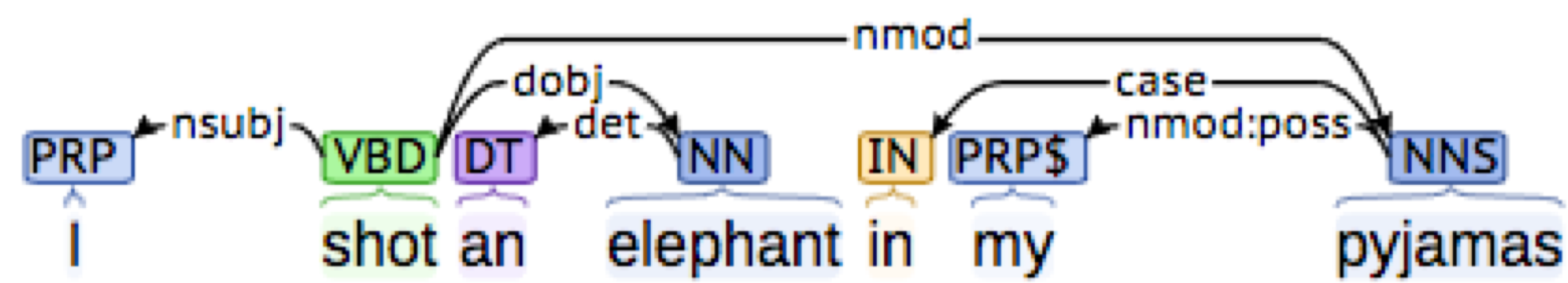


COMP90042 LECTURE 10

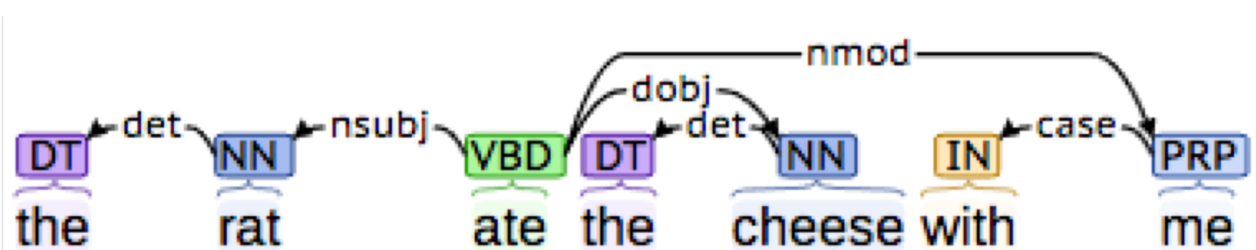
DEPENDENCY GRAMMAR & PARSING

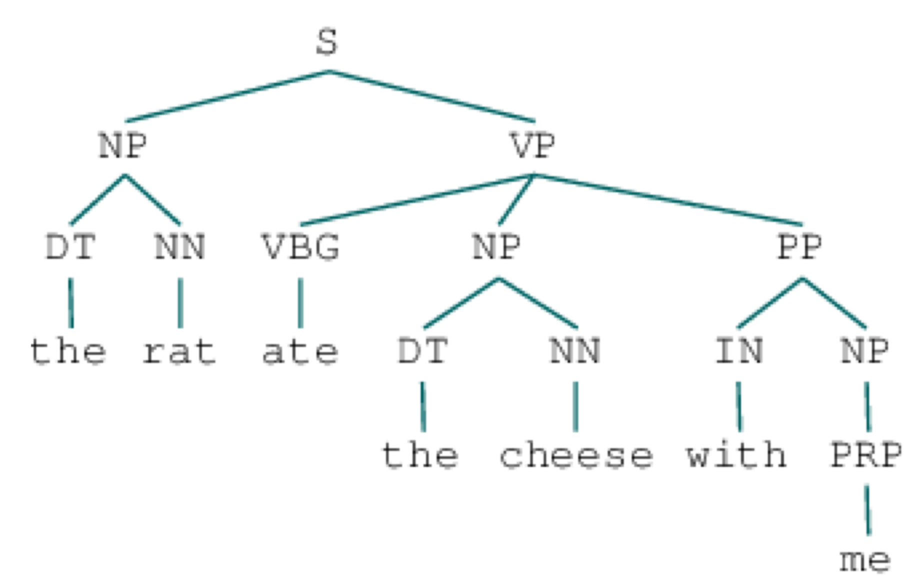
OUTLINE

- Dependency grammars
- Projectivity
- Parsing methods
  - transition-based parsing
  - graph-based

