Syntax

Dr. G. Bharadwaja Kumar

Language from Formal Language Theory

A language

- A language L is a possibly infinite set of strings.
- The strings are made from a finite alphabet.
  - The "alphabet" might be the words of English
  - Henceforth, we will call it the "vocabulary"
- Some strings of language L:
  - Bears live in the forest.
  - Revolutionary new ideas appear frequently.
- Some strings are not in L:
  - *Infrequently appear ideas new revolutionary.
  - *Live bears the in forest.
  - (* means that the string is not a member of the set of strings that comprise the language L.)

What is a Grammar Formalism?

A *formal grammar* consists of a collection of rules that specify how elements of the language, e.g., words, may be combined to form sentences, and how sentences are structured. Rules may be concerned with purely syntactic information, such as grammatical functions, subject–verb agreement, word ordering, etc., but some models may also incorporate issues such as lexical semantics.

Rules in Grammar Formalisms

I will go.
*I will gone/going/went.
I have gone.
*I have go/gone/going/went.
I am going.
*I am go/went.
    I am gone (adjective).
I will have been being arrested.
I will have been singing.
*I will be having sung. (wrong order)
*I will be sung. (wrong verb form.)
Etc.

Verb -Subcategorization

Subcategorization lists

(Representing the valency pattern by a list of the complements).

- walk: V, [ ] (*She walks*)
- like: V, [NP] (*She likes the dog*)
- give: V, [NP, NP] (*She gives the dog a bone*)
- give: V, [NP, PP(to)] (*She gives a bone to the dog*)
- pretend: V, [S(fin)] (*He pretended he had gone home*)
- suggest: V, [S(base)] (*He suggested we go home*)
- intend: V, [VP(to)] (*He intended to go home*)
- help: V, [VP(base)] (*He helped clean up*)
- tell: V, [NP, VP(to)] (*He told them to clean up*)
- make: V, [NP, VP(base)] (*He made them clean up*)
- say: V, [PP, S(fin)] (*He said to me he would clean up*)
- bet: V, [NP, NP, S(fin)] (*He bet me ten pounds he would clean up*)
- become: V, [AP] (*He became unhappy*)
- word: V, [NP, ADVP] (*He worded the reply cleverly*)

What is a Grammar Formalism?

A Grammar:

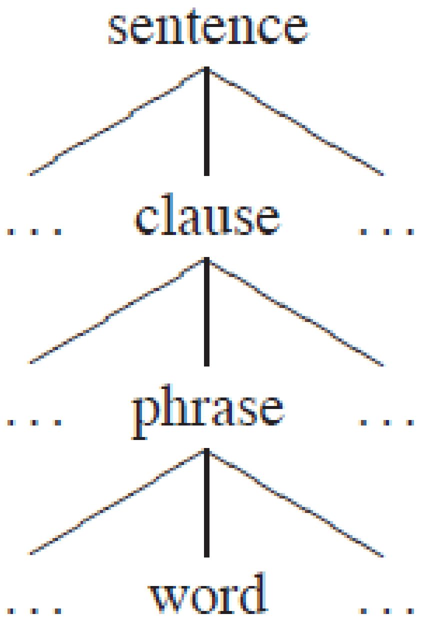
- A set of production rules.
- In addition to the vocabulary, the production rules can use other symbols
  - N (noun)
  - V (verb)
  - NP (noun phrase)
  - VP (verb phrase)
- One symbol is special:
  - S (sentence)

What is Syntax?

Study of structure of language
Refers to the way words are arranged together, and the relationship between them.
Roughly, goal is to relate surface form (what we perceive when someone says something) to semantics (what that utterance means)

Parsing

In the context of natural language processing, the term *parsing* refers to the process of automatically analyzing a given sentence, viewed as a sequence of words, in order to determine its possible underlying syntactic structures.



Why NLP needs grammars: Question Answering

This requires grammatical knowledge...:

John persuaded/promised Mary to leave.

- Who left?

There is a wide range of grammatical formalisms, which depend on various syntactic theories, and the structures that result from parsing, or *parses*, may differ substantially between one such formalism and another.

Many formalisms specify the syntactic analysis of a sentence in terms of a *phrase structure*, which is an ordered, labeled tree that expresses hierarchical relations among certain groupings of words called *phrases*.

An alternative representation is *dependency structure*, which indicates binary grammatical relations between words in a sentence.

Introduction

The syntactic parsing of a sentence consists of finding the correct syntactic structure of that sentence in a given formalism/grammar.

Dependency Grammar (DG) and Phrase Structure Grammar (PSG) are two such formalisms.

Phrase Structure Grammar (PSG)

Breaks sentence into constituents (phrases)

Which are then broken into smaller constituents

Describes phrase structure, clause structure

E.g.. NP, PP, VP etc..

Structures often recursive

The clever tall blue-eyed old … man

Example constituents

Subj. verb. Obj.

John met Mary.

The tall boy met the tall girl.

A boy from Seattle met a girl from Chicago.

A boy from Seattle met the tall girl.

John met a student who majors in mathematics.

What is subject? Something that the main verb agrees with.

How to find - Constituent tests

answers to questions
substitution (pronouns, names, do, etc.)
clefting (It was … that …)
Coordination
Deletion
Movement
Topicalization

