Algorithms for Natural Language Processing

Lecture 11: Formal Grammars

WHAT IS SYNTAX?

Syntax Is Not Morphology

- Morphology deals with the internal structure of words
  - Syntax deals with combinations of words
  - Phrases and sentences
- Morphology is often irregular
  - Syntax has its irregularities, but it is usually regular
  - Syntax is mostly made up of general rules that apply across-the-board

Syntax Is Not Semantics

- Semantics is about meaning; syntax is about structure alone
- A sentence can be syntactically well-formed but semantically ill-formed:
  - *Colorless green ideas sleep furiously*.
- Some well-known linguistic theories attempt to "read" semantic representations off of syntactic representations in a compositional fashion
- We'll talk about these in a later lecture

CONSTITUENCY AND ENGLISH PHRASES

Constituency

- One way of viewing the structure of a sentence is as a collection of nested constituents
  - **constituent**: a group of words that "go together" (or relate more closely to one another than to other words in the sentence)
- Constituents larger than a word are called *phrases*
- Phrases can contain other phrases

Noun Phrases (NPs)

- The elephant arrived.
- It arrived.
- Elephants arrived.
- The big ugly elephant arrived.
- The elephant I love to hate arrived.

Prepositional Phrases (PPs)

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

Every **prepositional phrase** contains a **noun phrase**.

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.

CONTEXT-FREE GRAMMARS

Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of non-terminal symbols, N
- Special start symbols, $S \in N$
- Production rules of the form $X \rightarrow \alpha$
  where
  $X \in N$
  $\alpha \in (N \cup \Sigma)^*$

The grammars are called “context-free” because there is no context in the LHS of rules—there is just one symbol. They are equivalent to Backus-Naur form or BNF.

Non-Terminals and Terminals

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

Context-Free Rules

- S → NP VP
- NP → Det Noun
- VP → Verb NP
- Det → the, a
- Noun → boy, girl, hotdogs
- Verb → likes, hates, eats

CFGs as Declarative Programming

- One way to look at context-free grammars is as declarative programs
  - Think Prolog, SQL, or XQuery
  - Instead of specifying how the task is to be accomplished…
    - How sentences are to be generated
    - How sentences are to be parsed
  - …CFGs specify what is to be computed in terms of rules and let generalized computation mechanisms solve for the particular cases
- The same goes for regular expressions as well as other types of grammars

Building Noun Phrases

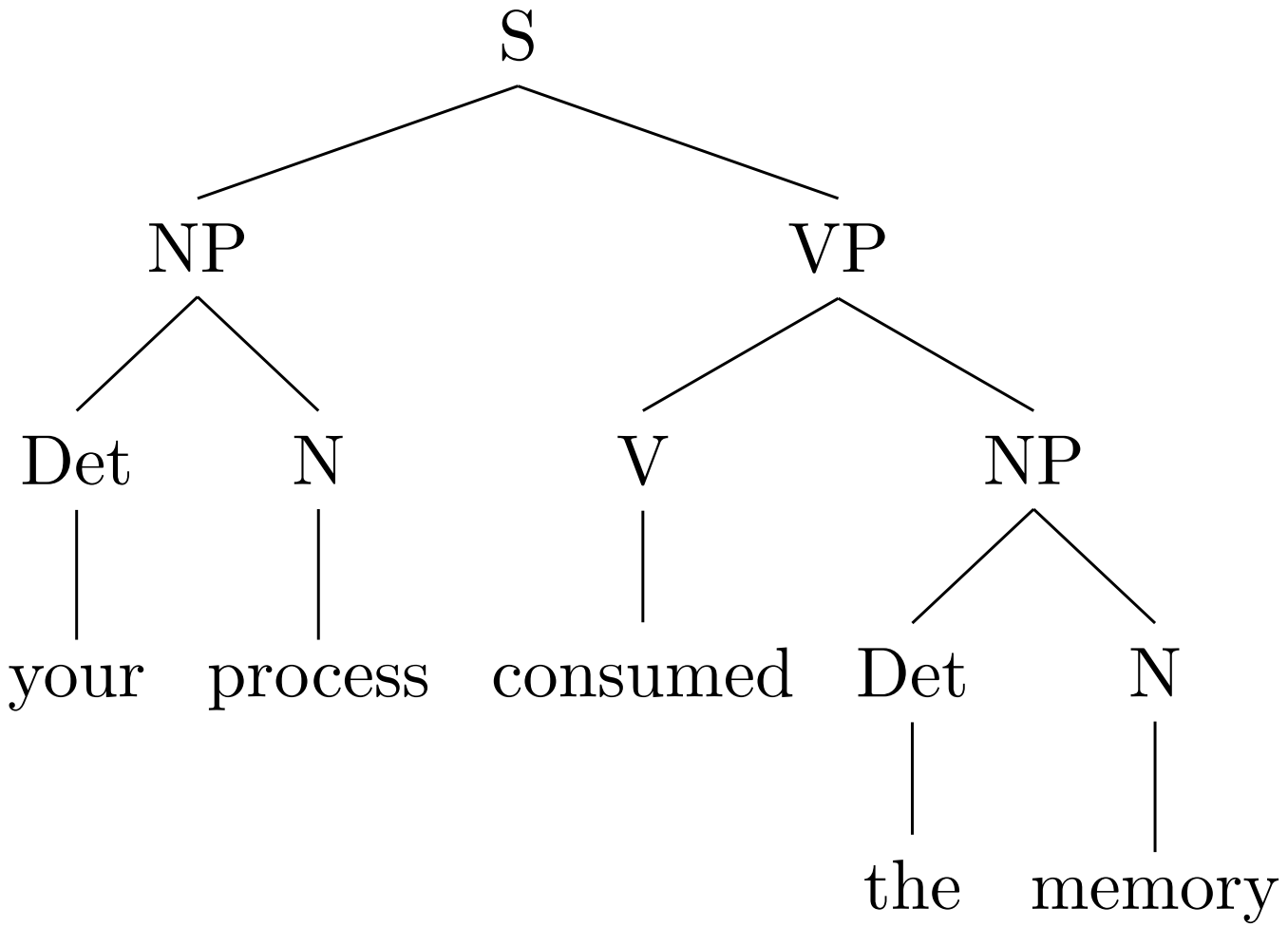
- NP → Determiner NounBar
- NP →ProperNoun
- NounBar → Noun
- NounBar → AP NounBar
- NounBar → NounBar PP
- AP → Adj AP
- AP → Adj
- PP → Preposition NP

