CS378: Natural Language Processing

Lecture 19: Trees

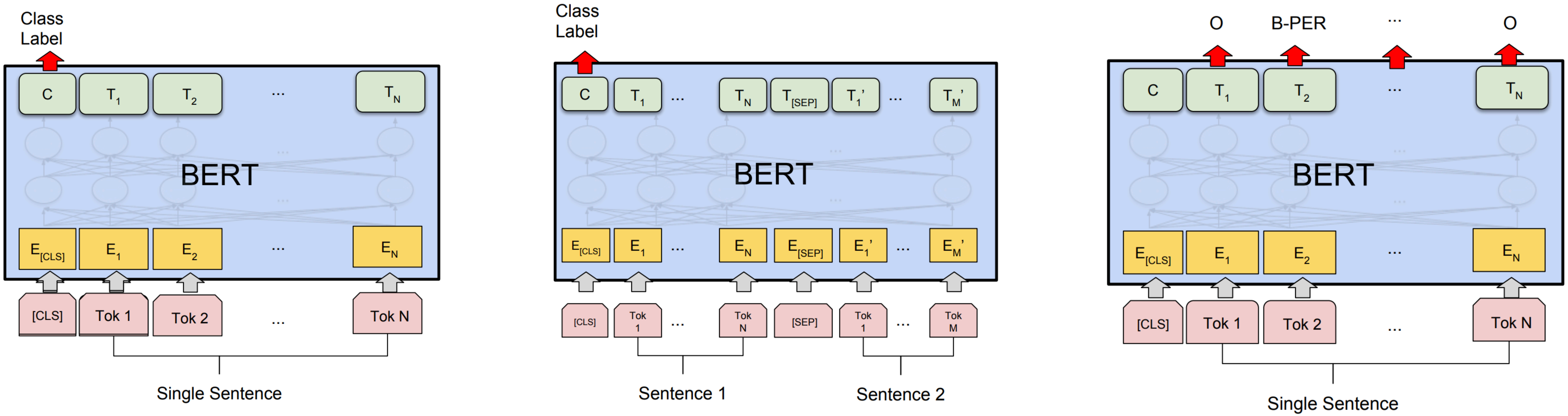
The University of Texas at Austin

Eunsol Choi

Some slides adapted from Greg Durrett, Yoav Artzi, Yejin Choi, Michael Collins, Princeton NLP

Timeline of Pretrained LM

Recap: BERT



(b) Single Sentence Classification Tasks: SST-2, CoLA

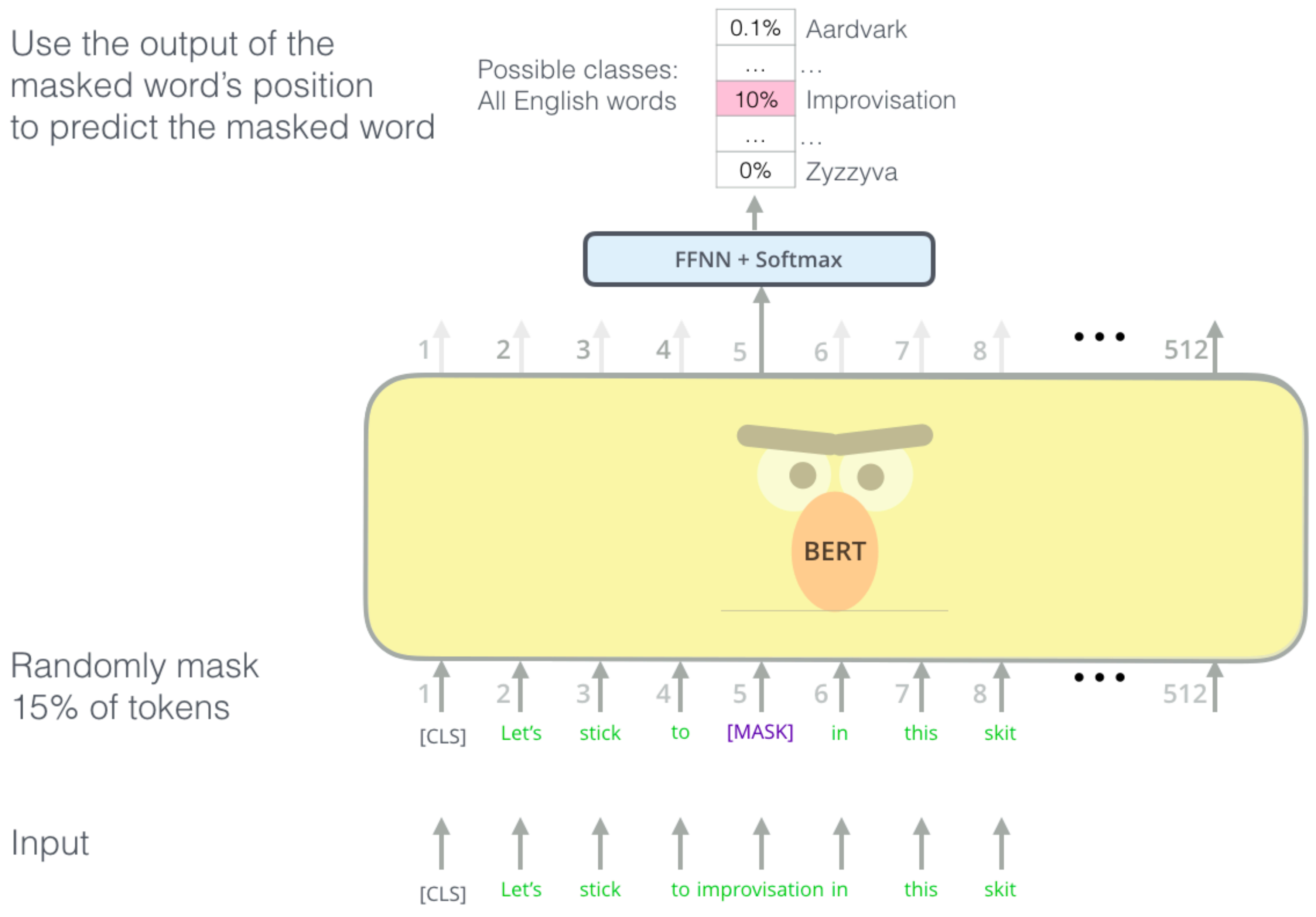
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

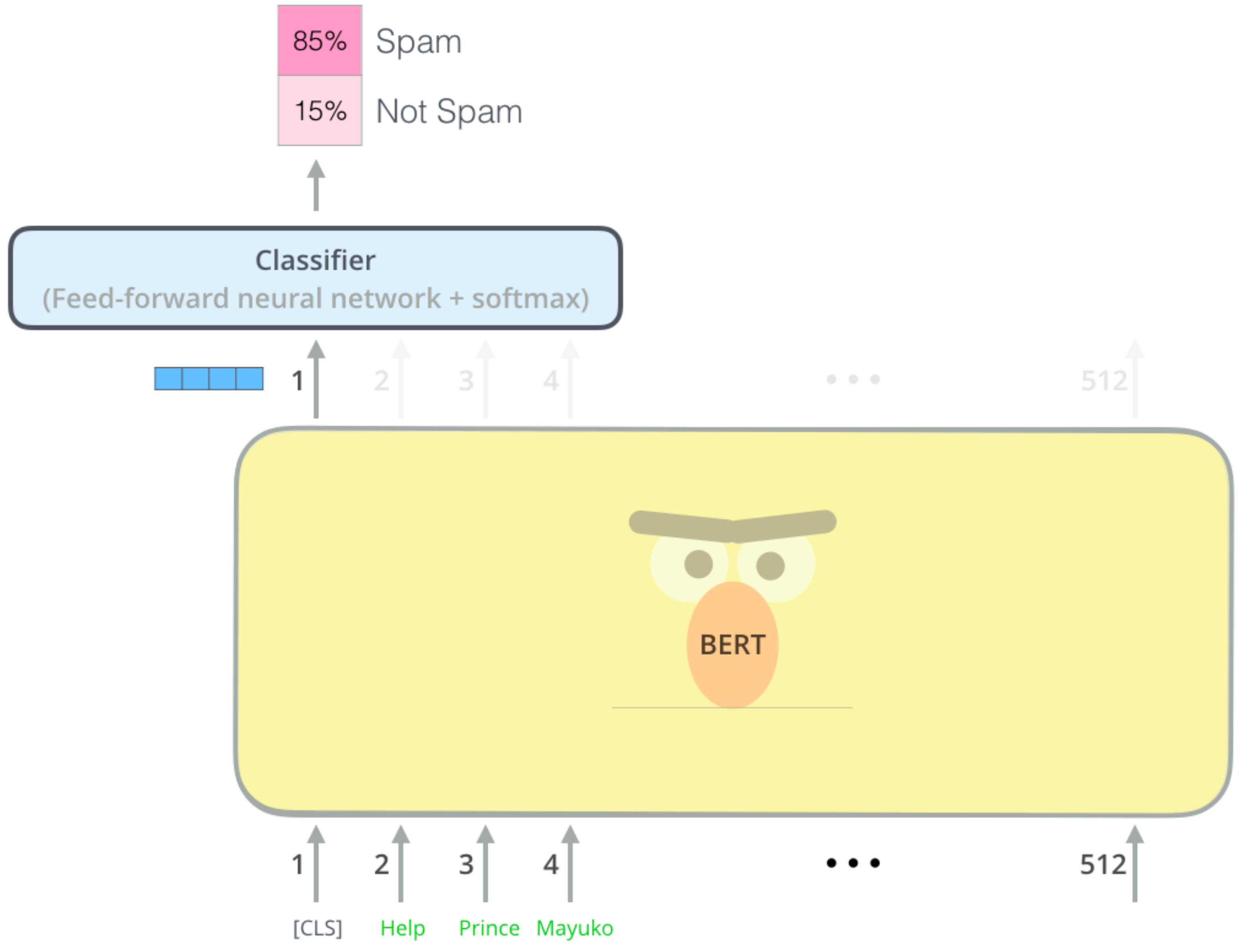
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Artificial [CLS] token is used as the vector to do classification from
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

Pretraining and fine-tuning





Pre-training

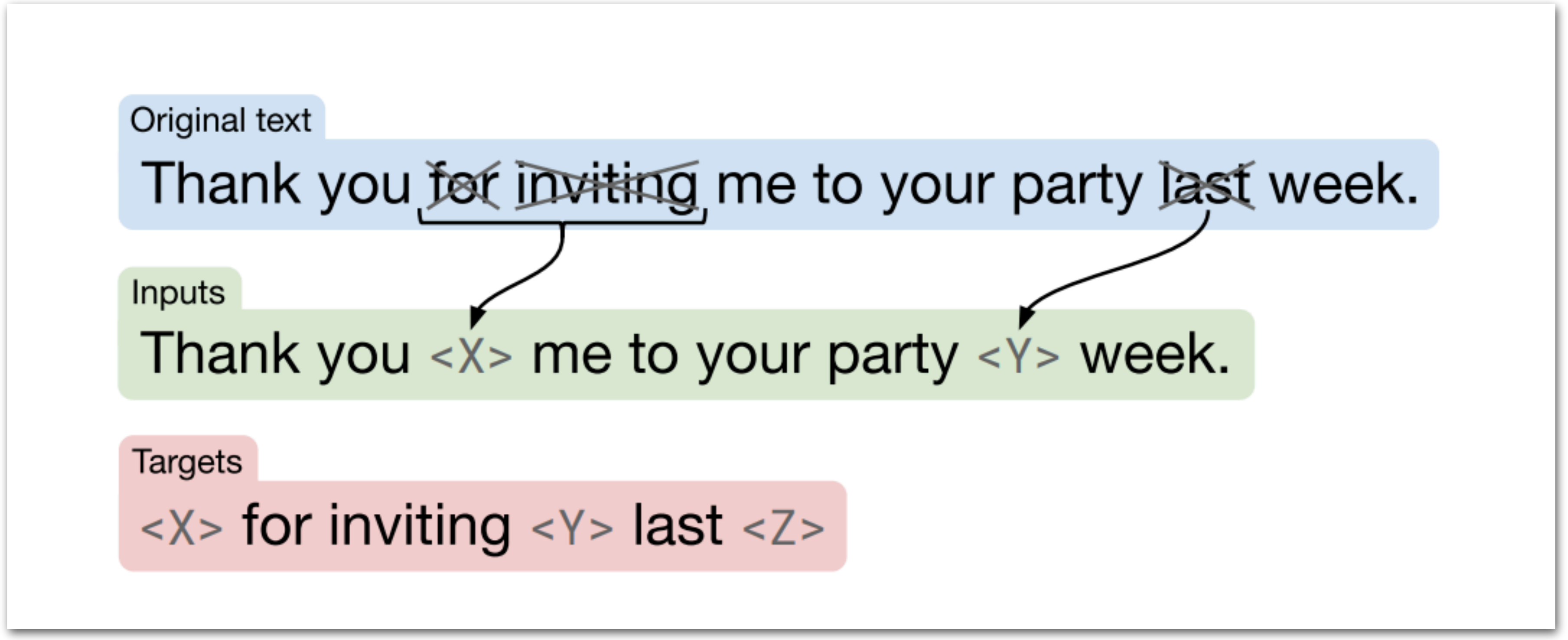
Fine-tuning

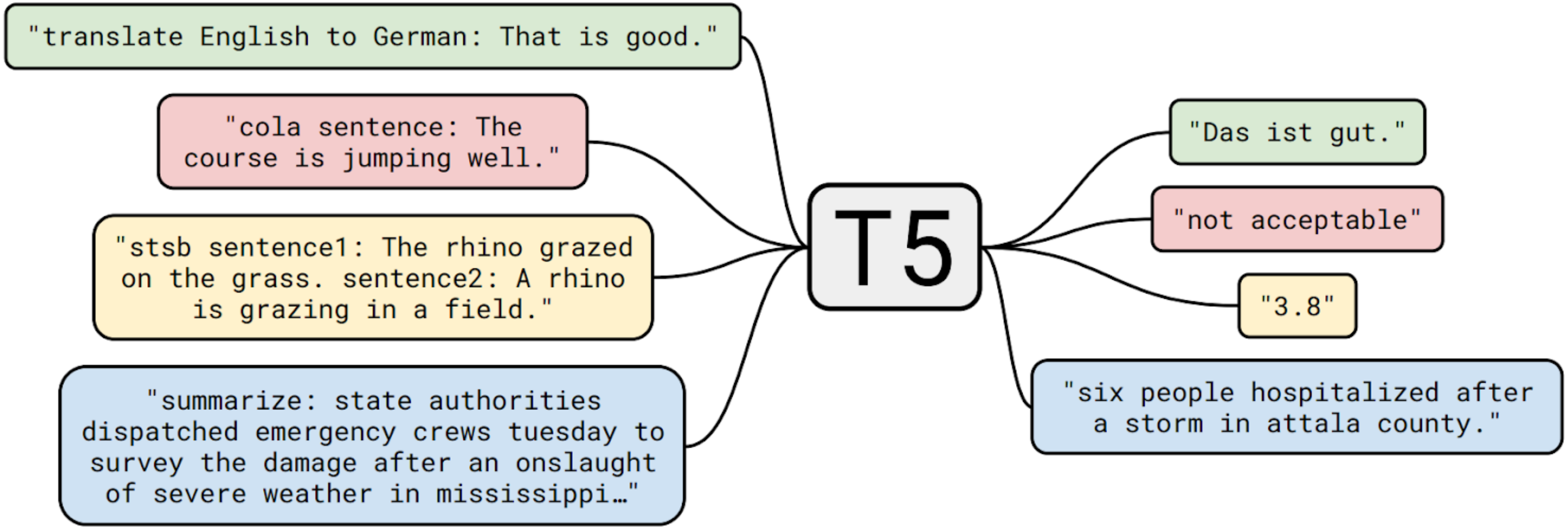
language modeling objective

task-specific objective

T5: Text-to-Text Transfer Transformer

- Also train on multiple supervised tasks
- Similar backbone architecture





Logistics

- Reading
  - Dependency parsing : J&M chapter 14
  - Constituency parsing: J&M chapter 13

