Homework 5

- Posted on Blackboard
- Due **MONDAY** November ~~30~~ **20** 11:59pm (Monday before Thanksgiving)
- Last homework of this class

[Typo corrected: HW is due Nov 20]

Dependency Parsing

CS-585

Natural Language Processing

Sonjia Waxmonsky

Based on slides from Kai Shu and
Derrick Higgins

REVISITING PCFGS: LEXICALIZATION

Lexicalization

- Another approach to incorporating more contextual information into PCFGs is *lexicalization*
- Note that the likelihood of specific rules applying often ***depends on specific words*** (especially, subcategorization)
- The idea of lexicalization is to use properties of phrasal head words to get better estimates of rule probabilities

Rule	P(Rule)
VP → V NP	?
VP → V NP NP	?
AP → Adj	?
AP → Adj PP	?

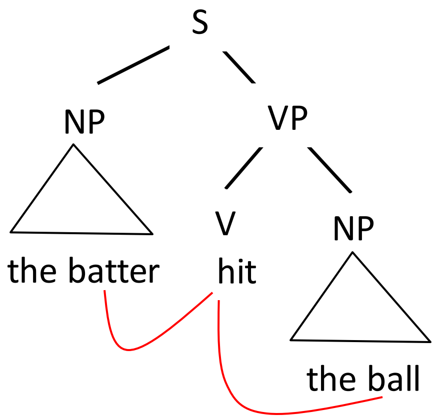
see
give
…

exciting
proud
…

Phrasal Heads

From Session 22

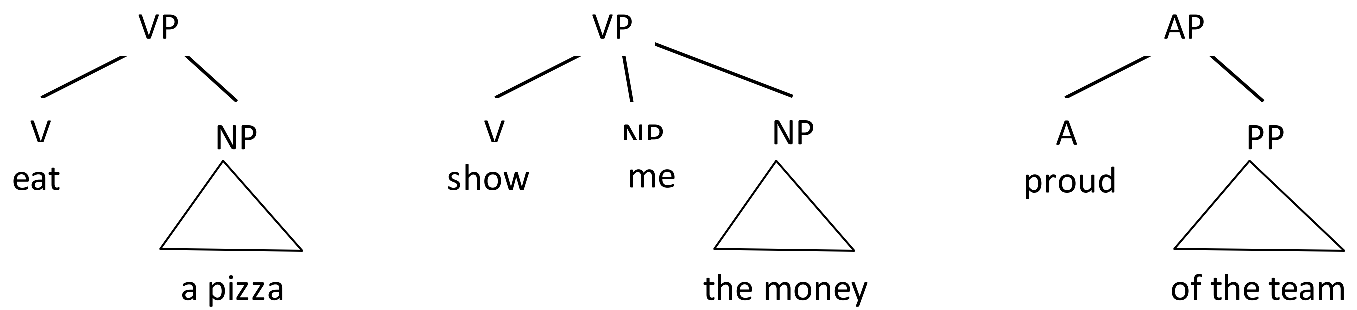
- The *head* of a phrase is the word that determines its attributes
  - Typically, of the same category as the phrase: the head of a noun phrase is a noun, the head of a prepositional phrase is a preposition, etc.
  - Attributes of the head (e.g., tense in the case of verbs, number and case in the case of nouns) are shared by the phrase as a whole
  - Relationships between heads of phrases are strongly predictive for parsing



Subcategorization

From Session 22

- Subcategorization is the relationship between a **syntactic** head word and the **dependents** it requires
  - A transitive verb like “eat” subcategorizes for a single noun phrase
  - A ditransitive verb like “show” subcategorizes for two noun phrases
  - The adjective “proud” subcategorizes for a prepositional phrases headed by “of”



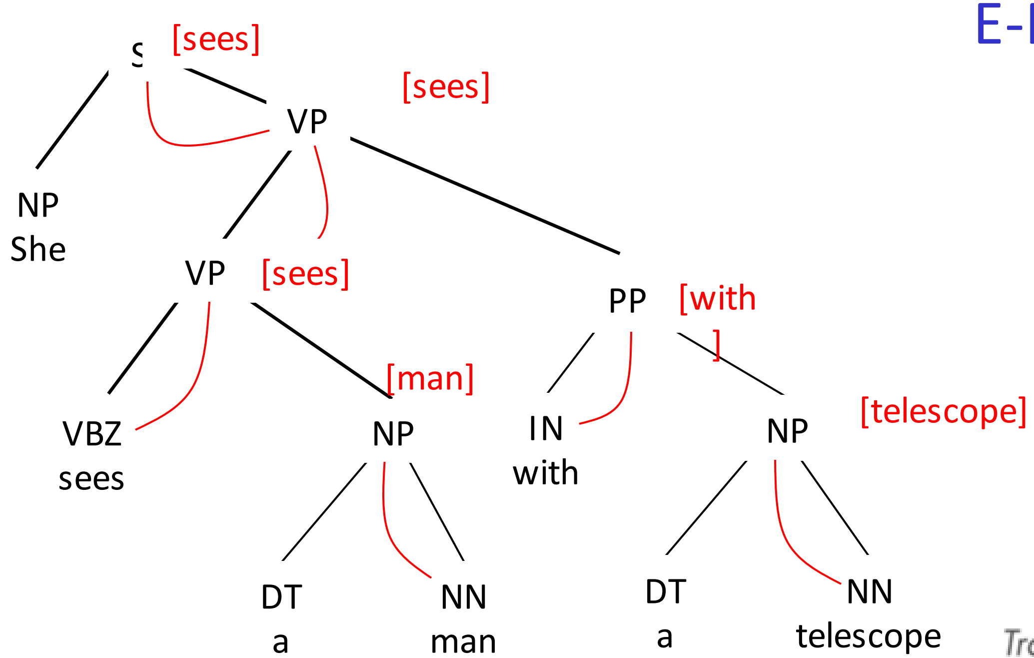
Head identification for lexicalization

- In a previous lecture we learned about phrasal heads:
- The head of a phrase is the word that determines its attributes
  - Typically of the **same category as the phrase**: the head of a noun phrase is a noun, the head of a prepositional phrase is a preposition, etc.
  - Attributes of the head (e.g., tense in the case of verbs, number and case in the case of nouns) are shared by the phrase as a whole

Head identification for lexicalization

Nonterminal	Direction	Priority
S	right	VP SBAR ADJP UCP NP
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP
NP	right	N* EX $ CD QP PRP …
PP	left	IN TO FW

E-NLP Table 10.3 p246

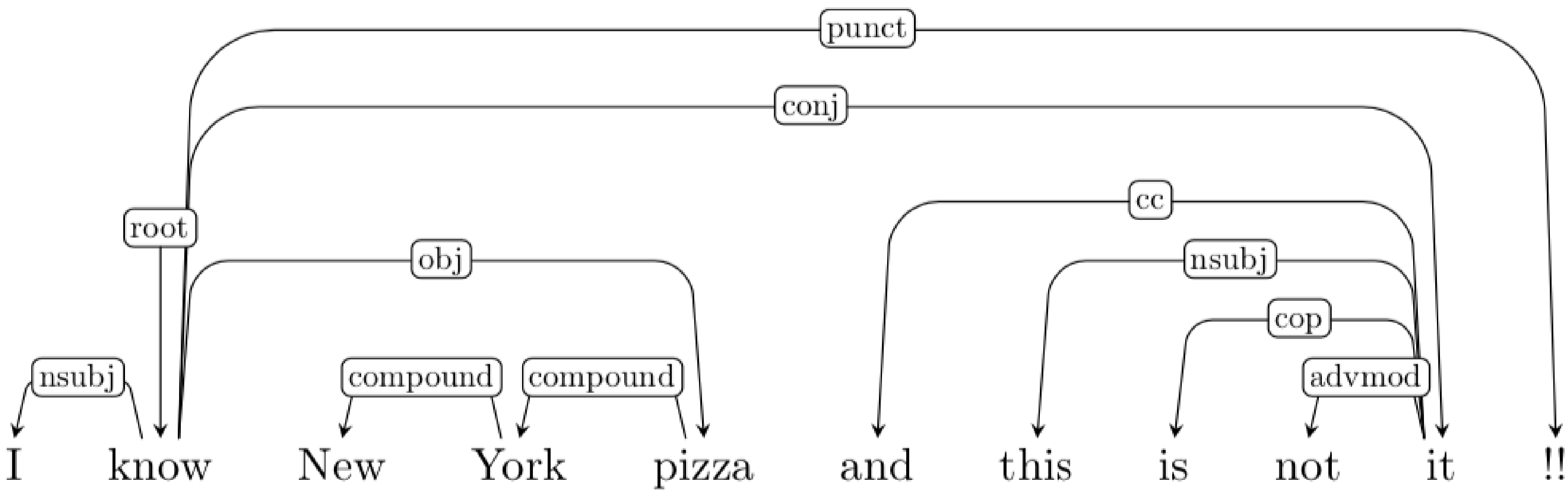


DEPENDENCY STRUCTURES

Dependency trees

Dependency:

Labeled, asymmetric relationship between two words



E-NLP Fig 11.3 (p260)

