2B. Graphical Dependency Parsing

Gerald Penn
Department of Computer Science, University of Toronto

Based on slides by Yuji Matsumoto, Dragomir Radev, David Smith, Sam Thomson and Jason Eisner
Predicting structured outputs

- Log-linear models great for n-way classification
- Also good for predicting sequences
  - CVEs, or, to allow fast dynamic programming, only use n-gram features

- Also good for dependency parsing
  - but to allow fast dynamic programming or MST parsing, only use single-edge features
Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

```
Byl jasný studený dubnový den a hodiny odbíjely třináctou
```

"It was a bright cold day in April and the clocks were striking thirteen"
Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

jasný $\leftarrow$ den ("bright day")

Byl jasný studený dubnový den a hodiny odbíjely třináctou

“It was a bright cold day in April and the clocks were striking thirteen”
Is this a good edge?

jasný $\leftrightarrow$ den
("bright day")

jasný $\leftrightarrow$ N
("bright NOUN")

"It was a bright cold day in April and the clocks were striking thirteen"
“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

"It was a bright cold day in April and the clocks were striking thirteen"
How about this competing edge?

not as good, lots of red ...

“It was a bright cold day in April and the clocks were striking thirteen”
How about this competing edge?

jasný ← hodiny
("bright clocks")

... undertrained ...

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

How about this competing edge?

jasný ← hodiny
(“bright clocks”)
... undertrained ...

jasn ← hodi
(“bright clock,”
stems only)

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

How about this competing edge?

jasný ← hodiny
("bright clocks")

... undertrained ...

jasn ← hodi
("bright clock,”
stems only)

A\text{plural} ← N\text{singular}

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

How about this competing edge?

jasný ← hodiny
A ← N
where N follows a conjunction

jasn ← hodi
(“bright clock,” stems only)

A$_{\text{plural}}$ ← N$_{\text{singular}}$

Byl jasný studený dubnový den a hodiny odbíjely třináctou

V A A A N J N V C

byl jasn stud dubn den a hodi odbí třin

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
  - “bright day” or “bright clocks”?

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
- Score of an edge $e = \theta \cdot \text{features}(e)$
- Standard algos $\rightarrow$ valid parse with max total score

“It was a bright cold day in April and the clocks were striking thirteen”
Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
- Score of an edge \( e = \theta \cdot \text{features}(e) \)
- Standard algos \( \Rightarrow \) valid parse with max total score

can't have both
(one parent per word)

can't have both
(no crossing links)

Thus, an edge may lose (or win) because of a consensus of other edges.

Can't have all three
(no cycles)

our current weight vector
Non-Projective Parses

I'll give a talk tomorrow on bootstrapping

The "projectivity" restriction. Do we really want it?

subtree rooted at "talk"
is a discontiguous noun phrase

can't have both
(no crossing links)
Non-Projective Parses

I’ll give a talk tomorrow on bootstrapping.

That glory may know my going-gray (i.e., it shall last till I go gray).

frequent non-projectivity in Latin, etc.
Non-Projective Parsing Algorithms

- Complexity considerations:
  - Projective (Proj)
  - Non-projective (NonP)

<table>
<thead>
<tr>
<th>Problem/Algorithm</th>
<th>Proj</th>
<th>NonP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete grammar parsing</td>
<td>(P)</td>
<td>(NP) hard</td>
</tr>
<tr>
<td>[Gaifman 1965, Neuhaus and Bröker 1997]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic parsing</td>
<td>(O(n))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>[Nivre 2003, Covington 2001]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First order spanning tree</td>
<td>(O(n^3))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>[McDonald et al. 2005b]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(N)th order spanning tree ((N &gt; 1))</td>
<td>(P)</td>
<td>(NP) hard</td>
</tr>
<tr>
<td>[McDonald and Pereira 2006]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
McDonald’s Approach (non-projective)

- Consider the sentence “John saw Mary” (left).
- The Chu-Liu-Edmonds algorithm finds the maximum-weight spanning tree (right) – may be non-projective.
- Can be found in time $O(n^2)$.

Every node selects best parent
If cycles, contract them and repeat
For each non-ROOT node \( v \), set \( \text{bestInEdge}[v] \) to be its highest scoring incoming edge.

