Parsing

CMSC 723 / LING 723 / INST 725

Hal Daumé III [he/him]

29 Oct 2019

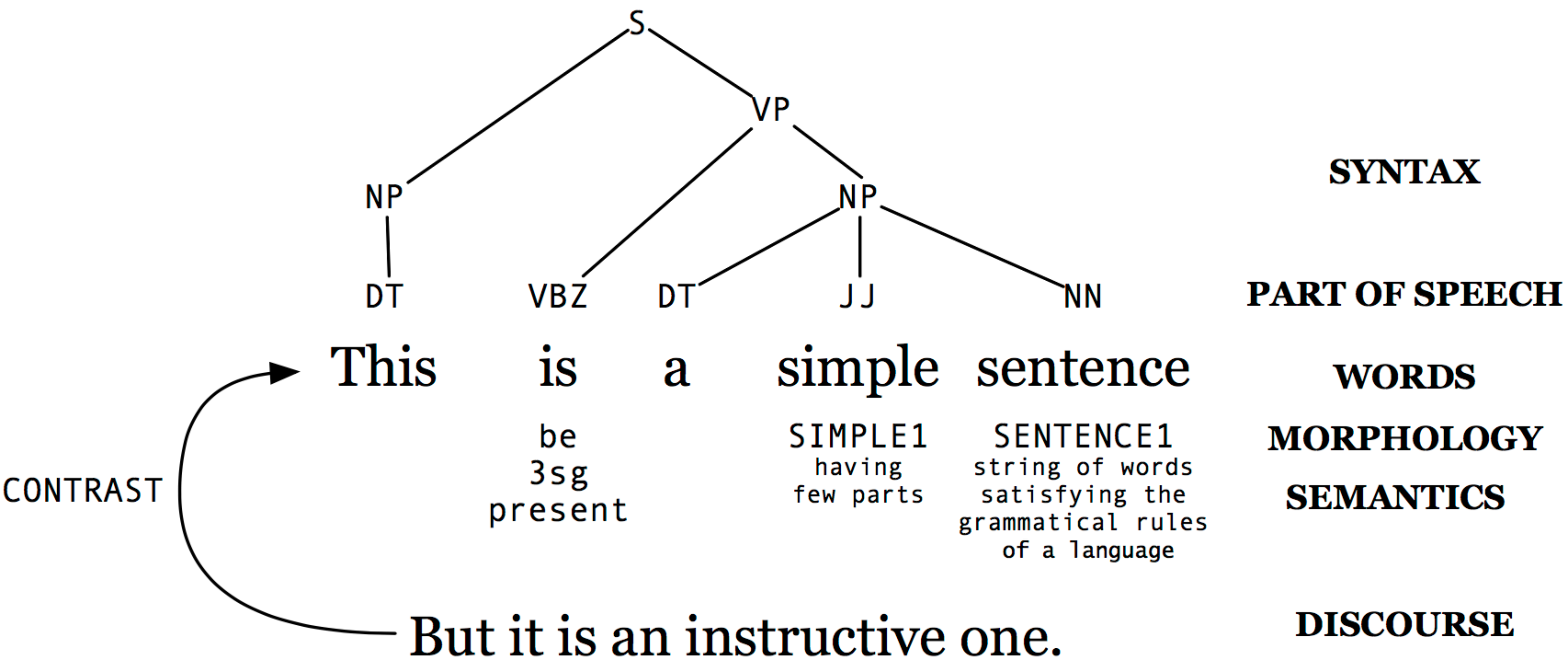
(many slides c/o Marine Carpuat or Joakim Nivre or Ryan McDonald)

Announcements, logistics

- Project
  - If you haven’t provided feedback already, please do it by the end of the day
  - I will also send each team feedback today/tomorrow (email me if you don’t get it)
  - **P2 due one week from today**
  - Assignment of teams to TAs posted, please meet with me and also with your TA before Thanksgiving break (sign-ups will be posted on ELMS)
- Homeworks 4 & 5 officially merged
  - Posted by next Tuesday
  - Worth 14% of grade (== equivalent to two HW assignments)
  - Due original HW5 deadline (26 Nov, right before Thanksgiving break)
- Grading
  - HW2 out – graded version soon (will post on ELMS)
  - Midterm out – MIN: 54.0, MED: 86.0, MAX: 98.0, MEAN: 83.82, STD: 10.7
    Solution posted tomorrow
  - HW3 – will be done soon (mechanical turk part)

Up until now…

- Input representations
  - Bag of words
  - Featurization
  - Sequences
- Output representations
  - Labels
  - Sequences (either 1-1 or not)
- New few lectures
  - Tree-based representations for input *or* output


S
VP
NP
NP
SYNTAX
DT
VBZ
DT
JJ
NN
PART OF SPEECH
This
is
a
simple
sentence
WORDS
be
3sg
present
SIMPLE1
having
few parts
SENTENCE1
string of words
satisfying the
grammatical rules
of a language
MORPHOLOGY
SEMANTICS
CONTRAST
But it is an instructive one.
DISCOURSE

