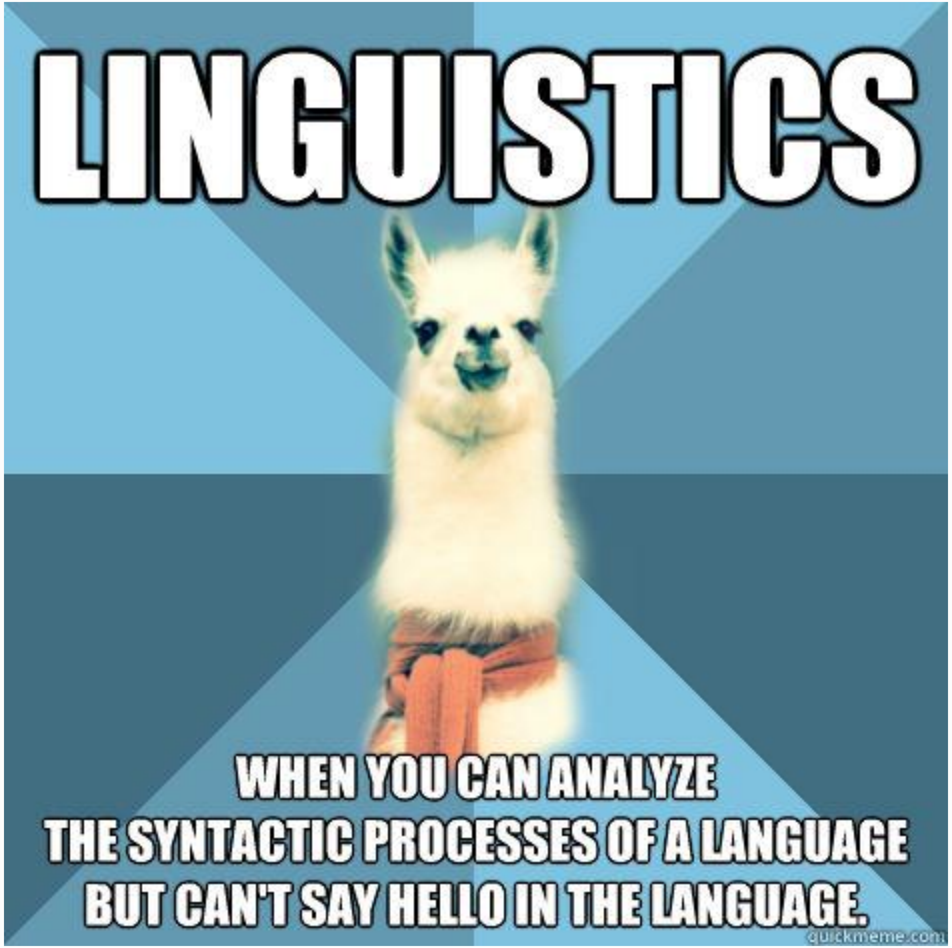


Source:
http://www.quickmeme.com/meme/tbt

CS 2731 Introduction to Natural Language Processing

Session 20: Constituency parsing, CFGs

Michael Miller Yoder

November 6, 2023

University of Pittsburgh | School of Computing and Information

Course logistics

- Homework 4 is due **this Thu 11-09**
  - Part 1: Complete Viterbi tables/lattices for 2 example sentences given HMM probabilities
  - Part 2: Fine-tune BERT-based models and evaluate on Spanish NER data
- Basic working project system **due next Thu 11-16**
- Su Lin Blodgett, author of Blodgett et al. 2020, "Language (Technology) is Power" paper is giving a talk at CMU
  - 12:30-1:50pm, this Fri 11-10 in CMU Posner Hall A35
  - Let Michael know if you're interested in going

Core tasks and applications of NLP

CORE TASKS

- text classification
- representation learning

MODULE 2

- language modeling

MODULE 3

- sequence labeling

MODULE 4

- syntactic parsing

MODULE 5

APPLICATIONS

- machine translation
- speech recognition & synthesis
- chatbots
- computational social science

Overview: Constituency parsing, CFGs

- What is syntax?
- Syntax formalisms (constituency, dependency)
- Constituency
  - Types of constituents in English
  - Context-free grammars (CFGs)
  - Derivations and parse trees
- Constituency parsing
  - Treebanks
- Activity: parse some sentences

What is syntax and why is it useful?