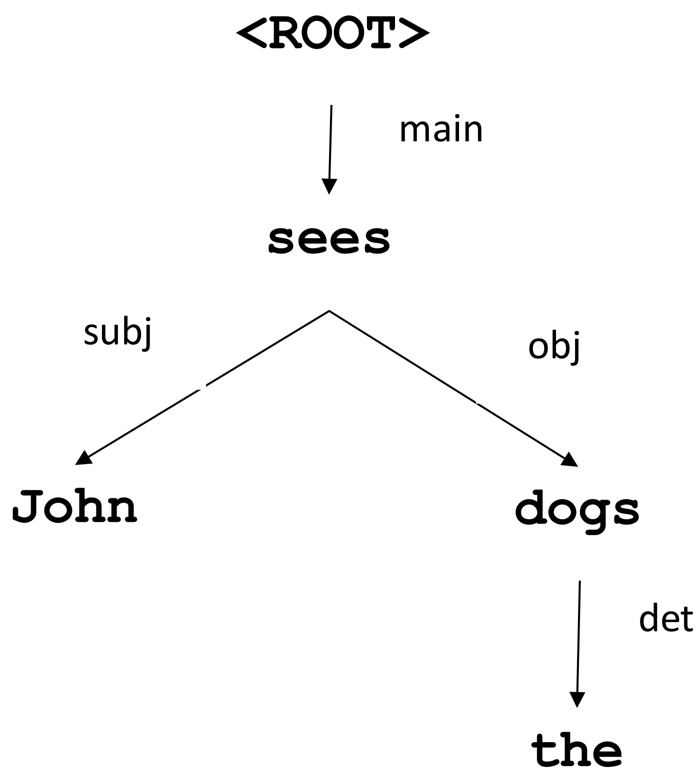
Dependency Grammar

- No constituency or phrase structure
- Binary **dependency relations** between words

  head → modifier (dependent)

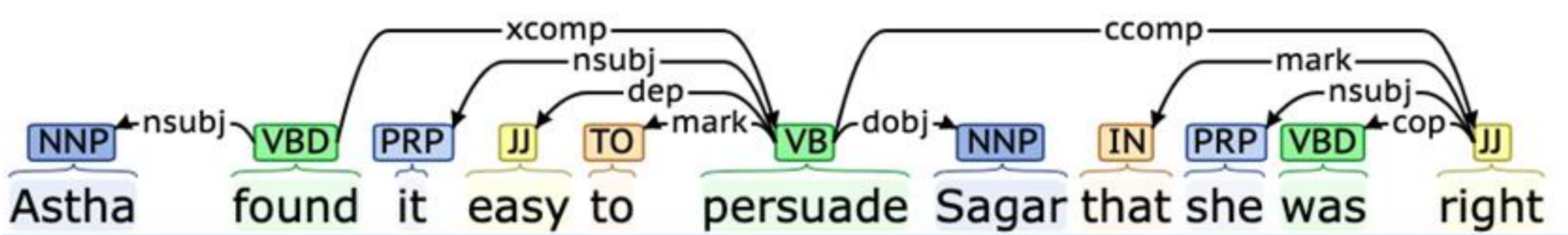
- Some dependency relations:
  - main = main verb
  - subj = syntactic subject
  - obj = direct object
  - det = determiner
  - mod = nominal postmodifier (e.g. PP)
  - attr = attributive (premodifying) nominal

Example Dependency Tree



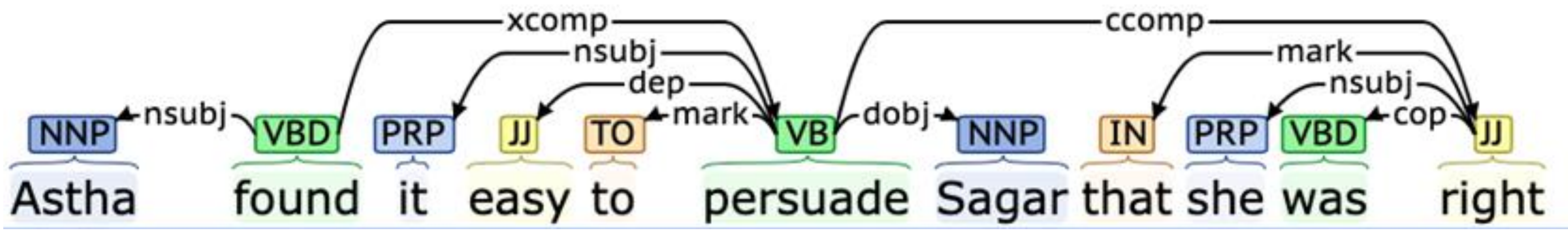
Another Example

Astha found it easy to persuade Sagar that she was right



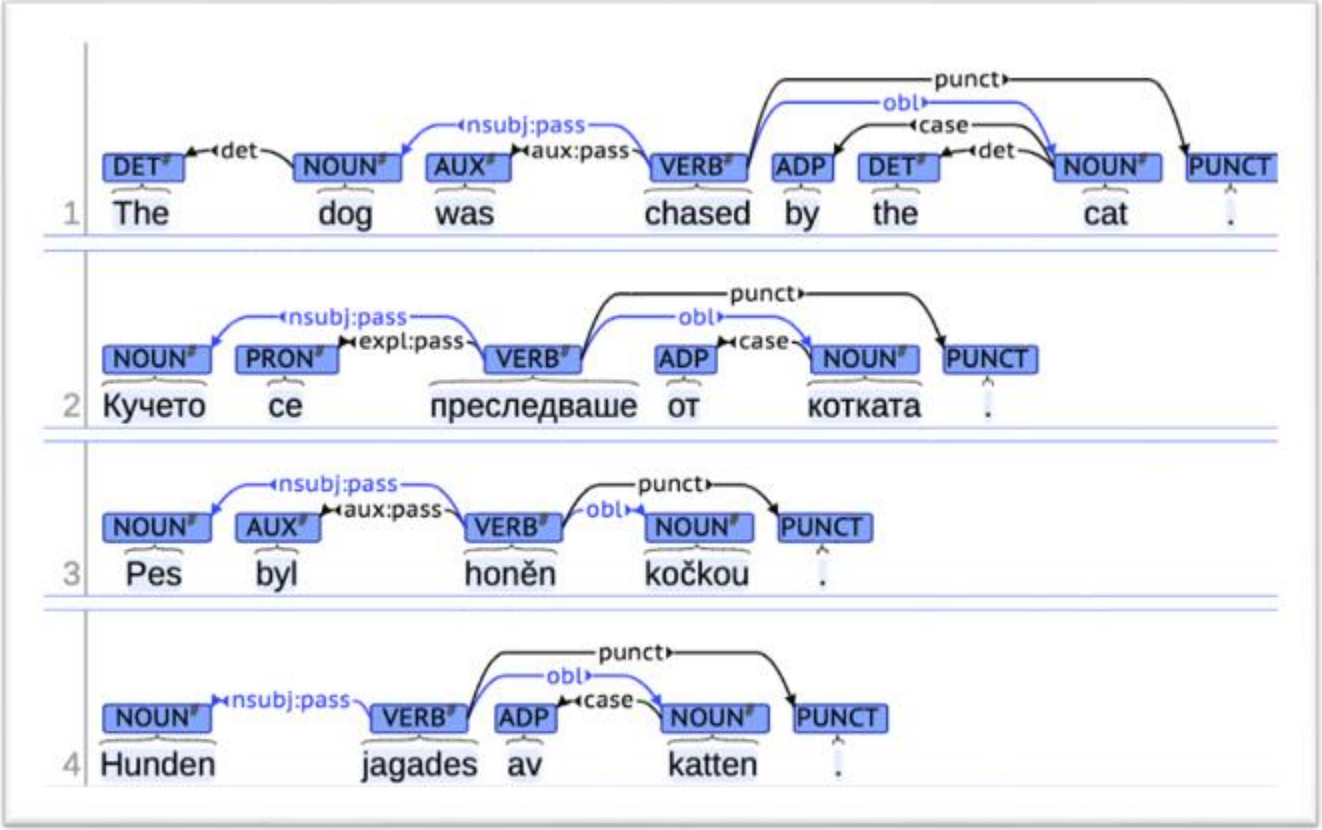
Syntactic Dependencies

- Each word is the **dependent of a single head**
- A head can have **multiple dependents**
- There are no clear "constituents", although there are some constraints on word ordering and contiguity (later…)
- Links are of different types



Universal Dependencies Project

- Goal: Framework for treebank annotation that is **consistent across languages**
- 100+ languages represented



English, Bulgarian, Czech and Swedish
https://universaldependencies.org/introduction.html

Example: Universal Dependencies

	Nominals	Clauses	Modifier words	Function words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case

https://universaldependencies.org/u/dep/

Robinson's Axioms (1970)

