Applications of Tree Automata Theory
Lecture II: Parsing — Basics and Evaluation

Andreas Maletti

Institute of Computer Science
Universität Leipzig, Germany

on leave from: Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Yekaterinburg — August 23, 2014
Roadmap

1. Theory of Tree Automata
2. Parsing — Basics and Evaluation
3. Parsing — Advanced Topics
5. Theory of Tree Transducers
6. Machine Translation — Advanced Topics

Always ask questions right away!
**Problem [intuitive]**

**Parsing** is the process of analyzing the syntactic structure of a sentence yielding the **parse tree**

- syntactic structure of a language linguistically motivated
- ideally expressed by a grammar
We must bear in mind the Community as a whole
И вы действительно сможете изменить свою жизнь, воспользовавшись принципом 80 к 20
Linguistic Background

Constituency Parsing

- each word is a **lexical item**
- each lexical item has a grammatical function
  → **part-of-speech** (POS) (noun, verb, etc.)
- they combine to phrases that form syntactic categories
  → **constituents** (noun phrase, verb phrase, etc.)
Constituency Parsing

- each word is a **lexical item**
- each lexical item has a grammatical function → **part-of-speech (POS)** (noun, verb, etc.)
- they combine to phrases that form syntactic categories → **constituents** (noun phrase, verb phrase, etc.)

Other linguistic theories

- dependency grammars
- combinatory categorial grammars
- ...
We must bear in mind the Community as a whole.
## Problem [realistic]

- Assume a hidden function $g : Q^* \rightarrow T_\Sigma(Q)$
- Given a finite set $T \subseteq T_\Sigma(Q)$ generated by $g$
- Develop a system representing $f : Q^* \rightarrow T_\Sigma(Q)$ approximating $g$
Problem [realistic]

- assume a hidden \( g : Q^* \rightarrow T_\Sigma(Q) \) reference parser
- given a finite set \( T \subseteq T_\Sigma(Q) \) example parse trees
  generated by \( g \)
- develop a system representing \( f : Q^* \rightarrow T_\Sigma(Q) \) parser
  approximating \( g \)
Problem [realistic]

- assume a hidden \( g : Q^* \rightarrow T_\Sigma(Q) \) reference parser
- given a finite set \( T \subseteq T_\Sigma(Q) \) example parse trees
generated by \( g \)
- develop a system representing \( f : Q^* \rightarrow T_\Sigma(Q) \) parser
  approximating \( g \)

Clarification

- \( T \) generated by \( g \) \iff \( T = g(L) \) for some finite \( L \subseteq Q^* \)
- for approximation we could use \( |\{ w \in Q^* \mid f(w) = g(w) \}| \)
Training
Observations

- Linguistic knowledge encoded in training set \( T \subseteq T_\Sigma(Q) \)
- We want to build grammars that can generate \( T' \supseteq T \)
- We follow the historical development
  1. Local tree grammars (LTG)
  2. Tree substitution grammars (TSG)
  3. Regular tree grammars (RTG)
LTG Production Extraction
simply read of CFG productions:

\[
\begin{align*}
S & \rightarrow NP \ VP \ NP \\
NP & \rightarrow PRP$ \ NN \\
PRP$ & \rightarrow My \\
NN & \rightarrow dog \\
VP & \rightarrow VBZ \\
VBZ & \rightarrow sleeps \\
NP & \rightarrow PRP \ PRP \\
PRP & \rightarrow I \\
VP & \rightarrow VBD \ ADVP \\
VBD & \rightarrow scored \\
ADVP & \rightarrow RB \\
RB & \rightarrow well
\end{align*}
\]
Statistical Parser Training

LTG Production Extraction
simply read of CFG productions:

S → NP VP
PRP$ → My
VP → VBZ
NP → PRP
VP → VBD ADVP
ADVP → RB

NP → PRP$ NN
NN → dog
VBZ → sleeps
PRP → I
VBD → scored
RB → well
### Observations