Phrase structure rules specify the well-formed structures of a sentence

Categorical Rules

S → NP VP

NP → Det N : a book

NP → (det) (Adj) N : a big book

NP → (det) (Adj) N (PP) : the big book on the chair

VP → V : saw

VP → V NP : saw a buffalo

VP → V NP PP : saw a buffalo in the zoo

VP → V NP PP Adv : saw a buffalo in the zoo silently

PP → P NP : in the zoo

AP → ADVP A : **Very beautiful**

ADVP → ADV : **Cleverly**

Lexical Rules

N → girl

N → boy

Adv → incredibly

A → conceited

V → seem

Modal → must

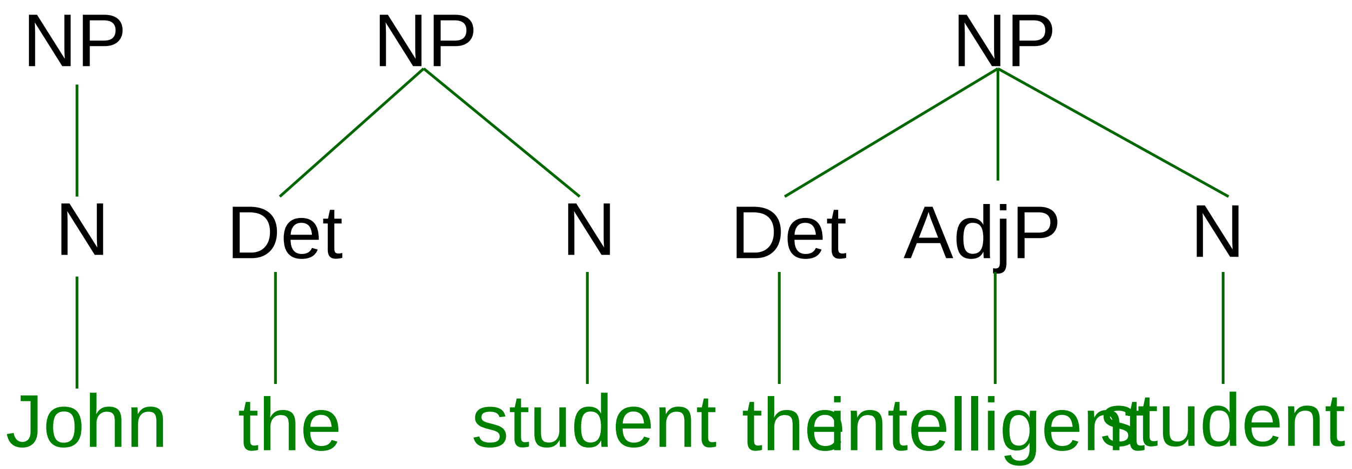
P → to

D → that

D → this

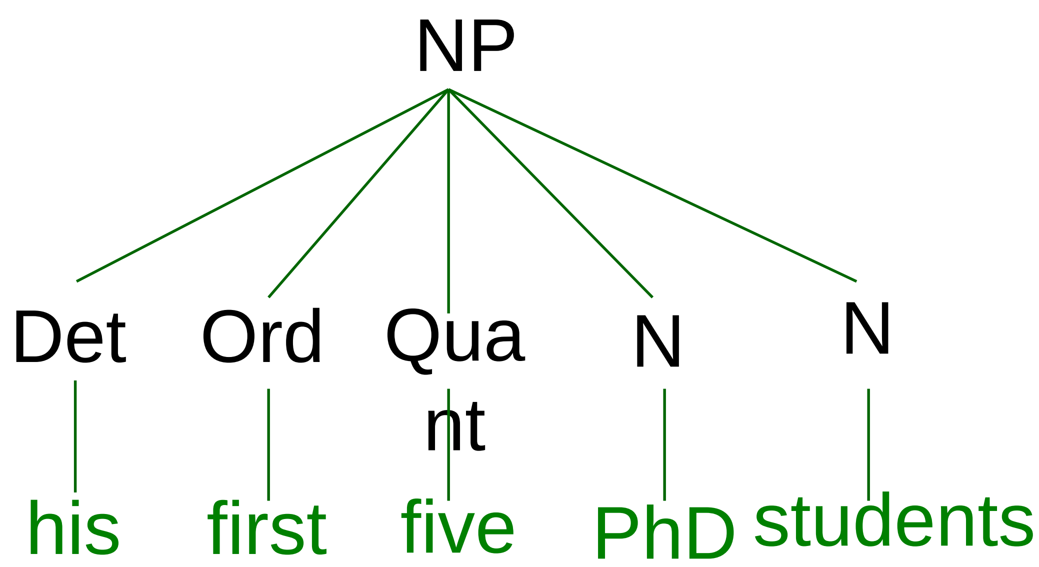
Noun Phrases

John | the student | the intelligent student



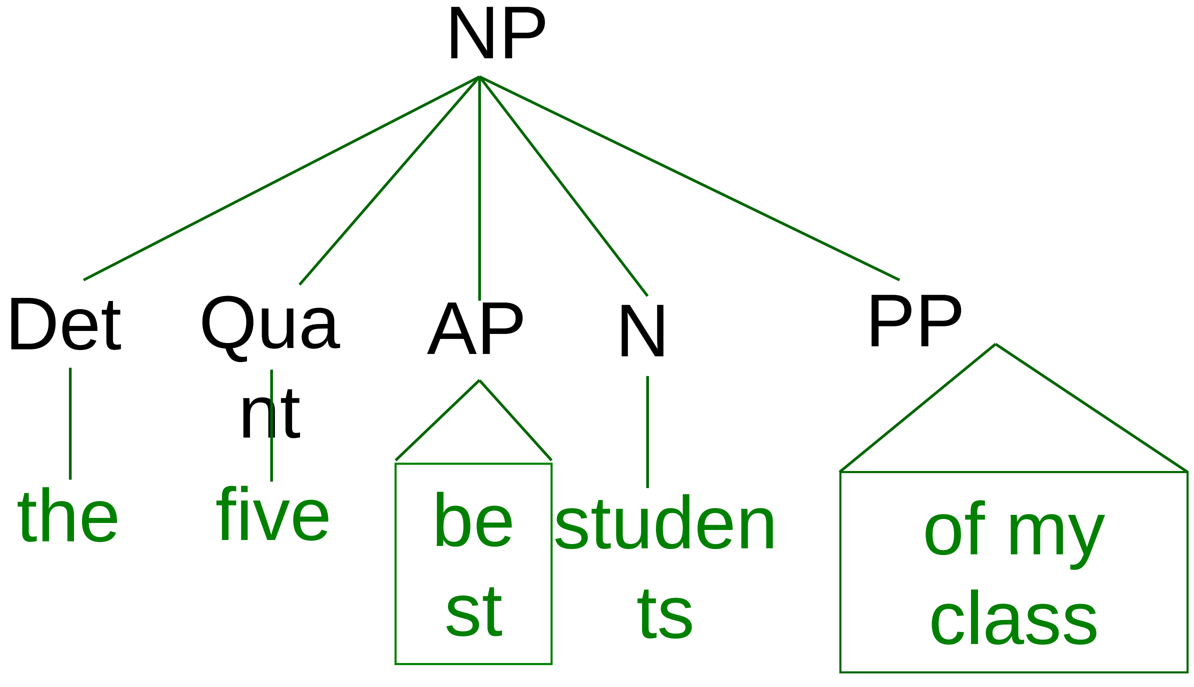
Noun Phrase

his first five PhD students



Noun Phrase

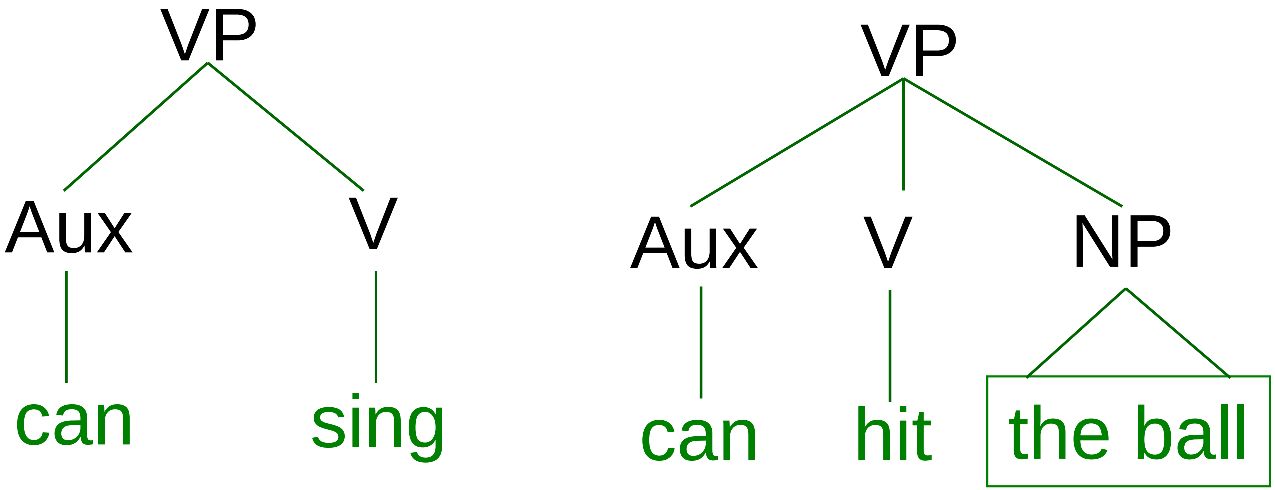
The five best students of my class



Verb Phrases

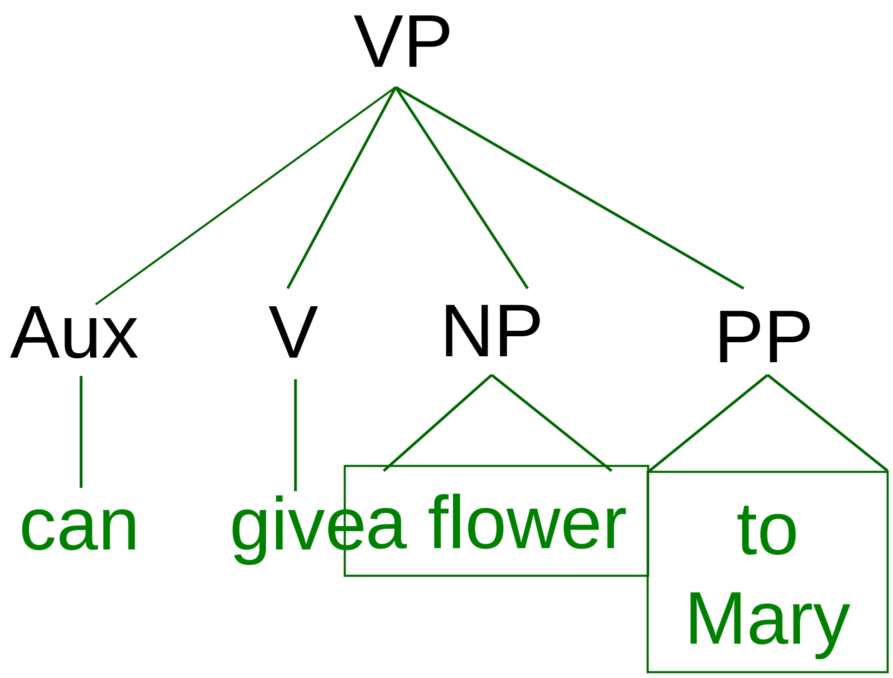
can sing

can hit the ball



