Ambiguity

One morning I shot an elephant in my pajamas.
How he got into my pajamas I don't know.

Groucho Marx, Animal Crackers, 1930
Probabilistic CFGs

\[ N \text{ a set of non-terminal symbols (or variables)} \]
\[ \Sigma \text{ a set of terminal symbols (disjoint from } N) \]
\[ R \text{ a set of rules or productions, each of the form } A \rightarrow \beta \mid p, \]
where \( A \) is a non-terminal,
\( \beta \) is a string of symbols from the infinite set of strings \((\Sigma \cup N)^+\),
and \( p \) is a number between 0 and 1 expressing \( P(\beta | A) \)
\[ S \text{ a designated start symbol} \]

Rule probabilities define conditional distributions over the different ways of rewriting each non-terminal: \( P(A \rightarrow \beta | A) \)

For consistency: \[ \sum_{\beta} P(A \rightarrow \beta | A) = 1 \]
PCFGs for Disambiguation

Two parse trees for:

- Book the dinner flight.

Rule probabilities for:

- Book the dinner flight.

<table>
<thead>
<tr>
<th>Rules</th>
<th>P</th>
<th>Rules</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>S — VP</td>
<td>.05</td>
<td>S — VP</td>
<td>.05</td>
</tr>
<tr>
<td>VP — Verb NP</td>
<td>.20</td>
<td>VP — Verb NP NP</td>
<td>.10</td>
</tr>
<tr>
<td>NP — Det Nominal</td>
<td>.20</td>
<td>NP — Det Nominal</td>
<td>.20</td>
</tr>
<tr>
<td>Nominal — Nominal Noun</td>
<td>.20</td>
<td>Nominal — Nominal</td>
<td>.15</td>
</tr>
<tr>
<td>Nominal — Noun</td>
<td>.75</td>
<td>Nominal — Noun</td>
<td>.75</td>
</tr>
<tr>
<td>Verb — book</td>
<td>.30</td>
<td>Verb — book</td>
<td>.30</td>
</tr>
<tr>
<td>Det — the</td>
<td>.60</td>
<td>Det — the</td>
<td>.60</td>
</tr>
<tr>
<td>Noun — dinner</td>
<td>.10</td>
<td>Noun — dinner</td>
<td>.10</td>
</tr>
<tr>
<td>Noun — flights</td>
<td>.40</td>
<td>Noun — flights</td>
<td>.40</td>
</tr>
</tbody>
</table>
PCFGs for Disambiguation

The probability of a parse tree $T$ and sentence $S$ is:
\[ P(T, S) = \prod_{i=1}^{n} P(A_i \rightarrow \beta_i | A_i) \]
where $A_i \rightarrow \beta_i$ is the rule used to expand one of the $n$ nonterminal nodes of $T$.

Note: $P(T, S) = P(T) P(S | T) = P(T)$, since $P(S | T) = 1$.

For $S = \text{Book the dinner flight}$,
\[ P(\text{first parse tree}) = 2.2 \times 10^{-6} \]
\[ P(\text{second parse tree}) = 6.1 \times 10^{-7} \]

PCFGs for Disambiguation

Let $\mathcal{T}(S)$ be the set of parse trees with $S$ as the yield.

We want to compute:
\[ \hat{T}(S) = \arg\max_{T \in \mathcal{T}(S)} P(T | S) \]
\[ = \arg\max_{T \in \mathcal{T}(S)} \frac{P(T, S)}{P(S)} \]
\[ = \arg\max_{T \in \mathcal{T}(S)} P(T, S) \]
\[ = \arg\max_{T \in \mathcal{T}(S)} P(T) \]
PCFGs for Language Modeling

Compute the probability of a sentence as:

\[ P(S) = \sum_{T \in \mathcal{T}(S)} P(T, S) \]

\[ = \sum_{T \in \mathcal{T}(S)} P(T) \]

Example:

- the contract ended with a loss of 7 cents after trading as low as 9 cents
- N-gram model uses “7 cents” to predict “after”
- In principle, PCFG model could use “contract ended” to predict “after”.

