Incrementally Learning an Incremental Parser

Yoav Seginer
yseginer@science.uva.nl

ILLC, Universiteit van Amsterdam

MLCCLA Workshop 2007
Overview

**Question:**
What is the relation between the language surface statistics and its hidden syntactic structure?

To explore this, I will present an unsupervised parser.

Some of the properties of natural language which will be used:

- Tree structures are skewed.
- Humans process language incrementally.
- Words have the Zipfian distribution.
- Bootstrapping in learning.
Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: \( B \) smallest bracket covering \( u \) and \( v \)
\( u \) is a generator of depth \( d \) of \( B \)
\[ \Rightarrow u \xrightarrow{d} v \]
Common Cover Links

\[
\left[ \left[ w \right] \right] \left[ x \right] \left[ y \right] \left[ z \right]_{y,z}
\]

**Bracketing:** non-crossing continuous brackets.

**Generator:** a word of minimal depth under a bracket.

**Link:** $B$ smallest bracket covering $u$ and $v$

\[
\left\{ \begin{array}{c}
\text{if } u \text{ is a generator of depth } d \text{ of } B \\
\Rightarrow u \xrightarrow{d} v
\end{array} \right.
\]
**Bracketing:** non-crossing continuous brackets.

**Generator:** a word of minimal depth under a bracket.

**Link:** $B$ smallest bracket covering $u$ and $v$  
$u$ is a generator of depth $d$ of $B$  
\[ \Rightarrow u \rightarrow v \]
Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: $B$ smallest bracket covering $u$ and $v$ if $u$ is a generator of depth $d$ of $B$.

$\Rightarrow u \overset{d}{\rightarrow} v$
Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: \( B \) smallest bracket covering \( u \) and \( v \)
\( u \) is a generator of depth \( d \) of \( B \) \( \Rightarrow u \xrightarrow{d} v \)
Removing Redundant Links

Representative Subset: May remove links of redundant generators.
Linear Transitivity: Longer links can be deduced from shorter links.

This is a shortest common cover link set.
Removing Redundant Links

Representative Subset: May remove links of redundant generators.

Linear Transitivity: Longer links can be deduced from shorter links.

This is a shortest common cover link set.
Removing Redundant Links

Representative Subset: May remove links of redundant generators.
Linear Transitivity: Longer links can be deduced from shorter links.

This is a shortest common cover link set.
Removing Redundant Links

Representative Subset: May remove links of redundant generators.
Linear Transitivity: Longer links can be deduced from shorter links.

This is a shortest common cover link set.
Common Cover Links vs. Dependencies

Dependency structure:

```
[[ I ]]  [ know ]  [[ the ← boy ]]  [[ sleeps ] ] ] ]
```

Shortest common cover link set:

```
[[ I ]]  [ know ]  [[ the ← boy ]]  [[ sleeps ] ] ] ]
```

Exocentric constructions: Don’t force the parser to decide which word is the head.

Depth 0 or 1: Links of depth 0 and 1 give the correct skewness of trees.

Adjacency: A link is formed from x to y only when y is adjacent to a unit containing x (in other words: every word between x and y is reachable from x).
The Incremental Parser

Parser: utterance $\mapsto$ shortest common cover link set

**Incremental parser**
Reads words one by one.
May only add links between the last word and previous words.

Utterance: $U = \langle x_1, \ldots, x_n \rangle$
Prefix: $U_k = \langle x_1, \ldots, x_k \rangle$
Parser: $P(U_k) = S_k$
Incrementality: $S_{k-1} \subseteq S_k$
Every link in $S_k \setminus S_{k-1}$ has one end at $x_k$.

$S_k$ is always a shortest common cover link set.
The Incremental Parser: Example

Dependency parsing:

\[
\text{[ [ I ] \ know \ [ [ the \leftarrow \ boy ] \ [ sleeps ] ] ] ]}
\]

\[
\text{[ [ I ] \ know \ [ [ the \leftarrow \ boy ] ] ]}
\]

Ambiguous prefix: \[
\text{I \ know \ the \leftarrow \ boy}
\]

With common cover links, the prefix is unambiguous:

\[
\text{[ [ I ] \ know \ [ [ the \leftarrow \ boy ] \ [ sleeps ] ] ] ]}
\]

\[
\text{[ [ I ] \ know \ [ [ the \leftarrow \ boy ] ] ]}
\]

Similar to psycholinguistic models (Weinberg '93,'95; Gorrell '95; Sturt and Crocker '96) based on D-Theory (Marcus et al. '83).
Parsing Algorithm

- Determine which links may be added.
- Use a weight function to select at most one of these.
  - One link selected ⇒ add it.
  - No link selected ⇒ read next word.
- Repeat.

The weight function has to be learned.