DEPENDENCY GRAMMAR (VS P.S.G.S)

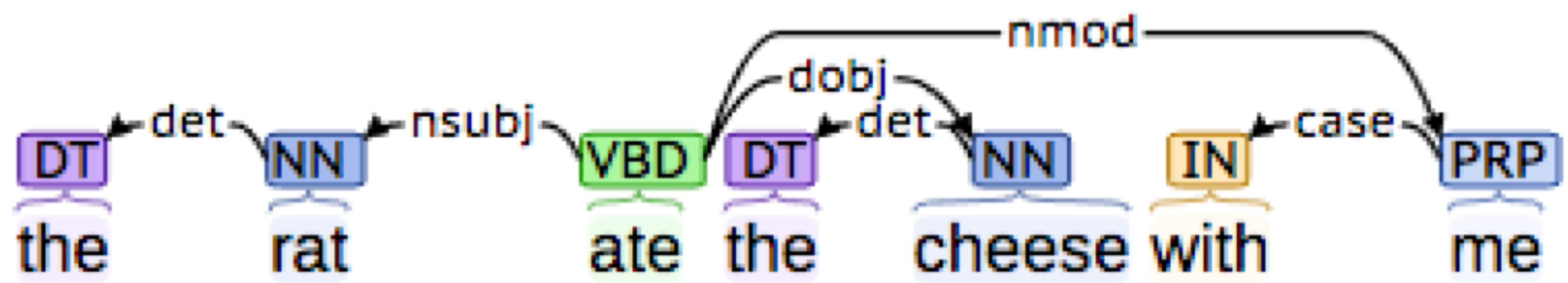
- *phrase-structure grammars* assume a *constituency tree* which identifies the *phrases* in a sentence
  - based on idea that these phrases are interchangable (e.g., swap an NP for another NP) and maintain grammaticality
- *dependency grammar* offers a simpler approach
  - describe binary relations between pairs of words
  - namely, between *heads* and *dependents*
- Building on notion of *head* as seen in phrase-structure parsers…





WHAT IS A DEPENDENCY?

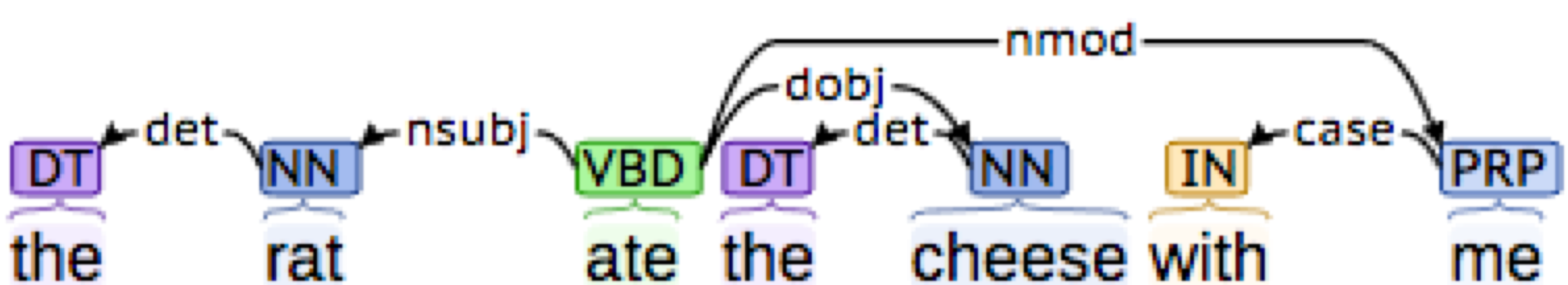
- Links between a *head* word and its *dependent* words in the sentence: either *syntactic roles* or *modifier relations*



- *argument* of a predicate, e.g., *ate(rat, cheese)*
  - *rat* is the *subject* of verb *ate* (thing doing the eating)
  - *cheese* is the *direct object* of verb ate (thing being eaten)
- head may determine *type* of relation, lexical form of dependent etc
  - verb-subject agreement, *I talk to myself*, vs **me talk to I*
  - agreement often for gender, number and case

WHAT IS A DEPENDENCY II

- Various other types of dependencies exist
  - a *modifier* which is typically optional (aka *adjunct*)
    - (with) *me* modifies the act of (the rat) *eating*
  - *specifiers*, e.g., *the rat, the cheese, with me*
    - help to specify the referent (which rat?), the head's relation, etc.

