Probabilistic Grammars

Martin Emms

August 17, 2007
Basic Definitions

Baker’s Unsupervised PCFG training algorithm

Variant applications of Inside-Outside algorithm

Other models and supervised learning
Probabilistic CFGs

Rule Probabilites

assume a function $A$ assigns each rule $M \rightarrow rhs$ a probability.
Probabilistic CFGs

Rule Probabilities

assume a function $A$ assigns each rule $M \rightarrow rhs$ a probability

probs for rules expanding $M$ must sum to 1

$$\sum_{rhs} A[M \rightarrow rhs] = 1$$
Probabilistic CFGs

### Rule Probabilities

Assume a function $A$ assigns each rule $M \rightarrow \text{rhs}$ a probability.

Probs for rules expanding $M$ must sum to 1:

$$\sum_{\text{rhs}} A[M \rightarrow \text{rhs}] = 1$$

So:

$$A[M \rightarrow D_1 \ldots D_n] = \text{the prob of dtrs } D_1 \ldots D_n \text{ given a mother } M$$

$$= P(D_1 \ldots D_n | M)$$
### Probabilistic CFGs

#### Rule Probabilities

Assume a function $A$ assigns each rule $M \rightarrow rhs$ a probability

Probs for rules expanding $M$ must sum to 1

$$\sum_{rhs} A[M \rightarrow rhs] = 1$$

So $A[M \rightarrow D_1 \ldots D_n] = \text{the prob of dtrs } D_1 \ldots D_n \text{ given a mother } M$

$$= P(D_1 \ldots D_n | M)$$

#### Root Probability

Also assume a function $\pi$ assign an initial probability to each category, quantifying its likelihood of being the root of a tree
Probabilistic CFGs

Rule Probabilities

assume a function $A$ assigns each rule $M \rightarrow \text{rhs}$ a probability

probs for rules expanding $M$ must sum to 1

so $A[M \rightarrow D_1 \ldots D_n] = \text{the prob of dtrs } D_1 \ldots D_n \text{ given a mother } M = P(D_1 \ldots D_n|M)$

Root Probability

also assume a function $\pi$ assign an initial probability to each category, quantifying its likelihood of being the root of a tree

Tree Probability

finally assume prob of a tree $T$, with root $T.\text{root}$

$$P(T) = \pi(T.\text{root}) \times \prod_{r \in T} A[r]$$
An example: one parse

\[ \pi \]
\[ S \quad 1.0 \]

\[ A \]
\[ S \to NP \ VP \quad 0.8 \]
\[ S \to VP \quad 0.2 \]
\[ NP \to \text{noun} \quad 0.4 \]
\[ NP \to \text{noun PP} \quad 0.4 \]
\[ NP \to \text{noun NP} \quad 0.2 \]
\[ VP \to \text{verb} \quad 0.3 \]
\[ VP \to \text{verb NP} \quad 0.3 \]
\[ VP \to \text{verb PP} \quad 0.2 \]
\[ VP \to \text{verb NP PP} \quad 0.2 \]
\[ PP \to \text{prep NP} \quad 1.0 \]
\[ \text{prep} \to \text{like} \quad 1.0 \]
\[ \text{verb} \to \text{swat} \quad 0.2 \]
\[ \text{verb} \to \text{flies} \quad 0.4 \]
\[ \text{verb} \to \text{like} \quad 0.4 \]
\[ \text{noun} \to \text{swat} \quad 0.05 \]
\[ \text{noun} \to \text{flies} \quad 0.45 \]
\[ \text{noun} \to \text{ants} \quad 0.5 \]

\[ \text{tree}_1 \text{ for } \text{swat} \text{ flies like ants} \text{ is } \]
\[ \begin{array}{c}
\text{S} \\
\mid \\
\text{VP} \\
\mid \\
\text{verb} \mid \text{NP} \mid \text{PP} \\
\mid \\
\text{swat} \mid \text{noun} \mid \text{prep} \mid \text{NP} \\
\mid \\
\text{flies} \mid \text{like} \mid \text{noun} \mid \text{ants} \\
\end{array} \]

\[ P(\text{tree}_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \]
\[ \times 1.0 \times 1.0 \times 0.4 \times 0.5 \]
\[ = 2.88 \times 10^{-4} \]
An example: one parse

\[ \pi \]

\[ S \quad 1.0 \]

\[ A \]

\[ S \rightarrow NP \ VP \quad 0.8 \]
\[ S \rightarrow VP \quad 0.2 \]
\[ NP \rightarrow noun \quad 0.4 \]
\[ NP \rightarrow noun \ PP \quad 0.4 \]
\[ NP \rightarrow noun \ NP \quad 0.2 \]
\[ VP \rightarrow verb \quad 0.3 \]
\[ VP \rightarrow verb \ NP \quad 0.3 \]
\[ VP \rightarrow verb \ PP \quad 0.2 \]
\[ VP \rightarrow verb \ NP \ PP \quad 0.2 \]
\[ PP \rightarrow prep \ NP \quad 1.0 \]
\[ prep \rightarrow like \quad 1.0 \]
\[ verb \rightarrow swat \quad 0.2 \]
\[ verb \rightarrow flies \quad 0.4 \]
\[ verb \rightarrow like \quad 0.4 \]
\[ noun \rightarrow swat \quad 0.05 \]
\[ noun \rightarrow flies \quad 0.45 \]
\[ noun \rightarrow ants \quad 0.5 \]

\[ tree_1 \] for \ styx \ flies \ like \ ants \ is

\[ S \]
\[ \quad | \]
\[ VP \]
\[ \quad | \]
\[ \quad | \]
\[ verb \]
\[ \quad | \]
\[ NP \]
\[ \quad | \]
\[ PP \]
\[ \quad | \]
\[ noun \]
\[ \quad | \]
\[ prep \]
\[ \quad | \]
\[ NP \]
\[ \quad | \]
\[ ants \]

\[ P(tree_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \]
\[ \times 1.0 \times 1.0 \times 0.4 \times 0.5 \]
\[ = 2.88 \times 10^{-4} \]
An example: one parse

\[ \pi \]

\begin{align*}
S & \rightarrow \text{NP VP} \quad 0.8 \\
S & \rightarrow \text{VP} \quad 0.2 \\
\text{NP} & \rightarrow \text{noun} \quad 0.4 \\
\text{NP} & \rightarrow \text{noun PP} \quad 0.4 \\
\text{NP} & \rightarrow \text{noun NP} \quad 0.2 \\
\text{VP} & \rightarrow \text{verb} \quad 0.3 \\
\text{VP} & \rightarrow \text{verb NP} \quad 0.3 \\
\text{VP} & \rightarrow \text{verb PP} \quad 0.2 \\
\text{VP} & \rightarrow \text{verb NP PP} \quad 0.2 \\
\text{PP} & \rightarrow \text{prep NP} \quad 1.0 \\
\text{prep} & \rightarrow \text{like} \quad 1.0 \\
\text{verb} & \rightarrow \text{swat} \quad 0.2 \\
\text{verb} & \rightarrow \text{flies} \quad 0.4 \\
\text{verb} & \rightarrow \text{like} \quad 0.4 \\
\text{noun} & \rightarrow \text{swat} \quad 0.05 \\
\text{noun} & \rightarrow \text{flies} \quad 0.45 \\
\text{noun} & \rightarrow \text{ants} \quad 0.5
\end{align*}

\[ \text{tree}_1 \] for \text{swat flies like ants} is

\begin{itemize}
\item \text{S}
\item \hspace{1cm} \text{VP}
\item \hspace{2cm} \text{verb NP PP}
\item \hspace{3cm} \text{swat noun prep NP}
\item \hspace{4cm} \text{flies like noun ants}
\end{itemize}

\[
P(\text{tree}_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \\
\times 1.0 \times 1.0 \times 0.4 \times 0.5 \\
= 2.88 \times 10^{-4}
\]
An example: one parse

\[ \pi \]

\[ S \rightarrow NP \ VP \quad 0.8 \]
\[ S \rightarrow VP \quad 0.2 \]
\[ NP \rightarrow noun \quad 0.4 \]
\[ NP \rightarrow noun \ PP \quad 0.4 \]
\[ NP \rightarrow noun \ NP \quad 0.2 \]
\[ VP \rightarrow verb \quad 0.3 \]
\[ VP \rightarrow verb \ NP \quad 0.3 \]
\[ VP \rightarrow verb \ PP \quad 0.2 \]
\[ VP \rightarrow verb \ NP \ PP \quad 0.2 \]
\[ PP \rightarrow prep \ NP \quad 1.0 \]
\[ prep \rightarrow like \quad 1.0 \]
\[ verb \rightarrow swat \quad 0.2 \]
\[ verb \rightarrow flies \quad 0.4 \]
\[ verb \rightarrow like \quad 0.4 \]
\[ noun \rightarrow swat \quad 0.05 \]
\[ noun \rightarrow flies \quad 0.45 \]
\[ noun \rightarrow ants \quad 0.5 \]

\[ tree_1 \] for \swat \ flies \ like \ ants\] is

\[
\begin{array}{c}
S \\
| \\
VP \\
\mid \\
\mid \\
\mid \\
\mid \\
\mid \\
verb \\
NP \\
\mid \\
NP \\
\mid \\
\mid \\
\mid \\
PP \\
flies \\
like \\
noun \\
\mid \\
\mid \\
\mid \\
\mid \\
ants
\end{array}
\]

\[
P(tree_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \\
\times 1.0 \times 1.0 \times 0.4 \times 0.5 \\
= 2.88 \times 10^{-4}
\]
An example: one parse

\[ \pi \]

\[
\begin{align*}
S & \rightarrow NP \ VP & 0.8 \\
S & \rightarrow VP & 0.2 \\
NP & \rightarrow \text{noun} & 0.4 \\
NP & \rightarrow \text{noun PP} & 0.4 \\
NP & \rightarrow \text{noun NP} & 0.2 \\
VP & \rightarrow \text{verb} & 0.3 \\
VP & \rightarrow \text{verb NP} & 0.3 \\
VP & \rightarrow \text{verb PP} & 0.2 \\
VP & \rightarrow \text{verb NP PP} & 0.2 \\
PP & \rightarrow \text{prep \ NP} & 1.0 \\
\text{prep} & \rightarrow \text{like} & 1.0 \\
\text{verb} & \rightarrow \text{swat} & 0.2 \\
\text{verb} & \rightarrow \text{flies} & 0.4 \\
\text{verb} & \rightarrow \text{like} & 0.4 \\
\text{noun} & \rightarrow \text{swat} & 0.05 \\
\text{noun} & \rightarrow \text{flies} & 0.45 \\
\text{noun} & \rightarrow \text{ants} & 0.5 \\
\end{align*}
\]

