ILLINOIS INSTITUTE OF TECHNOLOGY

Dependency Parsing

CS-585

Natural Language Processing

Derrick Higgins

DEPENDENCY STRUCTURES

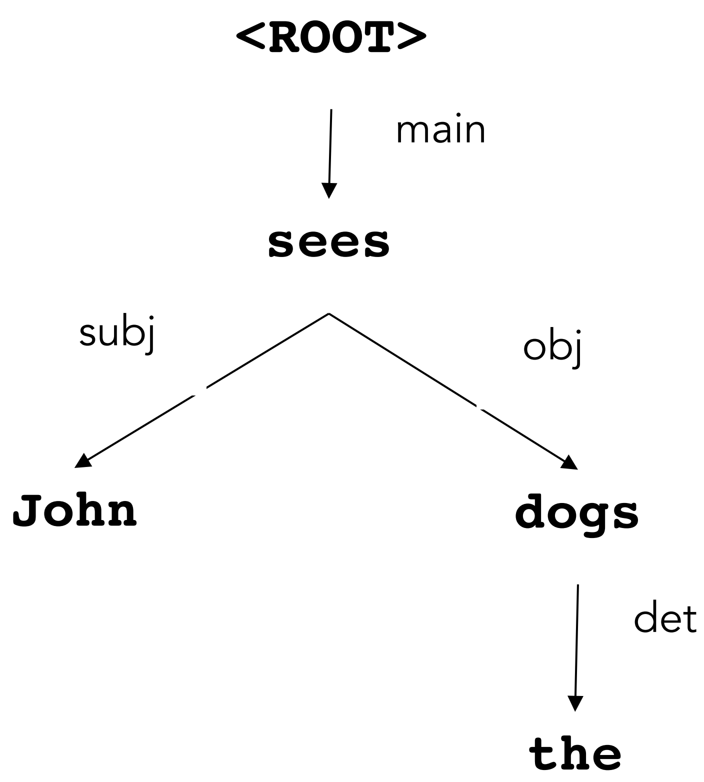
Dependency Grammar

- No constituency or phrase structure
- Binary dependency relations between words (really: *nuclei* = basic semantic units)

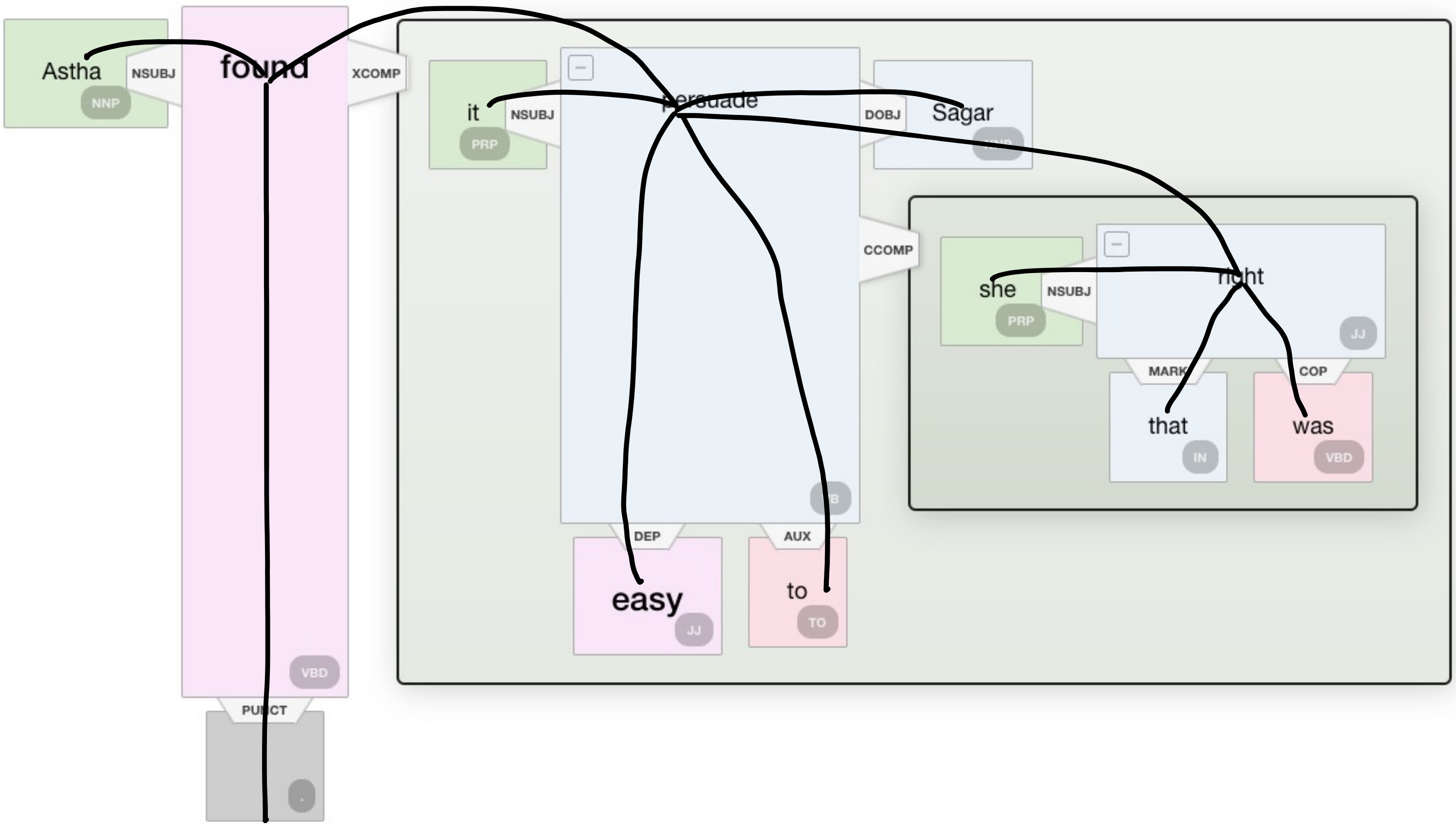
  head → modifier (dependent)