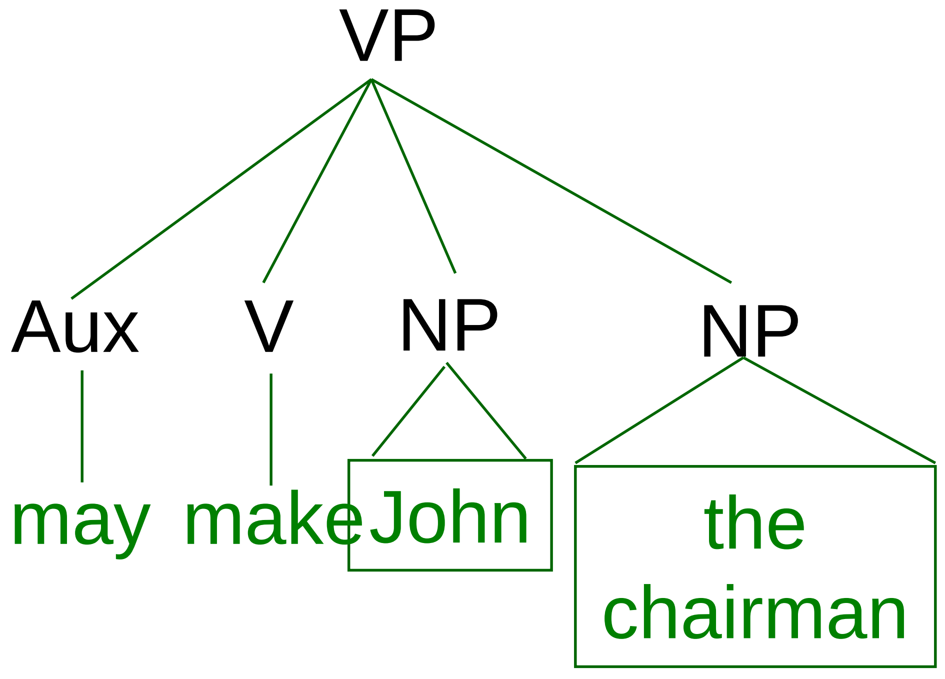
Verb Phrase

Can give a flower to Mary



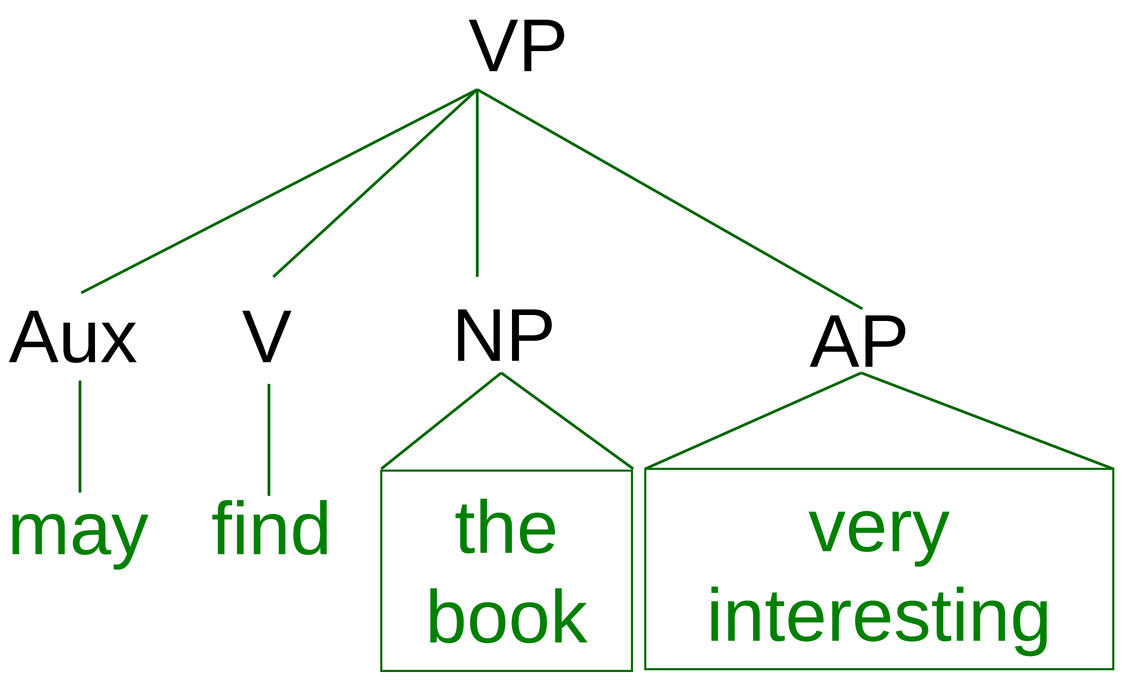
Verb Phrase

may make John the chairman



Verb Phrase

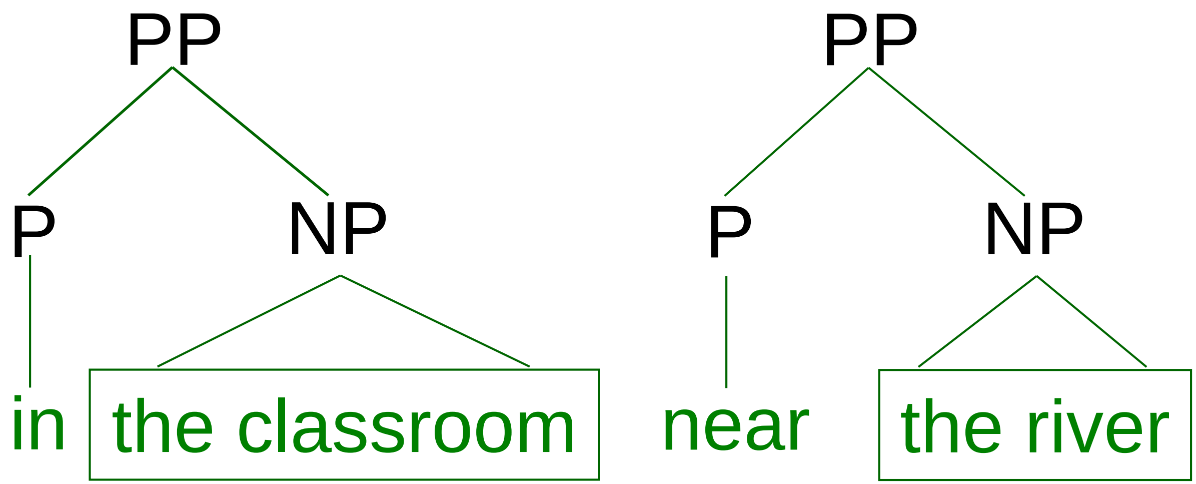
may find the book very interesting



Prepositional Phrases

in the classroom

near the river

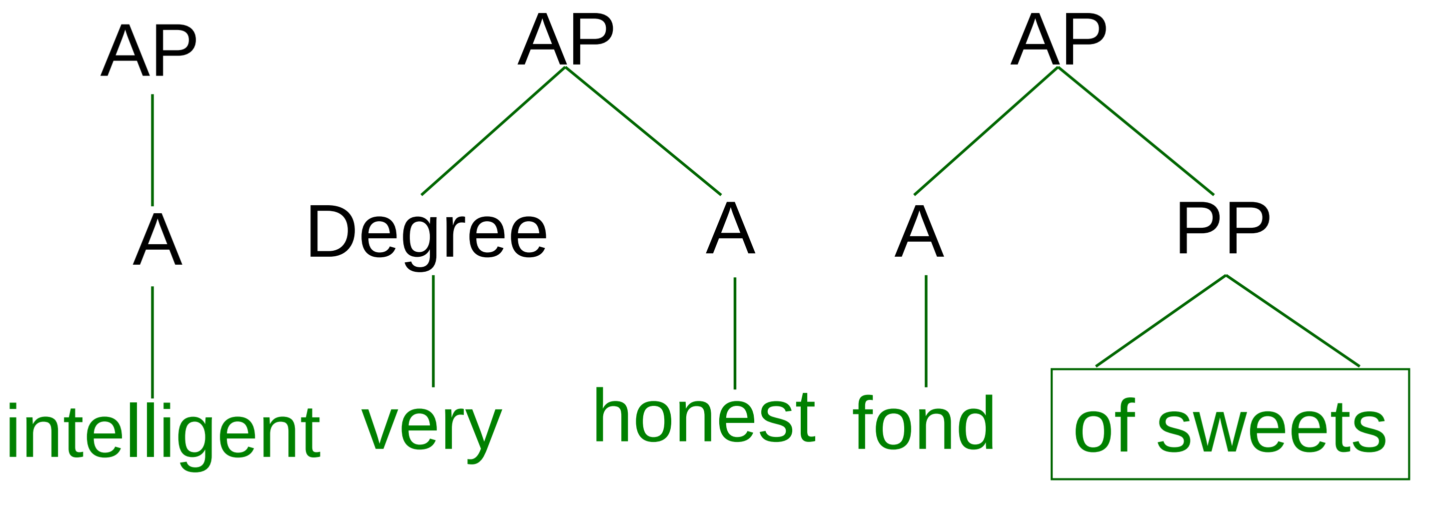


Adjective Phrases

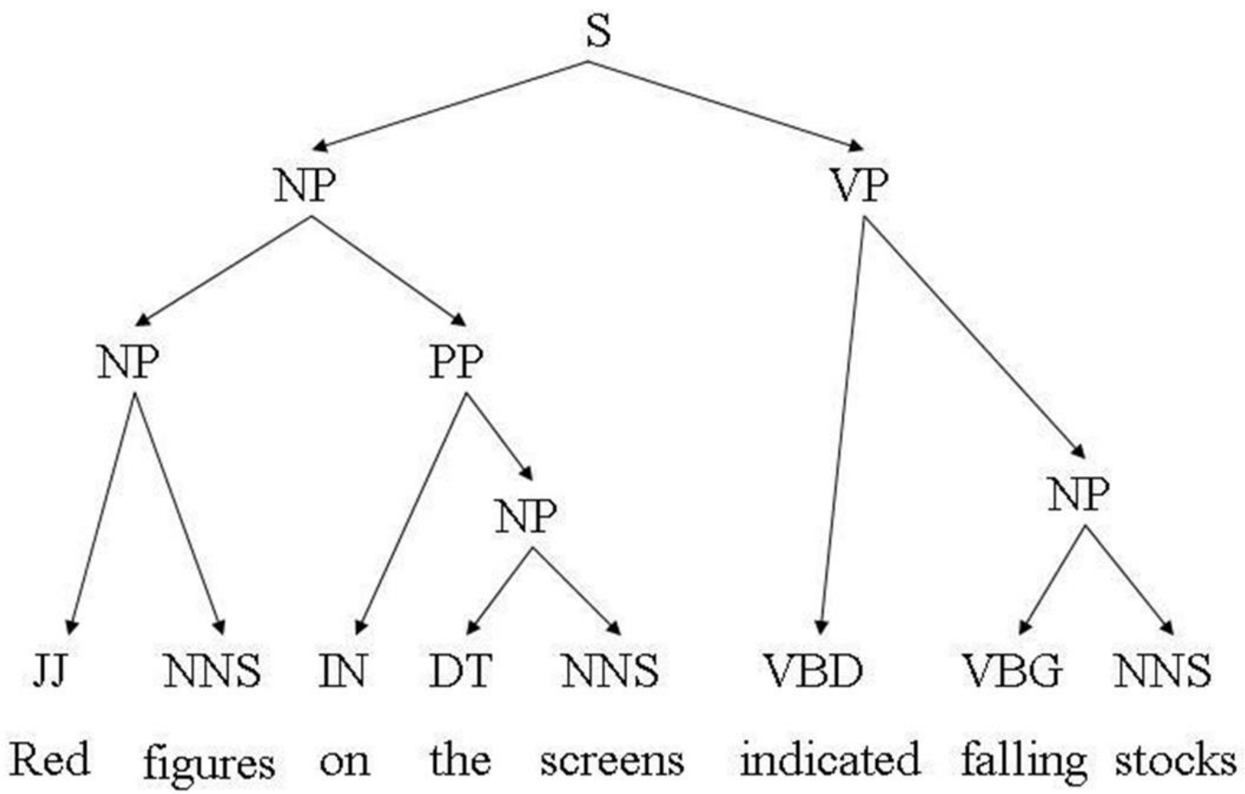
intellig
ent

very honest

fond of sweets

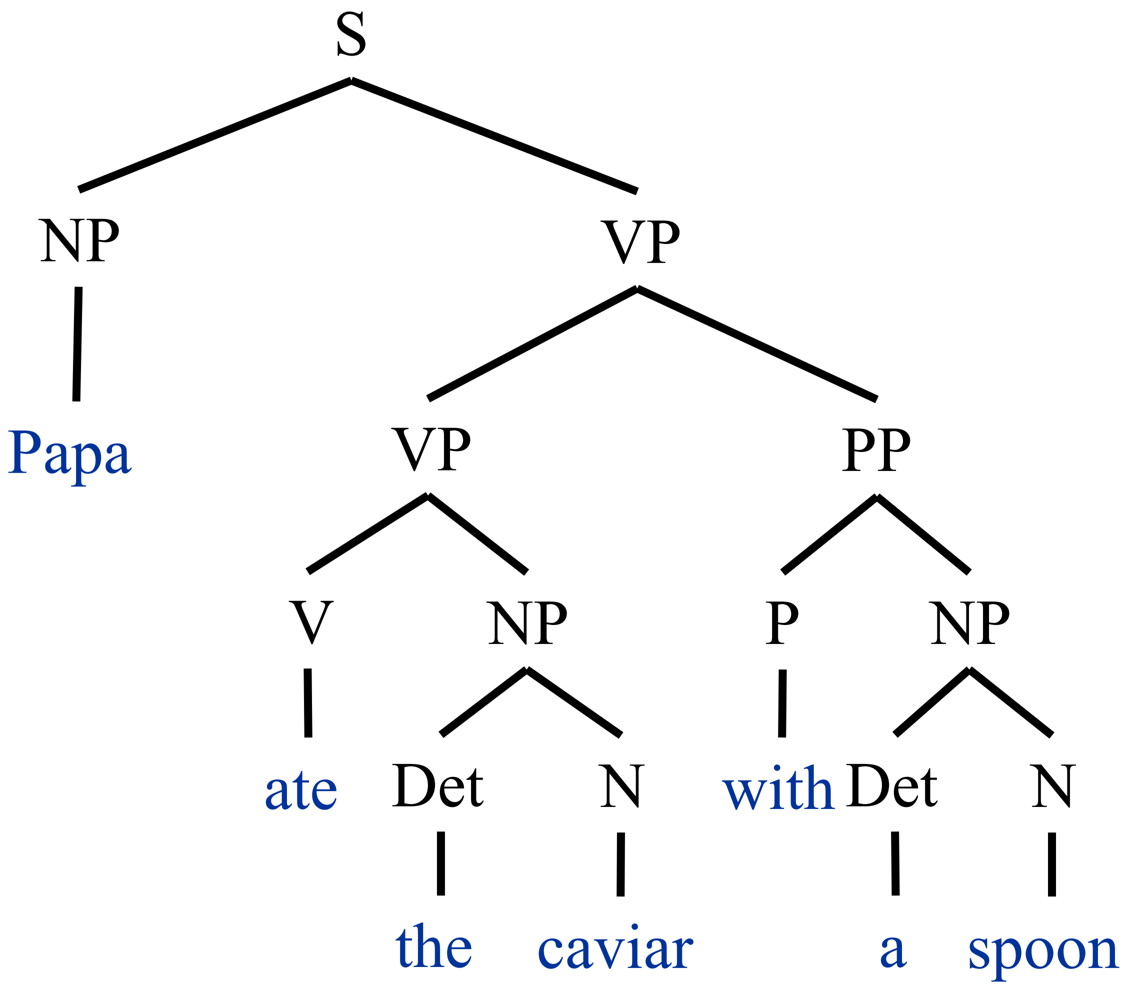


Phrase structure tree



Ambiguity

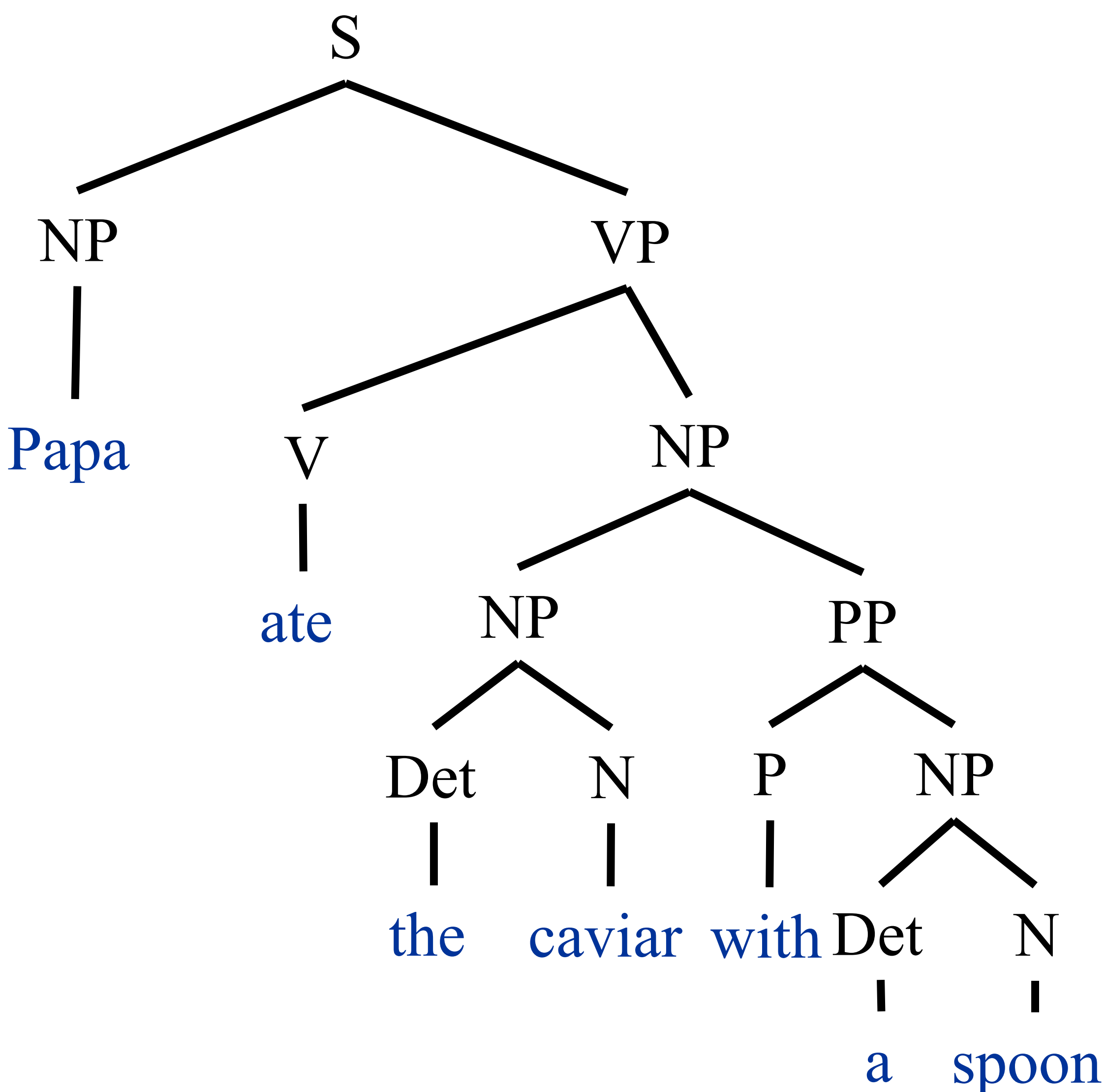
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

Ambiguity

S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

PP - Attachment

- How many distinct parses does the following sentence have due to PP attachment ambiguities?

John wrote the book with a pen in the room.

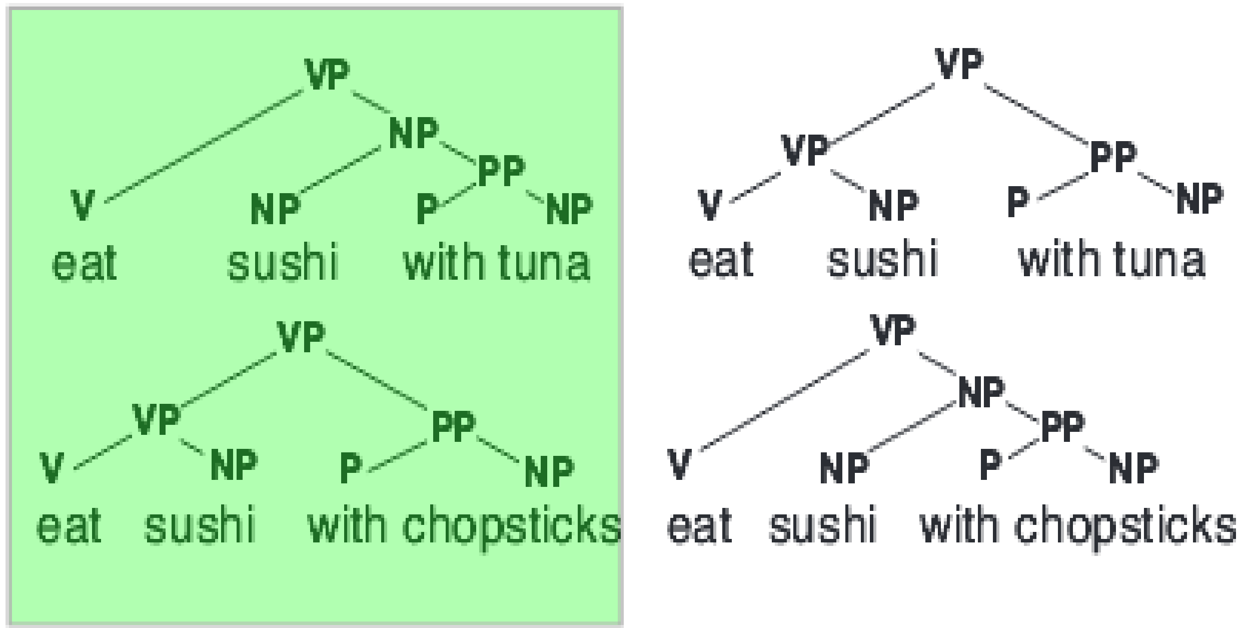
John wrote [the book] [with a pen] [in the room].
John wrote [[the book] [with a pen]] [in the room].
