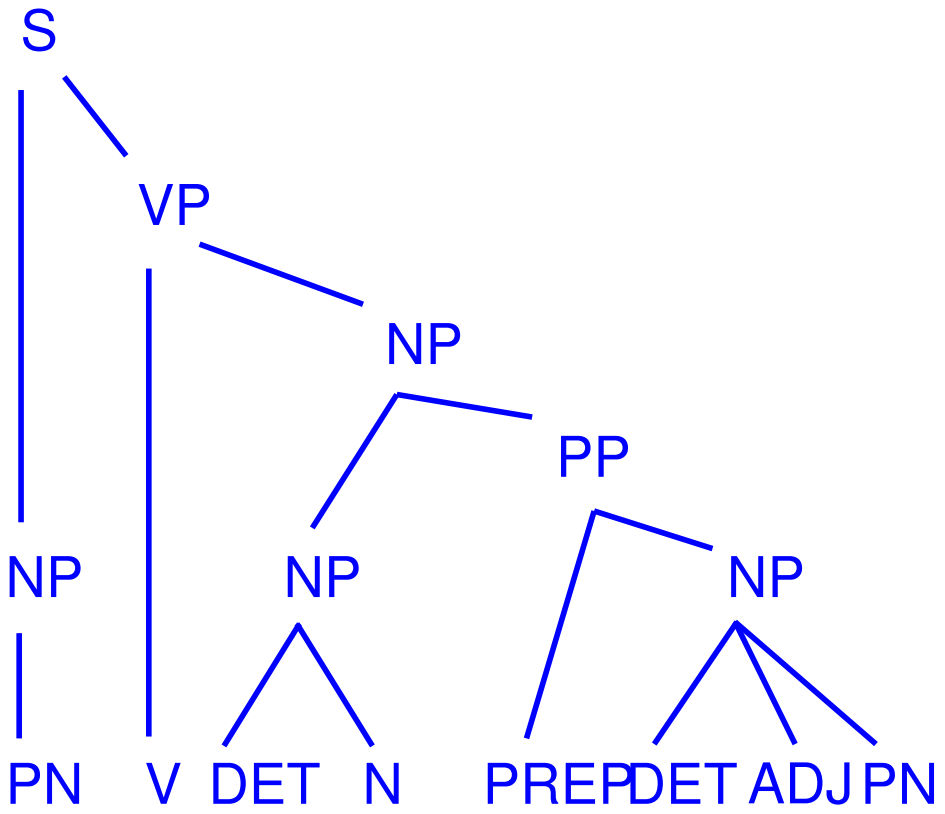
Dependency Parsing

Fabian M. Suchanek

Grammars help with structure



S -> NP VP

NP -> N

VP -> V

NP -> DET ADJ N

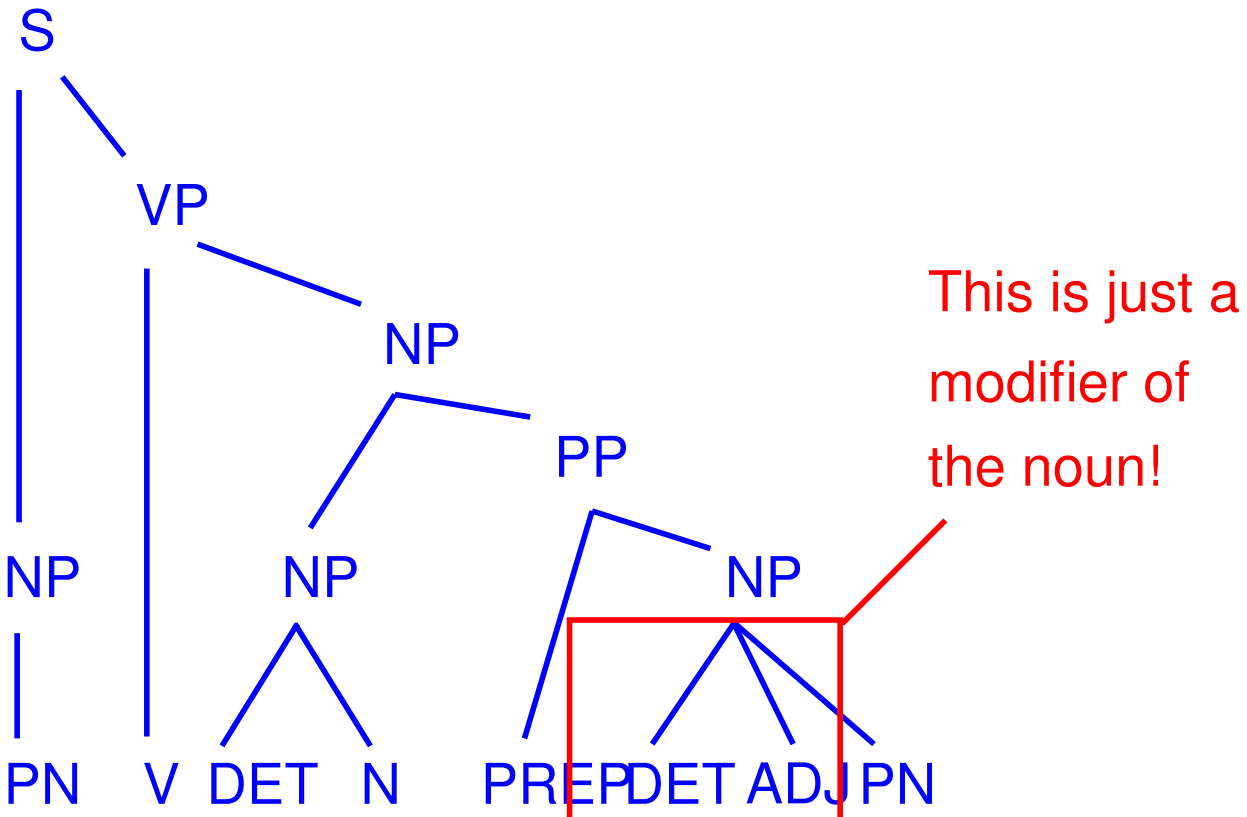
see Wikipedia/Phrase Structure Grammar

try it out

Alizée is the mother of the little Annily.

"X is the mother of Y"

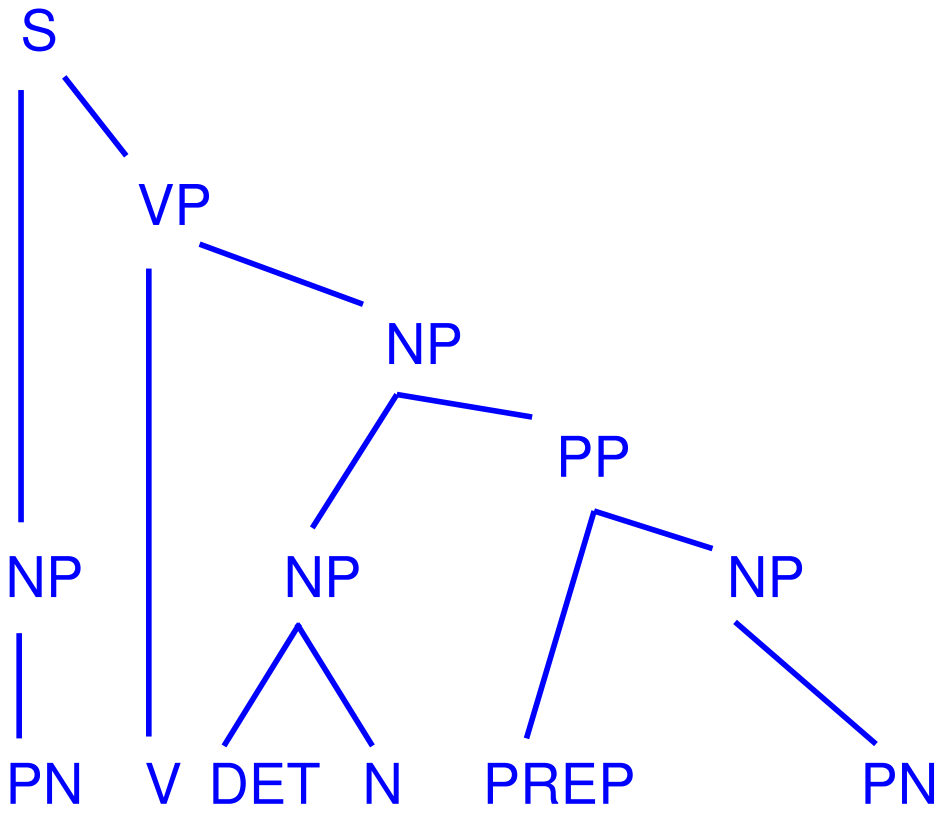
Grammars help with structure



Alizée is the mother of the little Annly.

