CSE 517
Natural Language Processing
Winter 2017

Parsing (Trees)

Yejin Choi - University of Washington

[Slides from Dan Klein, Michael Collins, Luke Zettlemoyer and Ray Mooney]
Topics

- Parse Trees
- (Probabilistic) Context Free Grammars
  - Supervised learning
  - Parsing: most likely tree, marginal distributions
- Treebank Parsing (English, edited text)
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
### Penn Treebank Non-terminals

**Table 1.2.** The Penn Treebank syntactic tagset

<table>
<thead>
<tr>
<th>Non-terminal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJP</td>
<td>Adjective phrase</td>
</tr>
<tr>
<td>ADVP</td>
<td>Adverb phrase</td>
</tr>
<tr>
<td>NP</td>
<td>Noun phrase</td>
</tr>
<tr>
<td>PP</td>
<td>Prepositional phrase</td>
</tr>
<tr>
<td>S</td>
<td>Simple declarative clause</td>
</tr>
<tr>
<td>SBAR</td>
<td>Subordinate clause</td>
</tr>
<tr>
<td>SBARQ</td>
<td>Direct question introduced by <em>wh</em>-element</td>
</tr>
<tr>
<td>SINV</td>
<td>Declarative sentence with subject-aux inversion</td>
</tr>
<tr>
<td>SQ</td>
<td>Yes/no questions and subconstituent of SBARQ excluding <em>wh</em>-element</td>
</tr>
<tr>
<td>VP</td>
<td>Verb phrase</td>
</tr>
<tr>
<td>WHADVP</td>
<td>Wh-adverb phrase</td>
</tr>
<tr>
<td>WHNP</td>
<td>Wh-noun phrase</td>
</tr>
<tr>
<td>WHPP</td>
<td>Wh-prepositional phrase</td>
</tr>
<tr>
<td>X</td>
<td>Constituent of unknown or uncertain category</td>
</tr>
<tr>
<td>*</td>
<td>“Understood” subject of infinitive or imperative</td>
</tr>
<tr>
<td>0</td>
<td>Zero variant of <em>that</em> in subordinate clauses</td>
</tr>
<tr>
<td>T</td>
<td>Trace of <em>wh</em>-Constituent</td>
</tr>
</tbody>
</table>
The Penn Treebank: Size

- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:

Canadian Utilities had 1988 revenue of C$ 1.16 billion, mainly from its natural gas and electric utility businesses in Alberta, where the company serves about 800,000 customers.
Phrase Structure Parsing

- Phrase structure parsing organizes syntax into \textit{constituents} or \textit{brackets}.
- In general, this involves nested trees.
- Linguists can, and do, argue about details.
- Lots of ambiguity.
- Not the only kind of syntax…

\textbf{Example:}

\begin{verbatim}
new art critics write reviews with computers.
\end{verbatim}
Constituency Tests

- How do we know what nodes go in the tree?

- Classic constituency tests:
  - Substitution by proform
    - he, she, it, they, ...
  - Question / answer
  - Deletion
  - Movement / dislocation
  - Conjunction / coordination

- Cross-linguistic arguments, too
Conflicting Tests

- Constituency isn’t always clear
  - Units of transfer:
    - think about ~ penser à
    - talk about ~ hablar de
  - Phonological reduction:
    - I will go → I’ll go
    - I want to go → I wanna go
    - a le centre → au centre

- Coordination
  - He went to and came from the store.

La vélocité des ondes sismiques
Classical NLP: Parsing in 70s/80s

- Write symbolic or logical rules:
  
  Grammar (CFG)        | Lexicon
  ---------------------|-------------------
  ROOT → S             | NN → interest     
  S → NP VP            | NNS → raises      
  NP → DT NN           | VBP → interest    
  NP → NN NNS          | VBZ → raises      
  VP → VBP NP PP       |                  
  PP → IN NP           |                  

- Use deduction systems to prove parses from words
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses

- This scaled very badly, but was a popular approach in the 70’s and 80’s before corpora were available.
- Didn’t yield broad-coverage tools.
I shot [an elephant] [in my pajamas]
Attachment Ambiguity

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for $27 a share]
[at its monthly meeting].
Syntactic Ambiguities I

- Prepositional phrases:
  *They cooked the beans in the pot on the stove with handles.*

- Particle vs. preposition:
  *The puppy tore up the staircase.*

- Complement structures
  *The tourists objected to the guide that they couldn’t hear.*
  *She knows you like the back of her hand.*

- Gerund vs. participial adjective
  *Visiting relatives can be boring.*
  *Changing schedules frequently confused passengers.*
Syntactic Ambiguities II

- Modifier scope within NPs
  *impractical design requirements*
  *plastic cup holder*

- Multiple gap constructions
  *The chicken is ready to eat.*
  *The contractors are rich enough to sue.*

- Coordination scope:
  *Small rats and mice can squeeze into holes or cracks in the wall.*
Dark Ambiguities

- Dark ambiguities: most analyses are shockingly bad (meaning, they don’t have an interpretation you can get your mind around)

This analysis corresponds to the correct parse of “This will panic buyers!”