tree\textsubscript{1} for \textit{swat flies like ants} is

\[
\begin{align*}
P(tree\textsubscript{1}) & = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \\
& \times 1.0 \times 1.0 \times 0.4 \times 0.5 \\
& = 2.88 \times 10^{-4}
\end{align*}
\]
An example: one parse

\[ \pi \]

\[ S \rightarrow NP \ VP \quad 0.8 \]
\[ S \rightarrow VP \quad 0.2 \]
\[ NP \rightarrow \text{noun} \quad 0.4 \]
\[ NP \rightarrow \text{noun PP} \quad 0.4 \]
\[ NP \rightarrow \text{noun NP} \quad 0.2 \]
\[ VP \rightarrow \text{verb} \quad 0.3 \]
\[ VP \rightarrow \text{verb NP} \quad 0.3 \]
\[ VP \rightarrow \text{verb PP} \quad 0.2 \]
\[ VP \rightarrow \text{verb NP PP} \quad 0.2 \]
\[ PP \rightarrow \text{prep NP} \quad 1.0 \]
\[ \text{prep} \rightarrow \text{like} \quad 1.0 \]
\[ \text{verb} \rightarrow \text{swat} \quad 0.2 \]
\[ \text{verb} \rightarrow \text{flies} \quad 0.4 \]
\[ \text{verb} \rightarrow \text{like} \quad 0.4 \]
\[ \text{noun} \rightarrow \text{swat} \quad 0.05 \]
\[ \text{noun} \rightarrow \text{flies} \quad 0.45 \]
\[ \text{noun} \rightarrow \text{ants} \quad 0.5 \]

\[ \text{tree}_1 \text{ for } \text{swat flies like ants is} \]

\[ \begin{align*}
\text{S} \\
\quad \text{VP} \\
\quad \quad \text{verb} \quad \text{NP} \quad \text{PP} \\
\quad \quad \quad \text{swat} \quad \text{noun} \quad \text{prep} \quad \text{NP} \\
\quad \quad \quad \quad \text{flies} \quad \text{like} \quad \text{noun} \\
\quad \quad \quad \quad \quad \text{ants}
\end{align*} \]

\[ P(\text{tree}_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \]
\[ \times 1.0 \times 1.0 \times 0.4 \times 0.5 \]

\[ = 2.88 \times 10^{-4} \]
An example: one parse

$$\pi$$

$\pi$

A

$$S \rightarrow NP \ VP \ 0.8$$
$$S \rightarrow VP \ 0.2$$
$$NP \rightarrow noun \ 0.4$$
$$NP \rightarrow noun \ PP \ 0.4$$
$$NP \rightarrow noun \ NP \ 0.2$$
$$VP \rightarrow verb \ 0.3$$
$$VP \rightarrow verb \ NP \ 0.3$$
$$VP \rightarrow verb \ PP \ 0.2$$
$$VP \rightarrow verb \ NP \ PP \ 0.2$$
$$PP \rightarrow prep \ NP \ 1.0$$

prep $\rightarrow like \ 1.0$
verb $\rightarrow swat \ 0.2$
verb $\rightarrow flies \ 0.4$
verb $\rightarrow like \ 0.4$
noun $\rightarrow swat \ 0.05$
noun $\rightarrow flies \ 0.45$
noun $\rightarrow ants \ 0.5$

tree$_1$ for $swat$ $flies$ $like$ $ants$ is

$$P(tree_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45$$
$$\times 1.0 \times 1.0 \times 0.4 \times 0.5$$
$$= 2.88 \times 10^{-4}$$
An example: one parse

\[ \pi \]
\[ S \quad 1.0 \]
\[ A \]
\[ S \rightarrow NP \ VP \quad 0.8 \]
\[ S \rightarrow VP \quad 0.2 \]
\[ NP \rightarrow noun \quad 0.4 \]
\[ NP \rightarrow noun \ PP \quad 0.4 \]
\[ NP \rightarrow noun \ NP \quad 0.2 \]
\[ VP \rightarrow verb \quad 0.3 \]
\[ VP \rightarrow verb \ NP \quad 0.3 \]
\[ VP \rightarrow verb \ PP \quad 0.2 \]
\[ VP \rightarrow verb \ NP \ PP \quad 0.2 \]
\[ PP \rightarrow prep \ NP \quad 1.0 \]
\[ prep \rightarrow like \quad 1.0 \]
\[ verb \rightarrow swat \quad 0.2 \]
\[ verb \rightarrow flies \quad 0.4 \]
\[ verb \rightarrow like \quad 0.4 \]
\[ noun \rightarrow swat \quad 0.05 \]
\[ noun \rightarrow flies \quad 0.45 \]
\[ noun \rightarrow ants \quad 0.5 \]

\[
tree_1 \text{ for } swat \text{ flies like ants is}
\]
\[
\begin{array}{c}
P \left( tree_1 \right) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \\
\quad \times 1.0 \times 1.0 \times 0.4 \times 0.5 \\
\quad = 2.88 \times 10^{-4}
\end{array}
\]
An example: one parse

\[ \pi \]

\[ S \rightarrow NP \ VP \ 0.8 \]
\[ S \rightarrow VP \ 0.2 \]
\[ NP \rightarrow noun \ 0.4 \]
\[ NP \rightarrow noun \ PP \ 0.4 \]
\[ NP \rightarrow noun \ NP \ 0.2 \]
\[ VP \rightarrow verb \ 0.3 \]
\[ VP \rightarrow verb \ NP \ 0.3 \]
\[ VP \rightarrow verb \ PP \ 0.2 \]
\[ VP \rightarrow verb \ NP \ PP \ 0.2 \]
\[ PP \rightarrow prep \ NP \ 1.0 \]
\[ prep \rightarrow like \ 1.0 \]
\[ verb \rightarrow swat \ 0.2 \]
\[ verb \rightarrow flies \ 0.4 \]
\[ verb \rightarrow like \ 0.4 \]
\[ noun \rightarrow swat \ 0.05 \]
\[ noun \rightarrow flies \ 0.45 \]
\[ noun \rightarrow ants \ 0.5 \]

\[ tree_1 \ for \ swat \ flies \ like \ ants \ is \]

\[
\begin{align*}
  & S \\
  & \quad \mid \\
  & \quad \vert \\
  & VP \\
  & \quad \mid \\
  & \quad \vert \\
  & \quad \vert \\
  & verb \quad NP \quad PP \\
  & \quad \mid \\
  & \quad \vert \\
  & \quad \vert \\
  & swat \quad noun \quad prep \quad NP \\
  & \quad \mid \\
  & \quad \vert \\
  & \quad \vert \\
  & \quad \vert \\
  & flies \quad like \quad noun \quad ants \\
\end{align*}
\]

\[
P(tree_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \\
\quad \times 1.0 \times 1.0 \times 0.4 \times 0.5 \\
= 2.88 \times 10^{-4}
\]
An example: one parse

$$\pi$$

$$S \quad 1.0$$

$$A$$

$$S \rightarrow NP \ VP \quad 0.8$$
$$S \rightarrow VP \quad 0.2$$
$$NP \rightarrow \text{noun} \quad 0.4$$
$$NP \rightarrow \text{noun PP} \quad 0.4$$
$$NP \rightarrow \text{noun NP} \quad 0.2$$
$$VP \rightarrow \text{verb} \quad 0.3$$
$$VP \rightarrow \text{verb NP} \quad 0.3$$
$$VP \rightarrow \text{verb PP} \quad 0.2$$
$$VP \rightarrow \text{verb NP PP} \quad 0.2$$
$$PP \rightarrow \text{prep NP} \quad 1.0$$
$$\text{prep} \rightarrow \text{like} \quad 1.0$$
$$\text{verb} \rightarrow \text{swat} \quad 0.2$$
$$\text{verb} \rightarrow \text{flies} \quad 0.4$$
$$\text{verb} \rightarrow \text{like} \quad 0.4$$
$$\text{noun} \rightarrow \text{swat} \quad 0.05$$
$$\text{noun} \rightarrow \text{flies} \quad 0.45$$
$$\text{noun} \rightarrow \text{ants} \quad 0.5$$

The parse tree for "swat flies like ants" is

$$P(\text{tree}_1) = 1.0 \times 0.2 \times 0.2 \times 0.2 \times 0.4 \times 0.45 \times 1.0 \times 1.0 \times 0.4 \times 0.5 = 2.88 \times 10^{-4}$$
Example: another parse

\[ \pi \]

\[ S \rightarrow NP \ VP \ 0.8 \]
\[ S \rightarrow VP \ 0.2 \]
\[ NP \rightarrow noun \ 0.4 \]
\[ NP \rightarrow noun \ PP \ 0.4 \]
\[ NP \rightarrow noun \ NP \ 0.2 \]
\[ VP \rightarrow verb \ 0.3 \]
\[ VP \rightarrow verb \ NP \ 0.3 \]
\[ VP \rightarrow verb \ PP \ 0.2 \]
\[ VP \rightarrow verb \ NP \ PP \ 0.2 \]
\[ PP \rightarrow prep \ NP \ 1.0 \]
\[ prep \rightarrow like \ 1.0 \]
\[ verb \rightarrow swat \ 0.2 \]
\[ verb \rightarrow flies \ 0.4 \]
\[ verb \rightarrow like \ 0.4 \]
\[ noun \rightarrow swat \ 0.05 \]
\[ noun \rightarrow flies \ 0.45 \]
\[ noun \rightarrow ants \ 0.5 \]

\[ P(tree_2) = 0.8 \times 0.2 \times 0.05 \times 0.4 \times 0.45 \]
\[ \times 0.3 \times 0.4 \times 0.4 \times 0.5 \]
\[ = 3.46 \times 10^{-5} \]

so \[ P(tree_1) > P(tree_2) \]
Independence Assumptions

- treating tree probability with

\[ P(T) = \pi(T.root) \times \prod_{r \in T} A[r] \]

represents independence assumptions
Independence Assumptions

- treating tree probability with

\[ P(T) = \pi(T.\text{root}) \times \prod_{r \in T} A[r] \]

represents independence assumptions

- in particular looking at a local tree \( M \rightarrow \text{rhs} \), probability of dtrs \( \text{rhs} \) is independent of everything above, left and right of the mother \( M \)
Independence Assumptions

