Problems with CFG parsing

Ambiguity is a serious problem in parsing since sentences have multiple parses. Common causes are:

- Attachment ambiguity. A constituent can be attached at different points in the parse tree.
  Example: `(Show N|V) (the Det) (menu N) (on Prep) (the Det) (Shatabdi PropN) (from Prep) (Kanpur PropN) (to Prep) (Delhi PropN).`
  $S \rightarrow VP \rightarrow V\ NP$
  $S \rightarrow V\ NP\ PP$
  NP and PP can be broken down in multiple ways.
- Coordination ambiguity. Conjuncts are ambiguous.
  `Young boys and girls will enjoy Techkriti.`
  Young (boys and girls)... or (Young boys) and girls...
- Coordination and attachment ambiguities can interact to produce large number of parses.

Probabilistic CFGs

- Standard CFL parsers return all parses. There is no way for a parser to decide which parse is more likely.
- One solution is to use Probabilistic CFGs where each rule has an associated probability. Consequently, each parse can be given a probability and the parse with the highest probability can be returned.

PCFGs

Definition 1 (PCFG)

A PCFG G is a 5-tuple (N, Σ, R, S, P) *where the first four elements of the tuple are the same as a CFG and* $P : R \rightarrow [0, 1]$ *is a mapping that assigns a value between* 0 *and* 1 *to each rule in* R*.*
A rule in a PCFG has the following structure:
$A \rightarrow \alpha_i, p_i$ *with* $\sum_{(A \rightarrow \alpha_i, p_i \in R)} p_i = 1$ *(that is probablities of all rules with non-terminal* A *add upto* 1*) - such a grammar is called a* <u>proper</u> *PCFG.*

Let $w = w_1, \ldots, w_n$ be the input sentence, let w_{ij} stand for the sequence $w_i, \ldots, w_j$, $A_{(ij)}$ denote that non-terminal A derives the string/sequence $w_i, \ldots, w_j$. The probability of the sentence $w = w_{1n}$ w.r.t. to grammar G is:

$$P(w) = P(w_{1n}) = \sum_t P(w_{1n}, t) = \sum_{(t|\text{yield}(t)=w_{1n})} P(t)$$

t is a parse tree of w using G and $P(t)$ is the product of probabilities of all rules used in the tree.

Assumptions

- Place invariance: probability of a sub-tree t generated by A does not depend on where in w, $A_{(ij)}$ occurs.
- Context freeness: probability of a sub-tree t does not depend on words or terminals not generated by t (leaves).
- Ancestor free: probability of tree t does not depend on other nodes that are ancestors of the non-terminal generating t.

Example

$$
\begin{aligned}
S &\rightarrow NP\ VP,\ 1.0 \\
PP &\rightarrow P\ NP,\ 1.0 \\
VP &\rightarrow V\ NP,\ 0.7 \mid VP\ PP,\ 0.3 \\
P &\rightarrow \text{with},\ 1.0 \\
V &\rightarrow \text{saw},\ 1.0
\end{aligned}
$$

$$
\begin{aligned}
NP &\rightarrow NP\ PP,\ 0.4 \\
&\rightarrow \text{astronomers},\ 0.1 \\
&\rightarrow \text{binoculars},\ 0.18 \\
&\rightarrow \text{saw},\ 0.04 \\
&\rightarrow \text{stars},\ 0.18 \\
&\rightarrow \text{telescopes},\ 0.18
\end{aligned}
$$

contd.

`Astronomers saw stars with binoculars.`

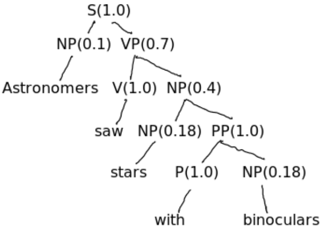

Figure: t_1

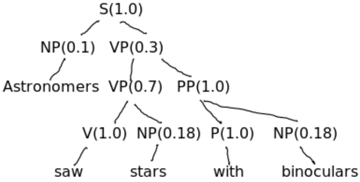

Figure: t_2

$P(t_1) = 1 \times 0.1 \times 0.7 \times 0.4 \times 0.18 \times 0.18 = 0.0009072$
$P(t_2) = 1 \times 0.1 \times 0.3 \times 0.7 \times 0.18 \times 0.18 = 0.0006804$
A PCFG is consistent if $\sum_{t \in \text{ All parses of } w_{1n}} = 1$

Consistency is not guaranteed by properness.

Features of PCFG

- PCFGs give some idea of the plausibility of parses. But this is based on purely structural constraints and lexical influences are not present.
- PCFGs can be used for grammar induction from bracketed labelled corpora.
- PCFGs are robust and can handle grammatical errors/disfluencies by giving lower probabilities to such sentences.

Limitations of PCFGs

- Do not always give information on plausibility of parse.
- The implied independence assumptions do not hold. For example, in the Switchboard corpus:

	Pronoun	Not Pronoun
Subject	0.91	0.09
Object	0.34	0.66

  While normal rule probability calculations give:

  $NP \rightarrow Det\ N$, 0.28; $\quad NP \rightarrow Pro$, 0.25

- Insensitive to lexical dependencies.