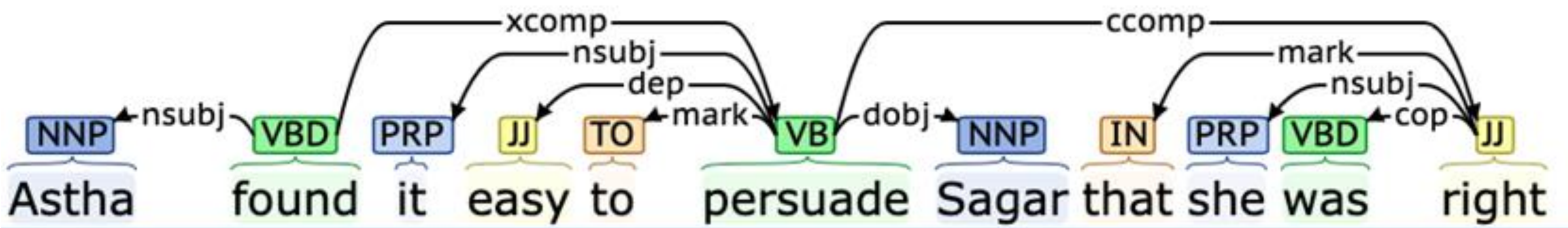
- One and only one element is independent (the *root*)
- All other elements depend on some other element
- No element depends directly on more than one element
- Projectivity: If A depends **directly** on B and some element C is between them in the **string**, then C must depend on A, B, or some other element between them
  - But, projectivity may not always be suitable

Projectivity

Property of Projectivity - the head *h* of any constituent that spans the nodes from ***i*** to ***j*** must have a path to every node in this span

- Does not allow "crossing branches"

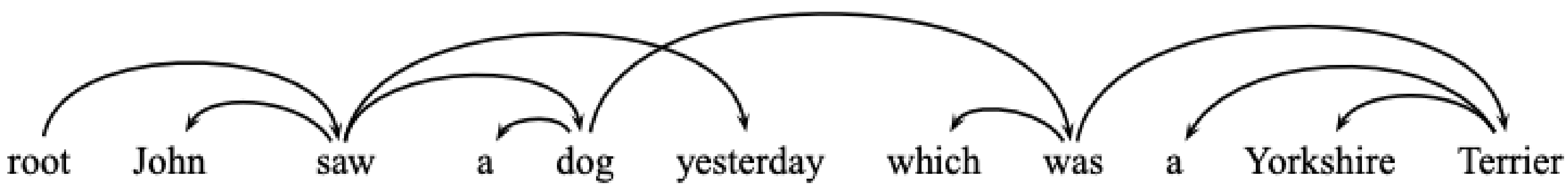
Any crossing branches in this example?



Discontinuous Constituents

- We can find examples of constituents (words that function as a unit that are not contiguous in word order
- In a dependency parse: "crossing branches"
- Violates axiom of projectivity

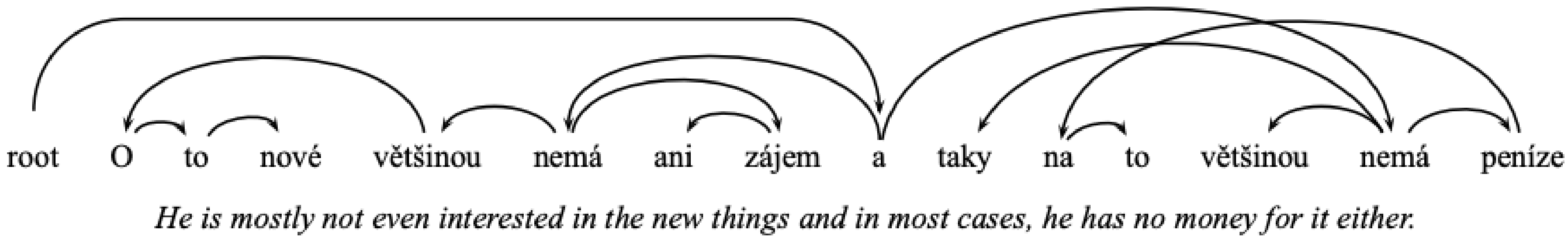
"John saw ***a dog*** *yesterday* ***which was a Yorkshire Terrier****"*



https://aclanthology.org/H05-1066.pdf

Free word order languages

- **Free word order** languages allow constituents to occur in different linear arrangements
- Difficult to capture with phrase structure production rules
- Dependency structures often considered to be a better representation for **free word order** languages (e.g. Czech)



Czech example:
https://aclanthology.org/H05-1066.pdf

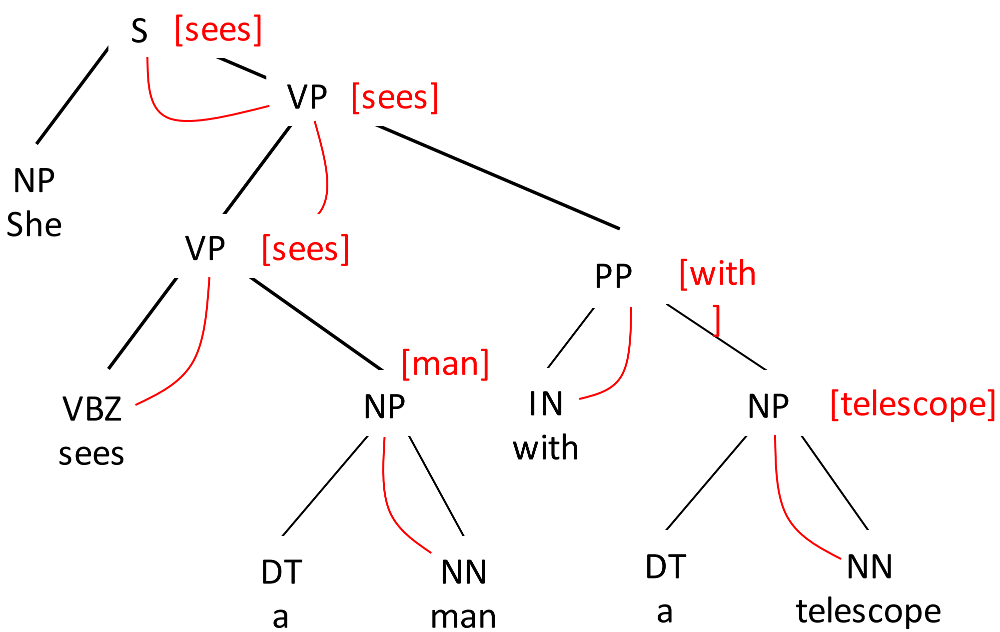
DEPENDENCY STRUCTURES AND PHRASE STRUCTURE

Dependencies and phrase structure

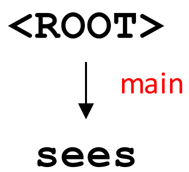
- Dependencies are relationships between heads of syntactic phrases
- We can identify **phrasal** heads using phrase structure representations
  - They have all the information we need for a dependency graph
  - And more information we don't about the order of words and phrases
- So for *projective* dependency parsing, we always have the option of just doing regular CFG parsing and converting to dependencies later

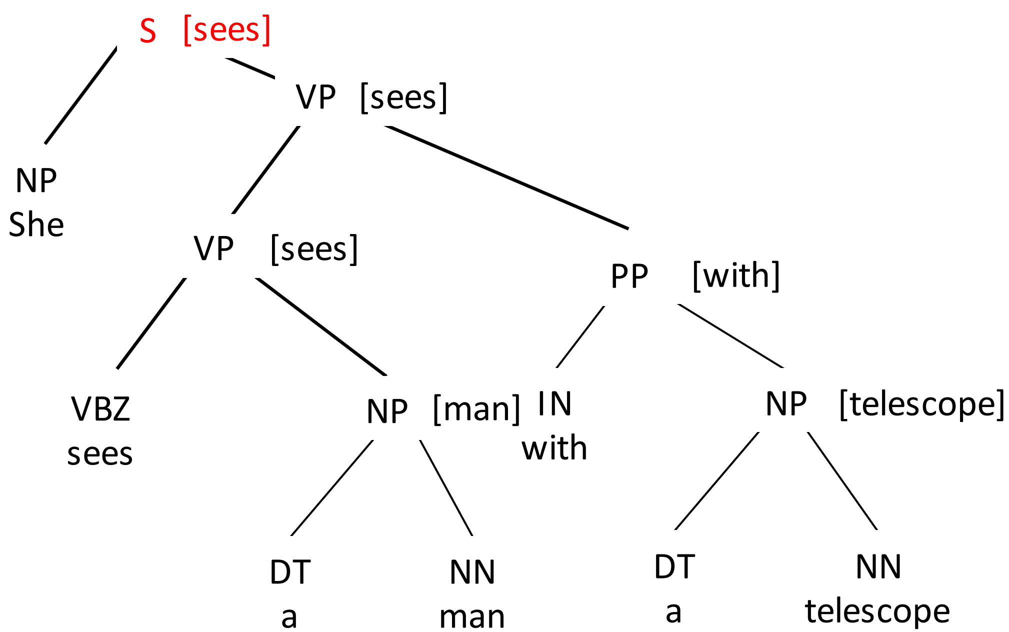
Head identification

Nonterminal	Direction	Priority
S	right	VP SBAR ADJP UCP NP
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP
NP	right	N* EX $ CD QP PRP …
PP	left	IN TO FW