- LTG offer unique explanation on tree level
- but ambiguity on the string level
- → weighted productions
Observations

- LTG offer unique explanation on tree level
- but ambiguity on the string level
  → weighted productions

Illustration

The diagrams show the syntactic structures for the sentences:

1. We saw her duck
2. S-BAR structure
Definition

A **weighted** local tree grammar (wLTG) is a weighted CFG $G = (N, Q, S, P, \text{wt})$

- finite set $N$
- finite set $Q$
- $S \subseteq N$
- finite set $P \subseteq N \times T_N(Q)$
- mapping $\text{wt}: P \to [0, 1]$

It will compute the weighted derivation trees of the wCFG.
LTG Production Extraction

simply read of CFG productions and keep counts:

- **S → NP VP** (2)
- **PRP$ → My** (1)
- **VP → VBZ** (1)
- **NP → PRP** (1)
- **VP → VBD ADVP** (1)
- **ADVP → RB** (1)
- **NP → PRP$ NN** (1)
- **NN → dog** (1)
- **VBZ → sleeps** (1)
- **PRP → I** (1)
- **VBD → scored** (1)
- **RB → well** (1)
LTG Production Extraction

normalize counts: (here by left-hand side)

\[
\begin{align*}
S & \rightarrow \text{NP VP } (2) \\
\text{NP} & \rightarrow \text{PRP$ NN } (1) \\
\text{PRP$} & \rightarrow \text{My } (1) \\
\text{NN} & \rightarrow \text{dog } (1) \\
\text{VP} & \rightarrow \text{VBZ } (1) \\
\text{VBZ} & \rightarrow \text{sleeps } (1) \\
\text{PRP} & \rightarrow \text{I } (1) \\
\text{VBD} & \rightarrow \text{scored } (1) \\
\text{ADVP} & \rightarrow \text{RB } (1) \\
\text{RB} & \rightarrow \text{well } (1) \\
\text{NP} & \rightarrow \text{PRP } (1)
\end{align*}
\]
LTG Production Extraction

normalize counts: (here by left-hand side)

\[
\begin{align*}
S & \xrightarrow{1} \text{NP VP} \\
\text{NP} & \xrightarrow{0.5} \text{PRP$ NN} \quad \text{NP} \xrightarrow{0.5} \text{PRP} \\
\text{PRP$} & \xrightarrow{1} \text{My} \\
\text{NN} & \xrightarrow{1} \text{dog} \\
\text{VP} & \xrightarrow{0.5} \text{VBZ} \quad \text{VP} \xrightarrow{0.5} \text{VBD ADVP} \\
\text{VBZ} & \xrightarrow{1} \text{sleeps} \\
\text{PRP} & \xrightarrow{1} \text{I} \\
\text{VBD} & \xrightarrow{1} \text{scored} \\
\text{ADVP} & \xrightarrow{1} \text{RB} \\
\text{RB} & \xrightarrow{1} \text{well}
\end{align*}
\]
Weighted parses

```
S
  NP
    PRP$ My
    NN dog
  VP
    VBZ sleeps

S
  NP
    PRP I
    VBD scored
  VP
    ADVP RB
       well
```

weight: 0.25  weight: 0.25

Weighted LTG productions

(only productions with weight \(\neq 1\))

- \(NP \xrightarrow{0.5} \text{PRP$ NN}\)
- \(VP \xrightarrow{0.5} \text{VBZ}\)
- \(NP \xrightarrow{0.5} \text{PRP}\)
- \(VP \xrightarrow{0.5} \text{VBD ADVP}\)
Statistical Parsing Approach

given sentence $w$, return highest-scoring parse for $w$
Statistical Parsing Approach

given sentence \( w \), return highest-scoring parse for \( w \)

Consequence

The first parse should be preferred

("duck" more frequently a noun, etc.)
### Observations

- works similarly for TSG
- needs additional cutting strategy to select fragments
Observations

- works similarly for TSG
- needs additional cutting strategy to select fragments

