Syntactic Dependencies

CS 585, Fall 2015

Introduction to Natural Language Processing

http://people.cs.umass.edu/~brenocon/inlp2015/

Brendan O'Connor

College of Information and Computer Sciences

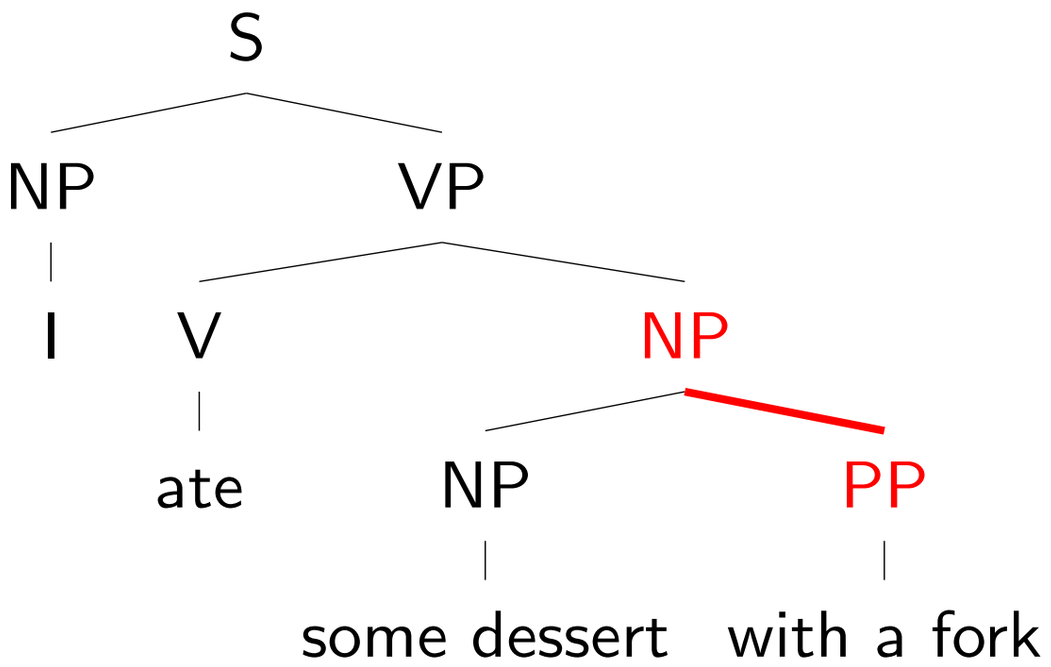
University of Massachusetts Amherst

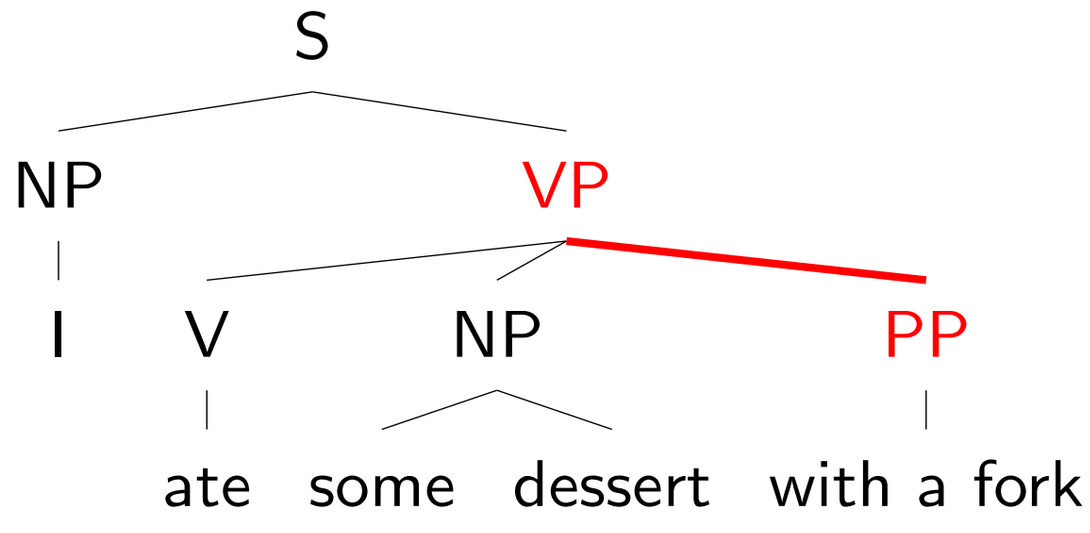
- today:
  - syntactic dependencies
  - start on coreference

- Longer distance graphs among words and phrases in a text.

