Parsing with Context-Free Grammars

CS 585, Fall 2017
Introduction to Natural Language Processing
http://people.cs.umass.edu/~brenocon/inlp2017

Brendan O’Connor
College of Information and Computer Sciences
University of Massachusetts Amherst
Context-Free Grammar

- CFG describes a generative process for an (infinite) set of strings
- 1. Nonterminal symbols
  - “S”: START symbol / “Sentence” symbol
- 2. Terminal symbols: word vocabulary
- 3. Rules (a.k.a. Productions). Practically, two types:

“Grammar”: one NT expands to >=1 NT
always one NT on left side of rulep

<table>
<thead>
<tr>
<th>Rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>I</td>
</tr>
<tr>
<td>Pro-Noun</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>Det Nominal</td>
<td>a + flight</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal Noun</td>
<td>morning + flight</td>
</tr>
<tr>
<td>Noun</td>
<td>flights</td>
</tr>
<tr>
<td>$VP \rightarrow$ Verb</td>
<td>do</td>
</tr>
<tr>
<td>Verb NP</td>
<td>want + a flight</td>
</tr>
<tr>
<td>Verb NP PP</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>Verb PP</td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td>$PP \rightarrow$ Preposition NP</td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>

Lexicon: NT expands to a terminal

- Noun $\rightarrow$ flights | breeze | trip | morning | …
- Verb $\rightarrow$ is | prefer | like | need | want | fly
- Adjective $\rightarrow$ cheapest | non-stop | first | latest
- | other | direct | …
- Pronoun $\rightarrow$ me | I | you | it | …
- Proper-Noun $\rightarrow$ Alaska | Baltimore | Los Angeles
- | Chicago | United | American | …
- Determiner $\rightarrow$ the | a | an | this | these | that | …
- Preposition $\rightarrow$ from | to | on | near | …
- Conjunction $\rightarrow$ and | or | but | …
Constituent Parse Trees

Representations:
Bracket notation
(12.2) \[ S \ [NP\ [Pro\ I]]\ [VP\ [V\ prefer]\ [NP\ [Det\ a]\ [Nom\ [N\ morning]\ [Nom\ [N\ flight]]]]] \]
Non-terminal positional spans
e.g. (NP, 0, 1), (VP, 1, 5), (NP, 2, 5), etc.
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:\(^1\)

- Attachment ambiguity: we eat sushi with chopsticks,
  I shot an elephant in my pajamas.

\(^1\)Examples borrowed from Dan Klein
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:\(^1\)

- Attachment ambiguity: *we eat sushi with chopsticks*,
  *I shot an elephant in my pajamas.*
- Modifier scope: *southern food store*

\(^1\)Examples borrowed from Dan Klein
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:

- Attachment ambiguity: we eat sushi with chopsticks,
  I shot an elephant in my pajamas.
- Modifier scope: southern food store
- Particle versus preposition: The puppy tore up the staircase.

---

1Examples borrowed from Dan Klein
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:\textsuperscript{1}

- Attachment ambiguity: we eat sushi with chopsticks,
  I shot an elephant in my pajamas.
- Modifier scope: southern food store
- Particle versus preposition: The puppy tore up the staircase.
- Complement structure: The tourists objected to the guide that they couldn’t hear.

\textsuperscript{1}Examples borrowed from Dan Klein
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:\(^1\)

- **Attachment ambiguity**: we eat sushi with chopsticks,
  I shot an elephant in my pajamas.
- **Modifier scope**: southern food store
- **Particle versus preposition**: The puppy tore up the staircase.
- **Complement structure**: The tourists objected to the guide
  that they couldn’t hear.
- **Coordination scope**: “I see,” said the blind man, as he
  picked up the hammer and saw.