- treating tree probability with

\[ P(T) = \pi(T.\text{root}) \times \prod_{r \in T} A[r] \]

represents independence assumptions

- in particular looking at a local tree \( M \rightarrow \text{rhs} \), probability of dtrs \( \text{rhs} \) is independent of everything above, left and right of the mother \( M \) dtrs

independent of \( M \)'s siblings

\[ P(\text{dtrs}|M) = P(\text{dtrs}|M \text{ and siblings of } M) \]
Independence Assumptions

- treating tree probability with

\[ P(T) = \pi(T.\text{root}) \times \prod_{r \in T} A[r] \]

represents independence assumptions

- in particular looking at a local tree \( M \rightarrow \text{rhs} \), probability of dtrs \( \text{rhs} \) is independent of everything above, left and right of the mother \( M \)

\[ \text{independent of } M\text{'s siblings} \]

\[ P(\text{dtrs}|M) = P(\text{dtrs}|M \text{ and siblings of } M) \]

\[ \text{dtrs independent of } M\text{'s ancestors} \]

\[ P(\text{dtrs}|M) = P(\text{dtrs}|M \text{ and ancestors of } M) \]
Parsing: most probable parse

PCFG assigns a probability to a complete tree, words included. The parsing task is then finding the most probable tree, given the words

\[
\text{parse}(O) = \arg \max_T P(T|O)
\]

\[
= \arg \max_{T, \text{yield}(T)=O} P(T)/P(O)
\]

\[
= \arg \max_{T, \text{yield}(T)=O} P(T)
\]
Observation probability

Relevant to unsupervised PCFG training is the fact that on the basis of the tree probability, a probability is defined for an observation sequence:

\[ P(\mathcal{O}) = \sum_{all \ T, \text{yield}(T)=\mathcal{O}} P(T) \]
Analogy with HMMs

- word sequence to be parsed is analogous to the obs sequence of an HMM.
- internal nodes and their linking are analogous to the hidden state sequence of an HMM.
Analogy with HMMs

- word sequence to be parsed is analogous to the obs sequence of an HMM.
- internal nodes and their linking are analogous to the hidden state sequence of an HMM.
- PCFG model of tree probability is very simple, with nothing conditioned on anything distant – entirely analogously to HMMs.
Analogy with HMMs

- word sequence to be parsed is analogous to the obs sequence of an HMM.
- internal nodes and their linking are analogous to the hidden state sequence of an HMM.
- PCFG model of tree probability is very simple, with nothing conditioned on anything distant – entirely analogously to HMMs.
- because of this, finding the most probable parse is relatively easy
  - need simple adaptation of a bottom-up chart-parsing algorithm, such as the CKY algorithm
  - as with Viterbi alg. for HMMs, key is to have each cell \((p, q)\) in the table record the best parse for each possible category \(C\) at that point, not just the best parse
  - at end pick an overall best from the cell spanning the entire input.
Analogy with HMMs

- **word sequence** to be parsed is analogous to the **obs sequence** of an HMM.
- **internal nodes and their linking** are analogous to the **hidden state sequence** of an HMM.
- PCFG model of tree probability is very simple, with nothing conditioned on anything distant – entirely analogously to HMMs.
- because of this, finding the most probable parse is relatively easy
  - need simple adaptation of a bottom-up chart-parsing algorithm, such as the CKY algorithm
  - as with Viterbi alg. for HMMs, key is to have each cell \((p, q)\) in the table record the best parse for each possible category \(C\) at that point, not just the best parse
  - at end pick an overall best from the cell spanning the entire input.
- there is also a generalisation of the Baum-Welch re-estimation procedure for HMMs to PCFGs: **the Inside/Outside algorithm** (due to Baker 1979).
Analogy with HMMs

- word sequence to be parsed is analogous to the obs sequence of an HMM.
- internal nodes and their linking are analogous to the hidden state sequence of an HMM.
- PCFG model of tree probability is very simple, with nothing conditioned on anything distant – entirely analogously to HMMs.
- because of this, finding the most probable parse is relatively easy
  - need simple adaptation of a bottom-up chart-parsing algorithm, such as the CKY algorithm
  - as with Viterbi alg. for HMMs, key is to have each cell \((p, q)\) in the table record the best parse for each possible category \(C\) at that point, not just the best parse
  - at end pick an overall best from the cell spanning the entire input.
- there is also a generalisation of the Baum-Welch re-estimation procedure for HMMs to PCFGs: the Inside/Outside algorithm (due to Baker 1979).
- the analogy with HMMs is very strong: for grammars with only \(C^i \rightarrow w\) and \(C^i \rightarrow w, C^j\) type rules, the PCFG case collapses into a HMM.
Recall for Lec 1: Unsupervised case for Seq-to-Tree
Recall for Lec 1: Unsupervised case for Seq-to-Tree

unsupervised case: learn

\[ f: \text{sequence over vocab } \Sigma \Rightarrow \text{tree over } (\Sigma \times \text{labels } C) \]
Recall for Lec 1: Unsupervised case for Seq-to-Tree

unsupervised case: learn

\[ f: \text{sequence over vocab } \Sigma \Rightarrow \text{tree over } (\Sigma \times \text{labels } C) \]

from

\[
\{ \begin{align*}
\vdots \\
\text{last week IBM bought Lotus} \\
\vdots 
\end{align*}
\}
\]

i.e. training data is (equivalent to) large corpus of possible inputs
Inside-Outside Unsupervised PCFG training

- **Inside-Outside re-estimation:**
  given an obs sequence $O$, an iterative method which refines parameters $(\pi, A)$ to maximise $P(O)$

- an instance of *Expectation-Maximisation* learning
EM re-estimation of PCFGs (brute-force version)
EM re-estimation of PCFGs (brute-force version)

Expectations
EM re-estimation of PCFGs (brute-force version)

**Expectations**
- using current parameters $(\pi, A)$, consider all trees for the obs $\mathcal{O}$
  
  
  | Tree  | $P(tree_i|\mathcal{O})$ |
  |-------|------------------------|
  | $tree_1$ | $P(tree_1|\mathcal{O})$ |
  | $\vdots$ | $\vdots$ |
  | $tree_n$ | $P(tree_n|\mathcal{O})$ |
EM re-estimation of PCFGs (brute-force version)

**Expectations**

- using current parameters \((\pi, A)\),
  consider all trees for the obs \(O\)
  
  \[
  \begin{align*}
  \text{tree}_1 & \quad P(\text{tree}_1|O) \\
  \vdots & \quad \vdots \\
  \text{tree}_n & \quad P(\text{tree}_n|O)
  \end{align*}
  \]

- use the tree probs to generate
  expected counts \(E(X)\) for an
  event \(X\) via

  \[
  \sum_{\text{tree}} \text{count}(X \text{ in tree}) \times P(\text{tree}|O)
  \]

  eg. \(X\) might be a \(M \rightarrow \text{rhs}\)
  sub-tree).
EM re-estimation of PCFGs (brute-force version)

**Expectations**

- using current parameters \((\pi, A)\), consider all trees for the obs \(O\)
  
  \[
  \begin{align*}
  &tree_1 & P(tree_1|O) \\
  \vdots & \vdots  \\
  &tree_n & P(tree_n|O)
  \end{align*}
  \]

- use the tree probs to generate expected counts \(E(X)\) for an event \(X\) via

\[
\sum_{tree} \text{count}(X \text{ in tree}) \times P(tree|O)
\]

eg. \(X\) might be a \(M \rightarrow rhs\) sub-tree).

- thus get a *distribution* for different types of event
EM re-estimation of PCFGs (brute-force version)

**Expectations**
- using current parameters \((\pi, A)\), consider all trees for the obs \(O\):
  
  \[
  \begin{align*}
  & \text{tree}_1 \quad P(\text{tree}_1|O) \\
  & \vdots \quad \vdots \\
  & \text{tree}_n \quad P(\text{tree}_n|O)
  \end{align*}
  \]

  use the tree probs to generate expected counts \(E(X)\) for an event \(X\) via

  \[
  \sum_{\text{tree}} \text{count}(X \text{ in tree}) \times P(\text{tree}|O)
  \]

  eg. \(X\) might be a \(M \rightarrow \text{rhs}\) sub-tree).

**Maximisation**
- generate new \((\hat{\pi}, \hat{A})\) from expectations eg.

  \[
  \hat{P}(\text{rhs}|M) = \frac{E(M \rightarrow \text{rhs})}{E(M)}
  \]
EM re-estimation of PCFGs (brute-force version)

**Expectations**
- using current parameters \((\pi, A)\),
  consider all trees for the obs \(O\)
  \[
  \begin{align*}
  \text{tree}_1 & \quad P(\text{tree}_1|O) \\
  & \vdots \\
  \text{tree}_n & \quad P(\text{tree}_n|O)
  \end{align*}
  \]
- use the tree probs to generate *expected counts* \(E(X)\) for an event \(X\) via
  \[
  \sum_{\text{tree}} \text{count}(X\ in\ \text{tree}) \times P(\text{tree}|O)
  \]
  eg. \(X\) might be a \(M \rightarrow \text{rhs}\) sub-tree).
- thus get a *distribution* for different types of event

**Maximisation**
- generate new \((\hat{\pi}, \hat{A})\) from expectations eg.
  \[
  \hat{P}(\text{rhs}|M) = \frac{E(M \rightarrow \text{rhs})}{E(M)}
  \]
- this maximises the likelihood of the *distribution* just obtained
EM re-estimation of PCFGs (brute-force version)

**Expectations**
- using current parameters \((\pi, A)\), consider all trees for the obs \(O\)
  
  \[
  \begin{align*}
  \text{tree}_1 & \quad P(\text{tree}_1|O) \\
  \vdots & \quad \vdots \\
  \text{tree}_n & \quad P(\text{tree}_n|O)
  \end{align*}
  \]

  use the tree probs to generate **expected counts** \(E(X)\) for an event \(X\) via

  \[
  \sum_{\text{tree}} \text{count}(X \text{ in tree}) \times P(\text{tree}|O)
  \]

  eg. \(X\) might be a \(M \rightarrow \text{rhs}\) sub-tree).