"X is the mother of Y"

Grammars help with structure

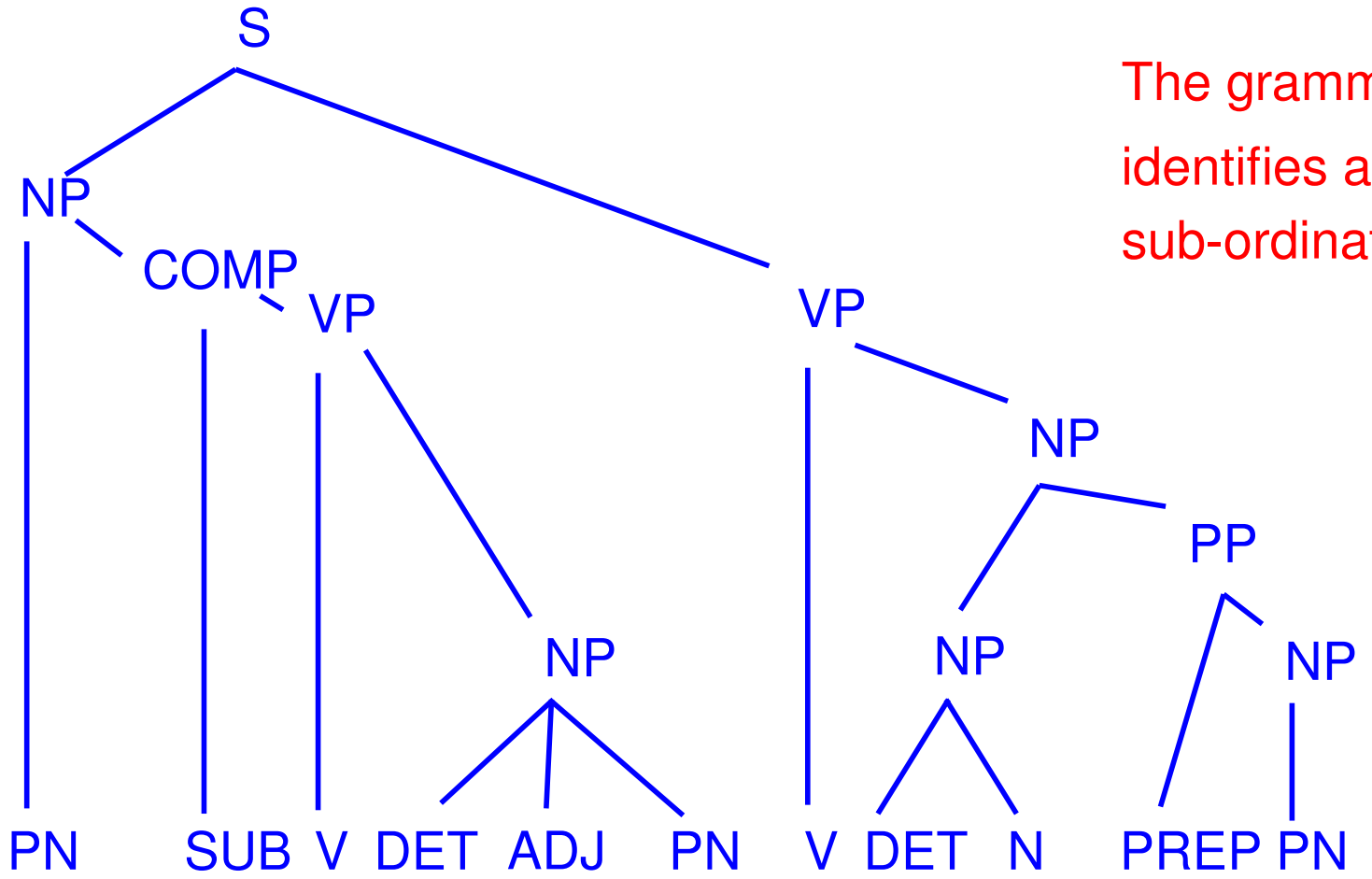


If we remove the modifiers, we match the pattern.

Alizée is the mother of th ıily.

"X is the mother of Y"