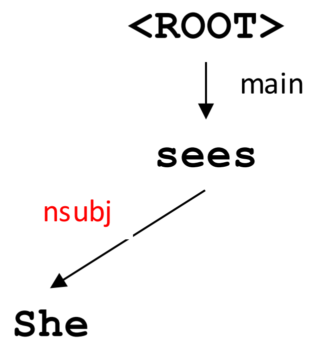


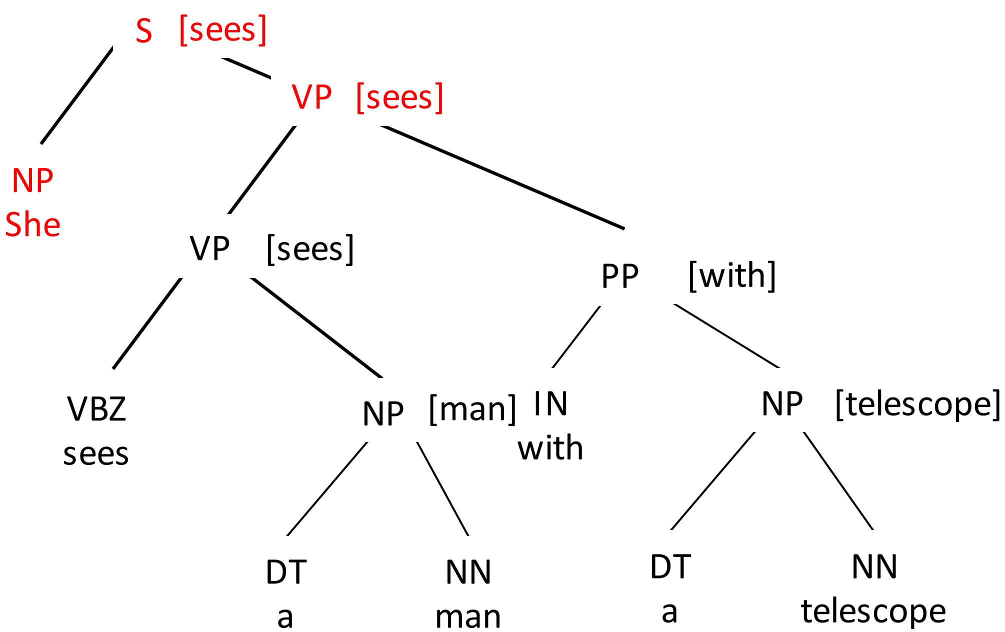
Dependency conversion



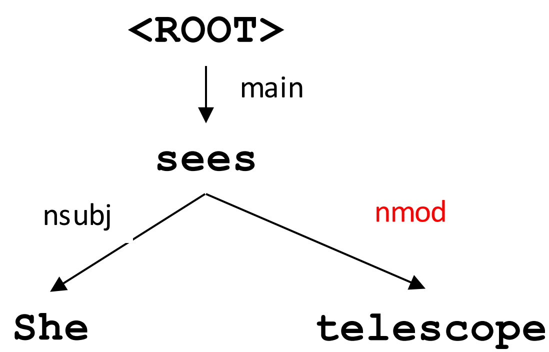


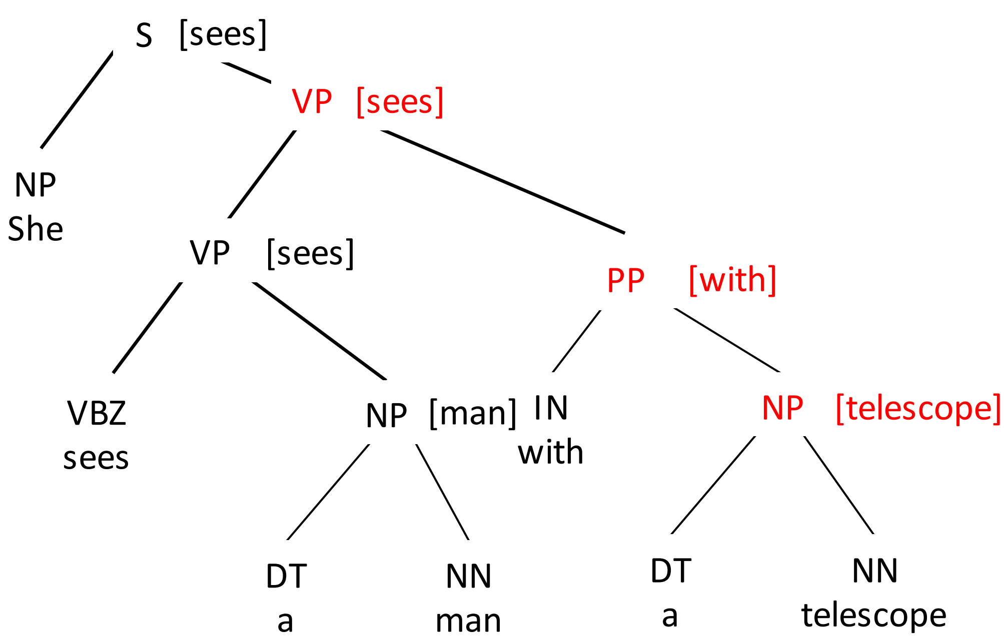
Dependency conversion



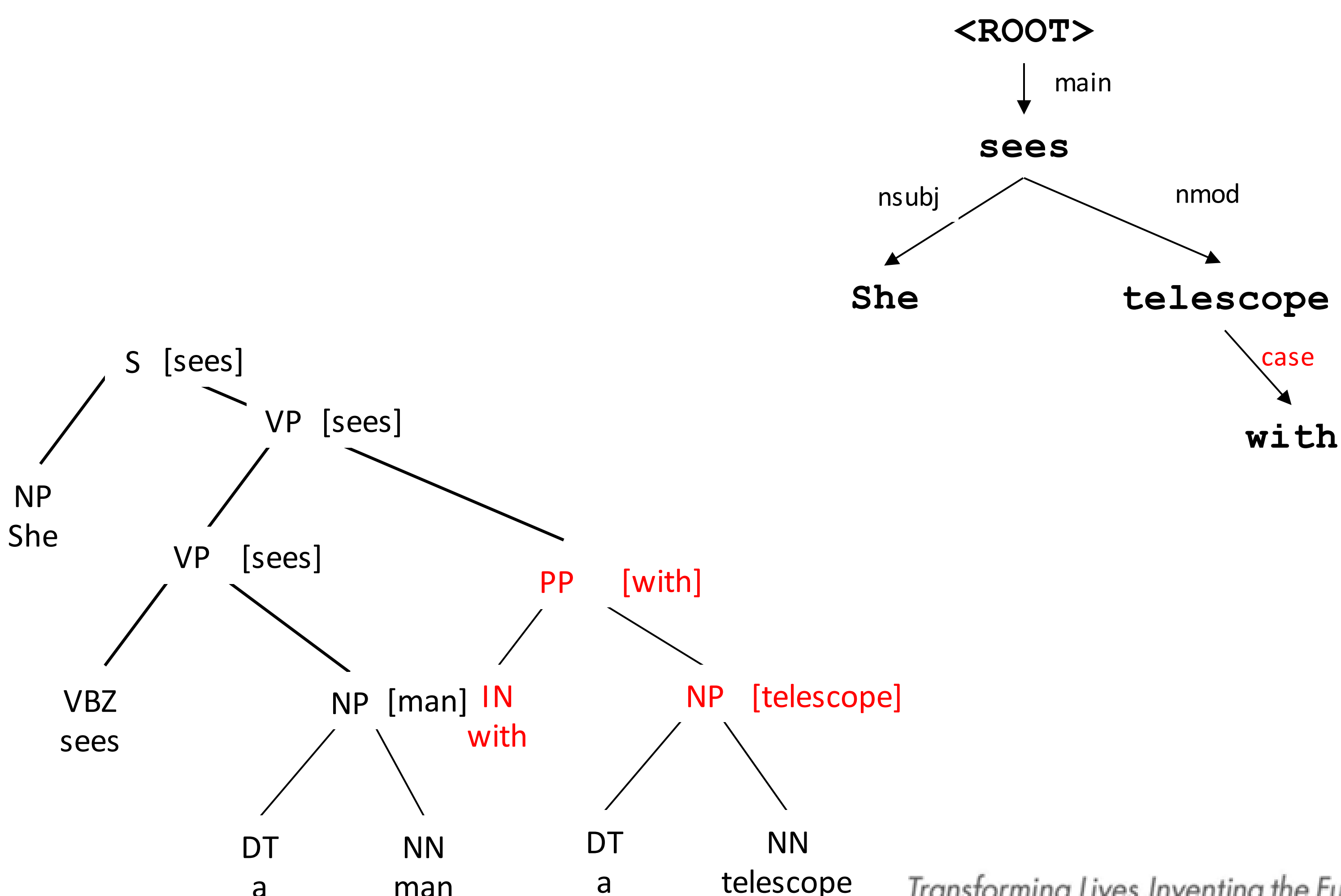


Dependency conversion

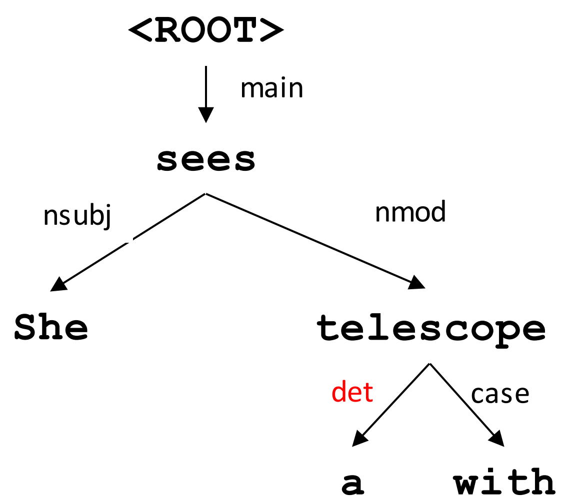


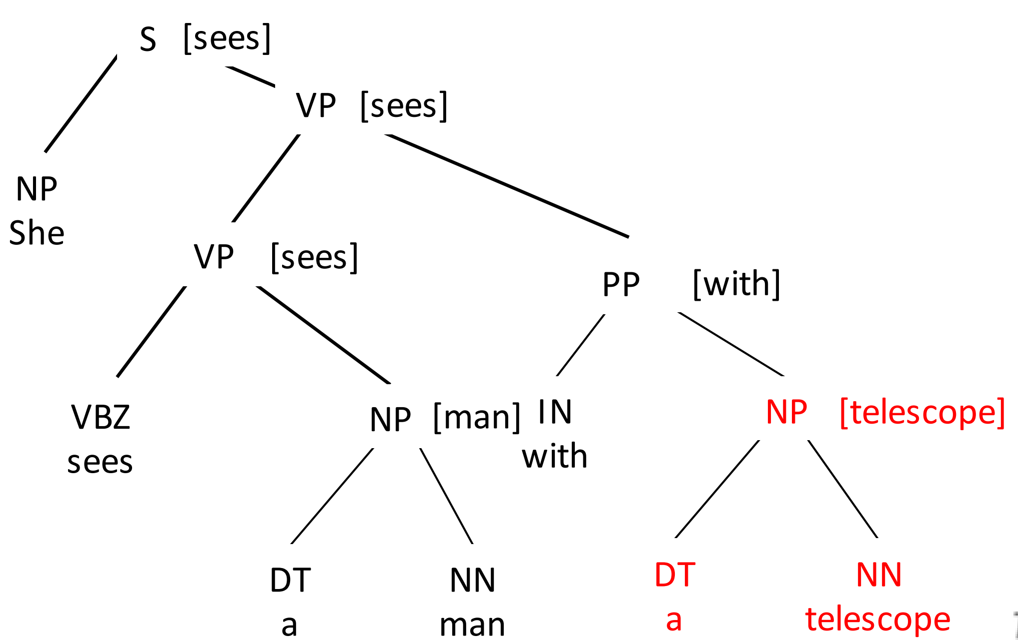


Dependency conversion

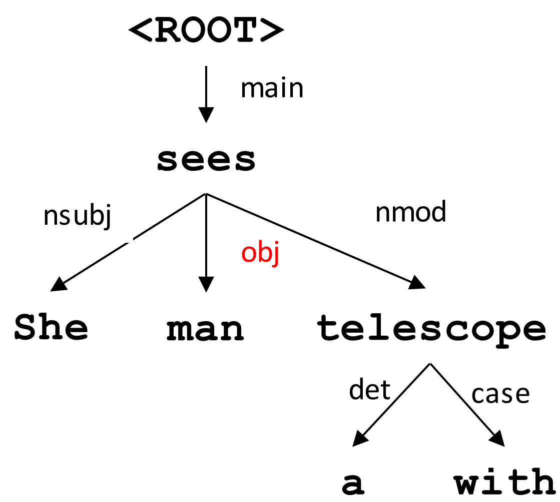


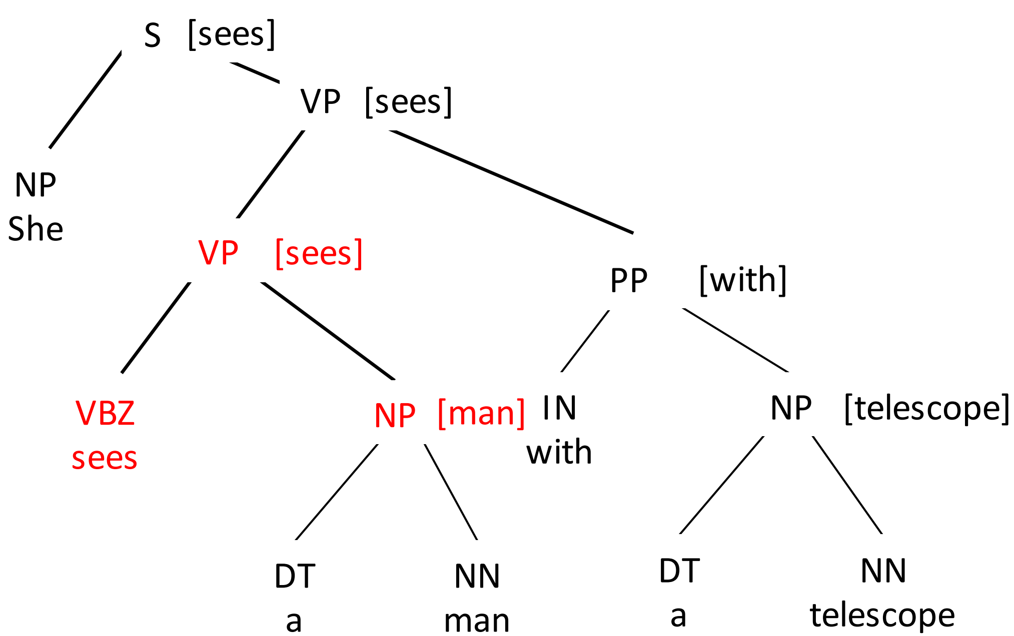