- Unknown words and new usages
- Solution: We need mechanisms to focus attention on the best ones, probabilistic techniques do this
A context-free grammar is a tuple \(<N, \Sigma, S, R>\)

- \(N\) : the set of non-terminals
  - Phrasal categories: S, NP, VP, ADJP, etc.
  - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
- \(\Sigma\) : the set of terminals (the words)
- \(S\) : the start symbol
  - Often written as ROOT or TOP
  - Not usually the sentence non-terminal S
- \(R\) : the set of rules
  - Of the form \(X \rightarrow Y_1 Y_2 \ldots Y_n\), with \(X \in N, n \geq 0, Y_i \in (N \cup \Sigma)\)
  - Examples: \(S \rightarrow NP \ VP, \ VP \rightarrow VP \ CC \ VP\)
  - Also called rewrites, productions, or local trees
Example Grammar

\[ N = \{ S, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN} \} \]

\[ S' = S \]

\[ \Sigma = \{ \text{sleeps, saw, man, woman, telescope, the, with, in} \} \]

\[ R = \begin{array}{|c|c|c|c|}
\hline
S & \Rightarrow & \text{NP} & \text{VP} \\
\hline
\text{VP} & \Rightarrow & \text{Vi} & \\
\hline
\text{VP} & \Rightarrow & \text{Vt} & \text{NP} \\
\hline
\text{VP} & \Rightarrow & \text{VP} & \text{PP} \\
\hline
\text{NP} & \Rightarrow & \text{DT} & \text{NN} \\
\hline
\text{NP} & \Rightarrow & \text{NP} & \text{PP} \\
\hline
\text{PP} & \Rightarrow & \text{IN} & \text{NP} \\
\hline
\end{array} \]

\[ \begin{array}{|l|c|}
\hline
\text{Vi} & \Rightarrow \text{sleeps} \\
\hline
\text{Vt} & \Rightarrow \text{saw} \\
\hline
\text{NN} & \Rightarrow \text{man} \\
\hline
\text{NN} & \Rightarrow \text{woman} \\
\hline
\text{NN} & \Rightarrow \text{telescope} \\
\hline
\text{DT} & \Rightarrow \text{the} \\
\hline
\text{IN} & \Rightarrow \text{with} \\
\hline
\text{IN} & \Rightarrow \text{in} \\
\hline
\end{array} \]

S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition
Example Parses

$R = \begin{array}{|c|c|c|}
\hline
S & \Rightarrow & NP \text{ VP} \\
\hline
VP & \Rightarrow & Vi \\
\hline
VP & \Rightarrow & Vt \text{ NP} \\
\hline
VP & \Rightarrow & VP \text{ PP} \\
\hline
NP & \Rightarrow & DT \text{ NN} \\
\hline
NP & \Rightarrow & NP \text{ PP} \\
\hline
PP & \Rightarrow & IN \text{ NP} \\
\hline
Vi & \Rightarrow & sleeps \\
\hline
Vt & \Rightarrow & saw \\
\hline
NN & \Rightarrow & man \\
\hline
NN & \Rightarrow & woman \\
\hline
NN & \Rightarrow & telescope \\
\hline
DT & \Rightarrow & the \\
\hline
IN & \Rightarrow & with \\
\hline
IN & \Rightarrow & in \\
\hline
\end{array}$

$S = \text{sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition}$

The man sleeps

The man saw the woman with the telescope
A context-free grammar is a tuple \( <N, \Sigma, S, R> \)
- \( N \): the set of non-terminals
  - Phrasal categories: S, NP, VP, ADJP, etc.
  - Parts-of-speech (pre-terminals): NN, JJ, DT, VB, etc.
- \( \Sigma \): the set of terminals (the words)
- \( S \): the start symbol
  - Often written as ROOT or TOP
  - \textit{Not} usually the sentence non-terminal S
- \( R \): the set of rules
  - Of the form \( X \rightarrow Y_1 Y_2 \ldots Y_n \), with \( X \in N \), \( n \geq 0 \), \( Y_i \in (N \cup \Sigma) \)
  - Examples: \( S \rightarrow NP \ VP \), \( VP \rightarrow VP \ CC \ VP \)

A PCFG adds a distribution \( q \):
- Probability \( q(r) \) for each \( r \in R \), such that for all \( X \in N \):
  \[
  \sum_{\alpha \rightarrow \beta \in R : \alpha = X} q(\alpha \rightarrow \beta) = 1
  \]
### PCFG Example