\(^1\)Examples borrowed from Dan Klein
Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:\(^1\)

- Attachment ambiguity: we eat sushi with chopsticks, I shot an elephant in my pajamas.
- Modifier scope: southern food store
- Particle versus preposition: The puppy tore up the staircase.
- Complement structure: The tourists objected to the guide that they couldn’t hear.
- Coordination scope: “I see,” said the blind man, as he picked up the hammer and saw.
- Multiple gap constructions: The chicken is ready to eat

---

\(^1\)Examples borrowed from Dan Klein
Attachment ambiguity

Probability of attachment sites

- [ imposed [ a ban [ on asbestos ]]]
- [ imposed [ a ban ][ on asbestos ]]
Attachment ambiguity

Probability of attachment sites

- [ imposed [ a ban [ on asbestos ]]]
- [ imposed [ a ban ][ on asbestos ]]

Include head of embedded NP

- ...[ it [ would end [ its venture [with Maserati]]]]
- ...[ it [ would end [ its venture ][with Maserati]]]
Attachment ambiguity

Probability of attachment sites

- [ imposed [ a ban [ on asbestos ]] ]
- [ imposed [ a ban ][ on asbestos ]]

Include head of embedded NP

- ...[ it [ would end [ its venture [with Maserati]]]]
- ...[ it [ would end [ its venture ][with Maserati]]]

Resolve multiple ambiguities simultaneously

- Cats scratch people with claws with knives
Ambiguities make parsing hard

• 1. Computationally: how to reuse work across combinatorially many trees?

• 2. How to make good attachment decisions?
Parsing with a CFG

- Task: given text and a CFG, answer:
  - Does there exist at least one parse?
  - Enumerate parse (backpointers)

- Approaches: top-down, left-to-right, bottom-up

- CKY (Cocke-Kasami-Younger) algorithm
  - Bottom-up dynamic programming:
    Find possible nonterminals for short spans of sentence, then possible combinations for higher spans
  - Requires converting CFG to Chomsky Normal Form
    a.k.a. binarization: <=2 nonterminals in expansion
    - instead of NP -> NP CC NP, could do:
      - NP -> NP_CC NP
      - NP_CC -> NP CC
For cell \([i,j]\) (loop through them bottom-up)
   For possible splitpoint \(k=(i+1)\ldots(j-1)\):
      For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
      If exists rule \(A \rightarrow B C\),
         \(\text{add } A \text{ to cell } [i,j] \) \(\text{(Recognizer)}\)
      ... or ...
      \(\text{add } (A,B,C, k) \text{ to cell } [i,j] \) \(\text{(Parser)}\)
For cell $[i,j]$ (loop through them bottom-up)
For possible splitpoint $k=(i+1)..(j-1)$:
For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
If exists rule $A \rightarrow B \ C$,
\begin{itemize}
  \item add $A$ to cell $[i,j]$ \textbf{(Recognizer)}
  \item ... or ...
  \item add $(A,B,C, \ k)$ to cell $[i,j]$ \textbf{(Parser)}
\end{itemize}

\textbf{Recognizer:} per span, record list of possible nonterminals

\textbf{Parser:} per span, record possible ways the nonterminal was constructed.
For cell $[i,j]$ (loop through them bottom-up)
For possible splitpoint $k=(i+1)\ldots(j-1)$:
   For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
      If exists rule $A \rightarrow B \ C$,
      \[\text{add } A \text{ to cell } [i,j] \text{ (Recognizer)}\]
      \[\ldots \text{ or } \ldots\]
      \[\text{add } (A,B,C, k) \text{ to cell } [i,j] \text{ (Parser)}\]

$\text{Recognizer:}$ per span, record list of possible nonterminals

$\text{Parser:}$ per span, record possible ways the nonterminal was constructed.
For cell $[i,j]$ (loop through them bottom-up)
   For possible splitpoint $k=(i+1)..(j-1)$:
      For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
         If exists rule $A \rightarrow B \ C$,
            add $A$ to cell $[i,j]$ (Recognizer)
      ... or ...
      add $(A,B,C, k)$ to cell $[i,j]$ (Parser)

Recognizer: per span, record list of possible nonterminals

Parser: per span, record possible ways the nonterminal was constructed.
**Grammar**

<table>
<thead>
<tr>
<th>Rule</th>
<th>NP</th>
<th>Adj</th>
</tr>
</thead>
<tbody>
<tr>
<td>yummy</td>
<td>foods</td>
<td>store</td>
</tr>
<tr>
<td>foods</td>
<td>NP</td>
<td>Adj</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>Adj</td>
</tr>
<tr>
<td>NP</td>
<td>Adj</td>
<td>NP</td>
</tr>
</tbody>
</table>

For cell $[i,j]$ (loop through them bottom-up)

For possible splitpoint $k=(i+1)\ldots(j-1)$:

For every $B$ in $[i,k]$ and $C$ in $[k,j]$,

If exists rule $A \rightarrow B \ C$,

*add* $A$ to cell $[i,j]$  (*Recognizer*)

... or ...

