THE OHIO STATE UNIVERSITY

CSE 5525: Foundations of Speech and Language Processing

Dependency I

Huan Sun (CSE@OSU)

Many thanks to Prof. Greg Durrett @ UT Austin for sharing his slides.
Some slides were adapted from Prof. Christopher Manning, Stanford.
Some images/examples were from the two textbooks by (1) Jurafsky and Martin and (2) Eisenstein.

Outline

- Dependency representation (in contrast with constituency)
- Projectivity
- Transition-based dependency parsing

Dependency Representation

Two Views of Linguistic Structure

- Phrase Structure (Phrasal constituency)

- Dependency structure

1. Two views of linguistic structure: Constituency = phrase structure grammar = context-free grammars (CFGs)

Phrase structure organizes words into nested constituents

Starting unit: words

the, cat, cuddly, by, door

Words combine into phrases

the cuddly cat, by the door

Phrases can combine into bigger phrases

the cuddly cat by the door

Slides adapted from Prof. Christopher Manning, Stanford

1. Two views of linguistic structure: Constituency = phrase structure grammar = context-free grammars (CFGs)

Phrase structure organizes words into nested constituents

Can represent the grammar with CFG rules

Starting unit: words are given a category (part of speech = pos)

the, cat, cuddly, by, door
Det N Adj P N

Words combine into phrases with categories

the cuddly cat, by the door
NP → Det Adj N PP → P NP

Phrases can combine into bigger phrases recursively

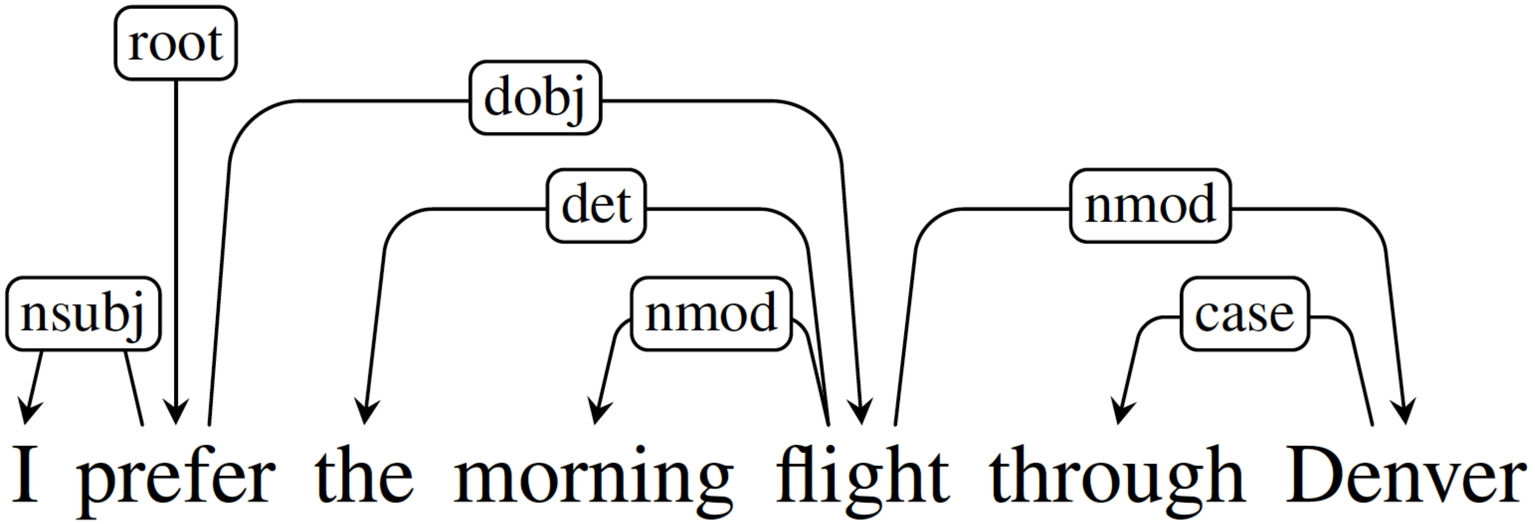
the cuddly cat by the door

NP → NP PP

Slides adapted from Prof. Christopher Manning, Stanford

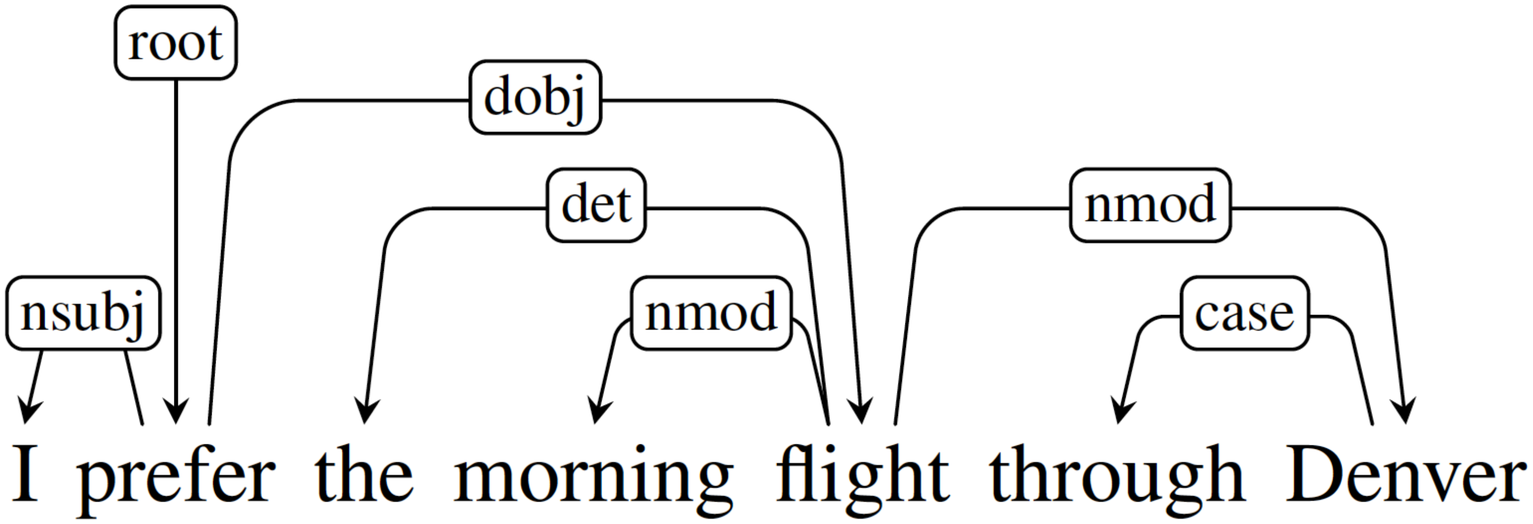
Dependency Structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.



