Incrementally Learning an Incremental Parser

Yoav Seginer
yseginer@science.uva.nl

ILLC, Universiteit van Amsterdam

MLCSLA 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.

Common Cover Links

[[w] [x [y z]]]

Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: $\left.\begin{array}{l} B \text{ smallest bracket covering } u \text{ and } v \\ u \text{ is a generator of depth } d \text{ of } B \end{array}\right\} \Rightarrow u \xrightarrow{d} v$

Common Cover Links

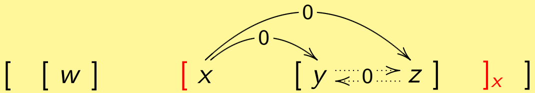
$$[\quad [\, w \,] \qquad [\, x \qquad {\color{red}[}\, y \overset{\longrightarrow}{\longleftarrow} 0 \overset{\longrightarrow}{=} z \,{\color{red}]_{y,z}} \,] \quad]$$

Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: $\left.\begin{array}{l} B \text{ smallest bracket covering } u \text{ and } v \\ u \text{ is a generator of depth } d \text{ of } B \end{array}\right\} \Rightarrow u \xrightarrow{d} v$

Common Cover Links

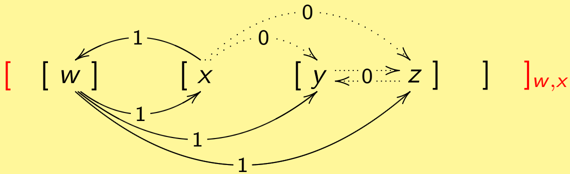


Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: $\left.\begin{array}{l} B \text{ smallest bracket covering } u \text{ and } v \\ u \text{ is a generator of depth } d \text{ of } B \end{array}\right\} \Rightarrow u \xrightarrow{d} v$

Common Cover Links

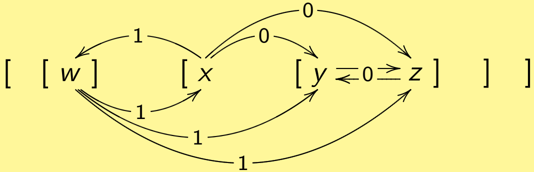


Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

Link: $\left.\begin{array}{l} B \text{ smallest bracket covering } u \text{ and } v \\ u \text{ is a generator of depth } d \text{ of } B \end{array}\right\} \Rightarrow u \xrightarrow{d} v$

Common Cover Links

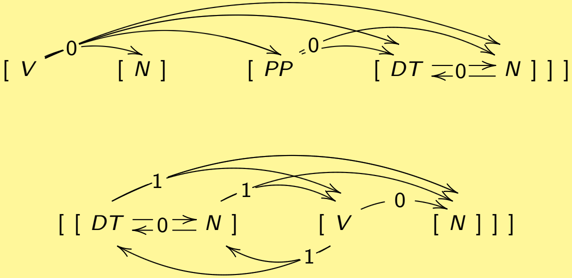


Bracketing: non-crossing continuous brackets.

Generator: a word of minimal depth under a bracket.

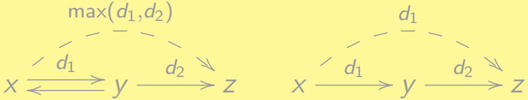
Link: $\left.\begin{array}{l} B \text{ smallest bracket covering } u \text{ and } v \\ u \text{ is a generator of depth } d \text{ of } B \end{array}\right\} \Rightarrow u \xrightarrow{d} v$

Removing Redundant Links



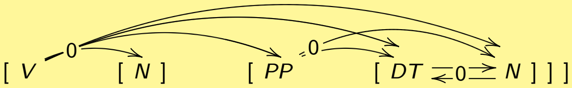
Representative Subset: May remove links of redundant generators.

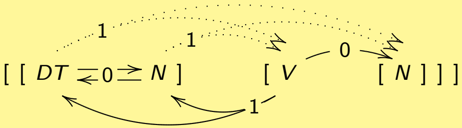
Linear Transitivity: Longer links can be deduced from shorter links.



This is a shortest common cover link set.