- thus get a **distribution** for different types of event

**Maximisation**
- generate new \((\hat{\pi}, \hat{A})\) from expectations eg.

  \[
  \hat{P}(\text{rhs}|M) = \frac{E(M \rightarrow \text{rhs})}{E(M)}
  \]

  this maximises the likelihood of the **distribution** just obtained

  now go back to the **E** phase
Inside-Outside properties

EM re-estimation (brute-force) in outline:

until a fixed-point {
    enumerate all trees
    use current \((\pi_i, A_i)\) to generate expected counts
    from expected counts generate new \((\pi_{i+1}, A_{i+1})\)
}

basically same properties as Baum-Welch for HMMs
Inside-Outside properties

EM re-estimation (brute-force) in outline:

\[
\text{until a fixed point } \{ \\
\text{enumerate all trees} \\
\text{use current } (\pi_i, A_i) \text{ to generate expected counts} \\
\text{from expected counts generate new } (\pi_{i+1}, A_{i+1}) \}
\]

basically same properties as Baum-Welch for HMMs

- at each iteration, something is maximised, namely the likelihood of the distribution of tree-event expectations at the E step. But something more objective than that also gets maximised: the observation probabilities get higher:

\[
P_i(O) \leq P_{i+1}(O)
\]
Inside-Outside properties

EM re-estimation (brute-force) in outline:

until a fixed-point {
    enumerate all trees
    use current \((\pi_i, A_i)\) to generate expected counts
    from expected counts generate new \((\pi_{i+1}, A_{i+1})\)
}

basically same properties as Baum-Welch for HMMs

- at each iteration, something is maximised, namely the likelihood of the distribution of tree-event expectations at the E step. But something more objective than that also gets maximised: the observation probabilities get higher:

\[
P_i(O) \leq P_{i+1}(O)
\]

- if obs \(O\) small, will be poor on unseen
Inside-Out properties

EM re-estimation (brute-force) in outline:

until a fixed-point {
  enumerate all trees
  use current \((\pi_i, A_i)\) to generate \textit{expected} counts
  from expected counts generate new \((\pi_{i+1}, A_{i+1})\)
}

basically same properties as Baum-Welch for HMMs

- at each iteration, something is maximised, namely the likelihood of the distribution of tree-event expectations at the E step. But something more objective than that also gets maximised: the observation probabilities get higher:

\[ P_i(\mathcal{O}) \leq P_{i+1}(\mathcal{O}) \]

- if obs \(\mathcal{O}\) small, will be poor on unseen
- local maxima: not guaranteed to converge to \textit{the} best model
Inside-Outside properties

EM re-estimation (brute-force) in outline:

\[
\text{until a fixed-point} \{ \\
\quad \text{enumerate all trees} \\
\quad \text{use current } (\pi_i, A_i) \text{ to generate expected counts} \\
\quad \text{from expected counts generate new } (\pi_{i+1}, A_{i+1}) \\
\}\]

basically same properties as Baum-Welch for HMMs

- at each iteration, something is maximised, namely the likelihood of the distribution of tree-event expectations at the E step. But something more objective than that also gets maximised: the observation probabilities get higher:

\[
P_i(\mathcal{O}) \leq P_{i+1}(\mathcal{O})
\]

- if obs \(\mathcal{O}\) small, will be poor on unseen
- local maxima: not guaranteed to converge to the best model
- increases \(P(\mathcal{O})\), but maybe really want to increase accuracy(\(\text{parse}(\mathcal{O})\))
Inside-Outside properties

EM re-estimation (brute-force) in outline:

\[
\text{until a fixed-point } \{ \\
\text{enumerate all trees} \\
\text{use current } (\pi_i, A_i) \text{ to generate expected counts} \\
\text{from expected counts generate new } (\pi_{i+1}, A_{i+1}) \\
\} 
\]

basically same properties as Baum-Welch for HMMs

- at each iteration, something is maximised, namely the likelihood of the distribution of tree-event expectations at the E step. But something more objective than that also gets maximised: the observation probabilities get higher:

\[
P_i(O) \leq P_{i+1}(O)
\]

- if obs \( O \) small, will be poor on unseen

- **local maxima**: not guaranteed to converge to *the* best model

- increases \( P(O) \), but maybe really want to increase \( \text{accuracy}(\text{parse}(O)) \)

- brute force version is infeasible: *Baker’s real Inside-Outside* is a cleverer, feasible version
Avoiding the brute force cost

- exponentially costly to enumerate all possible trees
- strategy: define various span-specific probabilities, and derive expectations by summing over all spans
recall HMM case

if you find a way to do this

'prob of tag i with word $o_t$, regardless of tags for other words'

you can get 'E for i anywhere' by summing over times $t$

$\alpha(t, i)$ $\beta(t, i)$

$\alpha_t(i)\beta_t(i) = \text{the probability emitting observations symbols}$

$\mathcal{O}_{1,T}$ $\text{and being in state } i \text{ at time } t$

$= P(o_1 \ldots o_t, s_t = i, o_{t+1} \ldots o_T)$
recall HMM case

if you find a way to do this

'prob of tag i with word $o_{t-1}$ and tag j with word $o_t$, regardless of tags for other words'

you can get 'E for i followed by j anywhere' by summing over times $t$

\[
\xi_t(i, j) = \frac{\alpha_{t-1}(i) \times a_{ij} b_j(o_t) \times \beta_t(j)}{P(O)}
\]
Let $C_{pq}^i$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$. 
rough plan for PCFG case

- Let $C^i_{pq}$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$.
- if we find a way to do this

  'prob of $C^i_{pq}$ for words $o_p \ldots o_q$, regardless of rest of structure

you can get 'E for $C^i$ anywhere' by summing over spans $p, q$
rough plan for PCFG case

- Let $C^i_{pq}$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$.
- If we find a way to do this
  
  'prob of $C^i_{pq}$ for words $o_p \ldots o_q$, regardless of rest of structure

  you can get 'E for $C^i$ anywhere' by summing over spans $p, q$

- If we find a way to do this
  
  'prob of a $C^i \rightarrow C^i C^k$ sub-tree spanning $p, q$, regardless of rest of structure

  you can get 'E for $C^i \rightarrow C^i C^k$ sub-tree anywhere' by summing over spans $p, q$
Outside probability $\alpha_{pq}(i)$

Let $C_{pq}^i$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$. 
Outside probability $\alpha_{pq}(i)$

Let $C_{pq}^i$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$.

$$\alpha_{pq}(i) = \text{the probability of a tree whose terminal sequence is } o_1, \ldots, o_{p-1} X o_{q+1}, \ldots, o_T \text{ and has a category } C^i, \text{ dominating the sequence } X$$

$$= P(o_1, \ldots, o_{p-1}, C_{pq}^i, o_{q+1}, \ldots, o_T)$$
Outside probability $\alpha_{pq}(i)$

Let $C^i_{pq}$ mean a node $C^i$ dominating terminals starting at position $p$ and ending at position $q$.

$$\alpha_{pq}(i) = \text{the probability of a tree whose terminal sequence is } o_1, \ldots, o_{p-1}Xo_{q+1}, \ldots, o_T \text{ and has a category } C^i, \text{ dominating the sequence } X$$

$$= P(o_1, \ldots, o_{p-1}, C^i_{pq}, o_{q+1}, \ldots, o_T)$$
Inside Probability $\beta_{pq}(i)$
Inside Probability $\beta_{pq}(i)$

\[
\beta_{pq}(i) = \text{the probability of tree having terminal sequence } o_p, \ldots, o_q, \text{ given mother } C^i \\
= P(o_p, \ldots, o_q|C^i_{pq})
\]
Inside Probability $\beta_{pq}(i)$

$$\beta_{pq}(i) = \text{the probability of tree having terminal sequence } o_p, \ldots, o_q, \text{ given mother } C^i$$

$$= P(o_p, \ldots, o_q | C^i_{pq})$$
Multiplying the two together gives the joint probability of the observation sequence and an occurrence of $C_p^i$:

$$\alpha_{pq}(i)\beta_{pq}(i) = P(o_1, \ldots, o_T, C_p^i)$$
α_{pq} and β_{pq}

\[ \alpha_{pq}(i) = \begin{cases} 
\text{the probability of a tree whose terminal sequence is } o_1, \ldots, o_{p-1} X o_{q+1}, \ldots, o_T \\
\text{and has a category } C_i, \text{ dominating the sequence } X \\
= P(o_1, \ldots, o_{p-1}, C^i_{pq}, o_{q+1}, \ldots, o_T) 
\end{cases} \]
\(\alpha_{pq}\) and \(\beta_{pq}\)

\[\beta_{pq}(i) = \text{the probability of tree having terminal sequence } o_p, \ldots, o_q, \text{ given mother } C^i\]

\[= P(o_p \ldots o_q|C^i_{pq})\]
\( \alpha_{pq} \) and \( \beta_{pq} \)

\[
\begin{align*}
\alpha_{pq}(i) \beta_{pq}(i) &= \text{the probability of } O_1, T \text{ and } C_i^{pq} \\
&= P(o_1, \ldots, o_T, C_i^{pq})
\end{align*}
\]
the real Inside-Outside: summing the span-specific probs
the real Inside-Outside: summing the span-specific probs

- $\alpha_{pq}(i)$ and $\beta_{pq}(i)$ can be recursively calculated in CKY–esque style
- expected counts can be calculated by summations over terms involving $\alpha$ and $\beta$
the real Inside-Outside: summing the span-specific prods

- $\alpha_{pq}(i)$ and $\beta_{pq}(i)$ can be recursively calculated in CKY–esque style
- expected counts can be calculated by summations over terms involving $\alpha$ and $\beta$
- 'category prob'
  $\gamma_{pq}(i) = \text{the probability of } C_{pq}^i \text{ given } O$
  $= \alpha_{pq}(i)\beta_{pq}(i)/P(O)$
the real Inside-Outside: summing the span-specific probs

- \( \alpha_{pq}(i) \) and \( \beta_{pq}(i) \) can be recursively calculated in CKY–esque style
- expected counts can be calculated by summations over terms involving \( \alpha \) and \( \beta \)
- 'category prob'
  \[ \gamma_{pq}(i) = \text{the probability of } C_{pq}^i \text{ given } \mathcal{O} \]
  \[ = \frac{\alpha_{pq}(i) \beta_{pq}(i)}{P(\mathcal{O})} \]
- 'sub tree prob'
  \[ \xi_{pq}(i, j, k) = \text{the probability of the use of } C^i \rightarrow C^j C^k \text{ spanning } p, q \]
  \[ = \frac{1}{P(\mathcal{O})} \left[ \alpha_{pq}(i) \times P(C^j C^k | C^i) \times (\sum_{d=p}^{q-1} \beta_{p,d}(j) \beta_{d+1,q}(k)) \right] \]
the real Inside-Outside: summing the span-specific probs