Parsing PCFGs

Goal is to compute:

\[ \hat{T}(S) = \arg\max_{T \in \mathcal{T}(S)} P(T) \]

Use probabilistic extensions of standard parsing algorithms:

- Probabilistic CKY algorithm
- Probabilistic Earley algorithm
- Probabilistic chart parsing algorithm
- ...

Review: The CKY Algorithm

CKY Parsing:

- Bottom-up search.
- Implemented with a dynamic programming table.

Requires grammar in Chomsky Normal Form:

- No $\varepsilon$ in grammar rules.
- All grammar rules in form:
  - $A \rightarrow B \ C$, or
  - $A \rightarrow a$

---

Parse Table for "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>0:2</td>
<td>VP</td>
<td>Nominal, Noun</td>
<td>VP</td>
<td><strong>NP</strong></td>
<td><strong>NP</strong></td>
</tr>
<tr>
<td>0:3</td>
<td></td>
<td>NP</td>
<td></td>
<td>[1:4]</td>
<td>[1:5]</td>
</tr>
<tr>
<td>0:4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Det</td>
<td>Nominal, Noun</td>
<td>DP</td>
<td>Nominal</td>
<td>Nominal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

CS 533: Natural Language Processing - Professor McCarty
Review: The CKY Algorithm

function CKY-Parse(words, grammar) returns table

for j = 1 to LENGTH(words) do
    table[j - 1, j] = {A | A → words[j] ∈ grammar}
for i = j - 2 downto 0 do
    for k = i + 1 to j - 1 do
        table[i, j] = table[i, j] ∪
        {A | A → BC ∈ grammar, B ∈ table[i, k], C ∈ table[k, j]}

* Fill the upper-triangular matrix a column at a time, working from left to right.
* Fill each column from bottom to top.

Review: The CKY Algorithm

Filling in the \([i,j]\)th cell in the parse table:

```
   \[
   \begin{array}{cccc}
   & \text{\ldots} & \text{[i+1]} & \text{[i+2]} \\
   \text{\ldots} & \text{\ldots} & \text{\ldots} & \text{\ldots} \\
   \text{[i+1]} & \text{\ldots} & \text{\ldots} & \text{\ldots} \\
   \text{[i+2]} & \text{\ldots} & \text{\ldots} & \text{\ldots} \\
   \end{array}
   \]
```

CS 533: Natural Language Processing – Professor McCarty
## Review: CKY Example

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, VP, Verb, Nominal, Noun</td>
<td>S, VP,X2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Det</td>
<td>NP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nominal, Noun</td>
<td>Nominal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep</td>
<td></td>
<td>NP, Proper-Noun</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Filling column 5
Review: CKY Example

Book

the

flight

through

Houston

S, VP, Verb, Nominal, Noun [0,1]