*add* $(A,B,C,k)$ to cell $[i,j]$  (*Parser*)

**Recognizer**: per span, record list of possible nonterminals

**Parser**: per span, record possible ways the nonterminal was constructed.
Grammar
Adj -> yummy
NP -> foods
NP -> store
NP -> NP NP
NP -> Adj NP

For cell [i,j] (loop through them bottom-up)
For possible splitpoint k=(i+1)..<j-1):
For every B in [i,k] and C in [k,j],
If exists rule A -> B C,
"add" A to cell [i,j] (Recognizer)
... or ...
"add" (A,B,C, k) to cell [i,j] (Parser)
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B \ C\),
add \(A\) to cell \([i,j]\) (Recognizer)
... or ...
add \((A,B,C, k)\) to cell \([i,j]\) (Parser)
For cell $[i,j]$ (loop through them bottom-up)
For possible splitpoint $k=(i+1)\ldots(j-1)$:
For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
If exists rule $A \to B C$,
add $A$ to cell $[i,j]$  \textit{(Recognizer)}
... or ...
add $(A,B,C, k)$ to cell $[i,j]$  \textit{(Parser)}

\textit{Recognizer}: per span, record list of possible nonterminals

\textit{Parser}: per span, record possible ways the nonterminal was constructed.
For cell \([i,j]\) (loop through them bottom-up)
   For possible splitpoint \(k=(i+1)\ldots(j-1)\):
      For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
         If exists rule \(A \rightarrow B C\),
            add \(A\) to cell \([i,j]\)  (Recognizer)
            ...
            ... or ...
            add \((A,B,C, k)\) to cell \([i,j]\)  (Parser)
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
  For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
    If exists rule \(A \rightarrow B\ C\),
      \text{add} \(A\) to cell \([i,j]\) \(\text{Recognizer}\)
      \ldots or \ldots
      \text{add} \((A,B,C, k)\) to cell \([i,j]\) \(\text{Parser}\)
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
   For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
      If exists rule \(A \rightarrow B\ C\),
         add \(A\) to cell \([i,j]\) \((Recognizer)\)
         ... or ...
         add \((A,B,C, k)\) to cell \([i,j]\) \((Parser)\)
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)..(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B \ C\),
    \(\text{add } A\) to cell \([i,j]\) \((\text{Recognizer})\)
... or ...
\(\text{add } (A,B,C, \ k)\) to cell \([i,j]\) \((\text{Parser})\)
For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]  (Recognizer)
  ... or ...
  add (A,B,C, k) to cell [i,j] (Parser)
For cell [i,j] (loop through them bottom-up)
For possible splitpoint k=(i+1)..(j-1):
For every B in [i,k] and C in [k,j],
If exists rule A -> B C,
add A to cell [i,j]  (Recognizer)
... or ...
add (A,B,C, k) to cell [i,j]  (Parser)
For cell \([i,j]\) (loop through them bottom-up):
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
  For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
    If exists rule \(A \rightarrow B \ C\),
      \textbf{add} \(A\) to cell \([i,j]\) \textbf{(Recognizer)}
    ... or ...
      \textbf{add} \((A,B,C, \ k)\) to cell \([i,j]\) \textbf{(Parser)}
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
  For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
  If exists rule \(A \rightarrow B C\),
  \textit{add} \(A\) to cell \([i,j]\) \textit{(Recognizer)}
  \ldots or \ldots
  \textit{add} \((A,B,C, k)\) to cell \([i,j]\) \textit{(Parser)}
For cell \([i,j]\) (loop through them bottom-up)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B \ C\),
\[add\] \(A\) to cell \([i,j]\) (Recognizer)
... or ...
\[add\] \((A,B,C, \ k)\) to cell \([i,j]\) (Parser)
For cell [i,j]
   For possible splitpoint k=(i+1)..(j-1):
      For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
      add A to cell [i,j]

How do we fill in C(1,2)?