The weight function is determined by a lexicon.
Incremental Learning (Bootstrapping)

Learning creates a sequence of lexicons: $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \ldots$.

$\mathcal{L}_0$ is the zero lexicon.

Learning produces $\mathcal{L}_{i+1}$ from $\mathcal{L}_i$:

- Parse an utterance using lexicon $\mathcal{L}_i$.
- Update $\mathcal{L}_i$ based on the parse.
- The result is $\mathcal{L}_{i+1}$.

This happens while parsing, so learning can remain always on.
Labels

A label is based on a word $w$:
- **class label**, $[w]$.
- **adjacency label**, $[w_\text{−}]$ or $[\_w]$.

The lexicon assigns each adjacency point of each word a set of labels with strengths.

The lexicon also assigns properties (e.g. *Stop*) a strength.

<table>
<thead>
<tr>
<th></th>
<th>$A_{−2}$</th>
<th>$A_{−1}$</th>
<th>$A_1$</th>
<th>$A_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Stop$</td>
<td>59459</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>[the]</td>
<td>10673</td>
<td></td>
<td>[the]</td>
<td>16461</td>
</tr>
<tr>
<td>[of_]</td>
<td>6871</td>
<td></td>
<td>[a]</td>
<td>3107</td>
</tr>
<tr>
<td>[in_]</td>
<td>5520</td>
<td></td>
<td>[_.the]</td>
<td>2787</td>
</tr>
<tr>
<td>[a]</td>
<td>3407</td>
<td></td>
<td>[of]</td>
<td>2347</td>
</tr>
<tr>
<td>[for_]</td>
<td>2572</td>
<td></td>
<td>[_.company]</td>
<td>2094</td>
</tr>
<tr>
<td>[to_]</td>
<td>2094</td>
<td></td>
<td>[']</td>
<td>1686</td>
</tr>
<tr>
<td>[on_]</td>
<td>2009</td>
<td></td>
<td>[in]</td>
<td>1388</td>
</tr>
<tr>
<td>[that_]</td>
<td>1495</td>
<td></td>
<td>[_.U.S.]</td>
<td>1199</td>
</tr>
<tr>
<td>[and_]</td>
<td>1489</td>
<td></td>
<td>[and]</td>
<td>1129</td>
</tr>
<tr>
<td>[at_]</td>
<td>1149</td>
<td></td>
<td>[to]</td>
<td>876</td>
</tr>
</tbody>
</table>
Calculating the Labels

Adjacency positions \( (p) \) of \( w \): \[ v \rightarrow w \]

\( p = -2 \quad p = -1 \quad p = 1 \)

\( A_w^p \) is updated by the word \( v \) in the \( p \)th adjacency position of \( w \):

\[
\begin{array}{cccc}
\textbf{v} & \text{\( A_1 \)} & \text{\( A_{-2} \)} & \text{\( w \)} \\
(873) & & & \\
\vdots & \vdots & \vdots & \vdots \\
[x] 154 & \vdots & \vdots & \vdots \\
[y] 87 & \vdots & \vdots & \vdots \\
\end{array}
\]

\( A_{-2}([-x]) += 154/873 \)

\( A_{-2}([-y]) += 87/873 \)

If there is no word in the \( p \)th adjacency point: \( A_w^p(\text{Stop}) += 1 \)

\[
\begin{array}{cccc}
\text{\( A_{-1} \)} & \text{\( A_1 \)} \\
\text{\( \text{Stop} \)} & 59459 & \text{\( \text{Stop} \)} & 8 \\
\text{[the]} & 10673 & \text{[the]} & 16461 \\
\text{[of]} & 6871 & \text{[a]} & 3107 \\
\text{[in]} & 5520 & \text{[.the]} & 2787 \\
\text{[a]} & 3407 & \text{[of]} & 2347 \\
\text{[for]} & 2572 & \text{[._company]} & 2094 \\
\text{[to]} & 2094 & \text{[.'s]} & 1686 \\
\end{array}
\]
Deducing Linking Properties

Two consecutive words: \( u \ v \)

Deducing linking properties of \( A_u^1 \) and \( A_{-1}^v \):

\[
A_{-1}^v = \begin{cases} 
true & \text{If no label in } A_{-1}^v \text{ is stronger than } A_{-1}^v (\text{Stop}). \\
false & \text{Otherwise}
\end{cases}
\]

If \( A_{-1}^v = true \) there is unlikely to be a link from \( v \) to \( u \).

\[
A_u^1(I_{n^*}) += \begin{cases} 
-1 & \text{if } A_{-1}^v \\
+1 & \text{if not } A_{-1}^v \text{ and } A_1^v
\end{cases}
\]

\[
A_1^u(Out) += \bar{A}_{-1}^v(I_{n^*})
\]

\[
A_1^u(In) += \bar{A}_{-1}^v(Out)
\]
Adding Links

The parser greedily adds the link with the largest weight.