Dependency Structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.



In contrast with phrase structure, dependency structure of a sentence is described solely in terms of the words (or lemmas) in a sentence and an associated set of directed binary grammatical relations that hold among the words.

Why do we need sentence structure?

We need to understand sentence structure in order to be able to interpret language correctly

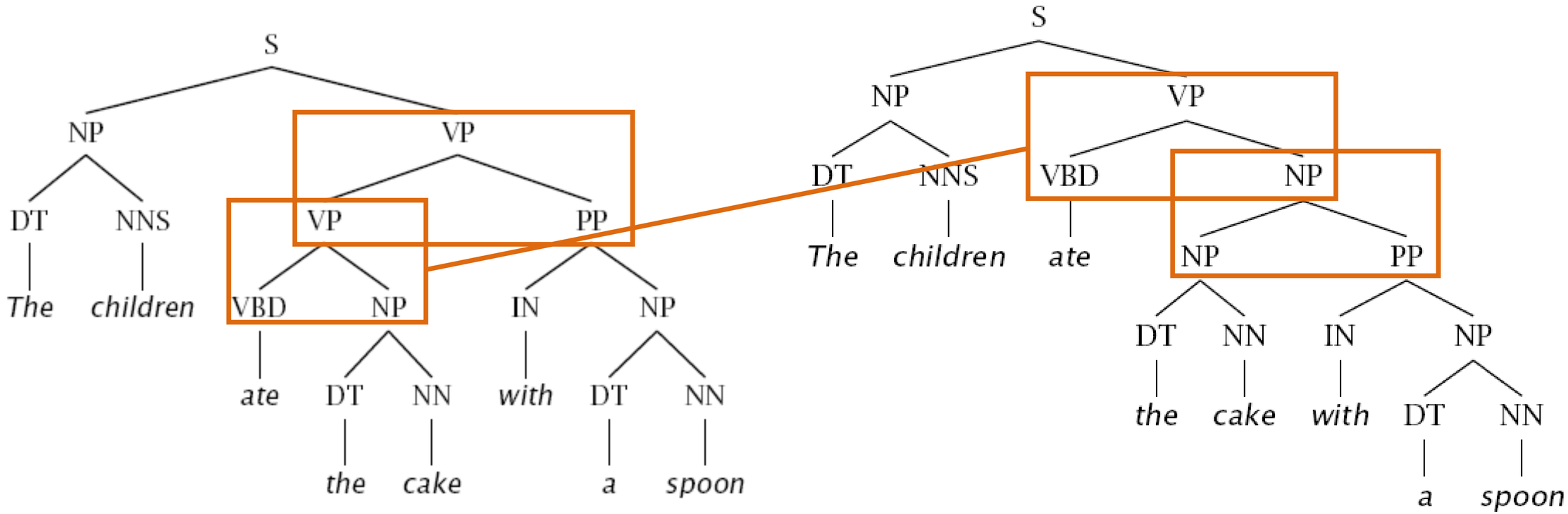
Humans communicate complex ideas by composing words together into bigger units to convey complex meanings

We need to know what is connected to what

Slides adapted from Prof. Christopher Manning, Stanford

Dependency vs. Constituency: PP Attachment Ambiguity

- Constituency: several rule productions need to change



Dependency vs. Constituency: PP Attachment Ambiguity

- Dependency: one word (“with”) may be assigned a different parent

- More predicate-argument focused view of syntax
- “What’s the main verb of the sentence? What is its subject and object?” — easier to answer under dependency parsing