Illustration

```
We saw her duck
```

```
We saw PRP$ duck
```
BERKELEY parser [Reference]:

CHARNIAK-JOHNSON parser:
Definition (ParseEval measures)

- **precision** = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse
Definition (ParseEval measures)

- **precision** = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse

- **recall** = number of correct constituents divided by number of all constituents in reference
Definition (ParseEval measures)

- **precision** = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse
- **recall** = number of correct constituents divided by number of all constituents in reference
- **combined measure**

\[ F_{\alpha} = (1 + \alpha^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\alpha^2 \cdot \text{precision} + \text{recall}} \]
We must bear in mind the Community as a whole.

- **precision** = \( \frac{9}{9} \) = 100%
Parser output

- precision $= \frac{9}{9} = 100\%$
- recall $= \frac{9}{10} = 90\%$
Reference

Parser output

- precision = \( \frac{9}{9} = 100\% \)
- recall = \( \frac{9}{10} = 90\% \)
- \( F_1 = 2 \cdot \frac{1 \cdot 0.9}{1 + 0.9} = 95\% \)
Standardized Setup

- **training data**: PENN treebank Sections 2–21
  (articles from the WALL STREET JOURNAL)
- **development test data**: PENN treebank Section 22
- **evaluation data**: PENN treebank Section 23
### Experiment [POST, GILDEA, 2009]

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
<th>precision</th>
<th>recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG</td>
<td>46k</td>
<td>75.37</td>
<td>70.05</td>
<td>72.61</td>
</tr>
<tr>
<td>“spinal” TSG</td>
<td>190k</td>
<td>80.30</td>
<td>78.10</td>
<td>79.18</td>
</tr>
<tr>
<td>“minimal subset” TSG</td>
<td>2,560k</td>
<td>76.40</td>
<td>78.29</td>
<td>77.33</td>
</tr>
</tbody>
</table>

These are bad compared to the state-of-the-art!
## Experiment [POST, GILDEA, 2009]

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG</td>
<td>46k</td>
<td>75.37</td>
<td>70.05</td>
<td>72.61</td>
</tr>
<tr>
<td>“spinal” TSG</td>
<td>190k</td>
<td>80.30</td>
<td>78.10</td>
<td>79.18</td>
</tr>
<tr>
<td>“minimal subset” TSG</td>
<td>2,560k</td>
<td>76.40</td>
<td>78.29</td>
<td>77.33</td>
</tr>
</tbody>
</table>

These are bad compared to the state-of-the-art!
State-of-the-Art Models
## State-of-the-art models

- context-free grammars with latent variables ($\text{CFG}_{\text{lv}}$)
  - [Collins, 1999](#)
  - [Klein, Manning, 2003](#)
  - [Petrov, Klein, 2007](#)

- tree substitution grammars with latent variables ($\text{TSG}_{\text{lv}}$)
  - [Shindo et al., 2012](#)

- other models
Grammars with Latent Variables

**Definition**

A grammar with latent variables is (grammar with relabeling)

- a grammar $G$ generating $L(G) \subseteq T^\Sigma(Q)$
- a (total) mapping $\rho : \Sigma \rightarrow \Delta$ functional relabeling

**Remark**

We use $X-n$ for symbols that are relabeled to $X$ $\rho(X-n) = X$
Grammars with Latent Variables

**Definition (Semantics)**

\[ L(G, \rho) = \rho(L(G)) = \{ \rho(t) \mid t \in L(G) \} \]

Language class: \( \text{REL}(\mathcal{L}) \) for language class \( \mathcal{L} \)
Grammars with Latent Variables

**Definition (Semantics)**

\[ L(G, \rho) = \rho(L(G)) = \{ \rho(t) \mid t \in L(G) \} \]

Language class: \( REL(\mathcal{L}) \) for language class \( \mathcal{L} \)

**Example (Typical fragments)**

![Diagram of grammatical structures]
Grammars with Latent Variables

Definition (Semantics)

\[ L(G, \rho) = \rho(L(G)) = \{ \rho(t) \mid t \in L(G) \} \]

