CSCI 5832
Natural Language Processing

Lecture 16
Jim Martin

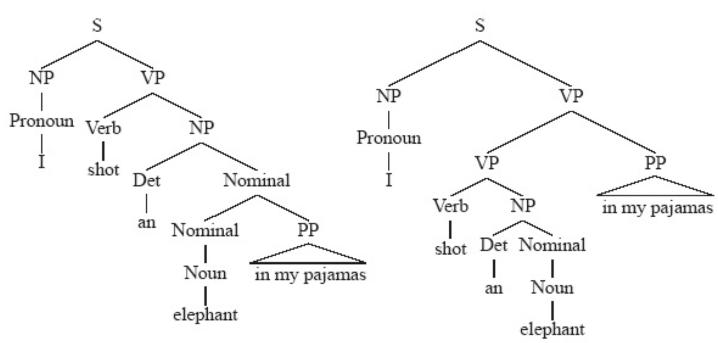
Today: March 13

- **Review CFG Parsing**
- **Probabilistic CFGs**
  - **Basic model**
  - **Lexicalized model**

Dynamic Programming Approaches

- **Earley**
  - **Top-down, no filtering, no restriction on grammar form**
- **CYK**
  - **Bottom-up, no filtering, grammars restricted to Chomsky-Normal Form (CNF)**
- **Details are not important...**
  - **Bottom-up vs. top-down**
  - **With or without filters**
  - **With restrictions on grammar form or not**

Back to Ambiguity



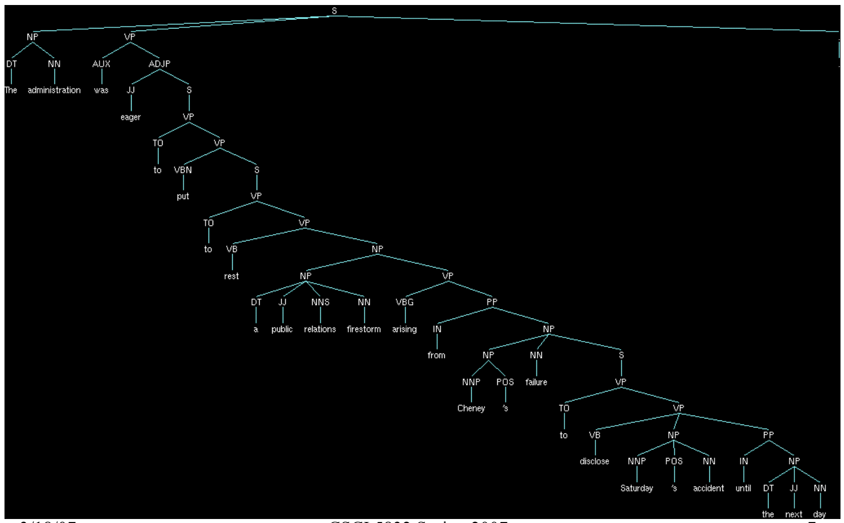
Disambiguation

- **Of course, to get the joke we need both parses.**
- **But in general we'll assume that there's one right parse.**
- **To get that we need knowledge: world knowledge, knowledge of the writer, the context, etc…**
- **Or maybe not..**

Disambiguation

- **Instead let's make some assumptions and see how well we do…**

Example



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)

Probability Model (1)

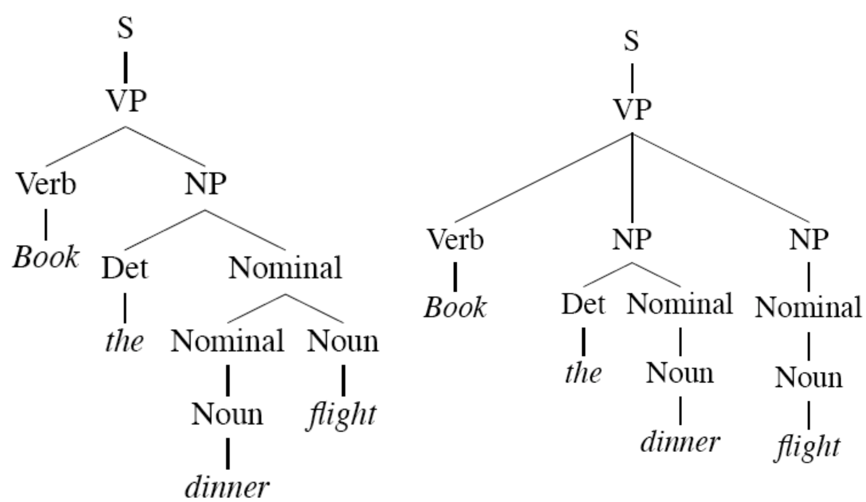
- A derivation (tree) consists of the bag of grammar rules that are in the tree
- The probability of a tree is just the product of the probabilities of the rules in the derivation.

