CMSC 723: Computational Linguistics I — Session #6

Syntax and Context-Free Grammars

Jimmy Lin
The iSchool
University of Maryland

Wednesday, October 7, 2009

Today's Agenda

- Words… **structure…** meaning…
- Formal Grammars
  - Context-free grammar
  - Grammars for English
  - Treebanks
  - Dependency grammars
- Next week: parsing algorithms

Grammar and Syntax

- By grammar, or syntax, we mean implicit knowledge of a native speaker
  - Acquired by around three years old, without explicit instruction
  - It’s already inside our heads, we’re just trying to formally capture it
- **Not** the kind of stuff you were later taught in school:
  - Don’t split infinitives
  - Don’t end sentences with prepositions

Syntax

- Why should you care?
- Syntactic analysis is a key component in many applications
  - Grammar checkers
  - Conversational agents
  - Question answering
  - Information extraction
  - Machine translation
  - …

Constituency

- Basic idea: groups of words act as a single unit
- Constituents form coherent classes that behave similarly
  - With respect to their internal structure: e.g., at the core of a noun phrase is a noun
  - With respect to other constituents: e.g., noun phrases generally occur before verbs

Constituency: Example

- The following are all noun phrases in English...

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- Why?
  - They can all precede verbs
  - They can all be preposed
  - …

Grammars and Constituency

- For a particular language:
  - What are the “right” set of constituents?
  - What rules govern how they combine?
- Answer: not obvious and difficult
  - That’s why there are so many different theories of grammar and competing analyses of the same data!
- Approach here:
  - Very generic
  - Focus primarily on the “machinery”
  - Doesn’t correspond to any modern linguistic theory of grammar

Context-Free Grammars

- Context-free grammars (CFGs)
  - Aka phrase structure grammars
  - Aka Backus-Naur form (BNF)
- Consist of
  - Rules
  - Terminals
  - Non-terminals

Context-Free Grammars

- Terminals
  - We'll take these to be words (for now)
- Non-Terminals
  - The constituents in a language (e.g., noun phrase)
- Rules
  - Consist of a single non-terminal on the left and any number of terminals and non-terminals on the right

Some NP Rules

- Here are some rules for our noun phrases

$$
\begin{aligned}
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow ProperNoun \\
Nominal &\rightarrow Noun \mid Nominal\ Noun
\end{aligned}
$$

- Rules 1 & 2 describe two kinds of NPs:
  - One that consists of a determiner followed by a nominal
  - Another that consists of proper names
- Rule 3 illustrates two things:
  - An explicit disjunction
  - A recursive definition

L_0 Grammar

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

CFG: Formal definition

N a set of **non-terminal symbols** (or **variables**)

Σ a set of **terminal symbols** (disjoint from N)

R a set of **rules** or productions, each of the form $A \rightarrow \beta$,
where A is a non-terminal,
β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$

