CSCI 5832
Natural Language Processing

Jim Martin

Lecture 16

3/11/08 1

Today 3/11

- Review
  - Partial Parsing & Chunking
    - Sequence classification
- Statistical Parsing

3/11/08 2

Back to Sequences

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W)$$
$$= \underset{T}{\operatorname{argmax}} P(W|T)P(T)$$
$$= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})$$

- HMMs

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W)$$
$$= \underset{T}{\operatorname{argmax}} \prod_i P(tag_i|word_i, tag_{i-1})$$

- MEMMs

And whatever other features you choose to use!

3/11/08 3

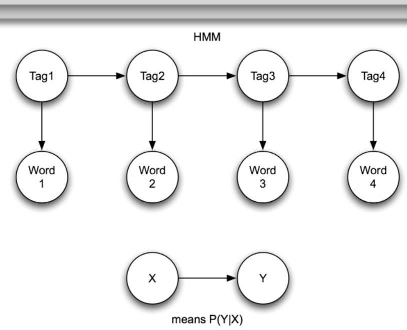
Back to Viterbi

$$v_t(j) = \max_{1 \le i \le N-1} v_{t-1}(i)\, P(s_j|s_i, o_t); \quad 1 < j < N, 1 < t < T$$

- The value for a cell is found by examining all the cells in the previous column and multiplying by the posterior for the current column (which incorporates the transition as a factor, along with any other features you like).

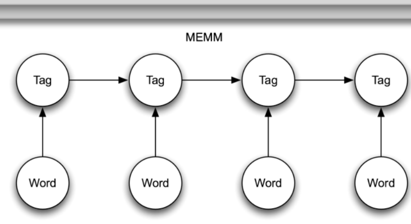
3/11/08 4

HMMs vs. MEMMs



3/11/08 5

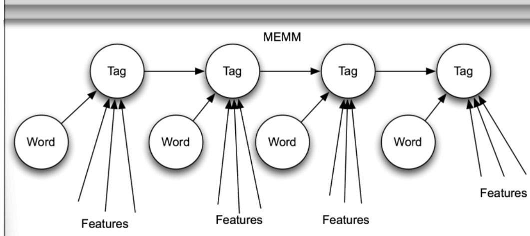
HMMs vs. MEMMs



$$P(T|W) = \prod P(t_i|t_{i-1}, w_i)$$

3/11/08 6

HMMs vs. MEMMs

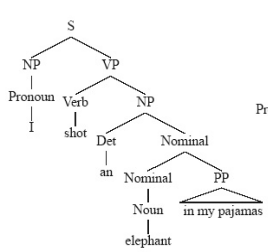


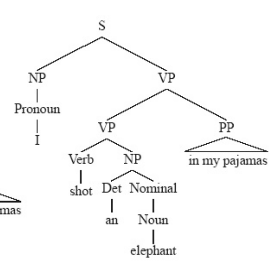
$$P(T|W) = \prod P(t_i|t_{i-1}, w_i, f_i)$$

Dynamic Programming Parsing 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..

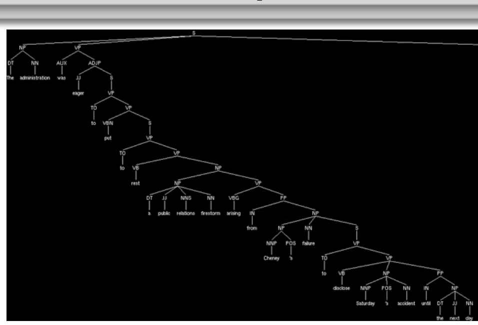
3/11/08 10

Disambiguation

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

3/11/08 11

Example



3/11/08 12

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

3/11/08 13

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)

3/11/08 14

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.