Syntax & Grammar

- Syntax
  - From Greek syntaxis, meaning “setting out together”
  - refers to the way words are arranged together.
- Grammar
  - Set of structural rules governing composition of clauses, phrases, and words in any given natural language
  - Descriptive, not prescriptive
  - Panini’s grammar of Sanskrit ~2000 years ago

Syntax and Grammar

- Goal of syntactic theory
  - “explain how people combine words to form sentences and how children attain knowledge of sentence structure”

- Grammar
  - implicit knowledge of a native speaker
  - acquired without explicit instruction
  - minimally able to generate all and only the possible sentences of the language

[Philips, 2003]

Syntax in NLP

- Syntactic analysis often a key component in applications
  - Grammar checkers
  - Dialogue systems
  - Question answering
  - Information extraction
  - Machine translation
  - …

Two views of syntactic structure

- Constituency (phrase structure)
  - Phrase structure organizes words in nested constituents

- Dependency structure
  - Shows which words depend on (modify or are arguments of) which on other words

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/postposed
  - …

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 many different theories of grammar and competing analyses of the same data!
- Our approach
  - Focus primarily on the “machinery”

Context-Free Grammars

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

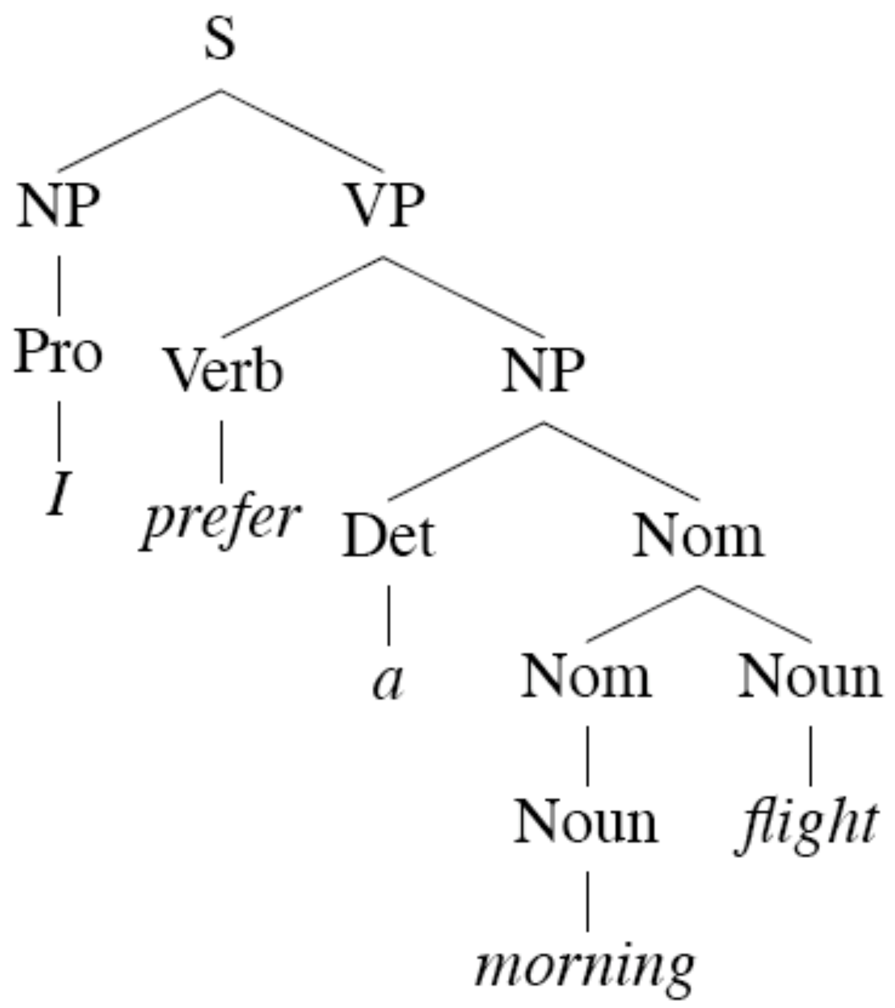
Context-Free Grammars

- Terminals
  - We'll take these to be words
- 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

An Example Grammar

Grammar Rules			Examples
S	$\rightarrow$	$NP\ VP$	I + want a morning flight
NP	$\rightarrow$	$Pronoun$	I
	$\mid$	$Proper\text{-}Noun$	Los Angeles
	$\mid$	$Det\ Nominal$	a + flight
$Nominal$	$\rightarrow$	$Nominal\ Noun$	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