$$P(T,S) = \prod_{node \in T} P(rule(n))$$

Probability Model (1.1)

- **The probability of a word sequence (sentence) is the probability of its tree in the unambiguous case.**
- **It's the sum of the probabilities of the trees in the ambiguous case.**
- **Since we can use the probability of the tree(s) as a proxy for the probability of the sentence…**
  - **PCFGs give us an alternative to N-Gram models as a kind of language model.**

Example



Rule Probabilities

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
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flights	.40	Noun	→	flights	.40

$2.2 * 10^{-6}$ $6.1 * 10^{-7}$

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.**

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

Smoothing

- Using this method do we need to worry about smoothing these probabilities?

Inside/Outside

- If we don't have a treebank, but we do have a grammar can we get reasonable probabilities?
- Yes. Use a prob parser to parse a large corpus and then get the counts as above.
- But
  - In the unambiguous case we're fine
  - In ambiguous cases, weight the counts of the rules by the probabilities of the trees they occur in.

Inside/Outside

- **But…**
- **Where do those probabilities come from?**
- **Make them up. And then re-estimate them.**
- **This sounds a lot like….**

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

- **Use CKY as the backbone of the algorithm**
- **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**
  - **$S\text{->}_0 NP_i VP_j$**

The probability of this S is…

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

The green stuff is already known if we're using some kind of sensible DP approach.

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?)**

CKY

Where does the max go?