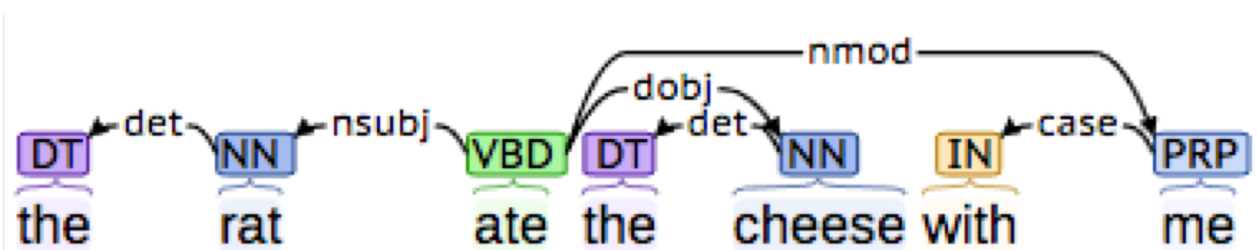


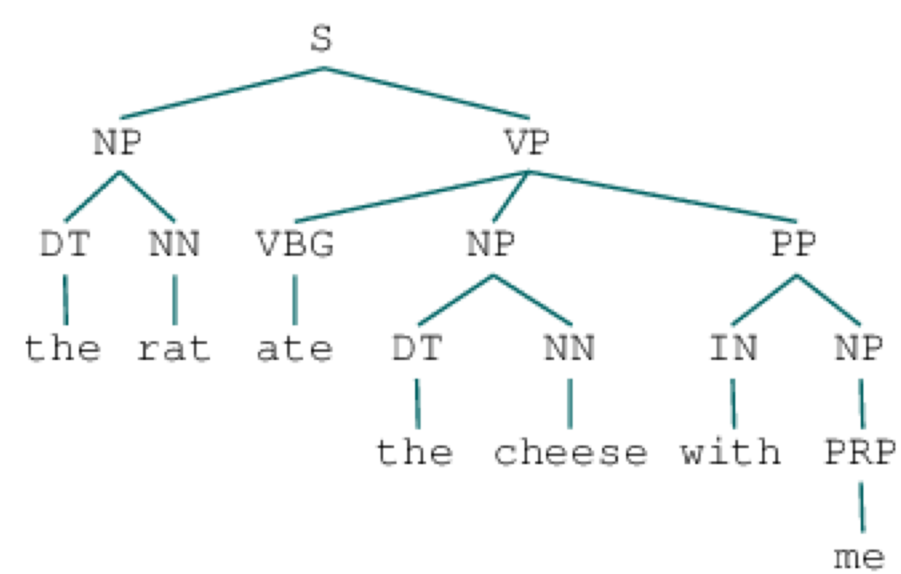
DEPENDENCY TYPES

- Edges labelled with the dependency *type*, *e.g.*, *Stanford types*, *e.g.*, sample types (key: *head*, **dependent**)
  - NSUBJ (nominal subject) **Julian** *speaks* Chinese
  - DOBJ (direct object) Trevor *presented* a **lecture** in English
  - IOBJ (indirect object) Morpheus *gave* **Neo** the red pill
  - APPOS (appositive) *Neo*, the main **character**, swallowed the pill
- See reading for more!

WHY DEPENDENCIES?

- Dependency tree more directly represents the core of the sentence: *who did what to whom?*
  - captured by the links incident on verb nodes, e.g., NSUBJ, DOBJ etc; easier to answer questions like:
    - what was the main thing being expressed in the sentence (*eat*)