Syntax is Sentence and Phrase Structure

- Syntax concerns the patterns according to which words are combined to form phrases and sentences.
- It is distinct from morphology (how morphemes combine to form words) and semantics (what sentences mean) but is related to both.
- *Colorless green ideas sleep furiously.*

Slide credit: David Mortensen

Syntax is the Door to Semantics

- To arrive at a semantic interpretation of a sentence, you have to know its syntax
- Parallel with programming languages
    - Semantics different from syntax (form versus function)
    - But semantics follows from syntax

Slide credit: David Mortensen

Who Did What to Whom?

If you want to know who did what to whom with what thing having what properties, you must have access to syntax in some form.

Different perspectives on syntax

There are Two Major Approaches to Syntax in NLP

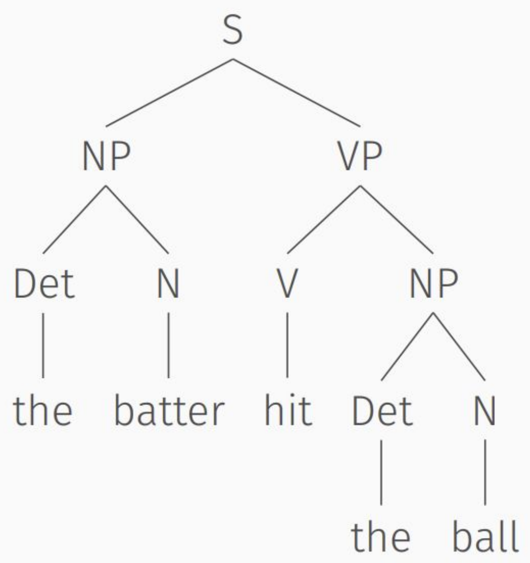
Two approaches:

- Syntax means taking sentences, dividing them into phrases and dividing those phrases into smaller phases until you arrive at individual words, yielding a tree of “constitutents”
- Syntax means taking sentences and characterizing the relationships between pairs of words in the sentence, yielding a tree or graph of “heads” and “dependents”

Slide credit: David Mortensen

Phrase Structure Grammar is also Called Constituency Grammar

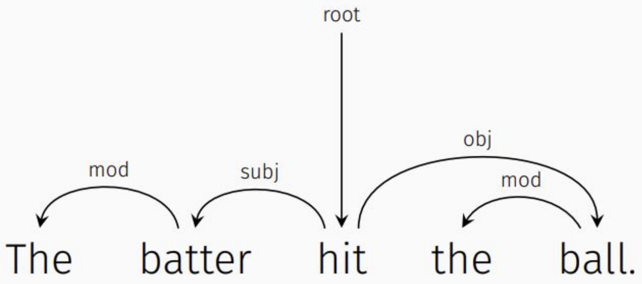
- The first approach is called PHRASE STRUCTURE GRAMMAR or CONSTITUENCY GRAMMAR.
- Basic unit — constituent
- Used by the parsers in the interpreters/compilers of most programming languages



Slide credit: David Mortensen

Dependency Grammar Is Based On Bilexical Dependencies

- The second approach is called DEPENDENCY GRAMMAR.
- Basic element — BILEXICAL DEPENDENCY
- Especially useful for many contemporary NLP tasks



Slide credit: David Mortensen

Constituency

Constituents Are “Clumps” of Words

- Constituents: sequences of one or more words that “go together” (form a unit of which the preceding and following words are not a part)
- Constituents larger than a word are called “phrases”
- Phrases can contain other phrases

Slide credit: David Mortensen

Constituents Can Be Defined Rigorously

- **Rigorous criteria** exist for diagnosing whether a sequence of words forms a constituent
- These are called "**tests for constituency**"
- These include coordination, substitution, topicalization/fronting, and so on