If a cycle \( C \) is formed:

- contract the nodes in \( C \) into a new node \( v_C \)
- edges outgoing from any node in \( C \) now get source \( v_C \)
- edges incoming to any node in \( C \) now get destination \( v_C \)
- For each node \( u \) in \( C \), and for each edge \( e \) incoming to \( u \) from outside of \( C \):
  - add to \( e.\text{kicksOut} \) the edge \( \text{bestInEdge}[u] \), and
  - set \( e.\text{score} \) to be \( e.\text{score} - e.\text{kicksOut}.\text{score} \).

Repeat until every non-ROOT node has an incoming edge and no cycles are formed.
An Example - Contracting Stage

<table>
<thead>
<tr>
<th>bestInEdge</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kicksOut</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

<table>
<thead>
<tr>
<th></th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>g</td>
</tr>
<tr>
<td>V2</td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

![Diagram of a graph with labeled edges and vertices.]

<table>
<thead>
<tr>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
</tr>
<tr>
<td>g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>i</td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

**bestInEdge**

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**kicksOut**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

(ROOT)

V4

V3

\(\text{a}: -5\)
\(\text{b}: -10\)
\(\text{c}: 1\)
\(\text{f}: 5\)
\(\text{i}: -3\)
\(\text{e}: 4\)
\(\text{h}: -1\)

- \(\text{bestInEdge}\):
  - V1: g
  - V2: d
  - V3: 
  - V4: 

- \(\text{kicksOut}\):
  - a: g
d
  - b: d
g
  - c: 
d
  - d: 
  - e: 
g
d
  - f: 
g
  - g: 
d
  - h: 
g
  - i: 
d
An Example - Contracting Stage

Graph:

- ROOT
  - a: -5
  - b: -10
  - c: 1

- V4
  - f: 5
  - e: 4
  - h: -1

- V3
  - g
  - d

Tables:

### bestInEdge

<table>
<thead>
<tr>
<th></th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>g</td>
</tr>
<tr>
<td>V2</td>
<td>d</td>
</tr>
<tr>
<td>V3</td>
<td>f</td>
</tr>
<tr>
<td>V4</td>
<td></td>
</tr>
</tbody>
</table>

### kicksOut

<table>
<thead>
<tr>
<th></th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
</tr>
</tbody>
</table>


An Example - Contracting Stage

**Diagram:**
- **ROOT**
- **V4**
- **V3**

- **Edges with Weights:**
  - $a : -5$
  - $b : -10$
  - $c : 1$
  - $d$
  - $e : 4$
  - $f : 5$
  - $g$
  - $h : -1$
  - $i : -3$

**Table:**

<table>
<thead>
<tr>
<th>V1</th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V2</th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V3</th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V4</th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

**Table:**

<table>
<thead>
<tr>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>i</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

```
<table>
<thead>
<tr>
<th></th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>g</td>
</tr>
<tr>
<td>V2</td>
<td>d</td>
</tr>
<tr>
<td>V3</td>
<td>f</td>
</tr>
<tr>
<td>V4</td>
<td>h</td>
</tr>
<tr>
<td>V5</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g, h</td>
</tr>
<tr>
<td>b</td>
<td>d, h</td>
</tr>
<tr>
<td>c</td>
<td>f</td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>d</td>
</tr>
</tbody>
</table>
```
An Example - Contracting Stage

**Diagram:**
- **ROOT** connected to **V5** via edges with weights:
  - \(a: -4\)
  - \(b: -9\)
  - \(c: -4\)

**Table:**

<table>
<thead>
<tr>
<th>bestInEdge</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>g</td>
<td>d</td>
<td>f</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kicksOut</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>g, h</td>
<td>d, h</td>
<td>f</td>
<td>f</td>
<td>g</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Example - Contracting Stage

![Diagram of a graph with nodes and edges labeled with values]

<table>
<thead>
<tr>
<th>Node</th>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>g</td>
</tr>
<tr>
<td>V2</td>
<td>d</td>
</tr>
<tr>
<td>V3</td>
<td>f</td>
</tr>
<tr>
<td>V4</td>
<td>h</td>
</tr>
<tr>
<td>V5</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g, h</td>
</tr>
<tr>
<td>b</td>
<td>d, h</td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>f</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>g</td>
</tr>
<tr>
<td>i</td>
<td>d</td>
</tr>
</tbody>
</table>
After the contracting stage, every contracted node will have exactly one `bestInEdge`. This edge will kick out one edge inside the contracted node, breaking the cycle.