Dependencies (vs. Constituents)

Disambiguation with lexical information





- (P)CFG structural information doesn't tell us much about which is more likely
- Lexical knowledge might help? (Or other knowledge?)
  - dessert -> with -> fork
  - ate -> with -> fork
- Intuitively: a notion of *modification* or *argument structure*.

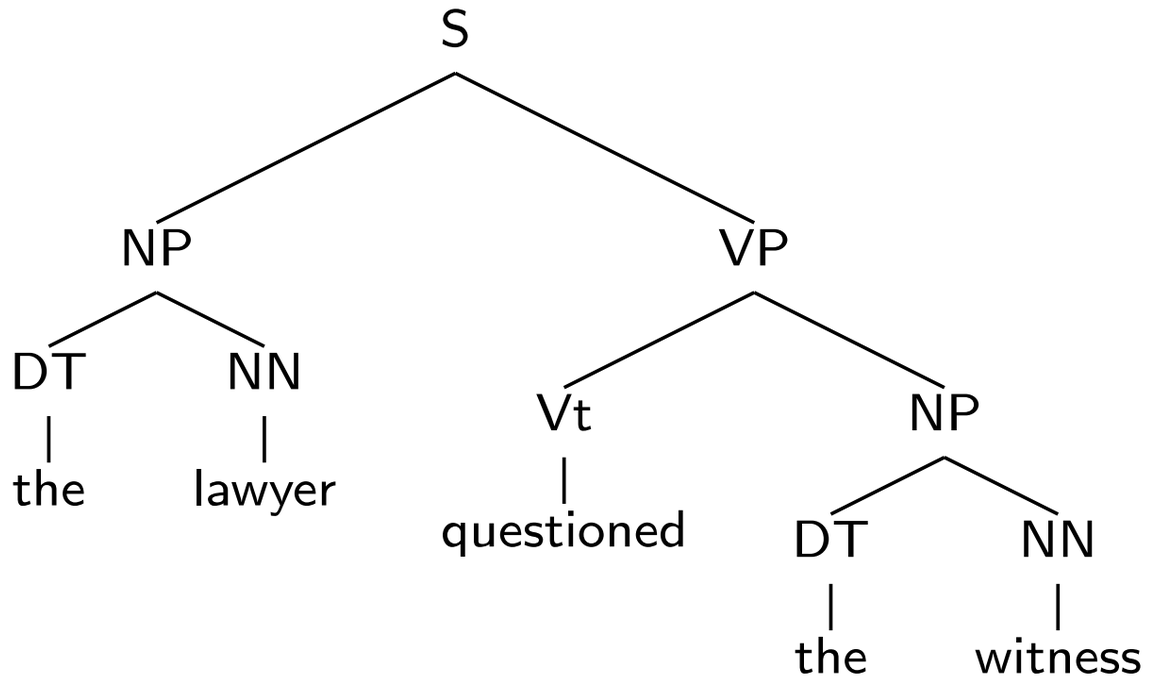
Constits -> Deps

- Syntactic theory: Every phrase has a ***head word***. It carries the primary syntactic (semantic?) properties of the phrase.
- Head rules: for every nonterminal in tree, choose one of its children to be its "head".
- Very simple example:

