Ling/CSE 472: Introduction to Computational Linguistics

5/16/12
Statistical Parsing

Overview

- Why statistical parsing?
- PCFGs
- Estimating rule probabilities
- Probabilistic CKY
- Ways to improve PCFG
- Evaluation

Why statistical parsing?

- Parsing = making explicit structure that is inherent (implicit) in natural language strings
- Useful for: language modeling + any app that needs access to the meaning of sentences
- Most application scenarios that use parser output want just one parse
  - Have to choose among all the possible analyses
- Most application scenarios need robust parsers
  - Need some output for every input, even if its not grammatical

PCFGs

- N: a set of non-terminal symbols
- Σ: a set of terminal symbols (disjoint from N)
- R: a set of rules, of the form A -> β [p]
    - A: non-terminal
    - β: string of symbols from Σ or N
    - p: probability of β given A
- S: a designated start symbol

PCFGs

- How does this differ from CFG?
- How do we use it to calculate the probability of a parse?
- The probability of a sentence?
- What assumptions does that require?

PCFGs

- How does this differ from CFG? -- added probability to each rule
- How do we use it to calculate the probability of a parse? -- multiply probability of each rule used (= $P(T|S) = P(T)$)
- The probability of a sentence? -- sum of probability of all trees
- What assumptions does that require? -- expansion of a node does not depend on the context

PCFGs: Why

- When would you want to know the probability of a parse?
- When would you want to know the probability of a sentence?

How to estimate the rule probabilities

- Get a Treebank
- Gather all instances of each non-terminal
- For each expansion of the non-terminal (= rule), count how many times it occurs

$$P(\alpha \to \beta \mid \alpha) = \frac{\mathrm{Count}(\alpha \to \beta)}{\mathrm{Count}(\alpha)}$$

Using the probabilities for best-first parsing

- Probabilistic CKY: in each cell, store just the most probable edge for each non-terminal
- Probabilities based on rule probability plus daughter edge probabilities

```
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]
```

Work through an example:
Kim adores snow in Oslo

$\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}$	$[.8]$	$\mathrm{NOM} \mid \mathrm{NP} \rightarrow \mathrm{Kim}$	$[.01]$
$\mathrm{VP} \rightarrow \mathrm{V}\ \mathrm{NP}$	$[.2]$	$\mathrm{NOM} \mid \mathrm{NP} \rightarrow \mathrm{snow}$	$[.01]$
$\mathrm{VP} \rightarrow \mathrm{VP}\ \mathrm{PP}$	$[.3]$	$\mathrm{NOM} \mid \mathrm{NP} \rightarrow \mathrm{Oslo}$	$[.01]$
$\mathrm{PP} \rightarrow \mathrm{P}\ \mathrm{NP}$	$[.9]$	$\mathrm{V} \mid \mathrm{VP} \rightarrow \mathrm{adores}$	$[.02]$
$\mathrm{NP} \rightarrow \mathrm{NOM}\ \mathrm{PP}$	$[.2]$	$\mathrm{V} \mid \mathrm{VP} \rightarrow \mathrm{snores}$	$[.01]$
		$\mathrm{P} \rightarrow \mathrm{in}$	$[.1]$

Why statistical parsing? (reprise)

- Most application scenarios that use parser output want just one parse
  - Have to choose among all the possible analyses
  - How does PCFG solve this problem?
- Most application scenarios need robust parsers
  - Need some output for every input, even if its not grammatical
  - How does PCFG solve this problem?

Problems with PCFG

- Independence assumption is wrong
  - What does “independence assumption” mean?
  - What is the evidence that it’s wrong?
- Not sensitive to lexical dependencies
  - What does that mean?

Ways to improve PCFGs

- Split the non-terminals
    - Rename each non-terminal based on its parent (NP-S vs. NP-VP)
    - Hand-written rules to split pre-terminal categories
    - Automatically search for optimal splits through split and merge algorithm
- Lexicalized PCFGs: add identity of lexical head to each node label
    - Data sparsity problem -> smoothing again

Evaluating parsing

- How would you do extrinsic evaluation of a parsing system?
- How would you do intrinsic evaluation?
  - Gold standard data?
  - Metrics?

Gold-standard data

- There's no ground truth in trees
- Semantic dependencies might be easier to get cross-framework agreement on, but even there it's non-trivial
- The Penn Treebank (Marcus et al 1993) was originally conceived of as a target for cross-framework parser evaluation
- For project-internal/regression testing, grammar-based treebanking is effective for creating (g)old-standard data

Parseval measures

- Labeled precision:

$$\frac{\quad}{-}$$

- Labeled recall:

$$\frac{\text{\# of correct constituents in candidate parse}}{\text{total \# of constituents in gold standard parse}}$$

  - Constituents defined by starting point, ending point, and non-terminal symbol of spanning node

- Cross brackets: average number of constituents where the phrase boundaries of the gold standard and the candidate parse overlap

  - Example overlap: ((A B) C) v. (A (B C))