Language class: \( \text{REL}(\mathcal{L}) \) for language class \( \mathcal{L} \)

Example (Typical fragments)

\[
\begin{align*}
S & \quad \text{NP} \quad \text{VP} \\
\text{NP} & \quad \text{PRP} \\
S & \quad \text{NP} \quad \text{VP} \\
\text{VBP} & \quad \text{love} \\
S & \quad \text{NP} \quad \text{VP} \\
\text{TO} & \quad \text{VP}
\end{align*}
\]
Theorem

$$\text{REL}(\text{LTL}) = \text{REL}(\text{TSL}) = \text{REL}(\text{RTL}) = \text{RTL}$$
Grammars with Latent Variables

Theorem

\[ \text{REL}(\text{LTL}) = \text{REL}(\text{TSL}) = \text{REL}(\text{RTL}) = \text{RTL} \]
Grammars with Latent Variables

**Experiment [SHINDO et al., 2012]**

| Grammar model                      | $F_1$ | \(|w| \leq 40\) | full |
|-----------------------------------|-------|-----------------|------|
| TSG [Post, Gildea, 2009]         | 82.6  |                 |      |
| TSG [COHN et al., 2010]          | 85.4  | 84.7            |      |
| CFG$_{lv}$ [Collins, 1999]       | 88.6  | 88.2            |      |
| CFG$_{lv}$ [Petrov, Klein, 2007] | 90.6  | 90.1            |      |
| CFG$_{lv}$ [Petrov, 2010]        |       |                 | 91.8 |
| TSG$_{lv}$ (single) [SHINDO et al., 2012] | 91.6  | 91.1            |      |
| TSG$_{lv}$ (multiple) [SHINDO et al., 2012] | 92.9  | 92.4            |      |

**Discriminative Parsers**

| CARRERAS et al., 2008             |       | 91.1            |      |
| CHARNIAK, JOHNSON, 2005           | 92.0  | 91.4            |      |
| HUANG, 2008                       | 92.3  | 91.7            |      |
Training of TA
Example (English grammar)

\[
\begin{align*}
S-1 & \rightarrow \text{ADJP-2 } S-1 & 0.0035453455987323125 \cdot 10^0 \\
S-1 & \rightarrow \text{ADJP-1 } S-1 & 2.108608433271444 \cdot 10^{-6} \\
S-1 & \rightarrow \text{VP-5 } \text{VP-3} & 1.6367163259885093 \cdot 10^{-4} \\
S-2 & \rightarrow \text{VP-5 } \text{VP-3} & 9.724998692152419 \cdot 10^{-8} \\
S-1 & \rightarrow \text{PP-7 } \text{VP-0} & 1.0686659961009547 \cdot 10^{-5} \\
S-9 & \rightarrow \text{" } \text{NP-3} & 0.012551243773149695 \cdot 10^0
\end{align*}
\]
**Example (English grammar)**

<table>
<thead>
<tr>
<th>Rule</th>
<th>0.0035453455987323125 · 10^0</th>
<th>2.108608433271444 · 10^{-6}</th>
<th>1.6367163259885093 · 10^{-4}</th>
<th>9.724998692152419 · 10^{-8}</th>
<th>1.0686659961009547 · 10^{-5}</th>
<th>0.012551243773149695 · 10^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-1 → ADJP-2 S-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-1 → ADJP-1 S-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-1 → VP-5 VP-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-2 → VP-5 VP-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-1 → PP-7 VP-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-9 → “ NP-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

~~weighted tree automaton~~
**Illustration**

**BERKELEY** parser production:

\[ S-1 \rightarrow \text{ADJP-2} \ S-1 \quad 0.0035453455987323125 \cdot 10^0 \]

This corresponds to tree automata production:

\[ S-1 \rightarrow S(\text{ADJP-2}, \ S-1) \quad 0.0035453455987323125 \cdot 10^0 \]
Training Approach