S a designated **start symbol**

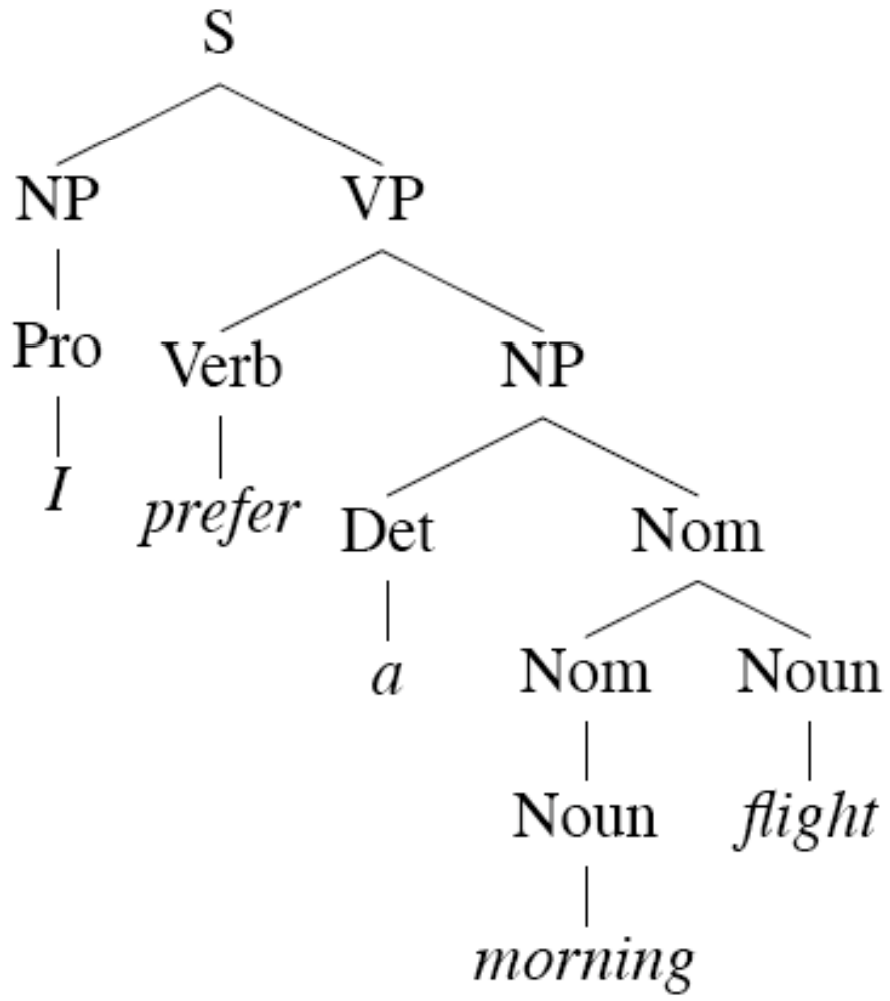
Three-fold View of CFGs

- Generator
- Acceptor
- Parser

Derivations and Parsing

- A derivation is a sequence of rules applications that
  - Covers all tokens in the input string
  - Covers only the tokens in the input string
- Parsing: given a string and a grammar, recover the derivation
  - Derivation can be represented as a parse tree
  - Multiple derivations?

Parse Tree: Example



Note: equivalence between parse trees and bracket notation

Natural vs. Programming Languages

- Wait, don’t we do this for programming languages?
- What’s similar?
- What’s different?

An English Grammar Fragment

- Sentences
- Noun phrases
  - Issue: agreement
- Verb phrases
  - Issue: subcategorization

Sentence Types

- Declaratives: A plane left.

  $S \rightarrow NP\ VP$

- Imperatives: Leave!

  $S \rightarrow VP$

- Yes-No Questions: Did the plane leave?

  $S \rightarrow Aux\ NP\ VP$

- WH Questions: When did the plane leave?