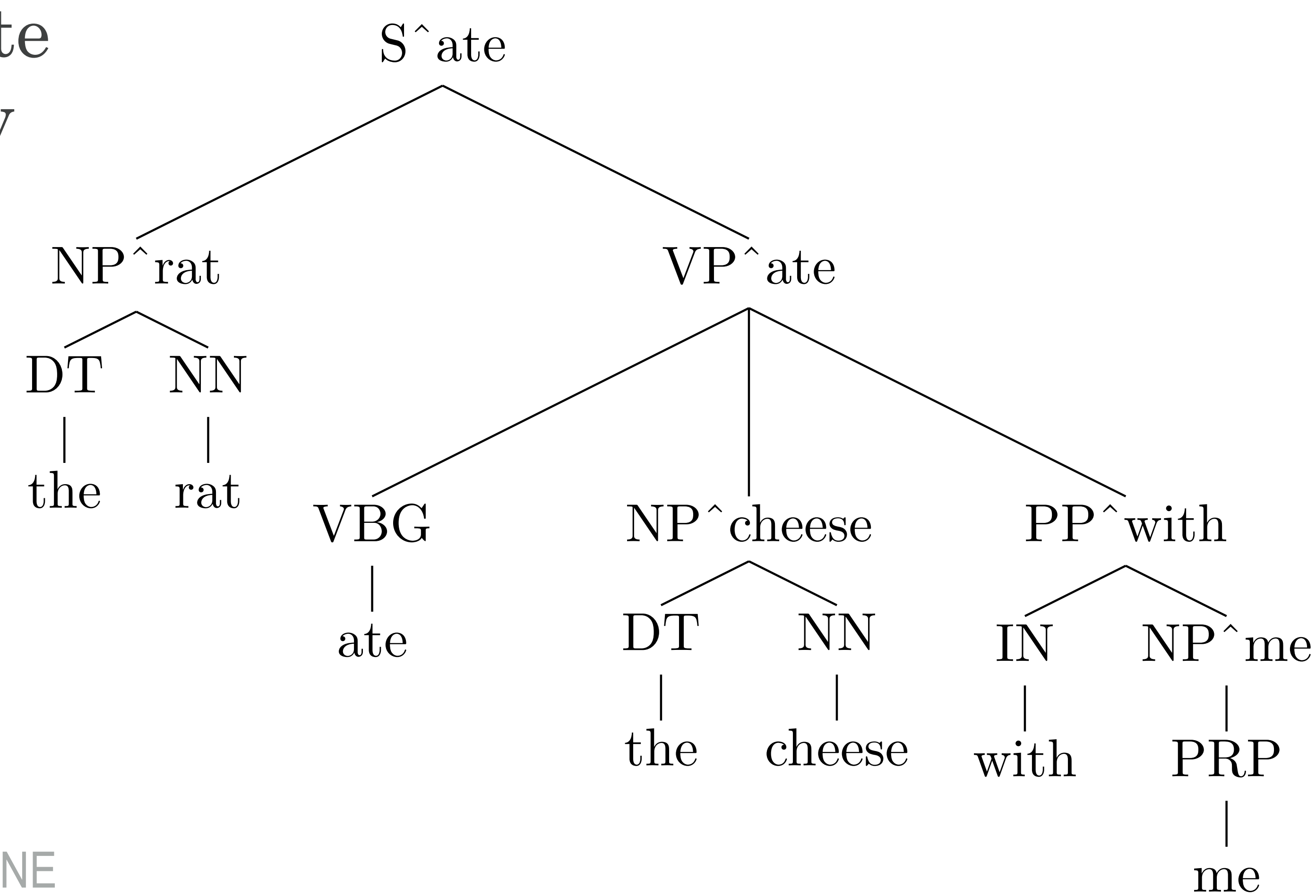




  - more minor details are buried deeper in the tree (e.g., adjectives, determiners etc)

DEPENDENCY VS HEAD

- Close similarity with ‘head’ in phrase-structure grammars
  - the ‘head’ of an XP is (mostly) an X,
    i.e., noun in a NP, verb in a VP etc.
    see https://en.wikipedia.org/wiki/Head_(linguistics)
  - main dependency edges captured in rewrite rules
    - S^ate -> NP^rat VP^ate
      captures dependency
      rat ← ate



DEPENDENCY *TREE*

- Dependency edges form a *tree*
  - each node is a *word token*
  - one node is chosen as the *root*
  - directed edges link heads and their dependents
- Cf. phrase-structure grammars
  - forms a hierarchical tree
  - word tokens are the *leaves*
  - internal nodes are 'constituent phrases' e.g., NP, VP etc
- Both use part-of-speech

(NON-)PROJECTIVITY

- A tree is *projective* if, for all arcs from head to dependent
  - there is a path from the head to every word that lies between the head and the dependent
- More simply, the tree can be drawn on a plane without any arcs crossing
- Most sentences are projective, however exceptions exist (fairly common in other languages)