- NP -> Adj NP*
- NP -> NP* PP
- PP -> Prep* NP

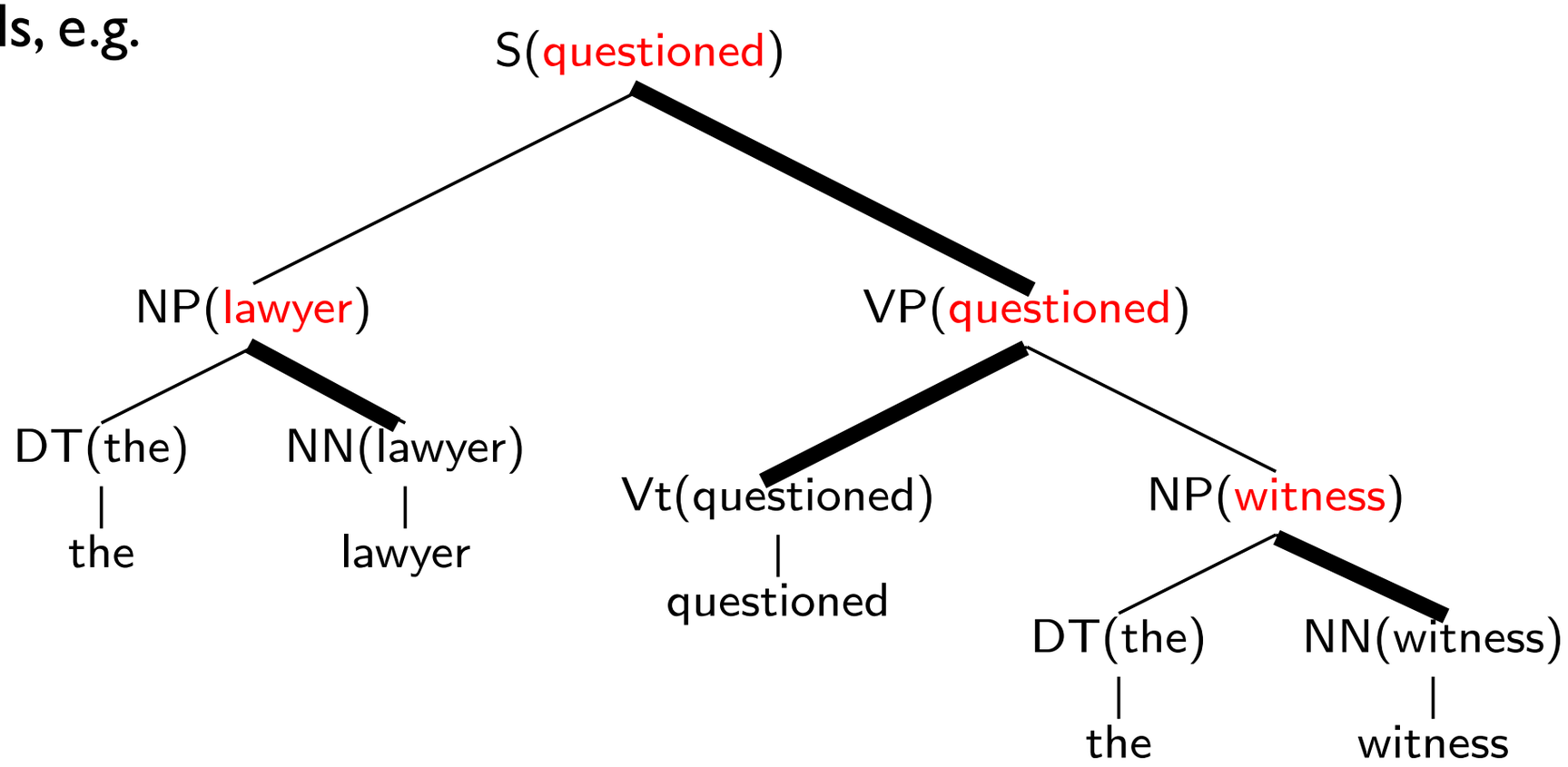
Head rules

S -> NP VP*
VP -> V* NP
NP -> Det NP*



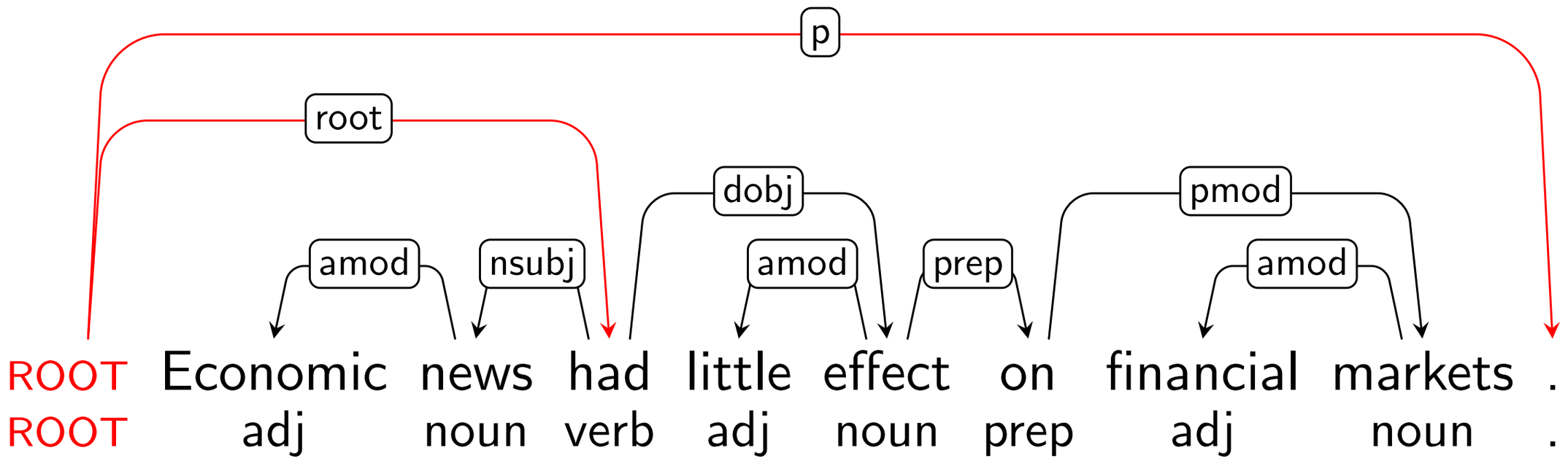
⇓

Rules more complicated for nonbinary expansions, allowing multiple non-heads, e.g.
VP -> V* PP PP