Parse Tree: Example



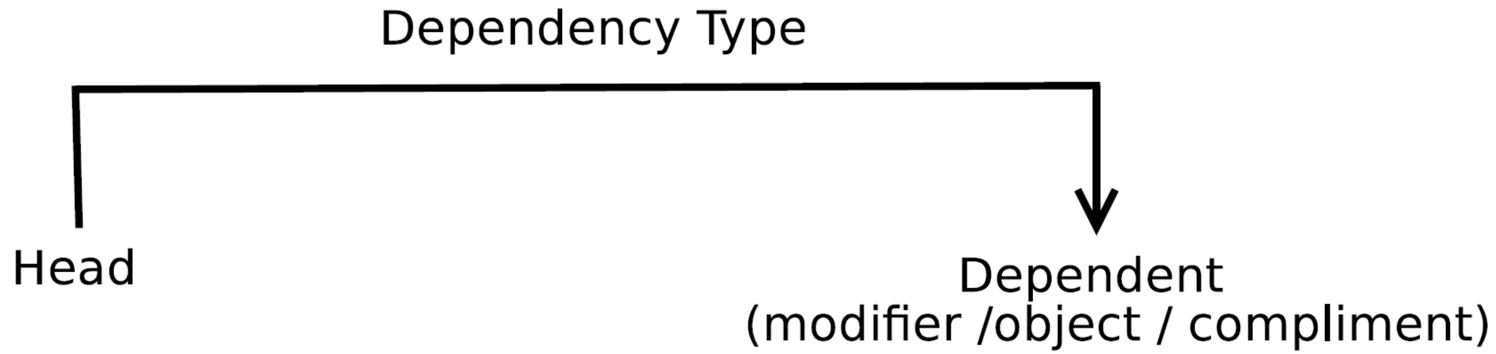
Dependency Grammars

- CFGs focus on constituents
  - Non-terminals don’t actually appear in the sentence

- In dependency grammar, a parse is a graph (usually a tree) where:
  - Nodes represent words
  - Edges represent dependency relations between words (typed or untyped, directed or undirected)

Dependency Grammars

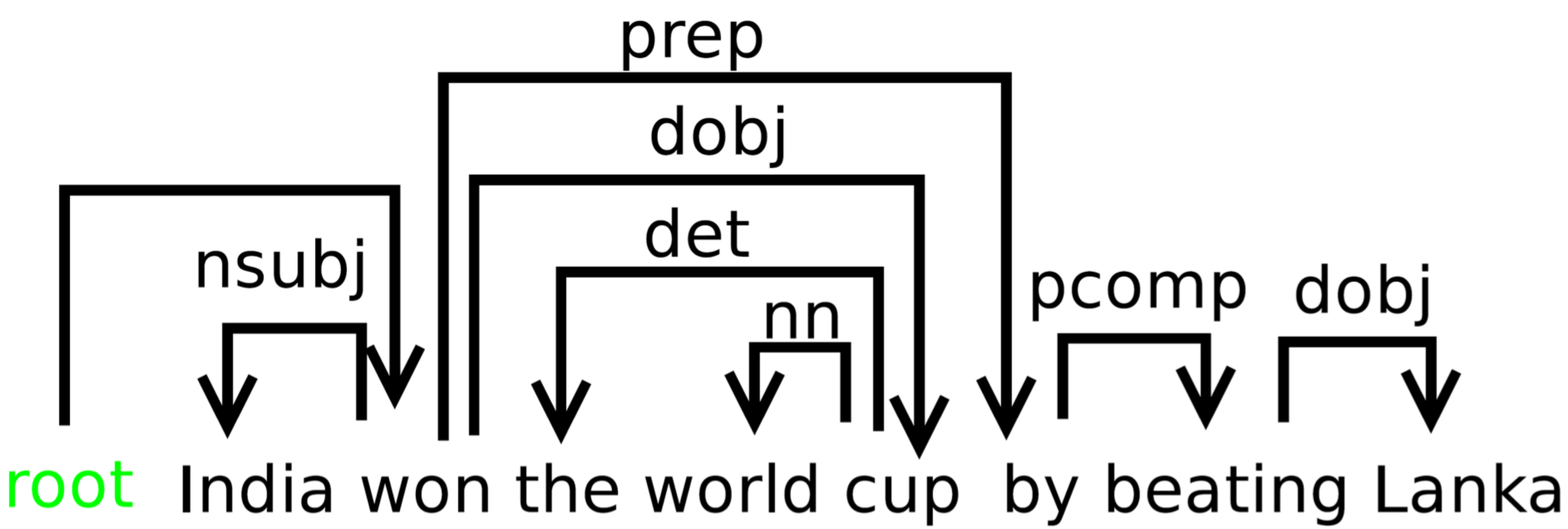
- Syntactic structure = lexical items linked by binary asymmetrical relations called dependencies



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



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*.

Figure 14.3 Examples of core Universal Dependency relations.