[0,2] S, VP, X2

[0,3] S, VP, X2

[0,4] S, VP, X2

[0,5] S, VP, X2

Det

NP

NP

Nominal, Noun

[1,2] Nominal

[1,3] Nominal

[1,4] Nominal

[1,5] Nominal

Prep

PP

PP

NP, Proper Noun

[2,2] NP, Proper Noun

[2,3] NP, Proper Noun

[2,4] NP, Proper Noun

[2,5] NP, Proper Noun

[3,4] NP, Proper Noun

[3,5] NP, Proper Noun

[4,4] NP, Proper Noun

[4,5] NP, Proper Noun
**Review: CKY Example**

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, VP, Verb</td>
<td>Nominal, Noun</td>
<td>[0,1]</td>
<td>S, VP, X2</td>
<td>S, VP, X2</td>
</tr>
<tr>
<td>Det</td>
<td>NP</td>
<td>[1,2]</td>
<td>NP</td>
<td>[1,3]</td>
</tr>
<tr>
<td>Nominal, Noun</td>
<td>[2,3]</td>
<td>Prep</td>
<td>PP</td>
<td>[3,4]</td>
</tr>
<tr>
<td>[3,5]</td>
<td>NP, Proper Noun</td>
<td>[4,5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Probabilistic CKY Algorithm**

```python
function PROBABILISTIC-CKY(words, grammar) returns most probable parse and its probability

for j ← from 1 to LENGTH(words) do
    for all \( A \mid A \rightarrow \text{words}[j] \in \text{grammar} \)
        \( table[j−1, j, A] = P(A \rightarrow \text{words}[j]) \)
    for i ← from j−2 downto 0 do
        for k ← i+1 to j−1 do
            for all \( A \mid A \rightarrow BC \in \text{grammar} , \) and \( table[i, k, B] > 0 \) and \( table[k+1, j, C] > 0 \)
                if \( table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k+1, j, C] \) then
                    \( table[i, j, A] = P(A \rightarrow BC) \times table[i, k, B] \times table[k+1, j, C] \)
                    \( back[i, j, A] = \{ k, B, C \} \)
            return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```
Probabilistic CKY Algorithm

Example:
Given the PCFG

<table>
<thead>
<tr>
<th>Rule</th>
<th>Non-terminals</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>.80</td>
<td></td>
</tr>
<tr>
<td>NP → Det N</td>
<td>.30</td>
<td></td>
</tr>
<tr>
<td>VP → V NP</td>
<td>.20</td>
<td></td>
</tr>
<tr>
<td>V → includes</td>
<td>.05</td>
<td></td>
</tr>
<tr>
<td>Det → the</td>
<td>.40</td>
<td></td>
</tr>
<tr>
<td>Det → a</td>
<td>.40</td>
<td></td>
</tr>
<tr>
<td>N → meal</td>
<td>.01</td>
<td></td>
</tr>
<tr>
<td>N → flight</td>
<td>.02</td>
<td></td>
</tr>
</tbody>
</table>

which is already in Chomsky Normal Form (CNF), parse the sentence:

\* The flight includes a meal.

Probabilistic CKY Algorithm

Det: .40

NP: .30 * .40 = .12

V: .20

The flight includes a meal

Det: .40

NP: .30 * .40 = .0024

V: .20

The flight includes a meal

Det: .40

NP: .30 * .40 = .0024

V: .20

The flight includes a meal

Det: .40

NP: .30 * .40 = .0024

V: .20

The flight includes a meal
Learning Rule Probabilities

From a parsed corpus, e.g., the Penn Treebank:

\[ P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_y \text{count}(\alpha \rightarrow y)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)} \]

But what if we don’t have a parsed corpus?

- Use EM algorithm (inside-outside algorithm):
  - E-Step computes the expected number of times that a particular rule will be used.
  - M-Step computes the maximum likelihood (re)-estimates of the rule probabilities.

Weaknesses of PCFGs

- Lack of sensitivity to lexical items:
  - Resolution of ambiguity often depends on individual words.
    - Solution: Lexicalized PCFGs.
- Lack of sensitivity to syntactic context:
  - Structural preferences.
    - Solution: Add context to probability model.
  - Structural dependencies.
    - Solution: Split nonterminals.
Lexical Dependencies

Example:

\* workers dumped sacks into a bin

Which parse is correct?

![Parse Trees](image)

