Statistical Parsing

Gerald Penn
CS224N
[based on slides by Christopher Manning]

(Head) Lexicalization of PCFGs

- The head word of a phrase gives a good representation of the phrase's structure and meaning
- Puts the properties of words back into a PCFG

Parser via classification decisions: Charniak (1997)

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is "top-down" like a regular PCFG (but actual computation is bottom-up)

Charniak (1997) example

a. \( h = \text{profits}; c = \text{NP} \)
b. \( ph = \text{rose}; pc = s \)
c. \( P(h|ph,c,pc) \)
d. \( P(r|h,c,pc) \)
Lexicalization sharpens probabilities: rule expansion

- E.g., probability of different verbal complement frames (often called “subcategorizations”)

<table>
<thead>
<tr>
<th>Local Time</th>
<th>come</th>
<th>take</th>
<th>think</th>
<th>want</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → v</td>
<td>0.5%</td>
<td>1.4%</td>
<td>4.8%</td>
<td>0.7%</td>
</tr>
<tr>
<td>VP → v PP</td>
<td>9.7%</td>
<td>22.1%</td>
<td>0.0%</td>
<td>16.4%</td>
</tr>
<tr>
<td>VP → v NP</td>
<td>2.8%</td>
<td>18.6%</td>
<td>0.0%</td>
<td>3.1%</td>
</tr>
<tr>
<td>VP → v V</td>
<td>6.8%</td>
<td>3.3%</td>
<td>7.1%</td>
<td>0.9%</td>
</tr>
<tr>
<td>VP → v V65</td>
<td>4.9%</td>
<td>5.3%</td>
<td>73.0%</td>
<td>0.2%</td>
</tr>
<tr>
<td>VP → v V5</td>
<td>0.1%</td>
<td>5.7%</td>
<td>0.2%</td>
<td>0.9%</td>
</tr>
<tr>
<td>VP → v V57</td>
<td>0.7%</td>
<td>0.2%</td>
<td>0.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>VP → v V67</td>
<td>0.1%</td>
<td>1.5%</td>
<td>0.2%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Lexicalization sharpens probabilities: Predicting heads

- “Bilexical probabilities”
- \( p(\text{prices} | \text{np, plural}) = 0.013 \)
- \( p(\text{prices} | \text{np, plural, NP}) = 0.013 \)
- \( p(\text{prices} | \text{np, plural, NP, S}) = 0.025 \)
- \( p(\text{prices} | \text{np, plural, NP, S, v-past}) = 0.052 \)
- \( p(\text{prices} | \text{np, plural, NP, S, v-past, felt}) = 0.146 \)

Charniak (1997) linear interpolation/shrinkage

\[
P(h|p,b,c) = \lambda_1(e)P_{\text{ideal}}(h|p,b,c) + \lambda_2(e)P_{\text{data}}(h|c,b,c) + \lambda_3(e)P_{\text{ideal}}(h|c,b) + \lambda_4(e)P_{\text{data}}(h|c)\]

- \( \lambda_1(e) \) is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- \( C(p,b) \) is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

Charniak (1997) shrinkage example

| \( P(\text{prf} | \text{rose, NP, S}) \) | \( P(\text{cor} | \text{prf}, JJ, NP) \) |
|-----------------|----------------|
| 0.245           | 0.0150         |
| 0.000627        | 0.00533        |
| 0.000557        | 0.00418        |

- Allows utilization of rich highly conditioned estimates, but smooths when sufficient data is unavailable
- One can’t just use MLEs; one commonly sees previously unseen events, which would have probability 0.

Sparseness & the Penn Treebank

- The Penn Treebank – 1 million words of parsed English WSJ – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADIP, of which only 6 aren’t ‘how much’ or ‘how many’, but there is an infinite space of these
  - Few conversational/incompetent (or risk assessment and evaluation)
- Most of the probabilities that you would like to compute, you can’t compute

Quiz question!

- Which of the following is also the beginning of a WHADIP?
  - a) how are
  - b) how cruel
  - c) how about
  - d) however long
Sparseness & the Penn Treebank (2)

- Many parse preferences depend on biliteral statistics: likelihoods of relationships between pairs of words (compound nouns, PP attachments, …)
- Extremely sparse, even on topics central to the WSJ:
  - stocks plummeted; 2 occurrences
  - stocks slumped; 1 occurrence
  - stocks skyrocketed; 0 occurrences
  - stocks skyrocketed; 0 occurrences
- So far there has been very modest success in augmenting the Penn Treebank with extra unannotated materials or using semantic classes – once there is more than a little annotated training data.
  - Cf. Charniak 1997, Charniak 2000, but see McCallum et al. 2006 (the recent self-training work is rather more successful)

Complexity of lexicalized PCFG parsing

- Time charged:
  - \[ A(d_1), B(d_2), C(d_3) \rightarrow n^3 \]
  - \[ A(d_1), B(d_2), C(d_3) \rightarrow G^2 \]
- Done naively, \[ G^2 \] is huge (\[ G^2 = g^{4n^2} \] unworkable)
  - \[ A, B, C \rightarrow g^2 \]
  - \[ d_1, d_2 \rightarrow n^2 \]