Terminology

- 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.

A (Constituency) Parse Tree



Ambiguity

- S → NP VP
- NP → Det Noun
- VP → Verb NP
- VP → VP PP
- PP → Prep NP
- Det → the, a
- Noun → boy, girl, hotdogs, park
- Verb → likes, hates, eats, sees
- Prep → in, with

Grammaticality—It Varies

- I'll write the company
- I'll write to the company
- It needs to be washed
- It needs washed
- They met Friday to discuss it
- They met on Friday to discuss it

On Getting it Right

- CFGs provide you with a tool set for creating grammars
  - Grammars that work well (for a given application)
  - Grammars that work poorly (for a given application)
- There is nothing about the theory of CFGs that tells you, a priori, what a "correct" grammar for a given application looks like
- A good grammar is generally one that:
  - Doesn't over-generate very much (high precision)
  - Doesn't under-generate very much (high recall)
- What these look like in practice is going to vary with your application space

MOTIVATION

Why Are We Building Grammars?

- Consider:
  - Oswald shot Kennedy
  - Kennedy was shot by Oswald
  - Oswald was shot by Ruby
- Who shot Kennedy
- Who shot Oswald?

Why Are We Building Grammars?

- Active/Passive
  - Oswald shot Kennedy
  - Kennedy was shot by Oswald
- Relative clauses
  - Oswald who shot Kennedy was shot by Ruby
  - Kennedy who Oswald shot didn't shoot anybody

Knowing Who Did What to Whom

- There are multiple reasons to build grammars but one important reason is knowing who did what to whom
- A parse tree does not tell us this directly, but it is one step in the process of discovering grammatical relations (subject, object, etc.) which can help us discover semantic roles (agent, patient, etc.)

Language Myths: Subject

- **Myth I**: the subject is the first noun phrase in a sentence
- **Myth II**: the subject is the actor in a sentence
- **Myth III**: the subject is what the sentence is about

All of these are **often** true, but none of them is **always** true, or tells you what a subject really is (or how to use it in NLP).

SUBJECT, OBJECT, AND DEPENDENCIES

Subject and Object

- Syntactic (not semantic)
  - The batter hit the ball. [subject is semantic agent]
  - The ball was hit by the batter. [subject is semantic patient]
  - The ball was given a whack by the batter. [subject is semantic recipient]
  - {George, the key, the wind} opened the door.
- Subject ≠ topic
  - I just married the most beautiful woman in the world.
  - Now beans, I like.
  - As for democracy, I think it's the best form of government.

