Applied Human Language Technology

Lecture 5

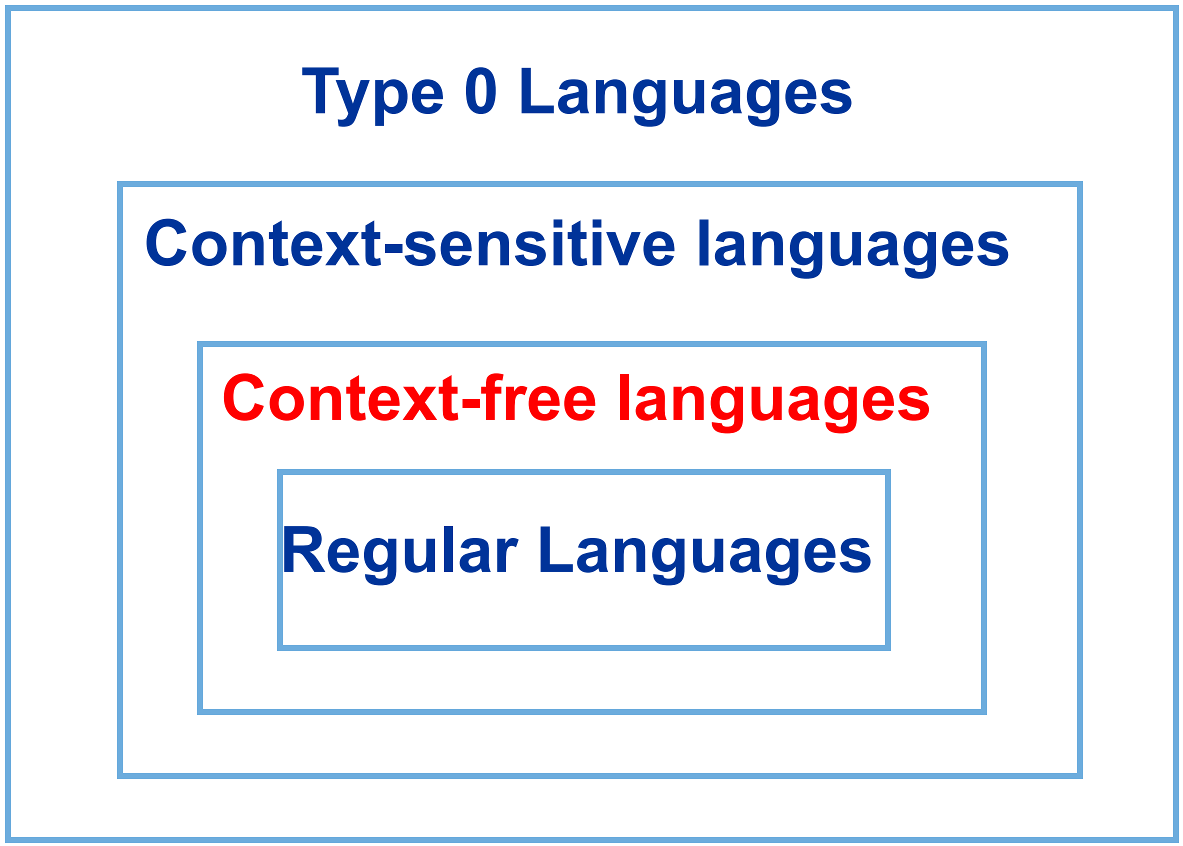
Context-Free Languages

Lecturer: Irene Murtagh

This week:

- Parts of Speech
- Tagging
  - Tagsets
  - Tagging with Rules
  - Transformation-Based Tagging
- Parsing Strategies
  - Top-down vs bottom-up parsing
  - Shallow parsing
- Some things to do

Back to Noam Chomsky...



The Chomsky Hierarchy of Languages

A formal definition of a grammar

Grammar and Description

A grammar of a language is a 4 tuple.

$G=(V_n, V_t, P, S)$

where

V_n is the nonterminal vocabulary finite set

V_t is the terminal vocabulary finite set

P is a finite set of production rules

S is the starting symbol for all productions

Parts of Speech

- What are they?
- Distribution of tags
- Tagsets

Parts of Speech

- Tags code part of speech distinctions
- Most tagsets implicitly encode fine-grained specialisations of eight basic parts of speech.

noun,	verb,
pronoun,	preposition,
adjective,	conjunction,
determiner,	adverb

= Lexical categories

- The categories are based on morphological and distributional similarities
- In some cases, this is straightforward (at least in a given language), in other cases it is not so clear and obvious.