<table>
<thead>
<tr>
<th>S</th>
<th>NP</th>
<th>VP</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>Vi</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>Vt</td>
<td>NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>VP</td>
<td>PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>DT</td>
<td>NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP</td>
<td>P</td>
<td>NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vi</th>
<th>sleeps</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vt</td>
<td>saw</td>
<td>1.0</td>
</tr>
<tr>
<td>NN</td>
<td>man</td>
<td>0.7</td>
</tr>
<tr>
<td>NN</td>
<td>woman</td>
<td>0.2</td>
</tr>
<tr>
<td>NN</td>
<td>telescope</td>
<td>0.1</td>
</tr>
<tr>
<td>DT</td>
<td>the</td>
<td>1.0</td>
</tr>
<tr>
<td>IN</td>
<td>with</td>
<td>0.5</td>
</tr>
<tr>
<td>IN</td>
<td>in</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Probability of a tree $t$ with rules

\[
\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n
\]

is

\[
p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)
\]

where $q(\alpha \rightarrow \beta)$ is the probability for rule $\alpha \rightarrow \beta$. 
PCFG Example

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>NP</td>
<td>VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP</td>
<td>Vi</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>Vt</td>
<td>NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>VP</td>
<td>PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>DT</td>
<td>NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP</td>
<td>P</td>
<td>NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

\[ t_1 = \]

\[ p(t_1) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0 \]

\[ \text{The man sleeps} \]

\[ t_2 = \]

\[ p(t_2) = 1.8 \times 0.3 \times 1.0 \times 0.7 \times 0.2 \times 0.4 \times 1.0 \times 0.3 \times 1.0 \times 0.5 \times 0.3 \times 1.0 \times 0.1 \]

\[ \text{The man saw the woman with the telescope} \]
PCFGs: Learning and Inference

- **Model**
  - The probability of a tree \( t \) with \( n \) rules \( \alpha_i \rightarrow \beta_i \), \( i = 1..n \)

  \[
  p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)
  \]

- **Learning**
  - Read the rules off of labeled sentences, use ML estimates for probabilities

  \[
  q_{ML}(\alpha \rightarrow \beta) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}
  \]

  - and use all of our standard smoothing tricks!

- **Inference**
  - For input sentence \( s \), define \( T(s) \) to be the set of trees whole *yield* is \( s \) (whole leaves, read left to right, match the words in \( s \))

  \[
  t^*(s) = \arg \max_{t \in T(s)} p(t)
  \]
Chomsky Normal Form

- **Chomsky normal form:**
  - All rules of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - In principle, this is no limitation on the space of (P)CFGs
    - N-ary rules introduce new non-terminals

- Unaries / empties are “promoted”
- In practice it’s kind of a pain:
  - Reconstructing n-aries is easy
  - Reconstructing unaries is trickier
  - The straightforward transformations don’t preserve tree scores
- Makes parsing algorithms simpler!
Original Grammar

S → NP VP 0.8
S → Aux NP VP 0.1
S → VP 0.1

NP → Pronoun 0.2
NP → Proper-Noun 0.2
NP → Det Nominal 0.6
Nominal → Noun 0.3
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5

VP → Verb 0.2

VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

Lexicon:

Noun → book | flight | meal | money
  0.1  0.5  0.2  0.2
Verb → book | include | prefer
  0.5  0.2  0.3

Det → the | a | that | this
  0.6  0.2  0.1  0.1
Pronoun → I | he | she | me
  0.5  0.1  0.1  0.3
Proper-Noun → Houston | NWA
  0.8  0.2
Aux → does 1.0
Prep → from | to | on | near | through
  0.25  0.25  0.1  0.2  0.2

CNF Conversion Example
## Original Grammar

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP → Verb</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

## Chomsky Normal Form

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → X1 VP</td>
<td>0.1</td>
</tr>
<tr>
<td>X1 → Aux NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
</tr>
</tbody>
</table>

## Lexicon

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun → book</td>
<td>flight</td>
</tr>
<tr>
<td>0.1 0.5 0.2 0.2</td>
<td></td>
</tr>
<tr>
<td>Verb → book</td>
<td>include</td>
</tr>
<tr>
<td>0.5 0.2 0.3</td>
<td></td>
</tr>
</tbody>
</table>
### Original Grammar

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**NP →**

- Pronoun: 0.2
- Proper-Noun: 0.2
- Det Nominal: 0.6
- Nominal → Noun: 0.3
- Nominal → Nominal Noun: 0.2
- Nominal → Nominal PP: 0.5

**Nominal →**

- Noun: 0.3

**VP →**

- Verb: 0.2
- Verb NP: 0.5
- VP PP: 0.3

### Chomsky Normal Form

<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → X1 VP</td>
<td>0.1</td>
</tr>
<tr>
<td>X1 → Aux NP</td>
<td>1.0</td>
</tr>
<tr>
<td>S → book</td>
<td>include</td>
</tr>
<tr>
<td>S → Verb NP</td>
<td></td>
</tr>
<tr>
<td>S → VP PP</td>
<td></td>
</tr>
</tbody>
</table>

**Lexicon**

(See previous slide for full list):