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
      table[j − 1, j] ← {A | A → words[j] ∈ grammar }
      for i ← from j − 2 downto 0 do
          for k ← i + 1 to j − 1 do
              table[i,j] ← table[i,j] ∪
                          {A | A → BC ∈ grammar,
                               B ∈ table[i,k],
                               C ∈ table[k,j] }
```

? ?

Probabilistic CKY (buggy)

function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar \}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$ **and** $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] > P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{j, B, C\}$
  return BUILD_TREE(*back*[1, LENGTH(*words*), *S*]), *table*[1, LENGTH(*words*), *S*]

Break

- **Faculty Search Colloquia**
  - **1 B 22**
  - **Tues and Thursdays 3:30 - 5:00 for the next N weeks**
  - **All systems, PL and SE folks**
  - **Lunch**
  - **You should go**

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**
  - **Doesn't really work**
    - **Most probable parse isn't usually the right one (the one in the treebank test set).**

Solution 1

- **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**

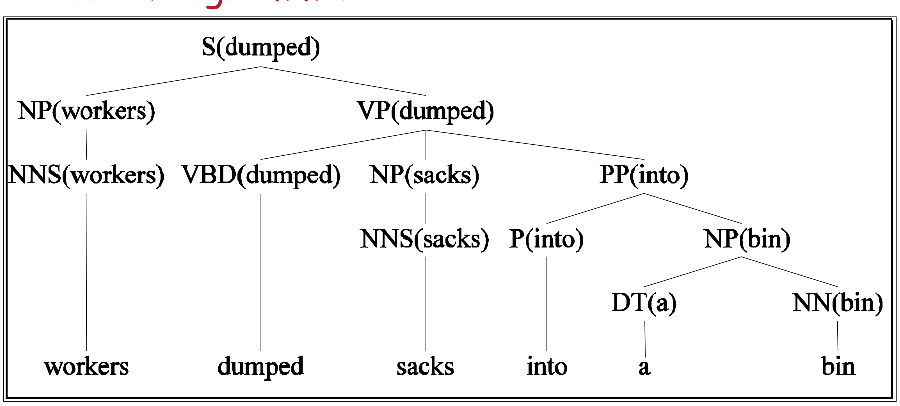
Heads

- **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**

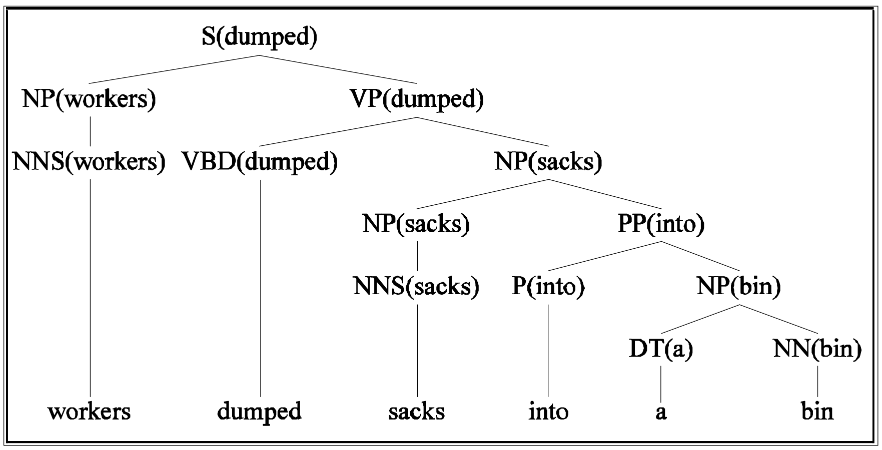
  (It's really more complicated than that but this will do.)

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)

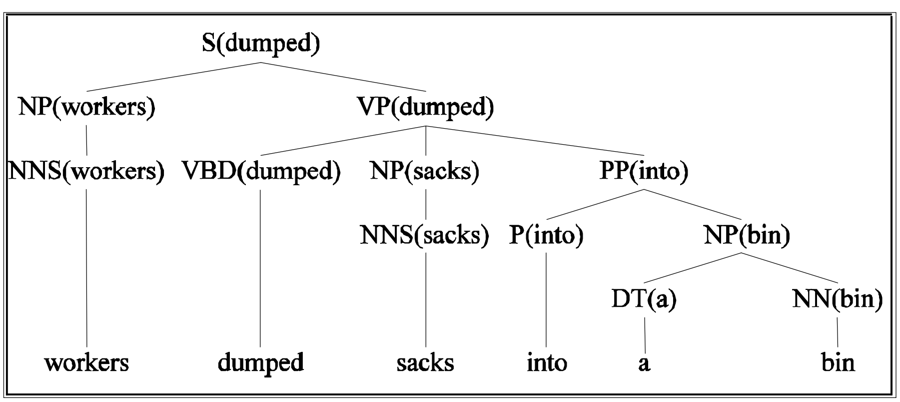
  What's the count?

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

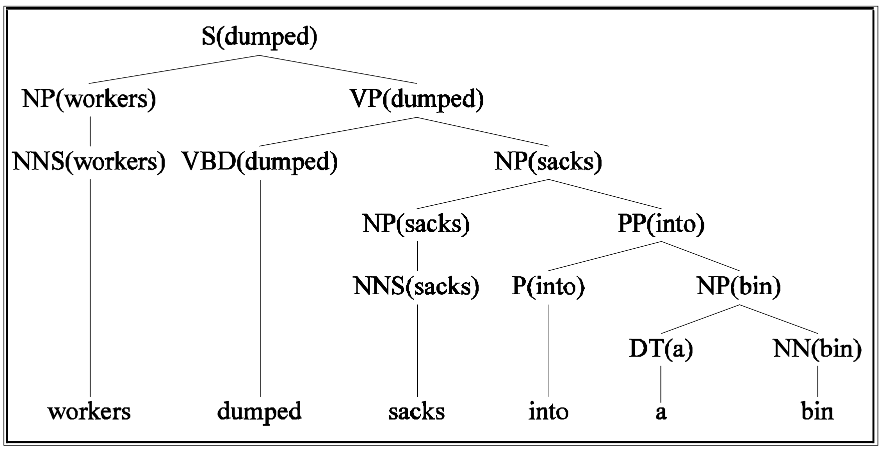
Preferences

- 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.**

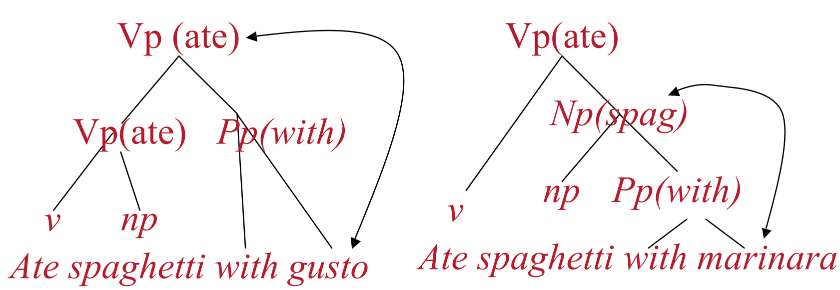
Preferences (2)

- **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**

3/18/07 CSCI 5832 Spring 2007 37

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.**



3/18/07 CSCI 5832 Spring 2007 38

Next Time

- **Finish up 13.**
  - **Rule re-writing approaches**
  - **Evaluation**