Part of Speech Tagging

Part of speech tagging is simply assigning the correct part of speech for each word in an input sentence.

You will need the following:

- A **set of tags** (a tagset)
- A **lexicon** that tells you the possible POS tags for each word (including all morphological variants).
- A **text** to be tagged.
- **References**:
  https://www.ldc.upenn.edu/collaborations/current-projects/bolt/annotation/treebank
  http://nlp.stanford.edu/software/tagger.shtml

Treebank POS Categories

Treebank POS categories – an expanded inventory.

String	Description	Example
CC	Coordinating conjunction	and
CD	Cardinal number	two
DT	Determiner	the
EX	Existential *there*	there (*There was an old lady*)
FW	Foreign word	omerta
IN	Preposition, subord. conjunction	over, but
JJ	Adjective	yellow
JJR	Adjective, comparative	better
JJS	Adjective, superlative	best
LS	List item marker	
MD	Modal	might
NN	Noun, singular or mass	rock, water
NNS	Noun, plural	rocks
NNP	Proper noun, singular	Joe
NNPS	Proper noun, plural	Red Guards
PDT	Predeterminer	all (*all the girls*)
POS	Possessive ending	's
PRP	Personal pronoun	I
PRP$	Possessive pronoun	mine
RB	Adverb	quickly
RBR	Adverb, comparative	higher (*shares closed higher.*)
RBS	Adverb, superlative	highest (*he jumped highest of all.*)
RP	Particle	up (*take up the cause*)
TO	*to*	to
UH	Interjection	hey!
VB	Verb, base form	choose
VBD	Verb, past tense	chose
VBG	Verb, gerund, or present participle	choosing
VBN	Verb, past participle	chosen
VBP	Verb, non-third person sing. present	jump
VBZ	Verb, third person singular present	jumps
WDT	Wh-determiner	which
WP	Wh-pronoun	who
WP$	Possessive wh-pronoun	whose
WRB	Wh-adverb	when (*When he came, it was late.*)

Tagsets

There are various tagsets to choose from.

- The choice of tagset depends on the nature of the application.
- Since accurate tagging can be performed with relatively large tagsets it makes sense to use one of the larger standard sets (See references on previous slide !!)
- If it makes distinctions you don't need, you can merge the finer grained tags.

The Distribution of Tags

Tags follow a frequency-based distributional behavior.

Most word types have only one part of speech.

Of the rest, most have two.

- Of course, as usual, the most frequently occurring word types tend to have multiple tags.
- .... they also tend to have more meanings.
- Therefore while its easy to determine the correct tag for most word types, it is not necessarily easy to correctly tag most texts.

Tagging with Rules

The rule-based approach uses handcrafted sets of rules to tag input sentences

Adverbial-that rule

Given input : "that"
If (**+1** ADJ/ADV/QUANT); /* if **next** word is adj, adverb, or quantifier */
(**+2** SENTence-LIM); /* and **following** which is a sentence boundary, */
(**NOT -1** SVO**C**/**A**); /* and the **previous word is not** a verb like */
/*`**consider'** which allows **adjs** as object complements */
then eliminate non-ADV tags
else eliminate ADV tag

Read this as:
if the **next** word is an adj, adverb, or quantifier and
the **following** which is a sentence boundary, and
the **previous word is not** a verb like `**consider'**
which allows **adjs** as object complements
then eliminate non-ADV tags
else eliminate ADV tag

Tagging with Rules: This approach does work & produces 99% accuracy of results

Transformation-Based Tagging

The **pure rule-based approach** is expensive, slow, tedious, etc.

- However, given hand-tagged data we can **train up a rule-based approach**

- Basic idea is to do a **quick and dirty job first**, and **then use learned rules** to patch things up.

Brill Tagging

Example... RACE$_{N}$ RACE$_{V}$

1. **On first pass, tag** all uses of "**race**" as **nouns**. i.e, RACE$_{N}$

2. **Then** on next pass, **replace** that tag with the one for **verb** for all "**race**" uses that are preceded by the tag **TO**. i.e, RACE$_{V}$

- This works fine for:

  is expected **to race** tomorrow

- Leaves the following alone:

  the **race** for outer space

- And makes a mess of this one:

  drawing the district line according **to race**

Brill Tagging

Assume some tagged training corpus.

1. **Tag the corpus** with the most likely tag for each word (**unigram** model).