- **Noun**: book | flight | meal | money
  - 0.1 0.5 0.2 0.2
- **Verb**: book | include | prefer
  - 0.5 0.2 0.3
**Original Grammar**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Chomsky Normal Form**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
</tr>
<tr>
<td>S → X1 VP</td>
<td>0.1</td>
</tr>
<tr>
<td>X1 → Aux NP</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**NP → Pronoun**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
</tr>
</tbody>
</table>

**NP → Proper-Noun**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
</tr>
</tbody>
</table>

**NP → Det Nominal**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

**Nominal → Noun**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
</tr>
</tbody>
</table>

**Nominal → Nominal Noun**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
</tr>
</tbody>
</table>

**Nominal → Nominal PP**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
</tbody>
</table>

**VP → Verb**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
</tr>
</tbody>
</table>

**VP → Verb NP**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
</tbody>
</table>

**VP → VP PP**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
</tr>
</tbody>
</table>

**PP → Prep NP**

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
</tr>
</tbody>
</table>

**Lexicon**

(See previous slide for full list): 

**Noun →** book | flight | meal | money

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
</tr>
</tbody>
</table>

**Verb →** book | include | prefer

<table>
<thead>
<tr>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
</tbody>
</table>
The Parsing Problem

Critics write reviews with computers.
A Recursive Parser

\[ \text{bestScore}(i,j,X) \]

\[
\begin{align*}
\text{if } (j == i) & \quad \text{return } q(X->s[i]) \\
\text{else} & \quad \text{return } \max_k \ q(X->YZ) \times \text{bestScore}(i,k,Y) \times \text{bestScore}(k+1,j,Z)
\end{align*}
\]

- Will this parser work?
- Why or why not?
- Memory/time requirements?
- Q: Remind you of anything? Can we adapt this to other models / inference tasks?
Dynamic Programming

- We will store: score of the max parse of $x_i$ to $x_j$ with root non-terminal $X$
  $$\pi(i, j, X)$$

- So we can compute the most likely parse:
  $$\pi(1, n, S) = \max_{t \in T_G(s)} p(t)$$

- Via the recursion:
  $$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

- With base case:
  $$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$
The CKY Algorithm

- **Input:** a sentence $s = x_1 \ldots x_n$ and a PCFG $= \langle N, \Sigma, S, R, q \rangle$
- **Initialization:** For $i = 1 \ldots n$ and all $X$ in $N$
  \[
  \pi(i, i, X) = \begin{cases} 
  q(X \to x_i) & \text{if } X \to x_i \in R \\
  0 & \text{otherwise} 
  \end{cases}
  \]
- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$ and $j = i+l$
    - For all $X$ in $N$
      \[
      \pi(i, j, X) = \max_{X \to YZ \in R, \atop s \in \{i \ldots (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
      \]
  - also, store back pointers
    \[
    bp(i, j, X) = \arg \max_{X \to YZ \in R, \atop s \in \{i \ldots (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
    \]
Probabilistic CKY Parser

S → NP VP 0.8
S → X1 VP 0.1
X1 → Aux NP 1.0
S → book | include | prefer
     0.01 0.004 0.006
S → Verb NP 0.05
S → VP PP 0.03
NP → I | he | she | me
     0.1 0.02 0.02 0.06
NP → Houston | NWA
     0.16 0.04
Det → the | a | an
     0.6 0.1 0.05
NP → Det Nominal 0.6
Nominal → book | flight | meal | money
     0.03 0.15 0.06 0.06
Nominal → Nominal Nominal 0.2
Nominal → Nominal PP 0.5
Verb → book | include | prefer
     0.5 0.04 0.06
VP → Verb NP 0.5
VP → VP PP 0.3
Prep → through | to | from
     0.2 0.3 0.3
PP → Prep NP 1.0
### Probabilistic CKY Parser

**Book** the **flight** through **Houston**

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>.01,</td>
<td>Verbs</td>
<td>.5</td>
<td>Nominals</td>
<td>.03</td>
</tr>
<tr>
<td></td>
<td>Det: 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP:</td>
<td>.6</td>
<td></td>
<td>.6</td>
<td>.15</td>
<td>.0024</td>
</tr>
<tr>
<td>VP:</td>
<td>.5</td>
<td>.5</td>
<td>.054</td>
<td></td>
<td>.000864</td>
</tr>
<tr>
<td>PP:</td>
<td>1.0</td>
<td>.2</td>
<td>.16</td>
<td></td>
<td>.032</td>
</tr>
<tr>
<td>S:</td>
<td>.01,</td>
<td></td>
<td></td>
<td></td>
<td>.000216</td>
</tr>
<tr>
<td>Nominals</td>
<td>.5</td>
<td>.15</td>
<td>.03</td>
<td>.0024</td>
<td>.000864</td>
</tr>
<tr>
<td>NP:</td>
<td>.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parse Tree**

- **Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.**
# Probabilistic CKY Parser