Prepositional phrase attachment ambiguity

Prepositional phrase attachment ambiguity

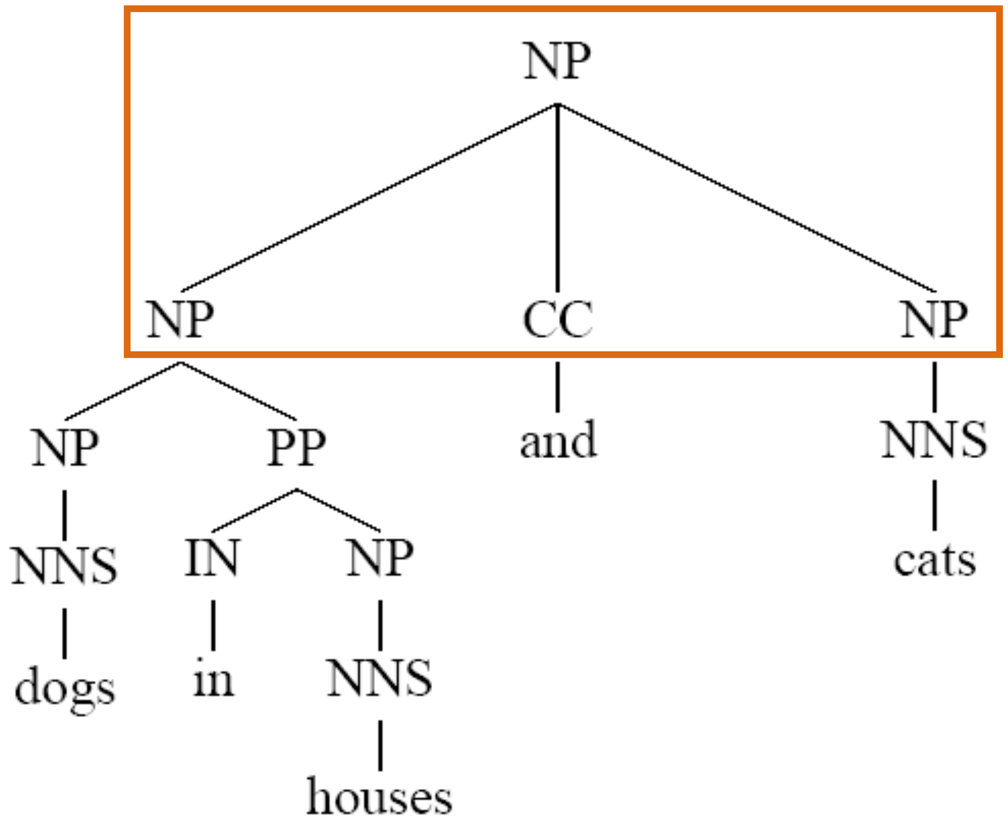
Scientists count whales from space

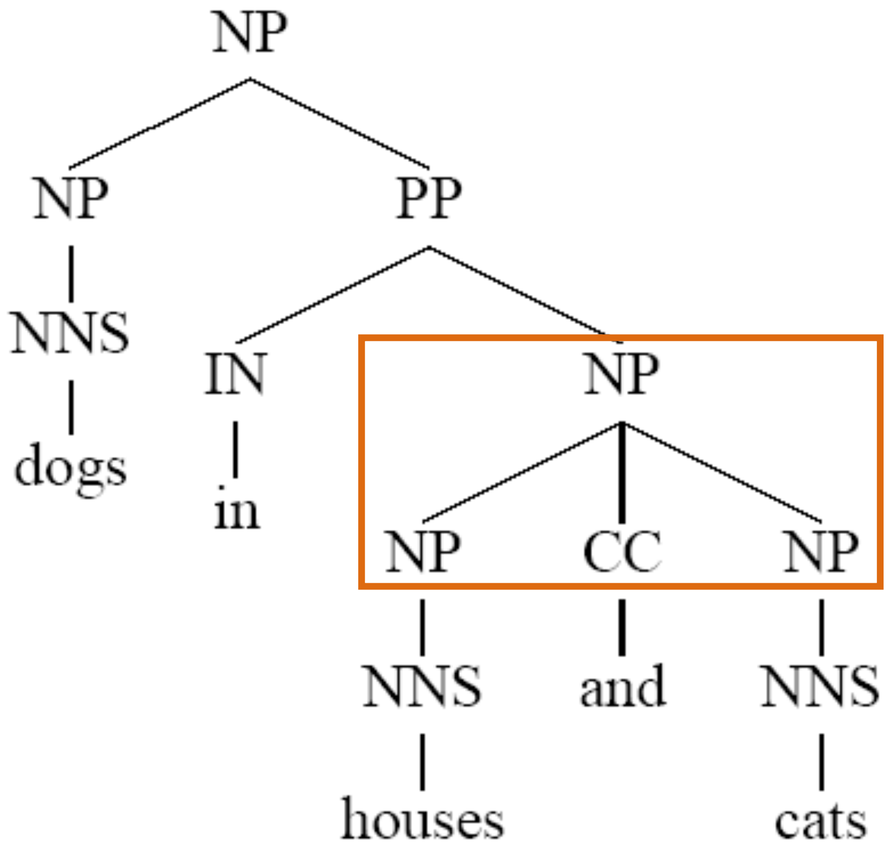
✓

Scientists count whales from space

Dependency vs. Constituency: Coordination Ambiguity

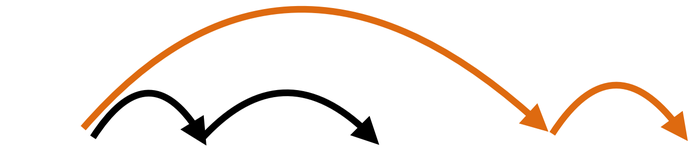
- Constituency: ternary rule NP -> NP CC NP





Dependency vs. Constituency: Coordination Ambiguity

- Dependency:



dogs in houses **and cats**

[dogs in houses] and cats

dogs in **houses and cats**

dogs in [houses and cats]

- Coordination is decomposed across a few arcs as opposed to being a single rule production as in constituency
- Can also choose *and* to be the head
- In both cases, headword doesn't really represent the phrase — constituency representation makes more sense