Dependency conversion



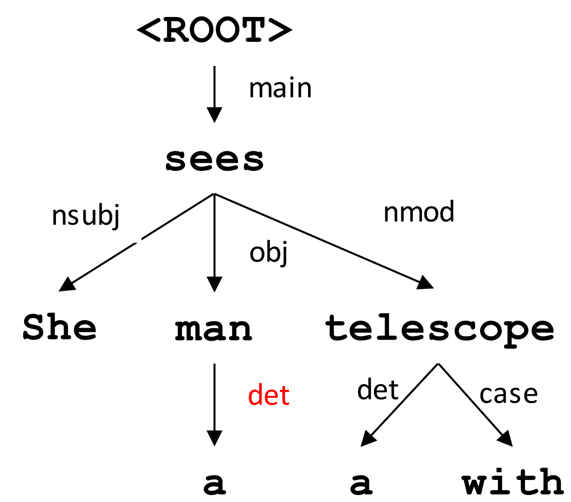


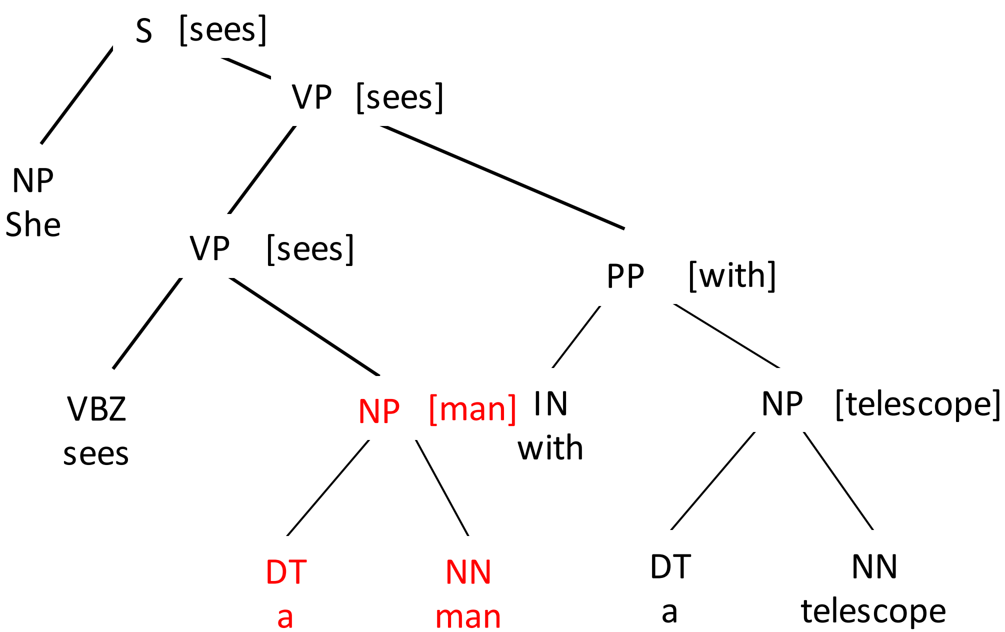
Dependency conversion





Dependency conversion





DEPENDENCY PARSING

Dependency parsing

- For ***projective*** dependencies, we have the option of working with phrase structures instead of dependencies directly
- But this may not be optimal: a single dependency relation may be represented by a variety of phrase structure configurations
- Also, doesn't handle non-projective case