Universal Dependencies project

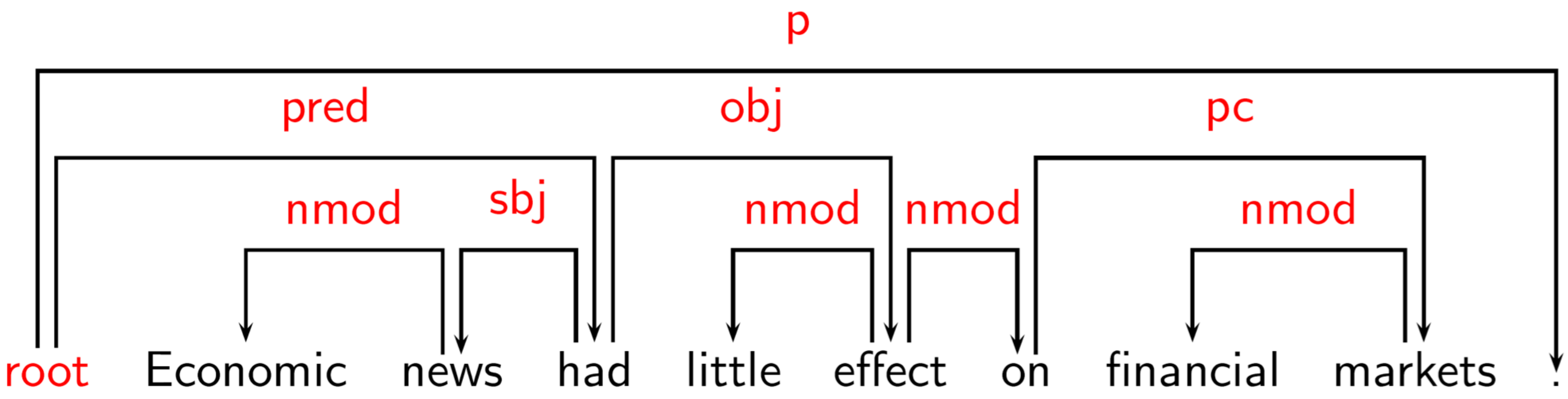
- Set of dependency relations that are
  - Linguistically motivated
  - Computationally useful
  - Cross-linguistically applicable
  - [Nivre et al. 2016]

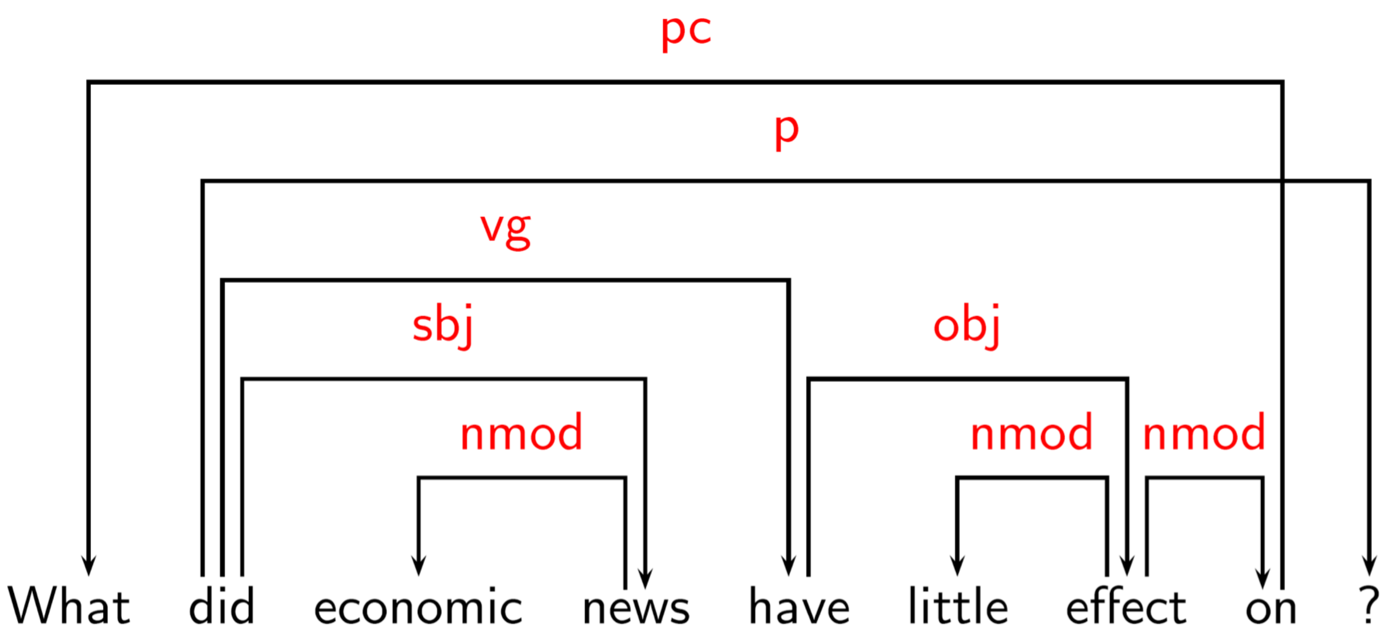
- Universaldependencies.org

Dependency formalisms

- Most general form: a graph G = (V,A)
  - V vertices: usually one per word in sentence
  - A arcs (set of ordered pairs of vertices): head-dependent relations between elements in V
- Restricting to **trees** provide computational advantages
  - Single designated ROOT node that has no incoming arcs
  - Except for ROOT, each vertex has exactly one incoming arc
  - Unique path from ROOT to each vertex in V

- Each word has a single head
- Dependency structure is connected
- There is a single root node from which there is a unique path to each word


p
pred
obj
pc
nmod
sbj
nmod
nmod
nmod
root Economic news had little effect on financial markets .


pc
p
vg
sbj
obj
nmod
nmod
nmod
What did economic news have little effect on ?