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
<th>Parse Tree #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : .01, Verb : .5 Nominal : .03</td>
<td>None</td>
<td>S : .05 * .5 * .054 = .00135</td>
<td>None</td>
<td>S : 00001296 S : 00000216</td>
<td></td>
</tr>
<tr>
<td>VP : .5 * .5 * .054 = .0135</td>
<td>None</td>
<td>NP : .6 * .6 * .15 = .054</td>
<td>None</td>
<td>Nominal : .5 * .15 * .032 = .0024</td>
<td></td>
</tr>
<tr>
<td>Det : .6 Nominal : .15</td>
<td>None</td>
<td></td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep : .2 PP : 1.0 * .2 * .16 = .032</td>
<td>NP : .16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Memory

- **How much memory does this require?**
  - Have to store the score cache
  - Cache size: |symbols|*n² doubles

- **Pruning: Beam Search**
  - score[X][i][j] can get too large (when?)
  - Can keep beams (truncated maps score[i][j]) which only store the best K scores for the span [i,j]

- **Pruning: Coarse-to-Fine**
  - Use a smaller grammar to rule out most X[i,j]
  - Much more on this later…
Time: Theory

- How much time will it take to parse?
  - For each diff \( (\text{diff} := j - i) \) (\(<= n\))
    - For each \( i \) (\(<= n\))
      - For each rule \( X \rightarrow Y Z \)
        - For each split point \( k \)
          Do constant work
  - Total time: \(|\text{rules}| * n^3\)
  - Something like 5 sec for an unoptimized parse of a 20-word sentences
Time: Practice

- Parsing with the vanilla treebank grammar:

  ~ 20K Rules
  (not an optimized parser!)
  Observed exponent: 3.6

- Why’s it worse in practice?
  - Longer sentences “unlock” more of the grammar
  - All kinds of systems issues don’t scale
Can also compute other quantities:

- **Best Inside**: score of the max parse of $w_i$ to $w_j$ with root non-terminal $X$

- **Best Outside**: score of the max parse of $w_0$ to $w_n$ with a gap from $w_i$ to $w_j$ rooted with non-terminal $X$
  - see notes for derivation, it is a bit more complicated

- Sum Inside/Outside: Do sums instead of maxes
Why Chomsky Normal Form?

Book       the        flight    through  Houston

S :01,
Verb:.5
Nominal:.03

None

VP:.5*.5*.054
=.0135

None

S:.05*.5*.054
=.00135

S:.03*.0135*.035
=.00001296

S:.05*.5*
=.000864

=0000216

Det:.6

NP:.6*.6*.15
=.054

None

Nominal:.6*.15*.032
=.0024

=000864

Nominal:.15

None

NP:.6*.6*
=.024

=000864

Prep:.2

PP:1.0*.2*.16
=.032

=00001296

NP:.16

Inference:

• Can we keep N-ary (N > 2) rules and still do dynamic programming?
• Can we keep unary rules and still do dynamic programming?

Learning:

• Can we reconstruct the original trees?
We need unaries to be non-cyclic

- Calculate closure \( \text{Close}(R) \) for unary rules in \( R \)
  - Add \( X \rightarrow Y \) if there exists a rule chain \( X \rightarrow Z_1, Z_1 \rightarrow Z_2, \ldots, Z_k \rightarrow Y \) with 
    \( q(X \rightarrow Y) = q(X \rightarrow Z_1) \times q(Z_1 \rightarrow Z_2) \times \ldots \times q(Z_k \rightarrow Y) \)
  - If no unary rule exist for \( X \), add \( X \rightarrow X \) with \( q(X \rightarrow X) = 1 \) for all \( X \) in \( N \)

- Rather than zero or more unaries, always exactly one
- Alternate unary and binary layers
- What about \( X \rightarrow Y \) with different unary paths (and scores)?
The CKY Algorithm

- **Input:** a sentence $s = x_1 \ldots x_n$ and a PCFG $= \langle N, \Sigma, S, R, q \rangle$
- **Initialization:** For $i = 1 \ldots n$ and all $X$ in $N$

  $$\pi(i, i, X) = \begin{cases} 
  q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
  0 & \text{otherwise}
  \end{cases}$$

- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$ and $j = i+l$
    - For all $X$ in $N$
      - [iterate all phrase lengths]
      - [iterate all phrases of length $l$]
      - [iterate all non-terminals]