Grammars help with structure

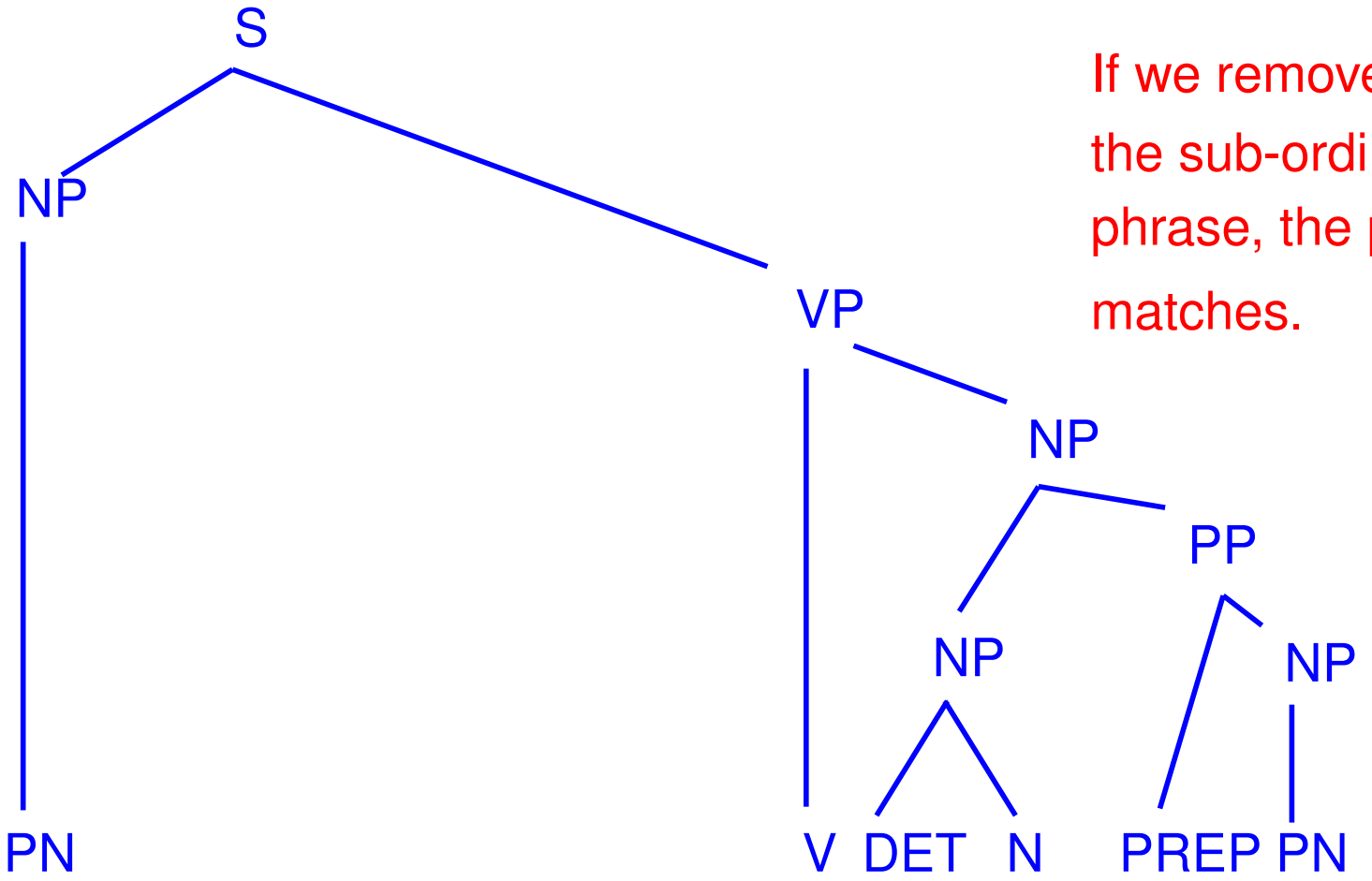


The grammar identifies a sub-ordinate phrase.

Alizée, who has a beautiful voice, is the mother of Annily.

"X is the mother of Y"

Grammars help with structure



If we remove the sub-ordinate phrase, the pattern matches.

Alizée,

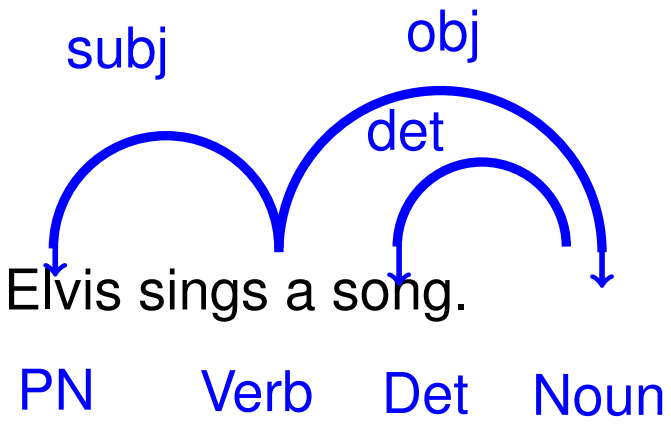
the mother of Annily.

"X

is the mother of Y"

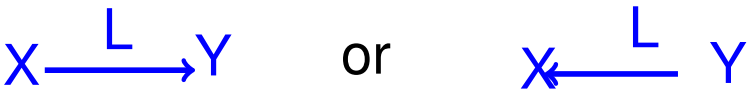
Dependency grammars

In IE, one often uses dependency grammars.



Def: Dependency grammar

A dependency grammar is a set of rules of the form

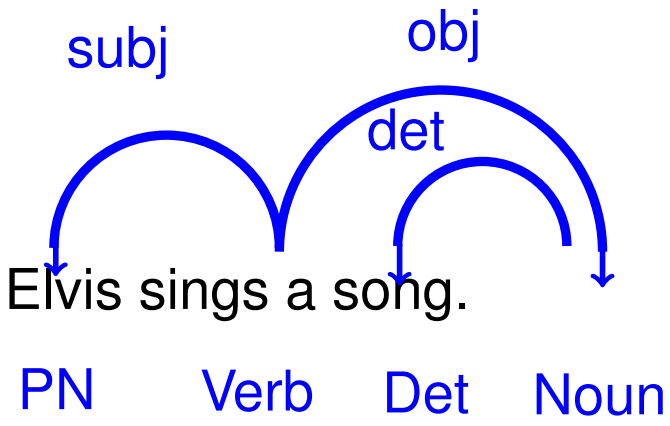


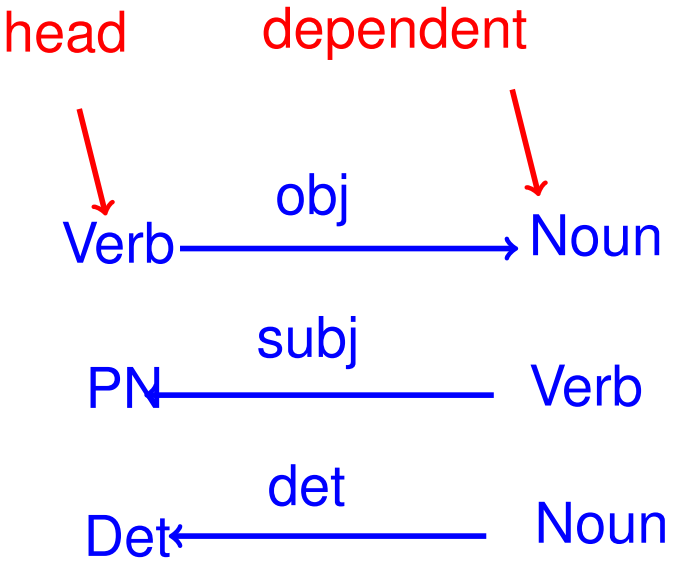
where X and Y are POS tags, L is a label.

(There are variants, where X and Y are words, or rules have more edges or no label)

(Dependency grammars and constituency grammars are strongly equivalent, as shown by H. Gaifman in 1965)