Projectivity

- **Arc** from head to dependent is **projective**
  - If there is a path from head to every word between head and dependent
- **Dependency tree** is **projective**
  - If all arcs are projective
  - Or equivalently, if it can be drawn with no crossing edges
- Projective trees make computation easier
- But most theoretical frameworks do not assume projectivity
  - Need to capture long-distance dependencies, free word order

Data-driven dependency parsing

Goal: learn a good predictor of dependency graphs

Input: sentence

Output: dependency graph/tree G = (V,A)

Can be framed as a structured prediction task

- very large output space
- with interdependent labels

2 dominant approaches: transition-based parsing and graph-based parsing

Transition-based dependency parsing

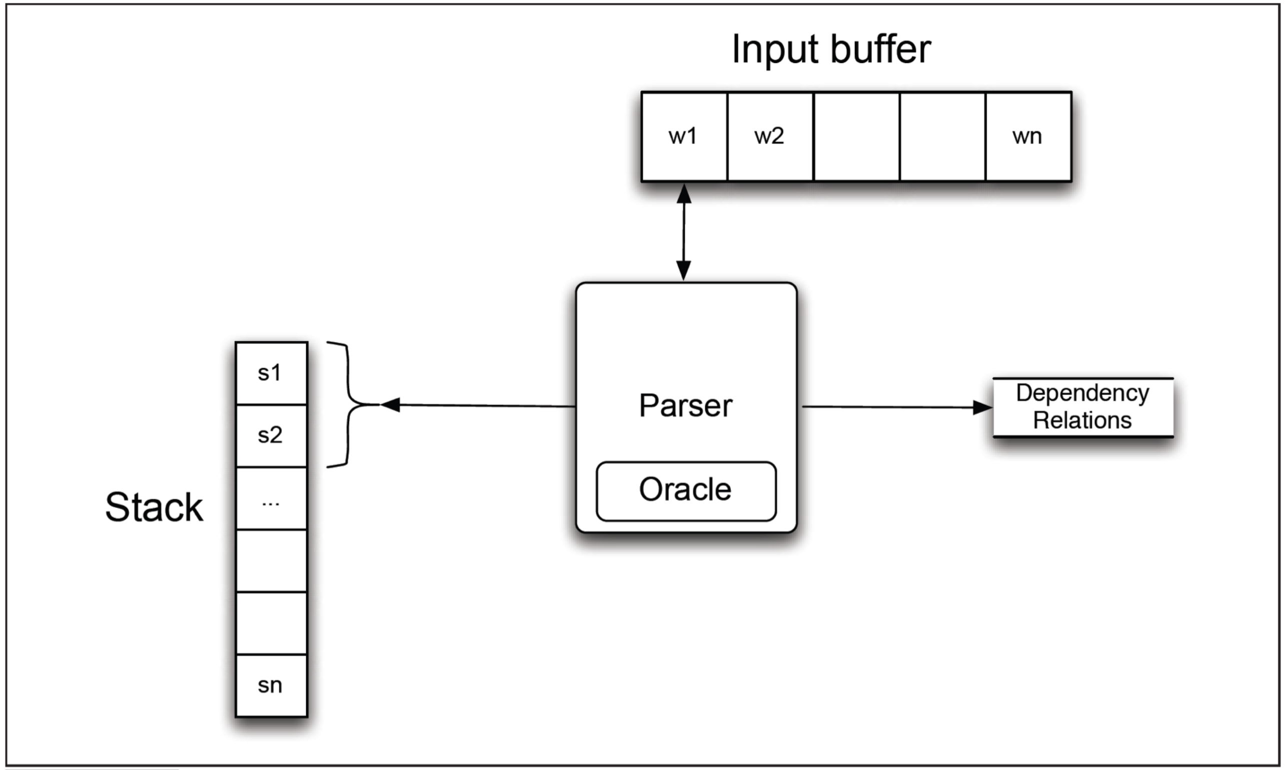


Figure 14.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

- Builds on shift-reduce parsing [Aho & Ullman, 1927]
- **Configuration**
  - **Stack**
  - **Input buffer** of words
  - Set of dependency relations
- Goal of parsing
  - find a final configuration where
  - all words accounted for
  - Relations form dependency tree

Transition operators

- Transitions: produce a new configuration given current configuration
- Parsing is the task of
  - Finding a sequence of transitions
  - That leads from start state to desired goal state

- Start state
  - Stack initialized with ROOT node
  - Input buffer initialized with words in sentence
  - Dependency relation set = empty
- End state
  - Stack and word lists are empty
  - Set of dependency relations = final parse

Arc Standard Transition System

- Defines 3 transition operators [Covington, 2001; Nivre 2003]
- LEFT-ARC:
  - create head-dependent rel. between word at top of stack and 2nd word (under top)
  - remove 2nd word from stack
- RIGHT-ARC:
  - Create head-dependent rel. between word on 2nd word on stack and word on top
  - Remove word at top of stack
- SHIFT
  - Remove word at head of input buffer
  - Push it on the stack

