10A. Log-Likelihood Dependency Parsing

Gerald Penn
Department of Computer Science, University of Toronto

Based on slides by Yuji Matsumoto, Dragomir Radev, David Smith and Jason Eisner
Word Dependency Parsing

**Raw sentence**
He reckons the current account deficit will narrow to only 1.8 billion in September.

**POS-tagged sentence**
He reckons the current account deficit will narrow to only 1.8 billion in September.
PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

**Word dependency parsed sentence**
He reckons the current account deficit will narrow to only 1.8 billion in September .

slide adapted from Yuji Matsumoto
Shift-Reduce Type Algorithms

- Data structures:
  - Stack $[\ldots, w_i]_S$ of partially processed tokens
  - Queue $[w_j, \ldots]_Q$ of remaining input tokens

- Parsing actions built from atomic actions:
  - Adding arcs ($w_i \rightarrow w_j$, $w_i \leftarrow w_j$)
  - Stack and queue operations

- Left-to-right parsing in $O(n)$ time

- Restricted to projective dependency graphs
Yamada’s Algorithm

- Three parsing actions:
  - **Shift**
    \[
    \begin{array}{c}
    \ldots \ldots \ldots S \quad \ldots \ldots \ldots Q \\
    \ldots \ldots w_i s \quad \ldots \ldots Q
    \end{array}
    \]
  - **Left**
    \[
    \begin{array}{c}
    \ldots \ldots w_i, w_j s \quad \ldots \ldots Q \\
    \ldots \ldots w_i s \quad \ldots \ldots Q \\
    w_i \rightarrow w_j
    \end{array}
    \]
  - **Right**
    \[
    \begin{array}{c}
    \ldots \ldots w_i, w_j s \quad \ldots \ldots Q \\
    \ldots \ldots w_j s \quad \ldots \ldots Q \\
    w_i \leftarrow w_j
    \end{array}
    \]

- **Algorithm variants:**
  - Originally developed for Japanese (strictly head-final) with only the **Shift** and **Right** actions [Kudo and Matsumoto 2002]
  - Adapted for English (with mixed headedness) by adding the **Left** action [Yamada and Matsumoto 2003]
  - Multiple passes over the input give time complexity \(O(n^2)\)
Nivre’s Algorithm

Four parsing actions:

- **Shift**
  \[
  \begin{array}{c}
  \text{[\ldots]s} \\
  \downarrow
  \end{array}
  \begin{array}{c}
  \text{[w_i, \ldots]Q} \\
  \text{[\ldots, w_i]s} \\
  \text{[\ldots]Q}
  \end{array}
  \]

- **Reduce**
  \[
  \begin{array}{c}
  \text{[\ldots, w_i]s} \\
  \downarrow
  \end{array}
  \begin{array}{c}
  \text{[\ldots]Q} \\
  \exists w_k : w_k \rightarrow w_i
  \end{array}
  \]

- **Left-Arc**
  \[
  \begin{array}{c}
  \text{[\ldots, w_i]s} \\
  \downarrow
  \end{array}
  \begin{array}{c}
  \text{[w_j, \ldots]Q} \\
  \text{[\ldots]s} \\
  \exists w_k : w_k \rightarrow w_i
  \end{array}
  \]

- **Right-Arc**
  \[
  \begin{array}{c}
  \text{[\ldots, w_i]s} \\
  \downarrow
  \end{array}
  \begin{array}{c}
  \text{[w_j, \ldots]Q} \\
  \text{[\ldots]s} \\
  \exists w_k : w_k \rightarrow w_j
  \end{array}
  \]

Characteristics:

- Integrated labeled dependency parsing
- Arc-eager processing of right-dependents
- Single pass over the input gives time complexity \( O(n) \)
Example

[root]_s [Economic news had little effect on financial markets .]_Q
Example

[\textcolor{red}{\text{root \ Economic}}][S \ \text{news had little effect on financial markets} .]_Q

Shift
Example

\[ \text{[root]}_s \quad \text{Economic} \quad \text{[news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad .]_Q \]

Left-Arc_{nmod}
Example

[\text{root} \ Economic \ news]_S [had \ little \ effect \ on \ financial \ markets \ .]_Q 

Shift
Example

[Sbj [root]s Economic news [had little effect on financial markets .]Q

Left-Arc_{Sbj}
Example

[root Economic news had]$_S$ [little effect on financial markets .]$_Q$

Right-Arc$_{pred}$

Economic news had [little effect on financial markets .]
Example

[\textbf{root} Economic news had little]_S [effect on financial markets .]_Q

Shift
Example

Economic news had little effect on financial markets.

Left-Arc$_{nmod}$
Example

"Economic news had little effect on financial markets."
Example

[Economic news had little effect on financial markets.]

Right-Arc_{nmod}
Economic news had little effect on financial markets.
Example

```
Economic news had little effect on financial markets.
```

Left-Arc$_{nmod}$
Example

Right-Arc$_{pc}$
Example

Economic news had little effect on financial markets.

Reduce
Example

Economic news had little effect on financial markets.

Reduce
Economic news had little effect on financial markets.

Example

Reduce
Economic news had little effect on financial markets.
Example

Economic news had little effect on financial markets.

Right-$\text{Arc}_{p}$
Classifier-Based Parsing

- Data-driven deterministic parsing:
  - Deterministic parsing requires an oracle.
  - An oracle can be approximated by a classifier.
  - A classifier can be trained using treebank data.