- $\alpha_{pq}(i)$ and $\beta_{pq}(i)$ can be recursively calculated in CKY–esque style
- expected counts can be calculated by summations over terms involving $\alpha$ and $\beta$
- 'category prob'
  \[ \gamma_{pq}(i) = \frac{\alpha_{pq}(i)\beta_{pq}(i)}{P(O)} \]
- 'sub tree prob'
  \[ \xi_{pq}(i,j,k) = \frac{1}{P(O)} \left[ \alpha_{pq}(i) \times P(C^i C^j C^k | C^i) \times \left( \sum_{d=p}^{q-1} \beta_{p,d}(j)\beta_{d+1,q}(k) \right) \right] \]
  \[ P(O) = \sum_i \alpha_{pq}(i)\beta_{pq}(i), \text{ any } p, q, p < q \]
the real Inside-Outside: summing the span-specific probs

- $\alpha_{pq}(i)$ and $\beta_{pq}(i)$ can be recursively calculated in CKY–esque style
- expected counts can be calculated by summations over terms involving $\alpha$ and $\beta$
- 'category prob'
  $\gamma_{pq}(i) = \text{the probability of } C_{pq}^i \text{ given } O$
  $= \frac{\alpha_{pq}(i) \beta_{pq}(i)}{P(O)}$
- 'sub tree prob'
  $\xi_{pq}(i, j, k) = \text{the probability of the use of } C^i \rightarrow C^j C^k \text{ spanning } p, q$
  given $O$
  $= \frac{1}{P(O)} \left[ \alpha_{pq}(i) \times P(C^j C^k | C^i) \times \left( \sum_{d=p}^{q-1} \beta_{p,d}(j) \beta_{d+1,q}(k) \right) \right]$
- $P(O) = \sum_i \alpha_{pq}(i) \beta_{pq}(i)$, any $p, q, p < q$
- so re-estimations

$\hat{P}(C^j, C^k | C^i) = \frac{\sum_{p=1}^{T-1} \sum_{q=p+1}^{T} (\xi_{pq}(i, j, k))}{\sum_{p=1}^{T} \sum_{q=p}^{T} (\gamma_{pq}(i))}$

$\hat{P}(w | C^i) = \frac{\sum_{p=1}^{T} \mid_{o_t=w} \gamma_{pp}(i)}{\sum_{p=1}^{T} \sum_{q=p}^{T} (\gamma_{pq}(i))}$
Baker's Unsupervised PCFG training algorithm

\[ P(C_j C_k | C_i) \]
\[ \times \beta(p,d,j) \beta(d+1,q,k) \text{ prob of stuff inside } C_j C_k \]

\[ \xi_{pq}(i, j, k) = \text{the probability of the use of } C^i \rightarrow C^j C^k \text{ spanning } p, q \text{ given } O \]

\[ = \frac{1}{P(O)}\left[ \alpha_{pq}(i) \times P(C^j C^k | C^j) \times (\sum_{d=p}^{q-1} \beta_{p,d}(j)\beta_{d+1,q}(k)) \right] \]
for *single* observation sequence, the formulae for category and sub-tree
probs have a $P(O)$ denominator

not strictly necessary, as ratios taken in re-estimation

but for *multiple* observations sequences, $O^1 \ldots O^M$, expectations
summed, and each $P(O^m)$ matters

\[
\hat{P}(C^i, C^k | C^i) = \frac{\sum_m \sum_{p=1}^{T_m-1} \sum_{q=p+1}^{T_m} (\xi_{pq}^{O^m}(i, j, k))}{\sum_m \sum_{p=1}^{T_m} \sum_{q=p}^{T_m} (\gamma_{pq}^{O^m}(i))}
\]

\[
\hat{P}(w | C^i) = \frac{\sum_m \sum_{p=1}^{T_m} |o_t=w \gamma_{pp}^{O^m}(i))}{\sum_m \sum_{p=1}^{T_m} \sum_{q=p}^{T_m} (\gamma_{pq}^{O^m}(i))}
\]
Outcomes on artificial corpora
Outcomes on artificial corpora

- PCFG for *palindromes*
  
  \[\begin{align*}
  S &\rightarrow AC \quad [0.4] \\
  S &\rightarrow BD \quad [0.4] \\
  S &\rightarrow AA \quad [0.1] \\
  S &\rightarrow BB \quad [0.1] \\
  C &\rightarrow SA \quad [1.0] \\
  D &\rightarrow SB \quad [1.0] \\
  A &\rightarrow a \quad [1.0] \\
  B &\rightarrow b \quad [1.0]
  \end{align*}\]

  - use the grammar to stochastically generate a training corpus
  - train back a PCFG from this
Outcomes on artificial corpora

- **PCFG for palindromes**
  
  \[
  S \rightarrow AC \quad [0.4] \\
  S \rightarrow BD \quad [0.4] \\
  S \rightarrow AA \quad [0.1] \\
  S \rightarrow BB \quad [0.1] \\
  C \rightarrow SA \quad [1.0] \\
  D \rightarrow SB \quad [1.0] \\
  A \rightarrow a \quad [1.0] \\
  B \rightarrow b \quad [1.0] \\
  \]

- use the grammar to stochastically generate a training corpus
- train back a PCFG from this

- A number of people have done experiments like this
  - for example Benedi and Sanchez 2005
    - generate a training set of 1000 strings
    - iterate the Inside-Outside procedure 340 times
    - 76% of randomly generated strings by the resulting grammar are palindromes.
Outcomes with more realistic corpora

- Aside: *dependency*-format CFGs (DEP-CFG)
  - categories just terminals $t_i$, or their ’1-bar’ projection $\overline{t_i}$
  - rules all of the form

$$\overline{t_i} \rightarrow \overline{t_0} \ldots \overline{t_{i-1}} \ t_i \ \overline{t_{i+1}} \ldots \overline{t_n}$$
Outcomes with more realistic corpora

- Aside: *dependency*-format CFGs (DEP-CFG)
  - categories just terminals $t_i$, or their '1-bar' projection $\bar{t_i}$
  - rules all of the form
    $$
    \bar{t_i} \rightarrow \bar{t_0} \ldots \bar{t_{i-1}} \ t_i \ \bar{t_{i+1}} \ldots \bar{t_n}
    $$

- Charniak and Carroll 92
  - made artificial, but modestly realistic DEP-CFG (approx 30 rules), added probs, stochastically generated corpus (approx 9000 words)
Outcomes with more realistic corpora

- Aside: *dependency*-format CFGs (DEP-CFG)
  - categories just terminals $t_i$, or their '1-bar' projection $\overline{t_i}$
  - rules all of the form
    \[
    \overline{t_i} \rightarrow \overline{t_0} \ldots \overline{t_{i-1}} \overline{t_i} \overline{t_{i+1}} \ldots \overline{t_n}
    \]

- Charniak and Carroll 92
  - made artificial, but modestly realistic DEP-CFG (approx 30 rules), added probs, stochastically generated corpus (approx 9000 words)
  - ran Inside-Outside to assign probs to a DEP-CFG (contains every possible DEP-CFG rule, subject to length 4 max for RHS)
Outcomes with more realistic corpora

- Aside: *dependency*-format CFGs (DEP-CFG)
  - categories just terminals $t_i$, or their '1-bar' projection $\overline{t_i}$
  - rules all of the form
    \[
    \overline{t_i} \rightarrow \overline{t_0} \ldots \overline{t_{i-1}} t_i \overline{t_{i+1}} \ldots \overline{t_n}
    \]

- Charniak and Carroll 92
  - made artificial, but modestly realistic DEP-CFG (approx 30 rules), added probs, stochastically generated corpus (approx 9000 words)
  - ran Inside-Outside to assign probs to a DEP-CFG (contains every possible DEP-CFG rule, subject to length 4 max for RHS)
  - did 300 runs, each with different random initial probs
Outcomes with more realistic corpora

- Aside: *dependency*-format CFGs (DEP-CFG)
  - categories just terminals $t_i$, or their ’1-bar’ projection $\overline{t_i}$
  - rules all of the form
    $$\overline{t_i} \rightarrow \overline{t_0} \ldots \overline{t_{i-1}} t_i \overline{t_{i+1}} \ldots \overline{t_n}$$

- **Charniak and Carroll 92**
  - made artificial, but modestly realistic DEP-CFG (approx 30 rules), added probs, stochastically generated corpus (approx 9000 words)
  - ran Inside-Outside to assign probs to a DEP-CFG (contains every possible DEP-CFG rule, subject to length 4 max for RHS)
  - did 300 runs, each with different random initial probs
  - **300 different** grammars were learnt

- result widely noted as showing that Inside-Outside algorithm is very vulnerable to local maxima
Charniak’s finding revisited

- Klein and Manning 2002,2004,2005 revisit this

\[^1\] F1 is \(\frac{2PR}{R+P}\)
Charniak’s finding revisited

- **Klein and Manning 2002,2004,2005** revisit this

- use a binary-branching version of the DEP-CFG format, terminals are POS tags of PTB

\[ F_1 \text{ is } \frac{2PR}{R+P} \]
Charniak’s finding revisited

- **Klein and Manning 2002,2004,2005** revisit this
- use a binary-branching version of the DEP-CFG format, terminals are POS tags of PTB
- initialise not with *random* probabilities, but *uniform*
Charniak's finding revisited

- **Klein and Manning 2002, 2004, 2005** revisit this

- use a binary-branching version of the DEP-CFG format, terminals are POS tags of PTB

- initialise not with *random* probabilities, but *uniform*

- train on length $\leq 10$ subcorpus of the PTB (7422 sentences)

\[ F_1 = \frac{2PR}{R+P} \]
Charniak’s finding revisited

- **Klein and Manning 2002,2004,2005** revisit this

- use a binary-branching version of the DEP-CFG format, terminals are POS tags of PTB

- initialise not with *random* probabilities, but *uniform*

- train on length $\leq 10$ subcorpus of the PTB (7422 sentences)

- a more encouraging outcome: the trained grammar achieves $F1: 48.2\%$, for unlabelled precision/recall \(^1\)

