Probabilistic Parsing

Vitaliy Batusov

April 2, 2015

Outline

Probabilistic Context-Free Grammars

- N-gram and Hidden Markov models are linear
- Natural language syntax is not linear in structure
- Bayesian Networks is one way to capture structure
- PCFG, a derivative of CFG, is another way

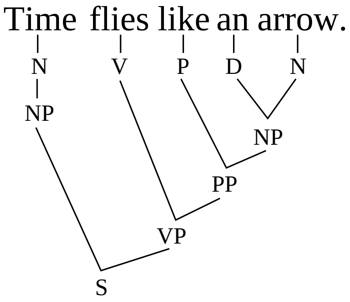
Probabilistic Context-Free Grammars

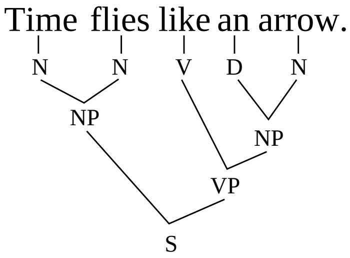
- Existing CFG parsers are very good

  Example: CYK, $O(n^3)$
- Can't apply to NL due to ambiguity

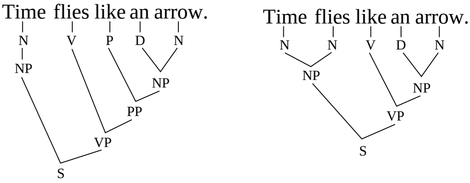
Probabilistic Context-Free Grammars

Consider a sentence with two parse trees:





Probabilistic Context-Free Grammars



These trees induce a CFG:

S	$\rightarrow$	NP VP	VP	$\rightarrow$	V NP	N	$\rightarrow$	time	V	$\rightarrow$	like
NP	$\rightarrow$	N	VP	$\rightarrow$	V PP	N	$\rightarrow$	arrow	V	$\rightarrow$	flies
NP	$\rightarrow$	N N	PP	$\rightarrow$	P NP	N	$\rightarrow$	flies	P	$\rightarrow$	like
NP	$\rightarrow$	D N				D	$\rightarrow$	an			

Probabilistic Context-Free Grammars

- If we parse the same sentence using this CFG, we will obtain at least two different trees

- Need a way to assign a score to each tree

PCFG as a Probabilistic Model

- Let's model derivations as stochastic processes

Consider the left-most derivation of the first tree:

$$\mathbf{S} \rightarrow \mathbf{NP}\,\mathbf{VP} \rightarrow \mathbf{N}\,\mathbf{VP} \rightarrow \textit{time}\,\mathbf{VP} \rightarrow \textit{time}\,\mathbf{V}\,\mathbf{PP}$$
$$\rightarrow \textit{time flies}\,\mathbf{PP} \rightarrow \textit{time flies}\,\mathbf{P}\,\mathbf{NP} \rightarrow \ldots$$

Each " $\rightarrow$ " involves a choice.

PCFG as a Probabilistic Model

- Assign a probability P to each rule $X \rightarrow \alpha$
  so that $\sum_{i=1}^{n} P(X \rightarrow \alpha_i) = 1$ for each unique non-terminal X

$$\mathbf{S} \xrightarrow{1.0} \mathbf{NP\ VP} \qquad \mathbf{V} \xrightarrow{0.3} \textit{like}$$

$$\mathbf{NP} \xrightarrow{0.4} \mathbf{N} \qquad \mathbf{V} \xrightarrow{0.7} \textit{flies}$$

$$\mathbf{NP} \xrightarrow{0.2} \mathbf{N\ N} \qquad \mathbf{P} \xrightarrow{1.0} \textit{like}$$

$$\mathbf{NP} \xrightarrow{0.4} \mathbf{D\ N} \qquad \mathbf{VP} \xrightarrow{0.5} \ldots$$

This constitutes a PCFG.

PCFG as a Probabilistic Model

- A **PCFG** is a CFG in which every production rule is associated with a probability.
- A PCFG is **proper** if its probability distribution is proper[1] over every subset of rules that have the same left-hand-side.
- A PCFG is **consistent** if its probability distribution is proper over the set of trees[2] it generates

[1] i.e., adds up to 1, see previous slide
[2] or sentences, doesn't matter

PCFG as a Probabilistic Model

- With rule probabilities, can calculate probability of a tree:

$$
\begin{aligned}
P(\mathit{TREE}1) &= P(\mathbf{N} \rightarrow \textit{time}) \times P(\mathbf{V} \rightarrow \textit{flies}) \times P(\mathbf{P} \rightarrow \textit{like}) \\
&\quad \times \ldots \times P(\mathbf{S} \rightarrow \mathbf{NP}\,\mathbf{VP}) \\
&= 0.0084
\end{aligned}
$$

$$P(\mathit{TREE}2) = 0.00036$$

- Now we have a winner

PCFG as a Probabilistic Model

But where do the probabilities come from?