2. **Choose a transformation** that **deterministically replaces an existing tag with a new tag** such that the resulting tagged training corpus has the lowest error rate out of all transformations.

3. **Apply that transformation** to the training set.

4. **Iterate**.

5. Return as your tagger one that
   - First tags using **unigrams** and then
   - Applies the learned transformations in order.

Brill Tagging

The transformations we can make....

Change Tag a to Tag b when:

- The proceeding (following) word is tagged ***z***.
- The word two before (after) is tagged ***z***.
- One of the two preceding (following) words is tagged ***z***.
- One of the three preceding (following) words is tagged **z**.
- The preceding word is tagged z and the following word is tagged **w**.
- The proceeding (following) word is tagged **z** and the word two before (after) is tagged **w**.

Where a, b, w, and z range over all the tags.

Brill Tagging

As with all **iterative improvement algorithms**,

... it is limited by the shape of the space it is searching.

For example:

- The templates we just looked at are based on **tags**.
- They don't look at individual **words**.

……From POS tagging to processing of syntax

Syntax

By syntax, we mean various aspects of :

1. how words are strung together to form components of sentences <u>and</u>

2. how those components are strung together to form sentences.

- We are talking about the syntax you learned fully by the time you were 4 or 5 years old.

- We are <u>not</u> talking about the kinds of grammar you may or may not have been taught in school.

- Notions of **meaning** play no role in what we're talking about (…. at least not yet).

You need **knowledge of syntax** in the following applications:

- Grammar checkers
- Question answering intelligent agents
- Information extraction
- Information retrieval
- Headline or text generation
- Machine Translation
- Internet search engines

Context Free Grammars

Captures constituency and ordering

We'll need something else for

- grammatical relations and
- dependency relations.

Modern linguistic theories of grammar are only vaguely based on context-free grammars.

Context Free Grammars

Consist of

- Sets of **terminals** (either lexical items or parts of speech).
- Sets of **non-terminals** (the constituents of the language).
- Sets of **rules** of the form $\mathbf{A} \rightarrow \alpha$,

  where α is a string of zero or more terminals and non-terminals.

They are exactly equivalent to **Backus-Naur form grammars**.

Examples

s - -> np vp

np - -> det nominal

nominal - -> n

nominal - -> n conj n

vp - -> v

det - -> [a]

noun - -> [flight]

v - -> [left]

Generativity

Just as we did with FSAs, you can view these rules as either

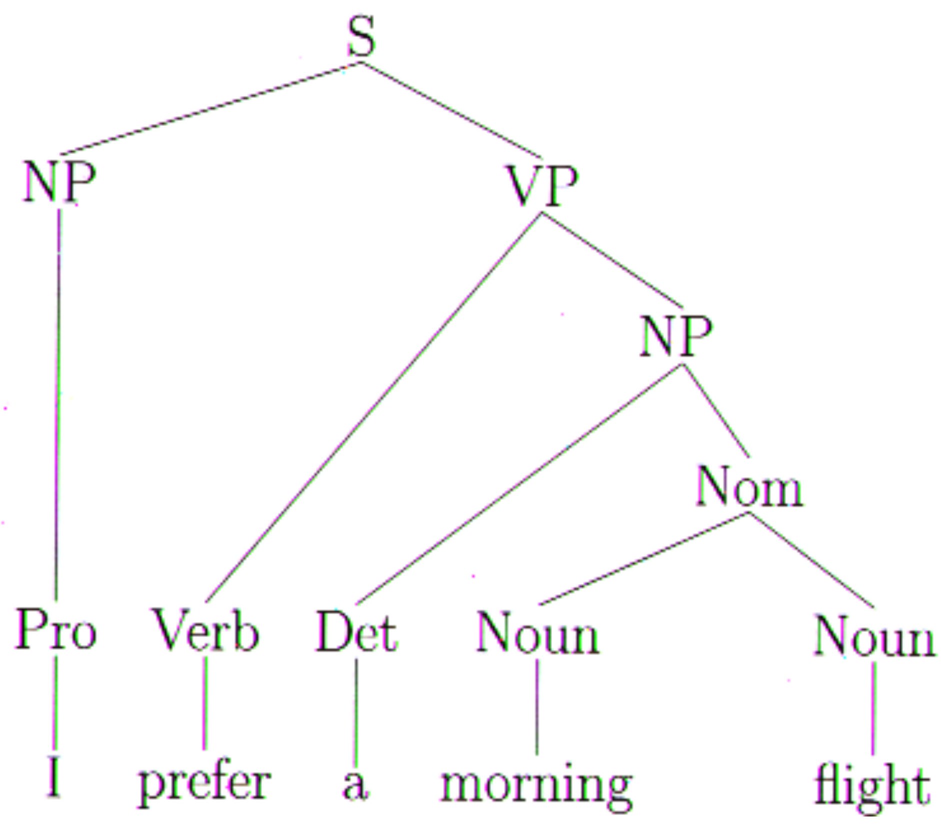
- structure imposing devices or
- generative devices.