- 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



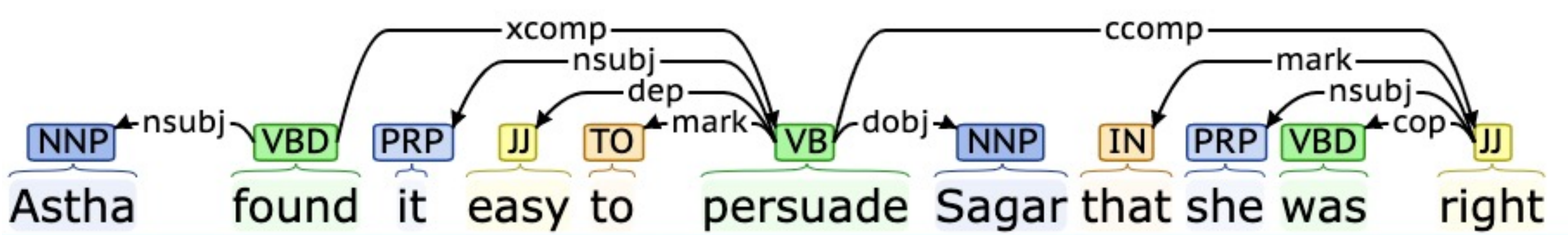
Another Example



Astha found it easy to persuade Sagar that she was right

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

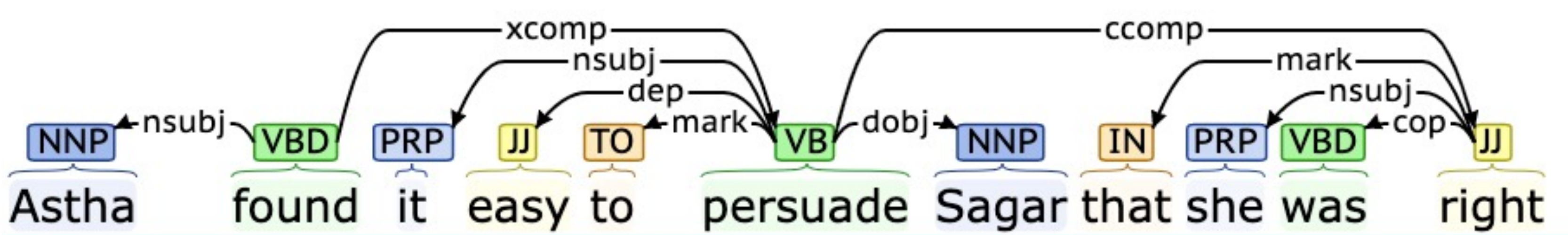
Example: Universal Dependencies

	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

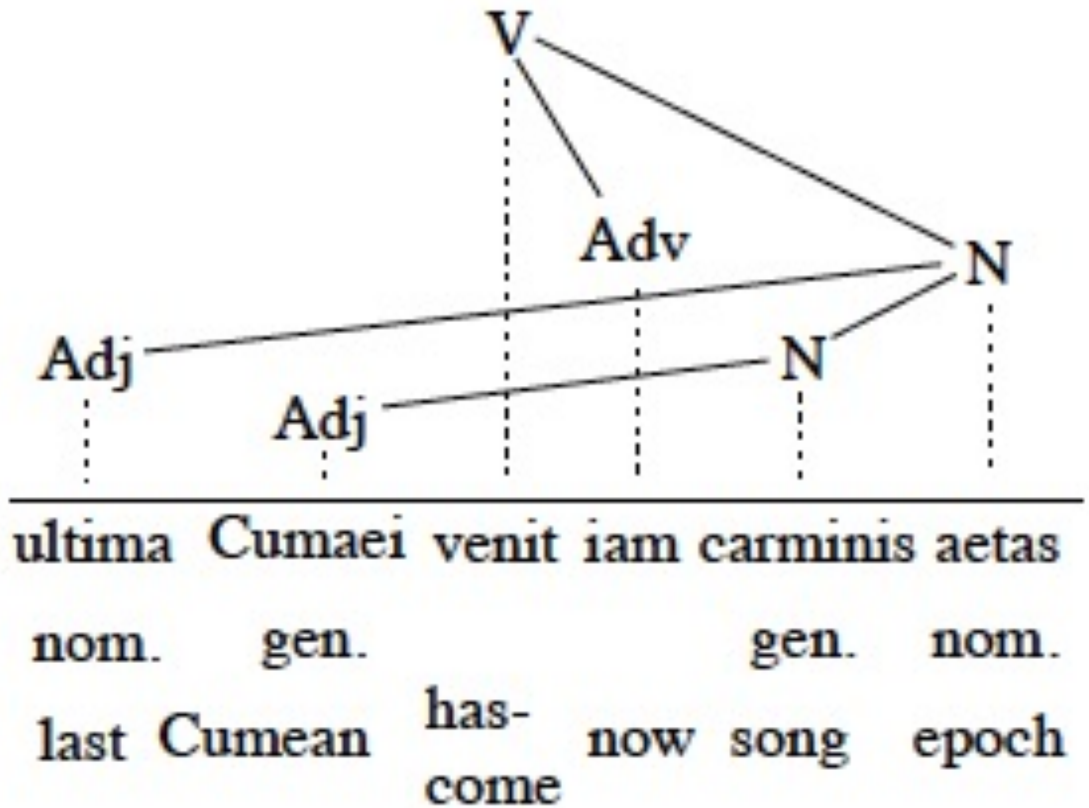
https://universaldependencies.org/u/dep/

Projectivity

Astha found it easy to persuade Sagar that she was right



Projectivity

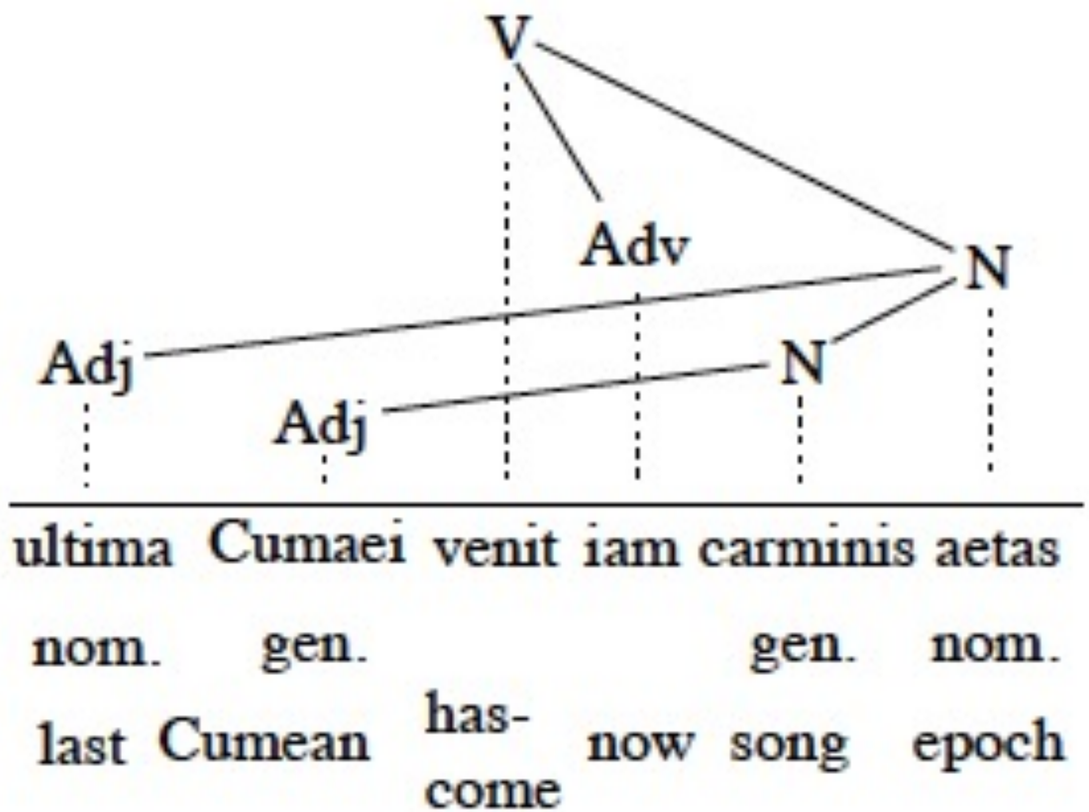


Now has come the last epoch of the Cumean's song.

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
- 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 (*projectivity*)

Projectivity



Now has come the last epoch of the Cumean's song.

Free word order languages

- Dependency structures have been argued to be a better representation than phrase structures for languages with *free word order*
- Free word order languages allow constituents to occur in different linear arrangements, so we would need many different phrase structure rules to capture them all
- For example, German allows dependents of the main verb to occur in many different orders

[$_{NP}$ Ich] lebe [$_{SBAR}$ seitdem wir uns schieden] [$_{PP}$ mit meinem Bruder]
I live since we divorced with my brother

[$_{SBAR}$ Seitdem wir uns schieden] lebe [$_{NP}$ ich] [$_{PP}$ mit meinem Bruder]
[$_{PP}$ Mit meinem Bruder] lebe [$_{NP}$ ich] [$_{SBAR}$ seitdem wir uns schieden]

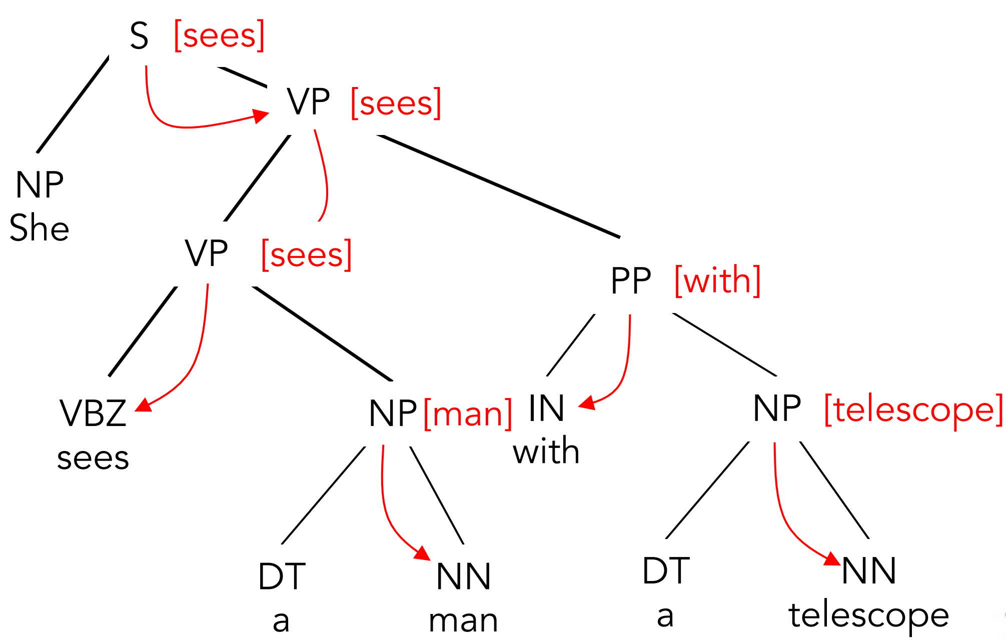
DEPENDENCY STRUCTURES AND PHRASE STRUCTURE

Dependencies and phrase structure

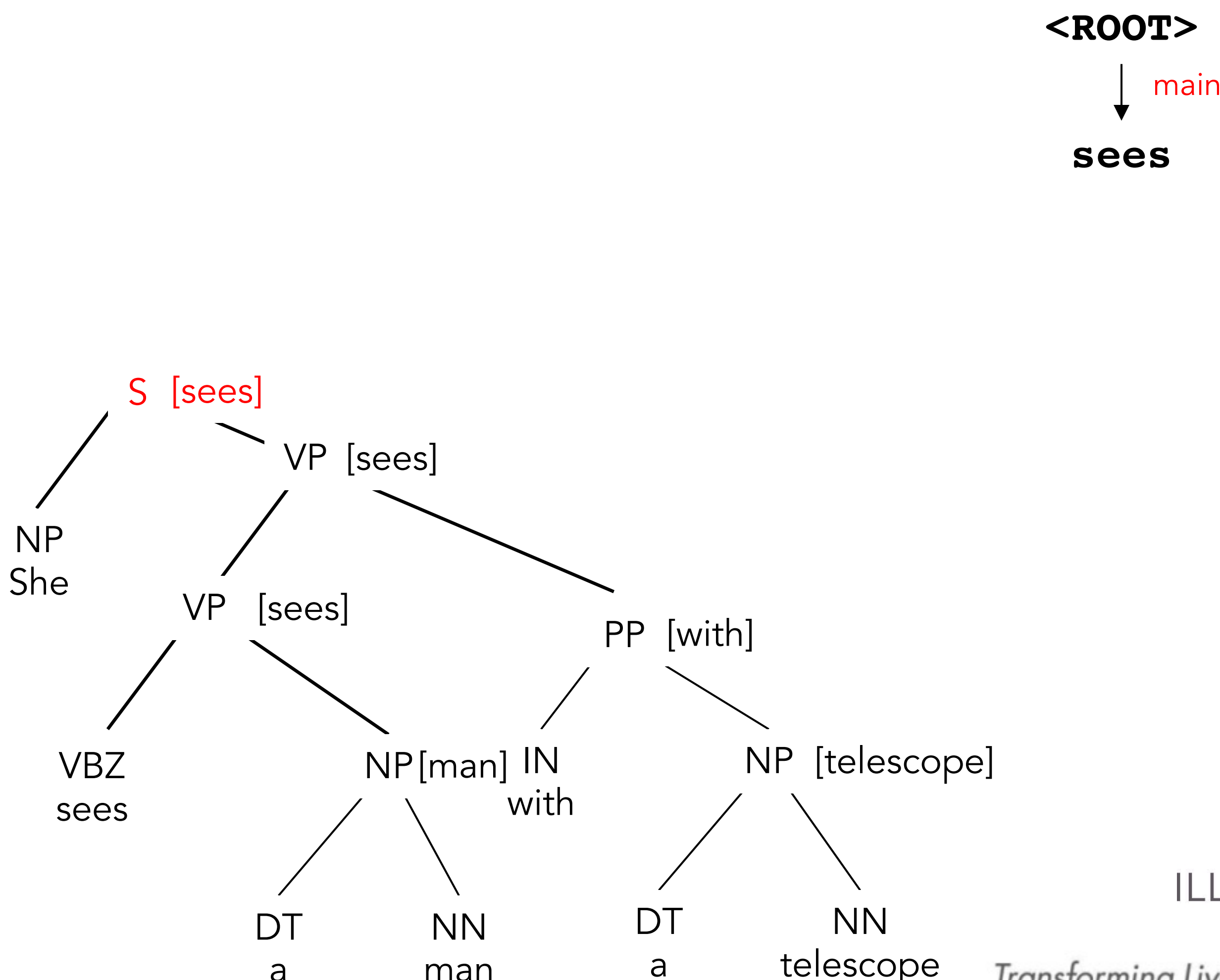
- Dependencies are relationships between heads of syntactic phrases
- We can identify phrasal heads using phrase structure representations
  - They have all the information we need for a dependency graph
  - And more information we don't about the order of words and phrases
- So for projective dependency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