John wrote [the book] [[with a pen] [in the room]].
John wrote [[the book] [[with a pen] [in the room]]].
John wrote [[[the book] [with a pen]] [in the room]].

Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$ - an exponentially growing series

1	1
2	2
3	5
4	14
5	42
6	132
7	429
8	1430

Structural ambiguity results in multiple parse trees

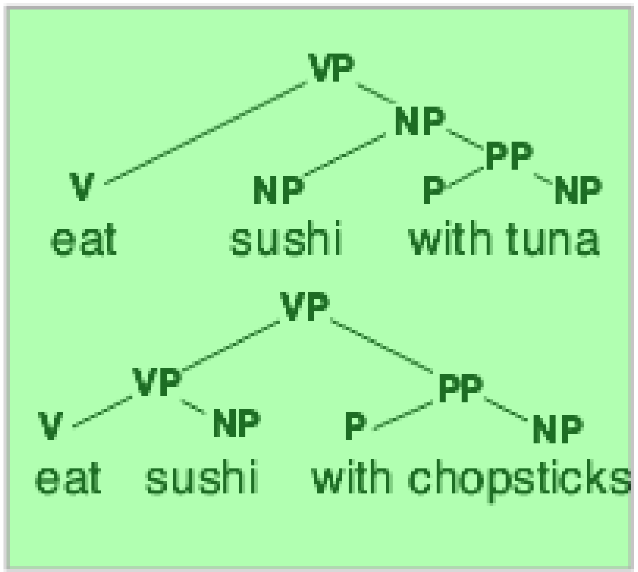
N → *{sushi, tuna}*
P → *{with}*
V → *{eat}*
NP → N
NP → NP PP
PP → P NP
VP → V NP
VP → VP PP



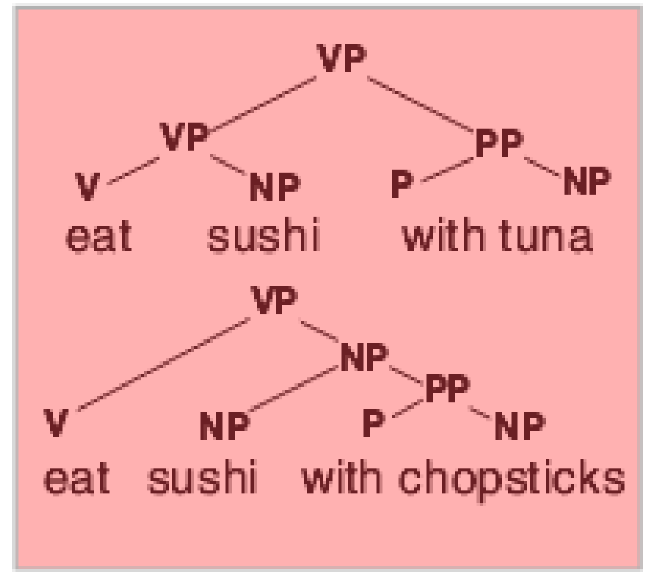
Correct Structures

Structural ambiguity results in multiple parse trees

N → *{sushi, tuna}*
P → *{with}*
V → *{eat}*
NP → N
NP → NP PP
PP → P NP
VP → V NP
VP → VP PP



Correct Structures



Incorrect Structures

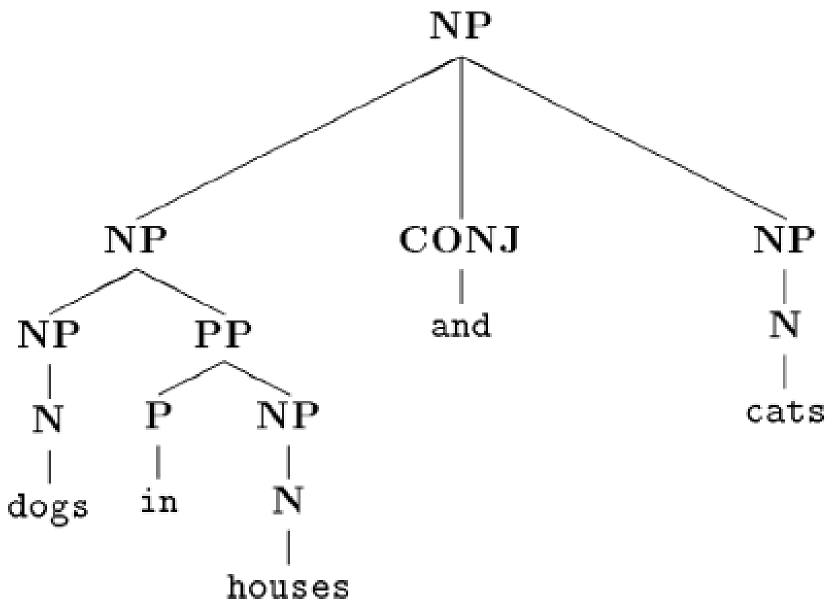
Probabilistic Context Free Grammar (PCFG)

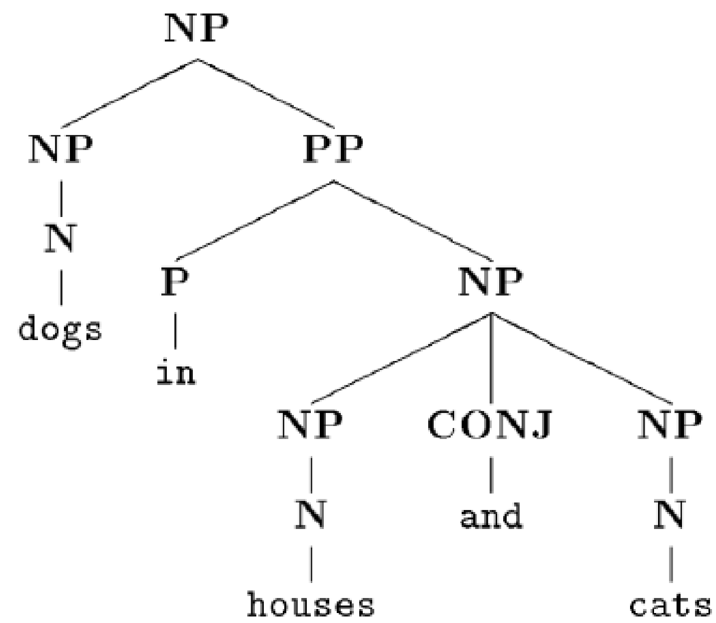
A PCFG is a probabilistic version of a CFG where each production has a probability.

Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal.

String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

Coordination Ambiguity





The two parse trees have identical rules, and therefore have identical probabilities under any assignment of PCFG rule probabilities

Parser for PCFG

The Cocke-Younger-Kasami (CYK) algorithm determines whether or not a string can be generated by a given context-free grammar and, if so, how it can be generated.

The standard version of CYK recognizes languages defined by context-free grammars written in Chomsky normal for m (CNF).

PCFG Parsing

Choose most likely parse tree…

Probabilistic CFG

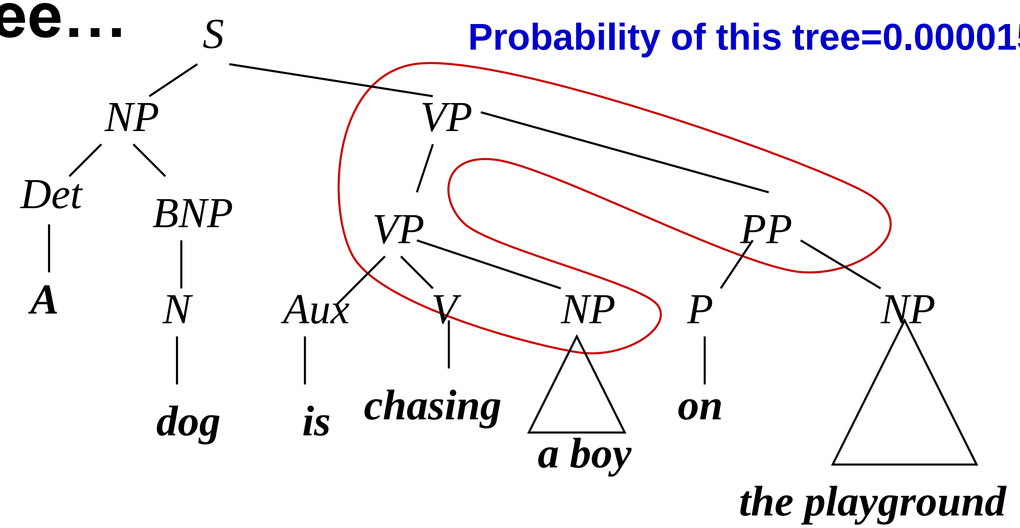
Grammar

$S \rightarrow NP\ \ VP$	1.0
$NP \rightarrow Det\ BNP$	0.3
$NP \rightarrow BNP$	0.4
$NP \rightarrow NP\ \ PP$	0.3
$BNP \rightarrow N$	…
$VP \rightarrow V$	
$VP \rightarrow Aux\ V\ NP$	
$VP \rightarrow VP\ PP$	…
$PP \rightarrow P\ NP$	1.0

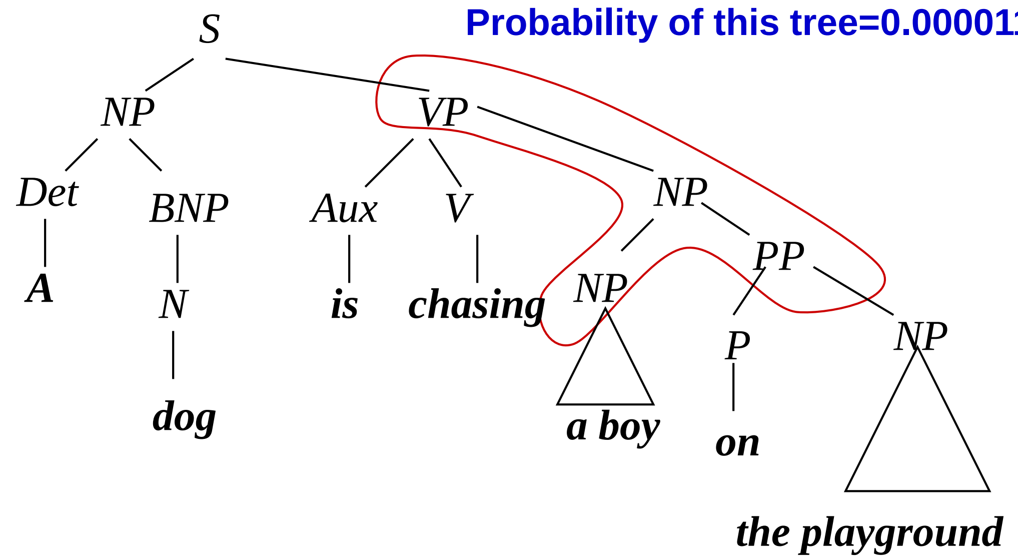
Lexicon

$V \rightarrow chasing$	0.01
$Aux \rightarrow is$	
$N \rightarrow dog$	0.003
$N \rightarrow boy$	
$N \rightarrow playground$	…
$Det \rightarrow the$	
$Det \rightarrow a$	…
$P \rightarrow on$	

Probability of this tree=0.000015



Probability of this tree=0.000011



Lexicalized PCFGs

A lexical *head* is associated to each syntactic constituent.

Each PCFG rule is augmented to identify one right-hand side constituent as its *head* daughter.

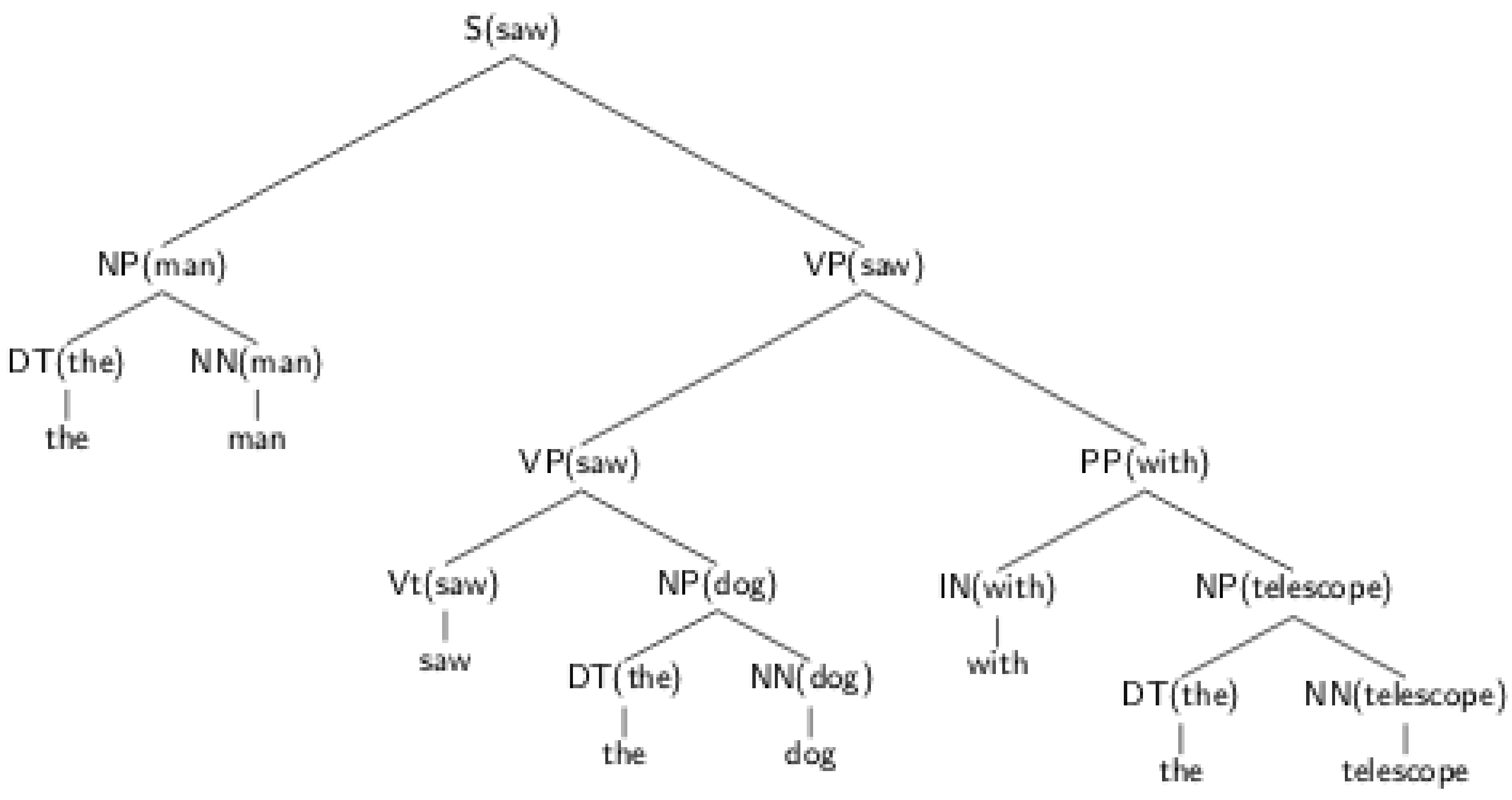
$$p(r(n) \mid n, h(n))$$

Problems with data sparseness: need to smooth to avoid 0 probabilities.

Lexicalized PCFGs

- Each PCFG rule is augmented to identify one right-hand side constituent as its head daughter.
  - $S \rightarrow NP\ VP$ (VP is the head)
  - $VP \rightarrow Vt\ NP$ (Vt is the head)
  - $NP \rightarrow DT\ NN$ (NN is the head)
- A core idea in linguistics (Dependency Grammar, X-bar Theory, Head-Driven Phrase Structure Grammar)

Lexicalized PCFG



Dependency Parsing

- Central Idea
  - verb imposes requirements on its syntactic dependents that reflect its interpretation as a semantic predicate

- The word forms of a sentence can be linked by three types of dependencies:
  - Morphological
  - Syntactic
  - Semantic

- Criteria for establishing dependency relations, and for distinguishing the head and the dependent in such relations, are clearly of central importance for dependency grammar.