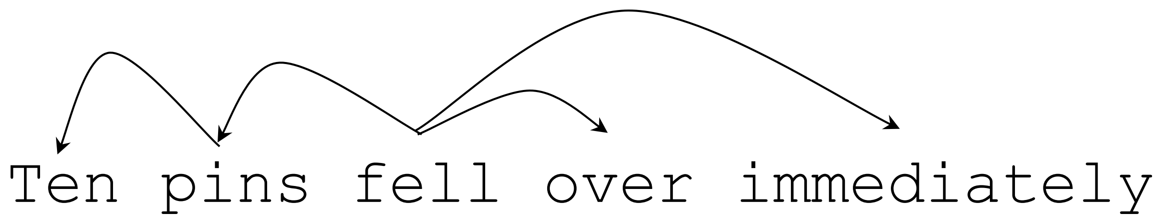
Dependency parsing

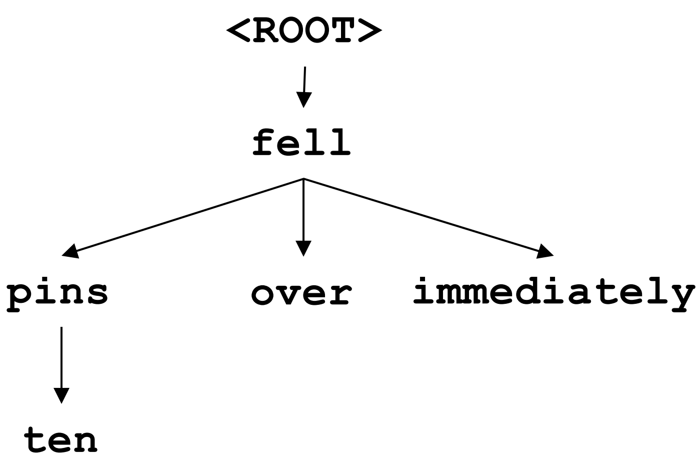
- Instead, we can work in a pure ***dependency*** framework. Concepts from PCFGs have dependency grammar analogs
  - Instead of the highest-probability CFG node with label S spanning the sentence, we want to find the Maximum Spanning Tree
  - Decomposition of score for sentence uses similar dynamic programming approach, but different data structures from our familiar chart

Spanning Trees

- A spanning tree for a sentence
  - Has a single root
  - Contains directed edges such that
  - every word can be reached from the root by following some sequence of edges
    - no word is the dependent of multiple elements (each word appears only once as the destination of an edge)
  - Contains no cycles
- If it is projective
  - Any element between a head and its direct dependent (in linear order) must be a (direct or indirect) dependent of one or the other

Spanning Trees





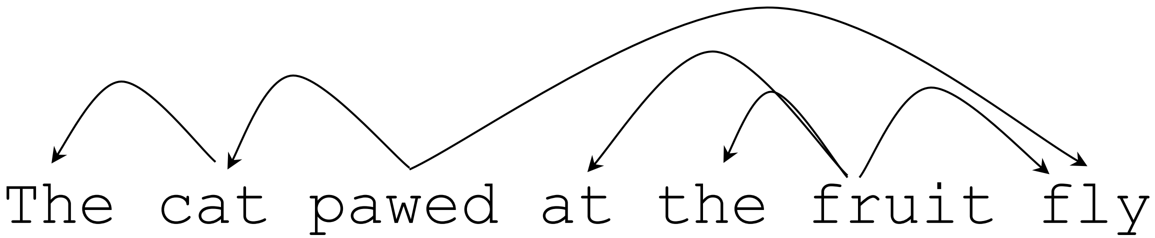
Spanning tree?

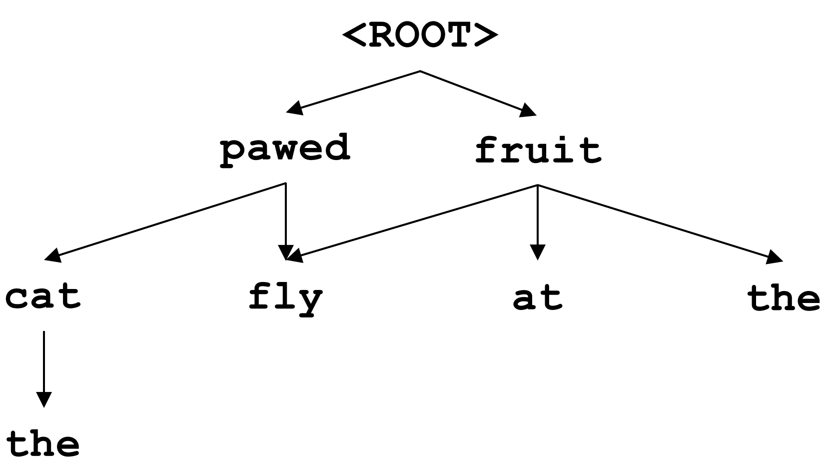
Yes

Projective?

Yes

Spanning Trees

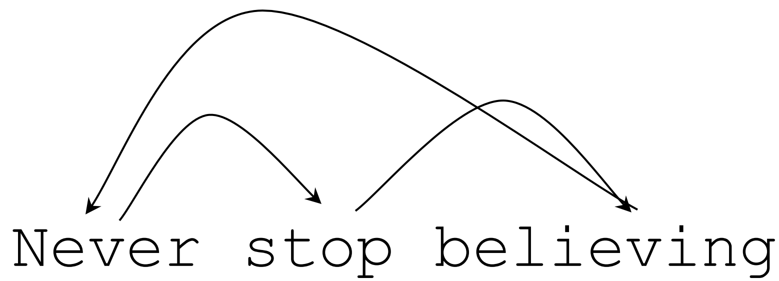


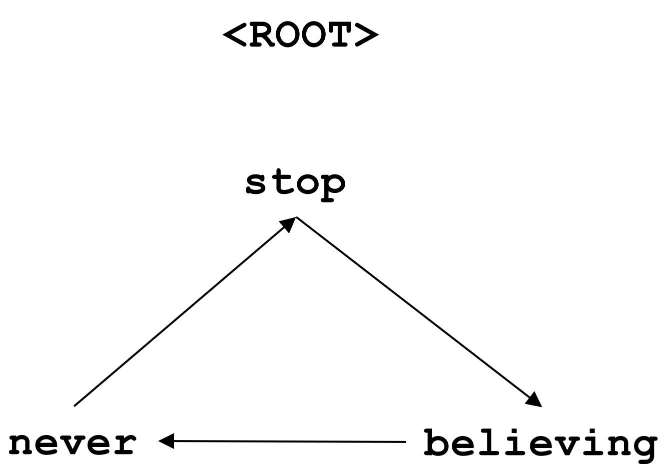


Spanning tree?

No

Spanning Trees

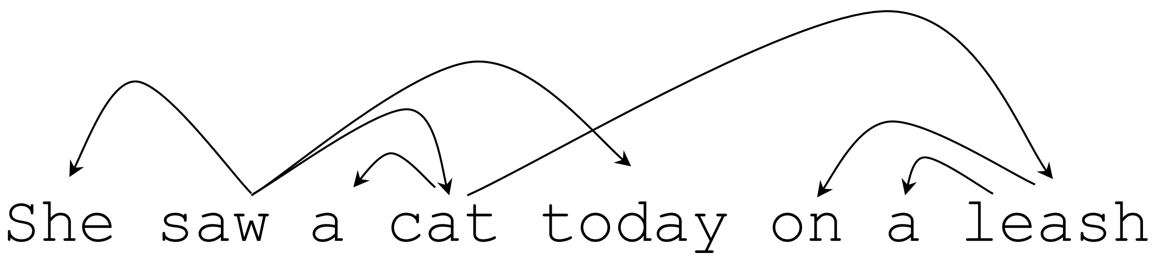


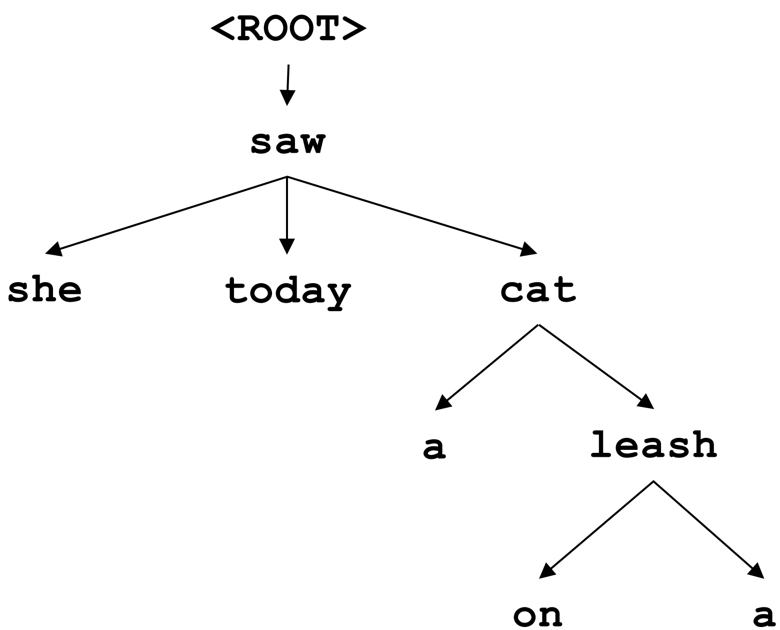


Spanning tree?

No

Spanning Trees





Spanning tree?

Yes

Projective?