Head identification

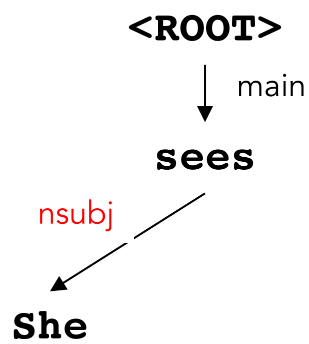
Nonterminal	Direction	Priority
S	right	VP SBAR ADJP UCP NP
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP
NP	right	N* EX $ CD QP PRP …
PP	left	IN TO FW

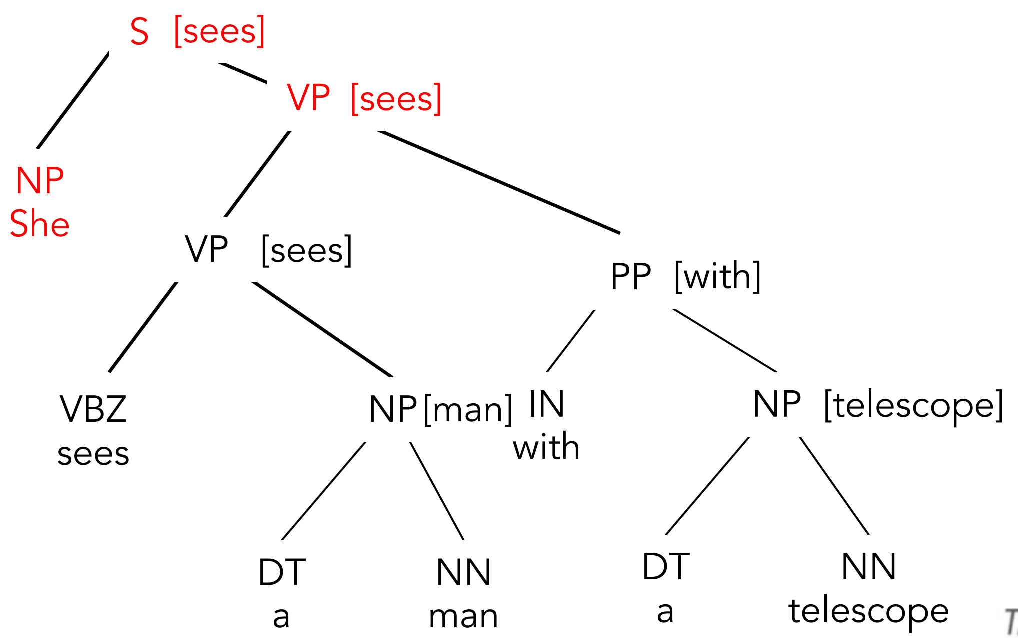


Dependency conversion

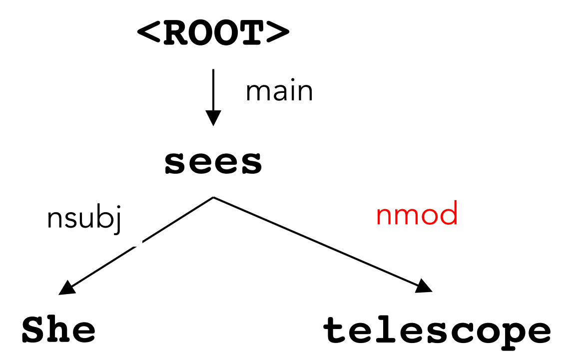


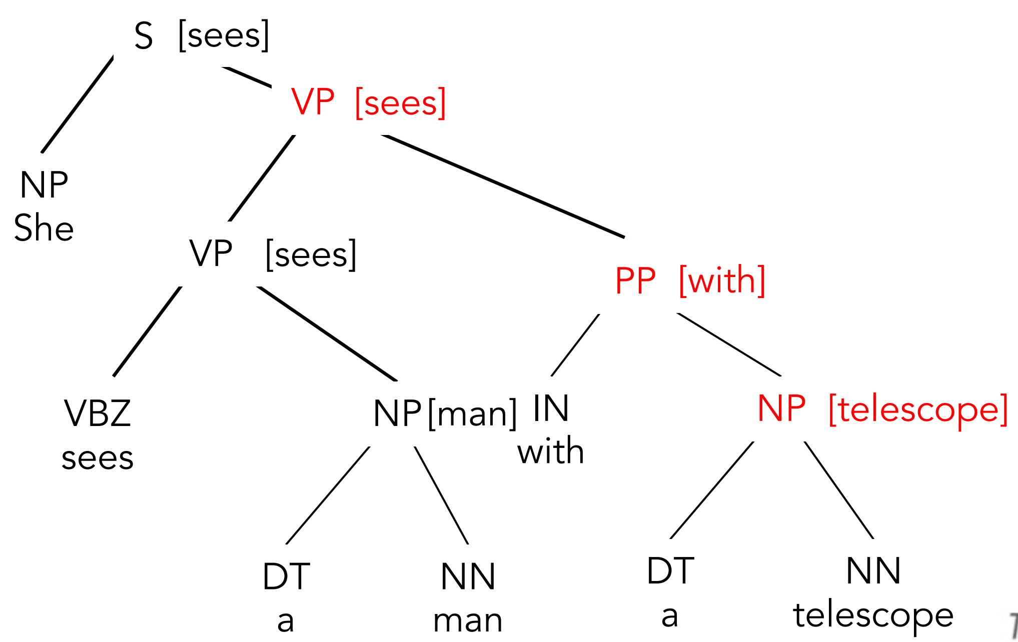
Dependency conversion



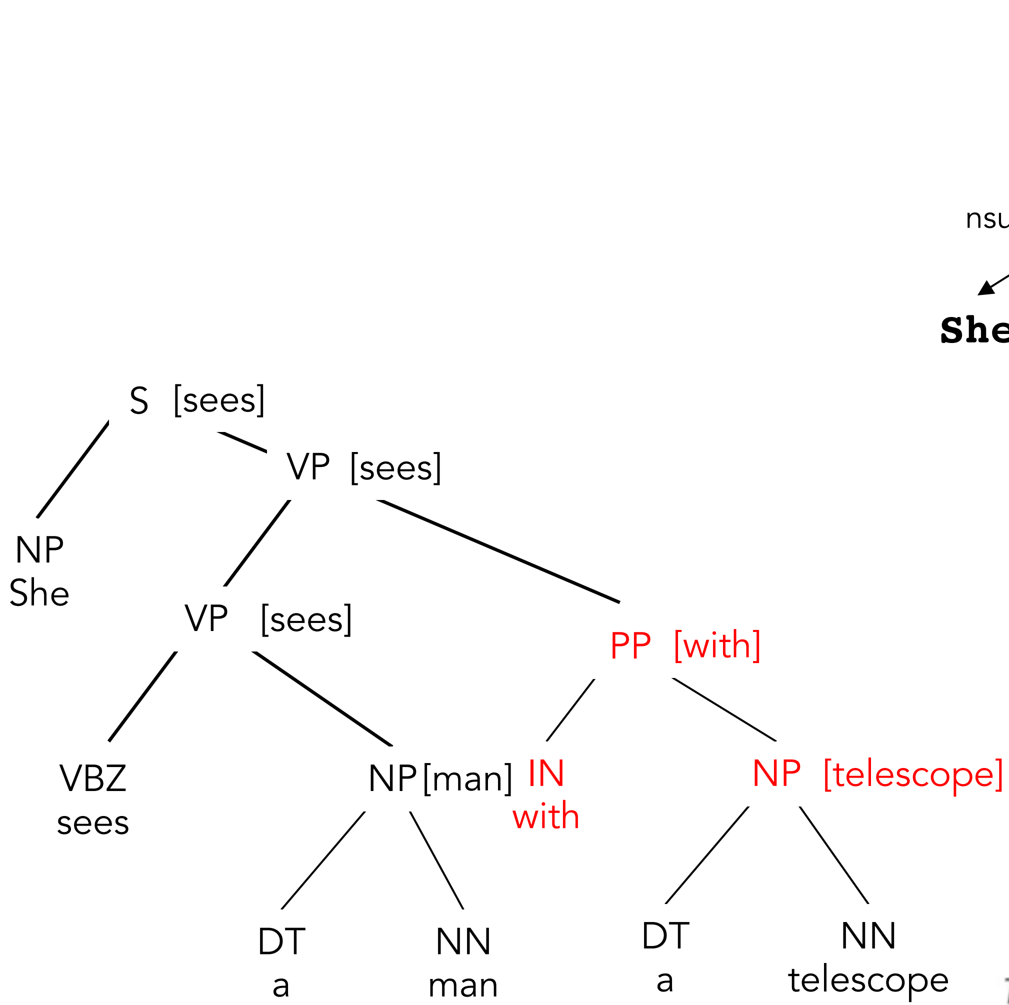


Dependency conversion