Coordination scope ambiguity

Shuttle veteran and longtime NASA executive Fred Gregory appointed to board

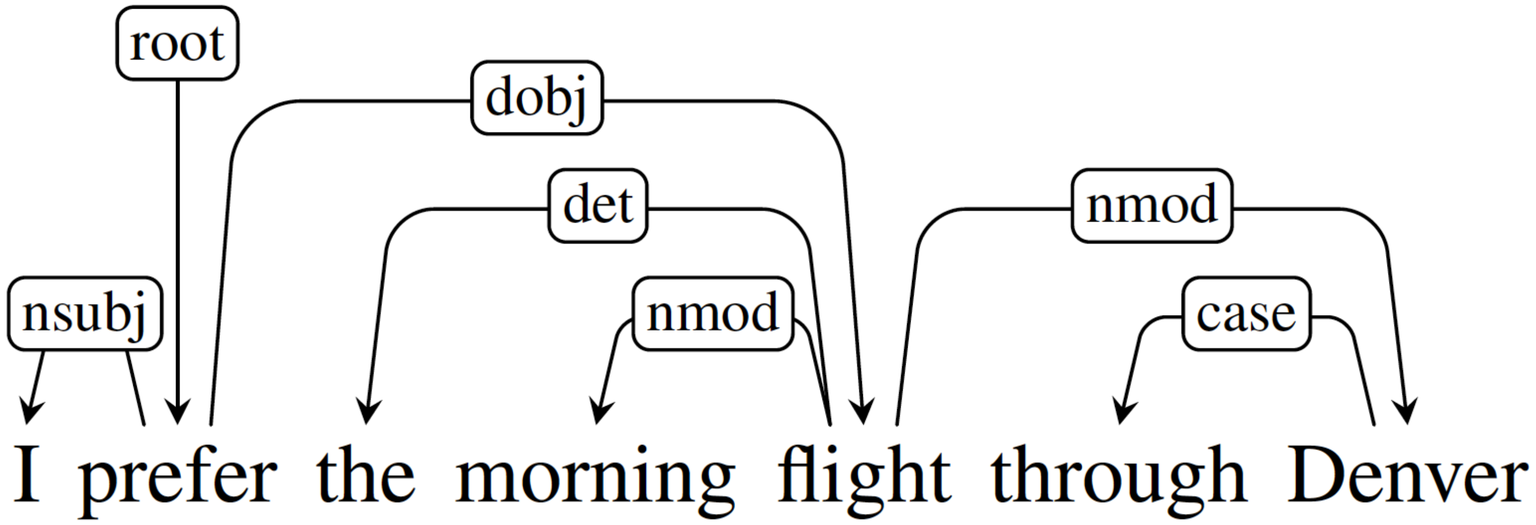
Shuttle veteran and longtime NASA executive Fred Gregory appointed to board

Dependency vs. Constituency

- Dependency is often more useful in practice (models predicate-argument structure)
- Slightly different representational choices:
  - PP attachment is better modeled under dependency
  - Coordination is better modeled under constituency
- Dependency parsers are easier to build: no “grammar engineering”, no unaries, easier to get structured discriminative models working well
- Dependency parsers are usually faster
- Dependencies are more universal cross-lingually

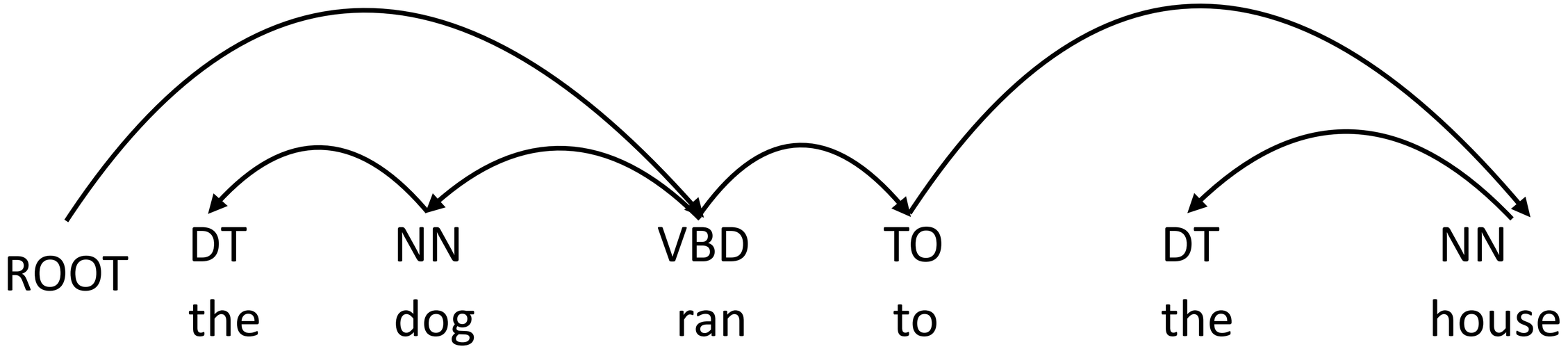
Dependency Syntax

- Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations ("arrows") called dependencies



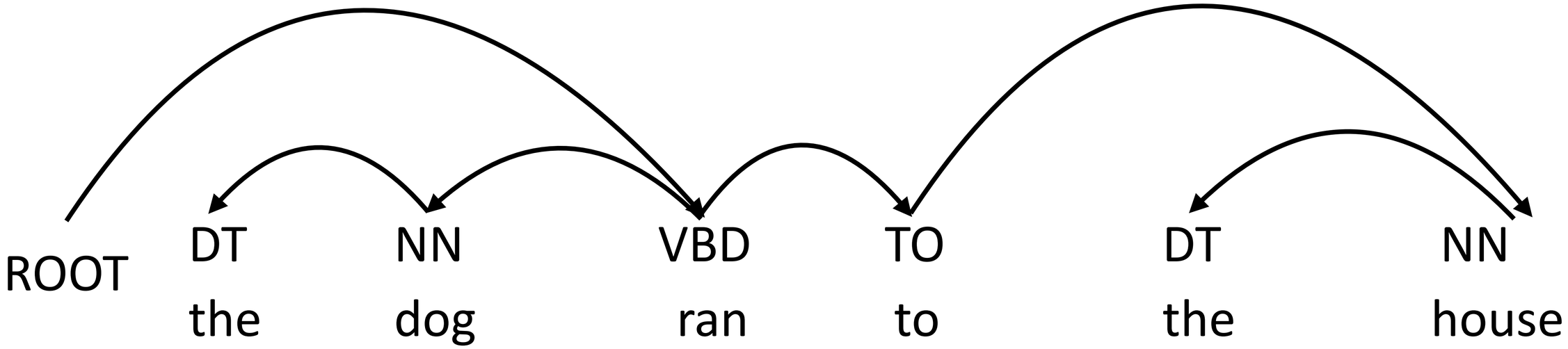
Dependency Syntax

- Dependency syntax: syntactic structure is defined by these arrows/arcs
  - Head (parent, governor) connected to dependent (child, modifier)