Subject and Object

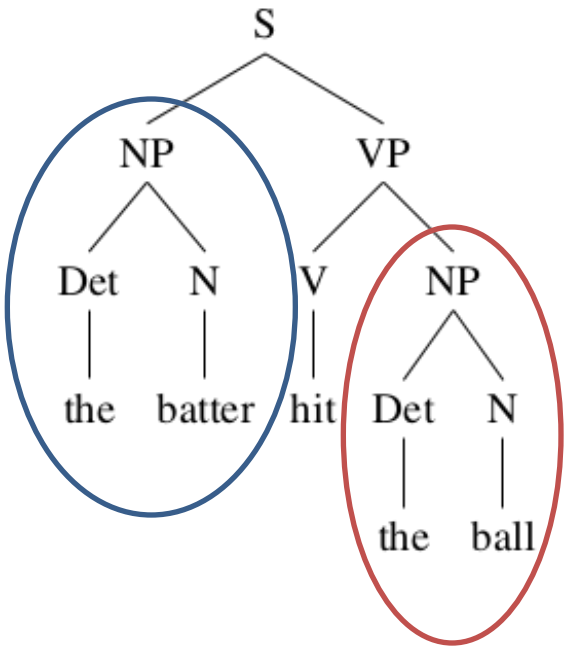
- English subjects
  - agree with the verb
  - when pronouns, in nominative case (I/she/he vs. me/her/him)
  - omitted from infinitive clauses (I tried __ to read the book, I hoped __ to be chosen)
- English objects
  - when pronouns, in accusative case
  - become subjects in passive sentences

Dependency Grammar

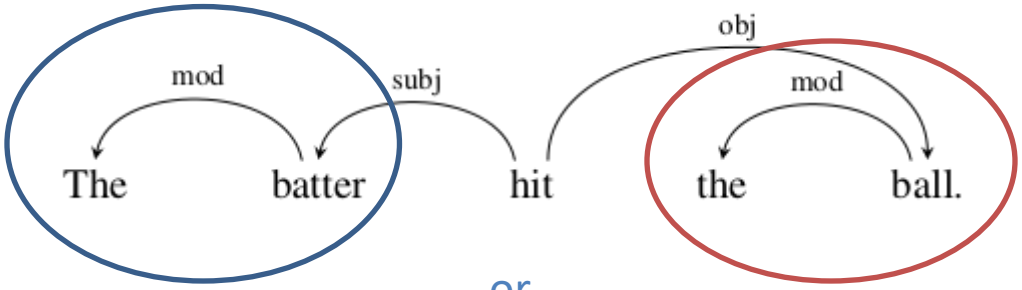
- There is another way of looking at syntax that highlights relations like subject and object
- Dependency grammar
  - Bilexical dependencies
    - Relationships between two words
    - One is “head” and one is “dependent”
    - Labels like “subj” and “obj” on arcs
- Example: verbs are **heads** relative to their subject and objects, which are **dependents**

Dependencies

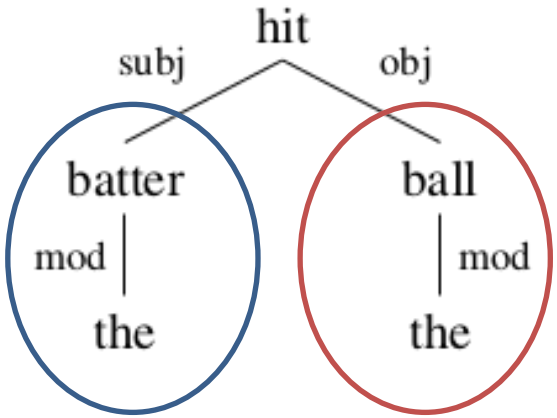
Constituency tree



Dependency tree



or



Advantages and Disadvantages

Advantages of Constituency/Phrase Structure Grammar

- There are widely agreed-upon tests for constituency; there is little agreement above what constitutes a dependency relation
- Constituency maps more cleanly on to formal semantic representations than dependency
- This makes constituency useful in natural language understanding

Advantages of Dependency Grammar

- It is easier to identify grammatical relations (like subject and object) in a dependency parse
- Dependency parses of sentences having the same meaning are more similar across languages that constituency parses
- Dependency parses are also useful for NLU (ask Google)
- Dependency trees are typically simpler

Additional Notes

- Some approaches to syntax, including Lexical Functional Grammar or LFG, use dependency and constituency as parallel representations
- Stanford parser does both constituency and dependency parsing (Neural Network Dependency Parser)
- Many other parsers for both constituency and dependency exist (e.g. Berkeley Parser, MaltParser, SyntaxNet & Parsey McParseface, TurboParser, MSTParser)

Looking Forward

- We will talk a lot about constituency parsing and CFGs (more than about dependency grammar, though you may want to use a dependency parser in your project)
- CFGs may not be entirely adequate for capturing the syntax of natural languages
  - They are almost adequate
  - They are computationally well-behaved (in that you can build relatively efficient parsers for them, etc.)
  - But they are not very convenient as a means for hand-crafting a grammar
  - Also, they are not probabalistic
- In future lectures, we will revisit these properties of CFGs