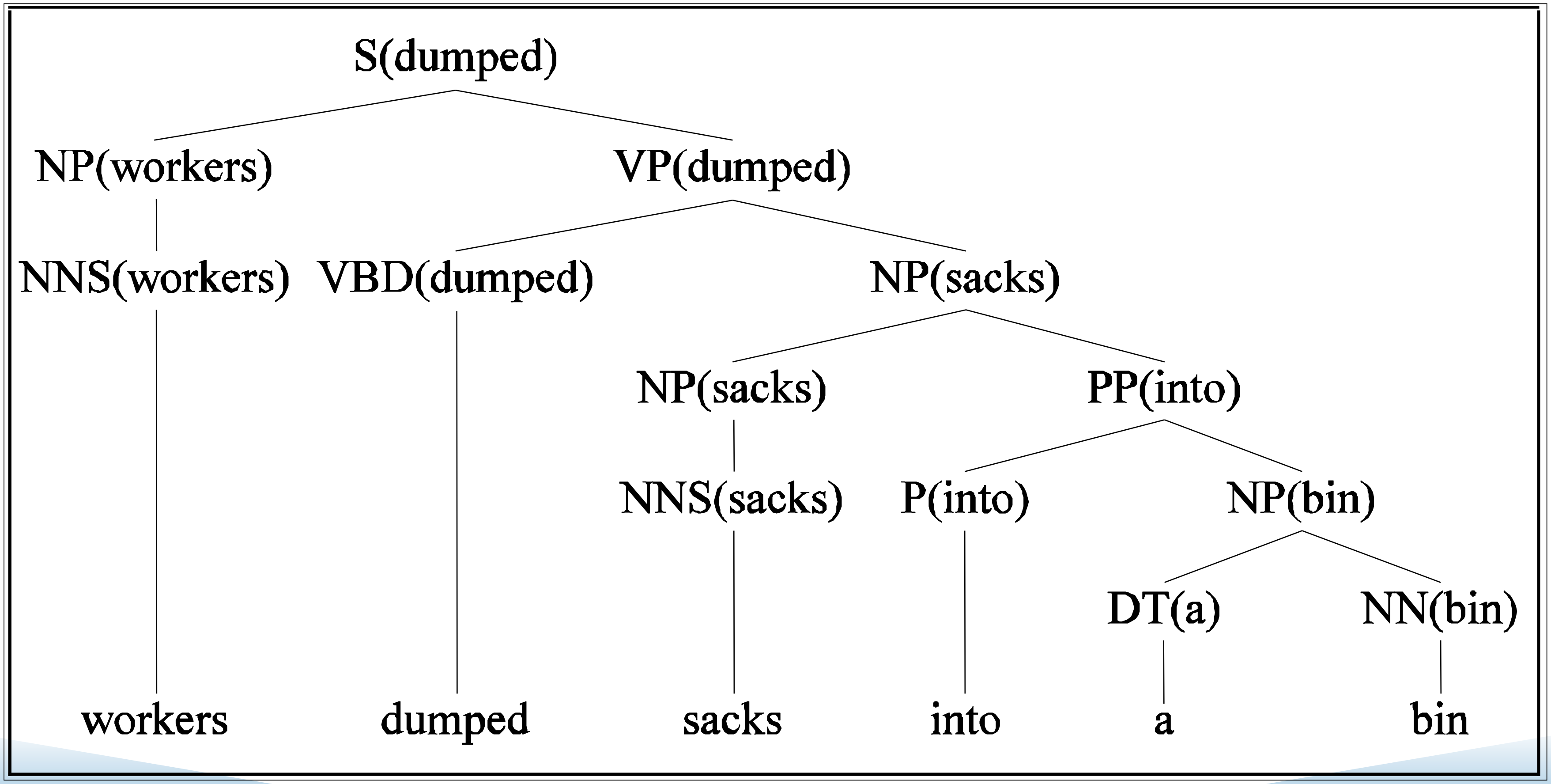
  (It's really more complicated than that but this will do.)

Example (right)

Attribute grammar



Example (wrong)



How?