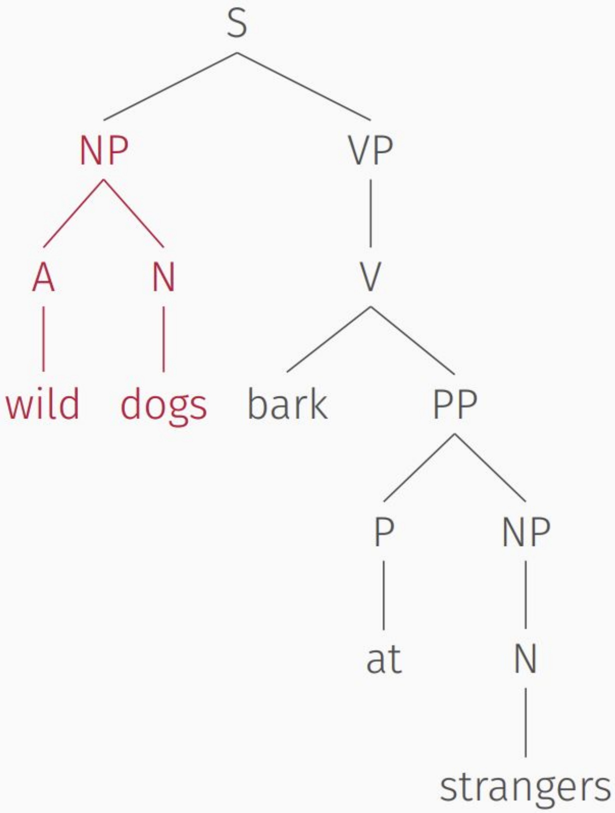
Slide credit: David Mortensen

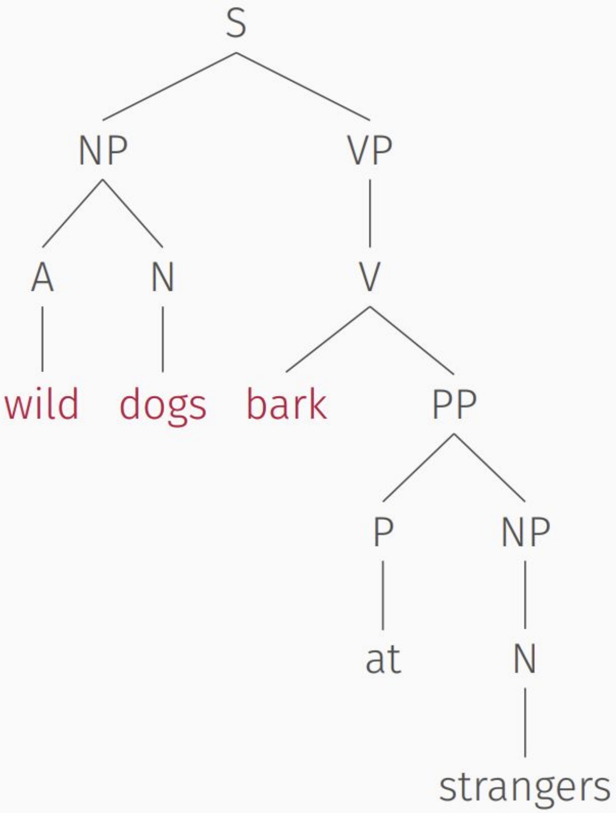
Pro-form substitution tests

Dogs bark at strangers	$\leftrightarrow$	**They** bark at strangers
Wild dogs bark at strangers	$\leftrightarrow$	**They** bark at strangers
The big dogs we saw by the house bark at strangers	$\leftrightarrow$	**They** bark at strangers
Dogs bark at strangers	$\leftrightarrow$	***They** at strangers

Slide credit: David Mortensen

Pro-form substitution trees





Slide credit: David Mortensen

Do-so substitution test

Cows also **eat grass**.	$\leftrightarrow$	Cows also **do so**.
Cows also **eat green grass in lush pastures**.	$\leftrightarrow$	Cows also **do so**.
Cows also **eat green grass** in lush pastures.	$\leftrightarrow$	Cows also **do so** in green pastures.
Cows also **eat green** grass in lush pastures.	$\leftrightarrow$	*Cows also **do so** grass in lush pastures.

Slide credit: David Mortensen

Types of constituents in English

Types of Constituents in English

There are several types of constituents in English

- Words: Nouns, verbs, prepositions, adjectives, etc
- Phrases: Ss, NPs, VPs, PPs, APs, etc.

