Probabilistic Context-Free Grammars

Linguistics 420
Statistical NLP
Spring 2006

Overview

- Properties of PCFGs
- Advantages/Limitations of PCFGs
- The Three Questions for PCFGs
- Probability estimation: Inside and Outside probabilities
- Best Path Estimation
- PCFG Training: the Inside-Outside algorithm

Properties of CFGs

Informally:

- A context-free grammar (CFG) is a way to capture syntactic regularities in languages (including recursive embedding)
- Go beyond FSMs/HMMs—can encode longer-range dependencies

Formally, consist of:

- Set of terminals (words): $\{w_1, ..., w_V\}$
- Set of nonterminals (categories): $\{N_1, ..., N_n\}$
- Designated start symbol: N_1 (often S)
- Set of rules/productions: $\{N_i \rightarrow \zeta_j\}$ (ζ_j is a sequence of terminals and nonterminals)

Motivating Probability: A Simple CFG

Given the following grammar:

$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP\ PP$
$NP \rightarrow DET\ N$	$NP \rightarrow NP\ PP$
$VP \rightarrow V\ NP$	
$PP \rightarrow P\ NP$	

$N \rightarrow$ girl	$N \rightarrow$ boy	$N \rightarrow$ park	$N \rightarrow$ telescope
$V \rightarrow$ saw			
$P \rightarrow$ with	$P \rightarrow$ in	$DET \rightarrow$ the	

and given the sentence *The boy saw the girl in the park with the telescope* ...

$\Rightarrow$ we have 5 different analyses

Motivating Probability: Multiple Analyses

For this sentence, the main problem is in where to attach the different PPs

- And this is a very serious problem: ambiguities in different places multiply
- Easy to get thousands or millions of analyses for even simple-seeming sentences

Thus, we want a method to disambiguate the different analyses

Properties of PCFGs

Informally:

- A probabilistic context-free grammar (PCFG) captures syntactic regularities in languages probabilistically
- i.e., what tends to happen

Formally, consist of everything a CFG does and additionally:

- A set of probabilities for each rule, s.t.:

  (1) $\forall i \sum_{j} P(N_i \rightarrow \zeta_j) = 1$

- These probabilities are the probability of a sequence of daughters given a particular mother, i.e., $P(N_i \rightarrow \zeta_j) = P(N_i \rightarrow \zeta_j | N_i)$

There are other ways to parse probabilistically, but PCFGs are a great place to start.

Probability of a tree

- To get $P(t)$, we simply multiply the probabilities of all the subtrees
- The probability of a sentence (w_{1m}), then, is the sum of all parses for that sentence

  (2) $P(w_{1m}) = \sum_t P(t)$, where t yields w_{1m}

See **example 1** on p. 384, referring to figure 11.1 and table 11.2

Assumptions of PCFGs

The following properties are based on the idea that rules are independent of one another and thus that we can multiply their probabilities together to get a tree's probability

- Place invariance: Probability of a subtree does not depend on where in the string it appears
- Context-free: Probability of a subtree does not depend on words outside of the subtree
- Ancestor-free: Probability of a subtree does not depend on nodes outside of the subtree

Probabilistic grammar

S $\rightarrow$ NP VP : 0.8 S $\rightarrow$ NP VP PP : 0.2
NP $\rightarrow$ DET N : 0.5 NP $\rightarrow$ NP PP : 0.5
VP $\rightarrow$ V NP : 1.0
PP $\rightarrow$ P NP : 1.0

N $\rightarrow$ girl : 0.25 N $\rightarrow$ boy : 0.25 N $\rightarrow$ park : 0.25 N $\rightarrow$ telescope : 0.25
V $\rightarrow$ saw : 1.0
P $\rightarrow$ with : 0.5 P $\rightarrow$ in : 0.5 DET $\rightarrow$ the : 1.0

This allows us to sort out the different analyses from before (for the most part) for *The boy saw the girl in the park with the telescope*

Advantages of PCFGs

- PCFGs give some idea about what constitutes a plausible parse.
- PCFGs are good for **grammar induction**, learning a grammar from a text; CFGs require negative data in order to learn.
- PCFGs tend to be robust to disfluencies and grammatical mistakes because these simply receive low probabilities.

Limitations of PCFGs

- PCFGs do not take lexical information into account, making parse plausibility less than ideal and making PCFGs worse than n-grams as a language model.