$\text{Weight}(x \rightarrow y)$ is based on the $In^*, In, Out$ properties of the best matching label from $x$ to $y$:

---

Using the best matching label solves these problems:

- Should the properties of $x$ or $y$ be used?
- Low frequency words: $Mr. \ [Mr.] \ [Mr.\_]\ Arbuth$
- Ambiguity:
  
  $\text{this}[\text{the}] \ [\text{the}.\_] \ \text{year}$
  
  $\text{this}[\_\text{is}] \ [\text{is}] \ \text{was}$
The Weight Function

\( s = \text{strength of best match} \)

**Best match** \( x_{[w]} [w.] y: \)

- \( \bar{A}_1^w(Out) > 0 \) \hspace{2cm} Weight(\( x \overset{0}{\rightarrow} y \)) = \min(\bar{A}_1^w(Out), s) \\
- \( \bar{A}_1^w(Out) = 0 \) and \( \bar{A}_1^w(In) \leq 0 \) \hspace{2cm} Weight(\( x \overset{0}{\rightarrow} y \)) = s 

**Best match** \( x_{[-w]} [w.] y: \)

- \( \bar{A}_{-1}^w(In) > 0 \) \hspace{2cm} Weight(\( x \overset{d}{\rightarrow} y \)) = \min(\bar{A}_{-1}^w(In), s) \\
- \( \bar{A}_{-1}^w(In) \geq |\bar{A}_{-1}^w(In)| \) \hspace{2cm} Weight(\( x \overset{0}{\rightarrow} y \)) = \min(\bar{A}_{-1}^w(In^*), s) \\
- \( \bar{A}_{-1}^w(In^*, In, Out) \leq 0 \) \hspace{2cm} Weight(\( x \overset{0}{\rightarrow} y \)) = s 

In all other cases, Weight(\( x \overset{d}{\rightarrow} y \)) = 0.
Compared with CCM, DMV+CCM (Klein and Manning 2002, 2004) and U-DOP, UML-DOP (Bod 2006).

<table>
<thead>
<tr>
<th>Model</th>
<th>WSJ10</th>
<th>WSJ40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UP</td>
<td>UR</td>
</tr>
<tr>
<td>Right-branching</td>
<td>55.1</td>
<td>70.0</td>
</tr>
<tr>
<td>Right-branching+punct.</td>
<td>59.1</td>
<td>74.4</td>
</tr>
<tr>
<td>Parsing from POS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCM</td>
<td>64.2</td>
<td>81.6</td>
</tr>
<tr>
<td>DMV+CCM(POS)</td>
<td>69.3</td>
<td>88.0</td>
</tr>
<tr>
<td>U-DOP</td>
<td>70.8</td>
<td>88.2</td>
</tr>
<tr>
<td>UML-DOP</td>
<td></td>
<td>82.9</td>
</tr>
<tr>
<td>Parsing from plain text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMV+CCM(DISTR.)</td>
<td>65.2</td>
<td>82.8</td>
</tr>
<tr>
<td>Incremental</td>
<td>75.6</td>
<td>76.2</td>
</tr>
<tr>
<td>Incremental (right to left)</td>
<td>75.9</td>
<td>72.5</td>
</tr>
</tbody>
</table>
### Experiments (cont.)

<table>
<thead>
<tr>
<th>Model</th>
<th>Negra10</th>
<th>Negra40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UP</td>
<td>UR</td>
</tr>
<tr>
<td>Right-branching</td>
<td>33.9</td>
<td>60.1</td>
</tr>
<tr>
<td>Right-branching+punct.</td>
<td>35.4</td>
<td>62.5</td>
</tr>
<tr>
<td>Parsing from POS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCM</td>
<td>48.1</td>
<td>85.5</td>
</tr>
<tr>
<td>DMV+CCM(POS)</td>
<td>49.6</td>
<td>89.7</td>
</tr>
<tr>
<td>U-DOP</td>
<td>51.2</td>
<td>90.5</td>
</tr>
<tr>
<td>UML-DOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parsing from plain text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incremental</td>
<td>51.0</td>
<td>69.8</td>
</tr>
<tr>
<td>Incremental (right to left)</td>
<td>50.4</td>
<td>68.3</td>
</tr>
</tbody>
</table>

Parsing (all corpora) is at a rate of about 4000 words per second. Learning slows this down by about 20%.
Conclusion

Link based representation:

▶ Skewed syntactic trees.
▶ Incremental parsing.
▶ Parsing decisions and learning at adjacent words.

Learning:

▶ Labels (replace parts-of-speech).
▶ Make use of the Zipfian distribution.
▶ Statistics collected from parses.
▶ Best matching label: from statistics to link selection.

The algorithm is fast and simple.
Thank you