Some of the criteria that have been proposed for identifying a syntactic relation between a head H and a dependent D in a construction C

1. H determines the syntactic category of C and can often replace C.
2. H determines the semantic category of C; D gives semantic specification.
3. H is obligatory; D may be optional.
4. H selects D and determines whether D is obligatory or optional.
5. The form of D depends on H (agreement or government).
6. The linear position of D is specified with reference to H.

Dependency Grammars

The way to analyze a sentence is by looking at the relations between words

A verb and its valents/arguments drive an analysis, which is closely related to the semantics of a sentence

No grouping, or constituency, is used

Dependency Grammar

- Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies
- Interested in grammatical relations between individual words (governing & dependent words)
- Does not propose a recursive structure
  - Rather a network of relations
- These relations can also have labels

- A major advantage of dependency grammars is their ability to deal with languages that are morphologically rich and have a relatively free word order

- The head-dependent relations provide an approximation to the semantic relationship between predicates and their arguments that makes them directly useful for many applications such as coreference resolution, question answering and information extraction

- Parsing is much faster than CFG-bases parsers

- Constituent-based approaches to parsing provide similar information, but it often has to be distilled from the trees via techniques such as the head finding rules

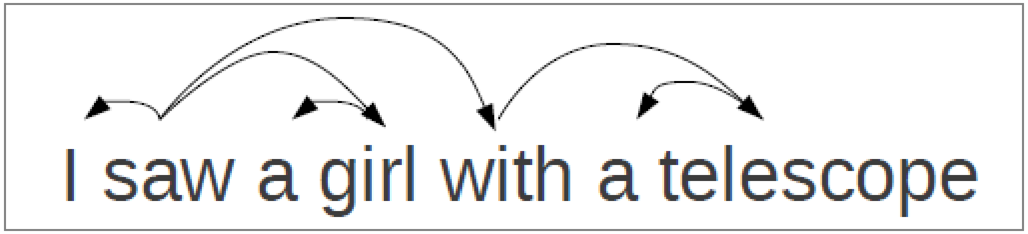
Interpreting Language is Hard!

I saw a girl with a telescope

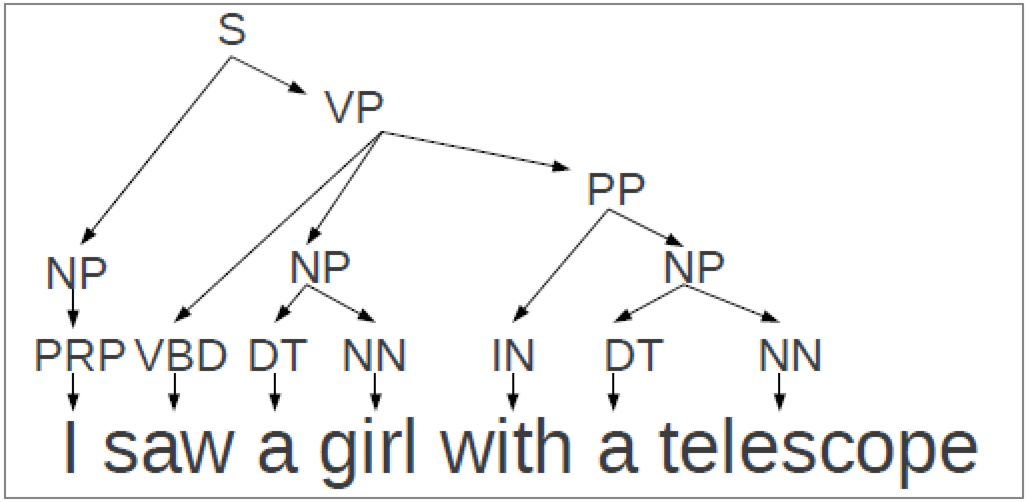
- “Parsing” resolves structural ambiguity in a formal way

Two Types of Parsing

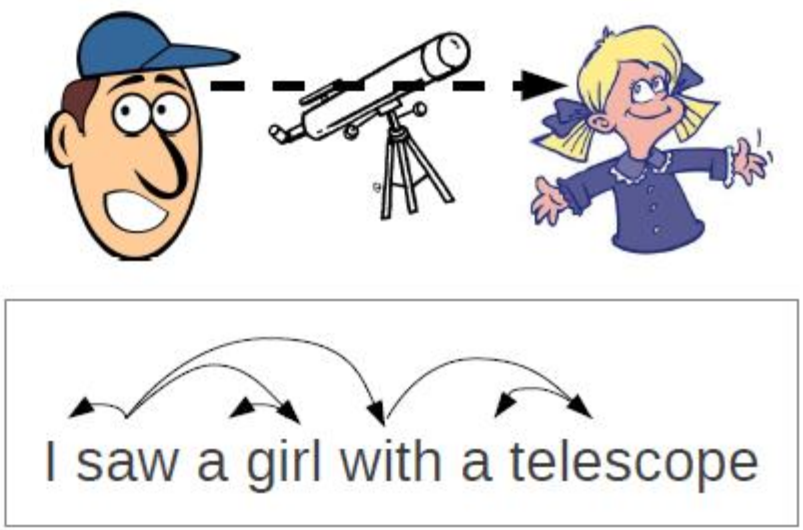
- Dependency: focuses on relations between words

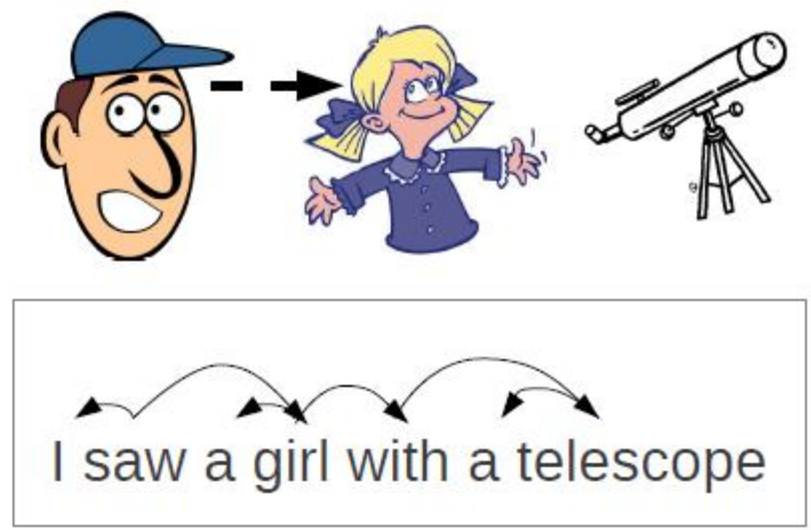


- Phrase structure: focuses on identifying phrases and their recursive structure



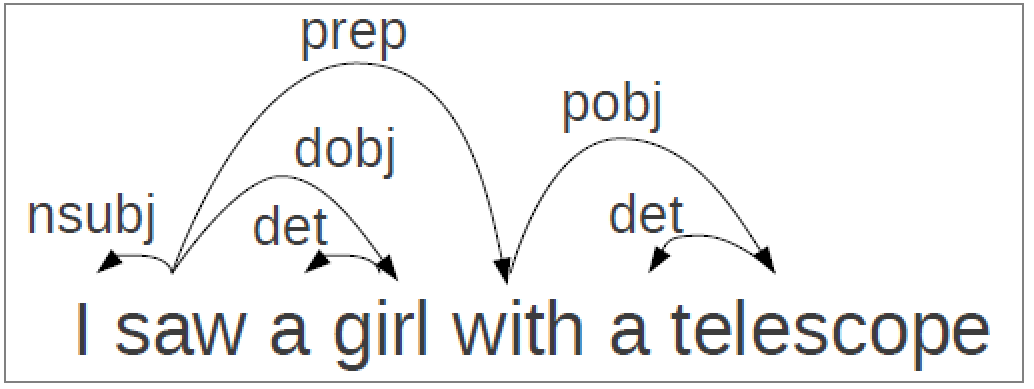
Dependencies Also Resolve Ambiguity



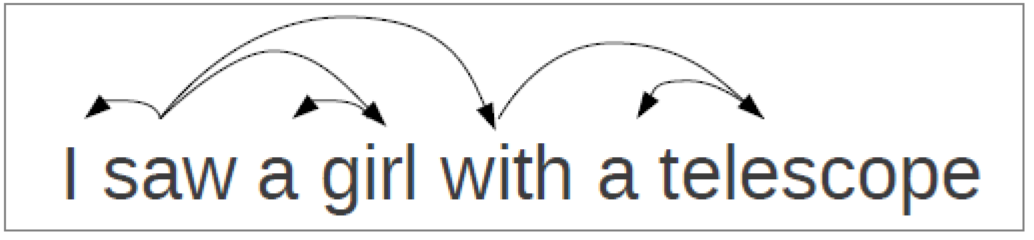


Dependencies

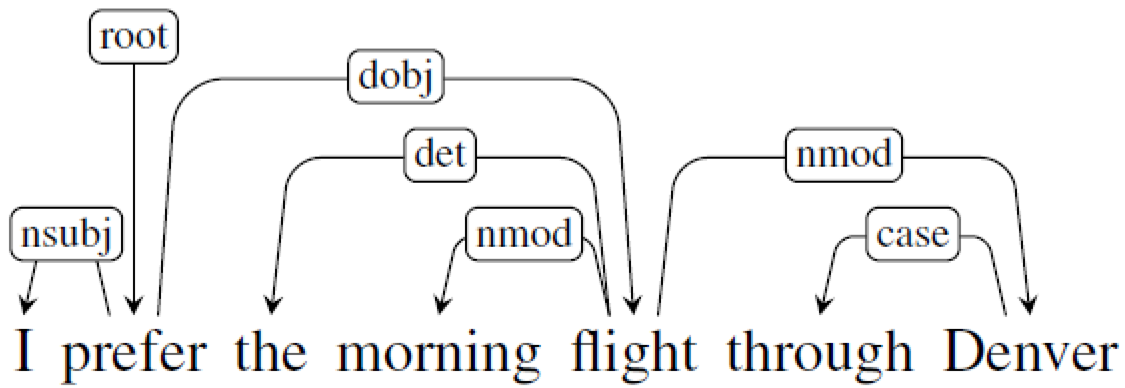
- Typed: Label indicating relationship between words