- Exam
  - Recap session next Tuesday
  - Exam Thursday *in person*, different room

This Lecture

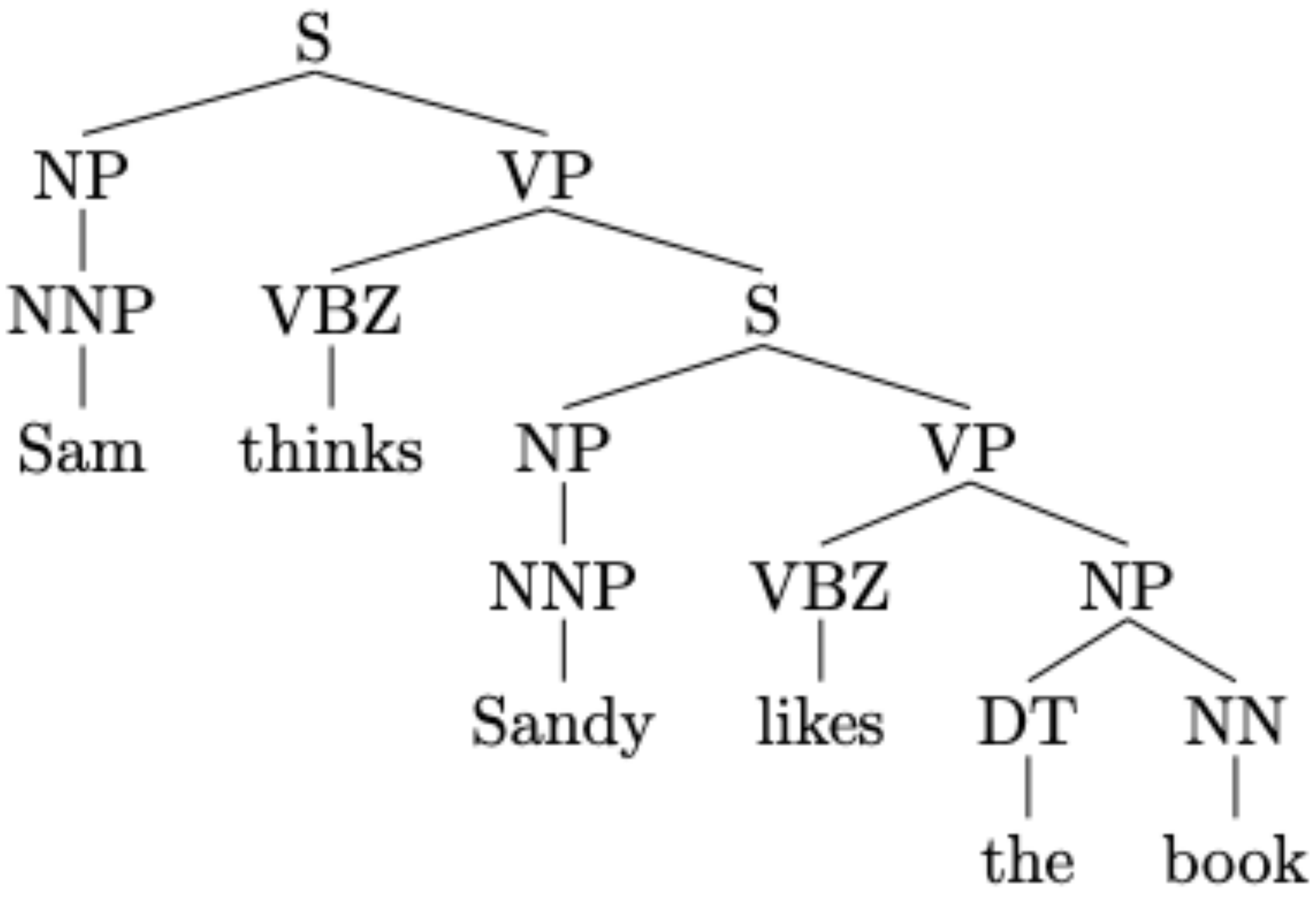
- Overview of syntax

- Dependency Parsing [Today]

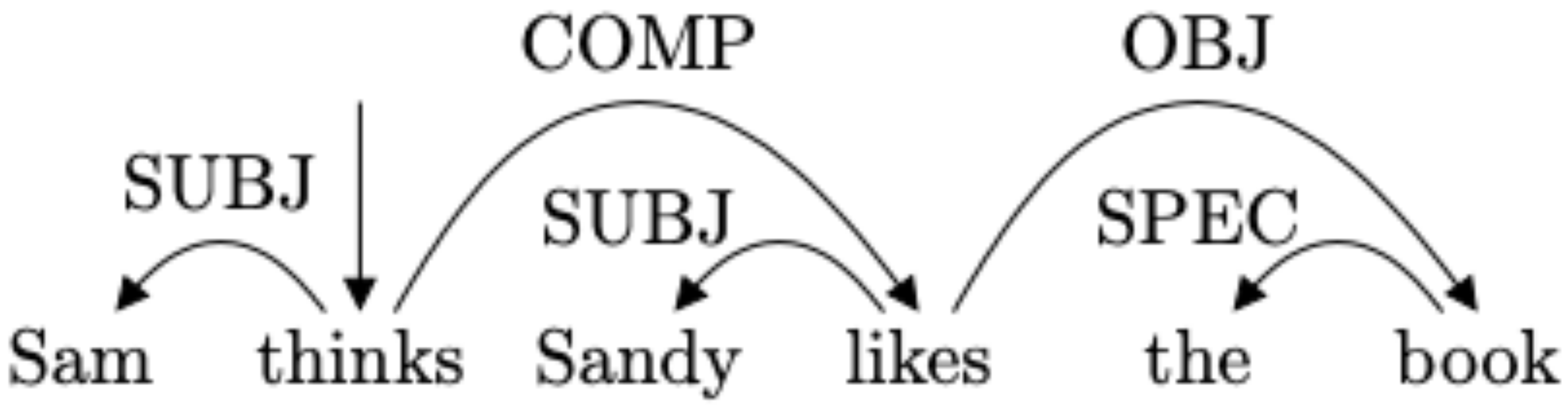
- Constituency Parsing [Thursday]

Syntax

- Study of word order and how words form sentences
  - Constituency Parsing
  - Dependency Parsing

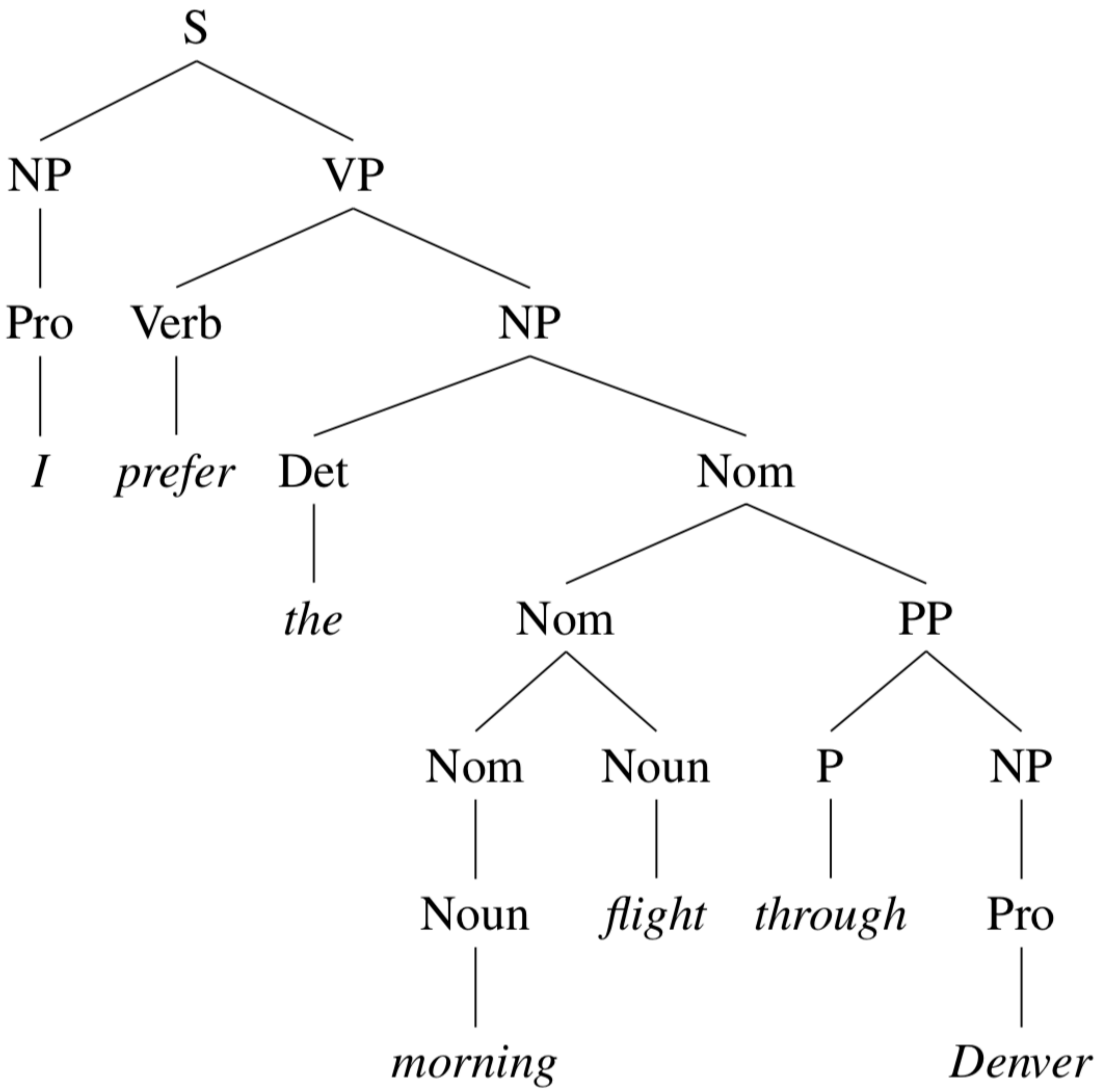


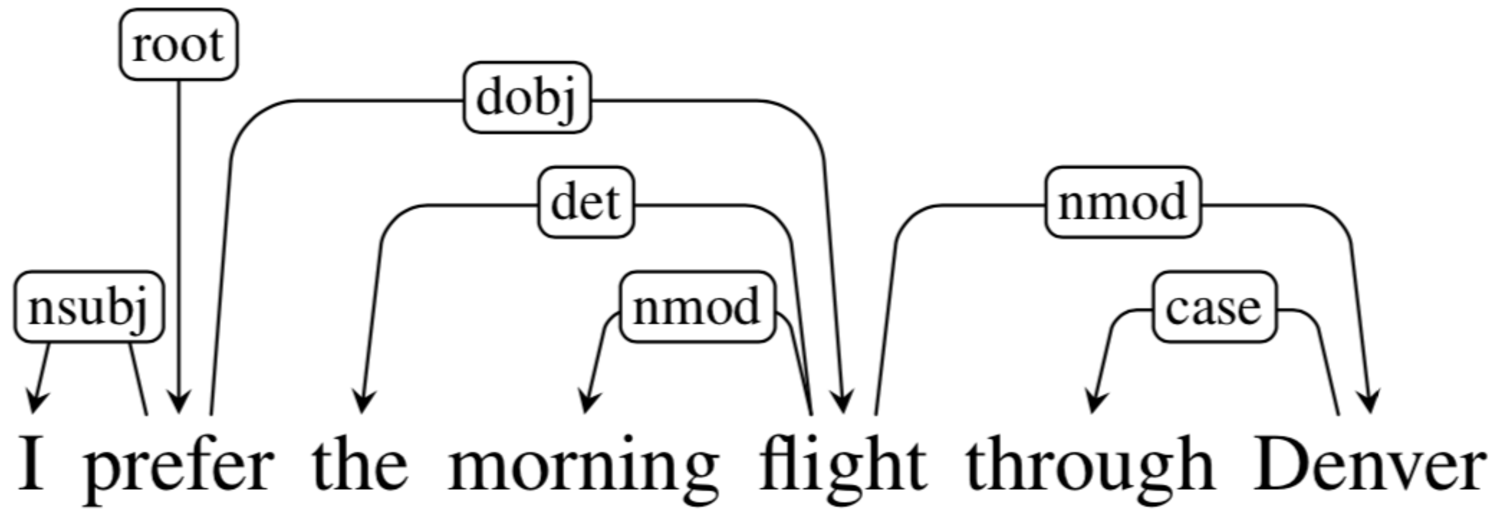
Words organized as nested structure of constituents