- Learning methods:
  - Support vector machines (SVM)
  - Memory-based learning (MBL)
    [Nivre et al. 2004, Nivre and Scholz 2004]
  - Maximum entropy modeling (MaxEnt)
    [Cheng et al. 2005]
Feature Models

Learning problem:
- Approximate a function from parser states, represented by feature vectors to parser actions, given a training set of gold standard derivations.

Typical features:
- Tokens:
  - Target tokens
  - Linear context (neighbors in S and Q)
  - Structural context (parents, children, siblings in G)
- Attributes:
  - Word form (and lemma)
  - Part-of-speech (and morpho-syntactic features)
  - Dependency type (if labeled)
  - Distance (between target tokens)
Great ideas in NLP: Log-linear models
(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.
  \[ p(A) \times p(B \mid A) \times p(C \mid A, B) \times p(D \mid A, B, C) \times \ldots \]
  each choice depends on a limited part of the history

  *but which dependencies to allow?*  \[ p(D \mid A, B, C) \]?
  *what if they're all worthwhile?*  \[ p(D \mid A, B, C) \]?

  \[ \ldots p(D \mid A, B) \times p(C \mid A, B, D) \]?
Great ideas in NLP: Log-linear models
(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.
  
  \[ p(A) \times p(B \mid A) \times p(C \mid A,B) \times p(D \mid A,B,C) \times \ldots \]

  which dependencies to allow? (given limited training data)

- Solution: Log-linear (max-entropy) modeling
  
  \[
  \frac{1}{Z} \times \Phi(A) \times \Phi(B,A) \times \Phi(C,A) \times \Phi(C,B) \\
  \times \Phi(D,A,B) \times \Phi(D,B,C) \times \Phi(D,A,C) \times \ldots
  \]

  ...throw them all in!

  - Features may interact in arbitrary ways
  - **Iterative scaling** keeps adjusting the feature weights until the model agrees with the training data.
How about structured outputs?

- Log-linear models great for n-way classification
- Also good for predicting sequences
  
  ![Diagram of word tags](image)
  
  but to allow fast dynamic programming, only use n-gram features

- Also good for dependency parsing
  
  ![Diagram of preferred links](image)
  
  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?

Yes, lots of green ...

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?

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

jasný ← den
("bright day")

jasný ← 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"
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)

- How about this competing edge?

```
It was a bright cold day in April and the clocks were striking thirteen
```

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

V A A A N J N V C
```
Edge-Factored Parsers (McDonald et al. 2005)

How about this competing edge?

jasný ← hodiny
("bright clocks")
... undertrained ...

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

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

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

A_{plural} ← N_{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”
how about this competing edge?

jasný ← hodiny

A ← N
where N follows a conjunction

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

Aplural ← Nsingular

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)
- Can’t have all three (no cycles)

Thus, an edge may lose (or win) because of a consensus of other edges.
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming

The cat in the hat wore a stovepipe. ROOT

let’s vertically stretch this graph drawing

each subtree is a linguistic constituent (here a noun phrase)
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
    - to score “cat ← wore” link, not enough to know this is NP
    - must know it’s rooted at “cat”
    - so expand nonterminal set by $O(n)$: \{NP\text{\_the}, \text{NP\_cat}, \text{NP\_hat}, \ldots\}
    - so CKY’s “grammar constant” is no longer constant 😞

Each subtree is a linguistic constituent (here a noun phrase)
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
  - Solution: Use a different decomposition (Eisner 1996)
    - Back to $O(n^3)$

Each subtree is a linguistic constituent (here a noun phrase)
Spans vs. constituents

Two kinds of substring.

» **Constituent** of the tree: links to the rest only through its headword (root).

『The cat in the hat wore a stovepipe. ROOT』

» **Span** of the tree: links to the rest only through its endwords.

『The cat in the hat wore a stovepipe. ROOT』
Decomposing a tree into spans

The cat in the hat wore a stovepipe. ROOT

The cat + cat in the hat wore a stovepipe. ROOT

cat in the hat wore + wore a stovepipe. ROOT

cat in + in the hat wore

in the hat + hat wore
Hard Constraints on Valid Trees

- 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)
- Can’t have all three (no cycles)

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

ROOT: I 'll give a talk tomorrow on bootstrapping

subtree rooted at “talk” is a discontiguous noun phrase

can't have both (no crossing links)

The “projectivity” restriction. Do we really want it?
Non-Projective Parses

I’ll give a talk tomorrow on bootstrapping

occasional non-projectivity in English

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

slide thanks to Dragomir Radev
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 $O(n^2)$.

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!

Tutte’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 \( r \).

Exactly the \( Z \) we need!

\( O(n^3) \) time!
Building the Kirchoff (Laplacian) Matrix

\[
\sum_{j \neq 1} s(1, j) \quad -s(2,1) \quad \cdots \quad -s(n,1) \\
-s(1,2) \quad \sum_{j \neq 2} s(2, j) \quad \cdots \quad -s(n,2) \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
-s(1,n) \quad -s(2,n) \quad \cdots \quad \sum_{j \neq n} s(n, j)
\]

- 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.
Why Should This Work?

Clear for 1x1 matrix; use induction

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

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

\( K'' \equiv K(\{1,2\} \mid \{1,2\}) \)

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

Chu-Liu-Edmonds analogy: Every node selects best parent

If cycles, contract and recur

Undirected case; special root cases for directed
Graph Deletion & Contraction

Important fact: $\kappa(G) = \kappa(G-e) + \kappa(G\backslash e)$