Joakim Nivre

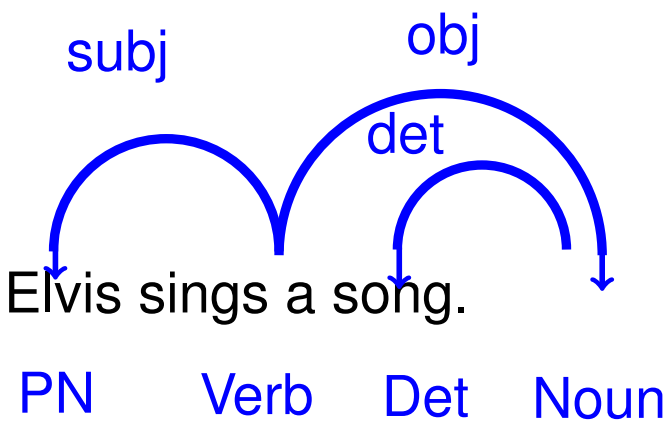


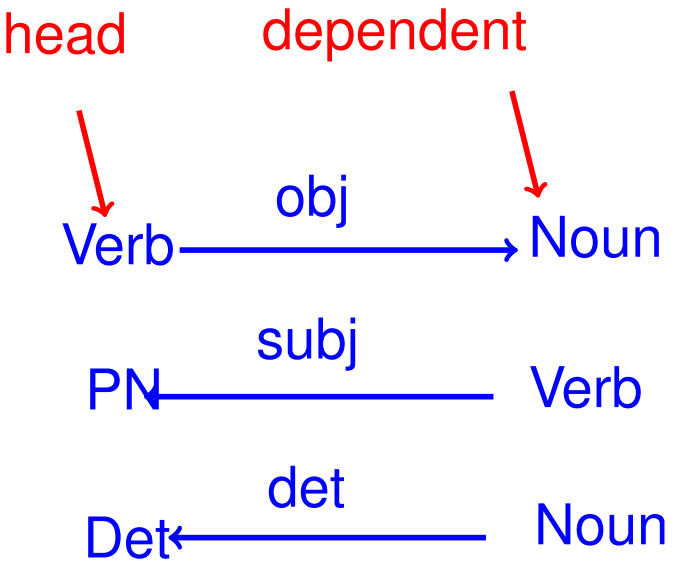


Dependency graph

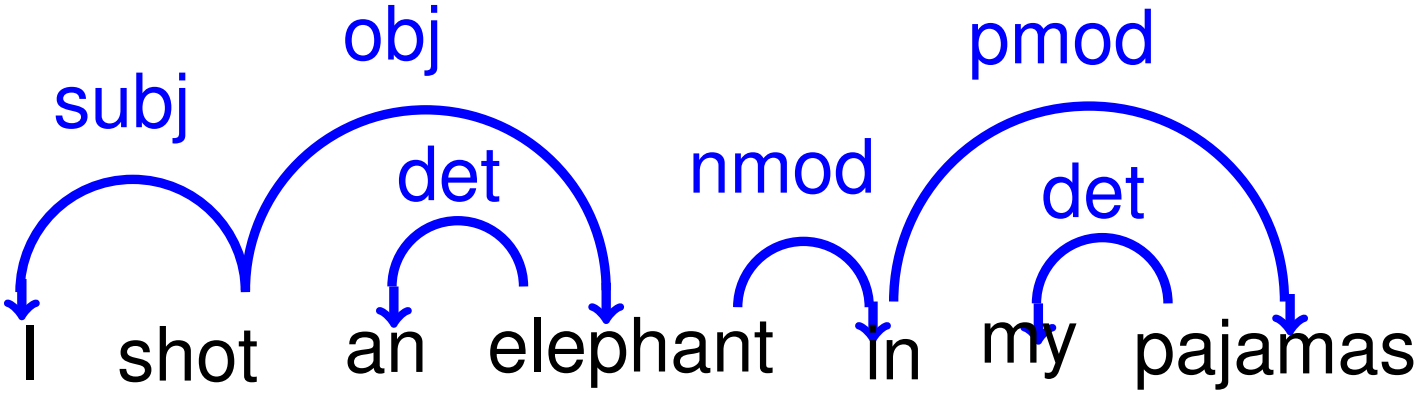
A dependency graph of a sentence is a connected acyclic graph whose nodes are the words and whose edges are given by the grammar, such that

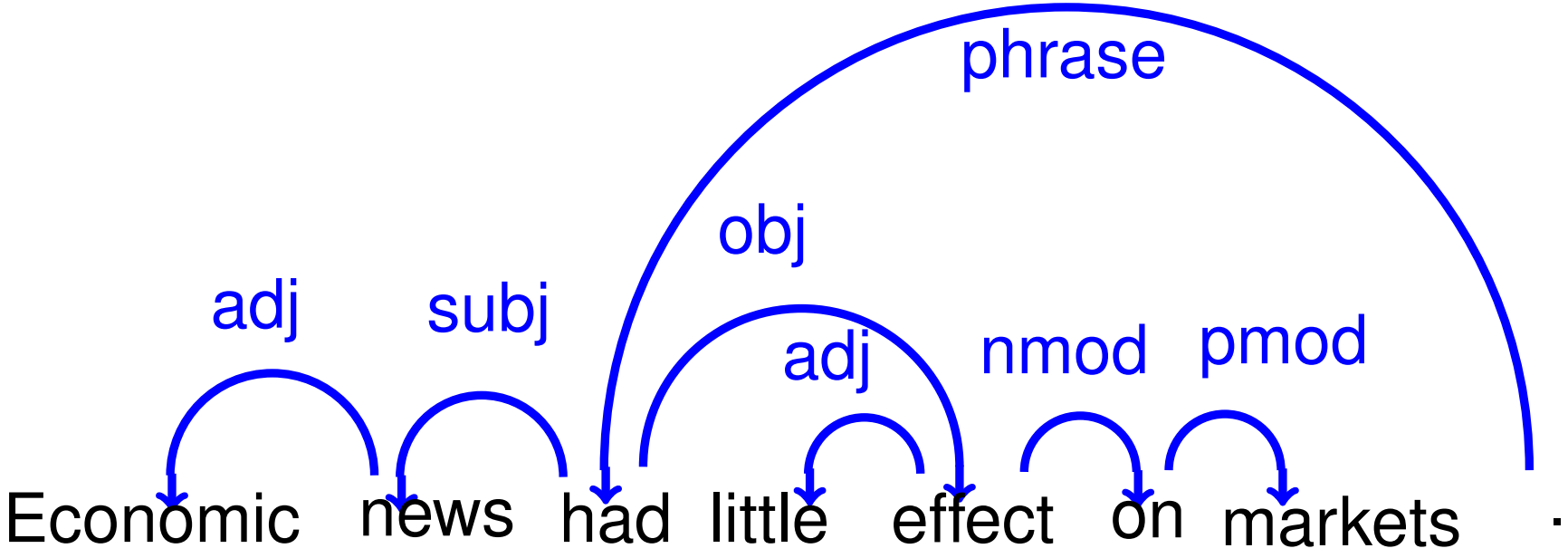
- Every word has at most one head
- There is a verb without a head





Examples: Dependency graph

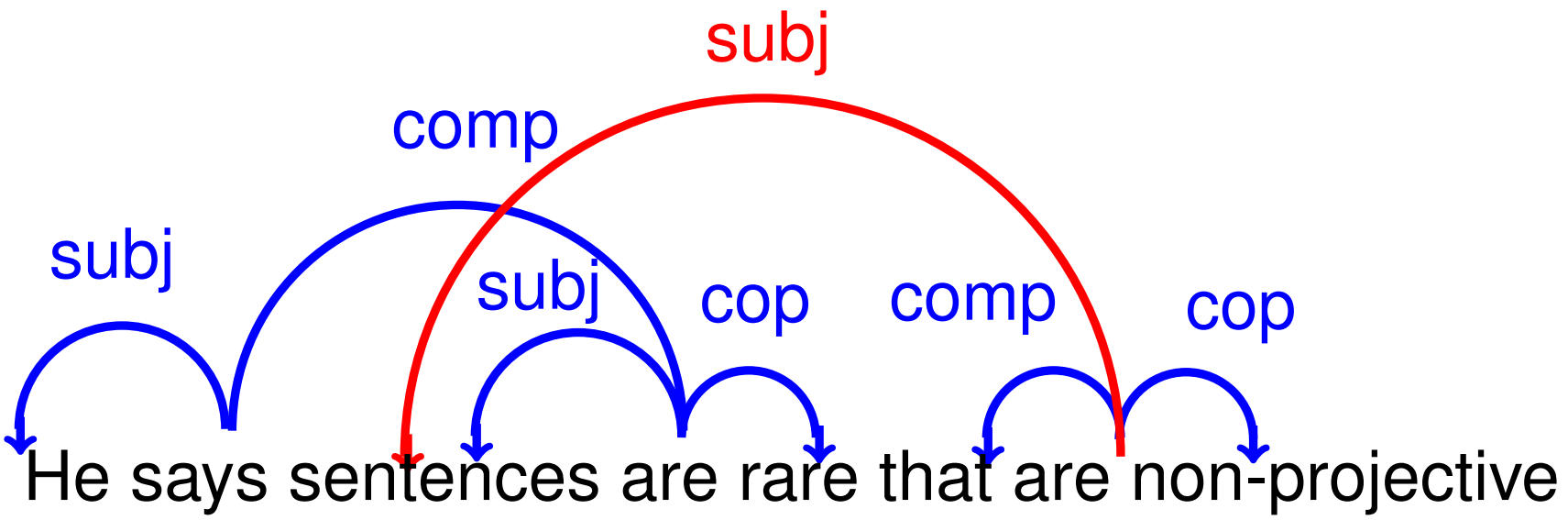




Try it out!

Projection

A dependency graph is projective if no arcs cross.