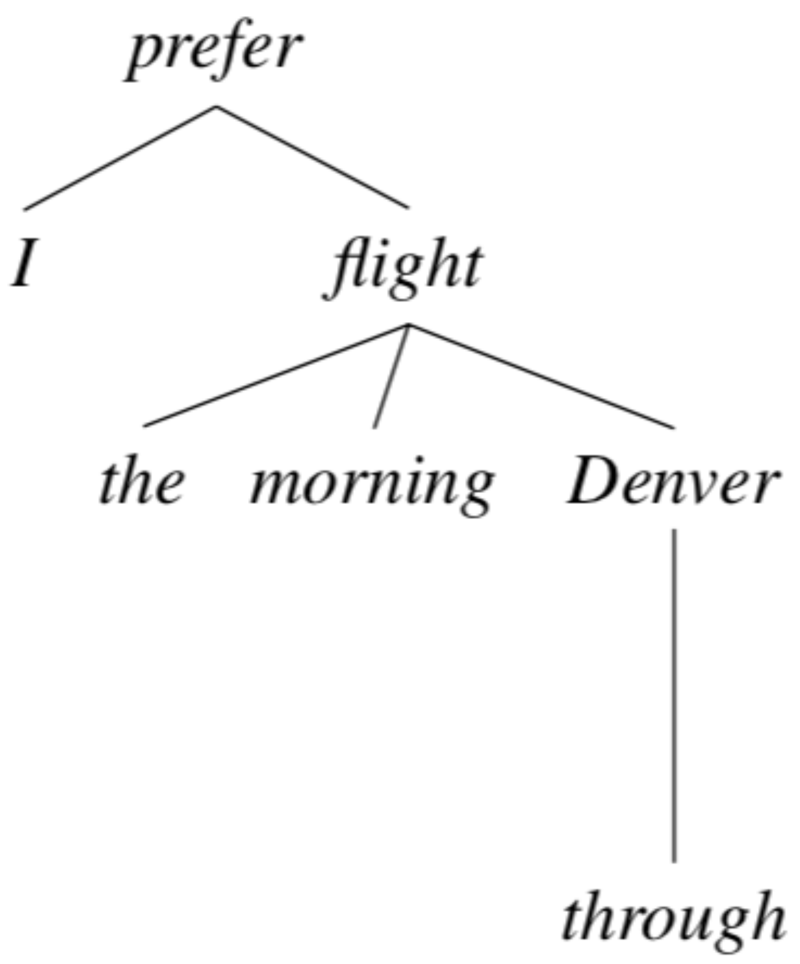


which words depend on (modify or are arguments of) which other words.

Constituency vs. Dependency







Why do we care about syntax?

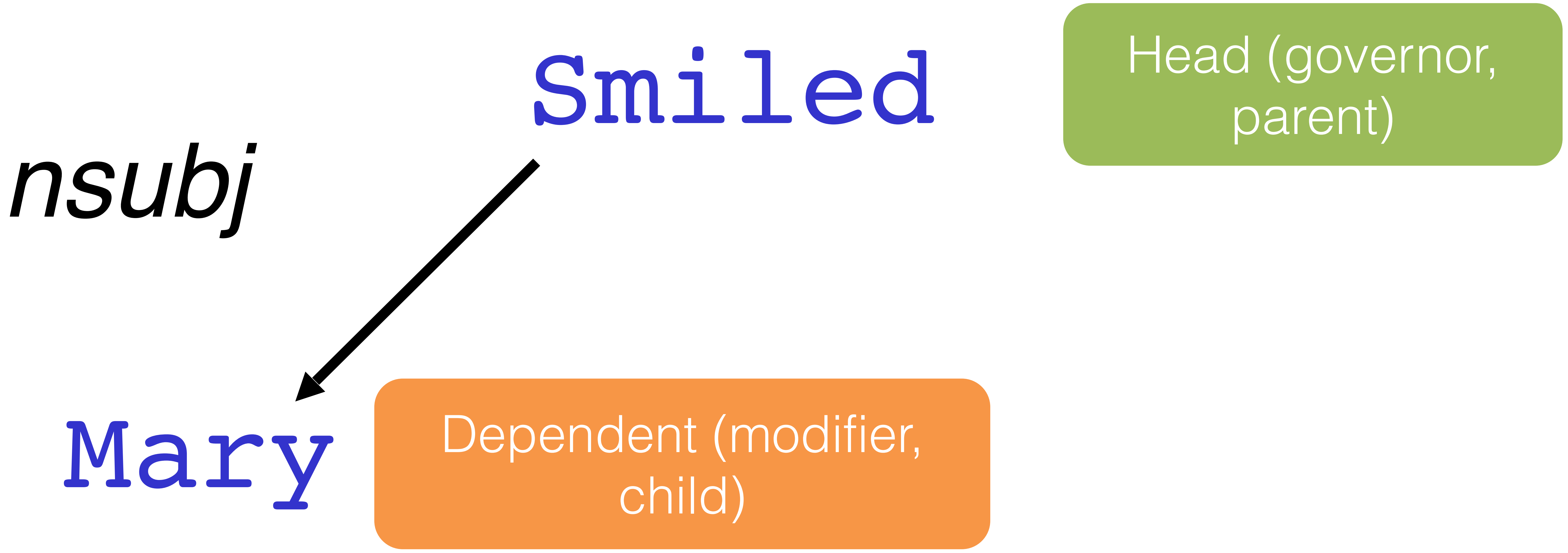
- Clarifies the ambiguities in language
  - Recognize verb-argument structures (who is doing what to whom?)
  - Recognize modifier scopes (e.g., plastic cup holder)
  - Coordination scope (e.g., small rats and mice)
- Provide higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV
- Sometimes can be used to help downstream tasks

This Lecture

- Overview
- Dependency Parsing [Today]
- Constituency Parsing [Thursday]

Dependency Parse

- Binary asymmetric relation between words

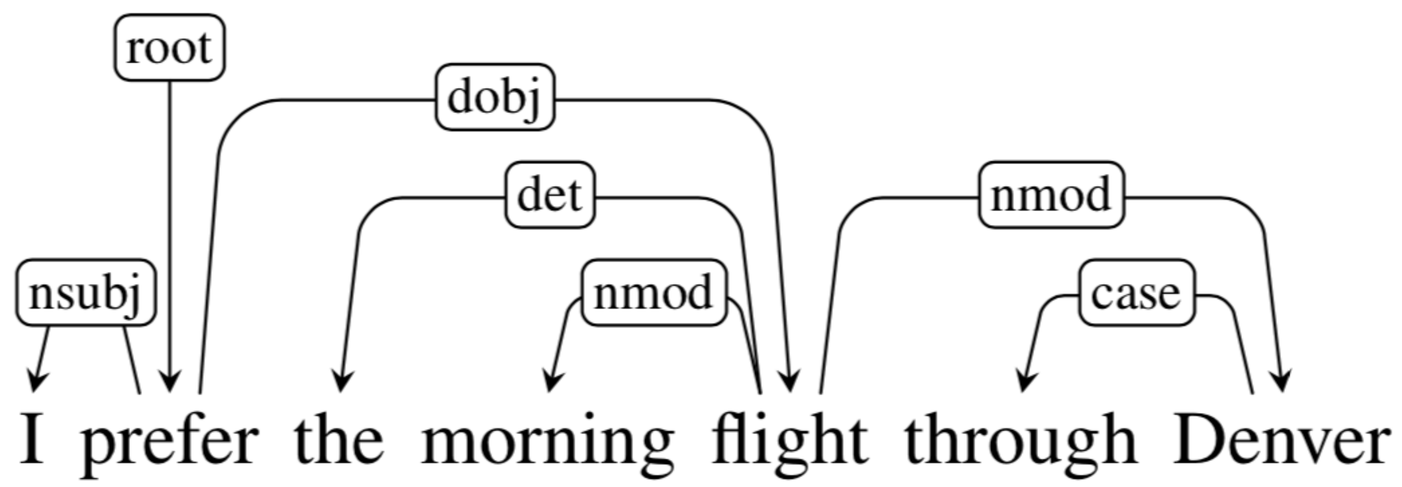


- Finding a non-cyclic graph where each word has one parent

Input:

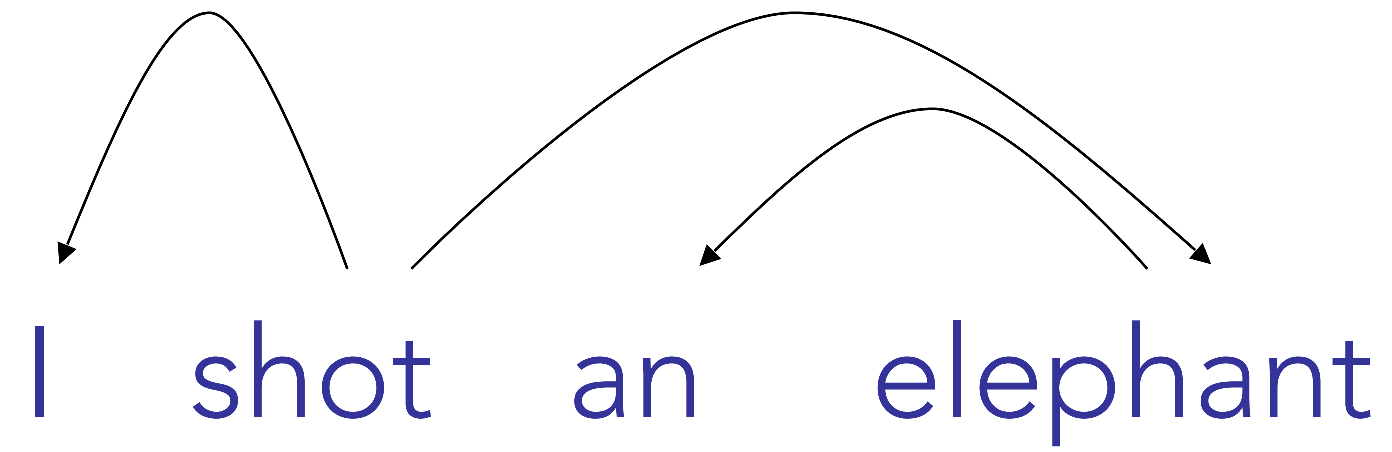
I prefer the morning flight through Denver

Output:



Dependency Grammar

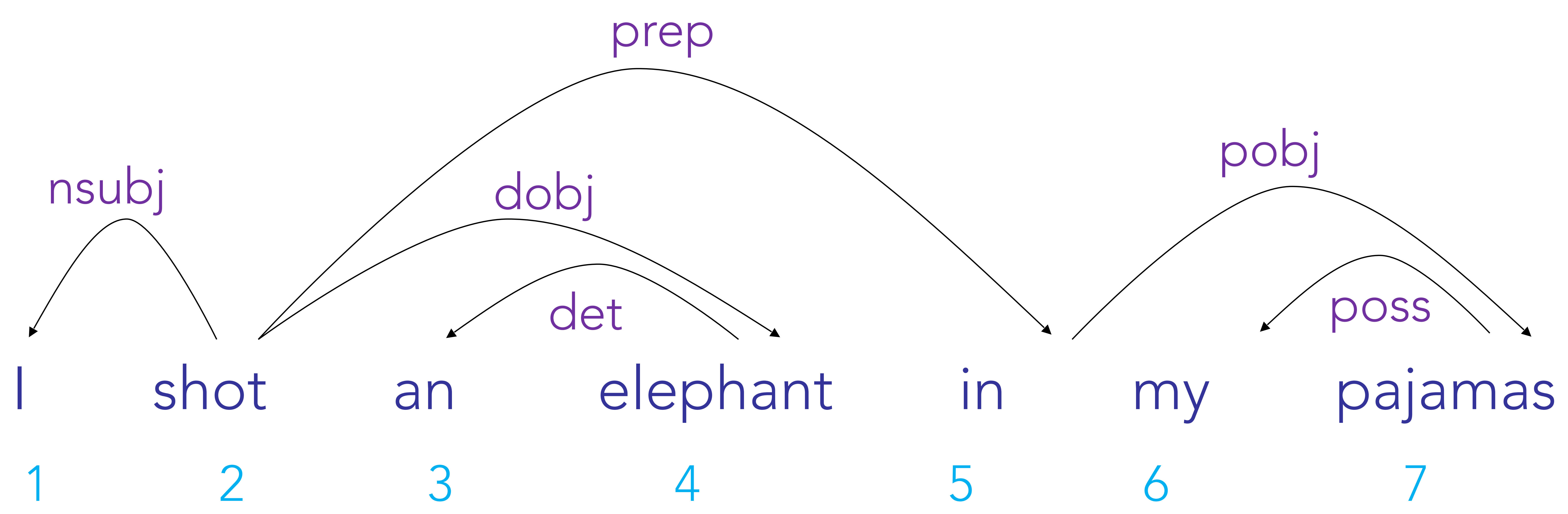
- For each word, find one parent
- A child is dependent on the parent
  - A child modifies a parent
  - A child is an argument of a parent



Typed Dependency

nsubj(shot-2, i-1)
root(**ROOT**-0, shot-2)
det(elephant-4, an-3)
dobj(shot-2, elephant-4)

prep(shot-2, in-5)
poss(pajamas-7, my-6)
pobj(in-5, pajamas-7)



Dependency Edge Types

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 14.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

