Tree-gram Parsing:
Lexical Dependencies
and
Structural Relations

From Sima’an (2002)

May, 7th 2002
What is the paper about?

1. Problem to be solved: Parsing natural language sentences using information from training treebank

2. Strategy: Devise a model that is

**Probabilistic:** assigns probability to every tree $T$ given sentence $S$. Search for $T^* = \arg\max_T P(T|S) = \arg\max_T P(T, S)$, where $P(T, S)$ is estimated from cooccurrence statistics from treebank

**Generative:** obtain tree $T$ in top-down derivations, rewriting start symbol TOP into sentence $S$. Each rewrite-step involves a rewrite-rule. The rewrite-rules in this model are called **Tree-grams**

**More general** than previous models by combining two complementary approaches
Approaches:

1. Lexical Dependencies
   - Main idea: condition probabilities on lexicalized non-terminals and estimate cooccurrence relations between pairs of lexicalized non-terminals in the tree
   - Instance: Head-Lexicalization
     Accomplished by associating each non-terminal in a parse-tree with the head word of the underlying constituent
   - Refs: Charniak (1997), Collins (1997)

2. Structural Relations
   - Main idea: extract cooccurrences of syntactic structures of arbitrary size present in the treebank, including non-terminals and/or terminals
   - Instance: Data-Oriented Parsing
     Rather than deriving a grammar from treebank, the model uses whichever fragments of trees that appear to be useful
Head-lexicalized models

Advantages

1. Capture verb selectional preferences

<table>
<thead>
<tr>
<th>Subcat frames</th>
<th>Verbs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → V NP</td>
<td>come</td>
<td>1.1%</td>
</tr>
<tr>
<td></td>
<td>take</td>
<td>32.1%</td>
</tr>
<tr>
<td>VP → V PP</td>
<td>come</td>
<td>34.5%</td>
</tr>
<tr>
<td></td>
<td>take</td>
<td>3.1%</td>
</tr>
</tbody>
</table>

From Manning and Schütze (2001)

2. Use robustness of Markov grammars: rather than storing explicit rules (V → NP), they store probabilities that allow creating rules on the fly (e.g. \( P(NP|VP) \)). Thus, they can handle sentences whose rules do not show up in the treebank

Disadvantages

1. Do not capture relations between non-head words:

   Her idea was more quickly understood than mine

   where ‘than mine’ depends on the non-head ‘more quickly’ and not on the head-word ‘understood’

2. Rule out relations among more than two words (e.g. constructions, as more ... than, and idioms)
Data-Oriented Parsing models

Advantages

1. Capture multiple word and non-word dependencies
   (a) \[ S \rightarrow NP \rightarrow V \rightarrow NP \]
   Sue \rightarrow saw \rightarrow John
   (b) \[ S \rightarrow NP \rightarrow VP \rightarrow NP \]
   Sue \rightarrow V \rightarrow NP \rightarrow John
   (c) \[ V \rightarrow NP \rightarrow VP \rightarrow NP \]
   saw \rightarrow V \rightarrow NP \rightarrow John

2. More powerful than PCFGs: assign to tree fragments probabilities that cannot be generated as multiplication of the rules in PCFG

Disadvantages

1. Probabilities not always lexicalized (some fragments have only non-terminals). Problem for novel input sentences, when a subcat frame is hypothesized high in the tree without reference to head-word

2. Does not capture all possible joint preferences
   (a) \[ S \rightarrow NP \rightarrow VP \]
   Sue \rightarrow NP \rightarrow John
   (b) \[ VP \rightarrow NP \]
   VP \rightarrow NP \rightarrow John
   (c) \[ VP \rightarrow V \]
   VP \rightarrow V \rightarrow saw
Why combine these approaches?

Lexical Dependencies and Structural Relations extract distinct information from treebanks

1. Head-lexicalization – DOP = relations between pairs of words and lexicalized non-terminals all the way up in the parse-tree

2. DOP – Head-lexicalization = relations among many words, such as idiom chunks (e.g. ‘take advantage of’), not only pairs of words

Proposed new model: Tree-grams

1. More general than Markov Grammars, since Tree-gram rules may be deeper, adding robustness

   (a) \[ S \]
   \[ A \]
   \[ B \]
   \[ C \]
   \[ \ldots \]

   (b) \[ S \]
   \[ A \]
   \[ B \]
   \[ C \]
   \[ \ldots \]

   \[ A_1 \]
   \[ A_2 \]

2. More general than DOP subtree fragments, since Tree-gram fragments can take any format, not only the ones allowed by the definition of DOP subtrees (either all children of the original tree or none)
How to combine both approaches? (1)

1. Pre-head enrichment:

- For every nonterminal $\mu$ (except POS-tags), attach pre-head representing its head-word