  $$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

- also, store back pointers

  $$bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$
CKY with Unary Closure

- **Input:** a sentence $s = x_1 .. x_n$ and a PCFG $= <N, \Sigma, S, R, q>$
- **Initialization:** For $i = 1 \ldots n$:
  - Step 1: for all $X$ in $N$:
    $$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$
  - Step 2: for all $X$ in $N$:
    $$\pi_U(i, i, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi(i, i, Y))$$
- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$ and $j = i+l$ [iterate all phrase lengths]
    - Step 1: (Binary)
      - For all $X$ in $N$ [iterate all non-terminals]
        $$\pi_B(i, j, X) = \max_{X \rightarrow YZ \in R, s \in \{i \ldots (j-1)} (q(X \rightarrow YZ) \times \pi_U(i, s, Y) \times \pi_U(s + 1, j, Z))$$
    - Step 2: (Unary)
      - For all $X$ in $N$ [iterate all non-terminals]
        $$\pi_U(i, j, X) = \max_{X \rightarrow Y \in \text{Close}(R)} (q(X \rightarrow Y) \times \pi_B(i, j, Y))$$
(S (NP-SBJ The move)
  (VP followed
    (NP (NP a round)
      (PP of
        (NP (NP similar increases)
          (PP by
            (NP other lenders))
          (PP against
            (NP Arizona real estate loans))))))
',
(S-ADV (NP-SBJ *)
  (VP reflecting
    (NP (NP a continuing decline)
      (PP-LOC in
        (NP that market))))))
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

<table>
<thead>
<tr>
<th>Tree Structure</th>
<th>Grammar Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>1</td>
</tr>
<tr>
<td>S → NP VP .</td>
<td>1</td>
</tr>
<tr>
<td>NP → PRP</td>
<td>1</td>
</tr>
<tr>
<td>VP → VBD ADJP</td>
<td>1</td>
</tr>
<tr>
<td>He was right</td>
<td></td>
</tr>
</tbody>
</table>

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.
Grammar encodings: Non-black states are active, non-white states are accepting, and bold transitions are phrasal. FSAs for a subset of the rules for the category NP.
Treebank Grammar Scale

- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller
Typical Experimental Setup

- Corpus: Penn Treebank, WSJ

Training: sections 02-21
Development: section 22 (here, first 20 files)
Test: section 23

- Accuracy – F1: harmonic mean of per-node labeled precision and recall.

- Here: also size – number of symbols in grammar.
  - Passive / complete symbols: NP, NP^S
  - Active / incomplete symbols: NP → NP CC •
How to Evaluate?

Correct Tree T

S
  VP
    Verb
    book
    Det
    the
    Nominal
    flight
    Prep
    through
    NP
    Houston

Computed Tree P

S
  VP
    Verb
    book
    Det
    the
    Nominal
    flight
    Prep
    through
    NP
    Proper-Noun
    Houston
PARSEVAL Example

Correct Tree T

S
VP
Verb
book
NP
Det
Nominal
the
Noun
flight
Prep
through
NP
Houston

Computed Tree P

S
VP
Verb
book
NP
Det
Nominal
the
Noun
flight
Prep
through
NP
Proper-Noun
Houston

# Constituents: 11

# Correct Constituents: 10

Recall = 10/11 = 90.9%
Precision = 10/12 = 83.3%
F_1 = 87.4%
**Evaluation Metric**

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If $P$ is the system’s parse tree and $T$ is the human parse tree (the “gold standard”):
  - Recall = (# correct constituents in $P$) / (# constituents in $T$)
  - Precision = (# correct constituents in $P$) / (# constituents in $P$)
- Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.
- $F_1$ is the harmonic mean of precision and recall.
  - $F_1 = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$
Performance with Vanilla PCFGs

- Use PCFGs for broad coverage parsing
- Take the grammar right off the trees

```
ROOT
   |   S
   |   NP
   |   VP
   |   PRP
   |   VBD
   |   ADJP
   |   He
   |   was
   |   JJ
   |   right
```

```
ROOT → S  1
S → NP VP .  1
NP → PRP  1
VP → VBD ADJP  1

Model | F1
-----|----
Baseline | 72.0
```

[Charniak 96]
Conditional Independence?

- Not every NP expansion can fill every NP slot
  - A grammar with symbols like “NP” won’t be context-free
  - Statistically, conditional independence too strong
Non-Independence

- Independence assumptions are often too strong.

Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).

Also: the subject and object expansions are correlated!
Grammar Refinement

Structure Annotation [Johnson ’98, Klein&Manning ’03]
Lexicalization [Collins ’99, Charniak ’00]
Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation
Vertical Markovization

- Vertical Markov order: rewrites depend on past $k$ ancestor nodes. (cf. parent annotation)
Horizontal Markovization

Order 1

Order $\infty$

Horizontal Markov Orde

Symbols

Horizontal Markov Order
## Vertical and Horizontal

- **Raw treebank:** \( v=1, h=\infty \)
- **Johnson 98:** \( v=2, h=\infty \)
- **Collins 99:** \( v=2, h=2 \)
- **Best F1:** \( v=3, h=2v \)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v=h=2v )</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
</tbody>
</table>
## Unlexicalized PCFG Grammar Size

<table>
<thead>
<tr>
<th>Vertical Order</th>
<th>Horizontal Markov Order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$h = 0$</td>
</tr>
<tr>
<td>$v = 1$ No annotation</td>
<td>71.27 (854)</td>
</tr>
<tr>
<td>$v \leq 2$ Sel. Parents</td>
<td>74.75 (2285)</td>
</tr>
<tr>
<td>$v = 2$ All Parents</td>
<td>74.68 (2984)</td>
</tr>
<tr>
<td>$v \leq 3$ Sel. GParents</td>
<td>76.50 (4943)</td>
</tr>
<tr>
<td>$v = 3$ All GParents</td>
<td>76.74 (7797)</td>
</tr>
</tbody>
</table>

**Figure 2:** Markovizations: $F_1$ and grammar size.
Tag Splits

- **Problem:** Treebank tags are too coarse.