1. extract wCFG productions
   \[ S \xrightarrow{c} \text{NP VP} \text{ corresponds to } S-1 \xrightarrow{c} S(\text{NP-1, VP-1}) \]

2. split all states and retrain
Training Approach

1. extract wCFG productions
   \[ S \xrightarrow{c} NP \text{ VP corresponds to } S^{-1} \xrightarrow{c} S(NP^{-1}, VP^{-1}) \]

2. split all states and retrain

3. check utility of splits
## Excursion: Berkeley Parser

### Training Approach

1. Extract wCFG productions
   
   $S \xrightarrow{c} \text{NP VP}$ corresponds to $S-1 \xrightarrow{c} S(\text{NP-1, VP-1})$

2. Split all states and retrain

3. Check utility of splits

4. Remerge if split not beneficial

---

Lecture II: Parsing
Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions
   \[ S \xrightarrow{c} NP \ VP \] corresponds to \[ S-1 \xrightarrow{c} S(NP-1, VP-1) \]

2. split all states and retrain

3. check utility of splits

4. remerge if split not beneficial

5. back to 2 unless converged
Training Approach

1. extract wCFG productions
   \[ S \xrightarrow{c} NP \text{ VP corresponds to } S-1 \xrightarrow{c} S(NP-1, \text{ VP-1}) \]
2. split all states and retrain
3. check utility of splits
4. remerge if split not beneficial
5. back to 2 unless converged
Excursion: Berkeley Parser

State splitting

assume $n$ states

- replace each production $X-i \xrightarrow{c} \sigma(Y-j, Z-\ell)$ by

1. $X-i \xrightarrow{c_1} \sigma(Y-j, Z-\ell)$
2. $X-i \xrightarrow{c_2} \sigma(Y-j, Z-(n + \ell))$
3. $X-i \xrightarrow{c_3} \sigma(Y-(n + j), Z-\ell)$
4. $X-i \xrightarrow{c_4} \sigma(Y-(n + j), Z-(n + \ell))$
5. $X-(n+i) \xrightarrow{c_5} \sigma(Y-j, Z-\ell)$
6. $X-(n+i) \xrightarrow{c_6} \sigma(Y-j, Z-(n + \ell))$
7. $X-(n+i) \xrightarrow{c_7} \sigma(Y-(n + j), Z-\ell)$
8. $X-(n+i) \xrightarrow{c_8} \sigma(Y-(n + j), Z-(n + \ell))$
Excursion: Berkeley Parser

Training the weights

- EM algorithm [Dempster, Laird, Rubin, 1977] (Expectation-Maximization)
- we present an inefficient version
  efficient version builds on inside & outside weights
Excursion: Berkeley Parser

Expectation Step

- derivation tree = element of $L(G)$ for the LTG $G$ (trees before relabeling)
- for each tree $t \in T$ of the training set $T$
  build and score all derivation trees $d$ such that $\rho(d) = t$

Given the current model, predict the latent variables
### Expectation Step

<table>
<thead>
<tr>
<th></th>
<th>S $\overset{0.25}{\rightarrow}$ NP-1 VP</th>
<th>S $\overset{0.75}{\rightarrow}$ NP-2 VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>$\overset{1}{\rightarrow}$ sleeps</td>
<td>$\overset{1}{\rightarrow}$ dragon</td>
</tr>
<tr>
<td>NP-1</td>
<td>$\overset{1}{\rightarrow}$ DT-1 NN</td>
<td>$\overset{1}{\rightarrow}$ DT-2 NN</td>
</tr>
<tr>
<td>DT-1</td>
<td>$\overset{0.9}{\rightarrow}$ the</td>
<td>$\overset{0.2}{\rightarrow}$ the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DT-2 $\overset{0.8}{\rightarrow}$ a</td>
</tr>
</tbody>
</table>
Excursion: Berkeley Parser