The grammar can be viewed as a formal device that specifies the strings in, and not in, a language.

Derivations and Trees

A **derivation** is the execution of a sequence of rule

Derivations can be visualized as parse trees...



The use of the term context-free in the description of this formalism has nothing to do with the ordinary use of the word **context**.

All it really means is that the **non-terminal** on the left-hand side of the rule is '*sitting over there all by itself*'.

$$\boxed{\mathbf{A}} \rightarrow \mathbf{B}\ \mathbf{C}$$

In other words, ...

we can rewrite an **A** as **B followed by C**, regardless of the <u>context</u> in which we find the **A** .

A Small Grammar

S	$\rightarrow$	$NP\ VP$	I + want a $\boxed{\text{morning flight}}$ n n
NP	$\rightarrow$	$Pronoun$	I
	$\mid$	$Proper-Noun$	Los Angeles
	$\mid$	$Det\ Nominal$	a + flight
Nominal	$\rightarrow$	$Noun\ Nominal$	morning + flight
	$\mid$	$Noun$	flights
VP	$\rightarrow$	$Verb$	do
	$\mid$	$Verb\ NP$	want + a flight
	$\mid$	$Verb\ NP\ PP$	leave + Boston + in the morning
	$\mid$	$Verb\ PP$	leaving + on Thursday
PP	$\rightarrow$	$Preposition\ NP$	from + Los Angeles

Key Constituents

The key **constituents** that we will care about are the following:

- sentences
- noun phrases
- verb phrases
- prepositional phrases

Sentence Types

Declaratives

John left.

s → np vp

Imperatives

Leave!

s → vp

Yes-No Questions

Did John leave?

s → aux np vp

WH-Questions

When did John leave?

s → whword aux np vp

Recursive Structures

One of the more interesting patterns we need to deal with involves recursive rules.

- These are rules where the non-terminal on the left-hand side also appears on the right-hand side.

Direct recursion:

The flight to Paris departed Dublin at noon

np → **np** pp   The flight to Paris

vp → **vp** np pp   departed Dublin at noon

Recursive Structures

This is what allows us to do the following:

Flights to Paris

Flights to Paris from Dublin

Flights to Paris from Dublin in April

Flights to Paris from Dublin in April on Friday

Flights to Paris from Dublin in April on Friday under €100.

Flights to Paris from Dublin in April on Friday under €100 with lunch.

Conjunctions

Any **phrasal constituent** can be conjoined with a constituent of the same type to form a new constituent of that type.

s → s and s

np → np and np

vp → vp and vp

In other words, English really has the following rule:

x → x and x

Some Difficulties

Agreement

Subcategorisation

Movement

Subcategorisation

Verbs have preferences for the kinds of constituents they co-occur with:

*I disappeared the cat.

The cat disappeared.

Agreement

This dog

Those dogs

*Those dog

*This dogs

Movement

I looked up his grade.

I looked his grade up.

Parsing

Parsing with a CFG is the task of assigning a correct tree / derivation to a string, given some grammar.

- Note that **correct** here means that it is **consistent with the input and the grammar**.
- It doesn't mean that it is the "right" tree in any more global sense of correct.

More specifically, ...

- the leaves of the tree cover all and only the input and
- the tree corresponds to a valid derivation according to the grammar.

Parsing as Search

- As with finite-state recognition (FSAs, FSTs), parsing can be viewed as a search.
- The search space corresponds to the space of trees generated by the grammar.
- The search is guided by the structure of the space and by the input.

We'll start with the basic (bad) methods of parsing, see what's wrong with them, and then move on to a better method.