- PCFGs have certain biases; i.e., the probability of a smaller tree is greater than a larger tree.

- When two different analyses use the same set of rules, they have the same probability.

Obtaining a PCFG

As with POS tagging, we can have supervised or unsupervised training; we will focus on supervised learning

- Given a treebank of annotated sentences, count the number of times each rule LHS $\rightarrow$ RHS occurs ($C(LHS \rightarrow RHS)$)
- Count the number of times each LHS occurs ($C(LHS)$)
- Divide: $\frac{C(LHS \rightarrow RHS)}{C(LHS)}$

And then you'll likely want to smooth

Questions for PCFGs

1. **Probability estimation**: What is the probability of a sentence w_{1m} according to a grammar G: $P(w_{1m}|G)$?

2. **Best tree estimation**: What is the most likely parse for a sentence: $\arg\max_t P(t|w_{1m}, G)$?

3. **Training**: How can we choose rule probabilities for the grammar G that maximize the probability of a sentence: $\arg\max_G P(t|w_{1m}, G)$?

   Very difficult problem since there are so many parameters; harder than for HMMs due to more local maxima.

Probability estimation

Trying to find $P(w_{1m}|G)$

The "easy" way (in principle) is to sum over all possible trees which could generate that sentence:

(3) $P(w_{1m}|G) = \sum_t P(w_{1m}, t|G)$

Problem: this sums over exponentially many trees

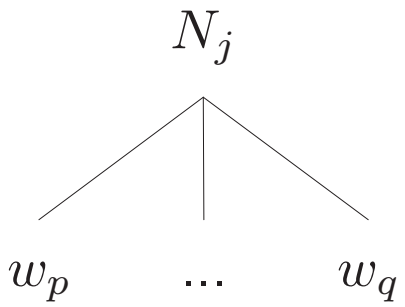
Solution: use an efficient algorithm based on inside and outside probabilities (see figure 11.3, page 391)

- Inside probability (backward): probability of generating words inside the non-terminal in focus ($\beta_j(p, q)$)
- Outside probability (forward): probability of generating the non-terminal in question from the start symbol and of generating words outside the non-terminal ($\alpha_j(p, q)$)

Inside probabilities

Inside probability ($\beta_j(p,q)$): probability that a non-terminal N_j expands to the words between p and q

(4) $\beta_j(p,q) = P(w_{pq}|N_j, G)$



The inside algorithm

The inside algorithm is a form of dynamic programming:

- When at a particular focus non-terminal node N_j, figure out the different ways to expand the node.
- But not all expansions are considered; only those that match with a previously calculated inside probability
- The base case is simply the inside probability of a single word's category.

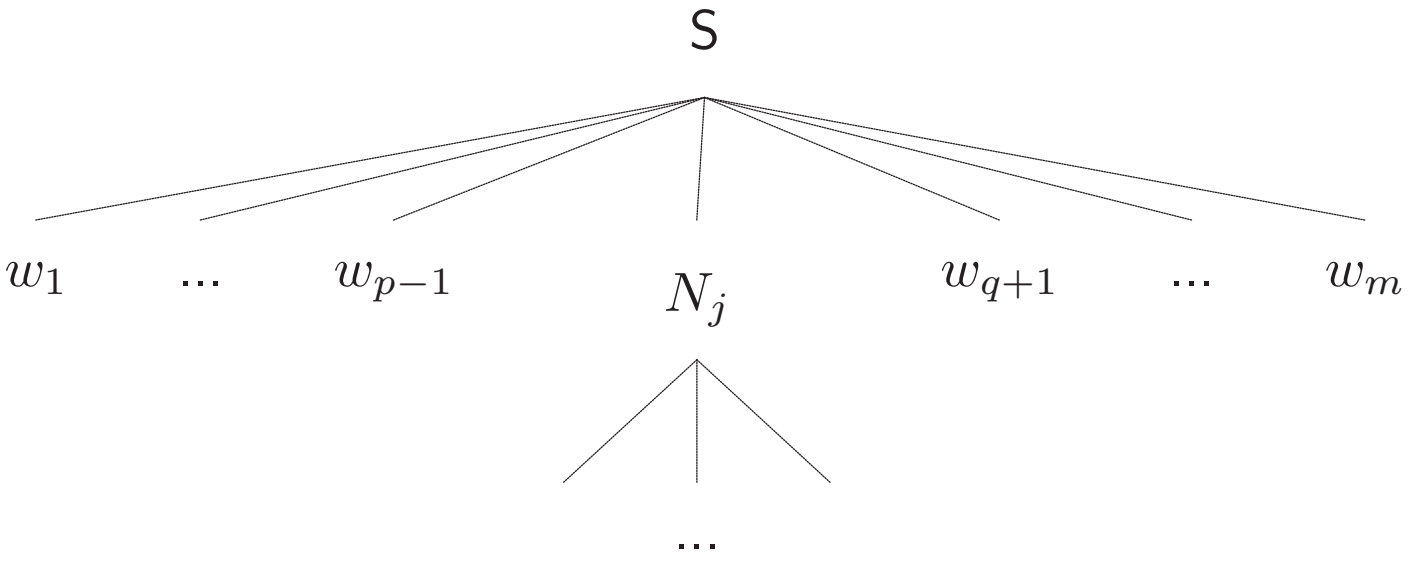
So, for example, in a sentence with a VP trying to cover positions 2 through 5 (*astronomers* [*saw stars with ears*]), we calculate:

(5) $\beta_{VP}(2,5) = P(\text{VP} \rightarrow \text{V NP})\beta_V(2,2)\beta_{NP}(3,5) + P(\text{VP} \rightarrow \text{VP PP})\beta_{VP}(2,3)\beta_{PP}(4,5)$

Outside probabilities

Outside probability ($\alpha_j(p,q)$): dual of the inside probability

(6) $\alpha_j(p,q) = P(w_{1(p-1)}, N_j, w_{(q+1)m}|G)$



The outside algorithm

The outside algorithm is slightly more complicated, in that it needs inside probabilities to get probabilities of the focus node's sisters. ... Use three different probabilities to get $\alpha_j(p, q)$:

- The outside probability of the node's mother
- The inside probability of the node's sister(s) (having to check different possibilties for how far it could expand)
- The probability of the rule MOM $\rightarrow$ N_j SIS (or MOM $\rightarrow$ SIS N_j)

The base case is simply $\alpha_S(1, m) = 1$ (i.e., we know our root node S will span the entire string)

The probability of a string

Putting these probabilities together, we can obtain the probability of a sentence:

(7) $P(w_{1m}|G) = \alpha_S(1,m)\beta_S(1,m)$

And in general the probability of some consituent N spanning from words p to q is:

(8) $P(w_{1m}, N|G) = \sum_j \alpha_j(p,q)\beta_j(p,q)$

Best Tree Estimation

A Viterbi-style algorithm is used to find the best tree, or most likely parse

- Use inside probabilities as our accumulators (sort of like we did with HMM tagging)
- Instead of summing over the different possibilities at a given node, take the non-terminal which maximizes the probability at that node

  Have to actually look at all the different subtrees which could make up a node and take the maximum one of those

PCFG Training

If we have a treebank, we can calculate the following directly:

(9) $P(N \rightarrow \zeta) = \frac{C(N \rightarrow \zeta)}{\sum_{\gamma} C(N \rightarrow \gamma)}$

If we don't, we do the following:

- Start with an initial guess for the parameters of the model (e.g., randomly assigned)
- Use inside and outside probabilities, in tandem with the data, to improve the guess
- Stop at a locally best model (guaranteed to improve; but not guaranteed to be the absolute best)
- Like the Forward-Backward algorithm, this is a variant of the Expectation-Maximization (EM) algorithm

The Inside-Outside algorithm

What we need is some way to generate expected counts for $N \rightarrow \gamma$ for all γ (including ζ), in order to get the probability $P(N \rightarrow \zeta)$ spanning from i to j

Need to know the following:

- Probability of generating outside words which would leave a hole for N
- Probability of generating inside words from i to k for first item in ζ
- Probability of generating inside words from k to j for other items in ζ

Use expected counts to generate new (inside/outside) probabilities ... which give new expected counts ... and so forth

Problems with the Inside-Outside algorithm

- Very slow compared to HMMs
- Very sensitive to the initial parameters, i.e., the local maximum you reach varies depending on where you started from
- Not guaranteed to end up with a parse that resembles anything linguistic