PARSING

David Kauchak
CS159 – Spring 2011

some slides adapted from Ray Mooney

Admin

- Updated slides/examples on backoff with absolute discounting (I'll review them again here today)
- Assignment 2
- Watson vs. Humans (tonight-Wednesday)

Backoff models: absolute discounting

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

- Subtract some absolute number from each of the counts (e.g. 0.75)
  - will have a large effect on low counts
  - will have a small effect on large counts

Backoff models: absolute discounting

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

What is α(xy)?

Backoff models: absolute discounting

see the dog	1
see the cat	2
see the banana	4
see the man	1
see the woman	1
see the car	1

the Dow Jones	10
the Dow rose	5
the Dow fell	5

p(rose | the Dow) = ?

p(jumped | the Dow) = ?

p(cat | see the) = ?

p(puppy | see the) = ?

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

Backoff models: absolute discounting

see the dog	1
see the cat	2
see the banana	4
see the man	1
see the woman	1
see the car	1

p(cat | see the) = ?

$$\frac{2 - D}{10} = \frac{2 - 0.75}{10} = .125$$

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

Backoff models: absolute discounting

see the dog	1
see the cat	2
see the banana	4
see the man	1
see the woman	1
see the car	1

p(puppy | see the) = ?

α(see the) = ?

How much probability mass did we reserve/discount for the bigram model?

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

Backoff models: absolute discounting

see the dog	1
see the cat	2
see the banana	4
see the man	1
see the woman	1
see the car	1

p(puppy | see the) = ?

α(see the) = ?

$$\frac{\text{\# of } \textit{types} \text{ starting with "see the"} * D}{\text{count("see the")}}$$

For each of the unique trigrams, we subtracted D/count("see the") from the probability distribution

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

Backoff models: absolute discounting

see the dog	1
see the cat	2
see the banana	4
see the man	1
see the woman	1
see the car	1

p(puppy | see the) = ?

α(see the) = ?

$$\frac{\text{\# of } \textit{types} \text{ starting with “see the” * D}}{\text{count(“see the”)}}$$

$$reserved\_mass(\text{see the}) = \frac{6 * D}{10} = \frac{6 * 0.75}{10} = 0.45$$

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & \textit{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \textit{otherwise} \end{cases}$$

distribute this probability mass to all bigrams that we backed off to

Calculating α

- We have some number of bigrams we're going to backoff to, i.e. those X where C(see the X) = 0, that is unseen trigrams starting with “see the”
- When we backoff, for each of these, we'll be including their probability in the model: P(X | the)
- α is the normalizing constant so that the sum of these probabilities equals the reserved probability mass

$$\sum_{X : C(\text{see the X}) = 0} p(\text{X} \mid \text{the}) = reserved\_mass(\text{see the})$$

Calculating α

- We can calculate α two ways
  - Based on those we haven't seen:

$$\alpha(\text{see the}) = \frac{reserved\_mass(\text{see the})}{\sum_{X : C(\text{see the X}) = 0} p(\text{X} \mid \text{the})}$$

  - Or, more often, based on those we do see:

$$\alpha(\text{see the}) = \frac{reserved\_mass(\text{see the})}{1 - \sum_{X : C(\text{see the X}) > 0} p(\text{X} \mid \text{the})}$$

Calculating α in general: trigrams

- Calculate the reserved mass

$$\text{reserved\_mass(bigram)} = \frac{\text{\# of } \textit{types} \text{ starting with bigram * D}}{\text{count(bigram)}}$$

- Calculate the sum of the backed off probability. For bigram “A B”:

$$1 - \sum_{X : C(\text{A B X}) > 0} p(\text{X} \mid \text{B})$$

either is fine in practice, the left is easier

$$\sum_{X : C(\text{A B X}) = 0} p(\text{X} \mid \text{B})$$

- Calculate α

$$\alpha(\text{A B}) = \frac{reserved\_mass(\text{A B})}{1 - \sum_{X : C(\text{A B X}) > 0} p(\text{X} \mid \text{B})}$$

1 – the sum of the bigram probabilities of those trigrams that we saw starting with bigram A B

Calculating α in general: bigrams

- Calculate the reserved mass

$$\text{reserved\_mass(unigram)} = \frac{\text{\# of } \textit{types} \text{ starting with unigram * D}}{\text{count(unigram)}}$$