Lexical Dependencies

VP attachment? or NP attachment?

\* workers dumped sacks into a bin
\* fishermen caught tons of herring
Lexical Dependencies

Which parse is preferred? Why?

Note: Both parses would have the same probability in a PCFG.

Structural Preferences

Which structure is preferred?

- president of a company in Africa
- a candidate for the election in New York

Which structure is preferred?

- John was believed to have been shot by Bill.
Structural Dependencies

In English, a subject NP is more likely to be a pronoun than is an object NP:

<table>
<thead>
<tr>
<th></th>
<th>Pronoun</th>
<th>Non-Pronoun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>91%</td>
<td>9%</td>
</tr>
<tr>
<td>Object</td>
<td>34%</td>
<td>66%</td>
</tr>
</tbody>
</table>

But PCFGs cannot represent this contextual fact.
Can the probability of expanding an NP by either \( NP \rightarrow PRP \) or \( NP \rightarrow DT \ \text{NN} \) be conditioned on whether the NP is a subject or an object?

Lexicalized PCFGs

Basic Idea:

- Annotate every nonterminal node of the parse tree with:
  - Head word (called the lexical head).
  - Head tag.
- Condition the probability model on these words and tags.

To Implement this Idea:

- Specify independence assumptions carefully, based on linguistic theory.
Finding Heads

Annotated parse tree from Penn Treebank:

What is the algorithm to find these lexical heads?
Finding Heads

To find the head of an NP:

- If the last word is tagged POS, return last word.
- Else search from right to left for the first child which is an NN, NNP, NNPS, NNS, NX, POS, JJR.
- Else search from left to right for the first child which is an NP.
- Else search from right to left for the first child which is a $, ADJP or PRN.
- Else search from right to left for the first child which is a CD.
- Else search from right to left for the first child which is a JJ, JJ, JJS, RB or QP.
Lexicalized PCFGs

Problem: Sparse Data.

- If we estimated rule probabilities by:

\[
\text{count}(\text{VP(dumped, VBD)}) - \text{count}(\text{VP(dumped, VBD)}|\text{NP(sacks, NNS})|\text{PP( into, P)})
\]

we would mostly compute zero counts.

Solution:

- Use a generative or history-based model with appropriate independence assumptions.
Collins' Model 1

Represent the rules as:

\[ LHS \rightarrow L_n L_{n-1} \ldots L_1 H R_1 \ldots R_{m-1} R_m \]

Generate each subtree of the parse tree by:

- Conditioning on parent \( P \), generate head \( H \).
- Conditioning on parent \( P \) and head \( H \), generate left dependents \( L_1, \ldots, L_{n-1}, L_n \).
- Conditioning on parent \( P \) and head \( H \), generate right dependents \( R_1, \ldots, R_{m-1}, R_m \).

Add a special STOP symbol before \( L_n \) and after \( R_m \).

Collins' Model 1

Probability Model:

- The probability of generating head \( H \) is:
  \[ P_H(H \mid P, h) \] where \( h \) is the head word.
- The probability of generating the modifiers to the left of the head is:
  \[ \prod_{i=1}^{n+1} P_L(L_i(w_i, t_i) \mid P, H, h) \] where \( L_{n+1} = \text{STOP} \).
- The probability of generating the modifiers to the right of the head is:
  \[ \prod_{j=1}^{m+1} P_R(R_j(w_j, t_j) \mid P, H, h) \] where \( R_{m+1} = \text{STOP} \).
Collins' Model 1

Generate head $H$:

- VP(dumped, VBD)
- VBD(dumped, VBD)

Generate left dependents:

- VP(dumped, VBD)
- STOP  VBD(dumped, VBD)

Generate right dependents:

- VP(dumped, VBD)
  - STOP  VBD(dumped, VBD)  NP(sacks, NNS)

- VP(dumped, VBD)
  - STOP  VBD(dumped, VBD)  NP(sacks, NNS)  PP(into, P)
**Collins’ Model 1**

Generate right dependents:

```
VP(dumped, VBD)  
        |  
STOP  VP(dumped, VBD)  NP(sacks, NNS)  PP(into, P)  STOP
```

**Probability of this rule:**

\[
P_H(VBD \mid VP, dumped) \\
\times P_L(STOP \mid VP, VBD, dumped) \\
\times P_R(NP(sacks, NNS) \mid VP, VBD, dumped) \\
\times P_R(PP(into, P) \mid VP, VBD, dumped) \\
\times P_R(STOP \mid VP, VBD, dumped)
\]

**Collins’ Model 1**

Add a distance parameter:

\[
P_L = P_L(L_i(w_i, t_i) \mid P, H, h, distance_L(i-1)) \\
P_R = P_R(R_i(w_i, t_i) \mid P, H, h, distance_R(i-1))
\]

which encodes two binary features:

- Is the intervening string of length zero?
- Does the intervening string contain a verb?
Collins' Model 1

Example:

♦ John was believed to have been shot yesterday by Bill.

(a)
(b)

Dependency Distance
by → shot 00
Collins' Model 1

Example:

⋆ John was believed to have been shot yesterday by Bill.

Smoothing

Zero counts are still a major problem; thus Collins' parser interpolates three "backed-off" models:

| Backoff Level | \( P(R(r_w, r_l) | ...) \) | Example |
|---------------|-----------------|---------|
| 1             | \( P(R(r_w, r_l) | P_{inv}, ht) \) | \( P(R(sacks, NNS) | VP, VBD, dumped) \) |
| 2             | \( P(R(r_w, r_l) | P_{inv}) \)   | \( P(R(sacks, NNS) | VP, VBD) \)    |
| 3             | \( P(R(r_w, r_l) | P) \)         | \( P(R(sacks, NNS) | VP) \)       |

The interpolation formula is:

\[
P_g(...) = \lambda_1 e_1 + (1-\lambda_1)(\lambda_2 e_2 + (1-\lambda_2)e_3)
\]

where \( \lambda_1 \) and \( \lambda_2 \) are set to implement Witten-Bell discounting.
Collins' Model 2

Complements vs. Adjuncts:

- Complements are directly related to the head they modify, while adjuncts are more indirectly related.
- Complements are usually arguments of the head.
  - Yesterday Hillary told ...
- Adjuncts add modifying information: time, place, manner, etc.
  - *Yesterday Hillary told ...
- Complements are usually required, whereas adjuncts are optional.
  - Hillary told ...
  - *Yesterday told ...

Add tags for the complement/adjunct distinction:

```
S
  NP-C[subject]
  VP[verb]

S
  NP[modifier]
  VP[verb]

S
  NP[yesterday
  NNP[Hillary]
  VP[told...]
```
Collins' Model 2

Add tags for the complement/adjunct distinction:

```
Collins' Model 2

Add subcategorization frames to the model:

- Generate head $H$ with probability $P_H(H | P, h)$.
- Generate left and right subcat frames, $LC$ and $RC$, with probabilities:
  
  \[ P_{LC}(LC | P, H, h) \]
  
  \[ P_{RC}(RC | P, H, h) \]

- Generate left and right modifiers with:
  
  \[ P_L = P_L(L_i(w_i, t_i) | P, H, h, distance_L(i-1), LC) \]
  
  \[ P_R = P_R(R_i(w_i, t_i) | P, H, h, distance_R(i-1), RC) \]
```
Collins' Model 3

Traces and Wh-Movement:
- The store [SBAR that TRACE bought Lotus] ...
- The store [SBAR that IBM bought TRACE] ...
- The store [SBAR that IBM bought Lotus from TRACE] ...

Add traces and gap features to parse trees.

Revise probability model to incorporate:

\[ P_G(G \mid P, H, h) \]

where \( G \) is either Head, Left or Right.

In Left and Right cases, add \( +\text{gap} \) to subcat frame.
Splitting Nonterminals

To distinguish a subject NP from an object NP:

- Add a parent annotation to the nonterminal nodes.

\[
\begin{array}{c}
\text{a)} \quad S \\
\text{NP} \quad \text{VP} \\
\text{PRP} \quad \text{VBD} \quad \text{NP} \\
\text{I} \quad \text{need} \quad \text{DT} \quad \text{NN} \\
\quad \quad \text{a flight}
\end{array}
\quad \begin{array}{c}
\text{b)} \quad S \\
\text{NP\’S} \quad \text{VP\’S} \\
\text{PRP} \quad \text{VBD} \quad \text{NP\’VP} \\
\text{I} \quad \text{need} \quad \text{DT} \quad \text{NN} \\
\quad \quad \text{a flight}
\end{array}
\]

Can also split pre-terminal nodes:

For a system that automatically splits and merges nonterminal nodes, see Petrov, et al. (2006).
Evaluating Parsers

PARSEVAL measures (Black, et al., 1991):

**labelled recall:** \[ \frac{\text{# of correct constituents in hypothesis parse}}{\text{# of constituents in reference parse}} \]

**labelled precision:** \[ \frac{\text{# of correct constituents in hypothesis parse}}{\text{total # of constituents in hypothesis parse}} \]

**crossing brackets:** \[ \# \text{ of constituents that violate constituent boundaries in reference parse} \]

For a constituent to be “correct” it must span the same set of words and have the same label as a constituent in the reference parse.

Results for Collins’ parser:

<table>
<thead>
<tr>
<th>Model</th>
<th>≤ 40 Words (2,245 sentences)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LR</td>
</tr>
<tr>
<td>Magerman 1995</td>
<td>84.6%</td>
</tr>
<tr>
<td>Collins 1996</td>
<td>85.8%</td>
</tr>
<tr>
<td>Goodman 1997</td>
<td>84.8%</td>
</tr>
<tr>
<td>Charniak 1997</td>
<td>87.9%</td>
</tr>
<tr>
<td>Model 1</td>
<td>87.9%</td>
</tr>
<tr>
<td>Model 2</td>
<td>88.5%</td>
</tr>
<tr>
<td>Model 3</td>
<td>88.6%</td>
</tr>
<tr>
<td>Charniak 2000</td>
<td>90.1%</td>
</tr>
<tr>
<td>Collins 2000</td>
<td>90.1%</td>
</tr>
</tbody>
</table>
Evaluating Parsers

Results for Collins' parser:

Collins' evaluation used Section 23 of the Wall Street Journal corpus.

Overall labeled recall / labeled precision was 88.0% / 88.3%.

Accuracy for particular syntactic structures:

- Verb complements: 93.76 / 92.96.
- Other complements: 94.74 / 94.12.
- Prepositional phrases: 82.29 / 81.51.
- Other verb adjuncts: 75.11 / 78.44.
- Coordinated conjunctions: 61.47 / 62.20.
Evaluating Parsers

Results from Petrov, et al. (2006):

<table>
<thead>
<tr>
<th></th>
<th>LP</th>
<th>LR</th>
<th>CB</th>
<th>OCB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>≤ 40 words</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Klein and Manning (2003)</td>
<td>86.9</td>
<td>85.7</td>
<td>1.10</td>
<td>60.3</td>
</tr>
<tr>
<td>Matsuzaki et al. (2005)</td>
<td>86.6</td>
<td>86.7</td>
<td>1.19</td>
<td>61.1</td>
</tr>
<tr>
<td>Collins (1999)</td>
<td>88.7</td>
<td>88.5</td>
<td>0.92</td>
<td>66.7</td>
</tr>
<tr>
<td>Charniak and Johnson (2005)</td>
<td>90.1</td>
<td>90.1</td>
<td>0.74</td>
<td>70.1</td>
</tr>
<tr>
<td>This Paper</td>
<td>90.3</td>
<td>90.0</td>
<td>0.78</td>
<td>68.5</td>
</tr>
<tr>
<td><strong>all sentences</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Klein and Manning (2003)</td>
<td>86.3</td>
<td>85.1</td>
<td>1.31</td>
<td>57.2</td>
</tr>
<tr>
<td>Matsuzaki et al. (2005)</td>
<td>86.1</td>
<td>86.0</td>
<td>1.39</td>
<td>58.3</td>
</tr>
<tr>
<td>Collins (1999)</td>
<td>88.3</td>
<td>88.1</td>
<td>1.06</td>
<td>64.0</td>
</tr>
<tr>
<td>Charniak and Johnson (2005)</td>
<td>89.5</td>
<td>89.6</td>
<td>0.88</td>
<td>67.6</td>
</tr>
<tr>
<td>This Paper</td>
<td>89.8</td>
<td>89.6</td>
<td>0.92</td>
<td>66.3</td>
</tr>
</tbody>
</table>

Human Parsing Performance

Imagine hearing (or reading):
- The ...
- The horse ...
- The horse raced ...
- The horse raced past ...
- The horse raced past the ...
- The horse raced past the barn ...
- The horse raced past the barn fell.

These are called “garden path” sentences. Preferred parse may depend on probabilities.
Legal Texts


Richard Carter and his wife, Carol, appeal and argue that the district court erred in granting summary judgment in favor of Exxon Company USA on their Petroleum Marketing Practices Act claim and on Exxon’s state law counterclaim.

Here is the parse tree for this sentence.