Following are some of the most important phrasal constituents.

Examples of Noun Phrases

Noun phrases typically, though not always, contain nouns (which form the center or "head" of the phrase). The subjects and objects of verbs and the objects of prepositions are noun phrases.

- **The elephant** arrived.
- **It** arrived.
- **Elephants** arrived.
- **The big ugly elephant** arrived.
- **The elephant I love to hate** arrived.
- **The elephant who ate my Cheetos** arrived.

Slide credit: David Mortensen

Prepositional Phrases (PPs)

Every prepositional phrase contains a preposition (sometimes more than one) and a noun phrase:

- I arrived on **Tuesday**.
- I arrived in **March**.
- I arrived under **the leaking roof**.

Slide credit: David Mortensen

Verb Phrases (VPs)

- Alan **awoke**.
- Alan **scheduled frantically**.
- Alan **scheduled a meeting**.
- Alan **scheduled a meeting with a scientist from Punjab**.
- Alan thought that **you thought that winter was here**.

Slide credit: David Mortensen

Sentences or Clauses (Ss)

- John loves Mary.
- John loves the woman **he thinks is Mary**.
- Sometimes, John thinks **he is Mary**.
- It is patently false that **sometimes John thinks he is Mary**.

Slide credit: David Mortensen

Context-Free Grammars (CFGs)

Context-Free Grammars are 4-Tuples

Context-free grammars are a tuple $\langle N, \Sigma, R, S \rangle$

- N, a finite set of non-terminal symbols
- Σ, a finite set of terminal symbols
- S, a special start symbol $S \in N$
- R, a finite relation in $N \times (N \cup \Sigma)^*$.

Members of R are typically represented as rewrite rules of the form $X \to \alpha$ where

$$X \in N$$
$$\alpha \in (N \cup \Sigma)^*$$

Slide credit: David Mortensen

CFGs Have no Context on the Left-Hand Side

The grammars are called “context-free” because there is no context in the LHS of rules—there is just one symbol.

Slide credit: David Mortensen

Non-Terminal Symbols Form Parent Nodes

A non-terminal symbol is one like S that can (and must!) be rewritten as either

- Other non-terminal symbols
- Terminal symbols

Non-terminals can be phrasal or pre-terminal (in which case they look like part of speech tags—Noun, Verb, etc.)

Slide credit: David Mortensen

Terminal nodes are leaf nodes

- In natural language syntax, terminals are usually words
- They cannot be rewritten; they mean that you're done

A Context-Free Grammar

S → NP VP

NP → Det Noun

VP → Verb NP

Det → the | a

Noun → boy | girl | hotdogs

Verb → likes | hates | eats

Slide credit: David Mortensen

The same context free grammar can be used for both analysis and generation.

Slide credit: David Mortensen

Derivations and parse trees

Some Definitions

Grammatical said of a sentence in the language

Ungrammatical said of a sentence not in the language

Derivation sequence of top-down production steps

Parse tree graphical representation of the derivation

A string is grammatical iff there exists a derivation for it.

Slide credit: David Mortensen

A Grammar Generates a Language

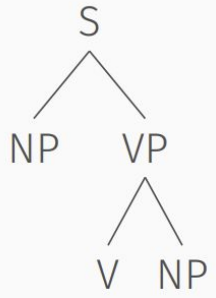
S → NP VP
NP → Det N
VP → V NP
Det → your, my, the, a
N → computer, sentence
V → parsed, misunderstood