- Pre-head of $\mu$ contains:
  a) POS-tag of head-word $\langle 1^{PH} \rangle$;
  b) label of $1^{PH}$ mother’s node $\langle 2^{PH} \rangle$;
  c) subcat frames

- So pre-heads represent partial lexicalization

\[\begin{array}{c}
\text{(a)}\\
\text{S} \\
\text{NP}_\text{Joe} \\
\text{VP} \\
\text{V} \text{eats} \\
\text{NP}_\text{pizza}
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\text{(b)}\\
\text{S}_{VPZ,VP} \\
\text{NP}_{NNP,NP}_\text{Joe} \\
\text{VP}_{V} \text{eats} \\
\text{NP}_{NN,NP}_\text{pizza}
\end{array}\]
How to combine both approaches? (2)

2. Tree-gram extraction:

- For each parse $T$ in the treebank, extract three disjoint Tree-gram sets, called roles:

  **Head-role ($H$):** carries the head-child
  **Left-dependent ($L$):** carries left-dependents
  **Right-dependent ($R$):** carries right-dependents

![Diagram](image)

The role gives the T-gram's contribution to derivation

- Given a T-gram $t$ and a non-terminal $\mu$ such that $\text{root}(t) = \mu$, then $t \in H$ if one of $\mu$'s children contains the head (Head-Child, $H$)

- $t \in L$ if $\mu$'s children contain $H$'s left modifiers
How to combine both approaches? (3)

3. Parse-tree derivation:

• Generates a head-role T-gram and attaches to it left- and right-dependent role T-grams

Algorithm:

\[ \pi = \text{current parse-tree}; \ C_\pi = \text{conditioning history} \]
\[ \mu = \text{non-terminal node}; \ A = \text{label of } \mu; \ t = \text{T-gram} \]
\[ \mathcal{H}_A = \text{subset of } \mathcal{H} \ (t \in \mathcal{H}_A \text{ if } t \in \mathcal{H} \text{ and } \text{root}(t) = A) \]

While not all \( \mu \in \pi \) are complete and not all leaf nodes are terminal symbols, do:

**Head-generation:** select from \( \pi \) a leaf node \( \mu \);

- generate head T-gram \( t \in \mathcal{H}_A \) with \( P_{\mathcal{H}}(t|A, C_\pi) \);
- extend \( \pi \) at \( \mu \) with \( t \)

**Modification:** select from \( \pi \) a leaf node \( \mu \) that is not not complete

**Left:** if \( \mu \) is not left-complete, generate left-dependent T-gram \( t \in \mathcal{L}_A \) with probability \( P_{\mathcal{L}}(t|A, C_\pi) \) and extend \( \pi \) at \( \mu \) with \( t \)

**Right:** mirror case
3.1 Example of derivation:

\[
\begin{align*}
&\text{a. Left T-gram with root } [S] \text{ is generated at node } S \\
&\text{b. Right T-gram with root } [VP] \text{ is generated at node } [VP] \\
&\text{c. All nodes are complete and all leaves are terminals}
\end{align*}
\]

4. Probability estimation:

1. In general, more than one derivation per tree and more than one tree per sentence (NP-hard problem)

2. MPParse approximated by MPDerivation:

- \( P(der) = P(t_1 \circ \ldots \circ t_n) = \prod_i P(t_i); t_i \text{ is T-gram} \)
- \( MPP = P(T, S) = \sum_{der} \prod_i P(t_i) \approx \prod_i P(t_i) = MPD \)
- \( P(t_i) \) is estimated in the T-gram model as:

\[
PX(t_i|A, C_\pi) = \frac{freq(t_i,X_A,C_\pi)}{\sum_{x \in X_A} freq(x,X_A,C_\pi)},
\]

where \( t_i \in X_A \) and \( X_A \in \{H_A, L_A, R_A\} \)
Model instance on Penn treebank (1)

1. Defining the context history ($C_\pi$)

**Adjacency:** flag $F_L(t)$ tells whether left-dependent $t$ (or right-dependent, $F_R(t)$), extracted from $\mu$, dominates string adjacent to $\mu$’s head-word

**Subcat frames:** to every $\mu$ in treebank associate multisets $SC^\mu_L$ and $SC^\mu_R$, representing left/right complement-children of $\mu$

**Markovian generation:** if $\mu$ has empty subcat frames, assume 1st-order Markov process in generating $\mathcal{L}$ and $\mathcal{R}$ around $\mathcal{H}$:

- add $XLM^\mu$ and $XRM^\mu$ (left-, right-most children of node $\mu$) as conditionals to probability