- Untyped: Only which words depend



Dependency Tree with Labels



Comparison

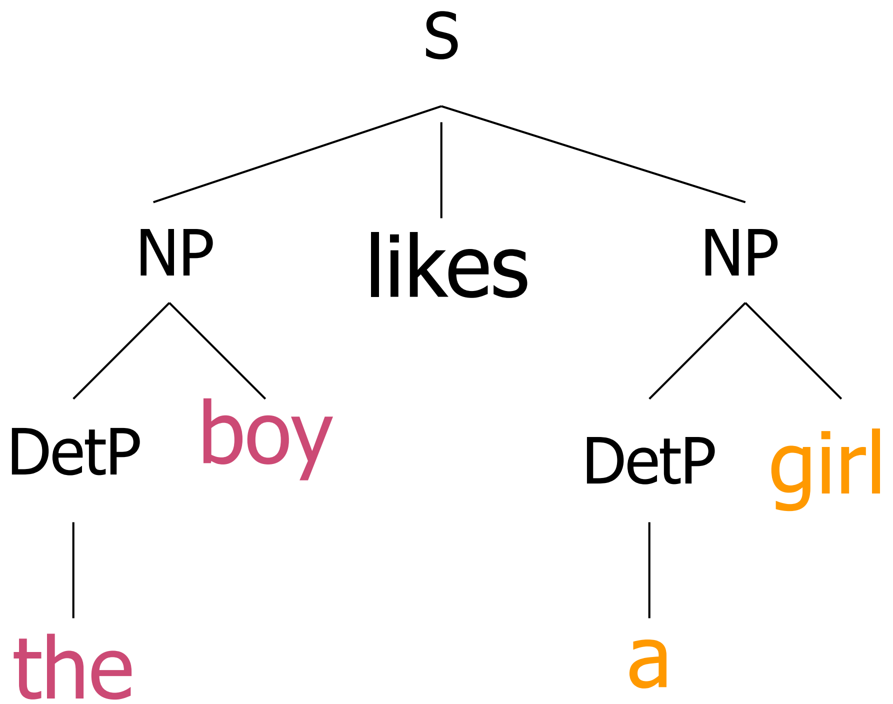
Dependency structures explicitly represent

- Head-dependent relations (directed arcs)
- Functional categories (arc labels)
- Possibly some structural categories (parts-of-speech)

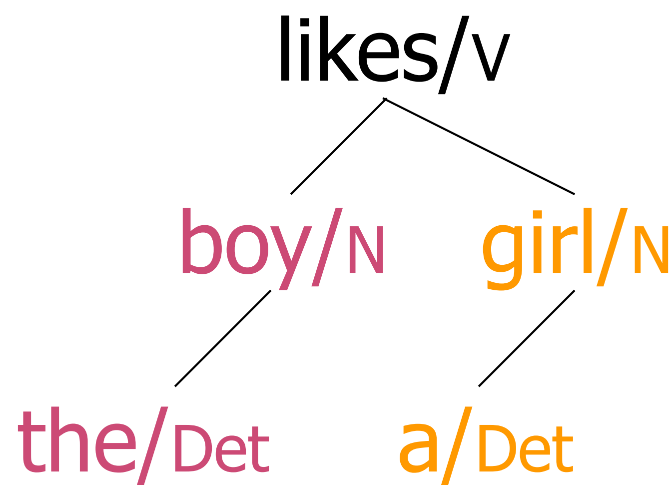
Phrase structure explicitly represent

- Phrases (non-terminal nodes)
- Structural categories (non-terminal labels)
- Possibly some functional categories (grammatical functions)

Phrase Structure and Dependency Structure

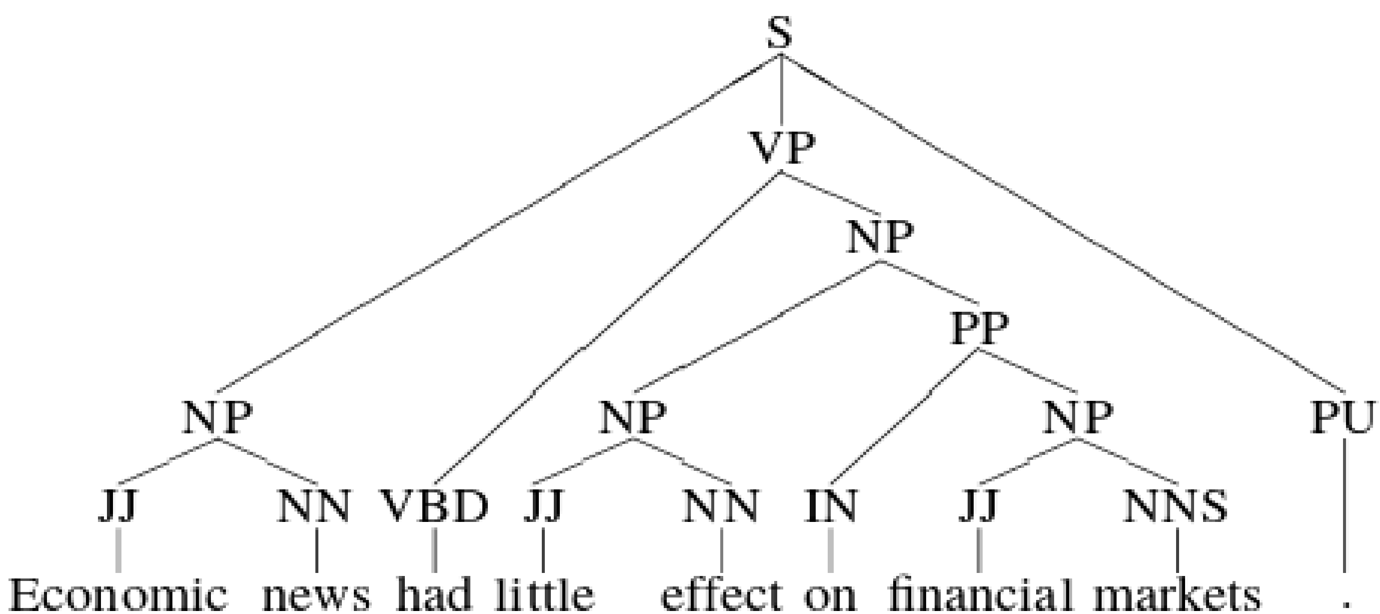


Only leaf nodes labeled with words!

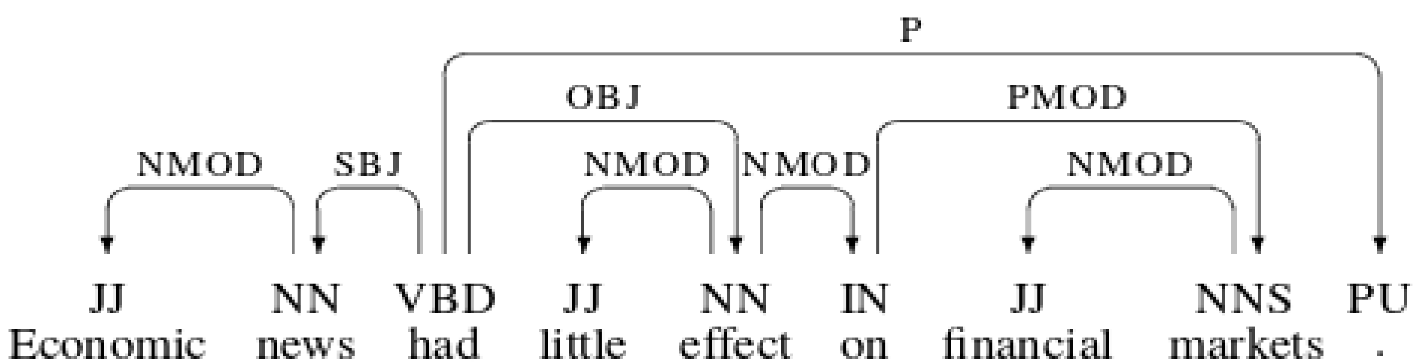


All nodes are labeled with words!

Phrase vs. dependency structure



Constituent structure for English sentence from the Penn Treebank



Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

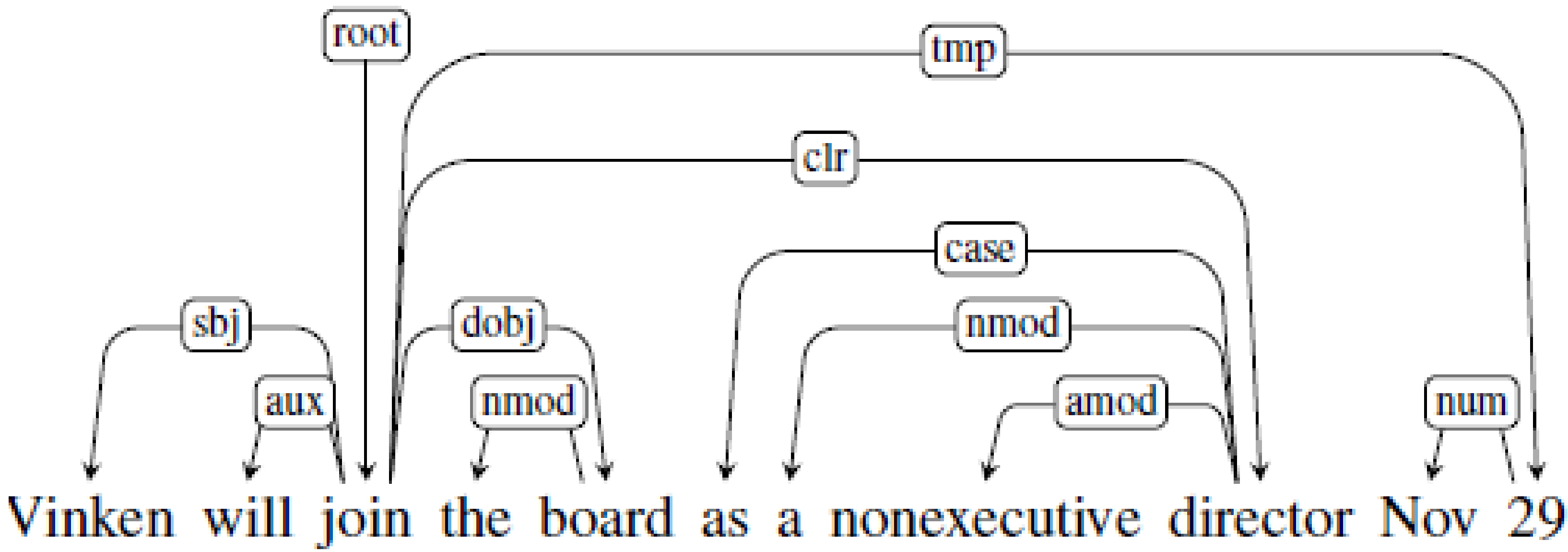
Selected dependency relations from the Universal Dependency set. (de Marn-effe et al., 2014)

Grammatical Relations

Types of relations between words
- Arguments: subject, object, indirect object, prepositional object
- Adjuncts: temporal, locative, causal, manner, ...
- Function Words

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Examples of core Universal Dependency relations.


root
tmp
clr
case
sbj
dobj
nmod
aux
nmod
amod
num
Vinken will join the board as a nonexecutive director Nov 29

Learning DG over PSG

Dependency Parsing is more straightforward

- Parsing can be reduced to labeling each token w_i with w_j

Direct encoding of predicate-argument structure

- Fragments are directly interpretable

Dependency structure independent of word order

- Suitable for free word order languages (like Indian languages)

Dependency Tree

Formal definition

- An input word sequence $w_1 \ldots w_n$
- Dependency graph $D = (W,E)$ where
  - W is the set of nodes i.e. word tokens in the input seq.
  - E is the set of unlabeled tree edges (w_i, w_j) ($w_i, w_j \in W$).
  - (w_i, w_j) indicates an edge from w_i (parent) to w_j (child).