**Expectation Step**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
<th>Parse Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>S ( \rightarrow ) NP-1 VP</td>
<td>0.25</td>
<td>S ( \rightarrow ) NP-2 VP</td>
</tr>
<tr>
<td>VP ( \rightarrow ) sleeps</td>
<td>1</td>
<td>NN ( \rightarrow ) dragon</td>
</tr>
<tr>
<td>NP-1 ( \rightarrow ) DT-1 NN</td>
<td>1</td>
<td>NP-2 ( \rightarrow ) DT-2 NN</td>
</tr>
<tr>
<td>DT-1 ( \rightarrow ) the</td>
<td>0.9</td>
<td>DT-2 ( \rightarrow ) the</td>
</tr>
<tr>
<td>DT-2 ( \rightarrow ) a</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

**Training set**

- S \( \rightarrow \) NP VP \( \rightarrow \) DT NN \( \rightarrow \) dragon \( \rightarrow \) sleeps
- S \( \rightarrow \) NP VP \( \rightarrow \) DT a \( \rightarrow \) dragon \( \rightarrow \) sleeps
Excursion: Berkeley Parser

Expectation Step

<table>
<thead>
<tr>
<th>Rule</th>
<th>Weight</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S $\xrightarrow{0.25}$ NP-1 VP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VP $\xrightarrow{1}$ sleeps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP-1 $\xrightarrow{1}$ DT-1 NN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT-1 $\xrightarrow{0.9}$ the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S $\xrightarrow{0.75}$ NP-2 VP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN $\xrightarrow{1}$ dragon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP-2 $\xrightarrow{1}$ DT-2 NN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT-2 $\xrightarrow{0.2}$ the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT-2 $\xrightarrow{0.8}$ a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Derivation trees

- Weight: $0.25 \cdot 0.9$
  - $S \rightarrow NP-1 VP \rightarrow VP \rightarrow sleeps \rightarrow NP-1 \rightarrow DT-1 NN \rightarrow DT-1 \rightarrow the$
  - $S \rightarrow NP-1 VN \rightarrow NP-2 \rightarrow NN \rightarrow dragon \rightarrow VP \rightarrow sleeps \rightarrow DT-2 \rightarrow the \rightarrow DT-2 \rightarrow a$

- Weight: $0.75 \cdot 0.2$
  - $S \rightarrow NP-2 VP \rightarrow VP \rightarrow sleeps \rightarrow DT-2 \rightarrow NN \rightarrow dragon \rightarrow VP \rightarrow sleeps \rightarrow DT-2 \rightarrow the \rightarrow DT-2 \rightarrow a$

- Weight: $0.75 \cdot 0.8$
  - $S \rightarrow NP-2 VN \rightarrow NP-2 \rightarrow NN \rightarrow dragon \rightarrow VP \rightarrow sleeps \rightarrow DT-2 \rightarrow a \rightarrow DT-2 \rightarrow dragon$
Maximization Step

- weighted count $c(\rho)$ of occurrences of each production $\rho$ (each occurrence weighted by derivation weight)
- reset production weights

$$
wt'(X-i \rightarrow Y-j \ Z-\ell) = \frac{c(X-i \rightarrow Y-j \ Z-\ell)}{\sum_{j',\ell'} c(X-i \rightarrow Y-j' \ Z-\ell')}
$$

Re-estimate the model parameter given the predictions
Excursion: Berkeley Parser

Derivation trees

Maximization Step

- \( c(S \rightarrow \text{NP-1 VP}) = 0.25 \cdot 0.9 = 0.225 \)
- \( c(S \rightarrow \text{NP-2 VP}) = 0.75 \cdot 0.2 + 0.75 \cdot 0.8 = 0.75 \)
Excursion: Berkeley Parser