- We used to have
  - VP -> V NP PP   P(rule|VP)
    - That's the count of this rule divided by the number of VPs in a treebank
- Now we have
  - VP(dumped)-> V(dumped) NP(sacks)PP(in)
  - P(r|VP ^ dumped is the verb ^ sacks is the head of the NP ^ in is the head of the PP)
  - Not likely to have significant counts in any treebank

Declare Independence

- When stuck, exploit independence and collect the statistics you can…
- We'll focus on capturing two things
  - Verb subcategorization
    - Particular verbs have affinities for particular VPs
  - Objects affinities for their predicates (mostly their mothers and grandmothers)
    - Some objects fit better with some predicates than others

Subcategorization

- Condition particular VP rules on their head… so

  r: VP -> V NP PP P(r|VP)

  Becomes

  P(r | VP ^ dumped)

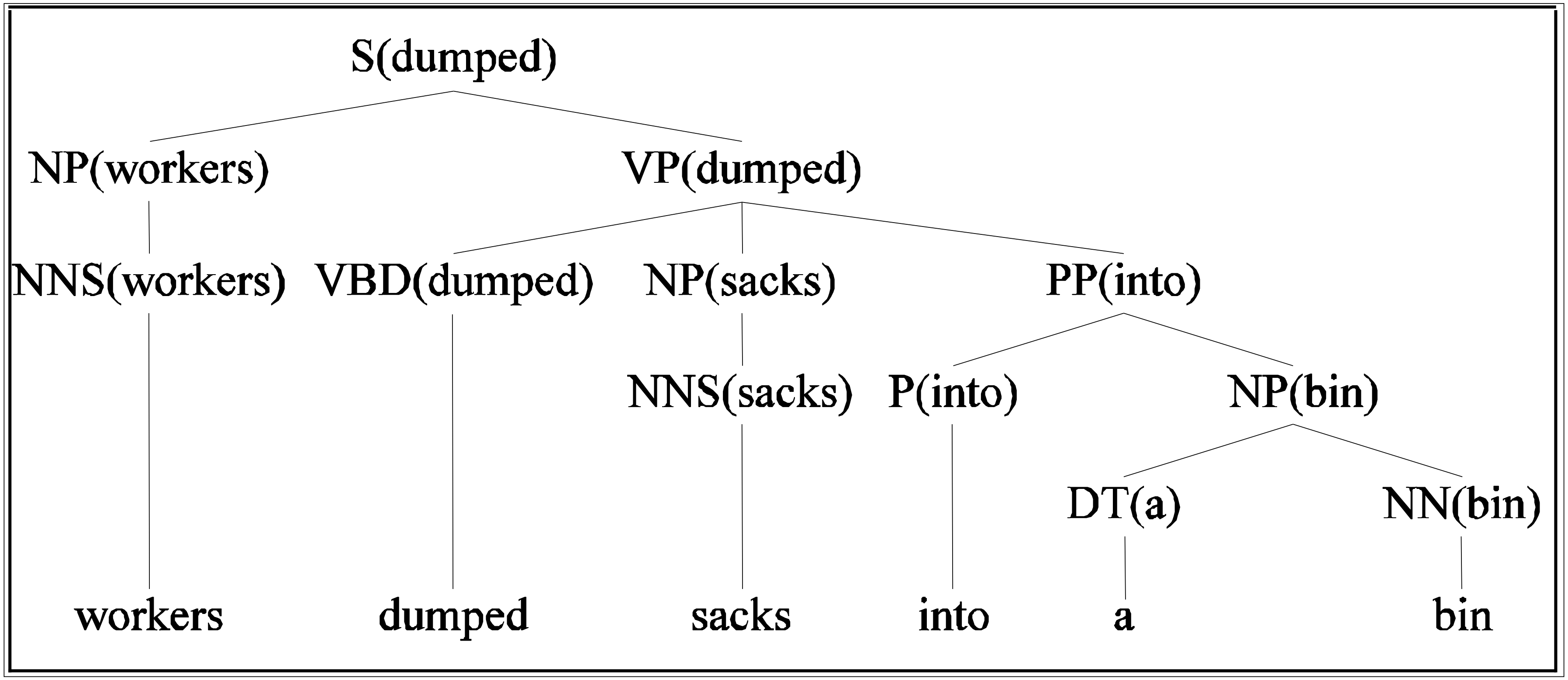
  What's the count?