- Calculate the sum of the backed off probability. For bigram "A B":

$$1 - \sum_{X: C(A\ X) > 0} p(X) \qquad \sum_{X: C(A\ X) = 0} p(X)$$

either is fine in practice, the left is easier

- Calculate α

$$\alpha(A) = \frac{reserved\_mass(A)}{1 - \sum_{X: C(A\ X) > 0} p(X)}$$

1 – the sum of the unigram probabilities of those bigrams that we saw starting with word A

Calculating backoff models in practice

- Store the αs in another table
  - If it's a trigram backed off to a bigram, it's a table keyed by the bigrams
  - If it's a bigram backed off to a unigram, it's a table keyed by the unigrams
- Compute the αs during training
  - After calculating all of the probabilities of seen unigrams/bigrams/trigrams
  - Go back through and calculate the αs (you should have all of the information you need)
- During testing, it should then be easy to apply the backoff model with the αs pre-calculated

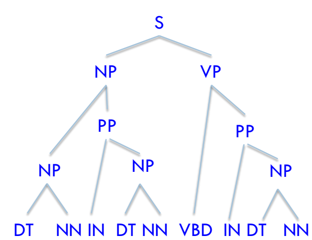
Backoff models: absolute discounting

$$\text{reserved\_mass} = \frac{\text{\# of } \textit{types} \text{ starting with bigram * D}}{\text{count(bigram)}}$$

- Two nice attributes:
  - decreases if we've seen more bigrams
    - should be more confident that the unseen trigram is no good
  - increases if the bigram tends to be followed by lots of other words
    - will be more likely to see an unseen trigram

Syntactic structure

(S (NP (NP (DT the) (NN man)) (PP (IN in) (NP (DT the) (NN hat)))) (VP (VBD ran) (PP (TO to (NP (DT the) (NN park))))))



The man in the hat ran to the park.

CFG: Example

- Many possible CFGs for English, here is an example (fragment):
  - S → NP VP
  - VP → V NP
  - NP → DetP N | AdjP NP
  - AdjP → Adj | Adv AdjP
  - N → boy | girl
  - V → sees | likes
  - Adj → big | small
  - Adv → very
  - DetP → a | the

Grammar questions

- Can we determine if a sentence is grammatical?
- Given a sentence, can we determine the syntactic structure?
- Can we determine how likely a sentence is to be grammatical? to be an English sentence?
- Can we generate candidate, grammatical sentences?

Parsing

- Parsing is the field of NLP interested in automatically determining the syntactic structure of a sentence
- parsing can also be thought of as determining what sentences are "valid" English sentences

Parsing

- Given a CFG and a sentence, determine the possible parse tree(s)

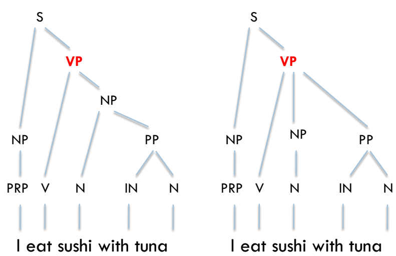
S -> NP VP
NP -> PRP
NP -> N PP
NP -> N
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

I eat sushi with tuna

What parse trees are possible for this sentence?

How did you figure it out?

Parsing



S -> NP VP
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

What is the difference between these parses?

Parsing

- Given a CFG and a sentence, determine the possible parse tree(s)

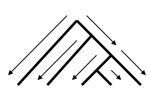
S -> NP VP
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

I eat sushi with tuna

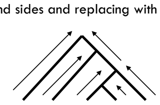
approaches?
algorithms?

Parsing

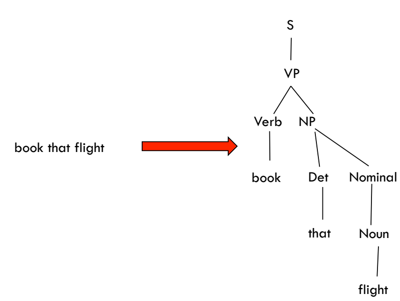
- Top-down parsing
  - start at the top (usually S) and apply rules
  - matching left-hand sides and replacing with right-hand sides



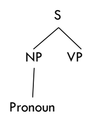
- Bottom-up parsing
  - start at the bottom (i.e. words) and build the parse tree up from there
  - matching right-hand sides and replacing with left-hand sides