Dependency Relations: Examples

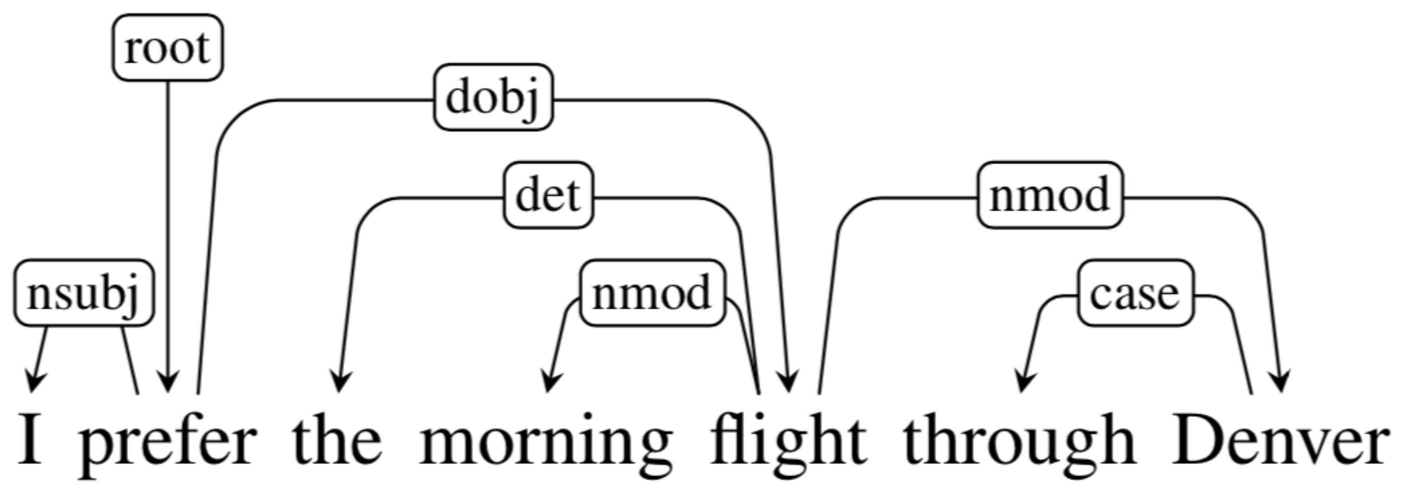
Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Dependency Parsing

Input:

I prefer the morning flight through Denver

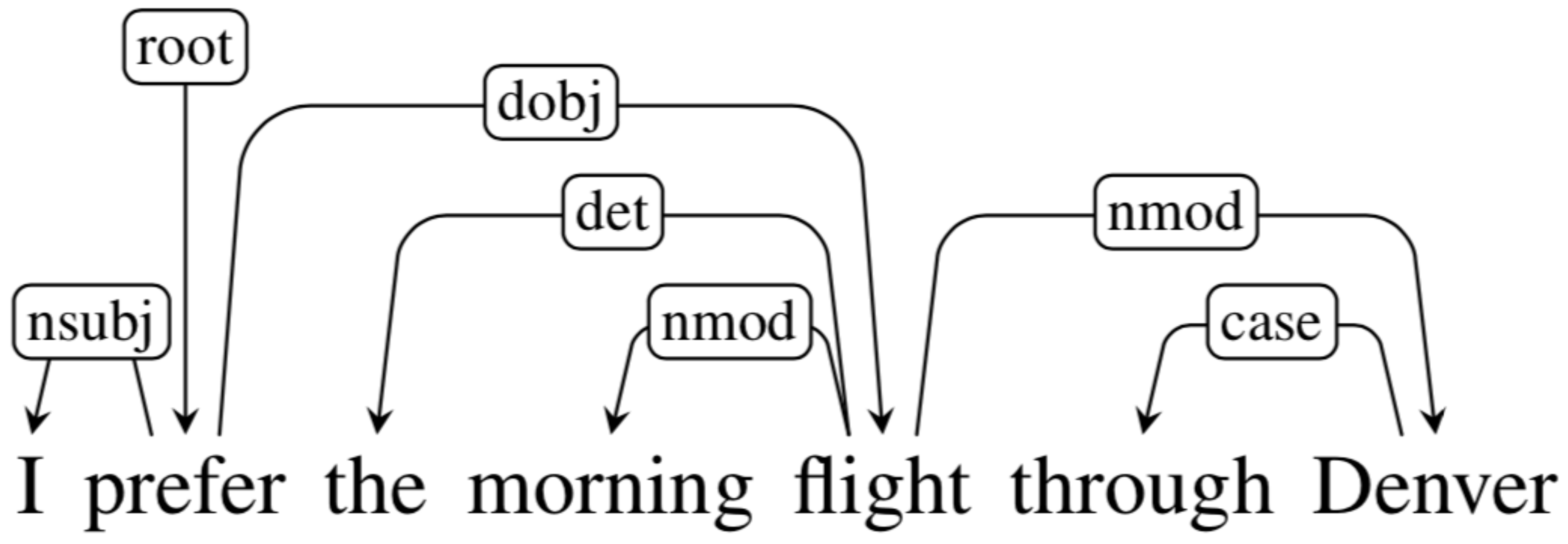
Output:



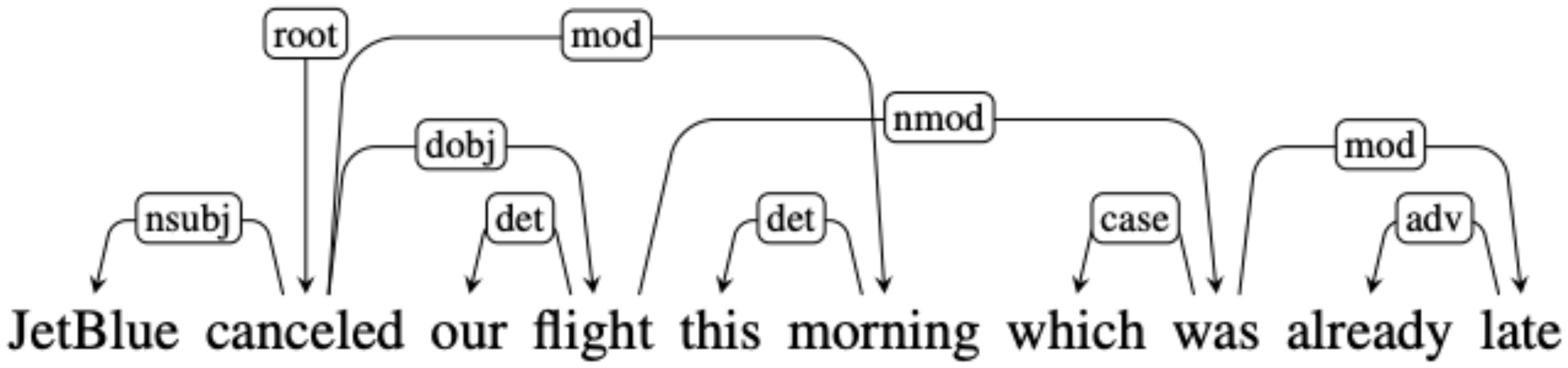
- A sentence is parsed by choosing for each word, what other word is it a dependent of (and also the type of relation)
- Add a fake ROOT at the beginning, such that every word has one head
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - No cycles

Projectivity

- Any subtree is a contiguous span of the sentence <-> tree is *projective*
- There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words



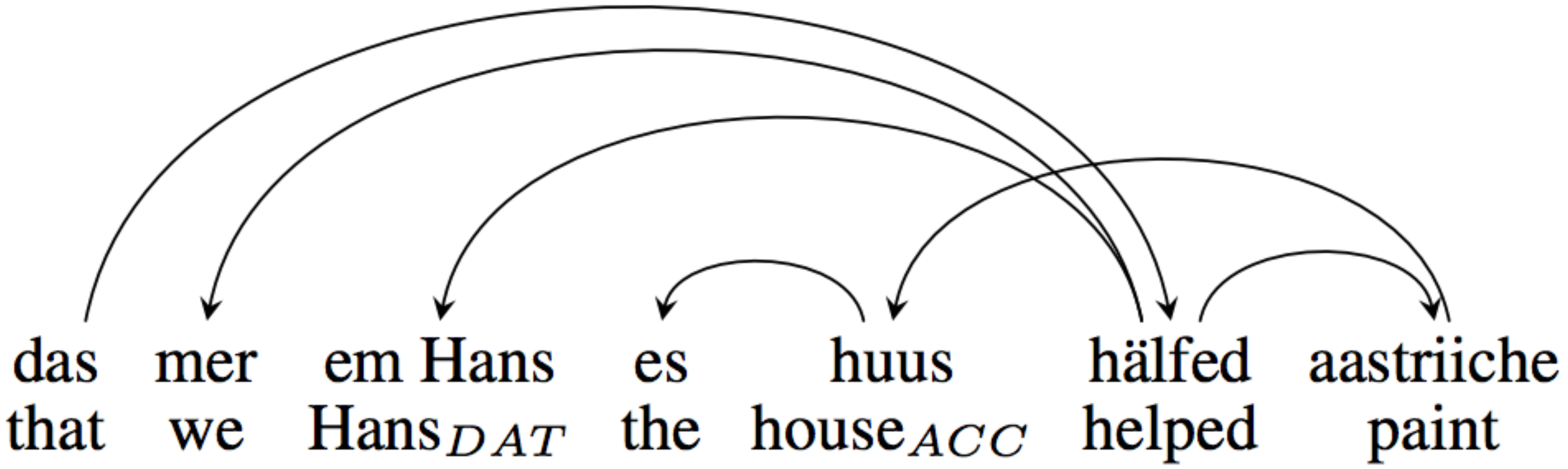
projective



non-projective

Projectivity in other languages

- Swiss German example



We helped Hans to paint the house

- Non-projectivity arises due to long range dependencies or flexible word order

Dataset	# Sentences	(%) Projective
English	39,832	99.9
Chinese	16,091	100.0
Czech	72,319	76.9
German	38,845	72.2

credit: Pitler et al. (2013)

Universal Dependencies

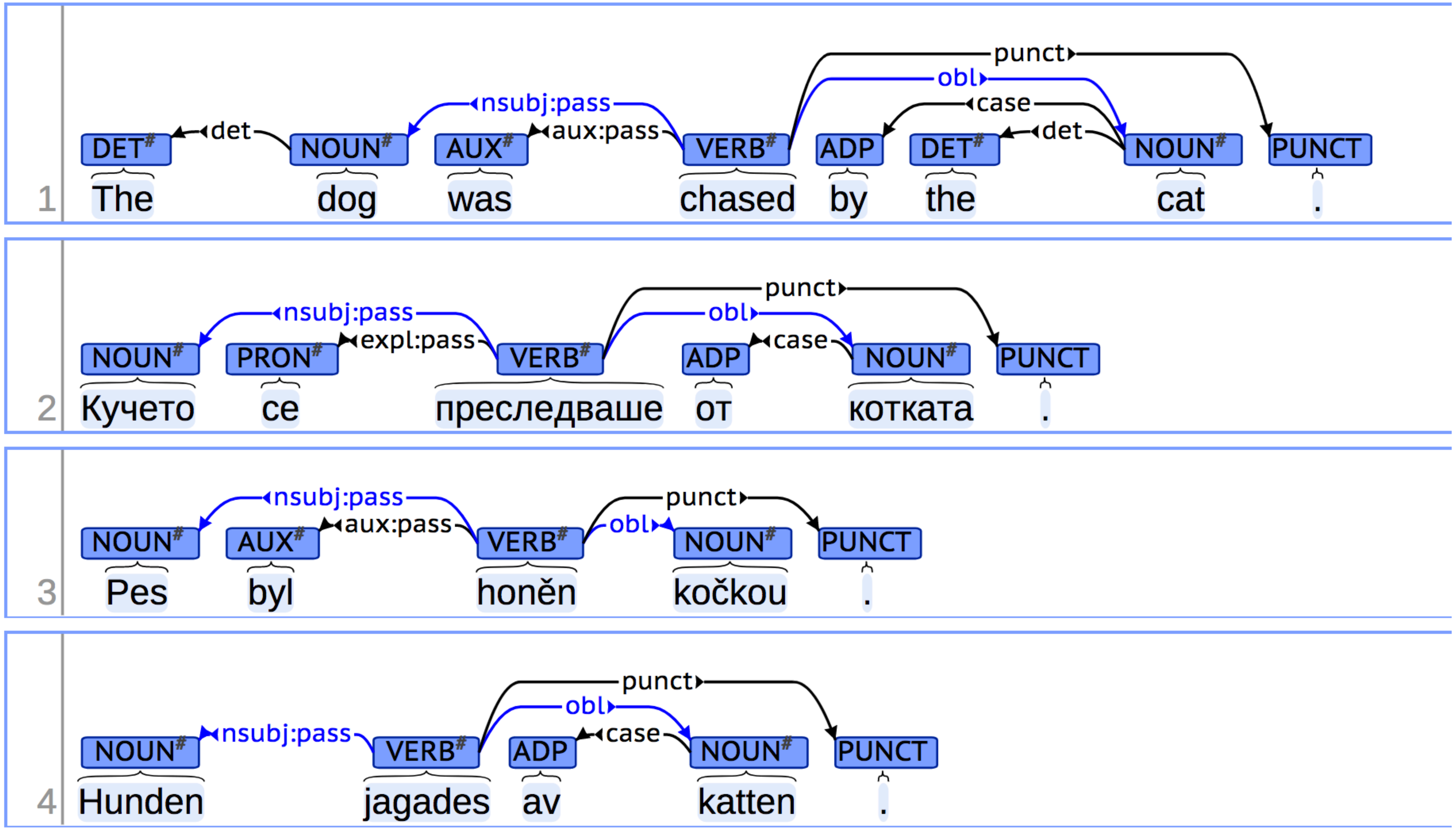
- Annotate dependencies with the same representation in 100+ languages

English

Bulgarian

Czech

Swiss



http://universaldependencies.org/

Let’s Parse Together!

Start with main verb, and draw dependencies. Don’t worry about labels. Just try to get the modifiers right.