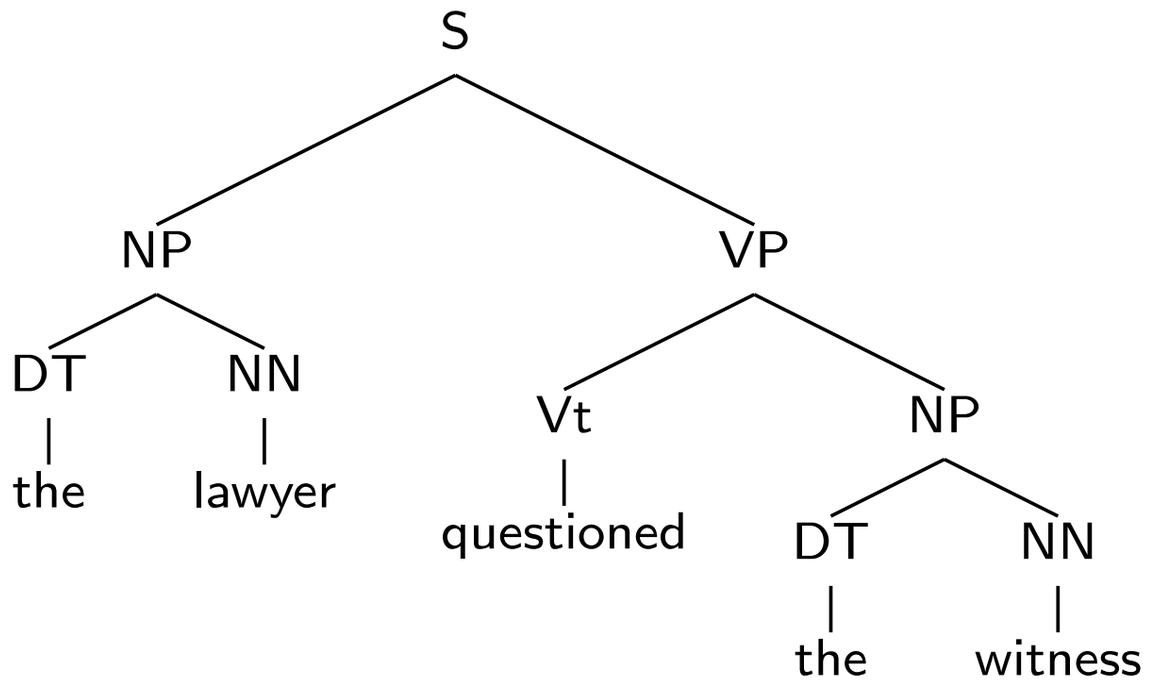
Constits -> Deps

- Head rules can be used to add words into PCFG nonterminals ("lexicalized PCFGs")
  - Helps a lot for attachment disambiguation
    eat-with-fork vs dessert-with-fork
- Or -- why not use dependency graph directly?
  - Grammatical relations are between ***individual words***
  - Graph is acyclic, connected, with a single root.