  Example:

  `[workers [dumped sacks [into a bin]]]`, $VP \rightarrow VBD\ NP\ PP$

  `[workers [dumped [sacks [into [a bin]]]]]`,

  $VP \rightarrow VBD\ PP$, $PP \rightarrow P\ NP$

  Contrast with: `fishermen caught tons of fish`.

- Tend to favour shallow trees.

Inducing a PCFG from a training corpus

- The training corpus $\mathcal{L}$ is a set of parse trees[1] $t_1, \ldots, t_m$. For example the Penn treebank.
- S is the common start symbol for all the trees in $\mathcal{L}$. The set N of non-terminals is the set of all non-terminals in the trees in $\mathcal{L}$. Σ is the set of all terminals in the set of trees in $\mathcal{L}$. R is the set of all rules in all the trees in $\mathcal{L}$.
- The probability estimate for a rule $A \rightarrow \alpha$ is the ML estimate given by $P(A \rightarrow \alpha) = \frac{\text{count}(A \rightarrow \alpha)}{\text{count}(A)}$, where counting is over all trees in $\mathcal{L}$.
- The training set $\mathcal{L}$ is normally contructed manually or using a standard CFG parser with manual curation of the parses output by the parser.
- Sequence to sequence methods are not as successful in constituent parse generation compared to their success in tagging problems.

[1] Derivations are assumed to be left-most.

Finding the max probability parse

Problem statement and Notation:
Let $T(w)$ be the set of all possible left-most parses for w. For $t \in T(w)$, $P(t)$ is the probability of the parse tree t.

The problem is to find $t_{max} = \underset{t \in T(w)}{\operatorname{argmax}} P(t)$.

Let G be a PCFG in CNF, $w = w_{1n}$ the input string to be parsed, w_{ij} a substring of w,
$\pi[i, j, A]$: table entry containing the NT A that has the max probability for substring w_{ij}.

A bottom-up recursive method (based on CYK) is able to find t_{max}. Now table entries contain both the non-terminals the can generate w_{ij} but also the probability of the corresponding sub-tree. Only the max probability is retained.

The table entries are filled by induction.
Choose max among all possibilities:

- Base case: $\pi(i, i, A) = P(A \rightarrow w_i)$ or 0 if $A \rightarrow w_i \notin R$.
- Recursive case: Strings of length > 1. $A \overset{\star}{\Rightarrow} w_{ij}$ iff $\exists$ a rule $A \rightarrow BC$ such that $B \overset{\star}{\Rightarrow} w_{ik}$ and $C \overset{\star}{\Rightarrow} w_{k+1\,j}$.
- Compute the probability by multiplying the probabilities of these two parts (they have been already calculated in the recursion).
  $$\pi[i, j, A] = \max_{\substack{A \rightarrow BC \in R \\ k \in \{i \ldots (j-1)\}}} (P(A \rightarrow BC) \times \pi[i, k, B] \times \pi[k+1, j, C])$$

The CYK PCFG algorithm[2]

Algorithm 0.1: $\mathrm{CYK}(w = w_{1n}, G)$

$$
\begin{array}{l}
\textbf{for each } (i \in 1 \ldots n), \textbf{for each } (A \in N)\\
\quad \textbf{do } \begin{cases} \textbf{if } (A \rightarrow w_i \in R) \\ \quad \textbf{then } \pi[i, i, A] = P(A \rightarrow w_i) \\ \quad \textbf{else } \pi[i, i, A] \leftarrow 0 \end{cases}\\
\textbf{for } l = 1 \ldots (n-1)\\
\quad \textbf{do for } i = 1 \ldots (n-l)\\
\quad \textbf{do } \begin{cases} j \leftarrow i + 1 \\ \textbf{for each } (A \in N) \\ \quad \textbf{do } \begin{cases} \pi[i, j, A] = \max\limits_{\substack{A \rightarrow BC \\ k \in i \ldots j-1}} (P(A \rightarrow BC) \times \pi[i, k, B] \times \pi[k+1, j, C]) \\ bkptr[i, j, A] = \operatorname*{argmax}\limits_{\substack{A \rightarrow BC \\ k \in i \ldots j-1}} (P(A \rightarrow BC) \times \pi[i, k, B] \times \pi[k+1, j, C]) \end{cases} \end{cases}\\
\textbf{return } (\pi[1, n, S], bkptr)
\end{array}
$$

[2] From Collins

Handling violation of indp. assumptions

- All violations have to be handled structurally.
- Split NTs. For example, instead of NP have NP_{sub} and NP_{obj} then $NP_{sub} \rightarrow Pro$ and $NP_{obj} \rightarrow Pro$ can have different probabilities. This will lead to a change in rules: $S \rightarrow NP_{sub}\, VP$, $VP \rightarrow V\, NP_{obj}$.
- This is a general approach to indp. problems. For example, adverbs also occur in different positions more commonly. RB_{adv} - also, now; RB_{VP} - n't, not; RB_{NP}- only, just.

Lexical conditioning

- Dependence on context words can be handled by lexicalized rules. Example: $VP[dumped] \rightarrow VBD[dumped]\ NP[sacks]\ PP[into]$. Often PoS tags are included.
- This leads to sparsity problems while estimating probabilities. Usually solved by using multiple smaller rules to generalize one complex rule which most often will not suffer from sparsity.