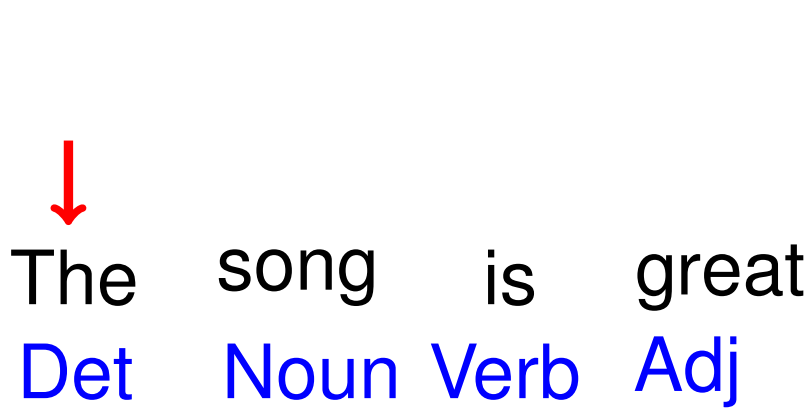


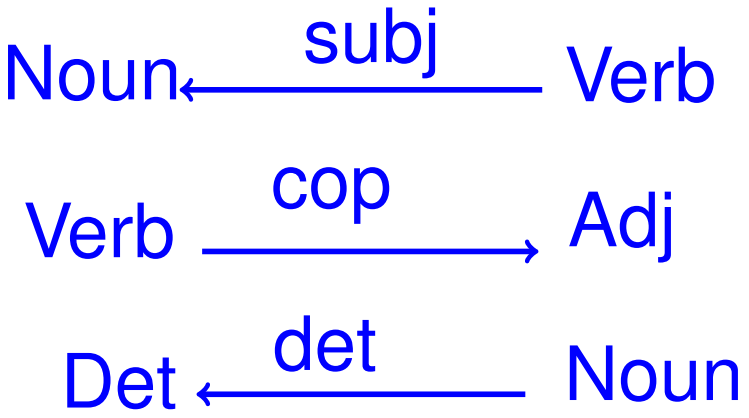
We consider only projective dependency graphs here.

convington>98

Convington's Algorithm

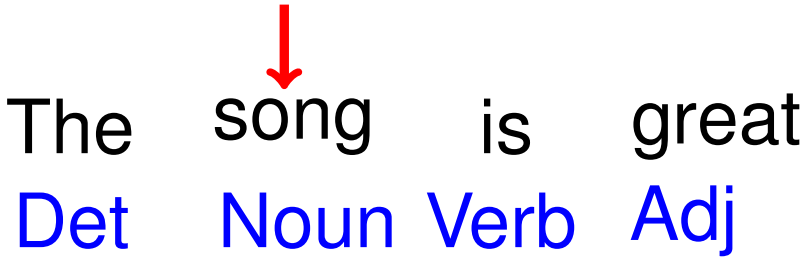
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

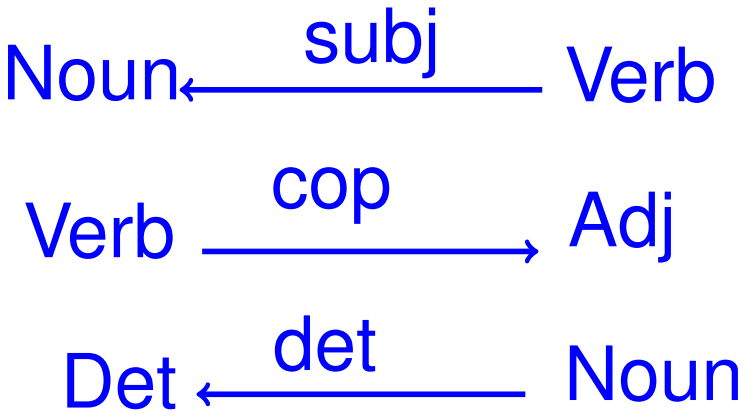




Convington's Algorithm

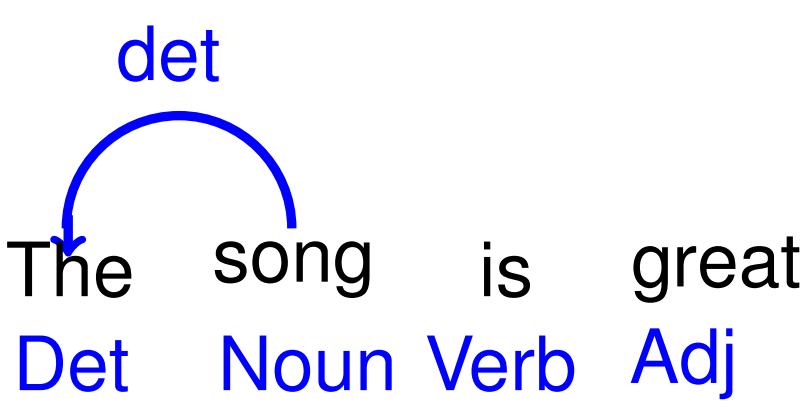
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

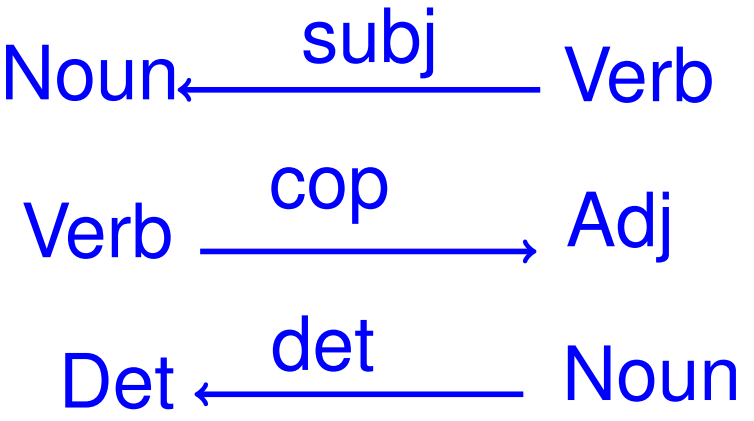




Convington's Algorithm

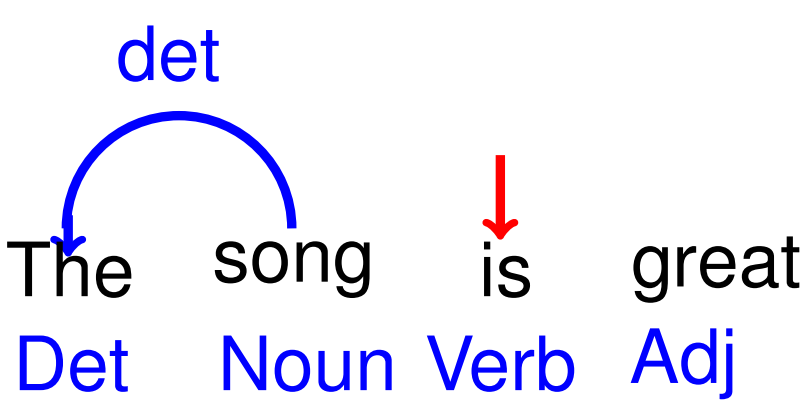
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