Slide credit: David Mortensen

S → NP VP

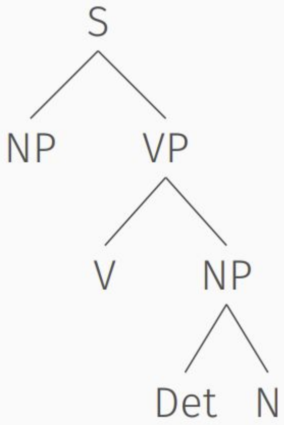
Slide credit: David Mortensen

VP → V NP



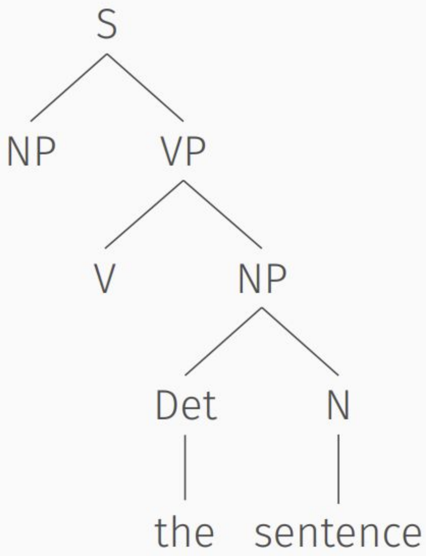
Slide credit: David Mortensen

NP → Det N



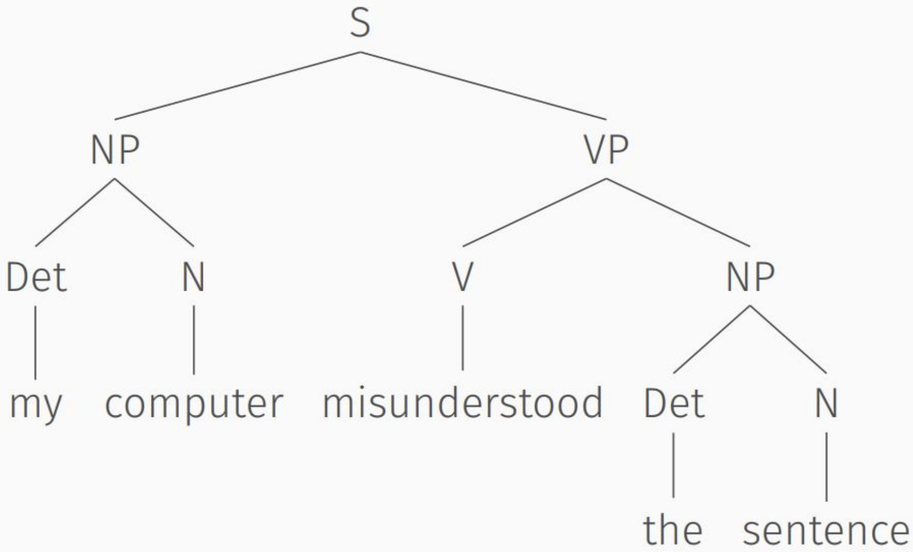
Slide credit: David Mortensen

Det → the
N → sentence



Slide credit: David Mortensen

S → NP VP
NP → Det N
VP → V NP
Det → my
Det → the
N → computer
N → sentence
V → misunderstood



Slide credit: David Mortensen

Constituency parsing

What is a Parser?

Input
- Sentence
- Grammar

Output
- A parse tree (a tree data structure representing the syntactic structure of the sentence)
- Nothing if the sentence is not recognized by the grammar

Parsing is a search problem

- In order to parse a sentence, a parser must find all the possible ways that that sentence could be derived in a grammar. It is searching through a large (potentially infinite space).
- Naïve approaches to this problem are intractable
- Recursive grammars allow for an unbounded number of derivations
- Fortunately, mathematicians and computer scientists have developed a wide range of algorithms for addressing the problems of context-free parsing

Slide adapted from David Mortensen

Constituency parsing algorithms

- A classic algorithm uses dynamic programming:
  Cocke-Younger-Kasami (CYK or CKY) Algorithm (first published by Itiroo Sakai in 1961).
  - Uses a table of partial parses to efficiently come up with parsing options
  - Produces all possible parses, but doesn't tell you which one is best!
- Neural span-based constituency parsing to the rescue:
  - Calculates a score for each constituent
  - Combine scores to find the best-scoring parse tree

Treebanks

Grammars Can Be Encoded Explicitly or Implicitly

Explicit

S → NP VP

NP → Det N

VP → V NP

Det → a | the

N → professor | students

V → delighted | annoyed

As in a hand-crafted grammar.