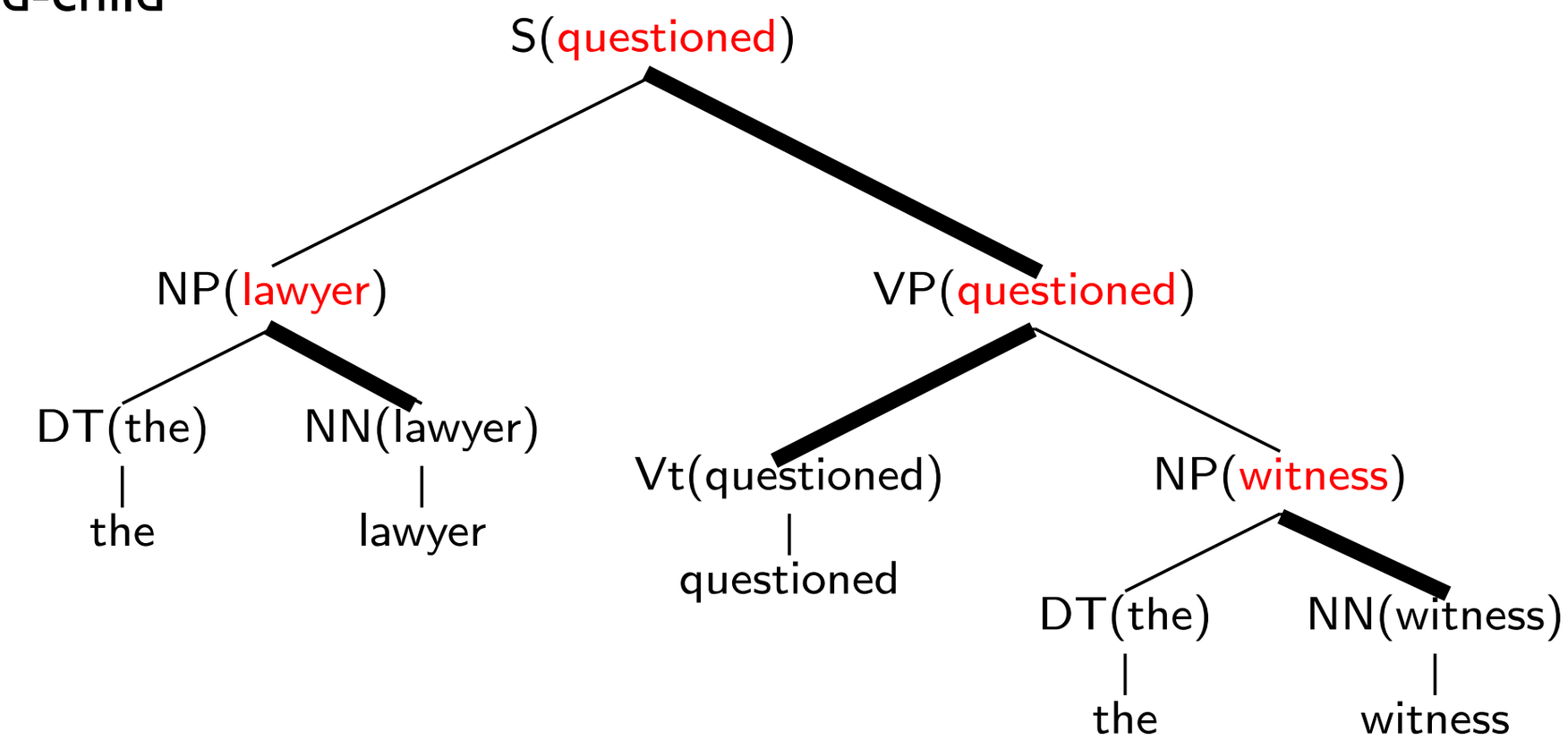
Head rules

S -> NP VP*
VP -> V* NP
NP -> Det NP*



⇓

Graph conversion: (12.7.1):
the head of each non-head-child
is subordinate to the
head of the head-child.