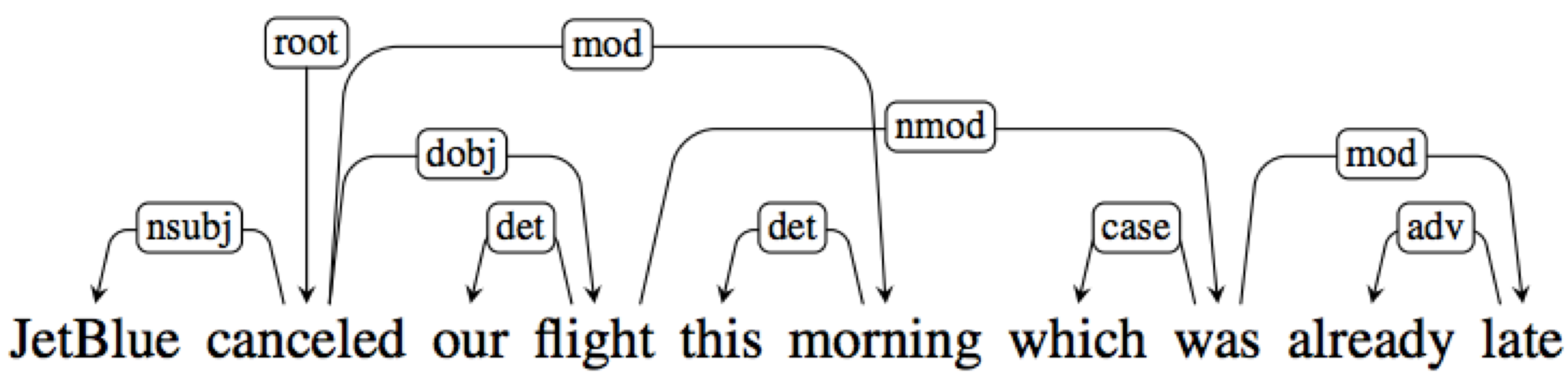


Figure JM3, Ch 14

DEPENDENCY GRAMMAR

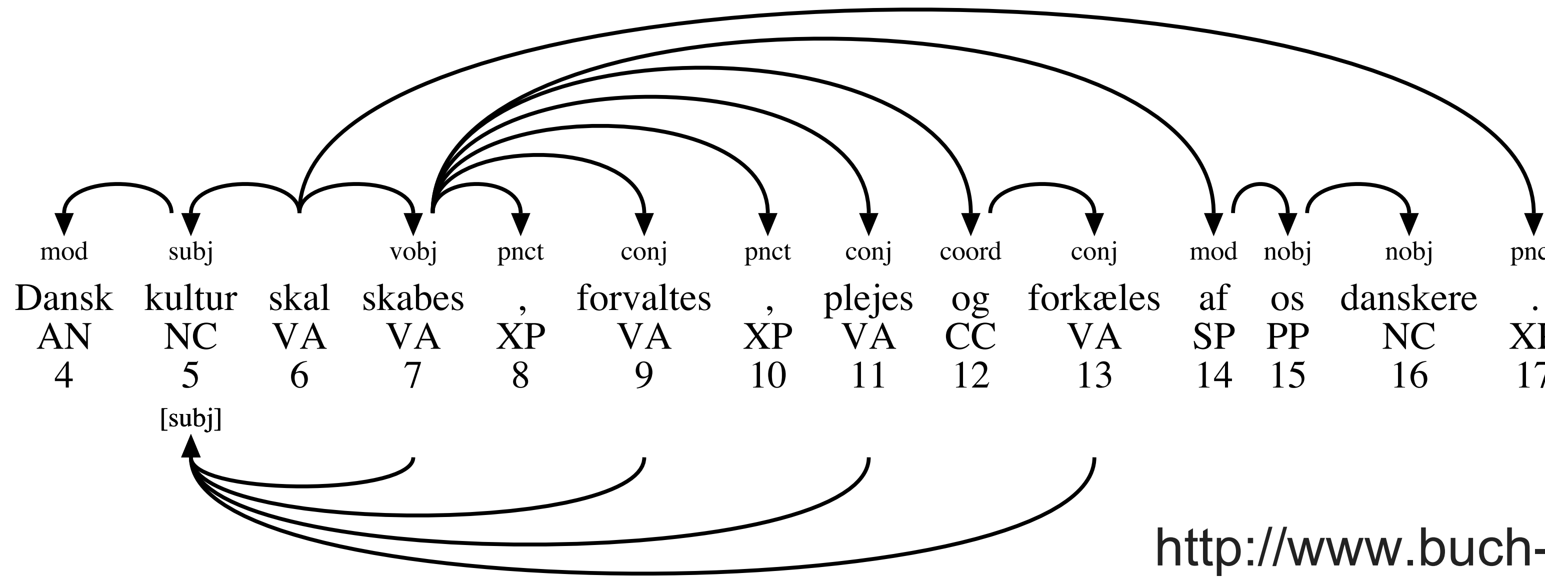
- Not really a grammar, in sense of a '*generative grammar*'
  - cannot be said to define a language, unlike a context free grammar
  - any structure is valid, job of *probabilistic model* to differentiate between poor and good alternatives
- However, very practical and closely matches what we want from a parser (most often predicates & arguments)