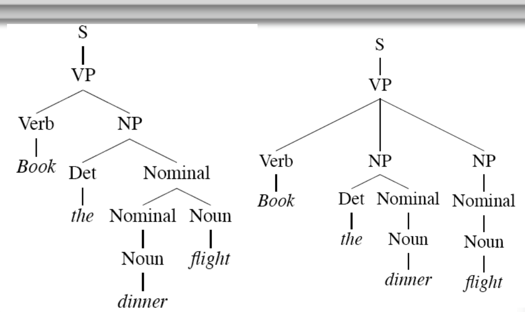
$$P(T,S) = \prod_{node \in T} P(rule(n))$$

3/11/08 15

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.

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

3/11/08 19

Smoothing

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

3/11/08 20

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.

3/11/08 21

Inside/Outside

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

3/11/08 22

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

3/11/08 23

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

3/11/08 24

What does that last bullet mean?

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

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.

3/11/08 25

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?)

3/11/08 26

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] }
```

? ?

3/11/08 27

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]
```

3/11/08 28

Break

- Next assignment details have been posted. See the course web page. It’s due March 20.
- Quiz is a week from today.

3/11/08 29

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 (shhh)
    - Most probable parse isn’t usually the right one (the one in the treebank test set).

3/11/08 30

Solution 1

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

3/11/08 31

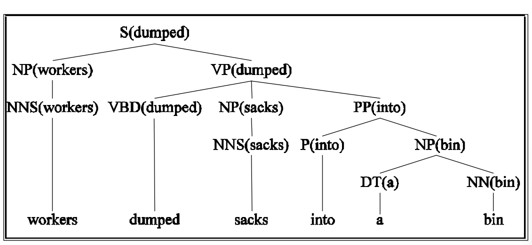
Heads

- 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.)

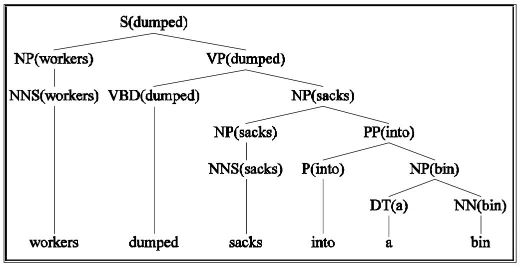
3/11/08 32

Example (right)



3/11/08 33

Example (wrong)



3/11/08 34

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

3/11/08 35

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 VP rules
  - Objects affinities for their predicates (mostly their mothers and grandmothers)
    - Some objects fit better with some predicates than others

3/11/08 36

Subcategorization

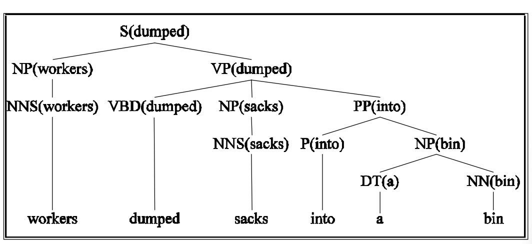
- Condition particular VP rules on their head… so
  r: VP -> V NP PP P(r|VP)
  Becomes
  P(r | VP ^ dumped)

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

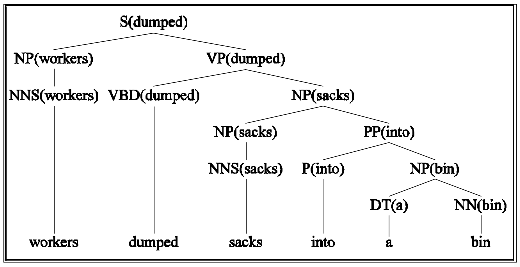
Preferences

- 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)



3/11/08 40

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.

3/11/08 41

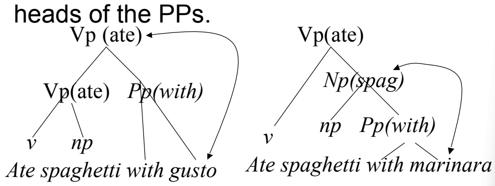
Preferences (2)

- 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

3/11/08 42

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.



Next Time

- Finish up 14
  - Rule re-writing approaches
  - Evaluation