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

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow NP \ VP )</td>
<td>0.80</td>
<td>( Det \rightarrow that )</td>
<td>0.05</td>
</tr>
<tr>
<td>( S \rightarrow Aux \ NP \ VP )</td>
<td>0.15</td>
<td>( Det \rightarrow the )</td>
<td>0.80</td>
</tr>
<tr>
<td>( S \rightarrow VP )</td>
<td>0.05</td>
<td>( Det \rightarrow a )</td>
<td>0.15</td>
</tr>
<tr>
<td>( NP \rightarrow Det \ Nom )</td>
<td>0.20</td>
<td>( Noun \rightarrow book )</td>
<td>0.10</td>
</tr>
<tr>
<td>( NP \rightarrow Proper-Noun )</td>
<td>0.35</td>
<td>( Noun \rightarrow flights )</td>
<td>0.50</td>
</tr>
<tr>
<td>( NP \rightarrow Nom )</td>
<td>0.05</td>
<td>( Noun \rightarrow meal )</td>
<td>0.40</td>
</tr>
<tr>
<td>( NP \rightarrow Pronoun )</td>
<td>0.40</td>
<td>( Verb \rightarrow include )</td>
<td>0.30</td>
</tr>
<tr>
<td>( Nom \rightarrow Noun )</td>
<td>0.75</td>
<td>( Verb \rightarrow want )</td>
<td>0.40</td>
</tr>
<tr>
<td>( Nom \rightarrow Noun \ Noun )</td>
<td>0.20</td>
<td>( Aux \rightarrow can )</td>
<td>0.40</td>
</tr>
<tr>
<td>( Nom \rightarrow Proper-Noun \ Noun )</td>
<td>0.05</td>
<td>( Aux \rightarrow does )</td>
<td>0.30</td>
</tr>
<tr>
<td>( Aux \rightarrow do )</td>
<td>0.30</td>
<td>( Proper-Noun \rightarrow TWA )</td>
<td>0.40</td>
</tr>
<tr>
<td>( Proper-Noun \rightarrow Denver )</td>
<td>0.40</td>
<td>( Pronoun \rightarrow you )</td>
<td>0.40</td>
</tr>
<tr>
<td>( Pronoun \rightarrow I )</td>
<td>0.60</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PCFG

(a) S
   /   |
  Aux  NP
   |
   V
   /   |
  NP  NP
   |
   Nom
   |
   Pro
   |
   can
   |
   you

(b) S
   /   |
  Aux  NP
   |
   V
   /   |
  NP  NP
   |
   Nom
   |
   Pro
   |
   can
   |
   you

<table>
<thead>
<tr>
<th>Rules</th>
<th>P</th>
<th>Rules</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>→</td>
<td>S</td>
<td>→</td>
</tr>
<tr>
<td>Aux</td>
<td>NP</td>
<td>Aux</td>
<td>NP</td>
</tr>
<tr>
<td>NP</td>
<td>VP</td>
<td>VP</td>
<td>NP</td>
</tr>
<tr>
<td>V</td>
<td>NP</td>
<td>V</td>
<td>NP</td>
</tr>
<tr>
<td>NP</td>
<td>Nom</td>
<td>NP</td>
<td>Nom</td>
</tr>
<tr>
<td>Nom</td>
<td>P Noun</td>
<td>Nom</td>
<td>P Noun</td>
</tr>
<tr>
<td>P Noun</td>
<td>Noun</td>
<td>Noun</td>
<td>Noun</td>
</tr>
<tr>
<td>Noun</td>
<td>→</td>
<td>Noun</td>
<td>→</td>
</tr>
<tr>
<td>→ Can</td>
<td></td>
<td>→ Can</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>→</td>
<td>NP</td>
<td>→</td>
</tr>
<tr>
<td>→ Pro</td>
<td></td>
<td>→ Pro</td>
<td></td>
</tr>
<tr>
<td>Pro</td>
<td>→</td>
<td>Pro</td>
<td>→</td>
</tr>
<tr>
<td>→ you</td>
<td></td>
<td>→ you</td>
<td></td>
</tr>
<tr>
<td>Verb</td>
<td>→</td>
<td>Verb</td>
<td>→</td>
</tr>
<tr>
<td>→ book</td>
<td></td>
<td>→ book</td>
<td></td>
</tr>
<tr>
<td>P Noun</td>
<td>→</td>
<td>P Noun</td>
<td>→</td>
</tr>
<tr>
<td>→ TWA</td>
<td></td>
<td>→ TWA</td>
<td></td>
</tr>
<tr>
<td>Noun</td>
<td>→</td>
<td>Noun</td>
<td>→</td>
</tr>
<tr>
<td>→ flights</td>
<td></td>
<td>→ flights</td>
<td></td>
</tr>
</tbody>
</table>
• 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

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

- \( P(T,S) = P(T)P(S \mid T) = P(T); \) since \( P(S \mid T) = 1 \)

\[
\begin{align*}
P(T_l) &= .15 \times .40 \times .05 \times .05 \times .35 \times .75 \times .40 \times .40 \times .40 \\
    &= 1.5 \times 10^{-6}
\end{align*}
\]

\[
\begin{align*}
P(T_r) &= .15 \times .40 \times .40 \times .05 \times .05 \times .75 \times .40 \times .40 \times .40 \\
    &= 1.7 \times 10^{-6}
\end{align*}
\]
• 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.
• 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.
(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)
• 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
  - \( S \rightarrow _0 NP_i VP_j \)

  The probability of the S is...
  \( P(S \rightarrow NP \ VP) \times P(NP) \times P(VP) \)

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

```
S(dumped)
  NP(workers)
    NNS(workers)  VBD(dumped)
      workers  dumped
  VP(dumped)
    NP(sacks)
      NNS(sacks)  P(into)
        sacks  into
    PP(into)
      NP(bin)
        DT(a)  NN(bin)
          a  bin
```
Example (wrong)

S(dumped)
  NP(workers)
    NNS(workers)
      workers
  VP(dumped)
    VBD(dumped)
      dumped
  NP(sacks)
    NNS(sacks)
      sacks
    P(into)
      into
    PP(into)
      DT(a)
    NP(bin)
      NN(bin)
      bin
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
• 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
  
  \[ r: \text{VP} \rightarrow \text{V} \text{ NP} \text{ PP} \ P(r | \text{VP}) \]
  
  Becomes
  
  \[ P(r | \text{VP} ^ \text{dumped}) \]

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 (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*
• 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

Dogs in houses and cats

Dogs in houses and cats

Slides adapted from Collins NLP
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 rightmost CD

• Else choose the rightmost child

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