DEPENDENCY TREEBANKS

- A few dependency treebanks
  - Czech, Arabic, Danish, Dutch, Greek, Turkish …
- Many more phrase-structure treebanks, which can be *converted* into dependencies
- More recently, *Universal Dependency Treebank*
  - collates 70 treebanks, 50 languages
  - unified part-of-speech, morphology labels, relation types
  - consistent handling of conjunctions and other tricky cases
- http://universaldependencies.org/

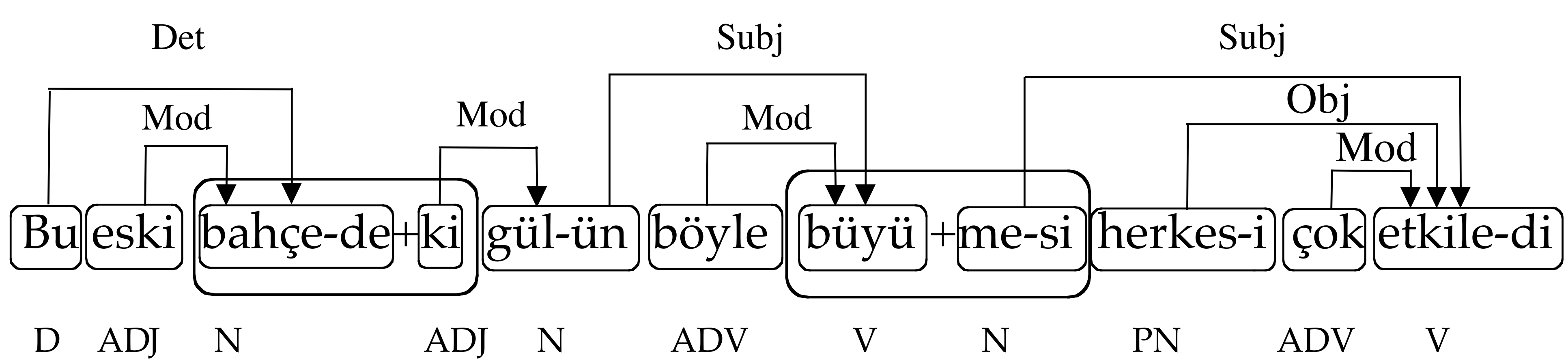
EXAMPLES FROM TREEBANKS

- Danish DDT includes additional ‘subject’ link for verbs



http://www.buch-kromann.dk/matthias/ddt1.0/

- METU-Sabancı Turkish treebank
  - edges between morphological units, not just words (-,+)



Oflazer, Kemal, et al. "Building a Turkish treebank." *Treebanks*. Springer, 2003. 261-277.

DEPENDENCY PARSING

- Parsing: task of finding the *best* structure for a given input sentence
  - i.e., $\arg\max_t \text{score}(\boldsymbol{t} \mid \boldsymbol{x})$
- Two main approaches:
  - *graph-based*: uses *chart* over possible parses, and dynamic programming to solve for the maximum
  - *transition-based*: treats problem as incremental sequence of decisions over next action in a state machine

TRANSITION BASED PARSING

- Frames parsing as sequence of simple parsing transitions
  - maintain two data structures
    - *buffer* = input words yet to be processed
    - *stack* = head words currently being processed
  - two types of transitions
    - *shift* = move word from buffer on to top of stack
    - *arc* = add arc (left/right) between top two items on stack (and *remove* dependent from stack)