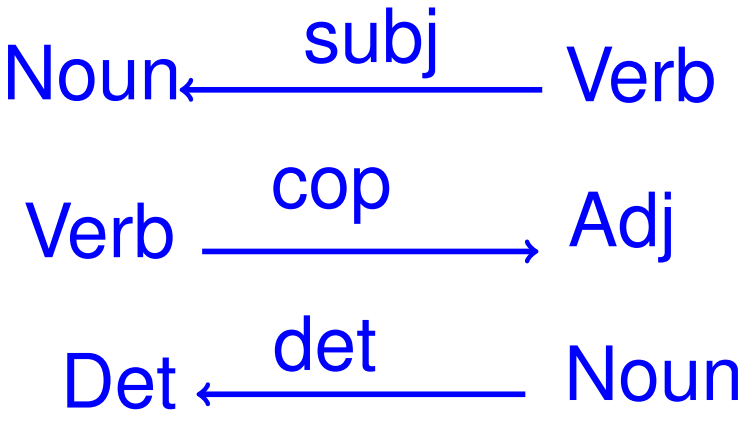




Convington's Algorithm

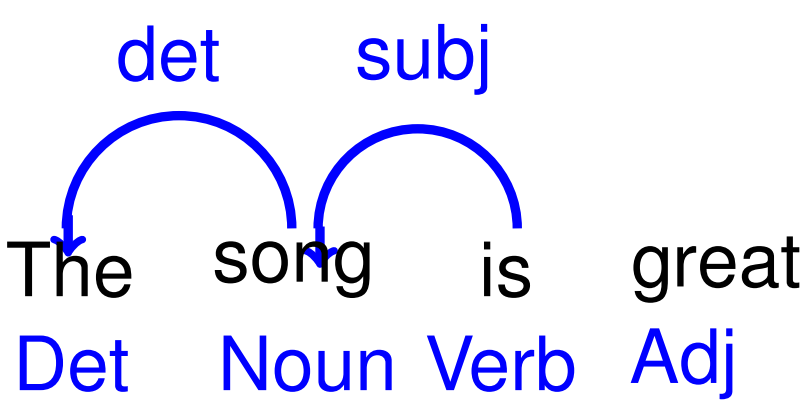
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

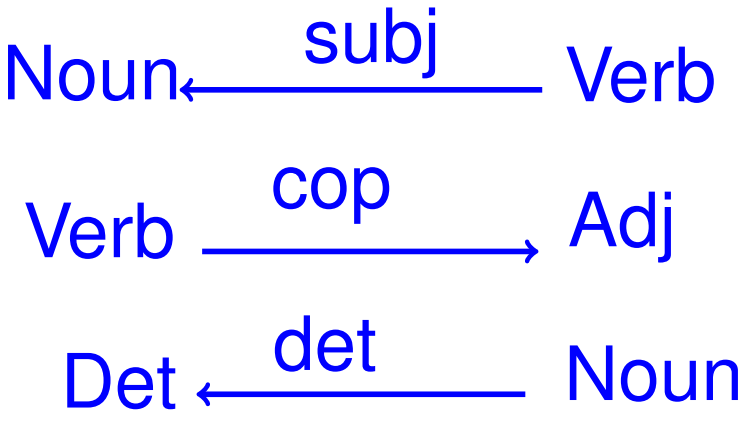




Convington's Algorithm

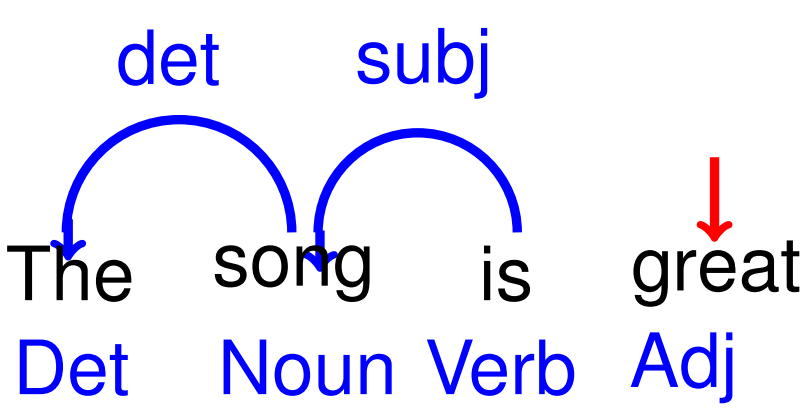
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

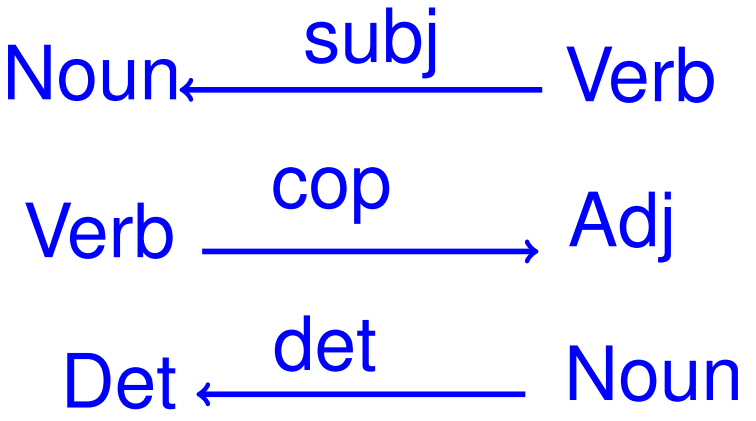




Convington's Algorithm

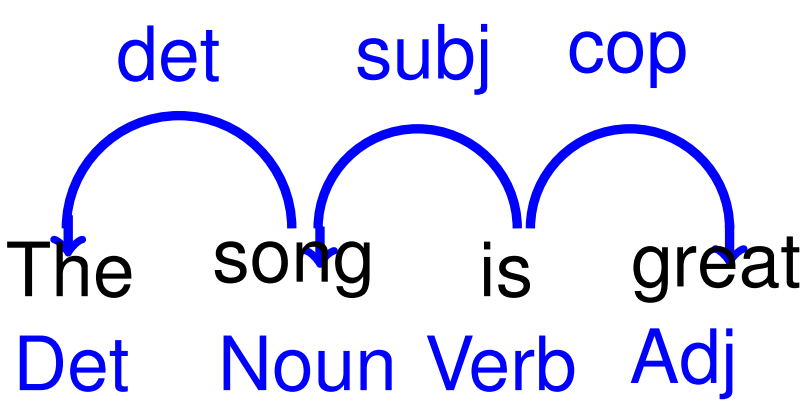
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.

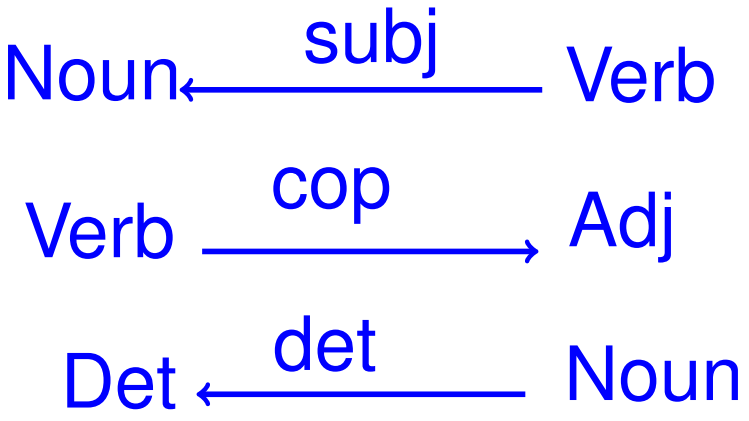




Convington's Algorithm

- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by the grammar, going backwards in the sentence, allowing only one head per word.





Task: Convington's Algorithm

Apply Convington's algorithm to

Not every idea by Coyote was brilliant.

NEG DET N PREPPN V ADJ

N —prep→ PREP

PREP —pobj→ PN

N ←subj— V

V —cop→ ADJ

DET ←det— N

NEG ←neg— N

Runtime of Convington's Algorithm

Not every idea by Coyote was brilliant.

