Advanced PCFG Models

Sara Stymne

Uppsala University
Department of Linguistics and Philology

Slides mostly from Joakim Nivre

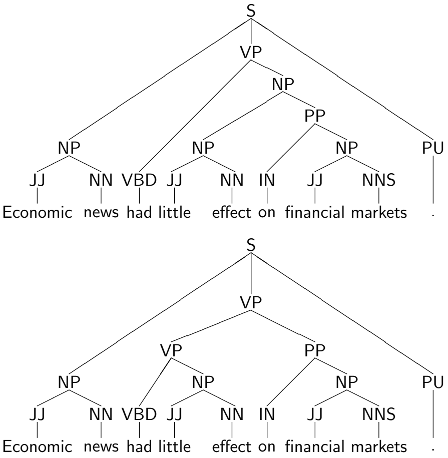
1. Problems with Treebank PCFGs
2. Parent Annotation
3. Lexicalization
4. Markovization
5. Latent Variables
6. Other Parsing Frameworks

Lack of Sensitivity to Structural Context

Tree Context	NP PP	DT NN	PRP
Anywhere	11%	9%	6%
NP under S	9%	9%	21%
NP under VP	23%	7%	4%

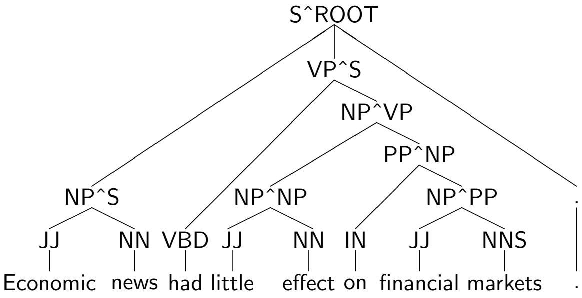
Lack of Sensitivity to Lexical Information

S	→	NP VP PU	1.00
VP	→	VP PP	0.33
VP	→	VBD NP	0.67
NP	→	NP PP	0.14
NP	→	JJ NN	0.57
NP	→	JJ NNS	0.29
PP	→	IN NP	1.00
PU	→	.	1.00
JJ	→	Economic	0.33
JJ	→	little	0.33
JJ	→	financial	0.33
NN	→	news	0.50
NN	→	effect	0.50
NNS	→	markets	1.00
VBD	→	had	1.00
IN	→	on	1.00



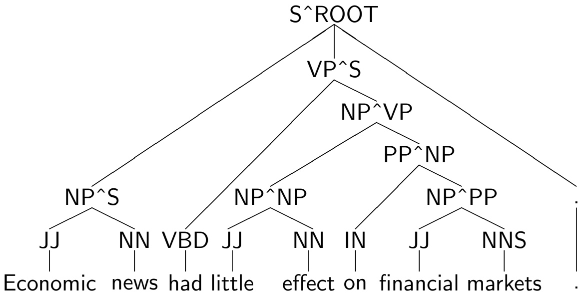
Parent Annotation

Replace nonterminal A with A^*B* when A is child of B.



Parent Annotation

Replace nonterminal A with A^*B* when A is child of B.



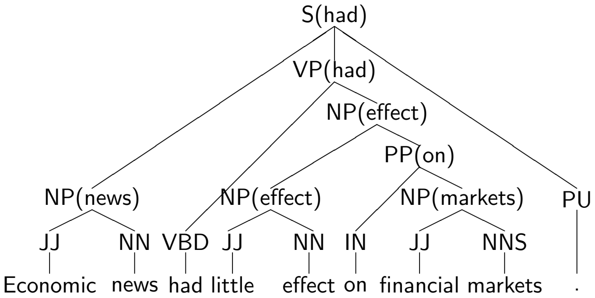
Described in the first seminar article

Lexicalization

Nonterminals: $N_{\text{lex}} = \{A(a) \mid A \in N, a \in \Sigma\}$

Rules: $A(a) \rightarrow \ldots B(a) \ldots$

$A(a) \rightarrow a$



Smoothing of the Lexicalized PCFG

$$
\begin{aligned}
q &= Q(\mathsf{A}(\mathsf{a}) \rightarrow \mathsf{B}(\mathsf{b})\ \mathsf{C}(\mathsf{a})) \\
&= P(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{b} \,|\, \mathsf{A}, \mathsf{a}) \\
&= P(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C} \,|\, \mathsf{A}, \mathsf{a}) \cdot P(\mathsf{b} \,|\, \mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{a})
\end{aligned}
$$

$$
\begin{aligned}
q_1 &= P(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C} \,|\, \mathsf{A}, \mathsf{a}) \\
&\approx \lambda \frac{\textsc{count}(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{a})}{\textsc{count}(\mathsf{A}, \mathsf{a})} + (1-\lambda) \frac{\textsc{count}(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C})}{\textsc{count}(\mathsf{A})}
\end{aligned}
$$

$$
\begin{aligned}
q_2 &= P(\mathsf{b} \,|\, \mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{a}) \\
&\approx \lambda \frac{\textsc{count}(\mathsf{b}, \mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{a})}{\textsc{count}(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C}, \mathsf{a})} + (1-\lambda) \frac{\textsc{count}(\mathsf{b}, \mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C})}{\textsc{count}(\mathsf{A} \rightarrow_2 \mathsf{B}\ \mathsf{C})}
\end{aligned}
$$

Non-lexicalized CKY Parsing

```
PARSE(G, x)
for j from 1 to n do
    for all A : A → x_j ∈ R
        C[j − 1, j, A] := Q(A → x_j)
for j from 2 to n do
    for i from j − 2 downto 0 do
        for k from i + 1 to j − 1 do
            for all A → B C ∈ R andC[i, k, B] > 0 and C[k, j, C] > 0
                if (C[i, j, A] < Q(A → B C) · C[i, k, B] · C[k, j, C]) then
                    C[i, j, A] := Q(A → B C) · C[i, k, B] · C[k, j, C]
                    B[i, j, A] := (k, B, C)
return max_h C[0, n, S], BUILD-TREE(B[0, n, S])
```