Dependency Syntax

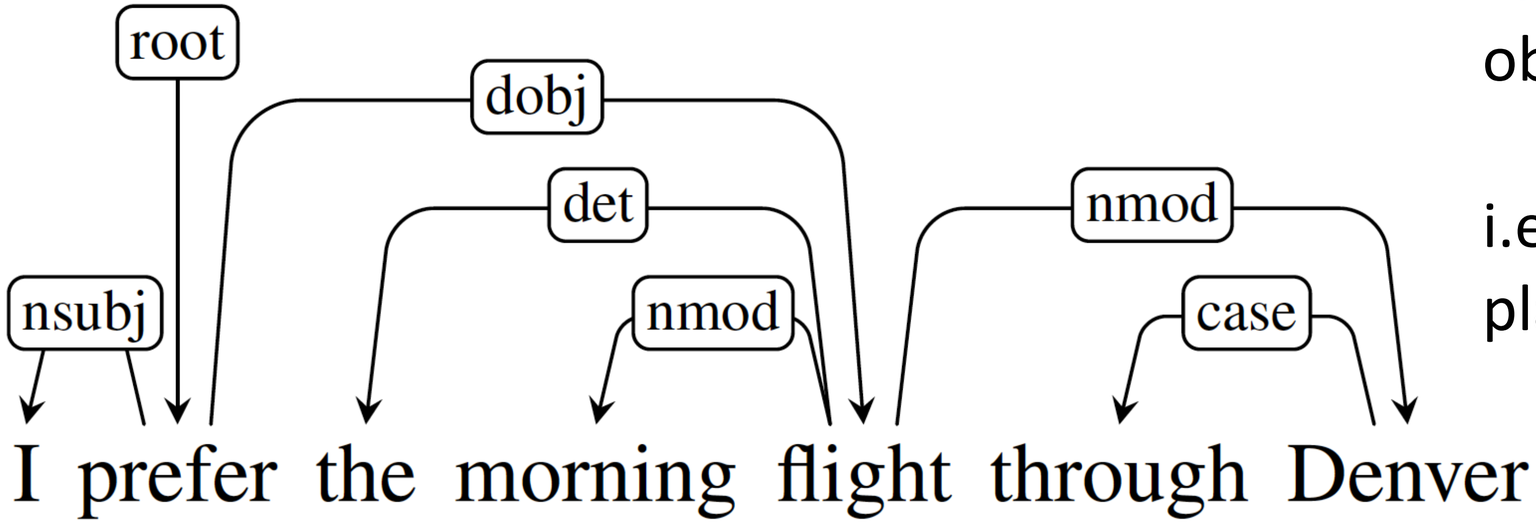
- Dependency syntax: syntactic structure is defined by these arcs
  - Head (parent, governor) connected to dependent (child, modifier)
  - Each word has exactly one parent except for the ROOT symbol, dependencies must form a directed acyclic graph (i.e., tree)



- POS tags same as before, usually run a tagger first as preprocessing

Dependency Syntax

- Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations ("arrows") called dependencies



The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)

i.e., the role that the dependent plays with respect to its head

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

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

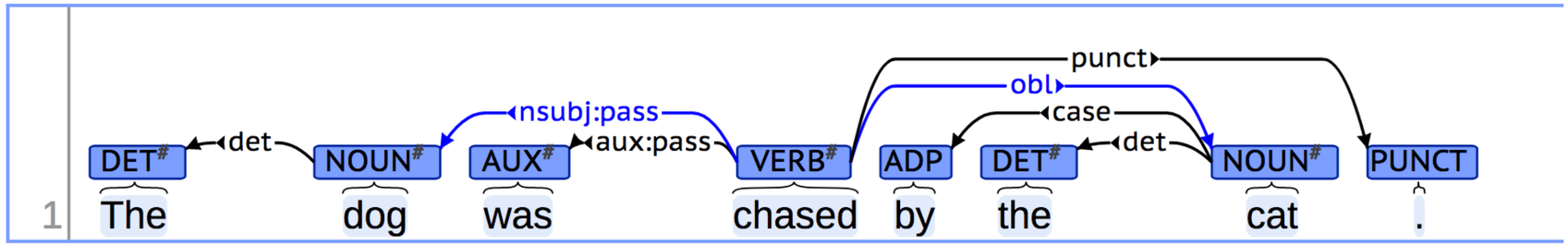
Figure 15.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

Universal Dependencies provide an inventory of dependency relations
that are linguistically motivated, computationally useful, and cross-linguistically applicable