Running time is \( O \{ g^2 \times n^2 \}! \)

Complexity of exhaustive lexicalized PCFG parsing

- Work such as Collins (1997) and Charniak (1997) is \( O(n^2) \) – but uses heuristic search to be fast in practice
- Eisner and Satta (2000, etc.) have explored various ways to parse more restricted classes of biliteral grammars in \( O(n) \) or \( O(n^2) \) time
  - Near algorithmic stuff
  - See example later from dependency parsing

Refining the node expansion probabilities

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes.
- But it is bad because of data sparseness.
- A pure dependency, one child at a time, model is worse.
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997, Charniak 2000)
  - Cf. the accurate unlexicalized parsing discussion

Collins (1997, 1999); Bikel (2004)

- Collins (1999): also a generative model
- Underlying lexicalized PCFG has rules of form
  \[ P \rightarrow L_1 L_{i-1} \cdots L_1 HR_1 \cdots R_k \]
- A more elaborate set of grammar transforms and factorizations to deal with data sparseness and interesting linguistic properties
  - Each child is generated in turn; given \( P \) has been generated, generate \( H \), then generate modifying nonterminals from head-adjacent outward with some limited conditioning
Overview of Collins’ Model

Modifying nonterminals generated in two steps

Smoothing for head words of modifying nonterminals

Collins model ... and linguistics

Bilexical statistics: Is use of maximal context of \( P_{Rh} \) useful?

Choice of heads

<table>
<thead>
<tr>
<th>Back-off level</th>
<th>( P_{Rh} (P(Rh^+)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( R_{Rh}, \text{coord, punct, } P_n, W_n^{Rh}, D_n, \text{subcat} _\text{nost} )</td>
</tr>
<tr>
<td>1</td>
<td>( R_{Rh}, \text{coord, punct, } P_n, W_n^{Rh}, D_n, \text{subcat} _\text{nost} )</td>
</tr>
<tr>
<td>2</td>
<td>( R_{Rh} )</td>
</tr>
</tbody>
</table>

• Collins (1999): “Most importantly, the model has parameters corresponding to dependencies between pairs of headwords.”

• Gildea (2001) reproduced Collins’ Model 1 (like regular model, but no subcats)
  • Removing maximal back-off level from \( P_{Rh} \) resulted in only 0.5% reduction in F-measure
  • Gildea’s experiment somewhat unconvincing to the extent that ‘his’ model’s performance was lower than Collins’ reported results

• Collins had 3 generative models: Models 1 to 3
• Especially as you work up from Model 1 to 3, significant linguistic modeling is present:
  • Distance measure favors close attachments
  • Model is sensitive to punctuation
  • Distinguish base NP from full NP with post-modifiers
  • Coordination feature
  • Mark gapped subjects
  • Model of subcategorization; arguments vs. adjuncts
  • Slash feature/gap threading treatment of displaced constituents
  • Didn’t really get clear gains from this last one.

• If not bilexical statistics, then surely choice of heads is important to parser performance...
• Chiang and Bikel (2002): parsers performed decently even when all head rules were of form “if parent is X, choose left/rightmost child”
• Parsing engine in Collins Model 2—emulation mode:
  • LR 88.55% and LP 88.80% on 500 (sent. len. ≤40 words)
  • Compared to LR 89.9%, LP 90.1%
Use of maximal context of $P_{\text{HP}}$  

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>LP</th>
<th>CBs</th>
<th>Cb2s</th>
<th>Cb3s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td>89.9</td>
<td>90.1</td>
<td>0.78</td>
<td>66.8</td>
<td>89.2</td>
</tr>
<tr>
<td>No bigrams</td>
<td>89.5</td>
<td>90.0</td>
<td>0.80</td>
<td>66.8</td>
<td>88.8</td>
</tr>
</tbody>
</table>

Performance on §00 of Penn Treebank on sentences of length ≤40 words

---

Bilingual statistics are used often  

- The 1.4% use of bilingual dependencies suggests they don’t play much of a role in parsing.
- But the parser parses many (very) incorrect theories.
- So, instead of asking how often the decoder can use bilingual probability on average, ask how often while pursuing the top-scoring theory.
- Answering question by having parser control-constructed parser own output.
  - This is normal in BIB.
- Some BIB.
- Harder to see on consistency.
- Recognizing the parser made use of bilingual statistics shell up to 28.9%.
- So, used often, but hard to identify overall parsing accuracy.
- Error analysis suggests explanation.
  - Data suggests that includes target errors are usually very similar to slow that do not, so it means almost all differences in terms of accuracy.

---


- There was nothing maximum entropy about it. It was a cleverly smoothed generative model.
- Smoothes estimates by smoothing ratio of conditional terms (which are a bit like segment features): $P(t_i|p_i, t_{i-1}) / P(t_i|p_i, t_{i-2})$.
- Biggest improvement is actually that generative model predicts head tag first and then does $P(w_i|...)$.
- Like Collins (1999).
- Modelizes rules similar to Collins (1999).
- Gets 90.1% LP/LR F score on sentences ≤ 40 wds.