TRANSITION BASED PARSING ALGORITHM

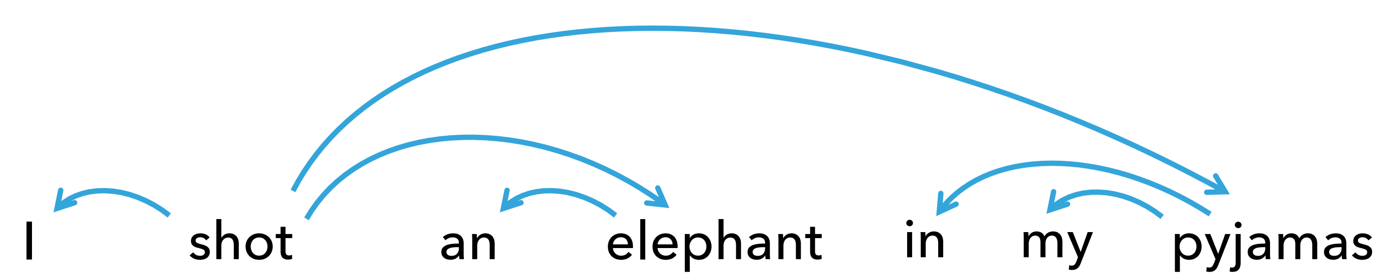
- For each word in input (buffer)
  - *shift* current word from buffer onto stack
  - while there are 2 or more items on stack:
    - either:
      - a) add an *arc (left or right)* between top two items, and remove the dependent; or
      - b) continue to outer loop
- Finished when buffer empty & stack has only 1 item
- Always results in a *projective tree*

EXAMPLE

- I shot an elephant in my pyjamas

Buffer	Stack	Action
I shot an elephant in my pyjamas		Shift

Generated parse:

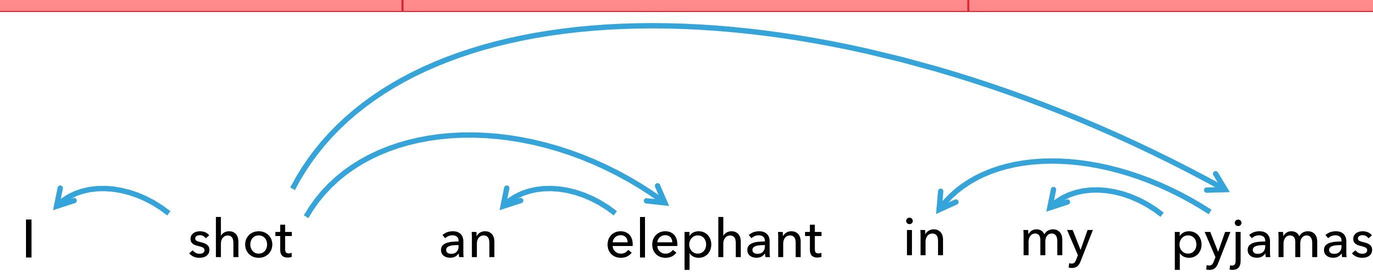


EXAMPLE

- I shot an elephant in my pyjamas

Buffer	Stack	Action
I shot an elephant in my pyjamas		Shift
shot an elephant in my pyjamas	I	Shift
an elephant in my pyjamas	I, shot	Arc-left
an elephant in my pyjamas	shot	Shift
elephant in my pyjamas	shot, an	Shift
in my pyjamas	shot, an, elephant	Arc-left
in my pyjamas	shot, elephant	Arc-right
in my pyjamas	shot	Shift
...	...	...
	shot	<done>

Generated parse:



TRANSITION BASED PARSING MODELS

- How do we know when to *arc* and whether to add *left* or *right* facing arcs?
- Use a scoring function, *score(buffer, stack, transition)*, based on the state, i.e.,
  - the next word(s) in the buffer
  - the contents of the stack, particularly the top two items
  - the transition type, one of *{continue, arc-left, arc-right}*
- Then select the *transition* with the highest score!

TRANSITION BASED SCORING

- Form a feature representation for the state
  - e.g., stack top has tag NN & next in stack has tag DT & transition = arc-left
  - learn a *weight* for each feature of this type, in order that the parser predicts the correct next action
- E.g., *percepton training (Goldberg & Nivre, COLING 2012)*

Algorithm 2 Online training with a static oracle

$\mathbf{w} \leftarrow 0$
for $I = 1 \rightarrow$ ITERATIONS **do**
  for sentence x with gold tree G_{gold} in corpus **do**
    $c \leftarrow c_s(x)$
    while c is not terminal **do**
      $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$
      $t_o \leftarrow o(c, G_{\text{gold}})$
      if $t_p \neq t_o$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
      $c \leftarrow t_o(c)$
return $\mathbf{w}$

Fig from Goldberg & Nivre (2012)