NEG DET N PREPPN V ADJ

- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by grammar

Runtime of Convington's Algorithm

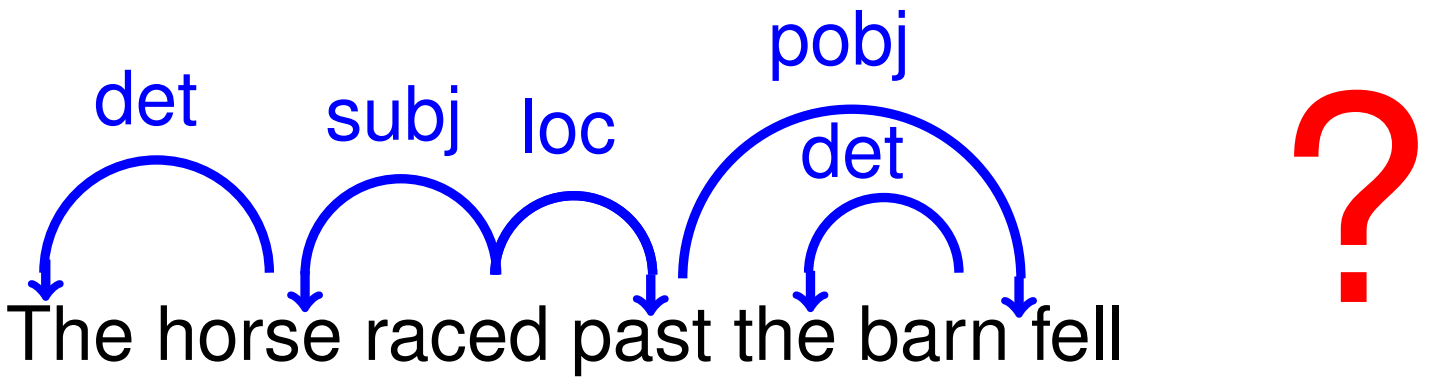
Not every idea by Coyote was brilliant.

NEG DET N PREPPN V ADJ

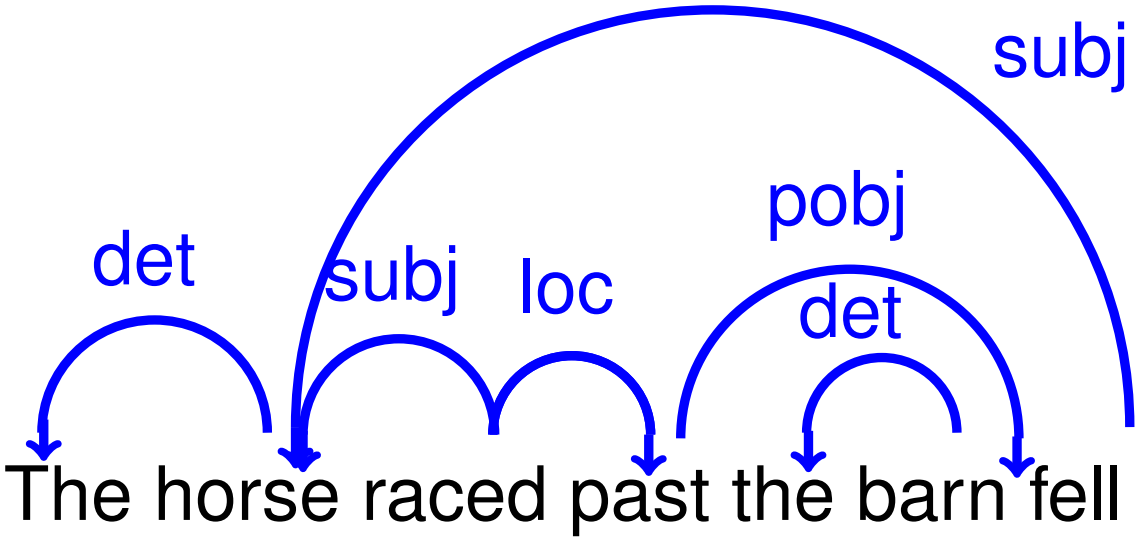
- Parse the sentence left to right
  - Try to link every word to every previous word as permitted by grammar

$$O(n^2)$$

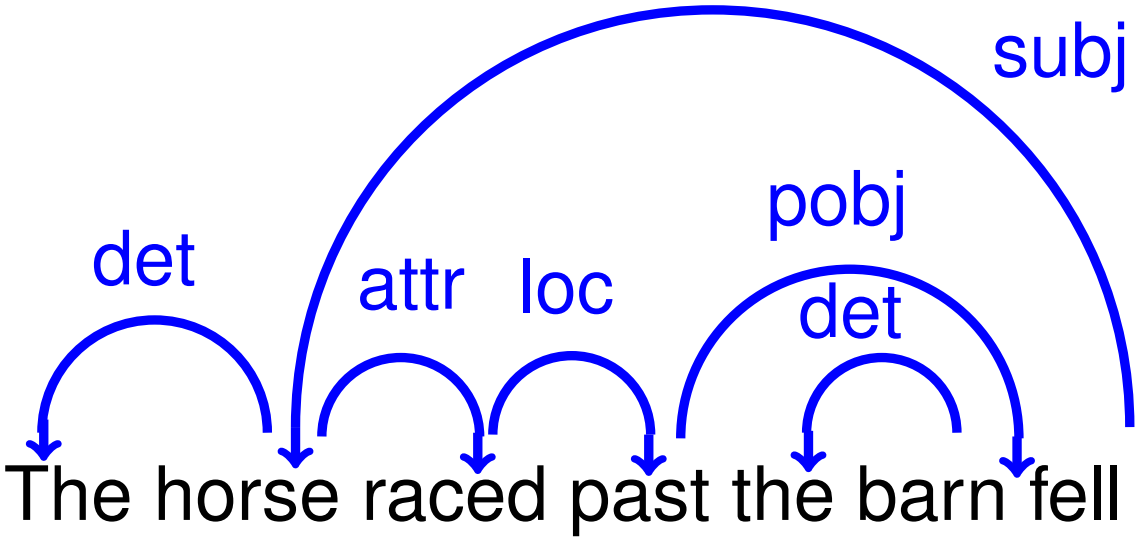
Complexity of Convington's Algo



Complexity of Convington's Algo



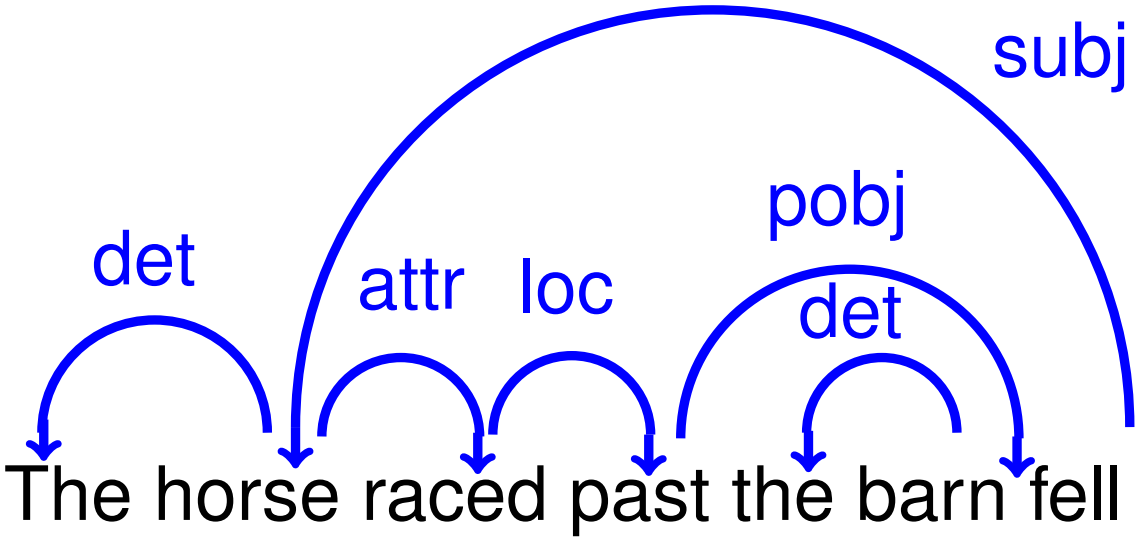
Complexity of Convington's Algo



Complexity of Convington's Algo

Parse left to right, connect every word to every possible preceding word (with constant access to the grammar): $O(N^2)$

With backtracking: $O(N^3)$



Parsing

When lexical ambiguity and agreement are present, parsing is NP-complete.