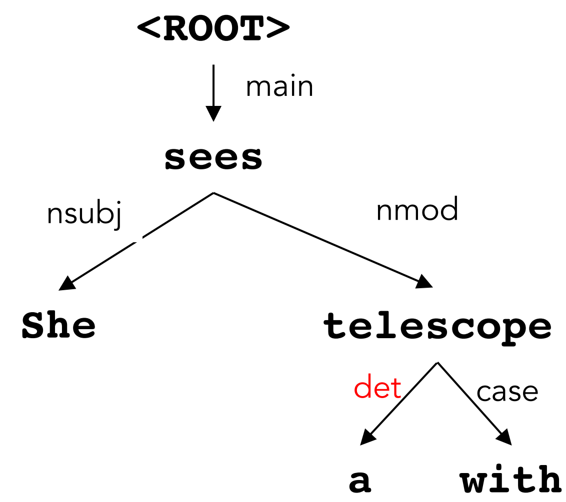


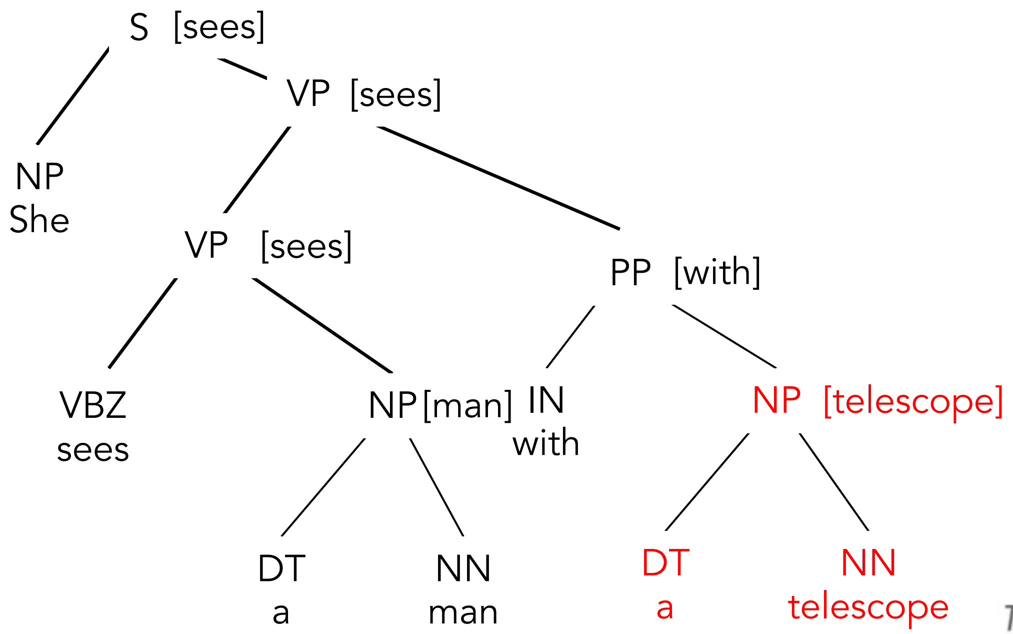


Dependency conversion

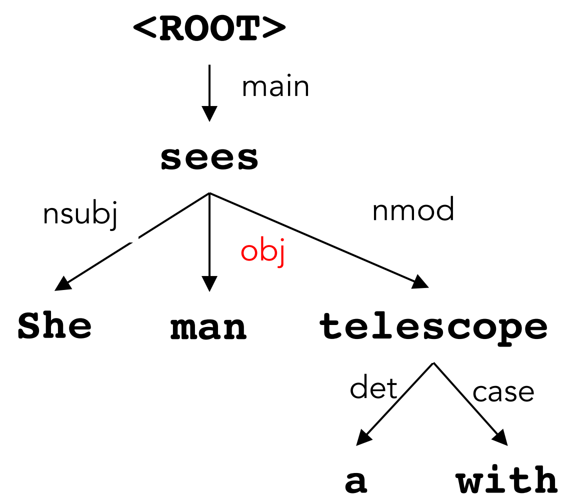


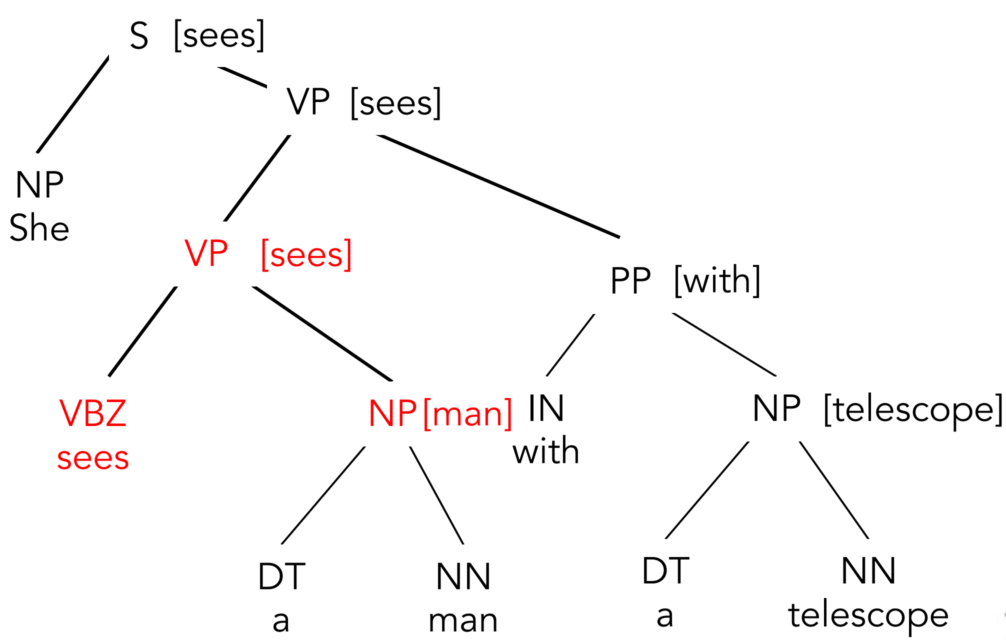
Dependency conversion



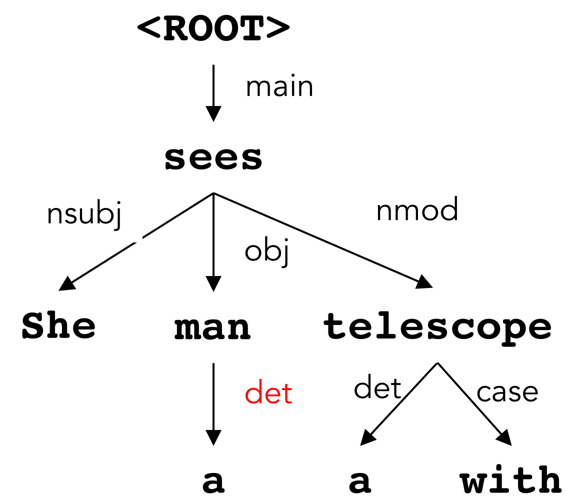


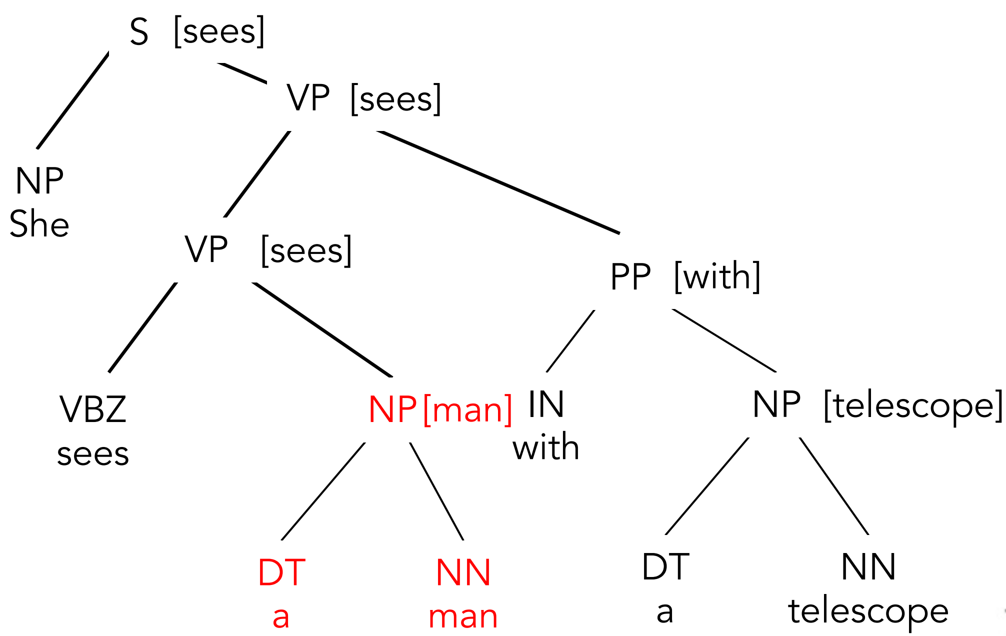
Dependency conversion





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, which fragments the concept to be learned (and related count statistics)
  - Also, doesn't handle non-projective case