No

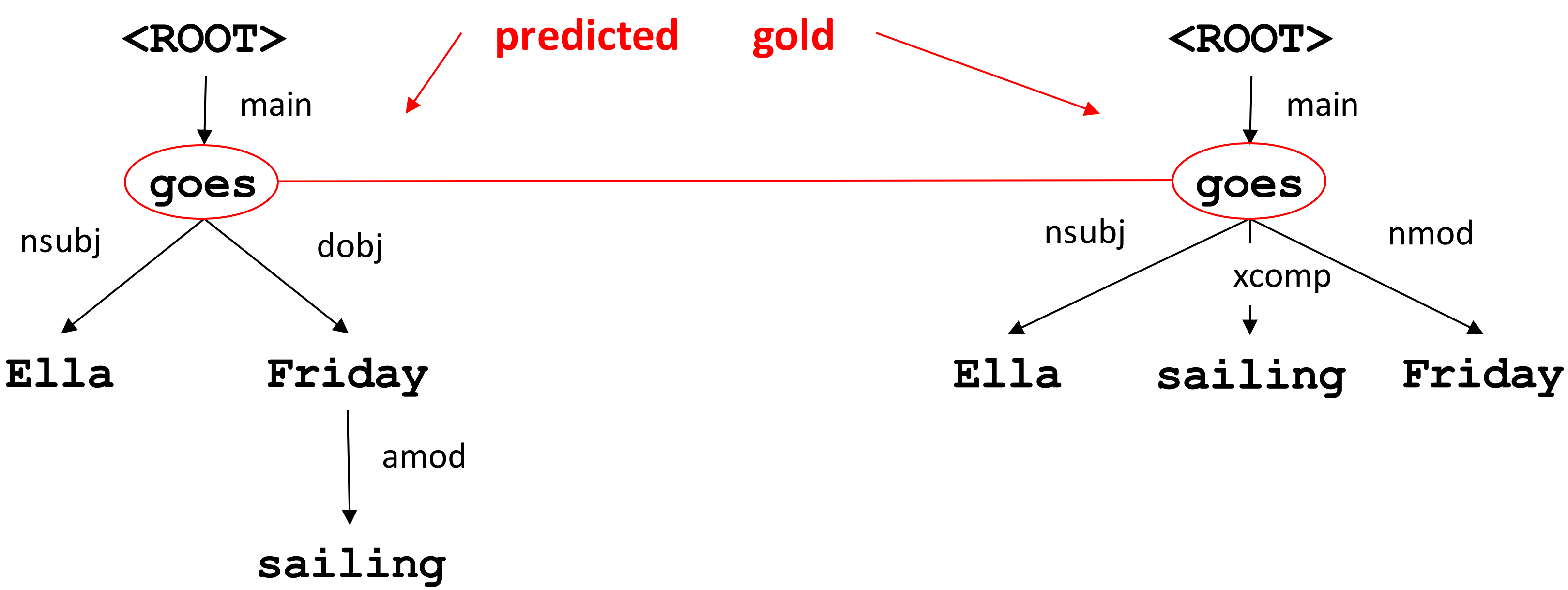
Dependency parsing algorithms

- The Maximum Spanning Tree is the spanning tree with the highest score (probability)
- Given a model (probabilistic or neural) for assigning scores to dependency relations between words, there are efficient algorithms for finding the MST
  - Eisner algorithm for projective parsing – $O(N^3)$ in the length of the sentence
  - Chu-Liu-Edmonds algorithm for non-projective parsing – $O(N^2)$ in the length of the sentence

EVALUATION

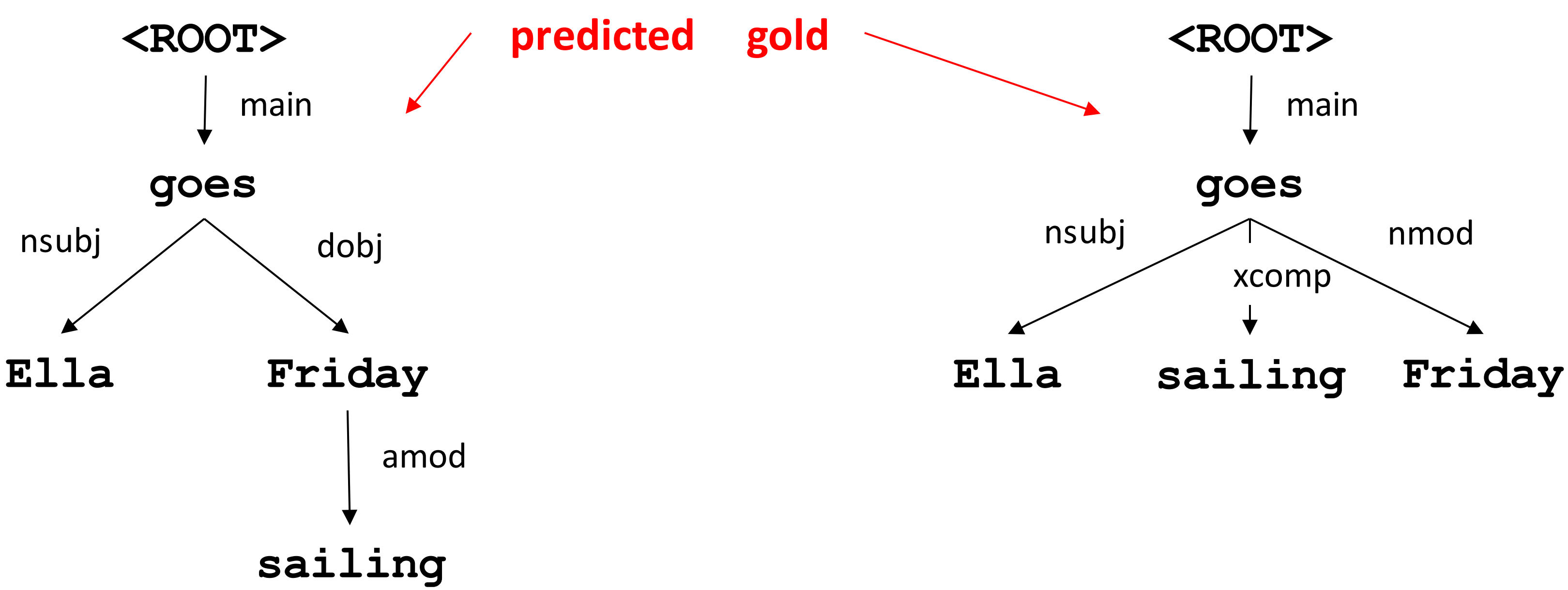
Evaluation measures for dependency parsing

- **Exact match** – as with CFGs, whether we got the entire structure correct. 0%
- **Correct root** – whether we found the correct head word for the entire sentence. 100%



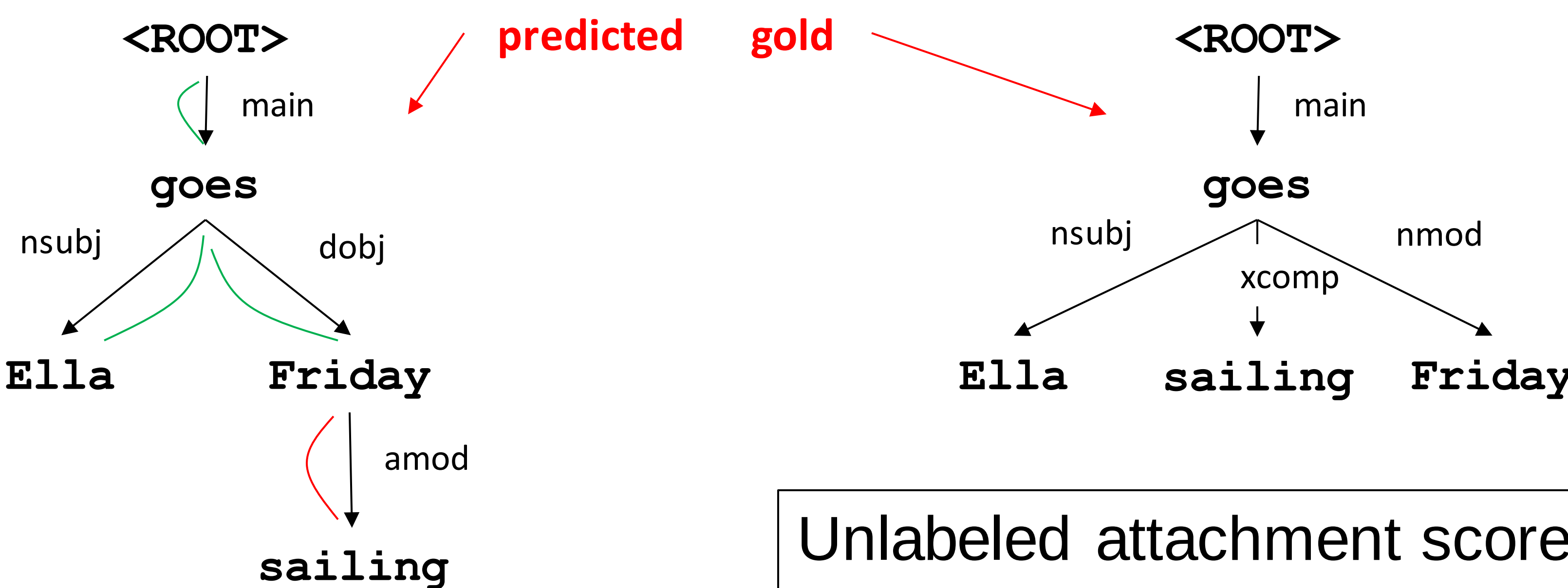
Evaluation measures for dependency parsing

- **Precision / Recall / F-measure per dependency relation** – e.g., here we have 100% precision on `nsubj`, 0% precision on `dobj`