Computational Complexity?
For cell \([i,j]\)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow BC\),
\textit{add} \(A\) to cell \([i,j]\)

How do we fill in \(C(1,2)\)?
Put together \(C(1,1)\) and \(C(2,2)\).
For cell \([i,j]\)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B\ C\),
\textit{add} \(A\) to cell \([i,j]\)

How do we fill in \(C(1,3)\)?
How do we fill in $C(1,3)$?

One way ...

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>n</th>
</tr>
</thead>
</table>

For cell $[i,j]$
- For possible splitpoint $k=(i+1)\ldots(j-1)$:
  - For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
  - If exists rule $A \rightarrow B \ C$,
    - add $A$ to cell $[i,j]$
For cell \([i,j]\)
For possible splitpoint \(k = (i+1) \ldots (j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B\ C\),
add \(A\) to cell \([i,j]\)

How do we fill in \(C(1,3)\)?

One way ...
Another way.

Computational Complexity?
For cell \([i,j]\)
  For possible splitpoint \(k=(i+1)\ldots(j-1)\):
    For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
    If exists rule \(A \rightarrow B C\),
    \textit{add} \(A\) to cell \([i,j]\)

How do we fill in \(C(1,n)\)?

Computational Complexity?
For cell $[i,j]$
  For possible splitpoint $k=(i+1)\ldots(j-1)$:
    For every $B$ in $[i,k]$ and $C$ in $[k,j]$,
    If exists rule $A \rightarrow B \ C$,
    \textit{add} $A$ to cell $[i,j]$

How do we fill in $C(1,n)$?

$n - 1$ ways!
For cell \([i,j]\)
For possible splitpoint \(k=(i+1)\ldots(j-1)\):
For every \(B\) in \([i,k]\) and \(C\) in \([k,j]\),
If exists rule \(A \rightarrow B\ C\),
\[\text{add } A\] to cell \([i,j]\)

How do we fill in \(C(1,n)\)?

\(n - 1\) ways!

Computational Complexity?

\[O(G \ n^3)\]
\[G = \text{grammar constant}\]
# Probabilistic CFGs

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>.80</td>
</tr>
<tr>
<td>$S \rightarrow Aux\ NP\ VP$</td>
<td>.15</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>.05</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>.35</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>.30</td>
</tr>
<tr>
<td>$NP \rightarrow Det\ Nominal$</td>
<td>.20</td>
</tr>
<tr>
<td>$NP \rightarrow Nominal$</td>
<td>.15</td>
</tr>
<tr>
<td>$Nominal \rightarrow Noun$</td>
<td>.75</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal\ Noun$</td>
<td>.20</td>
</tr>
<tr>
<td>$Nominal \rightarrow Nominal\ PP$</td>
<td>.05</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>.35</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP$</td>
<td>.20</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP\ PP$</td>
<td>.10</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ PP$</td>
<td>.15</td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP\ NP$</td>
<td>.05</td>
</tr>
<tr>
<td>$VP \rightarrow VP\ PP$</td>
<td>.15</td>
</tr>
<tr>
<td>$PP \rightarrow Preposition\ NP$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Det \rightarrow that$</td>
<td>.10</td>
</tr>
<tr>
<td>$Det \rightarrow a$</td>
<td>.30</td>
</tr>
<tr>
<td>$Det \rightarrow the$</td>
<td>.60</td>
</tr>
<tr>
<td>$Noun \rightarrow book$</td>
<td>.10</td>
</tr>
<tr>
<td>$Noun \rightarrow flight$</td>
<td>.30</td>
</tr>
<tr>
<td>$Noun \rightarrow meal$</td>
<td>.15</td>
</tr>
<tr>
<td>$Noun \rightarrow money$</td>
<td>.05</td>
</tr>
<tr>
<td>$Noun \rightarrow flights$</td>
<td>.40</td>
</tr>
<tr>
<td>$Noun \rightarrow dinner$</td>
<td>.10</td>
</tr>
<tr>
<td>$Verb \rightarrow book$</td>
<td>.30</td>
</tr>
<tr>
<td>$Verb \rightarrow include$</td>
<td>.30</td>
</tr>
<tr>
<td>$Verb \rightarrow prefer$</td>
<td>.40</td>
</tr>
<tr>
<td>$Pronoun \rightarrow I$</td>
<td>.40</td>
</tr>
<tr>
<td>$Pronoun \rightarrow she$</td>
<td>.05</td>
</tr>
<tr>
<td>$Pronoun \rightarrow me$</td>
<td>.15</td>
</tr>
<tr>
<td>$Pronoun \rightarrow you$</td>
<td>.40</td>
</tr>
<tr>
<td>$Proper-Noun \rightarrow Houston$</td>
<td>.60</td>
</tr>
<tr>
<td>$Proper-Noun \rightarrow TWA$</td>
<td>.40</td>
</tr>
<tr>
<td>$Aux \rightarrow does$</td>
<td>.60</td>
</tr>
<tr>
<td>$Aux \rightarrow can$</td>
<td>.40</td>
</tr>
<tr>
<td>$Preposition \rightarrow from$</td>
<td>.30</td>
</tr>
<tr>
<td>$Preposition \rightarrow to$</td>
<td>.30</td>
</tr>
<tr>
<td>$Preposition \rightarrow on$</td>
<td>.20</td>
</tr>
<tr>
<td>$Preposition \rightarrow near$</td>
<td>.15</td>
</tr>
<tr>
<td>$Preposition \rightarrow through$</td>
<td>.05</td>
</tr>
</tbody>
</table>