- Go through each `bestInEdge` e in the reverse order that we added them
- lock down e, and remove every edge in `kicksOut(e)` from `bestInEdge`. 
An Example - Expanding Stage

b: $-9$
a: $-4$
c: $-4$

V1
V2
V3
V4
V5

<table>
<thead>
<tr>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td>V5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>i</td>
</tr>
</tbody>
</table>
An Example - Expanding Stage

<table>
<thead>
<tr>
<th>bestInEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td>V5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>i</td>
</tr>
</tbody>
</table>
An Example - Expanding Stage

<table>
<thead>
<tr>
<th></th>
<th>bestInEdge</th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>a</td>
<td>g, h</td>
</tr>
<tr>
<td>V2</td>
<td>d</td>
<td>d, h</td>
</tr>
<tr>
<td>V3</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>V4</td>
<td>a, h</td>
<td></td>
</tr>
<tr>
<td>V5</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

- **ROOT**: $a: -5$, $b: -10$, $c: 1$
- **V4**: $f: 5$, $i: -3$, $e: 4$, $h: -1$
- **V3**: Connected by the edges mentioned in the table.
An Example - Expanding Stage

<table>
<thead>
<tr>
<th>Vertex</th>
<th>bestInEdge</th>
<th>kicksOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>a, g</td>
<td></td>
</tr>
<tr>
<td>V2</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>V4</td>
<td>a, h</td>
<td>g, h</td>
</tr>
<tr>
<td>V5</td>
<td>a</td>
<td>d</td>
</tr>
</tbody>
</table>

Edges and Weights:
- a: -5
- b: -10
- c: 1
- d
- e: 4
- f: 5
- g
- h: -1
- i: -3
An Example - Expanding Stage

**bestInEdge**

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
<td>d</td>
<td>f</td>
<td></td>
<td>a</td>
</tr>
</tbody>
</table>

**kicksOut**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g, h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>d, h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td>g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Example - Expanding Stage

**bestInEdge**

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
<td>d</td>
<td>f</td>
<td>a</td>
<td>h</td>
</tr>
</tbody>
</table>

**kicksOut**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g, h</td>
<td>d, h</td>
<td>f</td>
<td></td>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Consider the sentence “John saw Mary (left)”.  
The Chu-Liu-Edmonds algorithm finds the maximum-weight spanning tree – may be non-projective.  
Can be found in time $O(n^2)$.  

How about total weight $Z$ of all trees?  
Can be found in time $O(n^3)$ by matrix determinants and inverses (Smith & Smith, 2007).
Graph Theory to the Rescue!

Tes's Matrix-Tree Theorem (1948)

The determinant of the Kirchoff (aka Laplacian) adjacency matrix of directed graph $G$ without row and column $r$ is equal to the sum of scores of all directed spanning trees of $G$ rooted at node $r$.

$O(n^3)$ time!

Exactly the $Z$ we need!
Building the Kirchoff (Laplacian) Matrix

\[
\begin{bmatrix}
-\sum_{j \neq 1} s(1, j) & -s(2,1) & \cdots & -s(n,1) \\
-s(1,2) & \sum_{j \neq 2} s(2, j) & \cdots & -s(n,2) \\
\vdots & \vdots & \ddots & \vdots \\
-s(1,n) & -s(2,n) & \cdots & \sum_{j \neq n} s(n, j)
\end{bmatrix}
\]

- Negate edge scores
- Sum columns (children)
- Strike root row/col.
- Take determinant

N.B.: This allows multiple children of root, but see Koo et al. 2007.
Graph Deletion & Contraction

Important fact: $\kappa(G) = \kappa(G-e) + \kappa(G\backslash\{e\})$
Why Should This Work?

Clear for 1x1 matrix; use induction

\[
\begin{align*}
\sum_{j \neq 1} s(1, j) & - s(2, 1) \quad \cdots \quad - s(n, 1) \\
-s(1, 2) & \sum_{j \neq 2} s(2, j) \quad \cdots \quad - s(n, 2) \\
\vdots & \quad \vdots \quad \vdots \quad \vdots \\
-s(1, n) & - s(2, n) \quad \cdots \quad \sum_{j \neq n} s(n, j)
\end{align*}
\]

\(K' = K\) with contracted edge 1, 2

\(K'' = K\) with deleted edge 1, 2

\(|K| = s(1, 2)|K'| + |K''|

Chu-Liu-Edmonds analogy:
Every node selects best parent
If cycles, contract and recurse

Undirected case; special root cases for directed