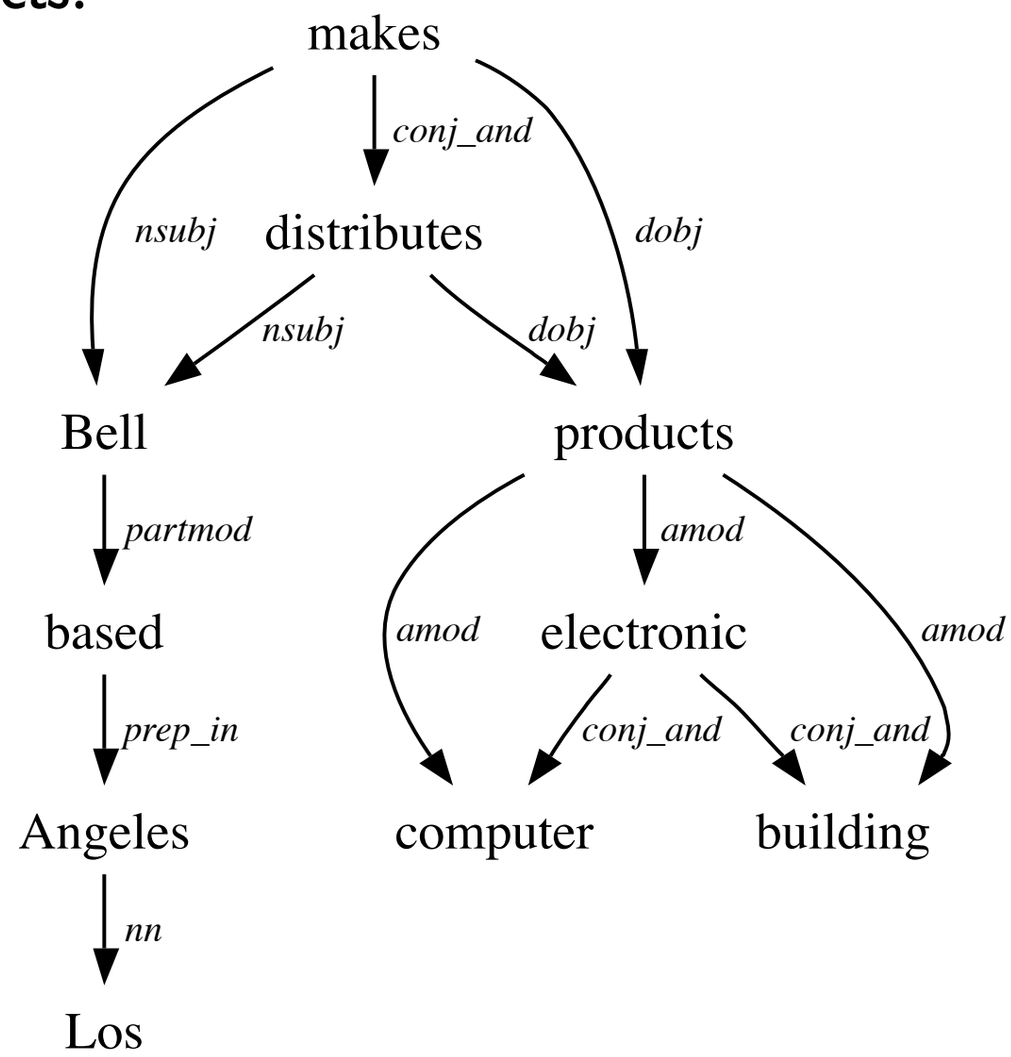
Constits -> Deps

- Two ways to parse to dependencies:
  - Run a constit parser, then run a (typically rule-based) constit->deps converter
  - Direct dependency parsing
- Dependencies useful for many applications
- Dependency annotations are available for more languages ... perhaps better suited for a wider variety of languages (e.g. free word order)

Dependency parse

- Edges between core words
- DAG (sometimes tree). Options to expand coordination, etc.

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

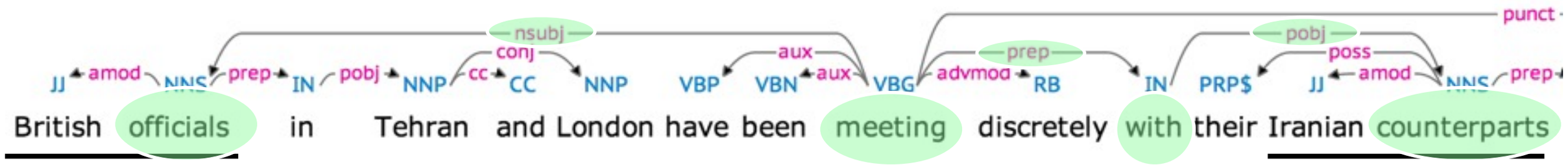


- X --relation--> Y graph edge
  e.g. nsubj(makes, Bell)
  - X: governor, head (parent...)
  - Y: dependent, modifier, subordinate (child...)

- Grammatical relations: see "Stanford Dependencies Manual"
  - *nsubj*: nominal subject
  - *dobj*: direct object
  - *prep_X*: prepositional argument
  - *amod*: adjective modifier
  - ...
- Using the graph: word-relation-word edges, paths, subgraphs...

Dependency paths

- Information extraction with long(er)-distance connections. Skip over modifiers and subclauses.



<--*nsubj*-- meet --*prep*--> with --*pobj*-->

"*X* meets with *Y*"

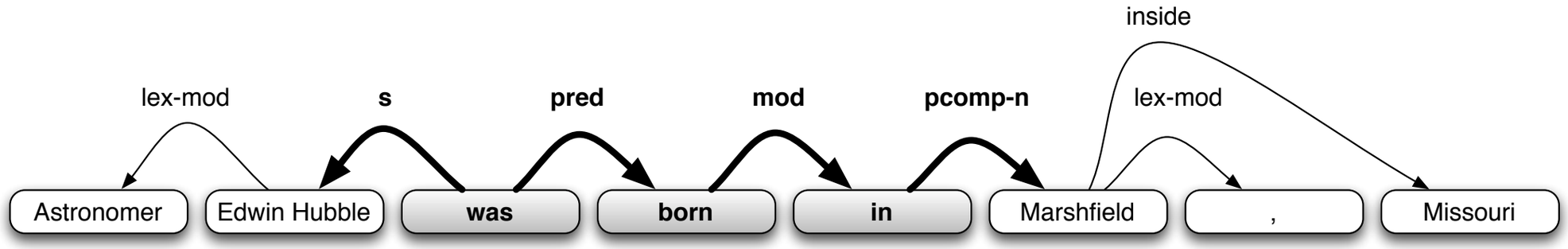
officials <--*nsubj*-- meet --*prep*--> with --*pobj*--> counterparts

British <--*amod*-- (NP) <--*nsubj*-- meet --*prep*--> with --*pobj*--> (NP) --*pobj*--> Iranian

Dependency paths

- Information extraction with long(er)-distance connections. Skip over modifiers and subclauses.