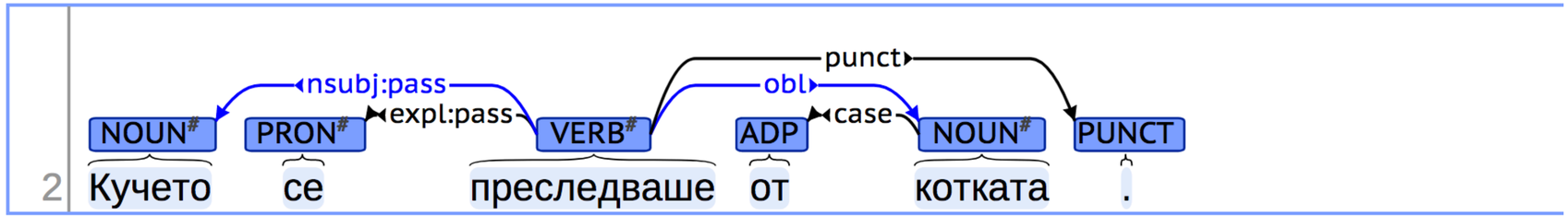
Universal Dependencies

- Annotate dependencies with the same representation in many languages

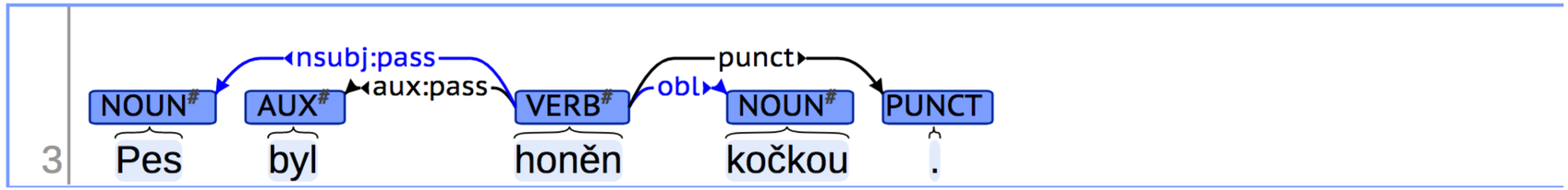
English



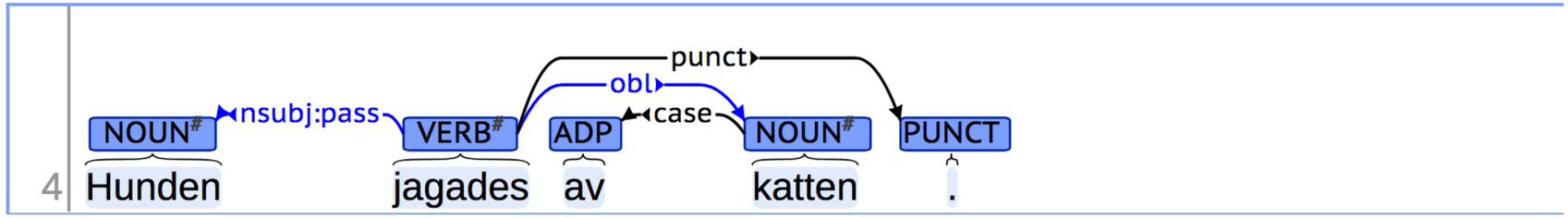
Bulgarian



Czech



Swiss



http://universaldependencies.org/

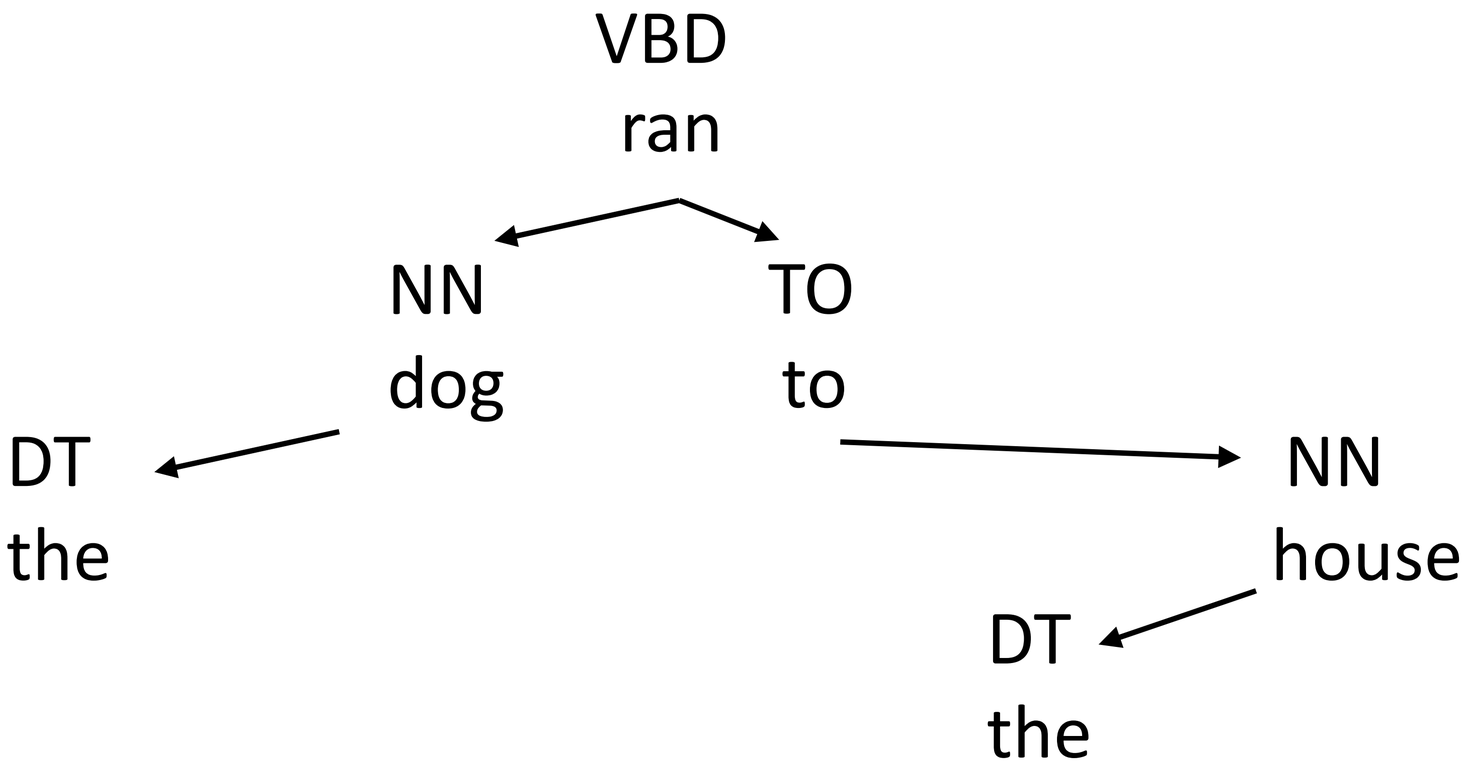
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET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

Figure 15.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

Core sets of frequently used relations: clausal relations that describe syntactic roles with respect to a predicate (often a verb), and modifier relations that categorize the ways that words that can modify their heads.