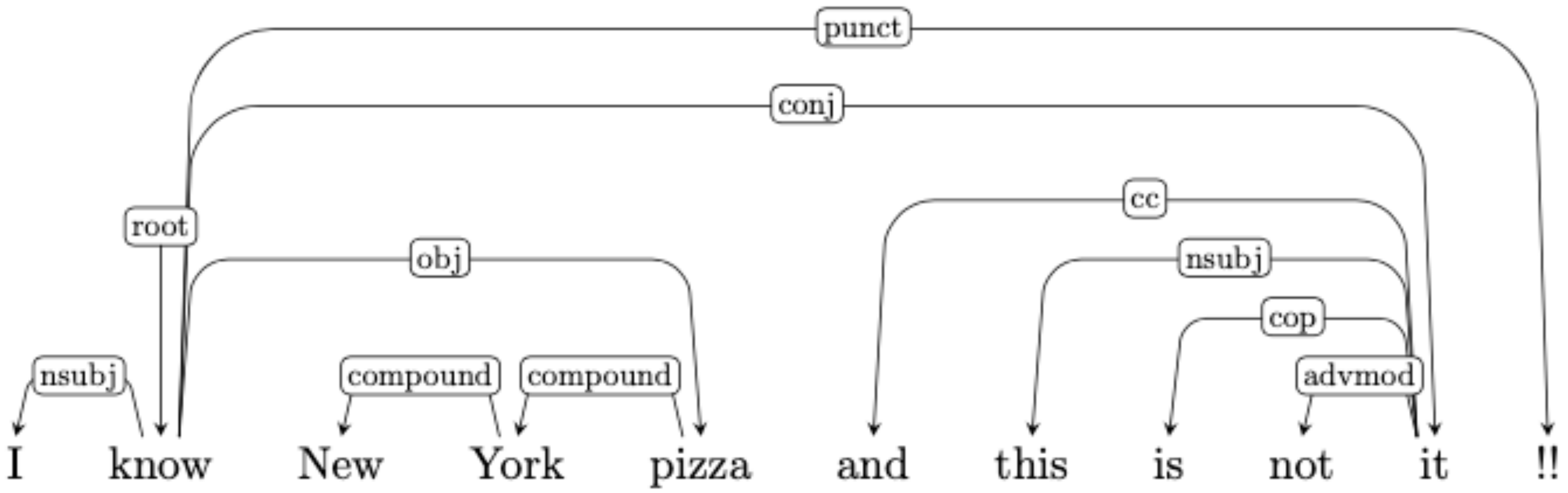
John saw Mary .

I know New York pizza and this is not it!

Let’s Parse Together!

Start with main verb, and draw dependencies. Don’t worry about labels. Just try to get the modifiers right.

John saw Mary .

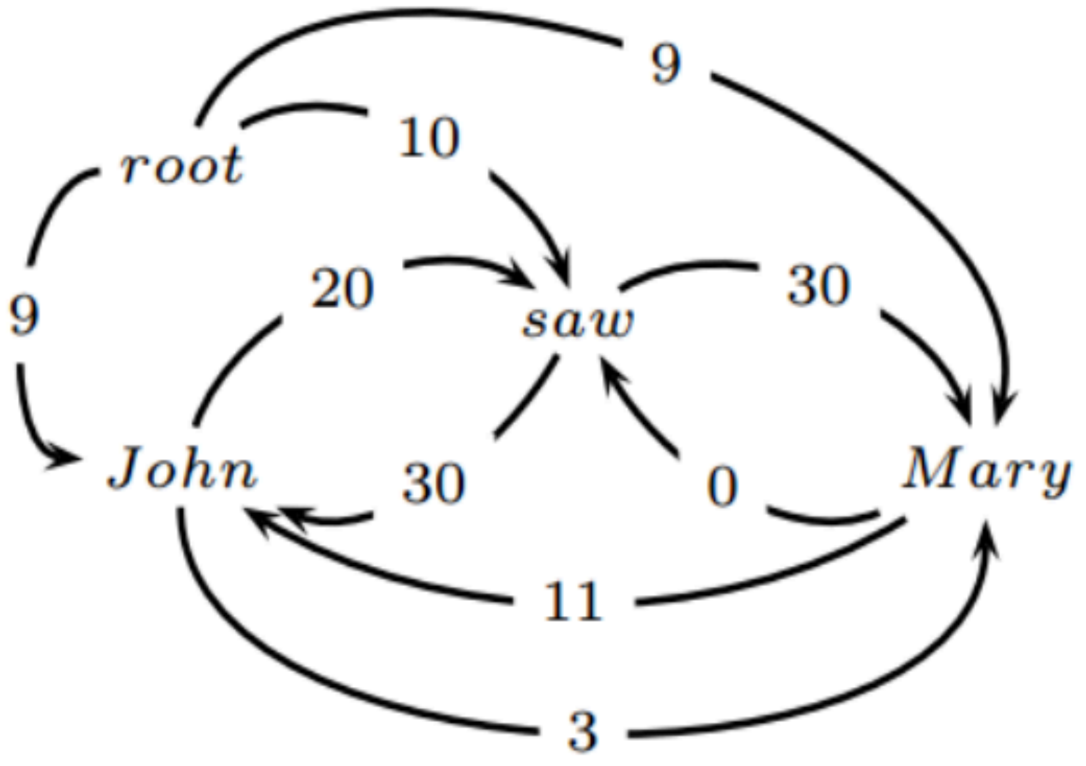


Making decisions

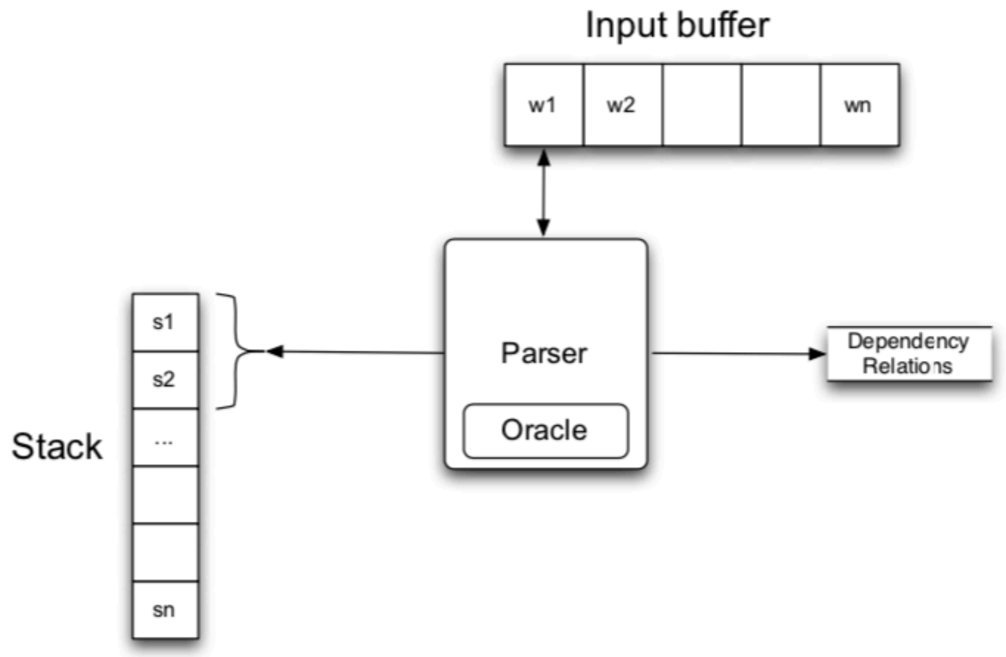
- Bilexical affinities
  - [issues -> the] is plausible
- Dependency distance
  - mostly with nearby words
- Intervening material
  - Dependencies rarely span intervening verbs or punctuation
- Valency of heads
  - How many dependents on which side are usual for a head

Dependency Parsers

- Graph-based parsing algorithm:
  - Score edges independently, find the maximum spanning tree

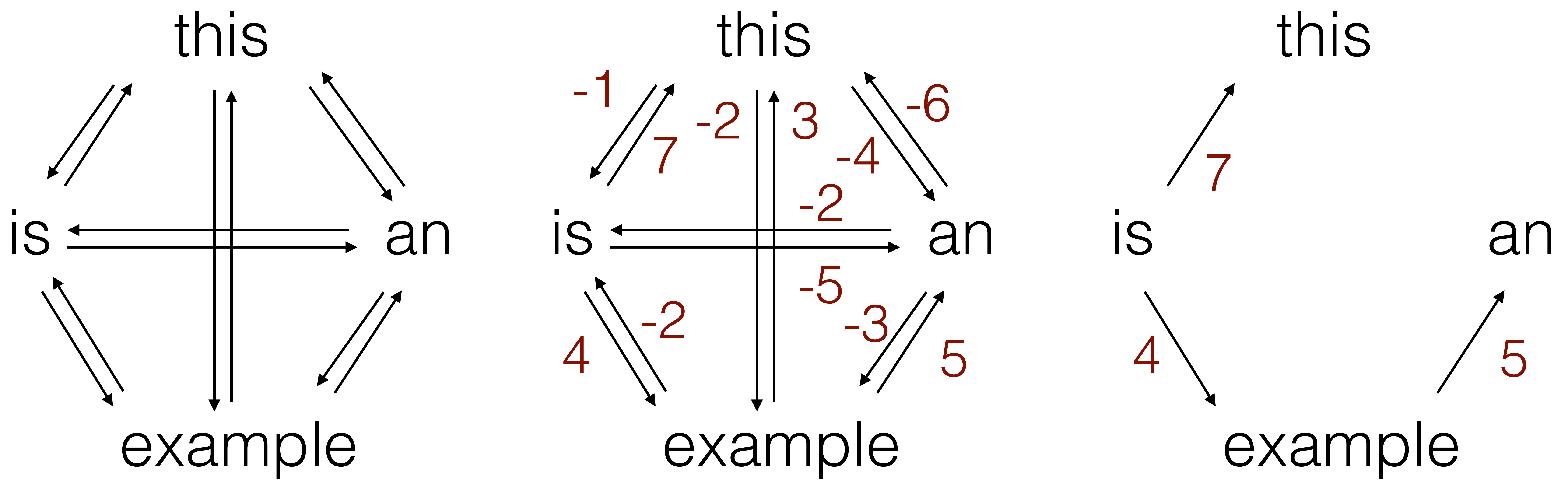


- Transition-based parsing:
  - Left-to-right, each choice is made with a classifier (faster)



Graph-based Dependency Parsing

- Express sentence as fully connected directed graph
- Score edges independently, and then find the maximum spanning tree



How do you score the edges?

- Features with arc direction / distance between the words
- Part-of-Speech tags and the word itself
- Then use your favorite classifier: logistic regression, perceptron, …

a)

Basic Uni-gram Features
p-word, p-pos
p-word
p-pos
c-word, c-pos
c-word
c-pos

b)

Basic Big-ram Features
p-word, p-pos, c-word, c-pos
p-pos, c-word, c-pos
p-word, c-word, c-pos
p-word, p-pos, c-pos
p-word, p-pos, c-word
p-word, c-word
p-pos, c-pos

c)

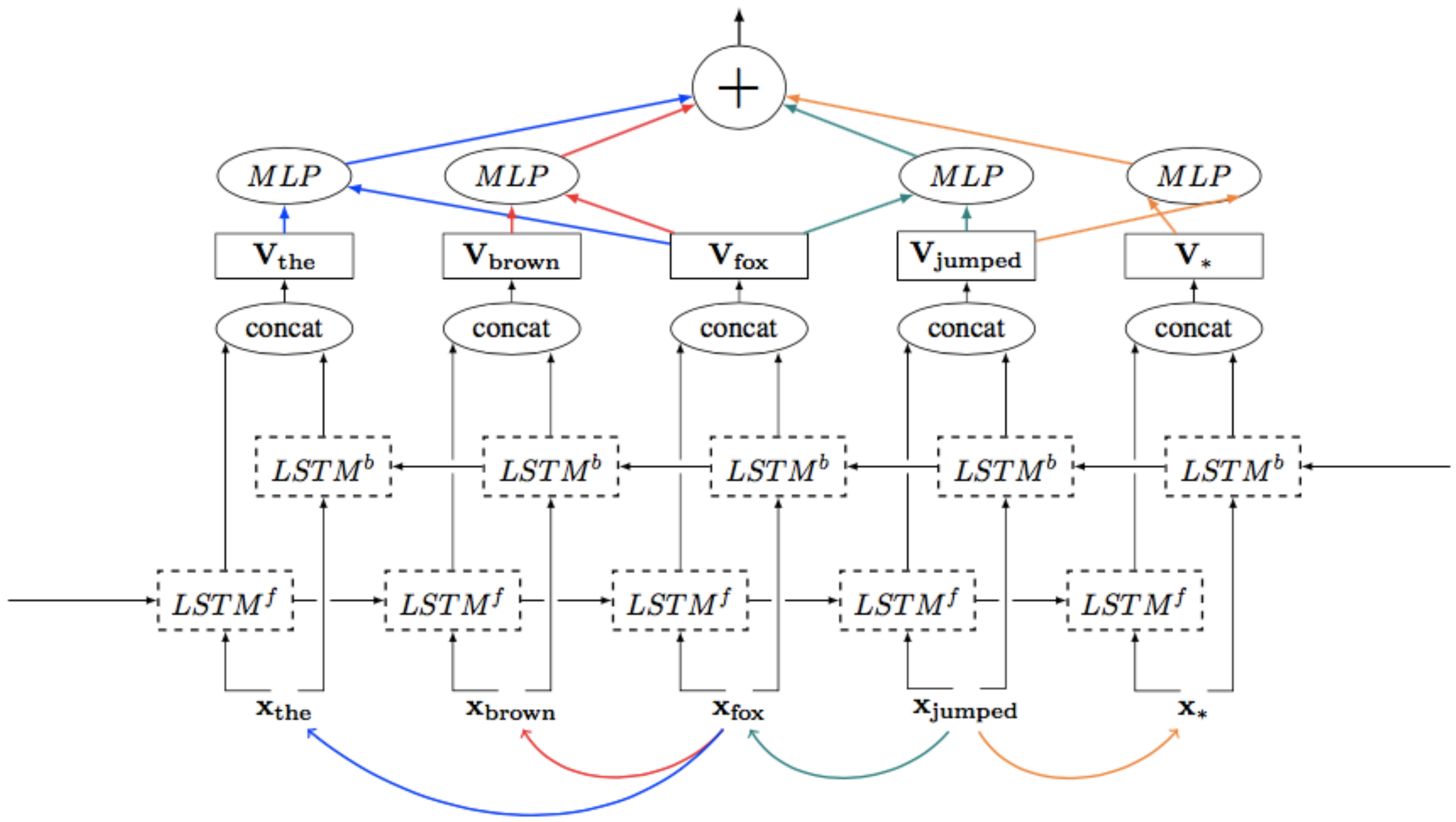
In Between POS Features
p-pos, b-pos, c-pos
Surrounding Word POS Features
p-pos, p-pos+1, c-pos-1, c-pos
p-pos-1, p-pos, c-pos-1, c-pos
p-pos, p-pos+1, c-pos, c-pos+1
p-pos-1, p-pos, c-pos, c-pos+1

Table 1: Features used by system. p-word: word of parent node in dependency tree. c-word: word of child node. p-pos: POS of parent node. c-pos: POS of child node. p-pos+1: POS to the right of parent in sentence. p-pos-1: POS to the left of parent. c-pos+1: POS to the right of child. c-pos-1: POS to the left of child. b-pos: POS of a word in between parent and child nodes.