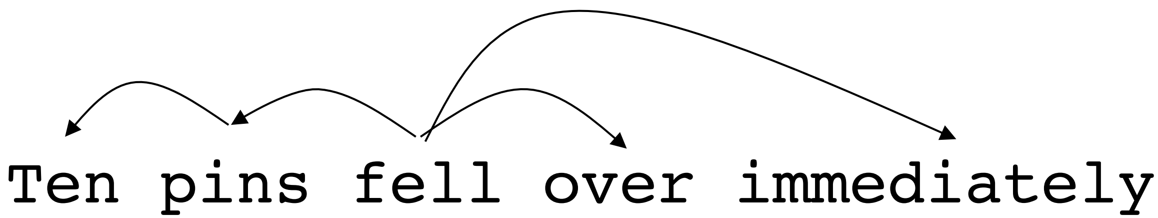
Dependency parsing

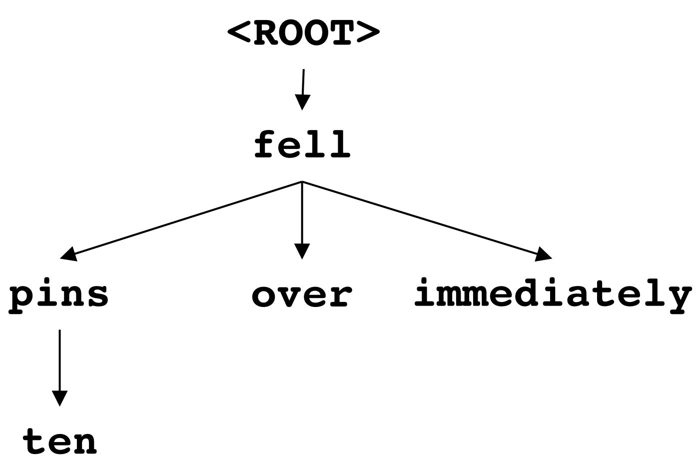
- Instead, we can work in a pure dependency framework. Concepts from PCFGs have dependency grammar analogs
  - Instead of the highest-probability CFG node with label S spanning the sentence, we want to find the Maximum Spanning Tree
  - Decomposition of score for sentence uses similar dynamic programming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





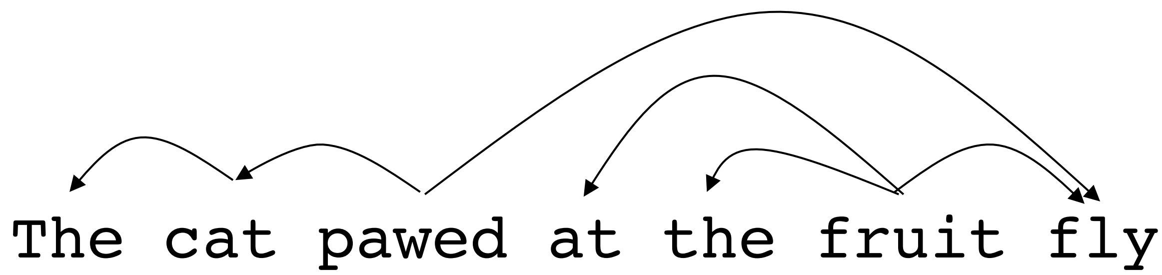
Spanning tree?

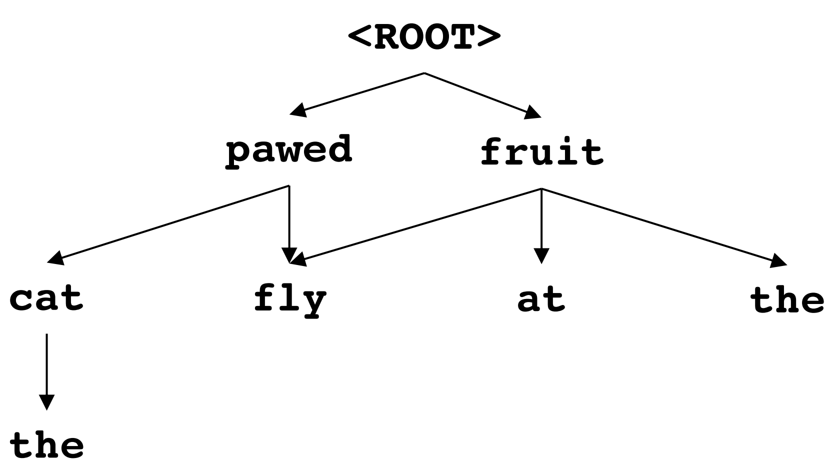
Yes

Projective?

Yes

Spanning Trees

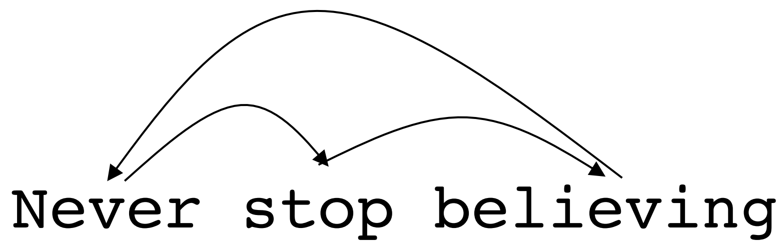


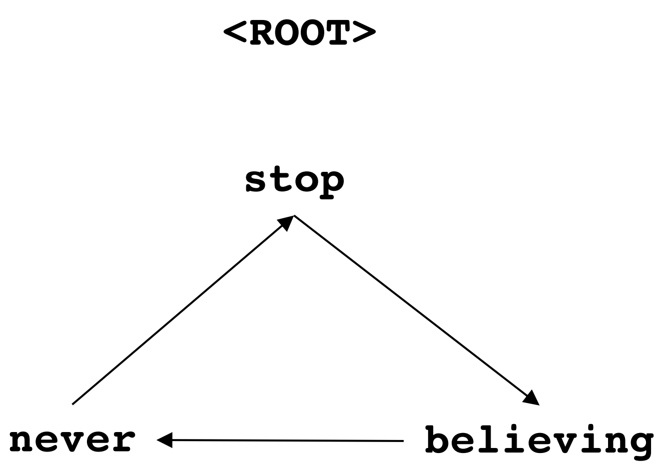


Spanning tree?

No

Spanning Trees

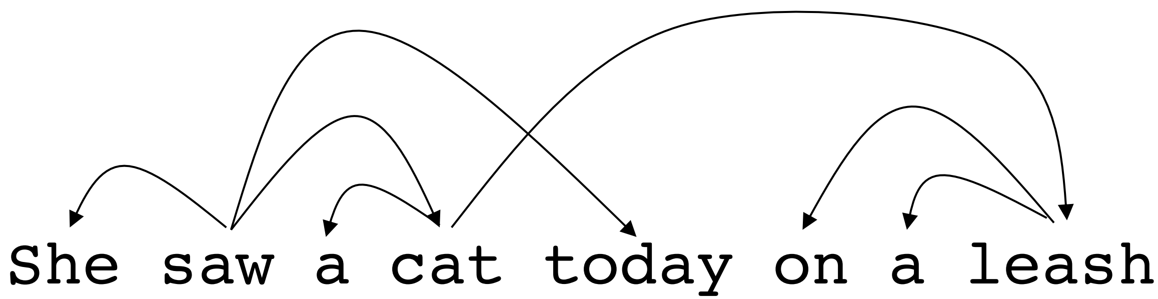


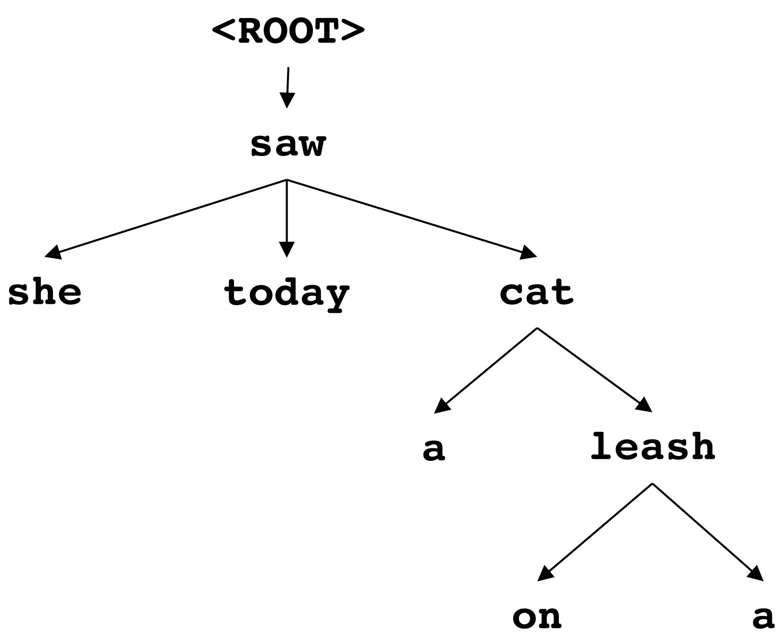


Spanning tree?

No

Spanning Trees





Spanning tree?

Yes

Projective?

