CSCI 5832
Natural Language Processing

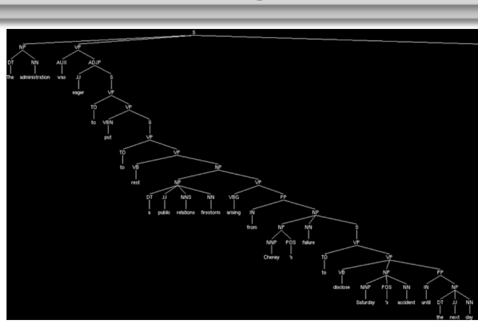
Jim Martin

Lecture 17

Today 3/13

- Statistical Parsing

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

3/13/08 4

Basic Probability Model

- 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.

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

3/13/08 5

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.

3/13/08 6

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.

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

Prob CKY

- Alter CKY so that the probabilities of constituents are stored on the way up…
  - Probability of a new constituent A derived from the rule A -> BC is:
    - P(A-> B C) * P(B) * P(C)
    - Where P(B) and P(C) are already in the table
    - But what we store is the MAX probability over all the A rules.

Prob CKY

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                           and its probability
  for j ← from 1 to LENGTH(words) do
     for all { A | A → words[j] ∈ grammar }
       table[j−1, j, A] ← P(A → words[j])
     for i ← from j−2 downto 0 do
        for k ← i+1 to j−1 do
           for all { A | A → BC ∈ grammar,
                        and table[i,k,B] > 0 and table[k,j,C] > 0 }
              if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                 table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                 back[i,j,A] ← {k,B,C}
     return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

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).

3/13/08 10

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

3/13/08 11

Heads

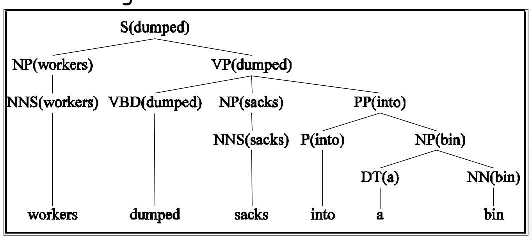
- To do that we're going to make use of the notion of the head of a phrase
  - The head of an NP is its noun
  - The head of a VP is its verb
  - The head of a PP is its preposition

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

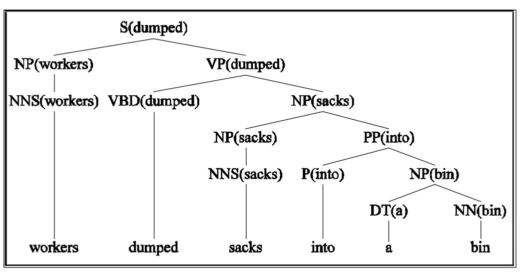
3/13/08 12

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(into)
  - P(r|VP ^ dumped is the verb ^ sacks is the head of the NP ^ into 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

3/13/08 16

Subcategorization

- Condition particular VP rules on their head… so

  r15: VP -> V NP PP P(r|VP)

  Becomes

  P(r15 | 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

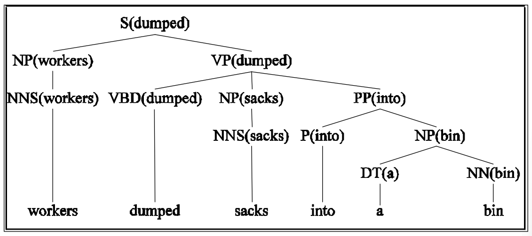
3/13/08 17

Preferences

- Verb subcategorization captures the affinity between VP heads (verbs) and the VP rules they go with.
  - That is the affinity between a node and one of its daughter nodes.
- What about the affinity between VP heads and the heads of the other daughters of the VP
- Back to our examples…

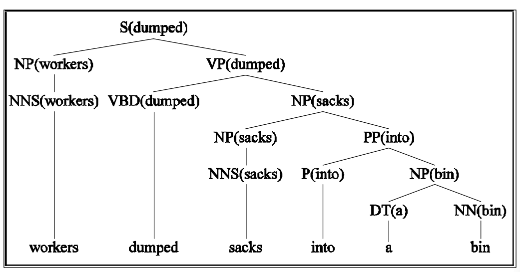
3/13/08 18

Example (right)



3/13/08 19

Example (wrong)



3/13/08 20

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.

3/13/08 21

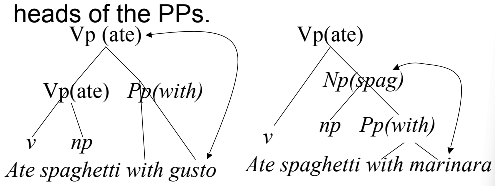
Preferences (2)

- Consider the VPs
  - Ate spaghetti with gusto
  - Ate spaghetti with marinara
- Here the heads of the PPs are the same (with) so that won't help.
- But 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 (we hope).

3/13/08 22

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.