McDonald et al, 2005

How do you score the edges?

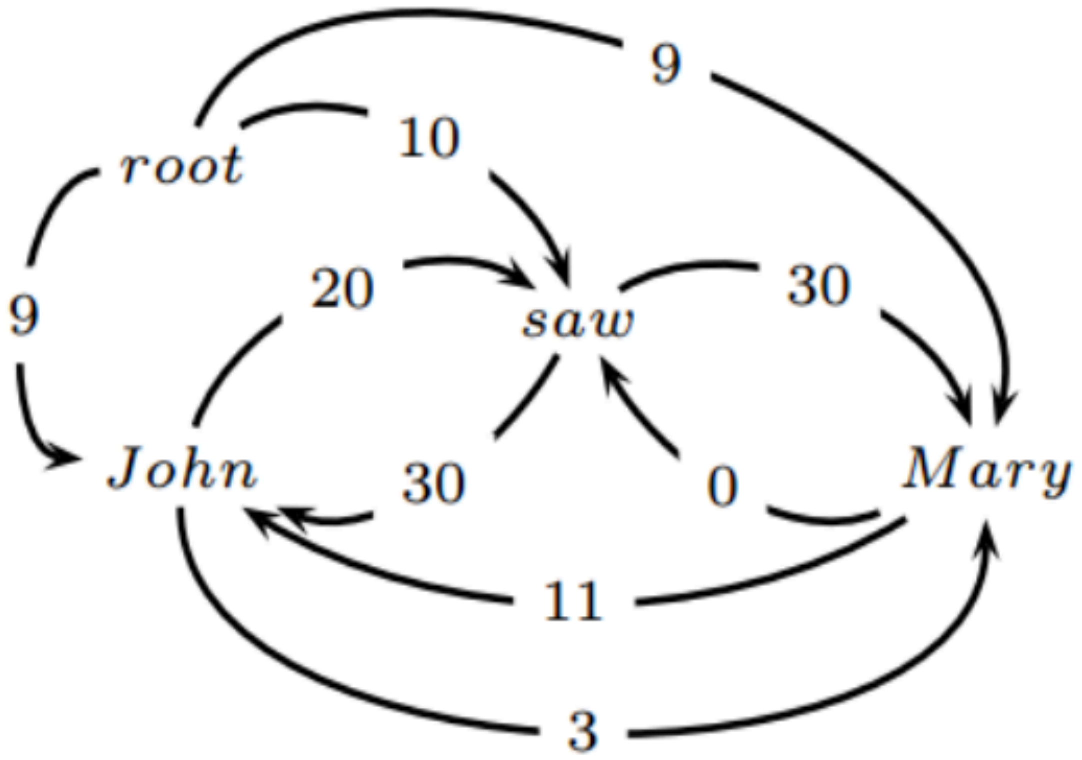
- Features learned from neural network (no feature engineering!)



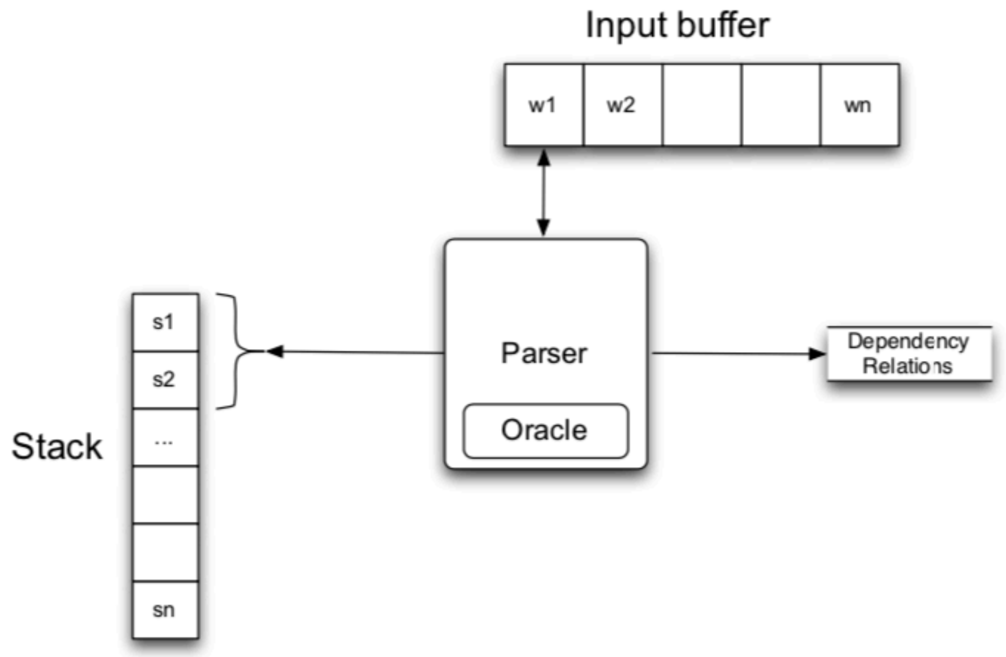
Kipperwasser and Goldberg, 2016

Dependency Parsers

- Graph-based parsing algorithm:
  - Score edges independently, find the maximum spanning tree



- Transition-based parsing:
  - Left-to-right, each choice is made with a classifier



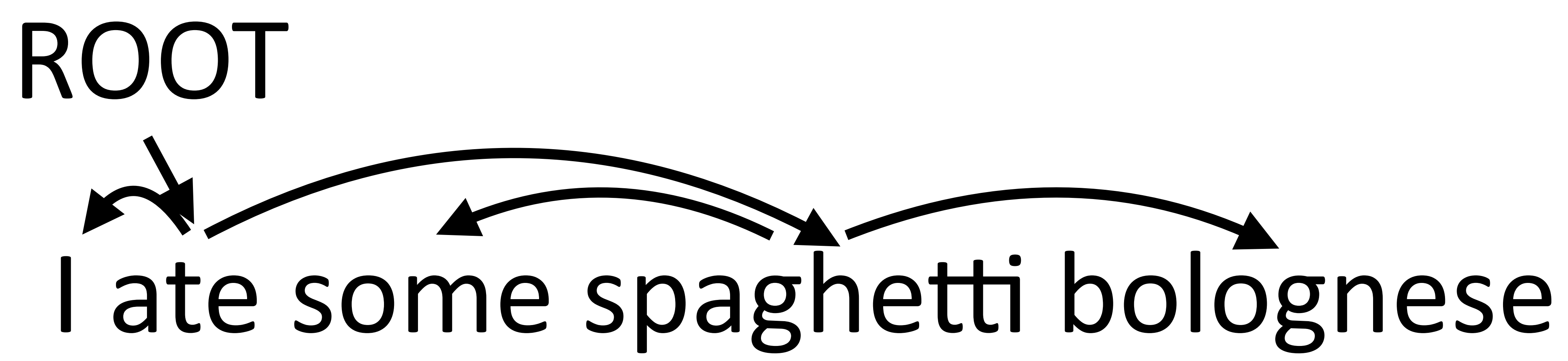
Transition Based Parsing

- Similar to shit reduce parsers for compilers
- Process a sentence word by word from a buffer
- You can temporarily place store words on a stack
- As you process, you can either:
  - Shift: Move a word from the buffer to a stack
  - Left: The top of the stack is the head of the second word on stack
  - Right: The second word on stack is head of top word

Transition Based Parsing

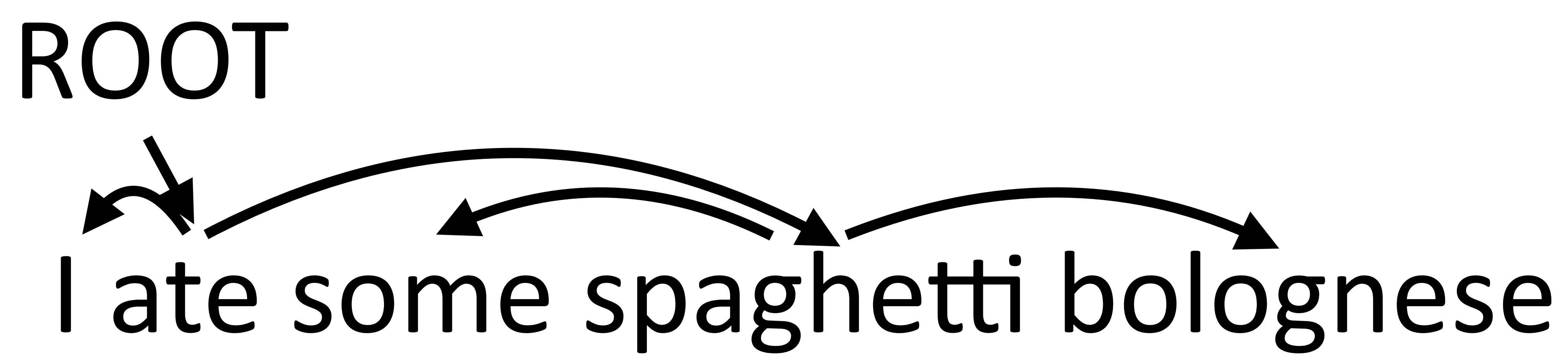
- Initially, the stack has root, the buffer has sentence's words, and no edges
  - Eg) stack contains [ROOT], buffer contains [I ate some spaghetti bolognese]
- At the end, stack contains [ROOT], buffer is empty []

Transition Based Parsing



- Initial state: Stack: [ROOT] Buffer: [I ate some spaghetti bolognese]
- Shift: top of buffer -> top of stack
  - Shift 1: Stack: [ROOT I] Buffer: [ate some spaghetti bolognese]
  - Shift 2: Stack: [ROOT I ate] Buffer: [some spaghetti bolognese]

Transition Based Parsing



- State: Stack: [ROOT I ate] Buffer: [some spaghetti bolognese]

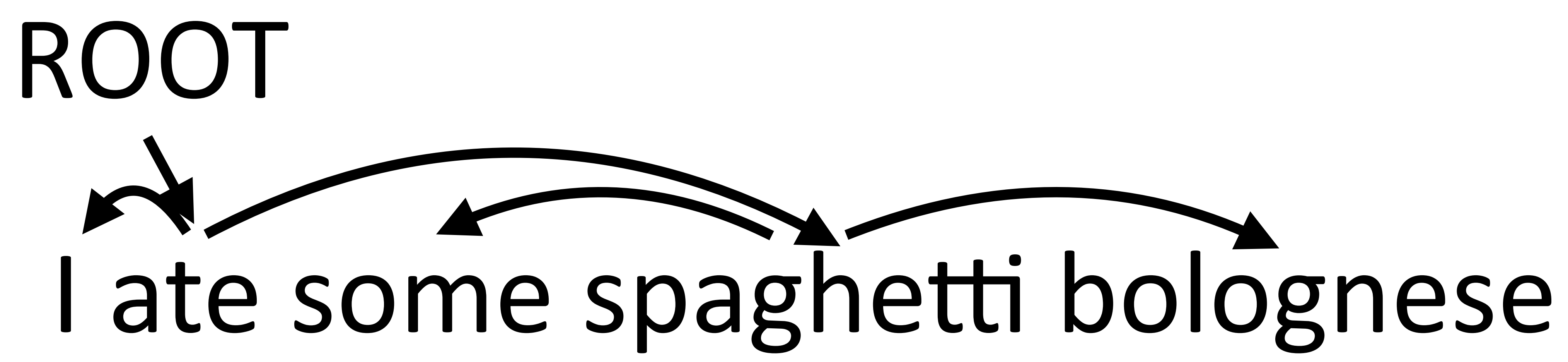
- Left-arc (reduce): Let σ denote the stack, $\sigma | w_{-1}$ = stack ending in w_{-1}
  - "Pop two elements, add an arc, put them back on the stack"

$\boxed{\sigma | w_{-2}, w_{-1}} \rightarrow \boxed{\sigma | w_{-1}}$ w_{-2} is now a child of w_{-1}

- State: Stack: [ROOT ate] Buffer: [some spaghetti bolognese]

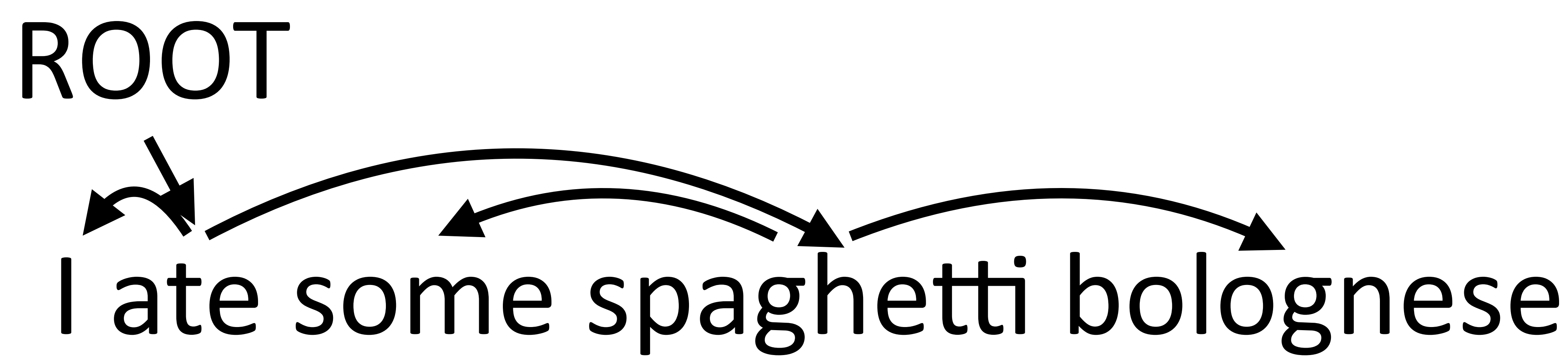
I

Arc-Standard Parsing



- Start: stack contains [ROOT], buffer contains [I ate some spaghetti bolognese]
- Three operations
  - Shift: top of buffer -> top of stack
  - Left-Arc: $\boxed{\sigma | w_{-2}, w_{-1}} \rightarrow \boxed{\sigma | w_{-1}}$, w_{-2} is now a child of w_{-1}
  - Right-Arc $\boxed{\sigma | w_{-2}, w_{-1}} \rightarrow \boxed{\sigma | w_{-2}}$, w_{-1} is now a child of w_{-2}
- End: stack contains [ROOT], buffer is empty []

Arc-Standard Parsing



S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

Stack	Action	Buffer
[ROOT]		[I ate some spaghetti bolognese]
	S	
[ROOT I]		[ate some spaghetti bolognese]
	S	
[ROOT I ate]		[some spaghetti bolognese]
	LA	
[ROOT ate]		[some spaghetti bolognese]

$$\begin{array}{c}\text{ate}\\ \downarrow \\ \text{I}\end{array}$$

- Can't attach [ROOT <- ate] yet even though this is a correct dependency!