  $S \rightarrow WH\text{-}NP\ Aux\ NP\ VP$

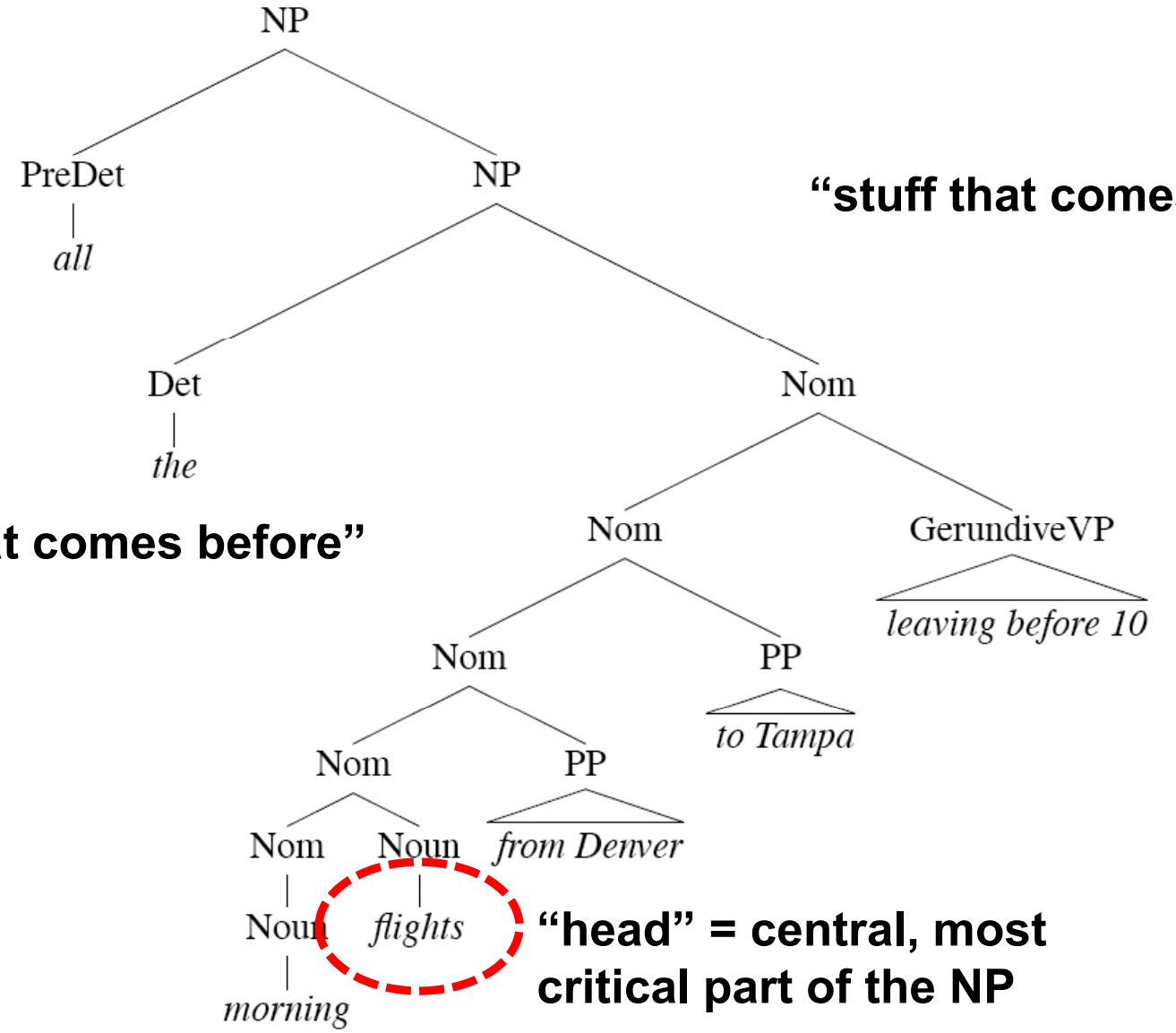
Noun Phrases

- Let's consider these rules in detail:

$$
\begin{aligned}
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow ProperNoun \\
Nominal &\rightarrow Noun \mid Nominal\ Noun
\end{aligned}
$$

- NPs are a bit more complex than that!
  - Consider: "All the morning flights from Denver to Tampa leaving before 10"

A Complex Noun Phrase



“stuff that comes after”

“stuff that comes before”

“head” = central, most critical part of the NP

Determiners

- Noun phrases can start with determiners...
- Determiners can be
  - Simple lexical items: the, this, a, an, etc. (e.g., “a car”)
  - Or simple possessives (e.g., “John’s car”)
  - Or complex recursive versions thereof (e.g., John’s sister’s husband’s son’s car)