\(^1\)F1 is $\frac{2PR}{R+P}$
The Consituent/Context Model

- Klein and Manning have developed an alternative unsupervised approach to learning unlabeled bracketing: the Consituent/Context Model (CCM)

- Works with length $\leq 10$ sub-corpus of the PTB, taking POS tags as terminals the observation symbols

- attainings $\textbf{F1}: 71\%$, on unlabeled precision/recall
CCM sketch

represent bracketings by span table  

\[ B(p, q) = 1: \text{constituent} \]
\[ B(p, q) = 0: \text{distituent} \]
CCM sketch

represent bracketings by span table

analogous to 'inside' prob, have probs relating constituency/distituency to inside terminals

\[ B(p, q) = 1: \text{ constituent} \]
\[ B(p, q) = 0: \text{ distituent} \]

\[ P(\mathcal{O}_{pq}|B_{pq} = 1) \]
\[ P(\mathcal{O}_{pq}|B_{pq} = 0) \]
CCM sketch

- Represent bracketings by span table

- Analogous to ‘inside’ prob, have probs relating constituency/distituency to inside terminals

- Analogous to ‘outside’ prob, have probs relating constituency/distituency to outside pair of terminal sequences

\[ B(p, q) = 1: \text{constituent} \]
\[ B(p, q) = 0: \text{distituent} \]

\[ P(O_{pq} | B_{pq} = 1) \]
\[ P(O_{pq} | B_{pq} = 0) \]

\[ P(O_{1,p-1} \sim O_{q+1,T} | B_{pq} = 1) \]
\[ P(O_{1,p-1} \sim O_{q+1,T} | B_{pq} = 0) \]
CCM sketch

- represent bracketings by span table
- analogous to 'inside' prob, have probs relating constituency/distituency to inside terminals
- analogous to 'outside' prob, have probs relating constituency/distituency to outside pair of terminal sequences

▷ CCM joint prob of observations $\mathcal{O}$ and a bracketing $B$:

\[
P(\mathcal{O}_{1:T}, B) = P(B) P(\mathcal{O}_{1:T} | B) \\
\approx P(B) \times \prod_{pq} P(\mathcal{O}_{pq} | B_{pq}) P(\mathcal{O}_{1:p-1} \sim O_{q+1:T} | B_{pq})
\]

▷ trained using a variant of Inside-Outside, again an instance of EM
Latent Tree Annotation

Petrov et al 2006
Latent Tree Annotation

Petrov et al 2006

- standard set-up for Inside-Outside
  visible:  **words**
  hidden:   **rules**
Latent Tree Annotation

Petrov et al 2006

- standard set-up for Inside-Outside
  visible: words
  hidden: rules

- variant set-up
  visible: PTB trees
  hidden: annotated PTB trees
Latent Tree Annotation

Petrov et al 2006

- standard set-up for Inside-Outside
  visible: words
  hidden: rules

- variant set-up
  visible: PTB trees
  hidden: annotated PTB trees

- Petrov et al use EM learning to split PTB labels into sub-cases: a tree with the sub-symbols is the annotated tree.
Probabilistic Grammars

Variant applications of Inside-Outside algorithm

Split/Merge cycles
Split/Merge cycles

- basically goes through a small number of *split-merge* stages $SM_1, \ldots SM_n$ at each of which all current labels are split, and then some merged back
Split/Merge cycles

- basically goes through a small number of split-merge stages $SM_1, \ldots SM_n$ at each of which all current labels are split, and then some merged back
- at each split-merge stage $SM_i$
  - share prob of un-split rule amongst split versions, with small amount of noise to break symmetry
  - iterate: use currents probs and PTB trees to get expected counts for events using split symbols; then re-estimate split rule probs
  - end $SM_i$ by a merge phase, abandoning a split if there is a small estimated loss in likelihood by making the merge
Split/Merge cycles

- basically goes through a small number of *split-merge* stages $SM_1, \ldots SM_n$ at each of which all current labels are split, and then some merged back
- at each split-merge stage $SM_i$
  - share prob of un-split rule amongst split versions, with small amount of noise to break symmetry
  - iterate: use currents probs and PTB trees to get expected counts for events using split symbols; then re-estimate split rule probs
  - end $SM_i$ by a *merge* phase, abandoning a split if there is a small estimated loss in likelihood by making the merge

- Performance
  - evaluation is prediction of un-annotated tree
  - initial PCFG with 98 symbols: **F1 63.4%**.
    after 6 split-merge re-estimation rounds, where 50% of newly split symbols are merged, grammar has 1043 symbols, **F1 90.2%**.

- this is very reminiscent of Decision Tree growth and pruning
learns linguistically plausible splits?

- claim that many of the refinements learnt are linguistically interpretable.
learns linguistically plausible splits?

- claim that many of the refinements learnt are linguistically interpretable.
- single pre-terminal category DT ends up divided into sub-symbols recognisable as definite determiners (e.g. the), indefinite (e.g. a), demonstratives (e.g. this) and quantificational elements (e.g. some).
learns linguistically plausible splits?

- claim that many of the refinements learnt are linguistically interpretable.

- single pre-terminal category DT ends up divided into sub-symbols recognisable as definite determiners (eg. the), indefinite (eg. a), demonstratives (eg. this) and quantificational elements (eg. some).

- excerpt from the table of the likeliest words in the subsymbols for VBZ:

<table>
<thead>
<tr>
<th>VBZ-0</th>
<th>gives</th>
<th>sells</th>
<th>takes</th>
<th>ditransitives ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBZ-4</td>
<td>says</td>
<td>adds</td>
<td>Says</td>
<td>sentence complement, communication</td>
</tr>
<tr>
<td>VBZ-5</td>
<td>believes</td>
<td>means</td>
<td>thinks</td>
<td>sentence complement, cognitive ?</td>
</tr>
<tr>
<td>VBZ-6</td>
<td>expects</td>
<td>makes</td>
<td>calls</td>
<td>control verbs ?</td>
</tr>
</tbody>
</table>
Supervised Learning

- *supervised* version of learning a model of tree probability
  - define some model of tree probability
    - ie. fix the equations
    - parameters still unknown
  - use tree-bank evidence to fix the parameters
  - evaluate most-probable parses against (a disjoint subset of) tree-bank evidence

- *supervised* approaches much more wide-spread than *unsupervised*
Base-line: treebank PCFG

- stay with PCFG model
- estimate $A, \pi$ parameters by counts in PTB
- this has come to be called the treebank PCFG
Base-line: treebank PCFG

- stay with PCFG model
- estimate $A, \pi$ parameters by counts in PTB
- this has come to be called the treebank PCFG
- how well does it work?

<table>
<thead>
<tr>
<th>Details</th>
<th>Labelled prec/rec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak 96 terminals are PTB POS-tags</td>
<td>F1: 79.6</td>
</tr>
<tr>
<td>Klein and Manning 2002 same but with $\leq$ 10 sub-corpus</td>
<td>F1: 82</td>
</tr>
<tr>
<td>Klein and Manning 2003 terminals are PTB words, full corpus</td>
<td>F1: 72.6</td>
</tr>
</tbody>
</table>

- consensus is that this works surprisingly well
might expect lower numbers because *data-sparsity of the treebank*
might expect lower numbers because *data-sparsity of the treebank*

- **missing rules problem** seems likely that best parse of a test item will need a rule that was never seen in the training data – a missing rule – so best parse will be prob 0
  - about 15% of trees in training use a rule which is a one-off
might expect lower numbers because *data-sparsity of the treebank*

- **missing rules problem** seems likely that best parse of a test item will need a rule that was never seen the training data – a missing rule – so best parse will be prob 0
  - about 15% of trees in training use a rule which is a one-off

- turns out that although the correct parse gets prob 0
  - a parse *can be* found
  - does weird around the place where the unseen rule should be
  - elsewhere does sensible stuff
might expect lower numbers because *data-sparsity of the treebank*

missing rules problem seems likely that best parse of a test item will need a rule that was never seen the training data – a missing rule – so best parse will be prob 0
  - about 15% of trees in training use a rule which is a one-off

turns out that although the correct parse gets prob 0
  - a parse *can be* found
  - does weird around the place where the unseen rule should be
  - elsewhere does does sensible stuff

note scoring procedure:
  - 80% does not mean getting a completely correct parse 80% of the time
  - it means getting 80% of the parts of the parse correct.
Moving beyond the base-line

Two trends in work beating the treebank PCFG base-line

- assume treebank data can be given additional annotation, and parsing model will refer to this information
- instead of treating local tree as an indivisible unit, assigned a probability in its entirety, this probability should be factorised
Example of added annotation: *heads*

- for each local tree, one daughters is designated the *head daughter*
- head features (such as head word, or head POS) are recursively percolated via
  
  head features of mother = head features of the head daughter

- the head daughter annotation by hand-written head-placing heuristics
Example of added annotation: *heads*

- for each local tree, one daughters is designated the *head daughter*
- head features (such as head word, or head POS) are recursively percolated via
  
  \[ \text{head features of mother} = \text{head features of the head daughter} \]
- the head daughter annotation by hand-written head-placing heuristics
Why might heads help

- the treebank PCFG derived straight from the PTB: gives a distribution of possible dtrs of $VP$, but this does not refer to particular verbs, or kinds of verbs
Why might heads help

- the treebank PCFG derived straight from the PTB: gives a distribution of possible dtrs of $VP$, but this does not refer to particular verbs, or kinds of verbs
- but for different head words, you would expect the empirical distribution of daughter sequences to be quite different

<table>
<thead>
<tr>
<th></th>
<th>come</th>
<th>take</th>
<th>think</th>
<th>want</th>
</tr>
</thead>
<tbody>
<tr>
<td>$VP \rightarrow V$</td>
<td>9.5%</td>
<td>2.6%</td>
<td>4.6%</td>
<td>5.7%</td>
</tr>
<tr>
<td>$VP \rightarrow V\ NP$</td>
<td>1.1%</td>
<td>31.1%</td>
<td>0.2%</td>
<td>13.9%</td>
</tr>
<tr>
<td>$VP \rightarrow V\ PP$</td>
<td>34.5%</td>
<td>3.1%</td>
<td>7.1%</td>
<td>0.3%</td>
</tr>
<tr>
<td>$VP \rightarrow V\ SBAR$</td>
<td>6.6%</td>
<td>0.3%</td>
<td>73.0%</td>
<td>0.2%</td>
</tr>
<tr>
<td>$VP \rightarrow V\ S$</td>
<td>2.2%</td>
<td>1.3%</td>
<td>4.8%</td>
<td>70.8%</td>
</tr>
</tbody>
</table>
Factorising probabilities

instead of treating daughters as indivisible unit, treat daughters right and left of the head daughter independently:

\[ P(\text{mother}, \text{dtrs}) = P(\text{mother}) \times P_H(\text{head dtr}|\text{mother}) \times P_L(\text{dtrs left of head}|\text{head dtr and mother}) \times P_R(\text{dtrs right of head}|\text{head dtr and mother}) \]
Markovisation