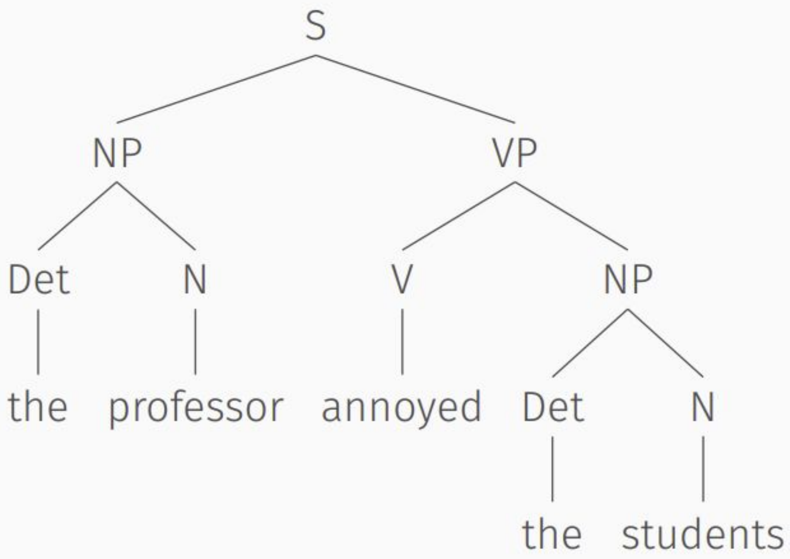
Implicit

```
(S
  (NP
    (Det the)
    (N professor))
  (VP
    (V annoyed)
    (NP
      (Det the)
      (N students))))
```

As in a treebank.

Slide credit: David Mortensen

New notation for constituency trees




```
(S
  (NP
    (Det the)
    (N professor))
  (VP
    (V annoyed)
    (NP
      (Det the)
      (N students))))
```

Slide credit: David Mortensen

The First Big Treebank Was the PTB

Penn Treebank

- Annotation of:
  - Brown corpus
  - ATIS (Air Travel Information Service corpus)
  - Switchboard Corpus
  - Wall Street Journal corpus
- Size: about 3 million words
  - Rules:
  - 17,500 types
  - “Flat”
  - Many types with only one token

Slide credit: David Mortensen

Example Sentence from PTB

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ) )
        (NP-TMP (NNP Nov.) (CD 29) ) ) )
    (. .) ) )
```

Slide credit: David Mortensen

Some PTB Rules by Frequency

40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
...
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP
...

100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
...
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

Slide credit: David Mortensen

Proper Ambivalence toward Treebanks

Why you should have great respect for treebanks

Why you should be cautious around treebanks

Slide credit: David Mortensen

Why You Should Respect Treebanks

Treebanks require great skill

- Expert linguists make thousands of decisions
- Many annotators must remember all of the decisions and use them consistently, including knowing which decision to use
- The “coding manual” containing all of the decisions is hundreds of pages long

Treebanks take many years to make

- Writing the coding manual, training coders, building user-interface tools, etc., all take a lot of time
- So does the actual coding of the data and quality assurance

Treebanks are expensive

Somebody has to secure funding for these projects

Slide credit: David Mortensen

You Should Be Cautious around Treebanks

- They are **too big to fail**
- The are **produced under pressure** of time and funding
- Although most of the decisions are made by experts, **most of the coding is done by non-experts**

Slide credit: David Mortensen

Activity: parse some sentences

Practice: parse these sentences

1. The boy with the coin sat down.
2. Enraged cow injures farmer with ax.
3. Hospitals are sued by seven foot doctors.
4. The woman saw the man with the telescope.

Some helpful rules:

S -> NP VP
S -> ADJP VP

NP -> N
NP -> N N
NP -> Det NP
NP -> ADJ NP
NP -> NP PP

VP -> V
VP -> VP NP
VP -> VP NP PP
VP -> VP VP
VP -> VP PP
VP -> VP Particle

PP -> Prep NP

Wrapping up

- Syntax concerns rules for grouping and ordering words into meaningful phrases and sentences
- Constituencies and dependencies are two high-level formalisms for syntax
- Constituencies are groups of words that act as a unit in a sentence
- Context-free grammars (CFGs) provide grouping rules for constituents
- Constituency parsing is determining the best/possible constituency structure of a sentence
- Treebanks contain sentences annotated for syntax structure and parts of speech

Questions?