- **Example:** Sentential, PP, and other prepositions are all marked IN.

- **Partial Solution:**
  - Subdivide the IN tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=h=2v</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>
Other Tag Splits

- **UNARY-DT**: mark demonstratives as DT\(^U\) (“the X” vs. “those”)
- **UNARY-RB**: mark phrasal adverbs as RB\(^U\) ("quickly" vs. “very”)
- **TAG-PA**: mark tags with non-canonical parents (“not” is an RB\(^V\)P)
- **SPLIT-AUX**: mark auxiliary verbs with –AUX [cf. Charniak 97]
- **SPLIT-CC**: separate “but” and “&” from other conjunctions
- **SPLIT-%**: “%” gets its own tag.

<table>
<thead>
<tr>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>80.4</td>
<td>8.1K</td>
</tr>
<tr>
<td>80.5</td>
<td>8.1K</td>
</tr>
<tr>
<td>81.2</td>
<td>8.5K</td>
</tr>
<tr>
<td>81.6</td>
<td>9.0K</td>
</tr>
<tr>
<td>81.7</td>
<td>9.1K</td>
</tr>
<tr>
<td>81.8</td>
<td>9.3K</td>
</tr>
</tbody>
</table>
A Fully Annotated (Unlex) Tree

```
ROOT
  | S^ROOT-v
  |  
  |   
```

```
  | "^S NP^S-B
  |  
  |   
```

```
  | VP^S-VBF-v
  |  
  |   
```

```
  | ^S "^S
  |  
  |   
```

```
  | DT-U^NP
  |  
  |   
```

```
  | Vbz^BE^VP
  |  
  |   
```

```
  | NP^VP-B
  |  
  |   
```

```
  | This
  |  
  |   
```

```
  | is
  |  
  |   
```

```
  | NN^NP
  |  
  |   
```

```
  | panic
  |  
  |   
```

```
  | NN^NP
  |  
  |   
```

```
  | buying
  |  
  |   
```
### Some Test Set Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td>84.7</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td>86.0</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td>86.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td>87.4</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td>88.6</td>
</tr>
</tbody>
</table>

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Structural annotation [Johnson ’98, Klein and Manning 03]
  - Head lexicalization [Collins ’99, Charniak ’00]
Problems with PCFGs

- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words
What’s different between basic PCFG scores here?
What (lexical) correlations need to be scored?
Add "headwords" to each phrasal node

- Headship not in (most) treebanks
- Usually use *(handwritten)* head rules, e.g.:
  - NP:
    - Take leftmost NP
    - Take rightmost N*
    - Take rightmost JJ
    - Take right child
  - VP:
    - Take leftmost VB*
    - Take leftmost VP
    - Take left child

Lexicalize Trees!
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

  $$\text{VP}(\text{saw}) \rightarrow \text{VBD}(\text{saw}) \ \text{NP-C}(\text{her}) \ \text{NP}(\text{today})$$

- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps
Complement / Adjunct Distinction

- *warning* - can be tricky, and most parsers don’t model the distinction

- **Complement**: defines a property/argument (often obligatory), ex: [capitol [of Rome]]

- **Adjunct**: modifies / describes something (always optional), ex: [quickly ran]

- A Test for Adjuncts: [X Y] --> can claim X and Y
  - [they ran and it happened quickly] vs. [capitol and it was of Rome]
Lexical Derivation Steps

- **Main idea:** define a linguistically-motivated Markov process for generating children given the parent

1. **Step 1:** Choose a head tag and word

2. **Step 2:** Choose a complement bag

3. **Step 3:** Generate children (incl. adjuncts)

4. **Step 4:** Recursively derive children

[Collins 99]
**Lexicalized CKY**

bestScore(i, j, X, h)

if (j = i+1)
    return tagScore(X, s[i])
else
    return max
    max
    score(X[h]→Y[h] Z[h']) * bestScore(i, k, Y, h') * bestScore(k+1, j, Z, h)
    score(X[h]→Y[h'] Z[h]) * bestScore(i, k, Y, h') * bestScore(k+1, j, Z, h)

still cubic time?
Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the $O(n^5)$ CKY
  - If we keep $K$ hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
  - Keeps things more or less cubic

- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Treebank Grammar</td>
<td>72.6</td>
</tr>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.6</td>
</tr>
</tbody>
</table>
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson’98]
  - Head lexicalization [Collins’99, Charniak’00]
  - Automatic Grammar Refinement?
Manual Annotation

- Manually split categories
  - NP: subject vs object
  - DT: determiners vs demonstratives
  - IN: sentential vs prepositional

- Advantages:
  - Fairly compact grammar
  - Linguistic motivations

- Disadvantages:
  - Performance leveled out
  - Manually annotated
Learning Latent Annotations

Latent Annotations:

- Brackets are known
- Base categories are known
- Hidden variables for subcategories

He was right.