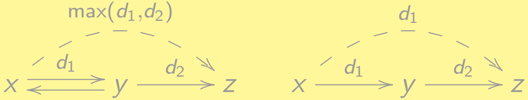
Removing Redundant Links





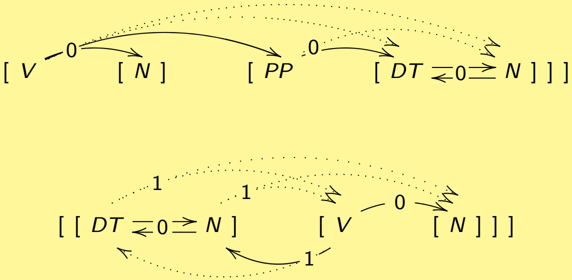
Representative Subset: May remove links of redundant generators.

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



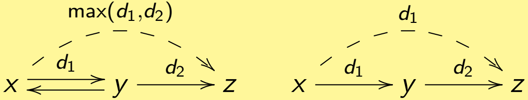
This is a shortest common cover link set.

Removing Redundant Links



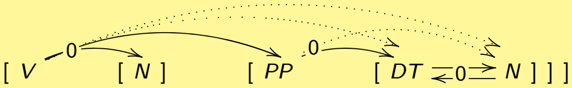
Representative Subset: May remove links of redundant generators.

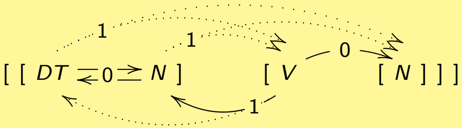
Linear Transitivity: Longer links can be deduced from shorter links.



This is a shortest common cover link set.

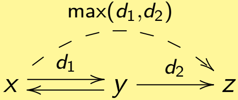
Removing Redundant Links

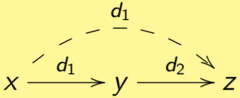




Representative Subset: May remove links of redundant generators.

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 $\leftarrow$ boy] [sleeps]]]]

Shortest common cover link set:

[[I] [know [[the $\overset{0}{\rightleftarrows}$ 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).

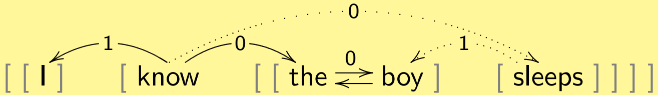
The Incremental Parser

Parser: utterance $\longmapsto$ 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:

[[I] [know [[the ← boy] [sleeps]]]]

[[I] [know [[the ← boy]]]]

??

Ambiguous prefix: I know the ← boy

With common cover links, the prefix is unambiguous:

0

1 0 1

[[I] [know [[the $\overset{0}{\rightleftarrows}$ boy] [sleeps]]]]

1 0

[[I] [know [[the $\overset{0}{\rightleftarrows}$ 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 $\Rightarrow$ add it.
  - No link selected $\Rightarrow$ 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**, $[w\_]$ 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.

the

A_{-2} A_{-1} A_1 A_2

A_{-1}		A_1	
Stop	59459	*Stop*	8
[the]	10673	[the]	16461
[of_]	6871	[a]	3107
[in_]	5520	[_the]	2787
[a]	3407	[of]	2347
[for_]	2572	[_company]	2094
[to_]	2094	['s]	1686
[on_]	2009	[in]	1388
[that_]	1495	[_U.S.]	1199
[and_]	1489	[and]	1129
[at_]	1149	[to]	876

Calculating the Labels

Adjacency positions (p) of w: $v \quad \bullet \leftarrow \bullet \leftarrow w \quad \circ$

$p{=}{-2} \qquad p{=}{-1} \qquad p{=}1$

A^w_p is updated by the word v in the pth adjacency position of w:

v (873) **w**

A_1

⋮	⋮
$[x]$	154
$[\_y]$	87
⋮	⋮

A_{-2}

$A^w_{-2}([v\_]) \mathrel{+}= 1$

$A^w_{-2}([x\_]) \mathrel{+}= 154/873$
$A^w_{-2}([y]) \mathrel{+}= 87/873$

If there is no word in the pth adjacency point: $A^w_p(\textit{Stop}) \mathrel{+}= 1$

the

A_{-1}

Stop	59459
[the]	10673
[of_]	6871
[in_]	5520
[a]	3407
[for_]	2572
[to_]	2094

A_1

Stop	8
[the]	16461
[a]	3107
[_the]	2787
[of]	2347
[_company]	2094
['s]	1686

Deducing Linking Properties

Two consecutive words: $u\ v$

Deducing linking properties of A_1^u and A_{-1}^v:

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

If $_\bullet A_{-1}^v = \mathit{true}$ there is unlikely to be a link from v to u.

$$A_1^u(\mathit{In}^*) \mathrel{+}= \begin{cases} -1 & \text{if } {}_\bullet A_{-1}^v \\ +1 & \text{if not } {}_\bullet A_{-1}^v \text{ and } {}_\bullet A_1^v \end{cases}$$

$$A_1^u(\mathit{Out}) \mathrel{+}= \bar{A}_{-1}^v(\mathit{In}^*)$$

$$A_1^u(\mathit{In}) \mathrel{+}= \bar{A}_{-1}^v(\mathit{Out})$$

Adding Links

The parser greedily adds the link with the largest weight.

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

was

A_{-1}	
Stop	466
[it_]	528
[he_]	225
[is]	181
['s]	161
[was]	128

A_1	
Stop	24
[_the]	438
[_a]	377
[was]	299
[_n't]	237
[is]	204

the

A_{-1}	
Stop	59459
[the]	10673
[of_]	6871
[in_]	5520
[a]	3407
[for_]	2572

A_1	
Stop	8
[the]	16461
[a]	3107
[_the]	2787
[of]	2347
[_company]	2094

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:

$$this_{[the]}\ {}_{[the\_]}year$$

$$this_{[\_is]}\ {}_{[is]}was$$

The Weight Function

$s =$ strength of best match

Best match $x_{[w]\ [w\_]}y$:

$\bar{A}_1^w(Out) > 0$ $\qquad Weight(x \xrightarrow{0} y) = \min(\bar{A}_1^w(Out), s)$

$\bar{A}_1^w(Out) = 0$ and $\bar{A}_1^w(In) \leq 0$ $\qquad Weight(x \xrightarrow{0} y) = s$

Best match $x_{[\_w]\ [w]}y$:

$\bar{A}_{-1}^w(In) > 0$ $\qquad Weight(x \xrightarrow{d} y) = \min(\bar{A}_{-1}^w(In), s)$

$$d = \begin{cases} 1 & \text{if } \bar{A}_{-1}^w(In^*) < 0 \text{ and } \bar{A}_{-1}^w(Out) \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

$\bar{A}_{-1}^w(In^*) \geq \left|\bar{A}_{-1}^w(In)\right|$ $\qquad Weight(x \xrightarrow{0} y) = \min(\bar{A}_{-1}^w(In^*), s)$

$\bar{A}_{-1}^w(In^*, In, Out) \leq 0$ $\qquad Weight(x \xrightarrow{0} y) = s$

In all other cases, $Weight(x \xrightarrow{d} y) = 0$.

Experiments

Compared with CCM, DMV+CCM (Klein and Manning 2002, 2004) and U-DOP, UML-DOP (Bod 2006).

Model	WSJ10			WSJ40		
	UP	UR	UF_1	UP	UR	UF_1
Right-branching	55.1	70.0	61.7	35.4	47.4	40.5
Right-branching+punct.	59.1	74.4	65.8	44.5	57.7	50.2
Parsing from POS						
CCM	64.2	81.6	71.9			
DMV+CCM(POS)	69.3	88.0	77.6			
U-DOP	70.8	88.2	78.5			63.9
UML-DOP			82.9			66.4
Parsing from plain text						
DMV+CCM(DISTR.)	65.2	82.8	72.9			
Incremental	75.6	76.2	75.9	58.9	55.9	57.4
Incremental (right to left)	75.9	72.5	74.2	59.3	52.2	55.6

Experiments (cont.)

Model	Negra10			Negra40		
	UP	UR	UF_1	UP	UR	UF_1
Right-branching	33.9	60.1	43.3	17.6	35.0	23.4
Right-branching+punct.	35.4	62.5	45.2	20.9	40.4	27.6
Parsing from POS						
CCM	48.1	85.5	61.6			
DMV+CCM(POS)	49.6	89.7	63.9			
U-DOP	51.2	90.5	65.4			
UML-DOP			67.0			
Parsing from plain text						
Incremental	51.0	69.8	59.0	34.8	48.9	40.6
Incremental (right to left)	50.4	68.3	58.0	32.9	45.5	38.2

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