Evaluation measures for dependency parsing

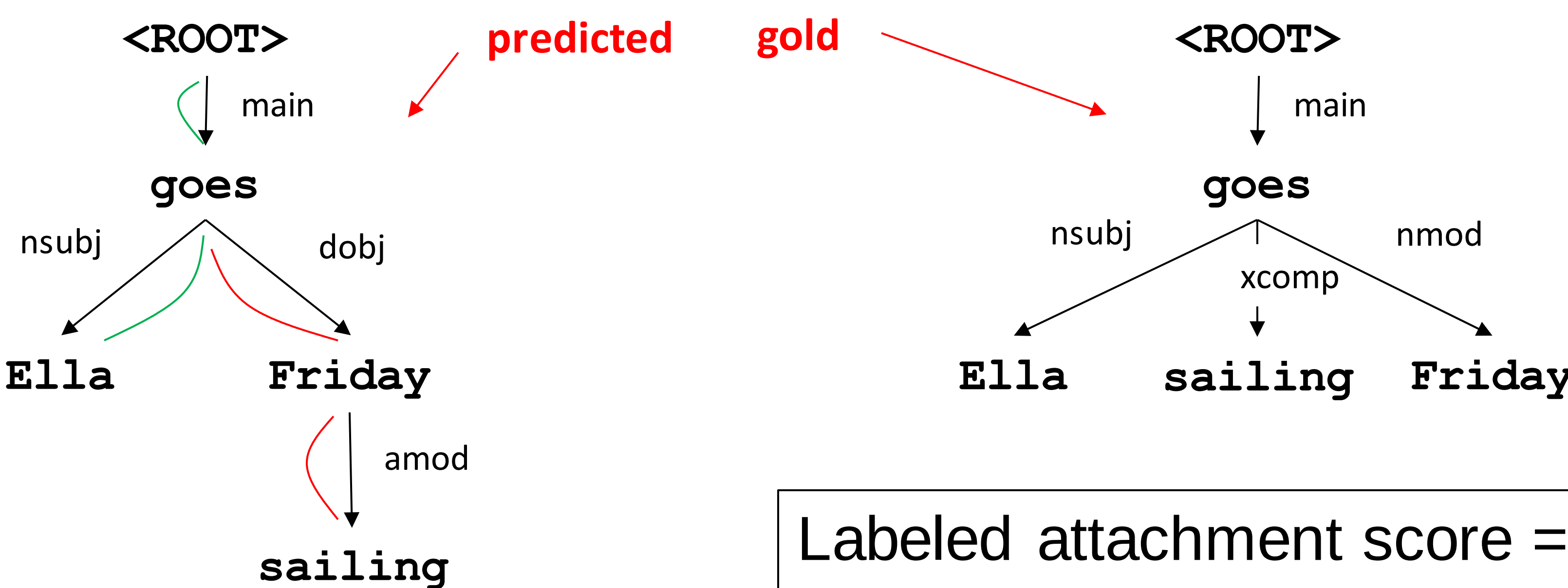
- **Attachment score** – percentage of words that are dependents of the correct head
  - Labeled or unlabeled



Unlabeled attachment score = ¾ = 75%

Evaluation measures for dependency parsing

- **Attachment score** – percentage of words that are dependents of the correct head
  - Labeled or unlabeled



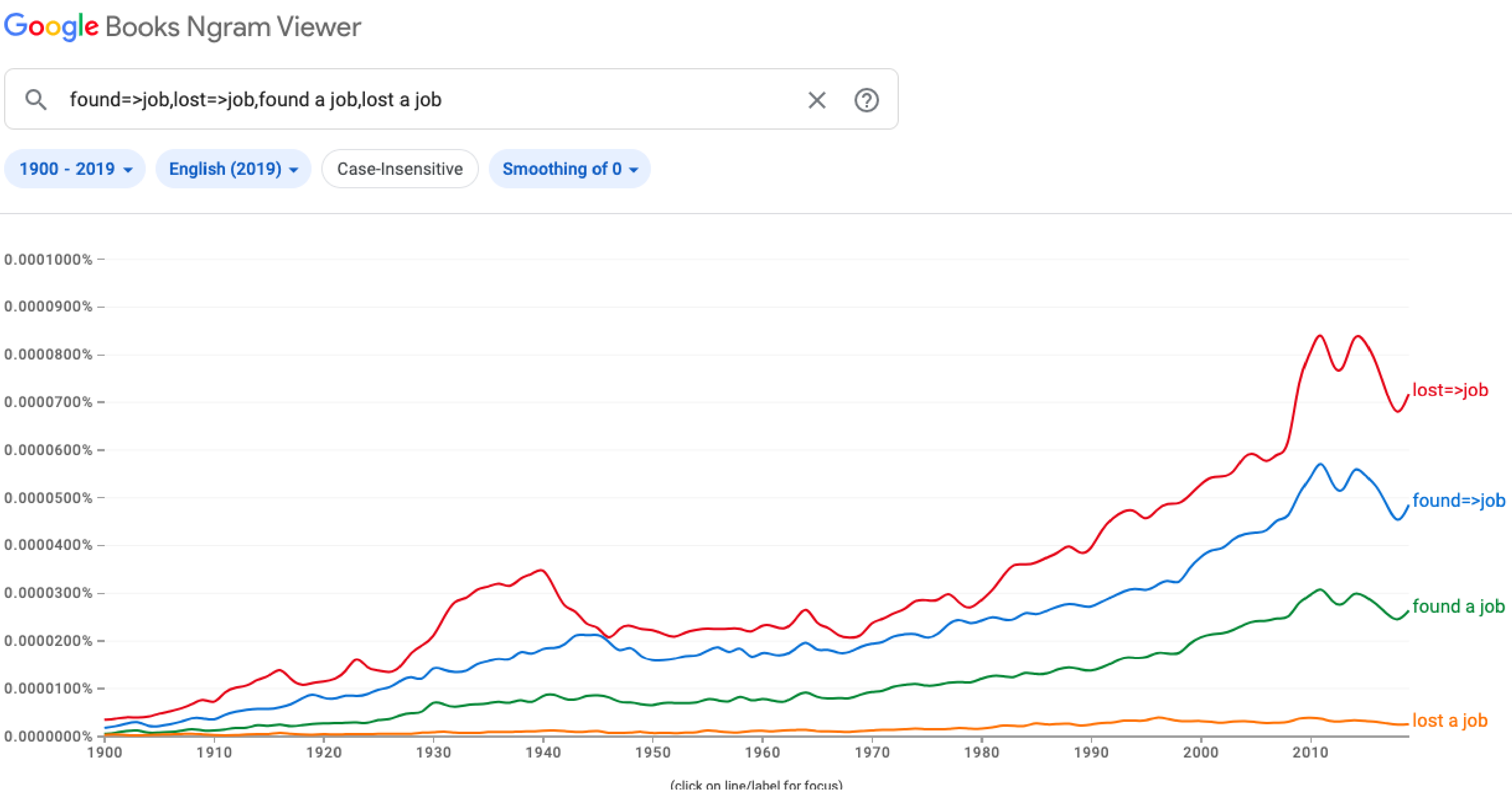
Labeled attachment score = 2/4 = 50%

APPLICATIONS

Generalized ngram search

- Tools like Google Ngrams are used to analyze linguistic change and stylistic patterns
- E.g., how has the usage of the word “job” changed over time?
- But Google Ngrams also supports dependency relations (with the => operator)

Generalized ngram search



Relation extraction

- Dependencies can also be used for **information extraction** tasks, where we are interested in identifying entities that stand in a **specific relationship** to one another

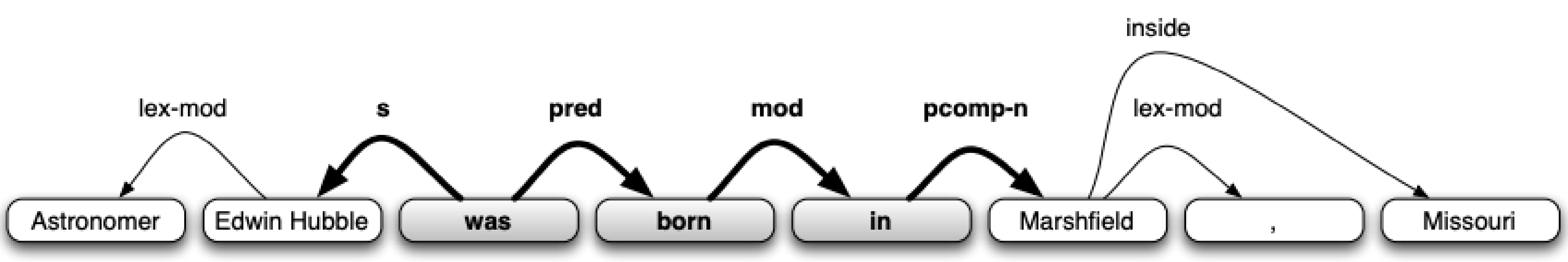
"**Walt Disney Co.** completed its landmark acquisition of **ABC** in 1996"

$\longrightarrow$ Relation: **X** was acquired by **Y**

Relation extraction

Distant supervision for relation extraction without labeled data (Mintz et al., 2009)

- "Distant supervision" - Ground truth labels extracted from existing database (e.g. person / place_of_birth)
- Features include <u>syntactic</u> information: <u>dependency path</u> between words or "chunks" (multi-word phrases)



‘Astronomer **Edwin Hubble** was born in **Marshfield**, Missouri’