Derivation trees

weight: 0.25 \cdot 0.9

S
  NP-1               VP
    DT-1   NN    sleeps
       the    dragon

weight: 0.75 \cdot 0.2

S
  NP-2               VP
    DT-2   NN    sleeps
       the    dragon

weight: 0.75 \cdot 0.8

S
  NP-2               VP
    DT-2   NN    sleeps
       a    dragon

Maximization Step

- \( c(S \rightarrow NP-1 \ VP) = 0.25 \cdot 0.9 = 0.225 \)
- \( c(S \rightarrow NP-2 \ VP) = 0.75 \cdot 0.2 + 0.75 \cdot 0.8 = 0.75 \)
- \( wt'(S \rightarrow NP-1 \ VP) = \frac{0.225}{0.225+0.75} = 0.23 \)
- \( wt'(S \rightarrow NP-2 \ VP) = \frac{0.75}{0.225+0.75} = 0.77 \)
### Training Approach

1. **extract wCFG productions**
   
   \[ S \xrightarrow{c} \text{NP VP} \text{ corresponds to } S-1 \xrightarrow{c} S(\text{NP-1}, \text{VP-1}) \]

2. **split all states and retrain**

3. **check utility of splits**
   
   (evaluate on development test)

4. **remerge if split not beneficial**

5. **back to 2**
   
   (unless converged)
### Training Approach

1. Extract wCFG productions
   
   \[ S \stackrel{c}{\rightarrow} \text{NP VP} \] corresponds to \[ S \stackrel{c}{\rightarrow} S(\text{NP-1, VP-1}) \]

2. Split all states and retrain

3. Check utility of splits
   
   (evaluate on development test)