- $P_L$ and $P_R$ are often designed to treat the daughter sequences as some kind of Markov process.
- Unlike PCFG model, this assigns a non-zero probability to daughter sequences of all lengths.
- This is somewhat in accord with the PTB trees, which are often wide and flat, especially in the treatment of optional items, such as adjectives and adverbs.
Daughter sequences as 0-order Markov processes
Daughter sequences as 0-order Markov processes
Daughter sequences as 0-order Markov processes

\[
P(S^{bought}) \times P_H(VP^{bought} | S^{bought}) \times P_L(NP^{IBM} | Condition) \times P_L(NP^{week} | Condition) \times P_L(STOP_L | Condition) \times P_R(STOP_R | Condition)
\]

Condition is

\[\text{head-dtr} = VP^{bought}, \text{mother} = S^{bought}\]

- This is basically Collins’ Model 1 (Collins 2003)
Daughter sequences as 0-order Markov processes

\[
P(S^{bought}) \times P_H(VP^{bought} | S^{bought}) \times P_L(NP^{IBM} | \text{Condition}) \times P_L(NP^{week} | \text{Condition}) \times P_L(STOP_L | \text{Condition}) \times P_R(STOP_R | \text{Condition})
\]

Condition is

\[
\text{head-dtr} = VP^{bought}, \text{mother} = S^{bought}
\]

- This is basically Collins’ Model 1 (Collins 2003)
Daughter sequences as 0-order Markov processes

\[
P(S^{bought}) \times P_H(\text{VP}^{bought}|S^{bought}) \times P_L(\text{NP}^{IBM}|\text{Condition}) \times P_L(\text{NP}^{week}|\text{Condition}) \times P_L(\text{STOP}_L|\text{Condition}) \times P_R(\text{STOP}_R|\text{Condition})
\]

Condition is

\[\text{head-dtr} = \text{VP}^{bought}, \text{mother} = S^{bought}\]

- This is basically Collins’ Model 1 (Collins 2003)
Daughter sequences as 0-order Markov processes

\[
P(S^{bought}) \times P_H(VP^{bought} | S^{bought}) \times P_L(NP^{IBM} | Condition) \times P_L(NP^{week} | Condition) \times P_L(STOP_L | Condition) \times P_R(STOP_R | Condition)
\]

**Condition is**

\(head-dtr = VP^{bought}, mother = S^{bought}\)

- \(STOP_L\) and \(STOP_R\) are pseudo-symbols.
- \(STOP_R\) probability represents how often, when reading daughters right-wards from the head, there are no more right-ward daughters.
- This is basically Collins’ Model 1 (Collins 2003)
Daughter sequences as 0-order Markov processes

\[
P(S^{bought}) \times P_H(VP^{bought} | S^{bought}) \times P_L(NP^{IBM} | Condition) \times P_L(NP^{week} | Condition) \times P_L(STOP_L | Condition) \times P_R(STOP_R | Condition)
\]

Condition is

\[\text{head-dtr} = VP^{bought}, \text{mother} = S^{bought}\]

- \(STOP_L\) and \(STOP_R\) are pseudo-symbols.
- \(STOP_R\) probability represents how often, when reading daughters right-wards from the head, there are no more right-ward daughters.
- This is basically Collins’ Model 1 (Collins 2003)
## Performance

<table>
<thead>
<tr>
<th>Details</th>
<th>Labelled Prec/Rec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein and Manning 2003</td>
<td>treebank PCFG</td>
</tr>
<tr>
<td>Collins 2003</td>
<td>model 1: heads, markovisation</td>
</tr>
</tbody>
</table>
Argument vs. Adjunct annotation

- another further annotation: *argument vs. adjunct* distinction.
Argument vs. Adjunct annotation

- another further annotation: *argument vs. adjunct* distinction.
- again added to treebank trees via hand-written heuristics
Argument vs. Adjunct annotation

- another further annotation: *argument vs. adjunct* distinction.
- again added to treebank trees via hand-written heuristics
- eg. showing *arguments* with subscript $A$
Putting Argument/Adjunct into model: subcategorisation

- associate head with a subcat list, divided into left and right parts, LC and RC.
Putting Argument/Adjunct into model: subcategorisation

- associate head with a *subcat* list, divided into left and right parts, *LC* and *RC*.
- *arguments* to the right conditioned on *RC*, to the left on *LC*
Putting Argument/Adjunct into model: subcategorisation

- associate head with a \textit{subcat} list, divided into left and right parts, \textit{LC} and \textit{RC}.
- \textit{arguments} to the right conditioned on \textit{RC}, to the left on \textit{LC}
- once encountered, strike off \textit{argument} daughter from subcat:

\[
P_R(D_1|\{D_1, D_2, \ldots\} \ldots) \times P_R(D_2|\{D_2, \ldots\} \ldots) \ldots
\]
Putting Argument/Adjunct into model: subcategorisation

- associate head with a `subcat` list, divided into left and right parts, $LC$ and $RC$.
- `arguments` to the right conditioned on $RC$, to the left on $LC$
- once encountered, strike off `argument` daughter from subcat:

$$P_R(D_1|\{D_1, D_2, \ldots\}) \times P_R(D_2|\{D_2, \ldots\}) \ldots$$

- sub-tree forced to have all and only the required arguments by constraints

$$P_R(D|RC) = 0 \text{ if } D \notin RC$$

$$P_R(STOP_R|RC) = 0 \text{ if } RC \neq \emptyset$$
Putting Argument/Adjunct into model: subcategorisation

- associate head with a *subcat* list, divided into left and right parts, *LC* and *RC*.
- *arguments* to the right conditioned on *RC*, to the left on *LC*
- once encountered, strike off *argument* daughter from subcat:

\[
P_R(D_1|\{D_1, D_2, \ldots\}) \times P_R(D_2|\{D_2, \ldots\}) \ldots
\]

- sub-tree forced to have all and only the required arguments by constraints

\[
P_R(D|RC) = 0 \text{ if } D \notin RC
\]

\[
P_R(STOP_R|RC) = 0 \text{ if } RC \neq \emptyset
\]

- *adjunct* daughters not conditioned on any subcat list
IBM bought last week. Lotus bought IBM.
$P(S^{bought})$
\[ \times P_H(VP^{bought} | S^{bought}) \]
\[ \times P_{lc}(\{NP_A\} | Cond) \]
\[ \times P_L(NP_A^{IBM} | Cond, \{NP_A\}) \]
\[ \times P_L(NP_{week}^{last} | Cond, \{\}) \]
\[ \times P_L(STOP_L | Cond, \{\}) \]
\[ \times P_{rc}(\{\} | Cond) \]
\[ \times P_R(STOP_R | Cond, \{\}) \]

Cond is

mother = $S^{bought}$, head-dtr is $VP^{bought}$
Probabilistic Grammars

Other models and supervised learning

\[ P(S^{bought}) \]
\[ \times P_H(VP^{bought} \mid S^{bought}) \]
\[ \times P_{ic}(\{NP_A\} \mid Cond) \]
\[ \times P_L(NP_{IBM}^{week} \mid Cond, \{NP_A\}) \]
\[ \times P_L(NP_{week}^{bought} \mid Cond, \{\}) \]
\[ \times P_{rc}(\{\} \mid Cond) \]
\[ \times P_R(STOP_R^{bought} \mid Cond, \{\}) \]

Cond is

\[ \text{mother} = S^{bought}, \text{ head-dtr is } VP^{bought} \]
P(\(S^{bought}\))
\times P_H(\(VP^{bought} \mid S^{bought}\))
\times P_{lc}(\{NP_A\} \mid Cond)
\times P_L(\(NP_{IBM}^{IBM} \mid Cond, \{NP_A\}\))
\times P_L(\(NP_{week}^{week} \mid Cond, \{\}\))
\times P_L(\(STOP_L \mid Cond, \{\}\))
\times P_{rc}(\{\} \mid Cond)
\times P_R(\(STOP_R \mid Cond, \{\}\))

Cond is

mother = \(S^{bought}\), head-dtr is \(VP^{bought}\)
Probabilistic Grammars

Other models and supervised learning

\[ P(S_{\text{bought}}) \]
\[ \times P_H(VP_{\text{bought}} | S_{\text{bought}}) \]
\[ \times P_{lc}(\{NP_A\} | \text{Cond}) \]
\[ \times P_L(NP_{\text{IBM}} | \text{Cond}, \{NP_A\}) \]
\[ \times P_L(NP_{\text{week}} | \text{Cond}, \{\}) \]
\[ \times P_L(STOP_L | \text{Cond}, \{\}) \]
\[ \times P_{rc}(\{\} | \text{Cond}) \]
\[ \times P_R(STOP_R | \text{Cond}, \{\}) \]

\text{Cond is}

mother = \( S^{\text{bought}} \), head-dtr is \( VP^{\text{bought}} \)
$$P(S^{bought})$$
$$\times P_H(VP^{bought}|S^{bought})$$
$$\times P_{lc}([NP_A]|Cond)$$
$$\times P_L(NP^{IBM}_A|Cond, \{NP_A\})$$
$$\times P_L(NP^{week}_A|Cond, \{\})$$
$$\times P_L(STOP_L|Cond, \{\})$$
$$\times P_{rc}(\{\}|Cond)$$
$$\times P_R(STOP_R|Cond, \{\})$$

Cond is

mother = $S^{bought}$, head-dtr is $VP^{bought}$
Probabilistic Grammars

Other models and supervised learning

\[ P(S_{bought}) \times P_H(VP_{bought} | S_{bought}) \times P_{lc}(\{NP_A\} | Cond) \times P_L(NP_{IBM}^{\text{A}} | Cond, \{NP_A\}) \times P_L(NP_{week}^{\text{A}} | Cond, \{\}) \times P_L(STOP_L | Cond, \{\}) \times P_{rc}(\{\} | Cond) \times P_R(STOP_R | Cond, \{\}) \]