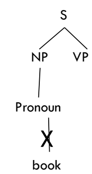
Parsing Example



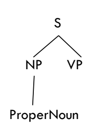
Top Down Parsing



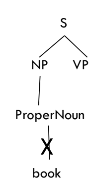
Top Down Parsing



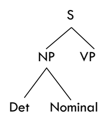
Top Down Parsing



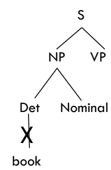
Top Down Parsing



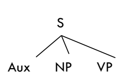
Top Down Parsing



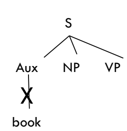
Top Down Parsing



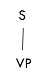
Top Down Parsing



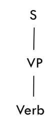
Top Down Parsing



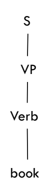
Top Down Parsing



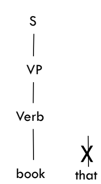
Top Down Parsing



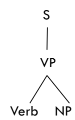
Top Down Parsing



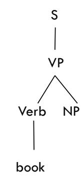
Top Down Parsing



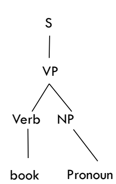
Top Down Parsing



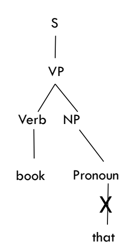
Top Down Parsing



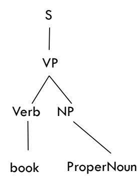
Top Down Parsing



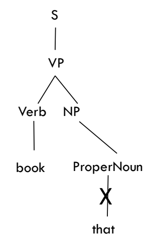
Top Down Parsing



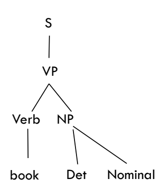
Top Down Parsing



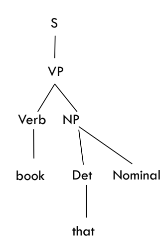
Top Down Parsing



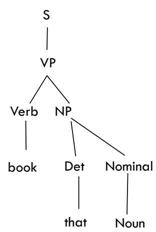
Top Down Parsing



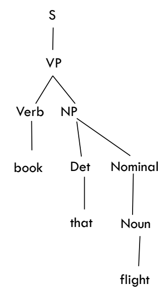
Top Down Parsing



Top Down Parsing



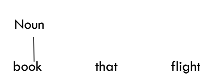
Top Down Parsing



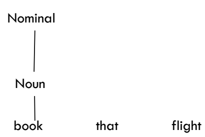
Bottom Up Parsing

book that flight

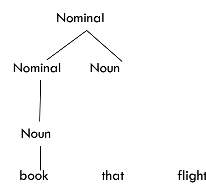
Bottom Up Parsing



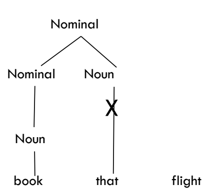
Bottom Up Parsing



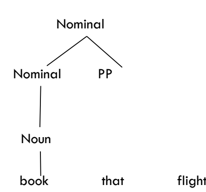
Bottom Up Parsing



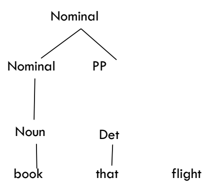
Bottom Up Parsing



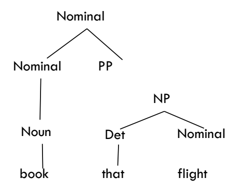
Bottom Up Parsing



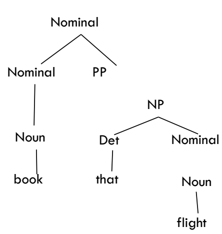
Bottom Up Parsing



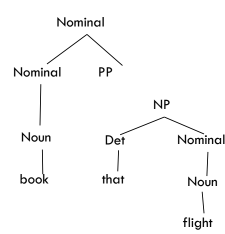
Bottom Up Parsing



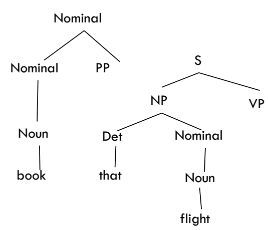
Bottom Up Parsing



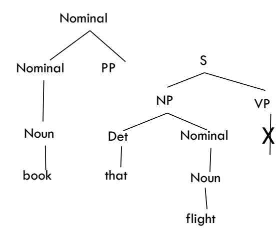
Bottom Up Parsing



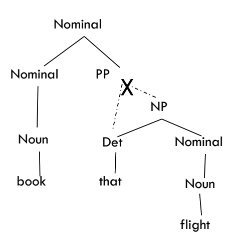
Bottom Up Parsing



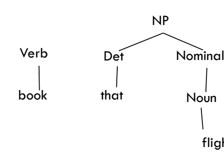
Bottom Up Parsing



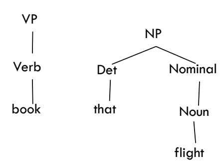
Bottom Up Parsing



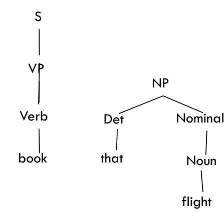
Bottom Up Parsing



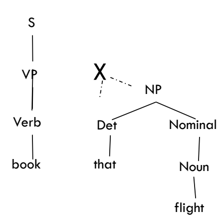
Bottom Up Parsing



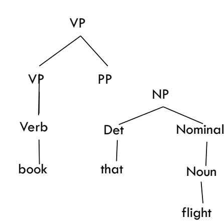