- Recall the Big Assignment
- Given set of parse trees — *a treebank* — count occurrences of each rule application and normalize wrt respective non-terminal
- Every PCFG built using a treebank is proper

Computational Tasks

Evaluation: assessing the value of a given tree

⇒ multiply probabilities associated with each rule used in building the tree

(saw this already)

Computational Tasks

Generation: producing sentences

$\Rightarrow$ monkeys again. Start with **S**, select derivation rule randomly according to the probability distribution, repeat for each resulting non-terminal until none left

If PCFG is proper, procedure will halt

Computational Tasks

Learning: building a PCFG from a treebank

$\Rightarrow$ Count occurences of each rule for X, divide result by number of all rules for X

(saw this already)

Probabilistic Inference

Inference Tasks:

$P(sentence)$	(Marginalization)
$P(tree \mid sentence)$	(Conditioning)
$arg\ max_{tree} P(tree \mid sentence)$	(Completion)

Probabilistic Inference

Consider marginalization

$$P(sentence) = P(w_1 w_2 \ldots w_n \mid \mathbf{S}) = \sum_{tree \in T} P(tree)$$

- Need to find the set T of all trees for *sentence*
- Need to compute each tree's probability
- Likely to lead to an exponential algorithm

  For grammar $\{S \to S\ S,\ S \to a\}$, how many trees does a^n have?

Efficient Probabilistic Inference

- Can use a version of CYK, an excellent CFG parser
- Only works on CFG in Chomsky Normal Form

Efficient Probabilistic Inference

Chomsky Normal Form
A CFG is in CNF if its every derivation rule has one of the two forms:

$$\mathbf{A} \rightarrow \mathbf{B}\ \mathbf{C}$$
$$\mathbf{A} \rightarrow w$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are non-terminals and w is a terminal.

- Any CFG can be converted to CNF
- How to translate probabilities?

  estimate by sampling or calculate directly

Efficient Probabilistic Inference

Calculating CNF:

- Eliminate empty rules $\mathbf{X} \rightarrow \epsilon$

  strike out RHS appearances of nullable terminals in all ways except one
- Eliminate unit rules $\mathbf{X} \rightarrow \mathbf{Y}$

  if A derives B and $B \rightarrow \phi$, add $A \rightarrow \phi$
- Eliminate terminals except singletons

  introduce new non-terminals as necessary
- Break rules $\mathbf{X} \rightarrow \mathbf{Y}_1\ \mathbf{Y}_2 \ldots \mathbf{Y}_n$, $n > 2$

  introduce $n - 2$ new non-terminals for each rule, replace rule with a set of new rules

Efficient Probabilistic Inference

Our grammar in CNF:

S	$\rightarrow$	NP VP	VP	$\rightarrow$	V NP	N	$\rightarrow$	time	V	$\rightarrow$	like
NP	$\rightarrow$	time	VP	$\rightarrow$	V PP	N	$\rightarrow$	arrow	V	$\rightarrow$	flies
NP	$\rightarrow$	N N	PP	$\rightarrow$	P NP	N	$\rightarrow$	flies	P	$\rightarrow$	like

Efficient Probabilistic Inference

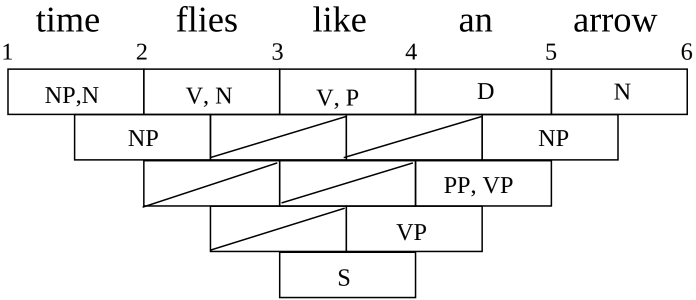
CYK — Cocke-Younger-Kasami Algorithm

Algorithm 1 CYK

Require: sentence $= w_1 \dots w_n$, and a CFG in CNF with nonterminals $N^1 \dots N^m$, N^1 is the start symbol
Ensure: parsed sentence

allocate matrix $\beta \in \{0,1\}^{n \times n \times m}$ and initialize all entries to 0
for $i \leftarrow 1$ to n **do**
    for all rules $N^k \rightarrow w_i$ **do**
        $\beta[i,1,k] \leftarrow 1$
for $j \leftarrow 2$ to n **do**
    for $i \leftarrow 1$ to $n-j+1$ **do**
        for $l \leftarrow 1$ to $j-1$ **do**
            for all rules $N^k \rightarrow N^{k_1} N^{k_2}$ **do**
                $\beta[i,j,k] \leftarrow \beta[i,j,k]$ OR $(\beta[i,l,k_1]$ AND $\beta[i+l,j-l,k_2])$
return $\beta[1,n,1]$

Efficient Probabilistic Inference



Basic idea: exploiting CNF, build a chierarchical chart of non-terminals

Efficient Probabilistic Inference

- Add probabilities to the mix

  NOT straight-forward, but has been done
- Obtain efficient marginalization for PCFGs

Applications

- Monkeys
- Everything that needs to resolve ambiguity of NL parsing
- Detection of grammatical errors in text