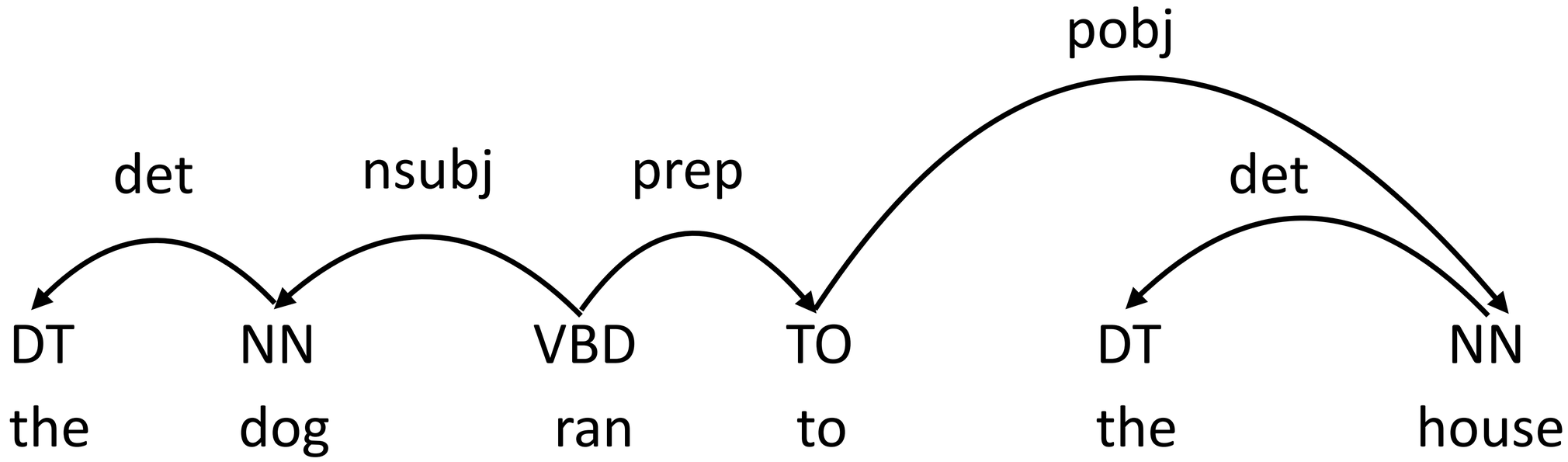
Dependency Syntax

- Still a notion of hierarchy! Subtrees often align with constituents



Dependency Parsing

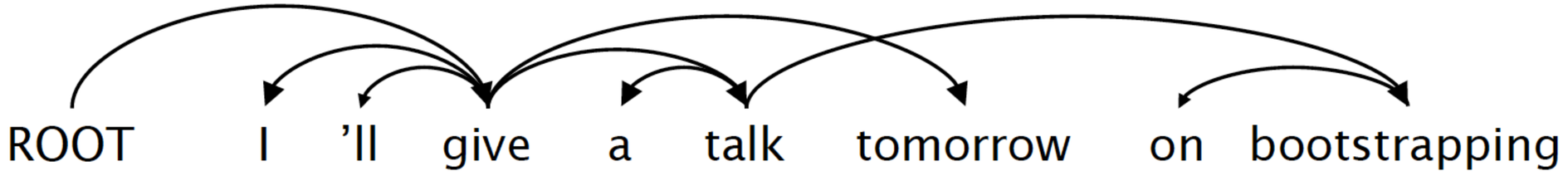
- Can label dependencies according to syntactic function
- Major source of ambiguity is in the structure, so we focus on that more (labeling separately with a classifier works pretty well)



Dependency Parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of

- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don't want cycles A → B, B → A
- This makes the dependencies a tree
- Final issue is whether arrows can cross (non-projective) or not