Bottom Up Parsing



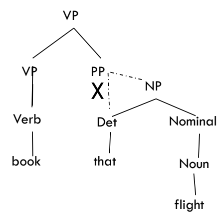
Bottom Up Parsing



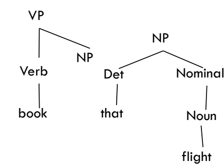
Bottom Up Parsing



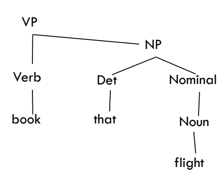
Bottom Up Parsing



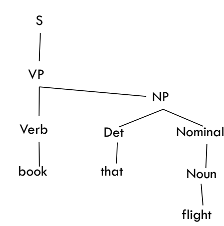
Bottom Up Parsing



Bottom Up Parsing



Bottom Up Parsing



Parsing

- Pros/Cons?
  - Top-down:
    - Only examines parses that could be valid parses (i.e. with an S on top)
    - Doesn't take into account the actual words!
  - Bottom-up:
    - Only examines structures that have the actual words as the leaves
    - Examines sub-parses that may not result in a valid parse!

Why is parsing hard?

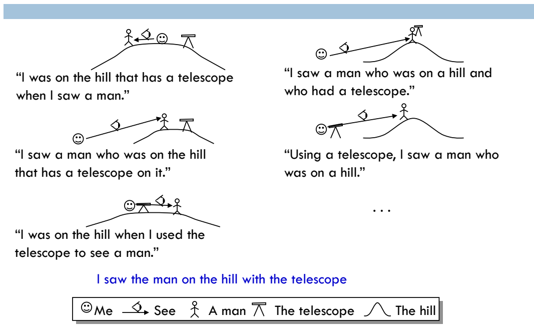
- Actual grammars are large
- Lots of ambiguity!
  - Most sentences have many parses
  - Some sentences have a lot of parses
  - Even for sentences that are not ambiguous, there is often ambiguity for subtrees (i.e. multiple ways to parse a phrase)

Why is parsing hard?

I saw the man on the hill with the telescope

What are some interpretations?

Structural Ambiguity Can Give Exponential Parses



Dynamic Programming Parsing

- To avoid extensive repeated work you must cache intermediate results, specifically found constituents
- Caching (memoizing) is critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve O(n^3) recognition time where *n* is the length of the input string.

Dynamic Programming Parsing Methods

- **CKY** (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and requires first normalizing the grammar.
- **Earley parser** is based on top-down parsing and does not require normalizing grammar but is more complex.
- These both fall under the general category of **chart parsers** which retain completed constituents in a chart

CKY

- First grammar must be converted to **Chomsky normal form (CNF)** in which productions must have either exactly 2 non-terminal symbols on the RHS or 1 terminal symbol (lexicon rules).
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart)

CNF Grammar

S -> VP	S -> VP
VP -> VB NP	VP -> VB NP
VP -> VB NP PP	VP -> VP2 PP
NP -> DT NN	VP2 -> VB NP
NP -> NN	NP -> DT NN
NP -> NP PP	NP -> NN
PP -> IN NP	NP -> NP PP
DT -> the	PP -> IN NP
IN -> with	DT -> the
VB -> film	IN -> with
VB -> trust	VB -> film
NN -> man	VB -> trust
NN -> film	NN -> man
NN -> trust	NN -> film
	NN -> trust