- Defines a probabilistic generative process for words in a sentence
- Extension of HMMs, strictly speaking
- Learning?
  - Fully supervised: need a treebank
  - Unsupervised: with EM
(P)CFG model, (P)CKY algorithm

- CKY: given CFG and sentence w
  - Does there exist at least one parse?
  - Enumerate parses (backpointers)

- Weighted CKY: given PCFG and sentence w
  - \( \Rightarrow \) Viterbi parse
    \[ \arg\max_y P(y \mid w) = \arg\max_y P(y, w) \]
  - Inside-outside: Likelihood of sentence \( P(w) \)
• Parsers’ computational efficiency
  • Grammar constant; pruning & heuristic search
  • $O(N^3)$ for CKY (ok? depends...)
  • $O(N)$ [or so...]: left-to-right incremental algorithms

• Parsing model accuracy: still lots of ambiguity!!
  • PCFGs lack lexical information to resolve attachment decisions
  • Vanilla PCFG accuracy: 70-80%
Rules vs. Annotations

• In the old days: hand-built grammars. Difficult to scale.

• Annotation-driven sup. learning

• ~1993: Penn Treebank

• Construct PCFG (or whatever) with supervised learning

• (Cool open research: unsupervised learning?)
Treebanks

- Penn Treebank (constituents, English)
  - [http://www.cis.upenn.edu/~treebank/home.html](http://www.cis.upenn.edu/~treebank/home.html)
- Recent revisions in Ononotes
- Universal Dependencies
  - [http://universaldependencies.org/](http://universaldependencies.org/)
- Prague Treebank (syn+sem)
- many others...
- Know what you’re getting!
Ambiguities make parsing hard

• 1. Computationally: how to reuse work across combinatorially many trees?

• 2. **How to make good attachment decisions?**
  • Enrich PCFG with
    • parent information: what’s above me?
    • lexical information via head rules
      • VP[fight]: a VP headed by “fight”
  • (or better, word/phrase embedding-based generalizations: e.g. recurrent neural network grammars (RNNGs))
Parent annotation

All NPs

NPs under S

NPs under VP

NP PP  DT NN  PRP  NP PP  DT NN  PRP  NP PP  DT NN  PRP

11%  9%  6%  9%  9%  21%  23%  7%  4%
Adding Headwords to Trees

Lexicalization

S
  NP
    DT the
    NN lawyer
  VP
    Vt questioned
      NP
        DT the
        NN witness

↓

S(questioned)
  NP(lawyer)
    DT the
    NN lawyer
  VP(questioned)
    Vt questioned
      NP(witness)
        DT the
        NN witness
Head rules

• Idea: Every phrase has a head word
• Head rules: for every nonterminal in tree, choose one of its children to be its “head”. This will define head words.
• Every nonterminal type has a different head rule; e.g. from Collins (1997):

  • If parent is NP,
    • Search from right-to-left for first child that’s NN, NNP, NNPS, NNS, NX, JJR
    • Else: search left-to-right for first child which is NP