Ling/CSE 472: Introduction to Computational Linguistics

5/16/17

Statistical Parsing

Overview

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

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

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

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 and daughter edge probabilities

Probabilistic 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]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of

Work through an example: Kim adores snow in Oslo

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

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

Overview

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

Reading questions

- Obviously, PCFG is a probabilistic version of CFG; have the same ideas been applied to CFG with feature structures, or dependency grammars?

- I am wondering what other approaches can be taken if probabilistic parsing still cannot clear the ambiguity. Do we just randomly pick one of the most probable trees or we can narrow it down by other method?

- Would probabilistic lexical CFGs deal with things like polarity items, assuming that polarity items are important enough to be included in models of syntactic structure/parsing?

- Why does split and merge non-terminals help to solve the poor independence assumptions?

Reading questions

- "The intuition for solving this chicken-and-egg problem [finding the probabilities] is to incrementally improve our estimates by beginning with a parser with equal rule probabilities, then parse the sentence, compute a probability for each parse, use these probabilities to weight the counts, re-estimate the rule probabilities, and so on, until our probabilities converge. Inside-outside The standard algorithm for computing this solution is called the inside-outside algorithm"

- How does this this not just run away with whatever initial conditions it has? Seems like a positive feedback loop, where the more likely a parse is, the more likely it will be parsed that way, and therefore the higher the probability it will be assigned, so parser is more likely to pick that parse, ad infinitum.

Reading questions

- For on-line sentence-processing experiments, how would we evaluate or interpret the unknown input?

- How could a (non-human) parser provide the correct parse for a garden-path sentence? Is there a guarantee it would produce the correct one after realizing there are terminal nodes that don't fit into the suggested parse so far + backtracking (like in 14.39.a. and 14.41.a)?

- How relevant is human parsing for designing machine parsers, and how much do machine parses teach us about human parsing? Garden path sentences seem like they wouldn't bother machine parsers since it will consider all of the possible parses anyways, and similarly there are sentences that are easy to parse for people but that would seem improbable to a machine parser.

Reading questions

- Can syntactic parsing somehow use features and forms to evaluate syntactic probability for certain words or ambiguous parsings?

- For example, inanimate or abstract things are less likely to be followed by (most) active verbs (besides 'to be').

- ex.'The answer was ___'

- Because of the type of noun, 'was' is more likely to be an auxiliary verb/main verb followed by a +passive verb or an adjective, than a verb that is a gerund.

- 'The dog was ___' would yield different probabilities as a gerund is more likely

Reading questions

- The book mentioned how NP -> DT NN & NP -> PRP have very similar probabilities, and also talked about how the independence assumption can lead to poor estimates.

    - How much of a difference would it make if the 91:9 Pronoun vs. Non-pronoun ratio could be represented (i.e. if contextual differences could accurately be taken into account)?

- Why did 4-gram and 5-gram models beat out the PCFGs when it seemed like the characteristics of a CFG would hold much more information? Are there benefits to using PCFGs over 4/5-grams besides training data scarcity? (I'm thinking storage space or speed). Are there methods through which we can combine PCFGs and N-grams like finding creating a probability for a tree of non-terminals and then using the probability of n-grams in order to fill in its terminal values?

Reading questions

- In the description of discriminative reranking (in the transition between pages 481-482): what goes into "choos[ing] the single best parse from the N-best list"? Is this done by another kind of probabilistic approach? And what values of N would be typical for this sort of thing?

- With the evidence that human parsing seems to utilize both syntactical and lexical probabilities, is there a way in which our two types of models (syntactic rules and lexical rules) can be combined?