Arc-Standard Parsing

ROOT

I ate some spaghetti bolognese

S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

[ROOT ate] [some spaghetti bolognese]

$\downarrow$ I

S

S

[ROOT ate some spaghetti] [bolognese]

$\downarrow$ I

LA

[ROOT ate spaghetti] [bolognese]

$\downarrow$ I $\downarrow$ some

S

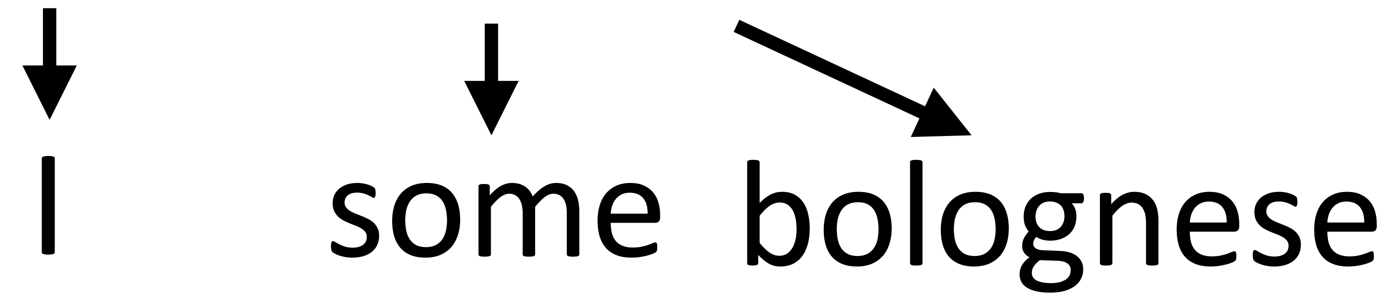
Arc-Standard Parsing

ROOT

[ROOT ate spaghetti bolognese] []

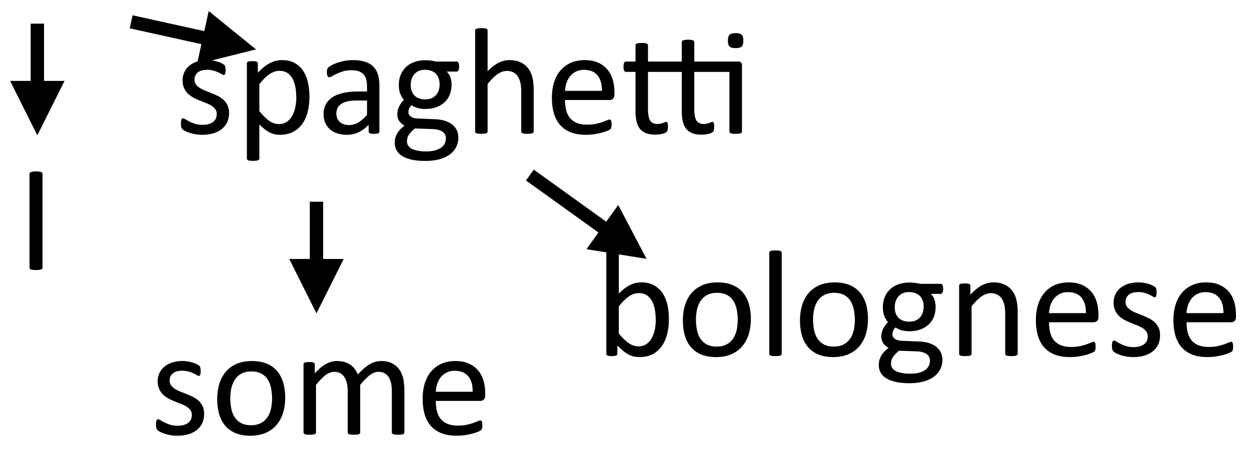
RA

[ROOT ate spaghetti] []



RA

[ROOT ate] []

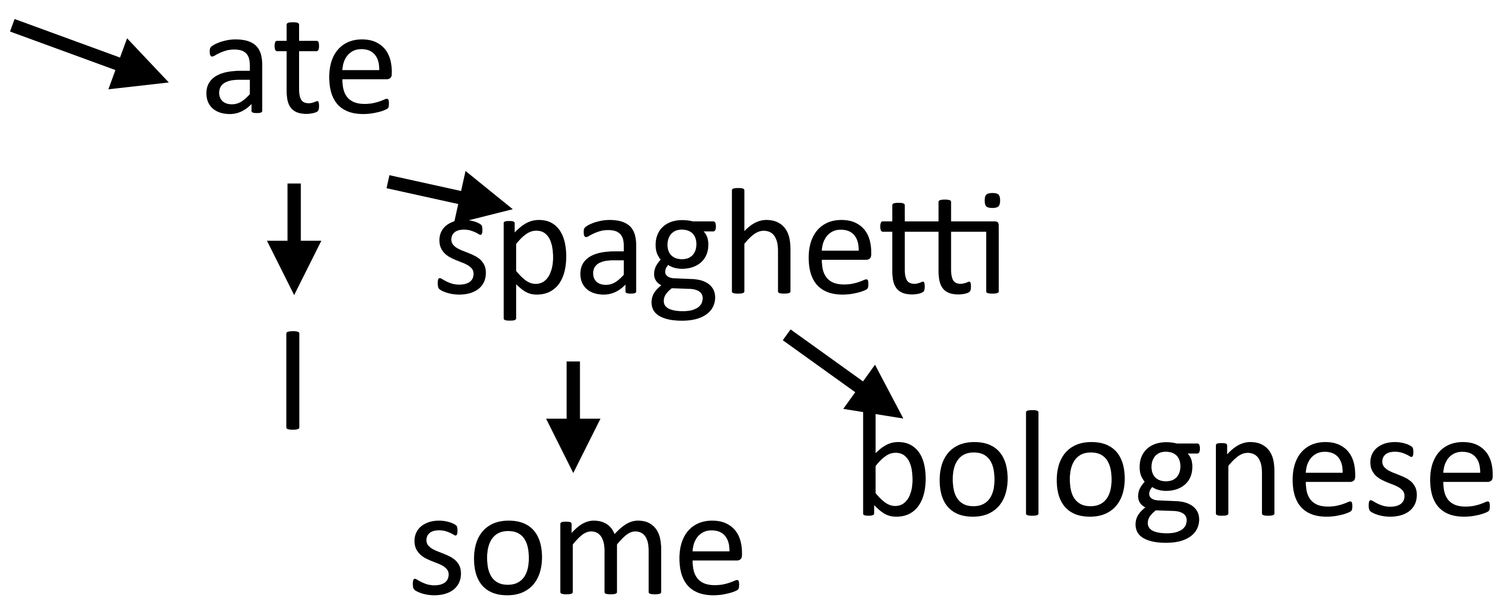


S	top of buffer -> top of stack
LA	pop two, left arc between them
RA	pop two, right arc between them

- Stack consists of all words that are still waiting for right children, end with a bunch of right-arc ops

Final state:

[ROOT] []



Train a classifier to predict actions

- How do we get a sequence of actions from the annotated parse tree?
- We should learn a training oracle, following the rule:

Given this information, the oracle chooses transitions as follows:

LEFTARC(r): **if** $(S_1\ r\ S_2) \in R_p$

RIGHTARC(r): **if** $(S_2\ r\ S_1) \in R_p$ **and** $\forall r', w\ s.t.(S_1\ r'\ w) \in R_p$ **then** $(S_1\ r'\ w) \in R_c$

SHIFT: **otherwise**

If the second element in the stack is the child of the top of the stack, then make a left edge.

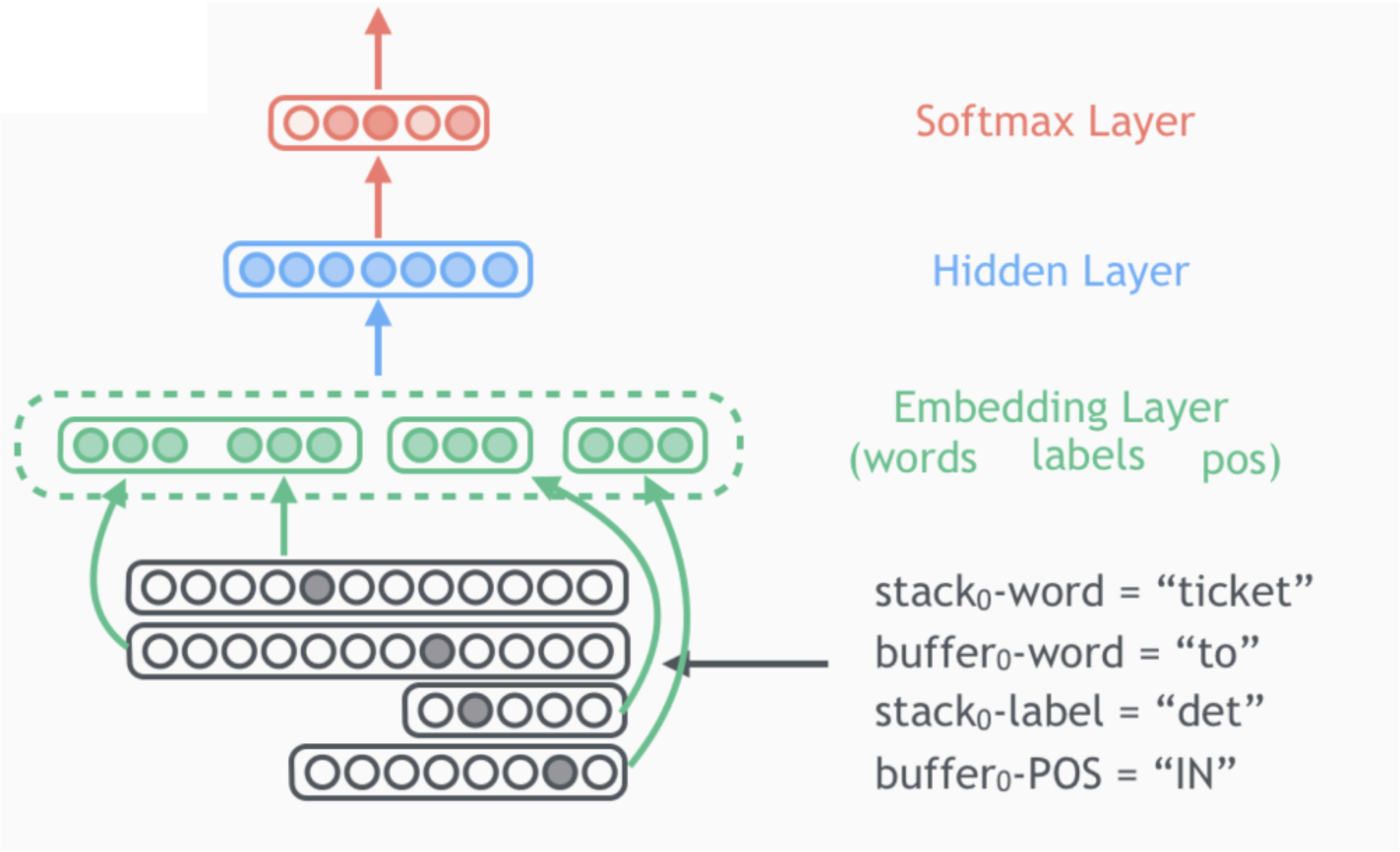
If the top of the stack is the child of the second element in the stack, **and** the second element of the buffer has no children that have yet to be added to the tree, then make a right edge.