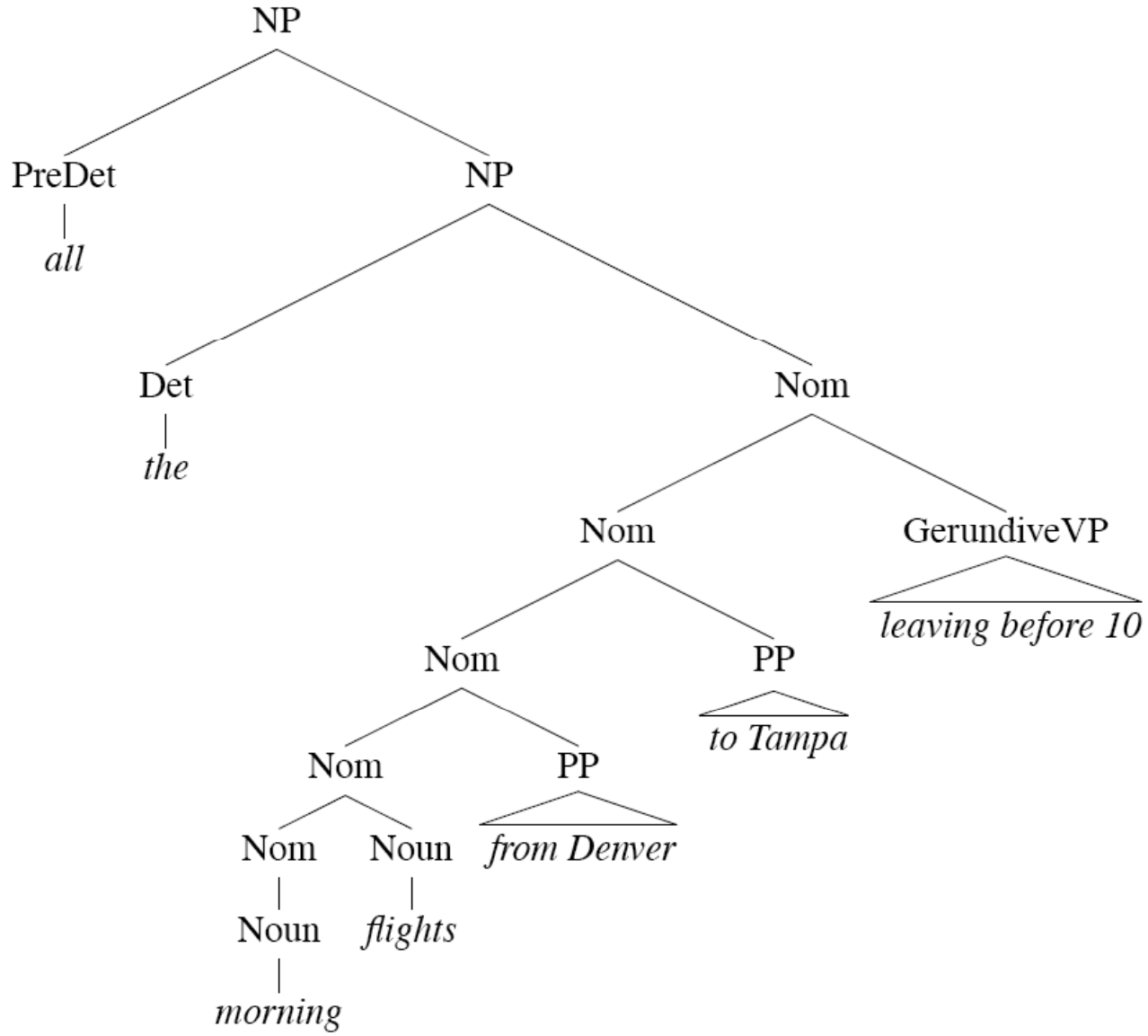
Premodifiers

- Come before the head
- Examples:
  - Cardinals, ordinals, etc. (e.g., “three cars”)
  - Adjectives (e.g., “large car”)
- Ordering constraints
  - “three large cars” vs. “?large three cars”

Postmodifiers

- Naturally, come after the head
- Three kinds
  - Prepositional phrases (e.g., “from Seattle”)
  - Non-finite clauses (e.g., “arriving before noon”)
  - Relative clauses (e.g., “that serve breakfast”)
- Similar recursive rules to handle these
  - Nominal $\rightarrow$ Nominal PP
  - Nominal $\rightarrow$ Nominal GerundVP
  - Nominal $\rightarrow$ Nominal RelClause

A Complex Noun Phrase Revisited



Agreement

- Agreement: constraints that hold among various constituents
- Example, number agreement in English

This flight	***This flights**
Those flights	***Those flight**
One flight	***One flights**
Two flights	***Two flight**

Problem

- Our NP rules don’t capture agreement constraints
  - Accepts grammatical examples (this flight)
  - Also accepts ungrammatical examples (*these flight)
- Such rules **overgenerate**
  - We’ll come back to this later

Verb Phrases

- English verb phrases consists of
  - Head verb
  - Zero or more following constituents (called arguments)
- Sample rules:

$VP \rightarrow Verb$ disappear
$VP \rightarrow Verb\ NP$ prefer a morning flight
$VP \rightarrow Verb\ NP\ PP$ leave Boston in the morning
$VP \rightarrow Verb\ PP$ leaving on Thursday

Subcategorization

- Not all verbs are allowed to participate in all VP rules
  - We can subcategorize verbs according to argument patterns (sometimes called “frames”)
  - Modern grammars may have 100s of such classes
- This is a finer-grained articulation of traditional notions of transitivity

Subcategorization

- Sneeze: John sneezed
- Find: Please find [a flight to NY]$_{NP}$
- Give: Give [me]$_{NP}$ [a cheaper fare]$_{NP}$
- Help: Can you help [me]$_{NP}$ [with a flight]$_{PP}$
- Prefer: I prefer [to leave earlier]$_{TO\text{-}VP}$
- Told: I was told [United has a flight]$_{S}$
- …

Subcategorization

- Subcategorization at work:
  - *John sneezed the book
  - *I prefer United has a flight
  - *Give with a flight
- But some verbs can participate in multiple frames:
  - I ate
  - I ate the apple
- How do we formally encode these constraints?

Why?