2. Probabilities with $C_\pi$ instance boil down to

- $P_H(t|A, C_\pi) \approx P_H(t|A, P)$
- $P_L(t|A, C_\pi) \approx P_L(t|A, H, SC^\mu_L, F_L(t), XRM^\mu)$
- $P_R(t|A, C_\pi) \approx P_R(t|A, H, SC^\mu_R, F_R(t), XLM^\mu)$
Model instance on Penn treebank (2)

3. Example of probabilities and enriched nodes:

\[
(t_1) \quad [VP^H_{(VBZ, VP, 0, \{NP\})}] \quad (t_2) \quad [VP^R_{(VBZ, VP, 1, \{\})}]
\]

\[
\begin{align*}
&[VBZ] \\
eats \\
\end{align*}
\]

\[
\begin{align*}
&[NP] \\
pizza \\
\end{align*}
\]

\[
P_H(t_1 | A, P) = P_H(t_1 | VP \cup VBZ, VP, 0, \{NP\}, S)
\]

\[
P_R(t_2 | A, H, SC^\mu_R, Fr(t), XLM^\mu) = \\
P_R(t_2 | VP \cup VBZ, VP, 1, \{\}, VBZ, \{\}, 1, eats)
\]

4. Implementation issues:

- \( f \geq 5, \ w \leq 3, \ n \leq 4, \ d \) varies, where \( f \) = frequency threshold, \( w \) = \#words, \( n \) = nonterminal leaves + open-nodes, \( d \) = depth of T-gram

- Unknown words from input and words occurring less than \( f \) times in the treebank were renamed 0/1 + Unknown + suffix

- Input words were tagged with all POS-tags with which they cooccurred in the treebank

- Parser: CKY
Results

Measures: If $P$ is a proposed parse and $T$ is the tree-bank parse, then

- $LR = \frac{\text{#correct constituents in } P}{\text{#constituents in } T}$
- $LP = \frac{\text{#correct constituents in } P}{\text{#constituents in } P}$
  \hspace{1cm} Fscore = \frac{2 \times LP \times LR}{LP + LR}$
- $CB = \text{#constituents in } P \text{ violating boundaries in } T$

Table 1. Comparison across models

<table>
<thead>
<tr>
<th>System</th>
<th>LR%</th>
<th>LP%</th>
<th>CB%</th>
</tr>
</thead>
<tbody>
<tr>
<td>head-lex (Collins, 1997)</td>
<td>88.1</td>
<td>88.6</td>
<td>0.91</td>
</tr>
<tr>
<td>head-lex (Charniak, 1999)</td>
<td>90.1</td>
<td>90.1</td>
<td>0.74</td>
</tr>
<tr>
<td>SCFG (Charniak, 1997)</td>
<td>71.7</td>
<td>75.8</td>
<td>2.03</td>
</tr>
<tr>
<td>T-gram ($d \leq 5$ ($2^{PH}$))</td>
<td>82.9</td>
<td>85.1</td>
<td>1.30</td>
</tr>
</tbody>
</table>

Results on WSJ section 23; sentence length $\leq$ 40 words

Table 2. Comparison across T-gram submodels

<table>
<thead>
<tr>
<th>System</th>
<th>$D_1,2^{PH}$</th>
<th>$D_4,2^{PH}$</th>
<th>$D_5,1^{PH}$</th>
<th>$D_5,0^{PH}$</th>
<th>$D_5,2^{PH}, Mk$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>80.03</td>
<td>82.42</td>
<td>82.85</td>
<td>81.35</td>
<td>82.93</td>
</tr>
<tr>
<td>LP</td>
<td>80.99</td>
<td>85.23</td>
<td>85.06</td>
<td>84.59</td>
<td>85.13</td>
</tr>
<tr>
<td>CB</td>
<td>1.70</td>
<td>1.32</td>
<td>1.43</td>
<td>1.48</td>
<td>1.30</td>
</tr>
<tr>
<td>#sens</td>
<td>2245</td>
<td>first 1000</td>
<td>2245</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Effects of T-gram depth on performance (cols.2,3), of pre-head enrichment (cols 4,5) and of of Markovian generation (col 5)

Table 3. Comparison with DOP model ($Fscores$)

<table>
<thead>
<tr>
<th>System</th>
<th>Tree height threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$D_5,0^{PH}$</td>
<td>81</td>
</tr>
<tr>
<td>$D_5,1^{PH}$</td>
<td>83</td>
</tr>
</tbody>
</table>

T-gram outperforms DOP, specially for deeper trees
Conclusions

1. Tree-grams capture more structural relations than DOP fragments and allow pre-head-driven parsing

2. Tree-grams improved performance of pure DOP model, even though did not outperform full-fledged head-lexicalized models

Future Directions

1. Test T-gram model on treebanks enriched with progressively more lexical information

2. Make use of semantic information (ADV, LOC, TMP) from treebanks