  How many times was this rule used with (head) dump, divided by the number of VPs that dump appears (as head) in total

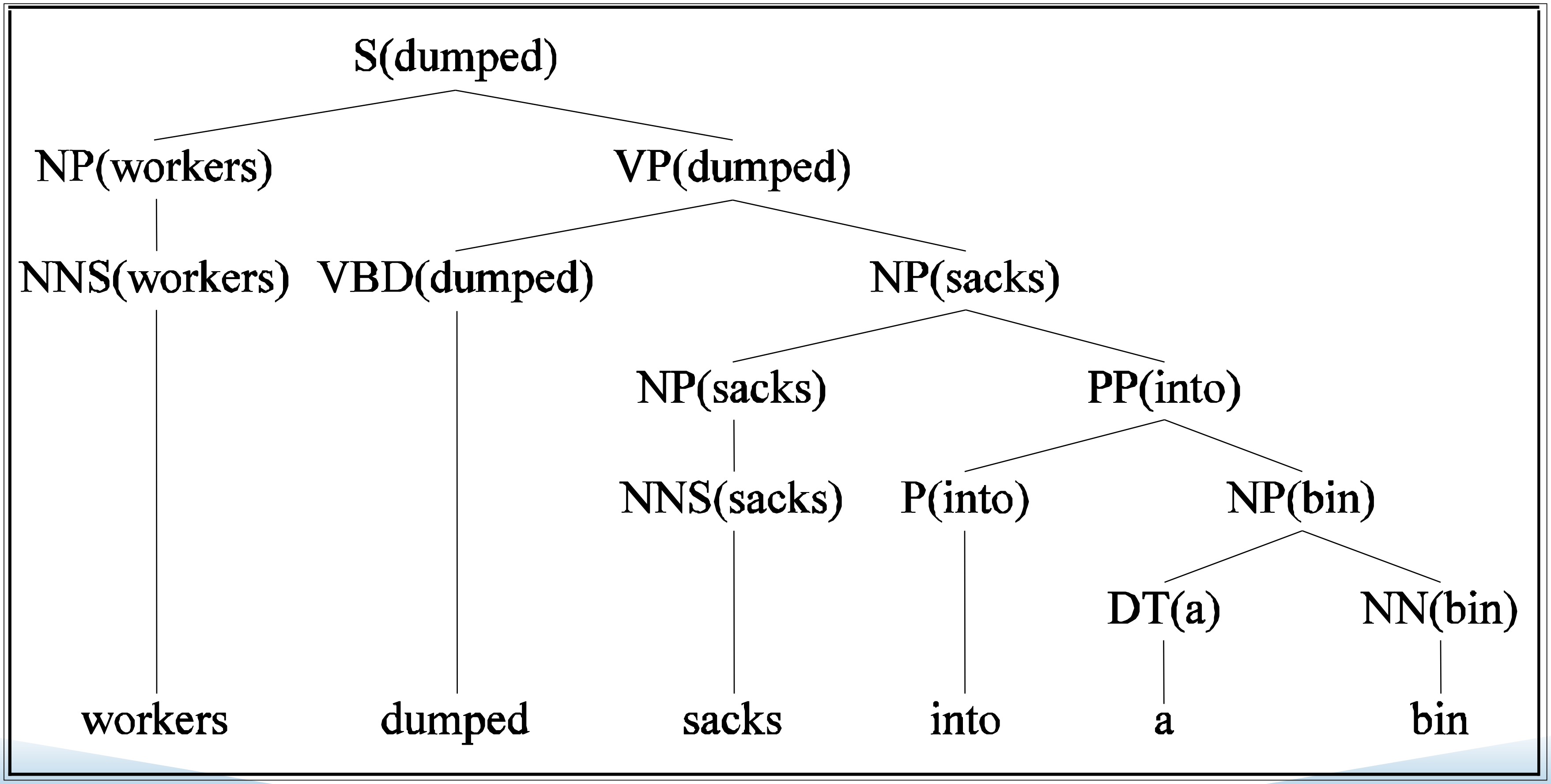
Preferences

- Subcat captures the affinity between VP heads (verbs) and the VP rules they go with.
- What about the affinity between VP heads and the heads of the other daughters of the VP
- Back to our examples…

Example (right)



Example (wrong)