Arc standard transition systems

- Preconditions
  - ROOT cannot have incoming arcs
  - LEFT-ARC cannot be applied when ROOT is the 2nd element in stack
  - LEFT-ARC and RIGHT-ARC require 2 elements in stack to be applied

Transition-based Dependency Parser

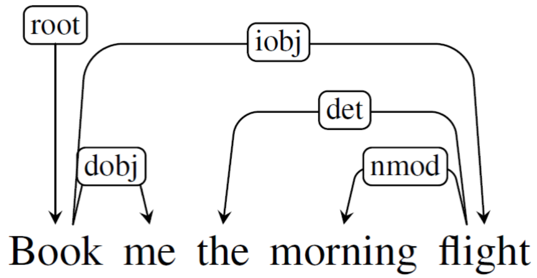
function DEPENDENCYPARSE(*words*) **returns** dependency tree

  state ← {[root], [*words*], [] } ; initial configuration
  while *state* **not final**
    t ← ORACLE(*state*) ; choose a transition operator to apply
    state ← APPLY(*t*, *state*) ; apply it, creating a new state
  return *state*

Figure 14.6 A generic transition-based dependency parser

- Assume an oracle
- Parsing complexity
  - Linear in sentence length!
- Greedy algorithm
  - Unlike Viterbi for POS tagging

Transition-Based Parsing Illustrated



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Figure 14.7 Trace of a transition-based parse.

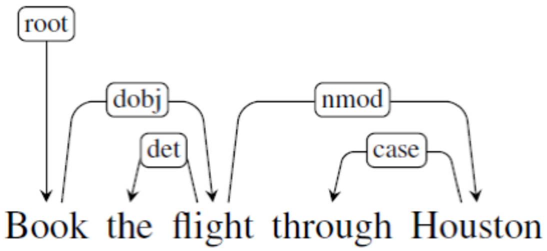
Where to we get an oracle?

- Multiclass classification problem
  - Input: current parsing state (e.g., current and previous configurations)
  - Output: one transition among all possible transitions
  - Q: size of output space?

- Supervised classifiers can be used
  - E.g., perceptron
- Open questions
  - What are good features for this task?
  - Where do we get training examples?

Generating Training Examples

- What we have in a treebank



- What we need to train an oracle
  - Pairs of configurations and

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Figure 14.8 Generating training items consisting of configuration/predicted action pairs by simulating a parse with a given reference parse.

Generating training examples

- Approach: simulate parsing to generate reference tree

- Given
  - A current config with stack S, dependency relations Rc
  - A reference parse (V,Rp)
- Do

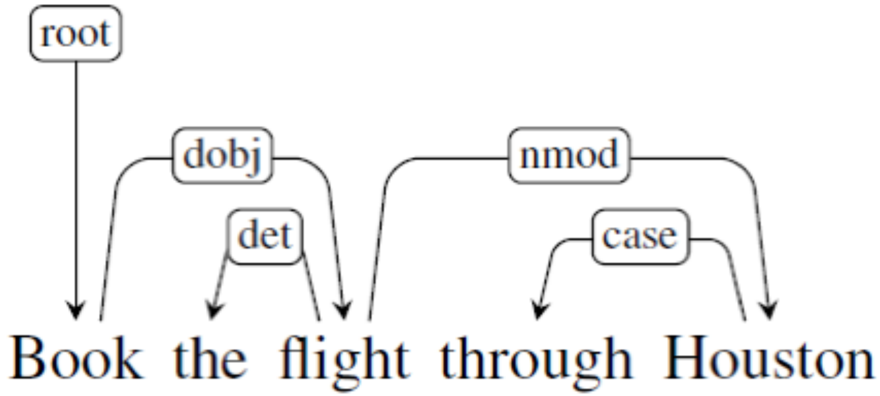
LEFTARC(r): **if** $(S_1\ r\ S_2) \in R_p$
RIGHTARC(r): **if** $(S_2\ r\ S_1) \in R_p$ **and** $\forall r', w\ s.t.(S_1\ r'\ w) \in R_p$ **then** $(S_1\ r'\ w) \in R_c$
SHIFT: **otherwise**

Let's try it out

LEFTARC(r): **if** $(S_1\ r\ S_2) \in R_p$

RIGHTARC(r): **if** $(S_2\ r\ S_1) \in R_p$ **and** $\forall r', w\ s.t.(S_1\ r'\ w) \in R_p$ **then** $(S_1\ r'\ w) \in R_c$

SHIFT: **otherwise**



Features

- Configuration consist of stack, buffer, current set of relations
- Typical features
  - Features focus on top level of stack
  - Use word forms, POS, and their location in stack and buffer

Features example

- Given configuration

Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled $\rightarrow$ United)
		(flights $\rightarrow$ morning)
		(flights $\rightarrow$ the)