- Then, train a classifier, which predicts an action for each configuration.
- Space of actions:
  - 3 (if untyped edges)
  - |R|*2 + 1 (if typed edges, R the number of types)

Features in Transition Based Parsing

- The top 3 words on the stack and buffer (6 features)
  $s_1, s_2, s_3, b_1, b_2, b_3$
- The two leftmost/rightmost children of the top two words on the stack (8 features)
  $lc_1(s_i), lc_2(s_i), rc_1(s_i), rc_2(s_i)$ $i=1,2$
- leftmost and rightmost grandchildren (4 features)
  $lc_1(lc_1(s_i)), rc_1(rc_1(s_i))$ $i=1,2$
- POS tags of all of the above (18 features)
- Arc labels of all children/grandchildren (12 features)

Neural Features



Chen and Manning 2014

Complexity

- A word can only enter the stack once
- So complexity is O(2n)

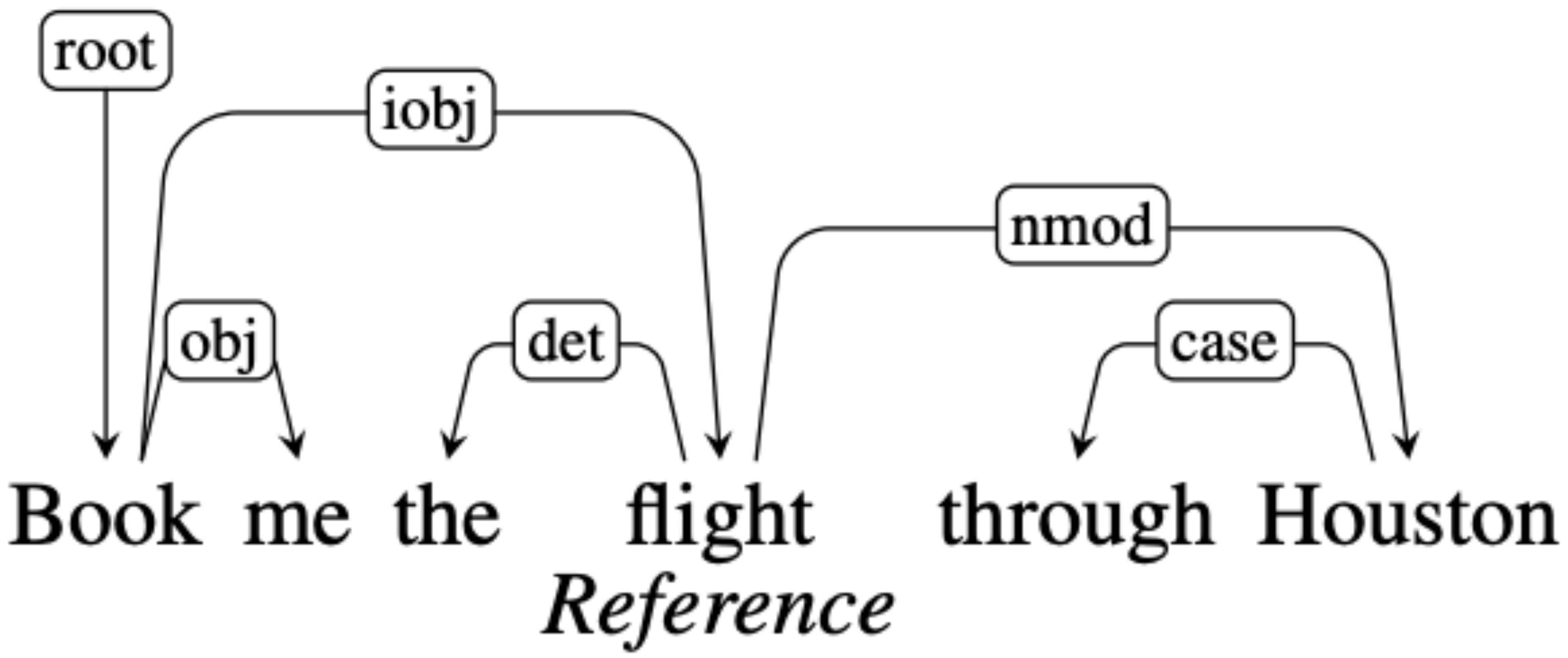
Graph-based vs. Transition-based

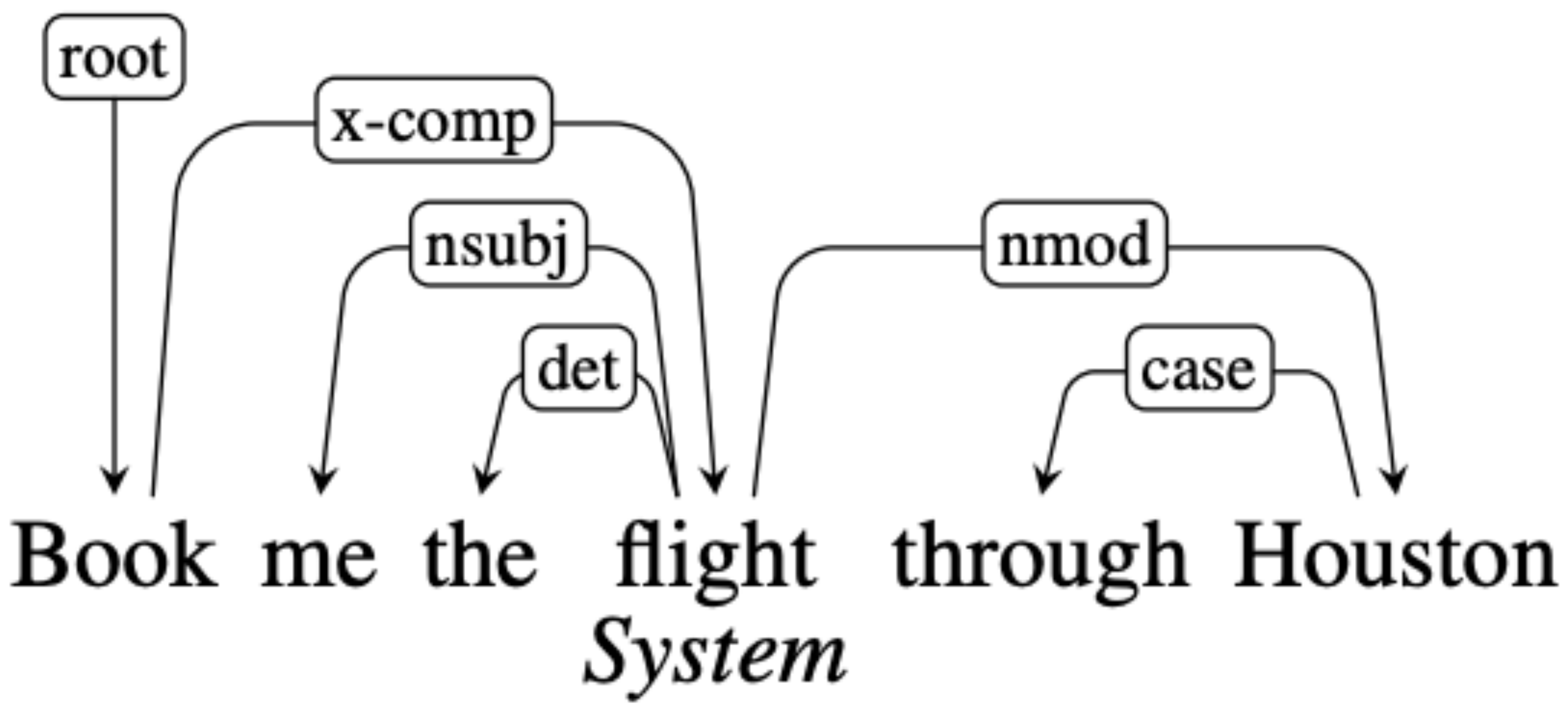
- Graph based:
  - Can find exact best global solution with dynamic programming
  - Local independence assumption

- Transition based:
  - Greedy algorithm
  - Cannot model non-projective trees
  - Can condition on longer tree context

Evaluation

- Unlabeled attachment score (UAS)
  - Percentage of words that have been assigned the correct head
- Labeled attachment score (LAS)
  - Percentage of words that have been assigned the correct head & edge label





Results

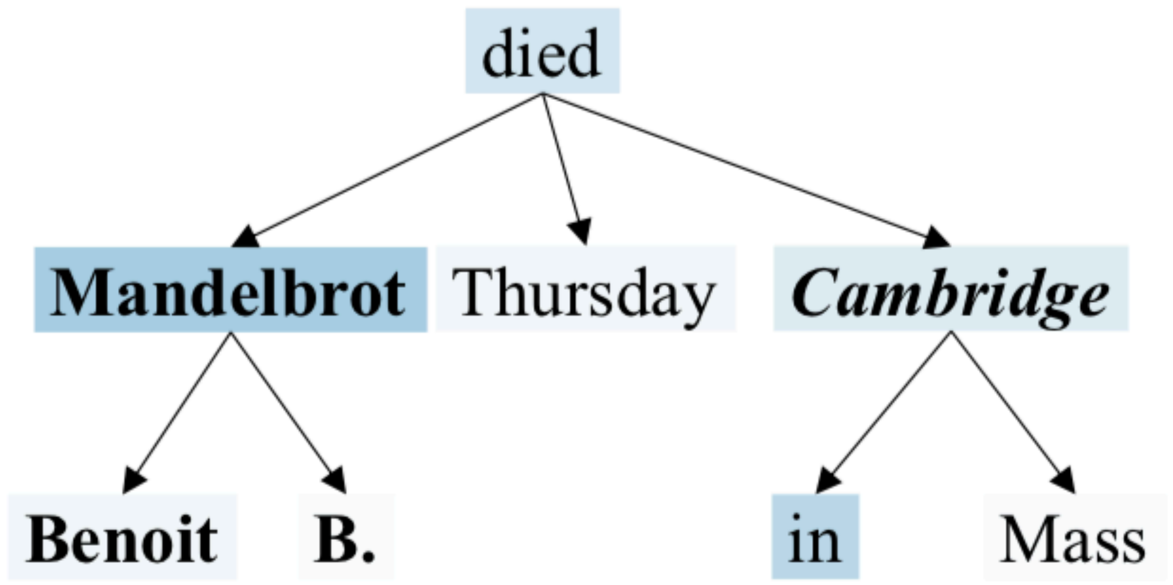
		English PTB-SD 3.3.0		Chinese PTB 5.1	
Type	**Model**	**UAS**	**LAS**	**UAS**	**LAS**
Transition	Ballesteros et al. (2016)	93.56	91.42	87.65	86.21
	Andor et al. (2016)	94.61	92.79	–	–
	Kuncoro et al. (2016)	**95.8**	**94.6**	–	–
Graph	Kiperwasser & Goldberg (2016)	93.9	91.9	87.6	86.1
	Cheng et al. (2016)	94.10	91.49	88.1	85.7
	Hashimoto et al. (2016)	94.67	92.90	–	–
	Deep Biaffine	95.74	94.08	**89.30**	**88.23**

Dozat and manning 2017

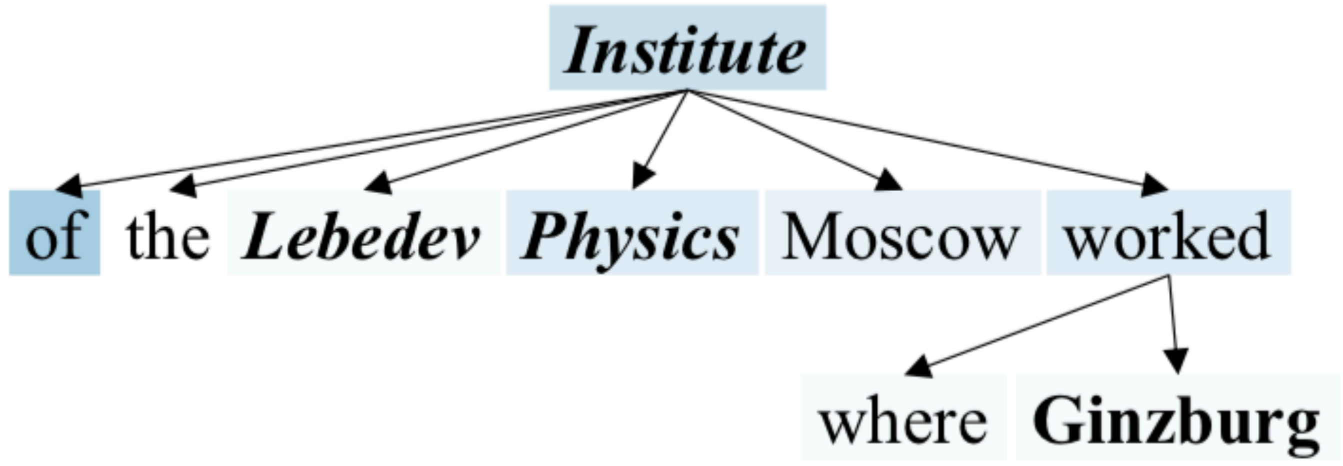
Using Dependency Parsers

- For information extraction
- Features capturing dependency path between entities

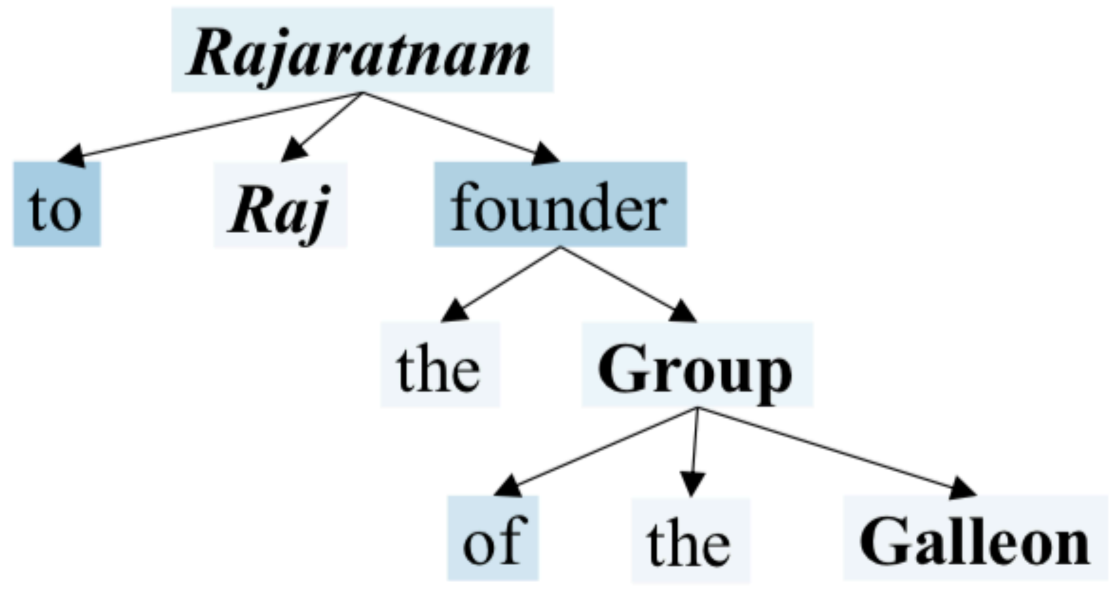
Relation: *per:city_of_death*
Benoit B. Mandelbrot, a maverick mathematician who developed an innovative theory of roughness and applied it to physics, biology, finance and many other fields, died Thursday in ***Cambridge***, Mass.



Relation: *per:employee_of*
In a career that spanned seven decades, Ginzburg authored several groundbreaking studies in various fields -- such as quantum theory, astrophysics, radio-astronomy and diffusion of cosmic radiation in the Earth's atmosphere -- that were of "Nobel Prize caliber," said Gennady Mesyats, the director of the ***Lebedev Physics Institute*** in Moscow, where **Ginzburg** worked .



Relation: *org:founded_by*
Anil Kumar, a former director at the consulting firm McKinsey & Co, pleaded guilty on Thursday to providing inside information to ***Raj Rajaratnam***, the founder of the **Galleon Group**, in exchange for payments of at least $ 175 million from 2004 through 2009.



Takeaways

- Dependency formalism provides an alternative to constituency, particularly useful in how portable it is across languages

- There are two dependency parsing paradigms: transition based, and graph based, both works well in practice.