4. Remerge if split not beneficial

5. Back to 2 (unless converged)
Determiner splits:

Figure taken from [PETROV et al, 2006]
### Excursion: Berkeley Parser

**Figure taken from [PETROV et al, 2006]**

<table>
<thead>
<tr>
<th>VBZ</th>
<th>DT</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>gives</td>
<td>the</td>
<td>In</td>
</tr>
<tr>
<td>sells</td>
<td>The</td>
<td>With</td>
</tr>
<tr>
<td>takes</td>
<td>a</td>
<td>After</td>
</tr>
<tr>
<td>comess</td>
<td>A</td>
<td>For</td>
</tr>
<tr>
<td>goes</td>
<td>An</td>
<td>At</td>
</tr>
<tr>
<td>works</td>
<td>Another</td>
<td></td>
</tr>
<tr>
<td>includes</td>
<td>The</td>
<td>in</td>
</tr>
<tr>
<td>owns</td>
<td>No</td>
<td>for</td>
</tr>
<tr>
<td>is</td>
<td>This</td>
<td>on</td>
</tr>
<tr>
<td>puts</td>
<td>Some</td>
<td>of</td>
</tr>
<tr>
<td>provides</td>
<td>These</td>
<td>for</td>
</tr>
<tr>
<td>takes</td>
<td>all</td>
<td>on</td>
</tr>
<tr>
<td>says</td>
<td>those</td>
<td>with</td>
</tr>
<tr>
<td>adds</td>
<td>some</td>
<td></td>
</tr>
<tr>
<td>Says</td>
<td>these</td>
<td></td>
</tr>
<tr>
<td>believes</td>
<td>some</td>
<td>both</td>
</tr>
<tr>
<td>means</td>
<td>these</td>
<td></td>
</tr>
<tr>
<td>thinks</td>
<td>that</td>
<td></td>
</tr>
<tr>
<td>expects</td>
<td>That</td>
<td>this</td>
</tr>
<tr>
<td>makes</td>
<td>This</td>
<td>each</td>
</tr>
<tr>
<td>calls</td>
<td>each</td>
<td></td>
</tr>
<tr>
<td>plans</td>
<td>The</td>
<td>a</td>
</tr>
<tr>
<td>expects</td>
<td>these</td>
<td></td>
</tr>
<tr>
<td>wants</td>
<td>that</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>the</td>
<td></td>
</tr>
<tr>
<td>‘s</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>remains</td>
<td>some</td>
<td></td>
</tr>
<tr>
<td>has</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>‘s</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>does</td>
<td>Is</td>
<td>this</td>
</tr>
<tr>
<td>Does</td>
<td></td>
<td>the</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NNP</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jr. Goldman INC.</td>
<td>1 100</td>
</tr>
<tr>
<td>Bush Noriega Peters</td>
<td>8,50 15 1.2</td>
</tr>
<tr>
<td>J. E. L.</td>
<td>8 10 20</td>
</tr>
<tr>
<td>York Francisco Street</td>
<td>1 30 31</td>
</tr>
<tr>
<td>Stock Exchange York</td>
<td>two three five</td>
</tr>
<tr>
<td>Corp. Inc. Group</td>
<td>one One Three</td>
</tr>
<tr>
<td>Congress Japan IBM</td>
<td>12 34 14</td>
</tr>
<tr>
<td>Friday September August</td>
<td>78 58 34</td>
</tr>
<tr>
<td>Shearon D. Ford</td>
<td>one two three</td>
</tr>
<tr>
<td>U.S. Treasury Senate</td>
<td>million billion trillion</td>
</tr>
<tr>
<td>John Robert James</td>
<td>PRP</td>
</tr>
<tr>
<td>Mr. Ms. President</td>
<td>PRP-0 It He I</td>
</tr>
<tr>
<td>Oct. Nov. Sept.</td>
<td>PRP-1 it he they</td>
</tr>
<tr>
<td>New San Wall</td>
<td>PRP-2 it them him</td>
</tr>
<tr>
<td>JJS</td>
<td>RBR</td>
</tr>
<tr>
<td>largest</td>
<td>further</td>
</tr>
<tr>
<td>latest</td>
<td>lower</td>
</tr>
<tr>
<td>biggest</td>
<td>higher</td>
</tr>
<tr>
<td>least</td>
<td>most</td>
</tr>
<tr>
<td>best</td>
<td>Most</td>
</tr>
<tr>
<td>worst</td>
<td>least</td>
</tr>
<tr>
<td>PRP</td>
<td>RBR-0</td>
</tr>
<tr>
<td>RBR-1</td>
<td>It</td>
</tr>
<tr>
<td>RBR-2</td>
<td>he</td>
</tr>
<tr>
<td>RBR-0</td>
<td>them</td>
</tr>
<tr>
<td>RBR-1</td>
<td>them</td>
</tr>
<tr>
<td>RBR-2</td>
<td>him</td>
</tr>
<tr>
<td>RBR-0</td>
<td>earlier</td>
</tr>
<tr>
<td>RBR-1</td>
<td>Earlier</td>
</tr>
<tr>
<td>RBR-2</td>
<td>later</td>
</tr>
<tr>
<td>RBR-0</td>
<td>n’t</td>
</tr>
<tr>
<td>RBR-1</td>
<td>not</td>
</tr>
</tbody>
</table>
Summary

- good source of (relevant) weighted tree automata
- large automata
  - **English**: 153 MB (1,133 states and 4,267,277 productions)
  - **Chinese**: 98 MB
- many operations still to be investigated
  (determinization, minimization, products, etc.)
Summary

- good source of (relevant) weighted tree automata
- large automata
  - English: 153 MB (1,133 states and 4,267,277 productions)
  - Chinese: 98 MB
- many operations still to be investigated
  (determinization, minimization, products, etc.)

(Nondeterministic) Minimization

- unweighted: PSpace (NFA) and ExpTime (TA)
- weighted over field:
  PTime (wNFA) and randomized PTime (wTA)
Articles

**Klein, Manning**
*Accurate Unlexicalized Parsing*  
Proc. 41st ACL, 2003

**Petrov, Barrett, Thibaux and Klein**
*Learning Accurate, Compact, and Interpretable Tree Annotation*  
Proc. 44th ACL, 2006

**Shindo, Miyao, Fujino and Nagata**
*Bayesian Symbol-Refined Tree Substitution Grammars for Syntactic Parsing*  
Proc. 50th ACL, 2012