Cond is

\[ \text{mother} = S_{bought}, \text{head-dtr is } VP_{bought} \]
Probabilistic Grammars

Other models and supervised learning

\[ P(S^{bought}) \]
\[ \times P_H(VP^{bought}|S^{bought}) \]
\[ \times P_{lc}(\{NP_A\}|Cond) \]
\[ \times P_L(NP_A^{IBM}|Cond, \{NP_A\}) \]
\[ \times P_L(NP_{week}|Cond, \{\}) \]
\[ \times P_L(STOP_L|Cond, \{\}) \]
\[ \times P_{rc}(\{\}|Cond) \]
\[ \times P_R(STOP_R|Cond, \{\}) \]

Cond is

\[ \text{mother} = S^{bought}, \text{ head-dtr is VP^{bought}} \]
Probabilistic Grammars

Other models and supervised learning

\[ P(S_{bought}) \times P_H(VP_{bought} | S_{bought}) \times P_{lc}(\{NP_A\} | Cond) \times P_L(NP_{IBM}^{\text{bought}} | Cond, \{NP_A\}) \times P_L(NP_{week}^{\text{bought}} | Cond, \{\}) \times P_L(STOP_L | Cond, \{\}) \times P_{rc}(\{\} | Cond) \times P_R(STOP_R | Cond, \{\}) \]

Cond is

mother = \( S_{bought} \), head-dtr is \( VP_{bought} \)

this is essentially Collins’ Model 2

<table>
<thead>
<tr>
<th>Klein and Manning 2003</th>
<th>treebank PCFG</th>
<th>F1: 72.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collins 2003</td>
<td>model 1: heads, markovisation</td>
<td>F1: 88</td>
</tr>
<tr>
<td>Collins 2003</td>
<td>model 2: heads, markovisation, subcat</td>
<td>F1: 88.5</td>
</tr>
</tbody>
</table>
Witten-Bell Smoothing

crucial to the high performance are smoothing techniques in the estimation of parameters from the treebank
Witten-Bell Smoothing

crucial to the high performance are smoothing techniques in the estimation of parameters from the treebank

Collins followed the general scheme of estimating $P(A|B)$, by interpolating with estimates of $\hat{p}(A|\Phi_i(B))$ where the $\Phi_i(B)$ are less and less specific versions of $B$

$$\Phi_0(B) = B \subset \Phi_1(B) \ldots \subset \Phi_n(B)$$
Witten-Bell Smoothing

crucial to the high performance are smoothing techniques in the estimation of parameters from the treebank

Collins followed the general scheme of estimating $P(A|B)$, by interpolating with estimates of $\hat{p}(A|\Phi_i(B))$ where the $\Phi_i(B)$ are less and less specific versions of $B$

$$\Phi_0(B) = B \subset \Phi_1(B) \ldots \subset \Phi_n(B)$$

for each $i$ there is a raw estimate $e_i$, based on raw counts

$$e_i = \frac{\text{count}(A, \Phi_i(B))}{\text{count}(\Phi_i(B))}$$
Witten-Bell Smoothing

crucial to the high performance are smoothing techniques in the estimation of parameters from the treebank

Collins followed the general scheme of estimating $P(A|B)$, by interpolating with estimates of $\hat{p}(A|\Phi_i(B))$ where the $\Phi_i(B)$ are less and less specific versions of $B$

$$\Phi_0(B) = B \subset \Phi_1(B) \ldots \subset \Phi_n(B)$$

for each $i$ there is a raw estimate $e_i$, based on raw counts

$$e_i = count(A, \Phi_i(B))/\text{count}(\Phi_i(B))$$

then for each $i$ a smoothed estimate $\tilde{e}_i$ is defined by a weighted combination of raw $e_i$ and smoothed $\tilde{e}_{i+1}$

$$\tilde{e}_i = \lambda_i e_i + (1 - \lambda_i) \tilde{e}_{i+1} \text{ for } 0 \leq i < n - 1$$

$$\tilde{e}_n = e_n$$
weights in Witten-Bell

In the Witten-Bell method the $\lambda_i$ are not constant, but instead are dependent on the particular conditioning event $\Phi_i$.

$$\tilde{e}_i(P(A|B)) = \lambda_i e_i(P(A|B)) + (1 - \lambda_i)(\tilde{e}_{i+1}(P(A|B)))$$

Recall $e_i(P(A|B))$ is based on simple counts referring to the conditioning event $\Phi_i(B)$:

$$e_i = \frac{\text{count}(A, \Phi_i(B))}{\text{count}(\Phi_i(B))} = \frac{\text{count}(A, \Phi_i(B))}{f_i}$$
weights in Witten-Bell

in the Witten-Bell method the $\lambda_i$ are not constant, but instead are dependent on the particular conditioning event $\Phi_i$.

$$\tilde{e}_i(P(A|B)) = \lambda_i e_i(P(A|B)) + (1 - \lambda_i)(\tilde{e}_{i+1}(P(A|B)))$$

recall $e_i(P(A|B))$ is based on simple counts referring to the conditioning event $\Phi_i(B)$:

$$e_i = \frac{\text{count}(A, \Phi_i(B))}{\text{count}(\Phi_i(B))} = \frac{\text{count}(A, \Phi_i(B))}{f_i}$$

let $u_i =$ number of distinct outcomes in the context specified by $\Phi_i(B)$
weights in Witten-Bell

in the Witten-Bell method the $\lambda_i$ are not constant, but instead are dependent on the particular conditioning event $\Phi_i$.

$$\tilde{e}_i(P(A|B)) = \lambda_i e_i(P(A|B)) + (1 - \lambda_i)(\tilde{e}_{i+1}(P(A|B)))$$

recall $e_i(P(A|B))$ is based on simple counts referring to the conditioning event $\Phi_i(B)$:

$$e_i = \frac{\text{count}(A, \Phi_i(B))}{\text{count}(\Phi_i(B))} = \frac{\text{count}(A, \Phi_i(B))}{f_i}$$

let $u_i = $ number of distinct outcomes in the context specified by $\Phi_i(B)$

$\lambda_i$ is then defined

$$\lambda_i = \frac{f_i}{\alpha u_i + f_i}$$

where $\alpha$ is a further constant to be set: Collins used $\alpha = 5$. 
Witten-Bell continued

\[ \lambda_i = \frac{f_i}{\alpha u_i + f_i} \]

\[ \tilde{e}_i(P(A|B)) = \lambda_i e_i(P(A|B)) + (1 - \lambda_i)(\tilde{e}_{i+1}(P(A|B))) \]

- \( f_i \) high (lots of cases of conditioning event \( \Phi_i \)) \( \Rightarrow \) weight is higher
- \( u_i \) high (lots of distinct outcomes other than \( A \)) \( \Rightarrow \) weight is lower
For example, in Collins’ Model 1, a sibling is conditioned on a head-sibling, with head-word, head-tag annotation, and a particular mother category (with head-word, head-tag annotation).

there is raw-estimate just based on counts with exactly the mentioned features:

$$\hat{P}_L^0(NP^{week} | head - dtr = VP^{bought, VBD}, mother = S^{bought, VBD})$$

the final estimate based also on trees which are unspecific for the head-word

$$\hat{P}_L^1(NP^{week} | head - dtr = VP^{VBD}, mother = S^{VBD})$$

and also on trees unspecific for the head-tag:

$$\hat{P}_L^2(NP^{week} | head - dtr = VP, mother = S)$$

final estimate for the desired probability:

$$\lambda_1 \hat{P}^0 + (1 - \lambda_1)(\lambda_2 \hat{P}^1 + (1 - \lambda_2)(\hat{P}^3))$$
Wrapping up

- PCFGs can be seen as a generalisation of HMMs
  - similar best-parse algorithm
  - similar unsupervised training algorithm: the Inside/Outside algorithm another instance of Expectation/Maximisation

- Inside-Outside algorithms also applied in
  - other unsupervised approaches: the *Constituent/Context Model*
  - inferring latent annotation for trees
Wrapping up

- **supervised** models: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (eg. head, argument, adjunct
  - factorises probability in terms of head-outwards processes
Wrapping up

- **supervised** models: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (e.g., head, argument, adjunct)
  - factorises probability in terms of head-outwards processes

- this is not unrepresentative of quite a lot of other people’s supervised models
Wrapping up

- **supervised** models: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (e.g. head, argument, adjunct
  - factorises probability in terms of head-outwards processes

- this is not unrepresentative of quite a lot of other people’s supervised models

- there is by now a very large body of work on supervised learning of parsing models, some of it moving a little further from the ideas discussed today
Wrapping up

- **supervised models**: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (e.g. head, argument, adjunct
  - factorises probability in terms of head-outwards processes

- this is not unrepresentative of quite a lot of other people’s supervised models

- there is by now a very large body of work on supervised learning of parsing models, some of it moving a little further from the ideas discussed today
  - dependency grammars: the topic of one of the course at this summer school
Wrapping up

- **supervised models**: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (e.g., head, argument, adjunct
  - factorises probability in terms of head-outwards processes

- this is not unrepresentative of quite a lot of other people’s supervised models

- there is by now a very large body of work on supervised learning of parsing models, some of it moving a little further from the ideas discussed today
  - dependency grammars: the topic of one of the course at this summer school
  - so-called **Data Oriented Parsing**, inferring tree probabilities by combining observed tree fragments, where these fragments do not necessarily correspond to grammar rules
Wrapping up

- **supervised models**: looked at some details of the parsing models considered by Collins
  - quite high performance
  - use annotations not explicit in the original trees (e.g., head, argument, adjunct)
  - factorises probability in terms of head-outwards processes

- this is not unrepresentative of quite a lot of other people’s supervised models

- there is by now a very large body of work on supervised learning of parsing models, some of it moving a little further from the ideas discussed today
  - dependency grammars: the topic of one of the course at this summer school
  - so-called **Data Oriented Parsing**, inferring tree probabilities by combining observed tree fragments, where these fragments do not necessarily correspond to grammar rules
  - re-ranking approaches, in which a probabilistic parser delivers its $N$-best parses, and then these parses are then re-ordered possibly referring to quite global features which are quite difficult to incorporate directly into the parser.
  - semi-supervised learning, combining annotated and unannotated training data.
Issues, Questions

- performance numbers not kicking on very much: stuck around 90%
  Why?
- performance when changing text type seems to degrade a lot
- what about trace information: lots of approaches pretend it's not there
- are the outputs useful: people work on getting predicate/argument structures out of the parses