Can learn with EM: like Forward-Backward for HMMs.
Automatic Annotation Induction

- **Advantages:**
  - Automatically learned:
    - Label *all* nodes with latent variables.
    - Same number $k$ of subcategories for all categories.

- **Disadvantages:**
  - Grammar gets too large
  - Most categories are oversplit while others are undersplit.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>
## Final Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 ≤ 40 words</th>
<th>F1 all words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein &amp; Manning ’03</td>
<td>86.3</td>
<td>85.7</td>
</tr>
<tr>
<td>Matsuzaki et al. ’05</td>
<td>86.7</td>
<td>86.1</td>
</tr>
<tr>
<td>Collins ’99</td>
<td>88.6</td>
<td>88.2</td>
</tr>
<tr>
<td>Charniak &amp; Johnson ’05</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Petrov et. al. 06</td>
<td>90.2</td>
<td>89.7</td>
</tr>
</tbody>
</table>
John has a dog ⇒

(\textit{S} (\textit{NP} \textit{NNP})_{\text{NP}} (\textit{VP} \textit{VBZ} (\textit{NP} \textit{DT} \textit{NN})_{\text{NP}})_{\text{VP}} \cdot)_{\text{S}}

- Linearize a tree into a sequence
- Then parsing problem becomes similar to machine translation
  - Input: sequence
  - Output: sequence (of different length)
- Encoder-decoder LSTMs (Long short-term memory networks)
John has a dog ➔

\[
\text{S} \rightarrow \text{NP} \rightarrow \text{VP} \rightarrow .
\]

\[
\text{NNP} \quad \text{VBZ} \quad \text{NP} \\
\quad \text{DT} \quad \text{NN}
\]

John has a dog ➔

\[
(S \ (\text{NP} \ NNP \ )_{NP} \ (\text{VP} \ VBZ \ (\text{NP} \ DT \ NN \ )_{NP} \ )_{VP} \ . \ )_{S}
\]

- Penn treebank (~40K sentences) is too small to train LSTMs
- Create a larger training set with 11M sentences automatically parsed by two state-of-the-art parsers (and keep only those sentences for which two parsers agreed)
## “Grammar as Foreign Language” (deep learning)

Vinyals et al., 2015

<table>
<thead>
<tr>
<th>Parser</th>
<th>Training Set</th>
<th>WSJ 22</th>
<th>WSJ 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline LSTM+D</td>
<td>WSJ only</td>
<td>&lt; 70</td>
<td>&lt; 70</td>
</tr>
<tr>
<td>LSTM+A+D</td>
<td>WSJ only</td>
<td>88.7</td>
<td>88.3</td>
</tr>
<tr>
<td>LSTM+A+D ensemble</td>
<td>WSJ only</td>
<td>90.7</td>
<td>90.5</td>
</tr>
<tr>
<td>baseline LSTM</td>
<td>BerkeleyParser corpus</td>
<td>91.0</td>
<td>90.5</td>
</tr>
<tr>
<td>LSTM+A</td>
<td>high-confidence corpus</td>
<td>93.3</td>
<td>92.5</td>
</tr>
<tr>
<td>LSTM+A ensemble</td>
<td>high-confidence corpus</td>
<td>93.5</td>
<td>92.8</td>
</tr>
<tr>
<td>Petrov et al. (2006) [12]</td>
<td>WSJ only</td>
<td>91.1</td>
<td>90.4</td>
</tr>
<tr>
<td>Zhu et al. (2013) [13]</td>
<td>WSJ only</td>
<td>N/A</td>
<td>90.4</td>
</tr>
<tr>
<td>Petrov et al. (2010) ensemble [14]</td>
<td>WSJ only</td>
<td>92.5</td>
<td>91.8</td>
</tr>
<tr>
<td>Zhu et al. (2013) [13]</td>
<td>semi-supervised</td>
<td>N/A</td>
<td>91.3</td>
</tr>
<tr>
<td>McClosky et al. (2006) [16]</td>
<td>semi-supervised</td>
<td>92.4</td>
<td>92.1</td>
</tr>
</tbody>
</table>
## Final Results (Accuracy)

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>≤ 40 words F1</th>
<th>all F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>Charniak&amp;Johnson ‘05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>GER</td>
<td>Dubey ‘05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td>CHN</td>
<td>Chiang et al. ‘02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>86.3</td>
<td>83.4</td>
</tr>
</tbody>
</table>

Still higher numbers from reranking / self-training methods