Preferences

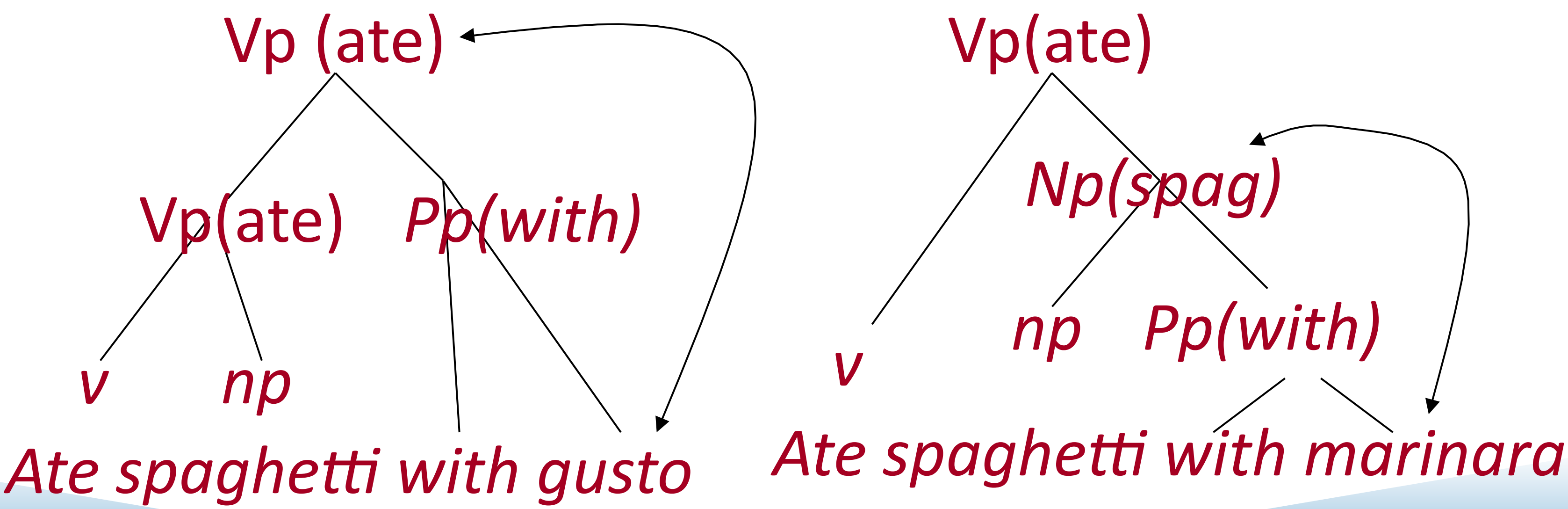
- The issue here is the attachment of the PP. So the affinities we care about are the ones between dumped and into vs. sacks and into.
- So count the places where dumped is the head of a constituent that has a PP daughter with into as its head and normalize
- Vs. the situation where sacks is a constituent with into as the head of a PP daughter.

Preferences (2)

- Consider the VPs
  - Ate spaghetti with gusto
  - Ate spaghetti with marinara
- The affinity of gusto for eat is much larger than its affinity for spaghetti
- On the other hand, the affinity of marinara for spaghetti is much higher than its affinity for ate

Preferences (2)

- Note the relationship here is more distant and doesn't involve a headword since gusto and marinara aren't the heads of the PPs.