Task: get features which describe
the "X was born in Y" semantic relation

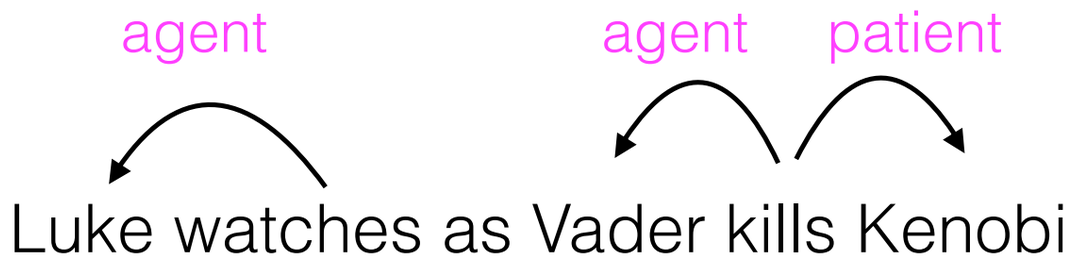


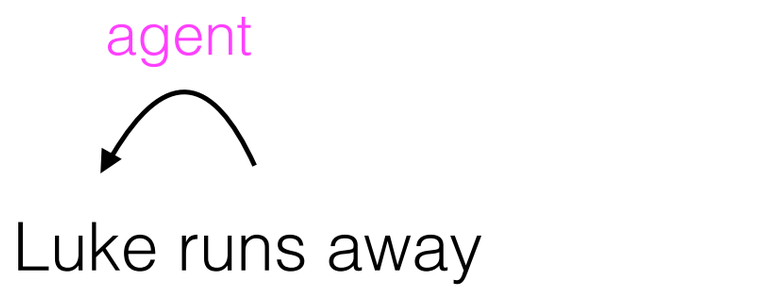
Feature type	Left window	NE1	Middle	NE2	Right window
Lexical	[]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[]
Lexical	[Astronomer]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[,]
Lexical	[#PAD#, Astronomer]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[, Missouri]
Syntactic	[]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[Edwin Hubble $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[Astronomer $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{lex-mod}$,]
Syntactic	[Edwin Hubble $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{lex-mod}$,]
Syntactic	[Astronomer $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{lex-mod}$,]
Syntactic	[]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{inside}$ Missouri]
Syntactic	[Edwin Hubble $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{inside}$ Missouri]
Syntactic	[Astronomer $\downarrow_{lex-mod}$]	PER	[$\uparrow_s$ was $\downarrow_{pred}$ born $\downarrow_{mod}$ in $\downarrow_{pcomp-n}$]	LOC	[$\downarrow_{inside}$ Missouri]

Table 3: Features for 'Astronomer Edwin Hubble was born in Marshfield, Missouri'.