- As presented, the various rules for VPs overgenerate:

$VP \rightarrow Verb$ disappear
$VP \rightarrow Verb\ NP$ prefer a morning flight
$VP \rightarrow Verb\ NP\ PP$ leave Boston in the morning
$VP \rightarrow Verb\ PP$ leaving on Thursday

- John sneezed [the book]$_{NP}$
  - Allowed by the second rule…

Possible CFG Solution

- Encode agreement in non-terminals:
  - SgS $\rightarrow$ SgNP SgVP
  - PlS $\rightarrow$ PlNP PlVP
  - SgNP $\rightarrow$ SgDet SgNom
  - PlNP $\rightarrow$ PlDet PlNom
  - PlVP $\rightarrow$ PlV NP
  - SgVP $\rightarrow$ SgV Np
- Can use the same trick for verb subcategorization

Possible CFG Solution

- Critique?
  - It works…
  - But it's ugly…
  - And it doesn't scale (explosion of rules)
- Alternatives?
  - Multi-pass solutions

Three-fold View of CFGs

- Generator
- Acceptor
- Parser

The Point

- CFGs have about just the right amount of machinery to account for basic syntactic structure in English
  - Lot’s of issues though...
- Good enough for many applications!
  - But there are many alternatives out there…

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree
  - Hopefully the right one!
- These are generally created:
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary
- But…
  - Detailed annotation guidelines are needed
  - Explicit instructions for dealing with particular constructions

Penn Treebank

- Penn TreeBank is a widely used treebank
  - 1 million words from the Wall Street Journal
- Treebanks implicitly define a grammar for the language

Penn Treebank: Example

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) (`` ``)
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Treebank Grammars

- Such grammars tend to be very flat
  - Recursion avoided to ease annotators burden
- Penn Treebank has 4500 different rules for VPs, including…
  - VP $\rightarrow$ VBD PP
  - VP $\rightarrow$ VBD PP PP
  - VP $\rightarrow$ VBD PP PP PP
  - VP $\rightarrow$ VBD PP PP PP PP

Why treebanks?

- Treebanks are critical to training statistical parsers
- Also valuable to linguist when investigating phenomena

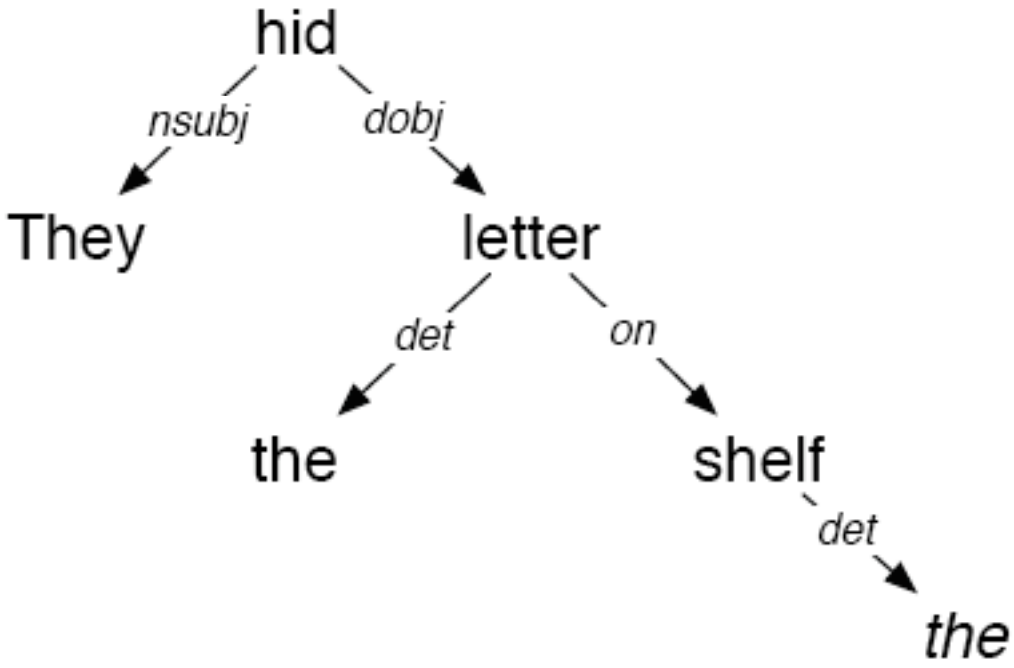
Dependency Grammars

- CFGs focus on constituents
  - Non-terminals don’t actually appear in the sentence
  - So what if you got rid of them?
- In dependency grammar, a parse is a graph where:
  - Nodes represent words
  - Edges represent dependency relations between words (typed or untyped, directed or undirected)

Dependency Relations

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Example Dependency Parse



They hid the letter on the shelf

Compare with constituent parse… What's the relation?

Summary

- CFG can be used to capture various facts about the structure of language
  - Agreement and subcategorization cause problems…
  - And there are alternative formalisms
- Treebanks as an important resource for NLP
- Next week: parsing