---


Can you automatically find good symbols?
- Brackets are known.
- Base categories are known.
- Induce subcategories.
-大洋splitmerge category refinement.

---

Number of phrasal subcategories

---
POS tag splits' commonest words: effectively a class-based model

- Proper Nouns (NNP):
  - NNP-10:  Ort Nov Sept
  - NNP-12: John Robert James
  - NNP-2: J E L
  - NNP-1: Bush Noreiga Peters
  - NNP-13: New San Wall
  - NNP-14: York Francisco Street

- Personal pronouns (PRP):
  - PRP-0: It He She It
  - PRP-1: It He She They
  - PRP-2: It Them Him

Recent Parsing Results...

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 at all words</th>
<th>F1 at all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning unlexicized 2003</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Moro et al. Simple EM latent state 2005</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Charniak generative, initialized “moresets advised” 2000</td>
<td>90.1</td>
<td>89.5</td>
</tr>
<tr>
<td>Petrov and Klein NACLI 2007</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>Charniak et al. discriminative IJCAI 2005</td>
<td>92.0</td>
<td>91.4</td>
</tr>
</tbody>
</table>

Statistical parsing inference: The General Problem

- Someone gives you a PCFG G
- For any given sentence, you might want to:
  - Find the best parse according to G
  - Find a bunch of reasonably good parses
  - Find the total probability of all parses licensed by G
- Techniques:
  - CKY, for best parse; can extend it:
    - To k-best: memory done, 2k high space and time cost - k^2 time/ space cost, but there are cleverer algorithms (Kuang and Chiang 2005, brown Pulleyblank)
    - To all parses, sum probability: the inside algorithm
  - Beam search (like in MT)
    - Appends/fixed based search
    - Mainly useful if just want the best parse

Parsing as search definitions

- Grammar symbols: S, NP, @S->NP
- Parse items/edges represent a grammar symbol over a span:
  - \[S(0.3)\]
    - NP(0.2)
    - VP(2.3)
- Backtraces/traversals represent the combination of adjacent edges into a larger edges:
  - The[0.1]

Parse trees and parse triangles

- A parse tree can be viewed as a collection of edges and traversals
- A parse triangle groups edges over the same span

Parsing as search: The parsing directed B-hypergraph

(Klein and Manning [36])
Chart example: classic picture

Space and Time Bounds

CKY Parsing
- In CKY parsing, we visit edges tier by tier:
  - Guarantees correctness by working inside-out.
  - Build all small bits before any larger bits that could possibly require them.
  - Exhaustive: the goal is in the last tier!

Agenda-based parsing
- For general grammars:
  - Start with a table for recording $\delta(X_{i,j})$
    - Records the best score of a parse of $X$ over $[i,j]$
    - If the scores are negative log probabilities, then entries start at $-\infty$ and small is good.
    - This can be a sparse or dense map
    - Again, you may want to record backtracks (traversals) as well, like CKY
  - Step 1: Initialize with the sentence and lexicon:
    - For each word $w$ and each tag $t$
      - Set $\delta(w,t) = \log(prob(w,t))$

Agenda-based parsing
- Keep a list of edges called an agenda
  - Edges are triples $(X_{i,j})$
  - The agenda is a priority queue
  
  Every time the score of some $\delta(X_{i,j})$ improves (i.e., gets lower):
  - Stick the edge $(X_{i,j})$ score into the agenda
  - Update the traceback for $\delta(X_{i,j})$ if you're storing them

Agenda-Based Parsing
- The agenda is a holding zone for edges.
- Visit edges by some ordering policy.
  - Combine edge with already-visited edges.
  - Resulting new edges go wait in the agenda.
- We might revisit parse items: A new way to form an edge might be a better way.
Agenda-based parsing

- Step II: While agenda not empty
  - Get the “next” edge \( (X,i,j) \) from the agenda
  - Fetch all compatible neighbors \((Y,j,k)\) or \((Z,i,l)\)
    - Compatible means that there are rules \( A \rightarrow Y \) or \( B \rightarrow Z \)
  - Build all parent edges \((A,i,j)\) or \((B,i,k)\) found
    - \( R(A,j) = 5(R(j,k) + R(Y,j,k) + P(A,k)Y) \)
  - If we’ve improved \( R(A,j) \), then stick it on the agenda
  - Also project unary rules:
    - Fetch all unary rules \( A \rightarrow z \), score \((A,j)\) built from this rule on \((X,i,j)\) and put on agenda if you’ve improved \( R(A,j) \)
  - When do we know we have a parse for the root?

---

Agenda-based parsing

- Open questions:
  - Agenda priority: What did “next” mean?
  - Efficiency: how do we do as little work as possible?
  - Optimality: how do we know when we find the best parse of a sentence?

- If we use \( S(X,I,J) \) as the priority:
  - Each edge goes on the agenda at most once
  - When an edge pops off the agenda, its best parse is known (why?)
  - This is basically uniform cost search (i.e., Dijkstra’s algorithm) (Cormen, Leiserson, and Rivest 1990; Knuth 1970)