3/13/08 23

Note

- In case someone hasn't pointed this out yet, this lexicalization stuff is a thinly veiled attempt to incorporate semantics into the syntactic parsing process…
  - Duhh..,. Picking the right parse requires the use of semantics.

3/13/08 24

Break

- Quiz
  - Chapter 12: 12.1 through 12.6
    - CFGs, Major English phrase types, problems with CFGs, relation to finite-state methods
  - Chapter 13: All except 13.4.3
    - CKY, Earley, partial parsing, sequence labeling
  - Chapter 14: 14.1 through14.6.1
    - Basic prob CFG model, getting the counts, prob CKY, problems with the model, lexicalization, and grammar rewriting
- Bring a cheat sheet.

3/13/08 25

Rule Rewriting

- An alternative to using these kinds of probabilistic lexical dependencies is to rewrite the grammar so that the rules do capture the regularities we want.
  - By splitting and merging the non-terminals in the grammar.
  - Example: split NPs into different classes…

3/13/08 26

NPs

- Our CFG rules for NPs don't condition on where the rule is applied (they're context-free remember)
- But we know that not all the rules occur with equal frequency in all contexts.

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

3/13/08 27

Other Examples

- Lots of other examples like this in the TreeBank
  - Many at the part of speech level
  - Recall that many decisions made in annotation efforts are directed towards improving annotator agreement, not towards doing the right thing.
    - Often this involves conflating distinct classes into a larger class
      - TO, IN, Det, etc.

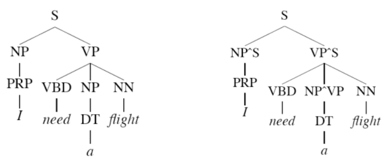
Rule Rewriting

- Three approaches
  - Use linguistic intuitions to directly rewrite rules
    - NP_Obj and the NP_Subj approach
  - Automatically rewrite the rules using context to capture some of what we want
    - Ie. Incorporate context into a context-free approach
  - Search through the space of rewrites for the grammar that maximizes the probability of the training set

Local Context Approach

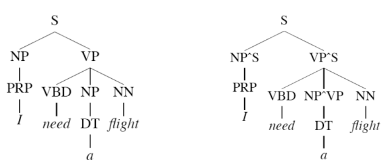
- Condition the rules based on their parent nodes
  - This splitting based on tree-context captures some of the linguistic intuitions

Parent Annotation



- Now we have non-terminals NP^S and NP^VP that should capture the subject/object and pronoun/full NP cases.

Parent Annotation



- Recall what's going on here. We're in effect rewriting the treebank, thus rewriting the grammar.
- And changing the probabilities since they're being derived from different counts…
  - And if we're splitting what's happening to the counts?

Auto Rewriting

- If this is such a good idea we may as well apply a learning approach to it.
- Start with a grammar (perhaps a treebank grammar)
- Search through the space of splits/merges for the grammar that in some sense maximizes parsing performance on the training/development set.

Auto Rewriting

- Basic idea…
  - Split every non-terminal into two new non-terminals across the entire grammar (X becomes X1 and X2).
  - Duplicate all the rules of the grammar that use X, dividing the probability mass of the original rule almost equally.
  - Run EM to readjust the rule probabilities
  - Perform a merge step to back off the splits that look like they don't really do any good.

3/13/08 34

Last Point

- Statistical parsers are getting quite good, but its still quite silly to expect them to come up with the correct parse given only statistically massage syntactic information.
- But its not so crazy to think that they can come up with the right parse among the top-N parses.
- Lots of current work on
  - Re-ranking to make the top-N list even better.

3/13/08 35

Evaluation

- So if it's unreasonable to expect these probabilistic parsers to get the right answer what can we expect from them and how do we measure it.
- Look at the content of the trees rather than the entire trees.
  - Assuming that we have gold standard trees for test sentences

3/13/08 36

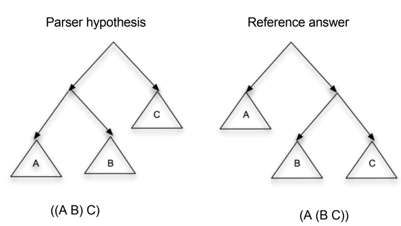
Evaluation

- Precision
  - What fraction of the sub-trees in our parse matched corresponding sub-trees in the reference answer
    - How much of what we're producing is right?
- Recall
  - What fraction of the sub-trees in the reference answer did we actually get?
    - How much of what we should have gotten did we get?

3/13/08 37

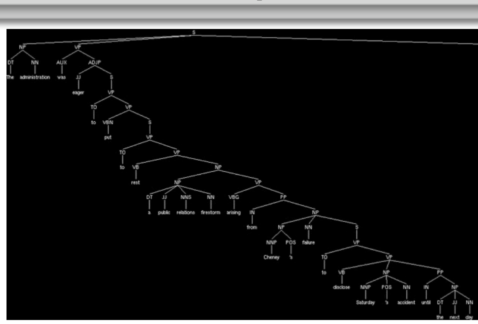
Evaluation

- Crossing brackets



3/13/08 38

Example



3/13/08 39