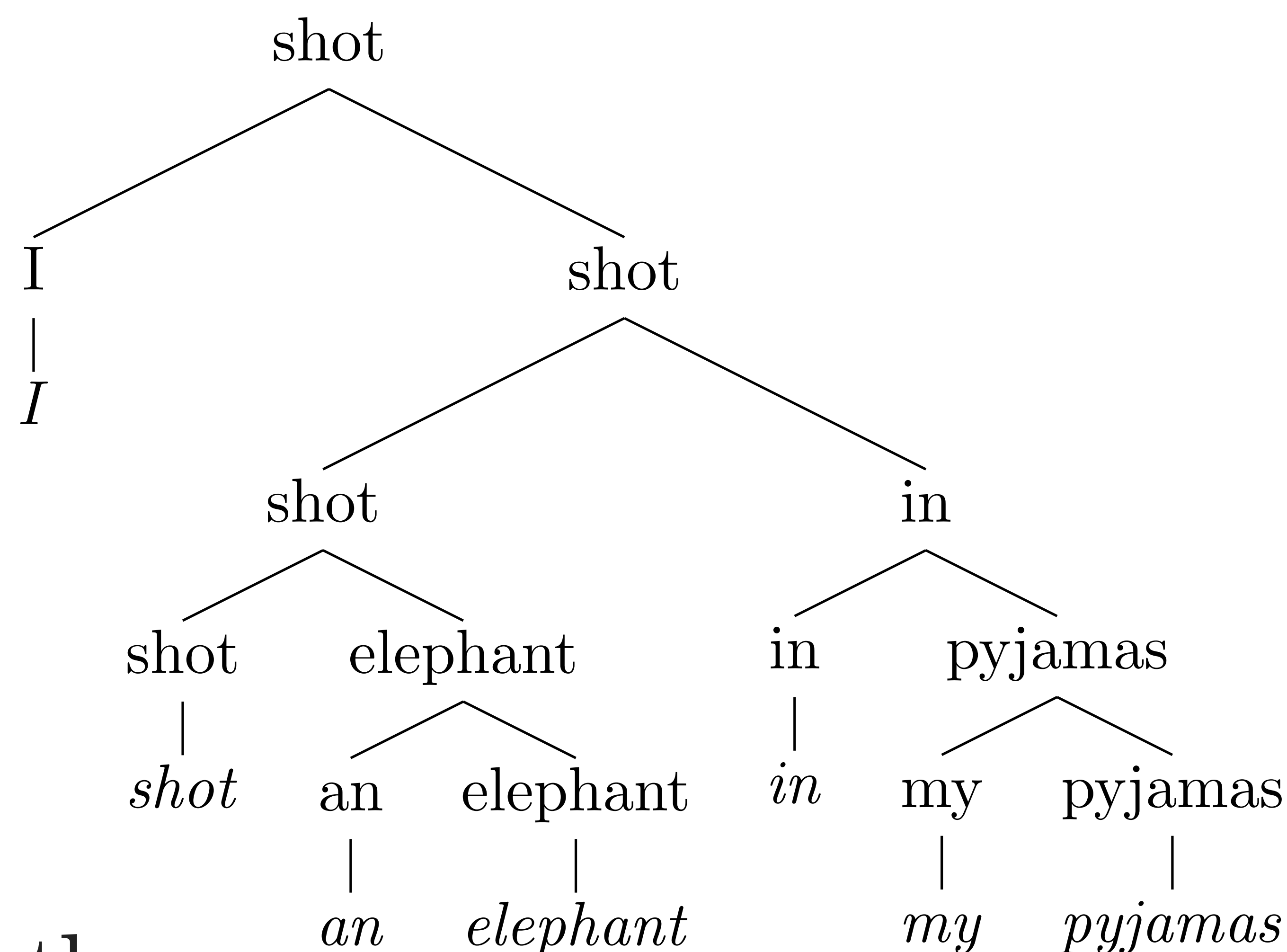
GRAPH BASED PARSING

- Dependency parsing using dynamic programming…
  - Can consider as a CFG, where lexical items (heads) are non-terminals

E.g., production

shot → shot in

means arc-right from "shot" to "in"



- Can be done more efficiently
  - E.g., Chu-Liu-Edmonds algorithms

A FINAL WORD

- Dependency parsing a compelling, alterative, formulation to constituency parsing
  - structures based on words as internal nodes
  - edges encode word-word syntactic and semantic relations
  - often this is the information we need for other tasks!
- Transition-based parsing algorithm
  - as sequence of shift and arc actions
- Graph-based parsing
  - uses classic dynamic programming methods (similar to CYK)

REQUIRED READING

- J&M3 Ch. 14