Barton et al: Computational Complexity and Natural Language.

Parsing

When lexical ambiguity and agreement are present, parsing is NP-complete.

Barton et al: Computational Complexity and Natural Language.

The worst case does not occur, i.e., people do not actually utter sentences that put any reasonable parsing algorithm into a worst-case situation.

Convington: A Fundamental Algorithm for Dependency Parsing

Parsing

When lexical ambiguity and agreement are present, parsing is NP-complete.

Barton et al: Computational Complexity and Natural Language.

The worst case does not occur, i.e., people do not actually utter sentences that put any reasonable parsing algorithm into a worst-case situation.

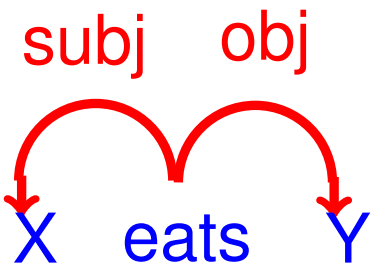
Convington: A Fundamental Algorithm for Dependency Parsing

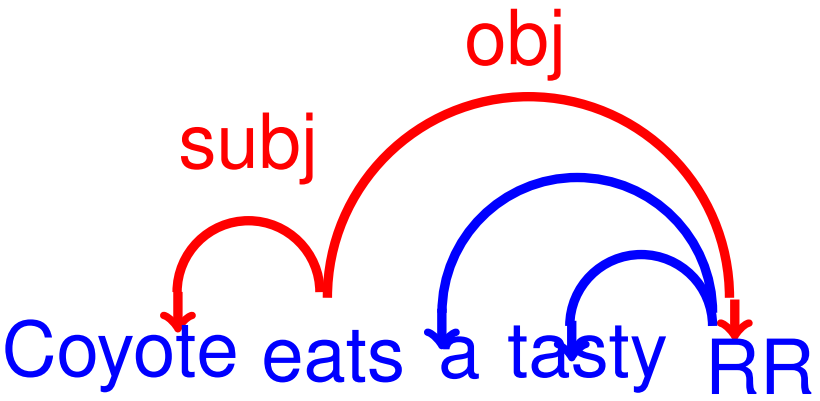
Dependency parsing works very well in practice.

Def: Dependency Patterns

A dependency pattern is a chain graph whose nodes are words or X or Y.

A dependency pattern p matches a dependency graph g, if there is a substitution σ for X and Y such that $\sigma(p) \subseteq g$.

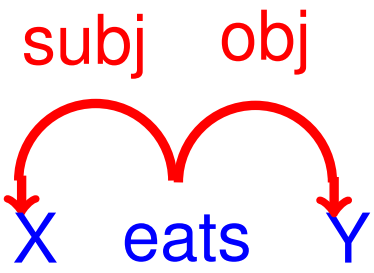


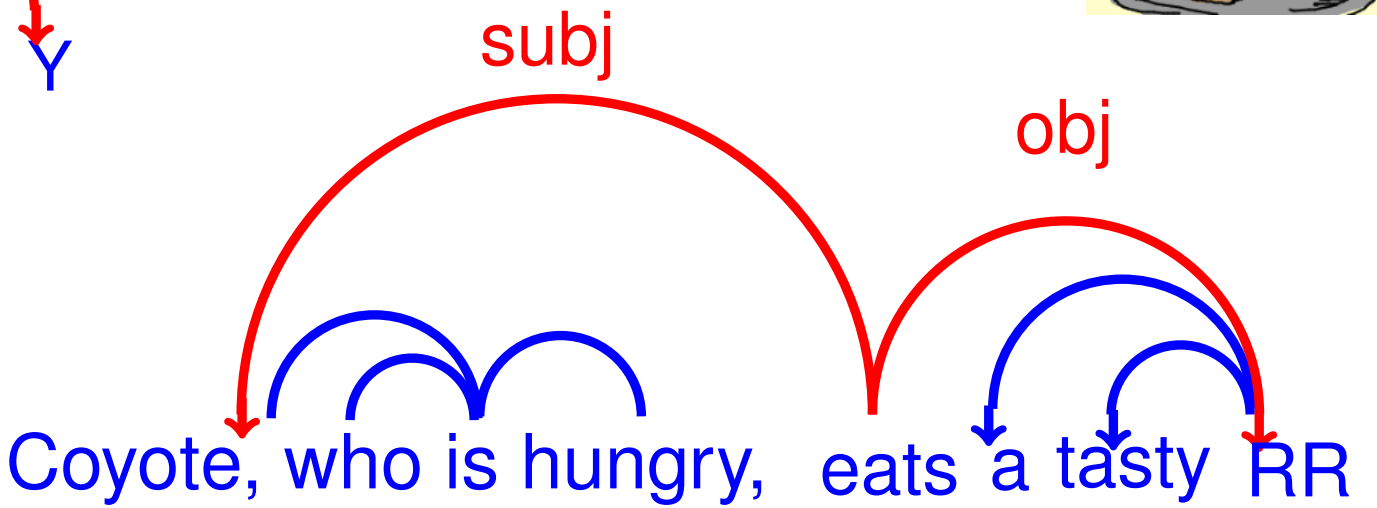


Def: Dependency Patterns

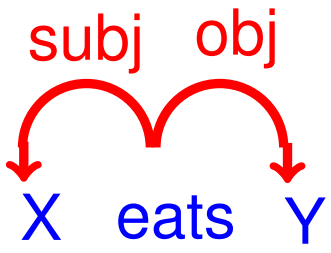
A dependency pattern is a chain graph whose nodes are words or X or Y.

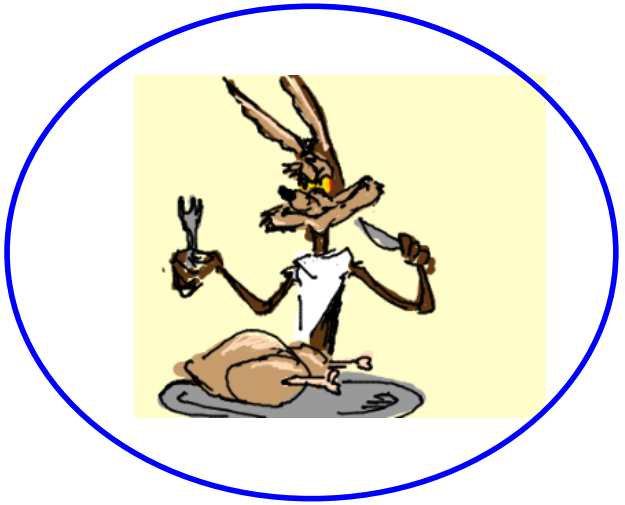
A dependency pattern p matches a dependency graph g, if there is a substitution σ for X and Y such that $\sigma(p) \subseteq g$.

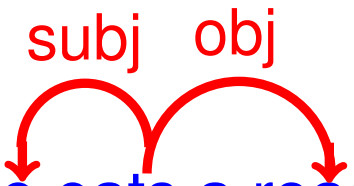




... but not all problems are solved







Coyote dreams that Coyote eats a roadrunner.

->ie-by-reasoning