Lexicalized CKY Parsing

PARSE(G, x)
for j from 1 to n do
  for all $A: A(x_j) \rightarrow x_j \in R$
    $\mathcal{C}[j-1, j, j, A] := Q(A(x_j) \rightarrow x_j)$
for j from 2 to n do
  for i from $j-2$ downto 0 do
    for k from $i+1$ to $j-1$ do
      for h from $i+1$ to k do
        for m from $k+1$ to j do
          for all $A: A(x_h) \rightarrow B(x_h)C(x_m) \in R$ and $\mathcal{C}[i,k,h,B] > 0$ and $\mathcal{C}[k,j,m,C] > 0$
            if $(\mathcal{C}[i,j,h,A] < Q(A(x_h) \rightarrow B(x_h)C(x_m)) \cdot \mathcal{C}[i,k,h,B] \cdot \mathcal{C}[k,j,m,C])$ then
              $\mathcal{C}[i,j,h,A] := Q(A(x_h) \rightarrow B(x_h)C(x_m)) \cdot \mathcal{C}[i,k,h,B] \cdot \mathcal{C}[k,j,m,C]$
              $\mathcal{B}[i,j,h,A] := (k, B, h, C, m)$
      for h from $k+1$ to j do
        for m from $i+1$ to k do
          for all $A: A(x_h) \rightarrow B(x_m)C(x_h) \in R$ and $\mathcal{C}[i,k,m,B] > 0$ and $\mathcal{C}[k,j,h,C] > 0$
            if $(\mathcal{C}[i,j,m,A] < Q(A(x_h) \rightarrow B(x_m)C(x_h)) \cdot \mathcal{C}[i,k,m,B] \cdot \mathcal{C}[k,j,h,C])$ then
              $\mathcal{C}[i,j,h,A] := Q(A(x_h) \rightarrow B(x_m)C(x_h)) \cdot \mathcal{C}[i,k,m,B] \cdot \mathcal{C}[k,j,h,C]$
              $\mathcal{B}[i,j,h,A] := (k, B, m, C, h)$
return $\max_h \mathcal{C}[0,n,h,S]$, BUILD-TREE($\mathcal{B}[0, n, \text{argmax}_h \mathcal{C}[0,n,h,S], S]$)

Complexity

- Two extra loops in the algorithm, for the head of left and right trees
- Complexity is thus $O(n^5)$ instead of $O(n^3)$
- Too slow for many practical applications
- Pruning techniques often used
  - Means that we do not necessarily find the best tree, even given our model

Binarization

N-ary rule:

$$\text{VP} \rightarrow \text{VB NP PP PP}$$

Exact binarization:

$$\begin{array}{rcl} \text{VP} & \rightarrow & \langle \text{VP:[VB] NP PP PP} \rangle \\ \langle \text{VP:[VB] NP PP PP} \rangle & \rightarrow & \langle \text{VP:[VB] NP PP} \rangle \text{ PP} \\ \langle \text{VP:[VB] NP PP} \rangle & \rightarrow & \langle \text{VP:[VB] NP} \rangle \text{ PP} \\ \langle \text{VP:[VB] NP} \rangle & \rightarrow & \langle \text{VP:[VB]} \rangle \text{ NP} \\ \langle \text{VP:[VB]} \rangle & \rightarrow & \text{VB} \end{array}$$

First-order markovization:

$$\begin{array}{rcl} \text{VP} & \rightarrow & \langle \text{VP:[VB]} \ldots \text{PP} \rangle \\ \langle \text{VP:[VB]} \ldots \text{PP} \rangle & \rightarrow & \langle \text{VP:[VB]} \ldots \text{PP} \rangle \text{ PP} \\ \langle \text{VP:[VB]} \ldots \text{PP} \rangle & \rightarrow & \langle \text{VP:[VB]} \ldots \text{NP} \rangle \text{ PP} \\ \langle \text{VP:[VB]} \ldots \text{NP} \rangle & \rightarrow & \langle \text{VP:[VB]} \rangle \text{ NP} \\ \langle \text{VP:[VB]} \rangle & \rightarrow & \text{VB} \end{array}$$

Latent Variables

- Extract treebank PCFG
- Repeat k times:
  1. Split every nonterminal A into A_1 and A_2 (and duplicate rules)
  2. Train a new PCFG with the split nonterminals using EM
  3. Merge back splits that do not increase likelihood

Some Famous Parsers

	Par	Lex	Mark	Lat
Collins	+	+	+	−
Charniak	+	+	+	−
Stanford	+	−	+	−
Berkeley	+	−	+	+

Other Parsing Frameworks

- Shift-reduce parsing (transition-based)
  - Does not need a chart
  - Greedy
  - Linear time complexity
- Neural networks in parsing
  - Can reduce independence assumptions
  - Typically gives better results
  - Example: Recurrent neural network grammars (RNNG)

Backtrace

Assume that backpointers are lists:
(lh, rh1, rh2, min, mid, max) if binary rule
(pos, word) if preterminal rule

BACKTRACE(bp, bpchart)

if not $defined(bp)$ *then*

return $NONE$

if $length(bp) == 2$ *then*

return $makeList(pos, word)$

else $// \ length(bp) == 6$

return

$makeList(lh, BACKTRACE(bpchart(min, mid, rh1), bpchart)$

$BACKTRACE(bpchart(mid, max, rh2), bpchart))$