No

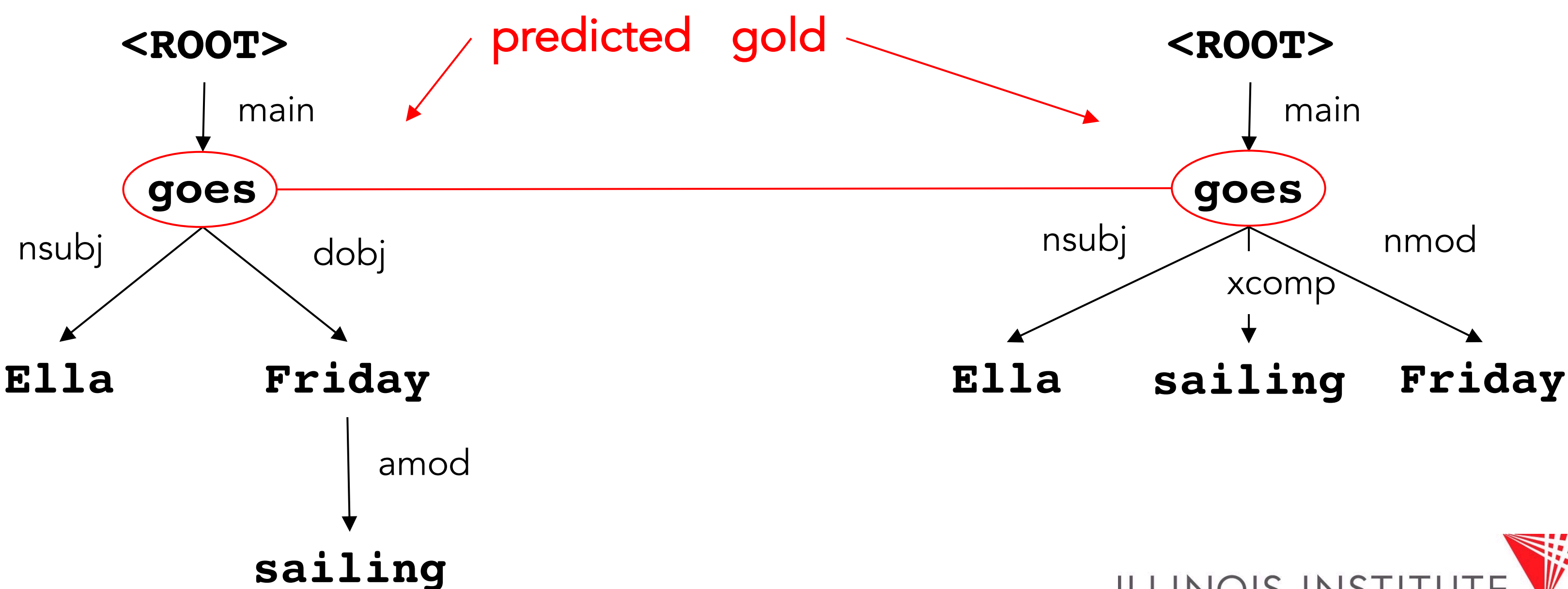
Dependency parsing algorithms

- 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

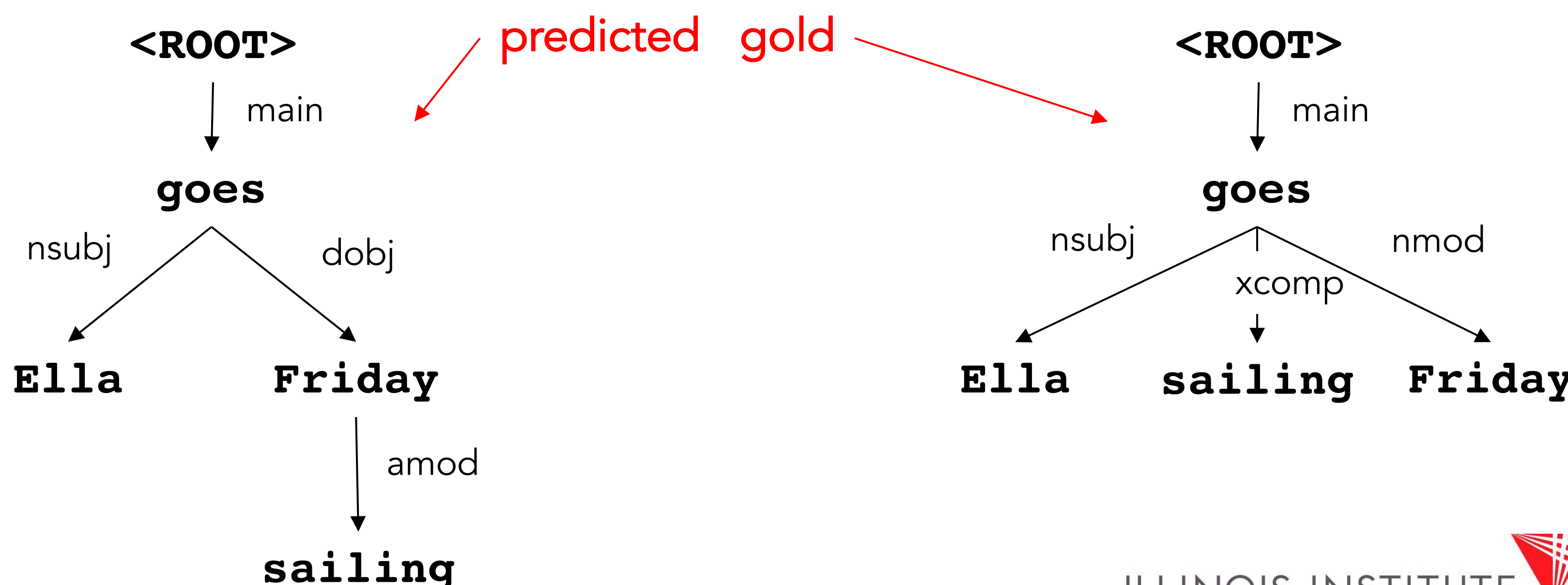
Evaluation measures for dependency parsing

- **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%



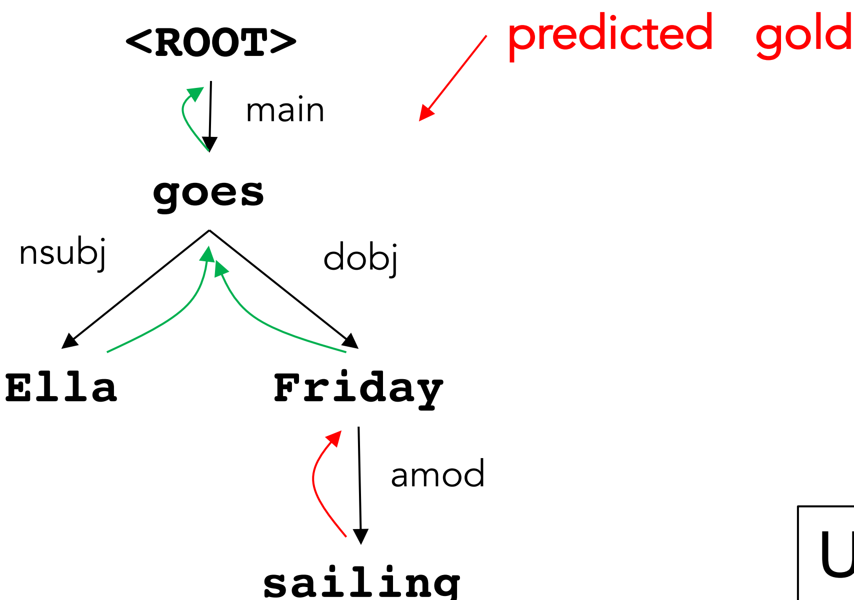
Evaluation measures for dependency parsing

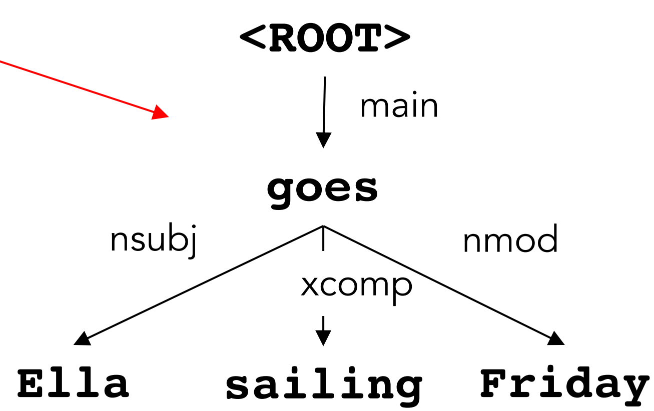
- **Precision / Recall / F-measure per dependency relation** – e.g., here we have 100% precision on `nsubj`, 0% precision on `dobj`