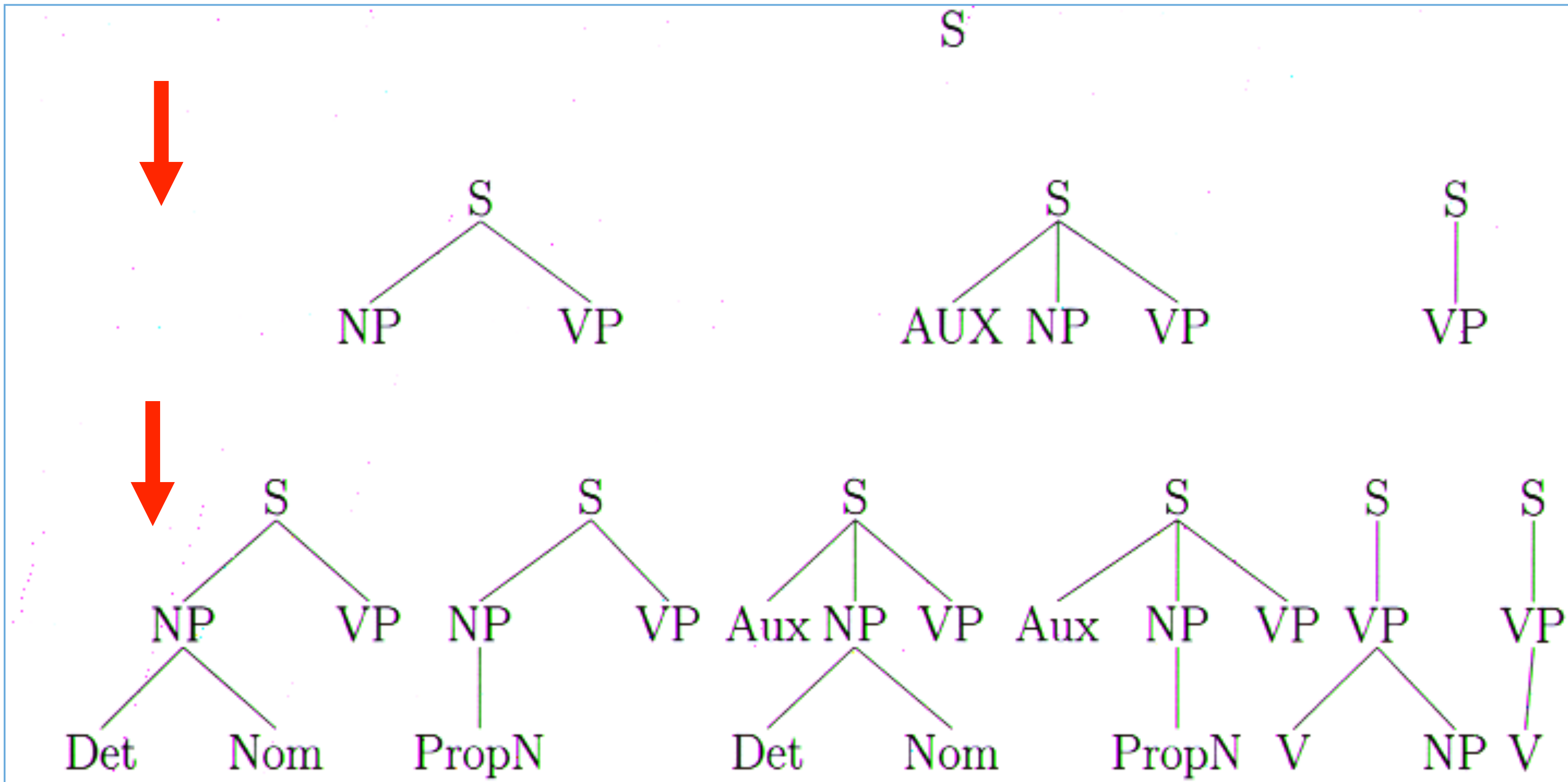
For now, we'll assume the following:

- The input is not tagged
- The input consists of unanalysed word tokens
- All the words in the input are known
- All the words in the input are available simultaneously (i.e. they are buffered into some variable in memory)

Top-Down Parsing

If the search is primarily **goal** or **expectation-driven** (by the structure of the grammar),
then we're doing some kind of top-down search.