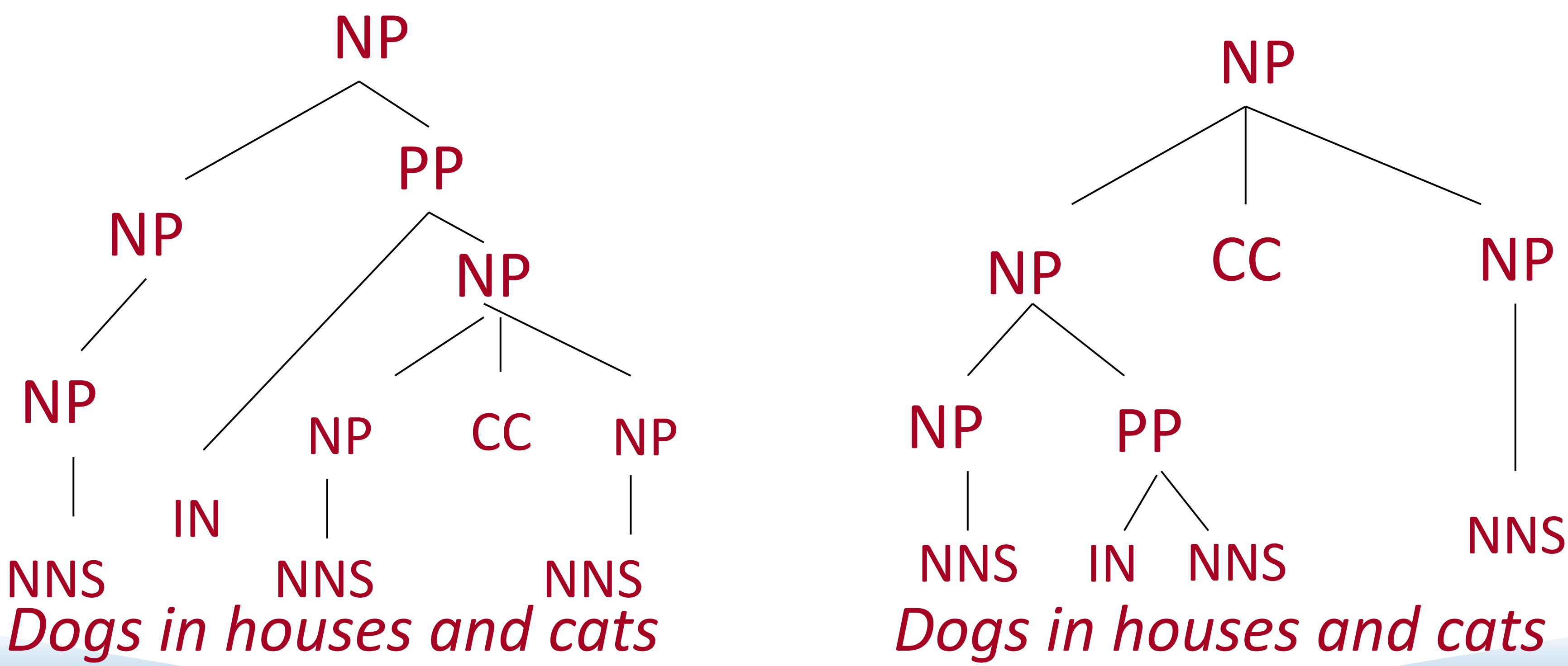
Summary

- Context-Free Grammars
- Parsing
  - Top Down, Bottom Up Metaphors
  - Dynamic Programming Parsers: CKY. Earley
- Disambiguation:
  - PCFG
  - Probabilistic Augmentations to Parsers
  - Treebanks

Other Issues with PCFGs

Other Issues with PCFGs

A Case of Coordinated Ambiguity



Conjunction

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
N → in
NNS → houses
CC → and
NNS → cats

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
N → in
NNS → houses
CC → and
NNS → cats

Here the two parses have identical rules, and therefore have identical probability under any assignment of PCFG rule probabilities

Structural Preferences: Close Attachment

- Example:
  A. President of (a company in Africa)
  B. (President of a company) in Africa
- Both parses have the same rules, therefore receive same probability under a PCFG
- "Close attachment" (structure A) is twice as likely in Wall Street Journal text.

Structural Preferences: Close Attachment

- **Previous example:**
  - John was believed to have been shot by Bill
- Here the low attachment analysis (Bill does the *shooting)* contains same rules as the high attachment analysis (Bill does the *believing),* so the two analyses receive same probability.

Adding “Heads”

- Each context-free rule has one "special" child that is the head of the rule, e.g.,

  S => NP VP

  VP => Vt NP

  NP => DT NN NN

- A core idea in syntax

  (e.g., see X-bar Theory, Head-Driven Phrase Structure Grammar)

- Some intuitions:
  - The central sub-constituent of each rule.
  - The semantic predicate in each rule.

Rules which Recover Heads:
An Example for NPs

- If rule contains NN, NNS, or NNP
  - Choose rightmost NN, NNS or NNP
- Else if rule contains NP
  - Choose leftmost NP
- Else if rule contains a JJ
  - Choose rightmost JJ
- Else if rule contains a CD
  - Choose right most CD
- Else choose the rightmost child

NP=> DT NNP NN
NP => DT NN NNP
NP => NP PP
NP => DT JJ
NP => DT