Evaluation measures for dependency parsing

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

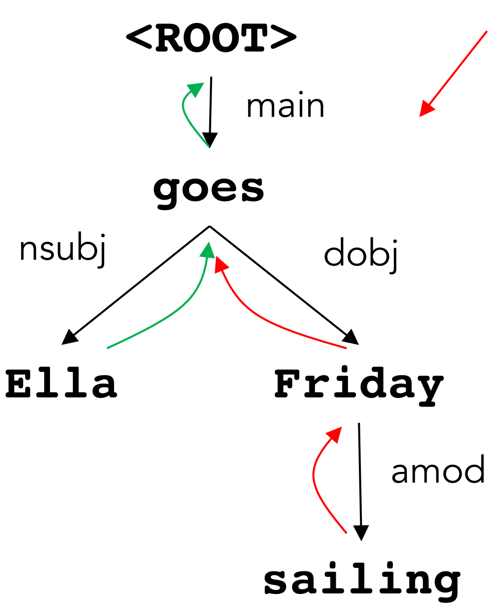


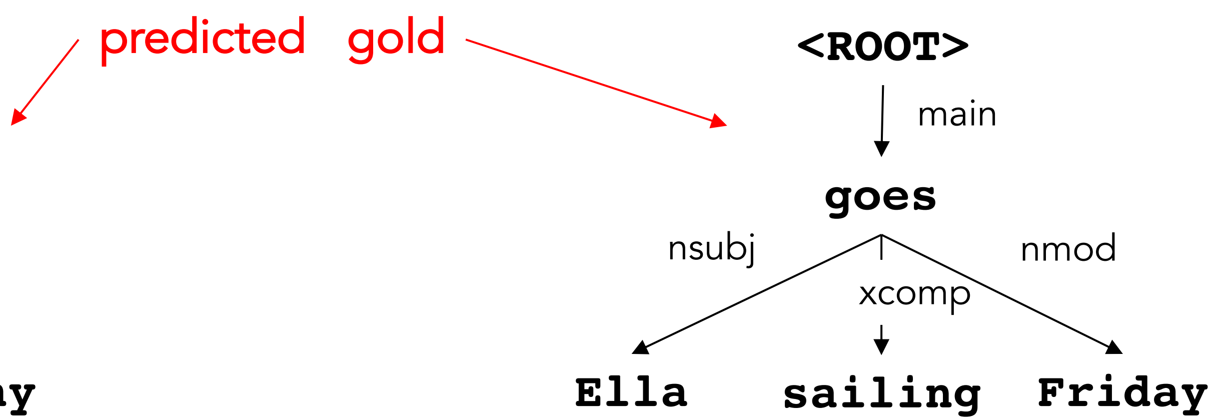


Unlabeled attachment score = ¾ = 75%

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%

CoNLL Evaluation

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Tot	SD	Bu
McD	**66.9**	85.9	**80.2**	**84.8**	**79.2**	**87.3**	90.7	**86.8**	**73.4**	**82.3**	82.6	63.2	**80.3**	8.4	**87.6**
Niv	**66.7**	86.9	78.4	**84.8**	**78.6**	85.8	**91.7**	**87.6**	70.3	**81.3**	**84.6**	**65.7**	**80.2**	8.5	**87.4**
O'N	**66.7**	86.7	76.6	82.8	77.5	85.4	90.6	84.7	71.1	79.8	81.8	57.5	78.4	9.4	85.2
Rie	**66.7**	**90.0**	67.4	83.6	**78.6**	86.2	90.5	84.4	71.2	77.4	80.7	58.6	77.9	10.1	0.0
Sag	62.7	84.7	75.2	81.6	76.6	84.9	90.4	**86.0**	69.1	77.7	82.0	63.2	77.8	9.0	0.0
Che	65.2	84.3	76.2	81.7	71.8	84.1	89.9	85.1	71.4	80.5	81.1	61.2	77.7	8.7	86.3
Cor	63.5	79.9	74.5	81.7	71.4	83.5	90.0	84.6	**72.4**	80.4	79.7	61.7	76.9	8.5	83.4
Cha	60.9	85.1	72.9	80.6	72.9	84.2	89.1	84.0	69.5	79.7	82.3	60.5	76.8	9.4	0.0
Joh	64.3	72.5	71.5	81.5	72.7	80.4	85.6	84.6	66.4	78.2	78.1	63.4	74.9	7.7	0.0
Car	60.9	83.7	68.8	79.7	67.3	82.4	88.1	83.4	68.4	77.2	78.7	58.1	74.7	9.7	83.3
Wu	63.8	74.8	59.4	78.4	68.5	76.5	90.1	81.5	67.8	73.0	71.7	55.1	71.7	9.7	79.7
Can	57.6	78.4	60.9	77.9	74.6	77.6	87.4	77.4	59.2	68.3	79.2	51.1	70.8	11.1	78.7
Bic	55.4	76.2	63.0	74.6	69.5	74.7	84.8	78.2	64.3	71.4	74.1	53.9	70.0	9.3	79.2
Dre	53.4	71.6	60.5	66.6	61.6	71.0	82.9	75.3	58.7	67.6	67.6	46.1	65.2	9.9	74.8
Yur	52.4	72.7	51.9	71.6	62.8	63.8	84.4	70.4	55.1	69.6	65.2	60.3	65.0	9.5	73.5
Liu	50.7	75.3	58.5	77.7	59.4	68.1	70.8	71.1	57.2	65.1	63.8	41.7	63.3	10.4	67.6
Sch	44.4	66.2	53.3	76.1	72.1	68.7	83.4	71.0	50.7	47.0	71.1	49.8	62.8	13.0	0.0
Att	53.8	54.9	59.8	66.4	58.2	69.8	65.4	75.4	57.2	67.4	68.8	37.8	[a]61.2	9.9	72.9
Shi	62.8	0.0	0.0	75.8	0.0	0.0	0.0	0.0	64.6	73.2	79.5	54.2	34.2	36.3	0.0
Av	59.9	78.3	67.2	78.3	70.7	78.6	85.9	80.6	65.2	73.5	76.4	56.0			80.0
SD	6.5	8.8	8.9	5.5	6.7	7.5	7.1	5.8	6.8	8.4	6.5	7.7			6.3

Table 5: Labeled attachment scores of parsers on the 13 test sets.

Labeled attachment scores for participants in a multilingual dependency parsing shared task.

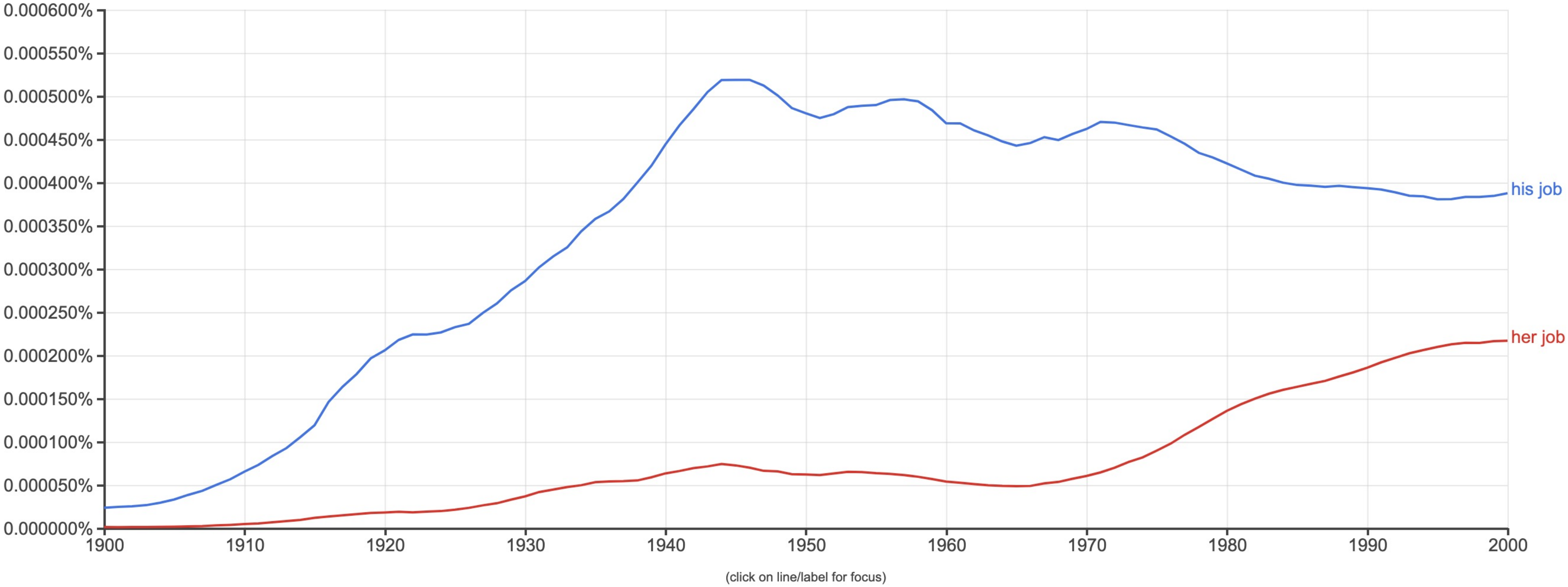
The best parser used a projective maximum spanning tree algorithm

APPLICATIONS

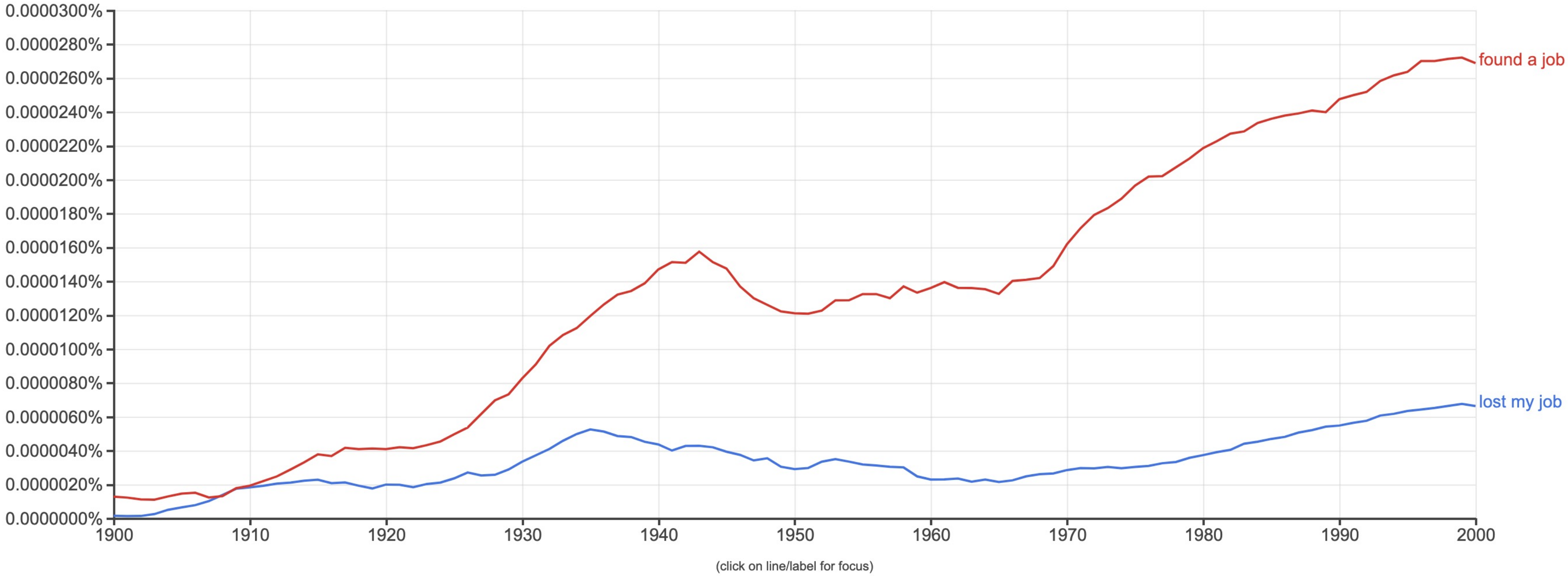
Generalized ngram search

- 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)