Task of mapping an input string to a dependency graph satisfying certain conditions is called dependency parsing

Dependency Parsing

Dependency based parsers can be broadly categorized into

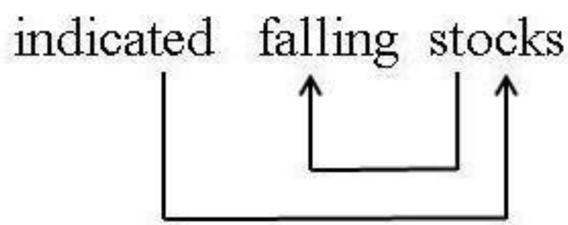
- Grammar driven approaches
  - Parsing done using grammars.
- Data driven approaches
  - Parsing by training on annotated/un-annotated data.

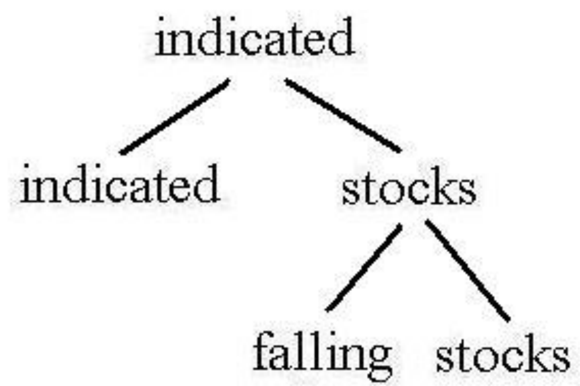
These approaches are not mutually exclusive

Dynamic Programming

Basic Idea: Treat dependencies as constituents.

Use, e.g. , CKY parser (with minor modifications)





Dependency Parsers for download

- MST parser by Ryan McDonald
- Malt parser by Joakim Nivre
- Stanford parser – Neural Network based dependency parser

- The availability of treebanks has been crucial to the development of data-driven parsing

Data-driven dependency parsing

Transition-based models

- Define a transition system (state machine) for mapping a sentence to its dependency graph.
- Learning: Induce a model for predicting the next state transition, given the transition history.
- Parsing: Construct the optimal transition sequence, given the induced model.

Data-driven dependency parsing

Graph-based models

- Define a space of candidate dependency graphs for a sentence.
- Learning: Induce a model for scoring an entire dependency graph for a sentence.
- Parsing: Find the highest-scoring dependency graph, given the induced model.

In transition-based parsing, sentences are processed in a linear left to right pass; at each position, the parser needs to choose from a set of possible actions defined by the transition strategy

Transition Systems

- A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$, where
  - C is a set of configurations.
  - T is a set of transitions, each of which is a (partial) function $t: C \rightarrow C$.
  - c_s is an initialization function, mapping a sentence x to its initial configuration $c_s(x)$.
  - $C_t \subseteq C$ is a set of terminal configurations.

Transition Systems

- A *configuration* for a sentence x is a triple c = (Σ, B, A), where:
  - Σ is a stack of nodes in V_x, known as the **Stack.**
  - B is a list of nodes in V_x, known as the **Buffer.**
  - A is a set of dependency arcs in $V_x \times L \times V_x$ (for some set L of dependency labels).

- A *transition sequence* for a sentence x in transition systems $S = (C, T, c_s, C_t)$ is a sequence $C_{0,m} = (c_0, c_1, \ldots, c_m)$ of configurations.
  - $C_0 = c_s(x)$.
  - $c_m \in C_t$.
  - For every i ($1 \leq i \leq m$), $c_i = t(c_{i-1})$ for some $t \in T$.

- The parse assigned to x by $C_{0,m}$ is the dependency graph $G_{cm} = (V_x , A_{cm})$,
  - A_{cm} is the set of dependency arcs in cm .
  - More generally, the dependency graph associated with any configuration c_i for x is $G_{ci} = (V_x, A_{ci})$.

Oracle: an oracle is a deterministic (or may be non-deterministic) algorithm that taking a gold tree associated to a sentence, it produces the set of actions the algorithm should follow in order to reach the gold tree.

Nivre shift reduce dependency parser

Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A\cup$ nsubj(has,He)
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A\cup$amod(control,good)
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A\cup$ dobj(has,control)
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A\cup$ root(ROOT,has)

Stanford Dependency parser -NNDP

Softmax layer:

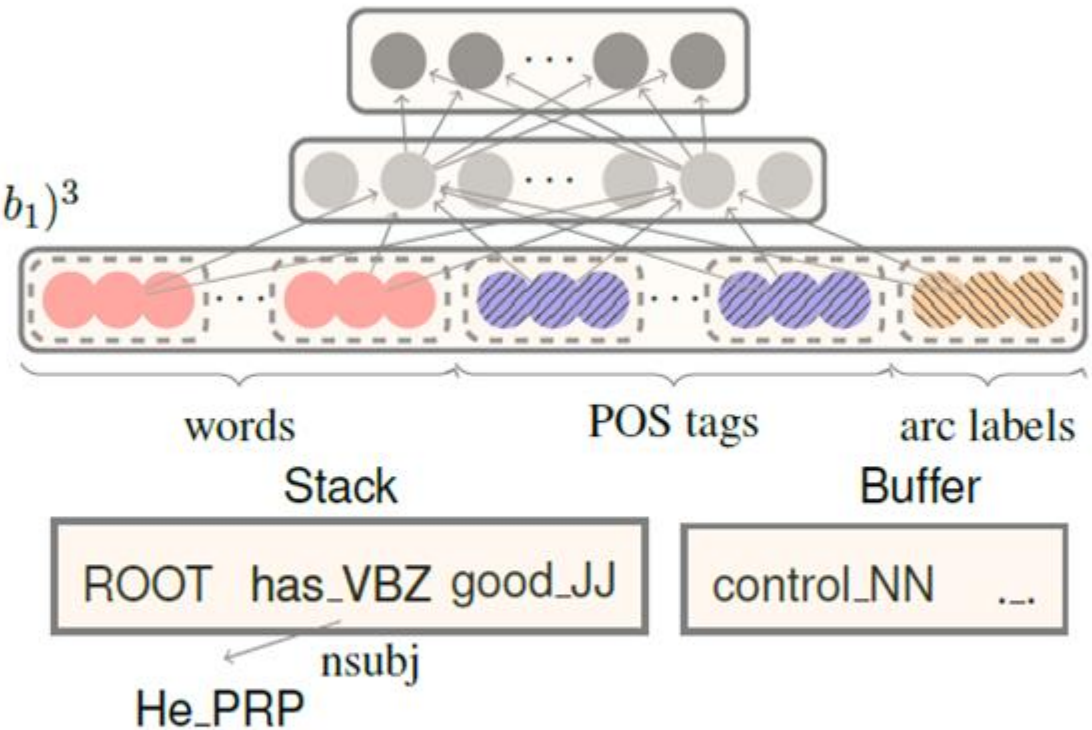
$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

Configuration



Input Layer - Word, POS, Dependency Label embedding

First, as usual word embeddings, we represent each word as a d-dimensional vector $e_i^w \in \mathbb{R}^d$ and the full embedding matrix is $E^w \in \mathbb{R}^{d \times N_w}$ where N_w is the dictionary size. Meanwhile, we also map POS tags and arc labels to a d-dimensional vector space, where $e_i^t, e_j^l \in \mathbb{R}^d$ are the representations of i^{th} POS tag and j^{th} arc label. Correspondingly, the POS and label embedding matrices are $E^t \in \mathbb{R}^{d \times N_t}$ and $E^l \in \mathbb{R}^{d \times N_l}$ where N_t and N_l are the number of distinct POS tags and arc labels.

A set of elements chosen based on the stack / buffer positions for each type of information (word, POS or label), which might be useful for our predictions.

Every hidden unit is computed by a (non-linear) mapping on a weighted sum of input units plus a bias.

Using $g(x) = x^3$ can model the product terms of $x_i x_j x_k$ for any

$$g(w_1x_1 + \ldots + w_mx_m + b) =$$
$$\sum_{i,j,k}(w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \ldots$$

It was shown that cube activation function better captures the interaction of word, PoS and label embeddings which is a very desired property of dependency parsing.

A softmax layer is finally added on the top of the hidden layer for modeling multi-class probabilities $p = \mathtt{softmax}(W_2 h)$, where $W_2 \in \mathbb{R}^{|\mathcal{T}| \times d_h}$.

Shallow Parsing

Shallow parsing (also chunking, "light **parsing**") is an analysis of a sentence which identifies the constituents (noun groups, verbs, verb groups, etc.), but does not specify their internal structure, nor their role in the main sentence.

CONLL Chunking task

Shallow parsing has been influenced by Abney's work , who has suggested to "chunk" sentences to base level phrases.

Chunking was defined as the task of dividing the text into syntactically non-overlapping phrases.

Here by `non-overlapping' we mean, one word can become a member of only one chunk.

Example

"He reckons the current account deficit will narrow to only #1.8 billion in September." can be divided as follows:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September].

Useful for many NLP tasks:

- information retrieval
- information extraction
- text summarization
- Machine Translation

Phrases in shallow parsing

type
NP (noun phrase)
VP (verb phrase)
PP (prepositional phrase)
ADVP (adverb phrase)
SBAR (subordinated clause)
ADJP (adjective phrase)
PRT (particles)
CONJP (conjunction phrase)
INTJ (interjection)
LST (list marker)
UCP (unlike coordinated phrase)

Examples

Sentence:

TSA spokeswoman Lisa Farbstein said the dogs undergo 12 weeks of training, which costs about $200,000, factoring in food, vehicles and salaries for trainers.

Parsed Sentence:

[NP TSA spokeswoman Lisa Farbstein] [VP said] [NP the dogs] [VP undergo] [NP 12 weeks]

[PP of] [NP training] , [NP which] [VP costs] [PP about] [NP $200,000] ,

[NP factoring] [PP in] [NP food, vehicles and salaries] [PP for] [NP trainers] .

The horse raced past the barn fell.

[NP The horse] [VP raced] [PP past] [NP the barn] [VP fell] .

I am going to school tomorrow.

[NP I] [VP am going] [PP to] [NP school] [NP tomorrow] .

A large number of the systems applied to the CoNLL-2000 shared task uses statistical methods.

[NP A large number] [PP of] [NP the systems] [VP applied] [PP to]

[NP the CoNLL-2000 shared task] [VP uses] [NP statistical methods] .

Approaches: ML Chunking

- Require annotated corpus
- Train classifier to classify each element of input in sequence (e.g. IOB Tagging)
  - B (beginning of sequence)
  - I (internal to sequence)
  - O (outside of any sequence)
  - No end-of-chunk coding – it's implicit
  - Easier to detect the beginning than the end

He/B-NP reckons/B-VP the/B-NP
current/I-NP account/I-NP
deficit/I-NP will/B-VP narrow/I-VP
to/B-PP only/B-NP £/I-NP
1.8/I-NP billion/B-NP in/B-PP
September/B-NP ./O

Mostly used System

- HMM
- CRF
- FSA
- SVM
- Memory based
- Maximum Entropy