- Example of useful features

$$\langle s_1.w = \textit{flights}, op = \textit{shift}\rangle$$
$$\langle s_2.w = \textit{canceled}, op = \textit{shift}\rangle$$
$$\langle s_1.t = \textit{NNS}, op = \textit{shift}\rangle$$
$$\langle s_2.t = \textit{VBD}, op = \textit{shift}\rangle$$
$$\langle b_1.w = \textit{to}, op = \textit{shift}\rangle$$
$$\langle b_1.t = \textit{TO}, op = \textit{shift}\rangle$$
$$\langle s_1.wt = \textit{flightsNNS}, op = \textit{shift}\rangle$$

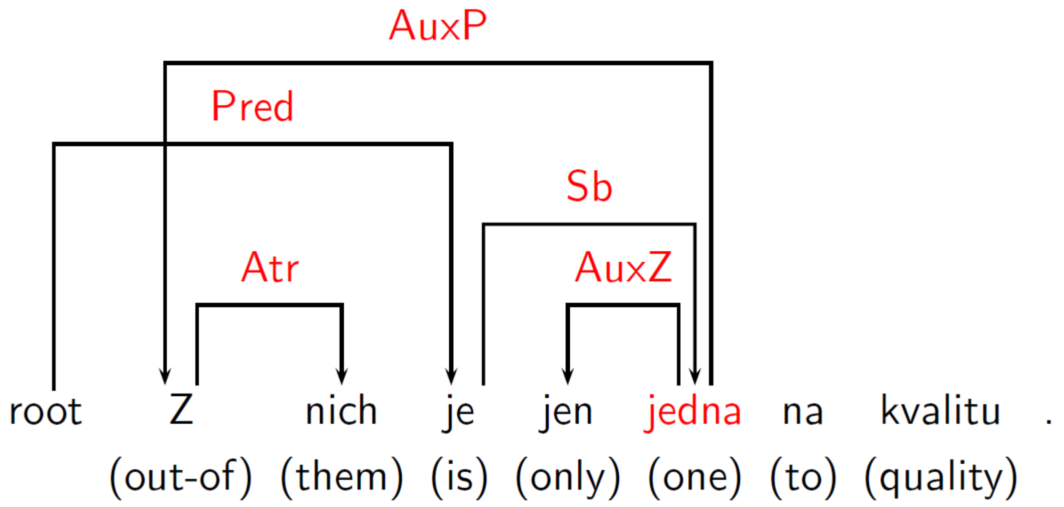
$$\langle s_1t.s_2t = \textit{NNSVBD}, op = \textit{shift}\rangle$$

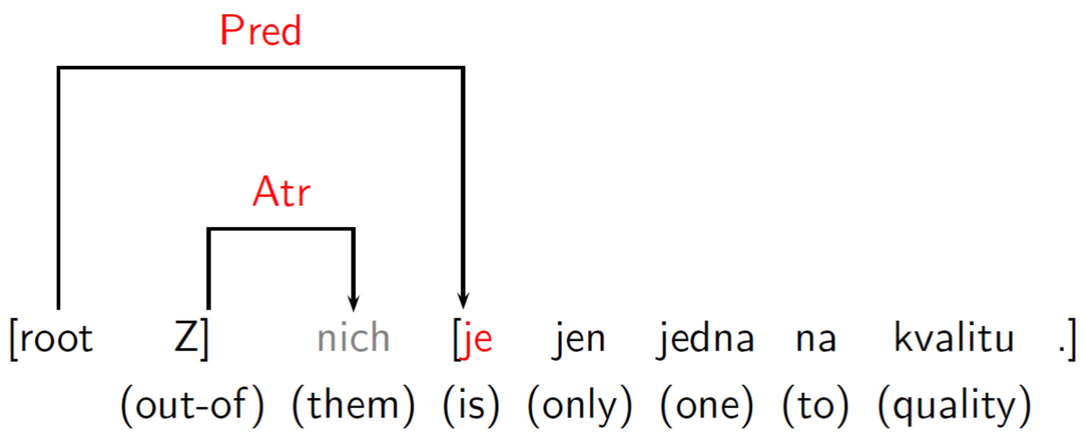
Dealing with non-projectivity

Projectivity

- **Arc** from head to dependent is **projective**
  - If there is a path from head to every word between head and dependent
- **Dependency tree** is **projective**
  - If all arcs are projective
  - Or equivalently, if it can be drawn with no crossing edges
- Projective trees make computation easier
- But most theoretical frameworks do not assume projectivity
  - Need to capture long-distance dependencies, free word order

Arc-standard parsing can't produce non-projective trees




Pred
Atr
[root Z] nich [je jen jedna na kvalitu .]
(out-of) (them) (is) (only) (one) (to) (quality)

How frequent are non-projective structures?