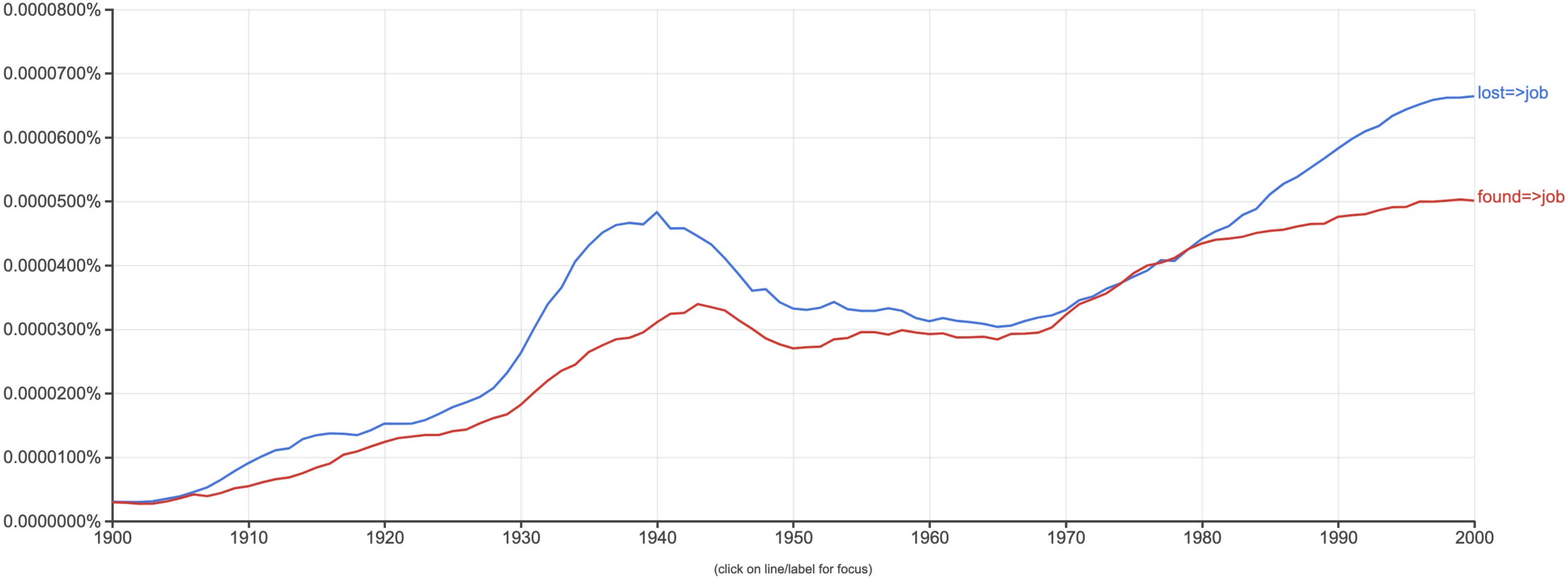
Generalized ngram search



Generalized ngram search



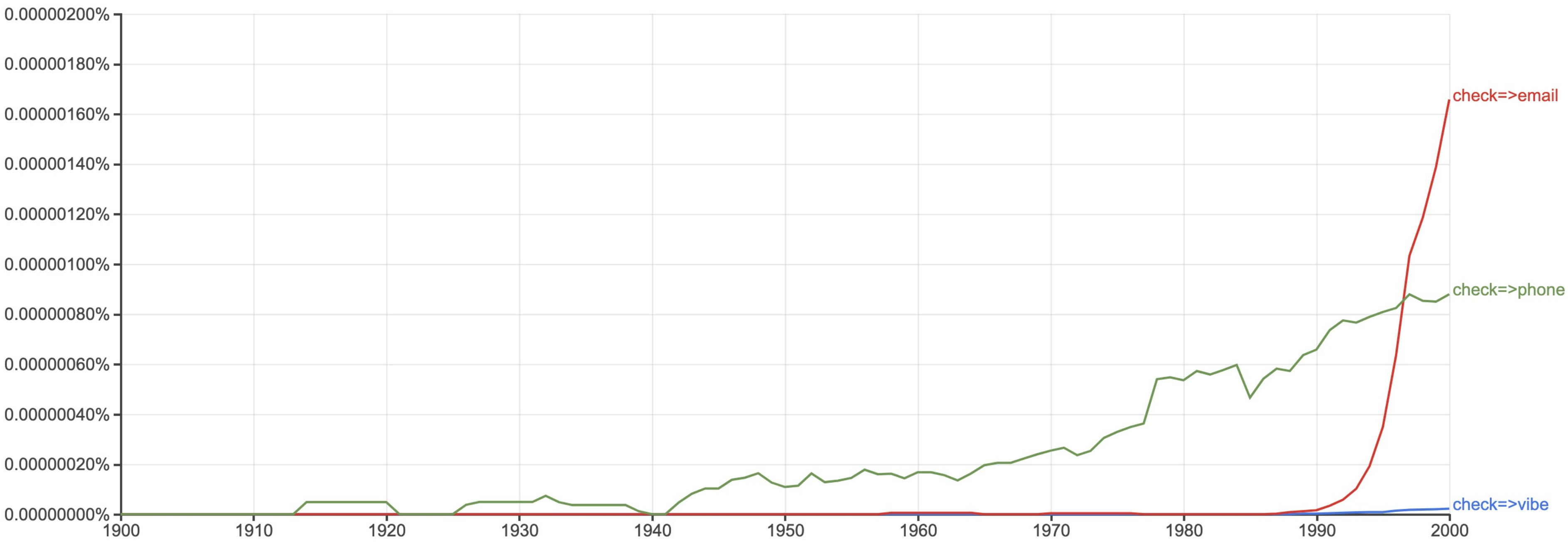
Generalized ngram search



Generalized ngram search

Graph these comma-separated phrases: check=>vibe,check=>email,check=>phone case-insensitive

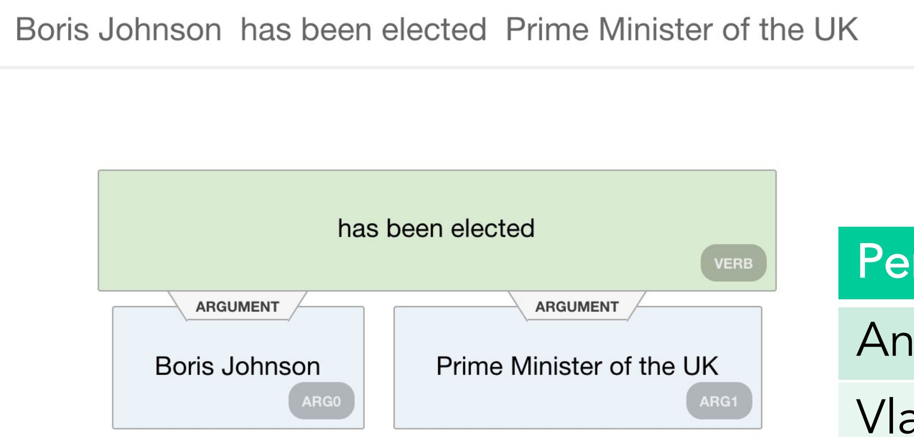
between 1900 and 2000 from the corpus English with smoothing of 3. Search lots of books

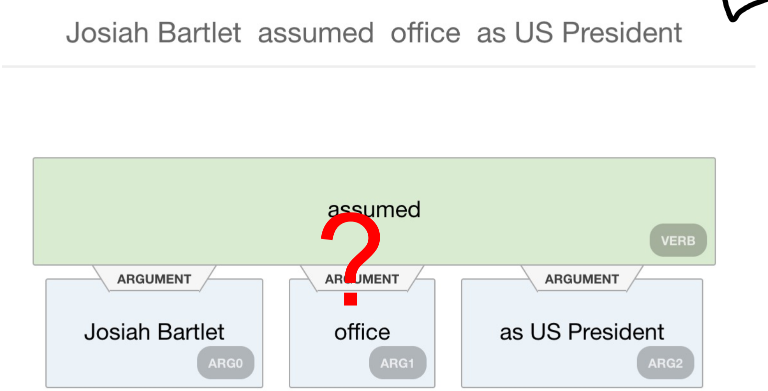


Relation extraction

- 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
- Specifically, *open information extraction* involves crawling open-domain texts to identify facts that can be used to populate a knowledge base (automatically or semi-automatically)
  - Dependency relations between heads can indicate consistent semantic relations

Open information extraction





Person	Role
Angela Merkel	German PM
Vladimir Putin	Russian president
Justin Trudeau	Canadian Prime Minister
Boris Johnson	Prime Minister of the UK
Xi Jinping	President of PRC
Narendra Modi	Prime Minister of India
…	…

https://demo.allennlp.org/open-information-extraction