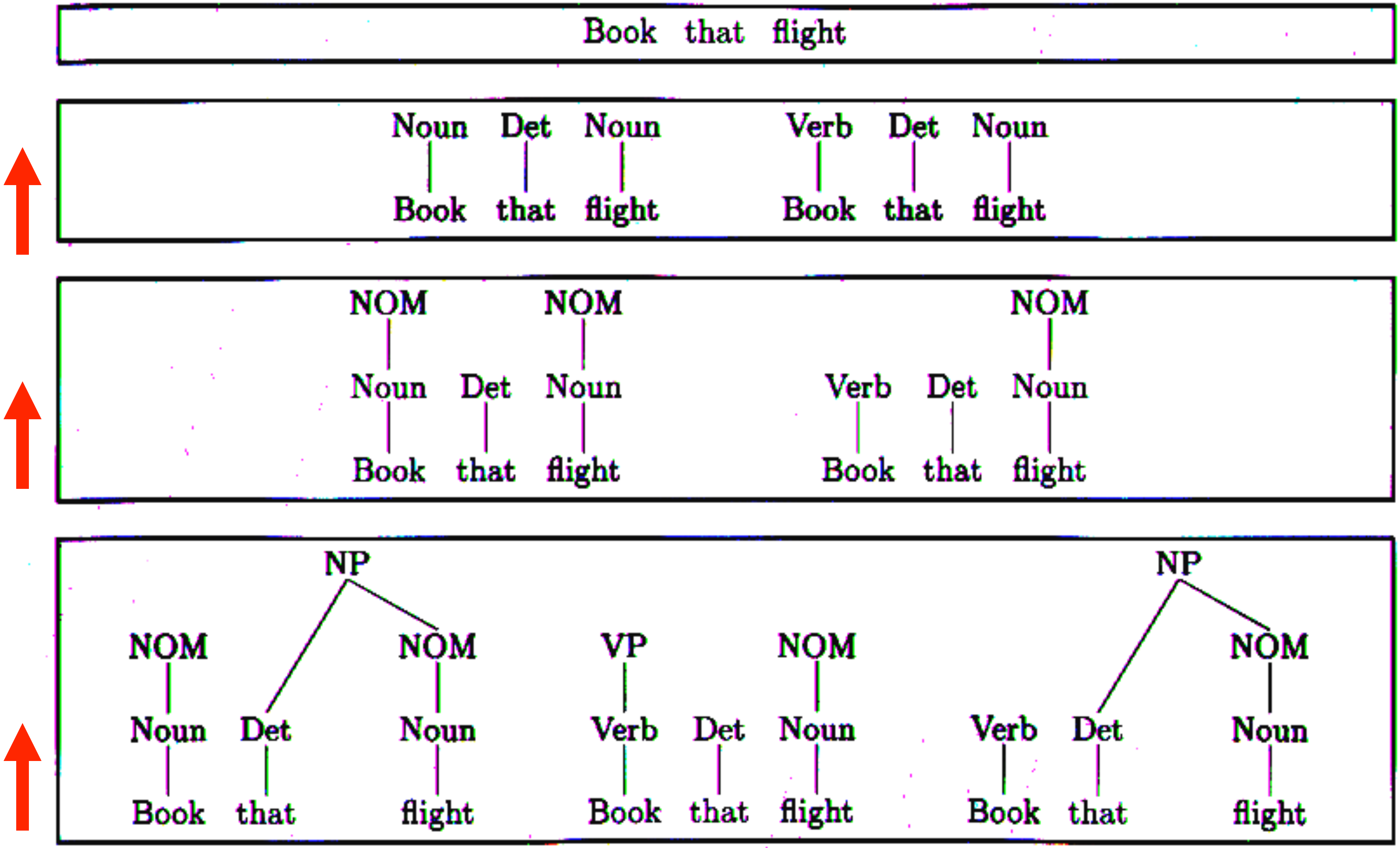
- The primary goal is that we can start out by trying to **find a tree** rooted as **S**, since we're trying to parse sentences.



Bottom-Up Parsing

When the search is primarily **data-driven** (by the input words), then we're doing some kind of bottom-up search.

The primary consideration here is that the lowest sub-trees of the final tree must hook up with the top (= S)



Top-Down and Bottom-Up Considered

There are advantages and disadvantages to both.

Top-Down

- Only searches in the **space of reasonable answers**
- Suggests hypotheses that are not consistent with the data.
- Note: we saw examples of this using the query: -s([L, [ ]).

Bottom-Up

- Only forms **hypotheses consistent with the data**.
- Suggests hypotheses that make no sense globally.

SOME TASKS TO DO

1. **Read** Ch.10 & 11: *Speech and Language Processing* by Jurafsky and Martin

2. Using the link below **read** the chapter on the NLTK POS tagger and how you can manipulate this using Python.

http://www.nltk.org/book/ch05.html