- Statistics from CoNLL shared task
  - NPD = non projective dependencies
  - NPS = non projective sentences

Language	**%NPD**	**%NPS**
Dutch	5.4	36.4
German	2.3	27.8
Czech	1.9	23.2
Slovene	1.9	22.2
Portuguese	1.3	18.9
Danish	1.0	15.6

How to deal with non-projectivity? (1) change the transition system

Transition		Preconditio
NP-Left$_r$	$(\sigma \| w_i \| w_k, w_j \| \beta, A) \Rightarrow (\sigma \| w_k, w_j \| \beta, A \cup \{(w_j, r, w_i)\})$	$i \neq 0$
NP-Right$_r$	$(\sigma \| w_i \| w_k, w_j \| \beta, A) \Rightarrow (\sigma \| w_i, w_k \| \beta, A \cup \{(w_i, r, w_j)\})$	

- Add new transitions
  - That apply to 2nd word of the stack
  - Top word of stack is treated as context

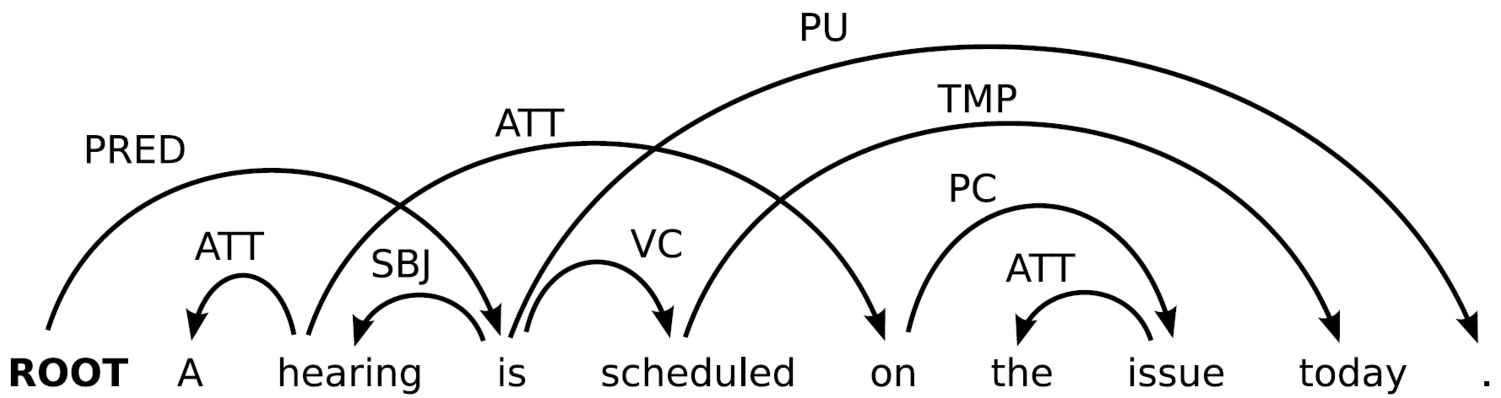
[Attardi 2006]

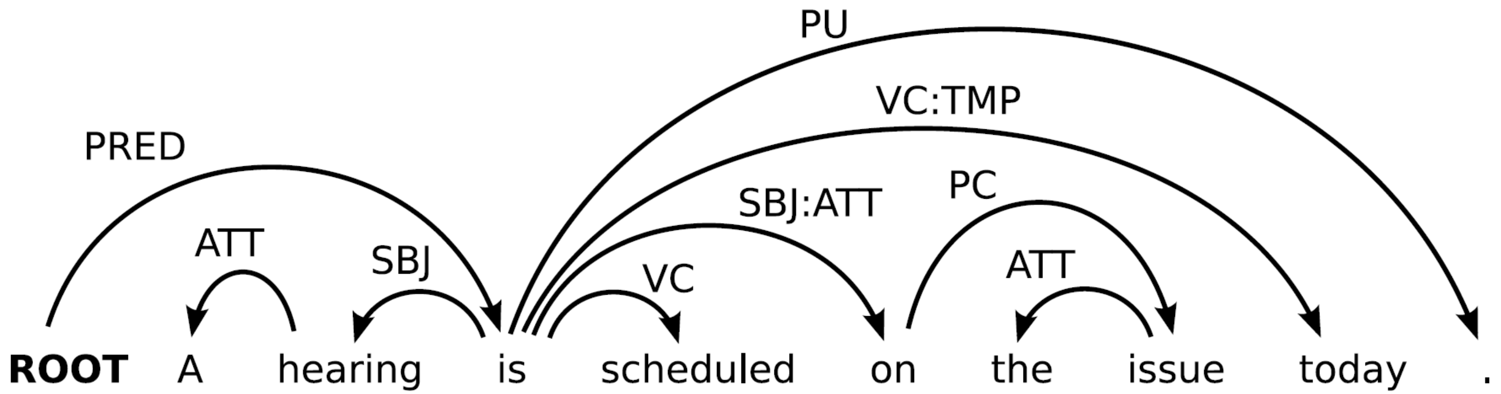
How to deal with non-projectivity? (2) pseudo-projective parsing

Solution:

- "projectivize" a non-projective tree by creating new projective arcs
- That can be transformed back into non-projective arcs in a post-processing step

How to deal with non-projectivity? (2) pseudo-projective parsing





Summary

- Two views of syntactic structures
  - Context-Free Grammars
  - Dependency grammars
  - Can be used to capture various facts about the structure of language (but not all!)

- Treebanks as an important resource for NLP

- Transition-based dependency parsing
  - Shift-reduce parsing
  - Transition system
  - Learning/predicting parsing actions