Dependency paths

- Rule-based semantic relation extraction

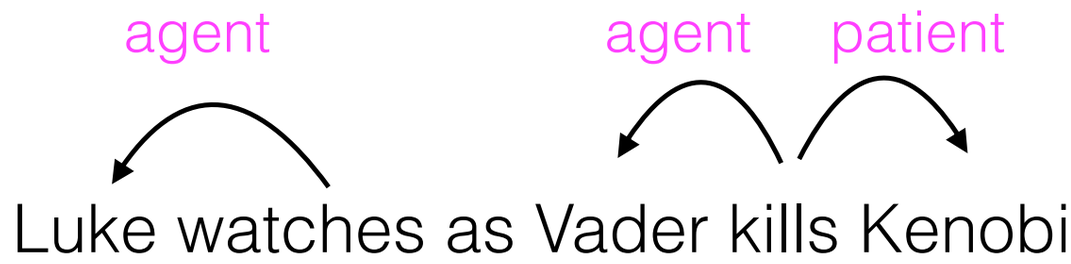


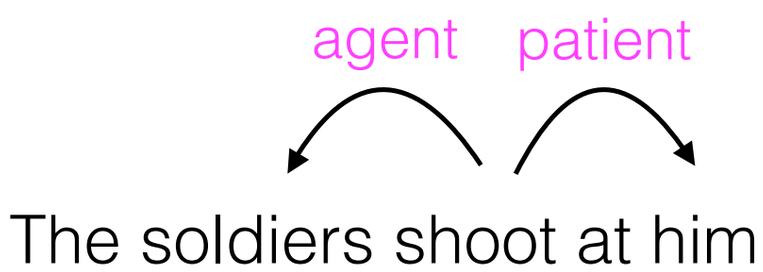


Dependency paths

- Rule-based semantic relation extraction

- e.g. assume a verb's subjects and objects denote arguments in an event

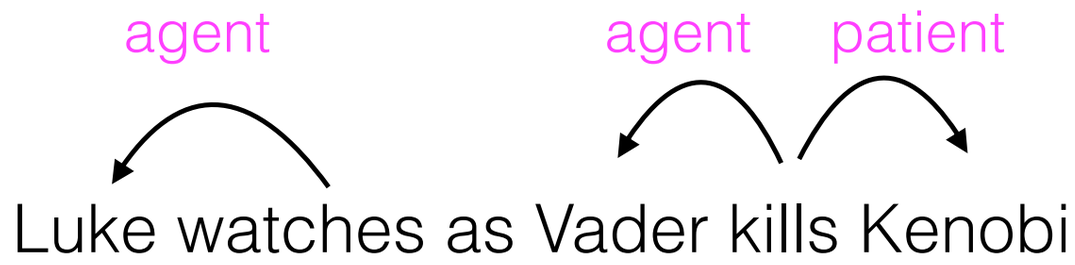




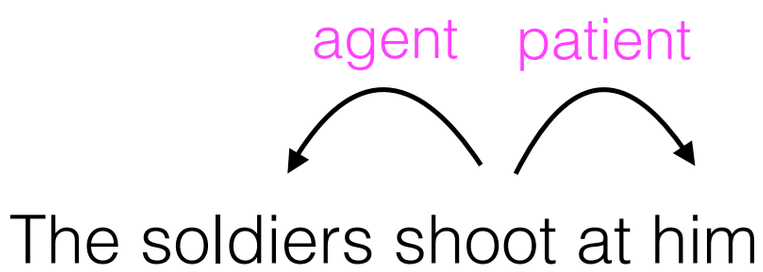
Dependency paths

- Rule-based semantic relation extraction

- e.g. assume a verb's subjects and objects denote arguments in an event



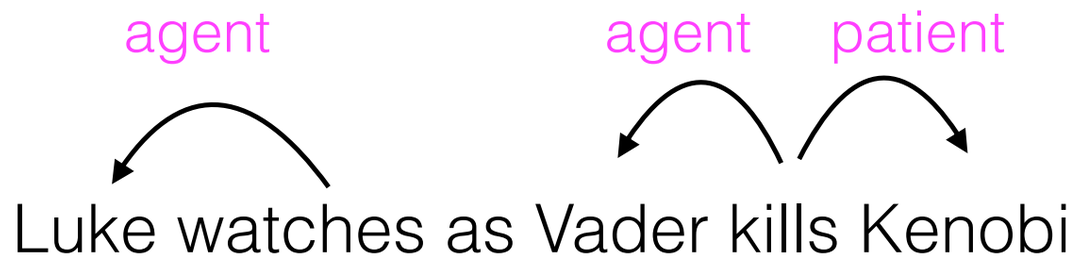
- But gets complicated (syntax-semantics interface)



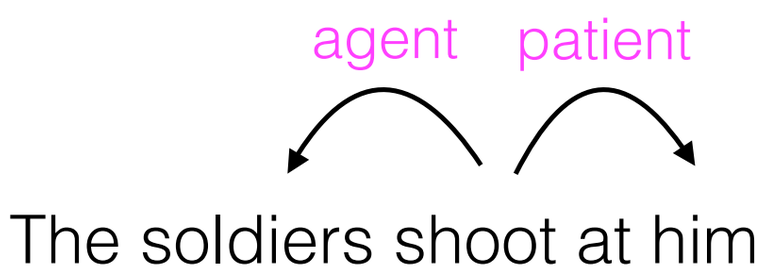
Dependency paths

- Rule-based semantic relation extraction

- e.g. assume a verb's subjects and objects denote arguments in an event



- But gets complicated (syntax-semantics interface)
  - "the Death Star's destruction"



Should you use a parser in your project?

- Dependency n-grams as features
  - e.g. dep bigrams (word, REL, word)
- Parsers performance and efficiency varies
  - "Shift-reduce" or "incremental" dependency parsers: tend to be fastest, currently
  - Performance: is your data similar to newswire text? (The usual training data)