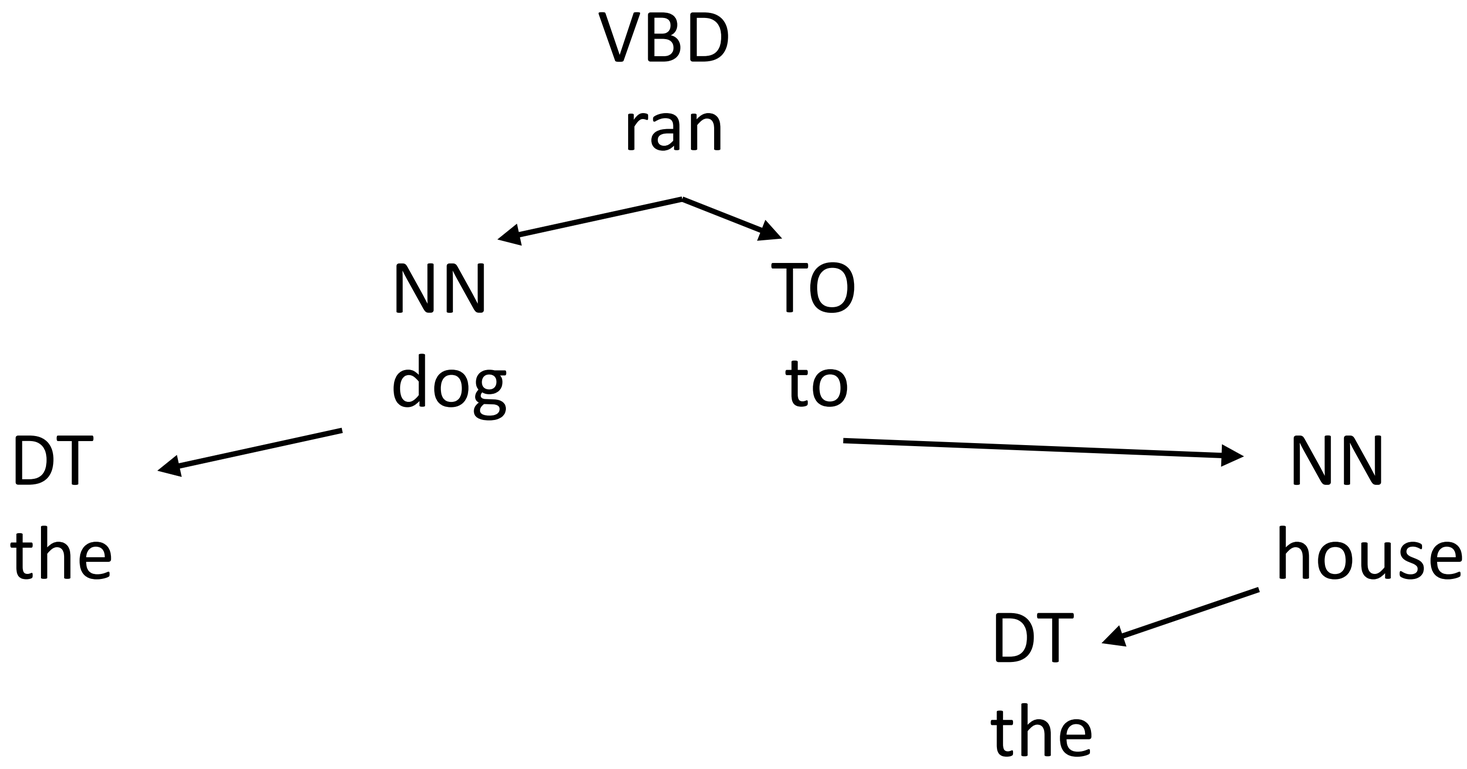


Outline

- Dependency representation (in contrast with constituency)
- Projectivity
- Transition-based dependency parsing

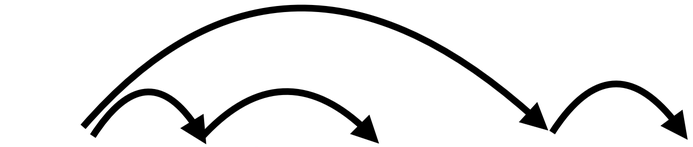
Projectivity

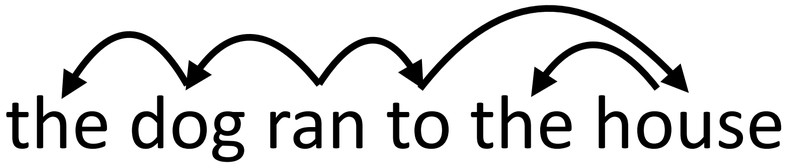
- Any subtree is a contiguous span of the sentence <-> tree is *projective*



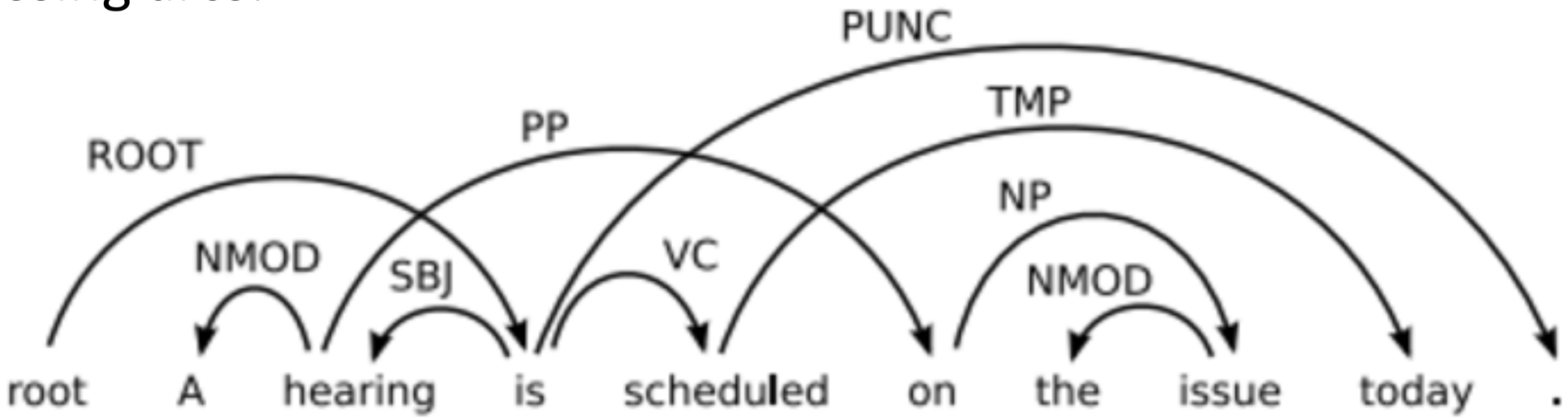
Projectivity

- Projective <-> no “crossing” arcs





- Crossing arcs:



credit: Language Log

Projectivity

- Number of trees produceable under different formalisms

	Arabic	Czech	Danish
Projective	1297 (88.8)	55872 (76.8)	4379 (84.4)
Sentences	1460	72703	5190

- Many trees in other languages are nonprojective

Pitler et al. (2013)

Projectivity

- Number of trees produceable under different formalisms

	Arabic	Czech	Danish
1-Endpoint-Crossing	1457 (99.8)	71810 (98.8)	5144 (99.1)
Well-nested, block degree 2	1458 (99.9)	72321 (99.5)	5175 (99.7)
Gap-Minding	1394 (95.5)	70695 (97.2)	4985 (96.1)
Projective	1297 (88.8)	55872 (76.8)	4379 (84.4)
Sentences	1460	72703	5190

- Many trees in other languages are nonprojective
- Some other formalisms (that are harder to parse in), most useful one is 1-Endpoint-Crossing

Pitler et al. (2013)

Outline

- Dependency representation (in contrast with constituency)
- Projectivity
- Transition-based dependency parsing

Transition-based Parsing

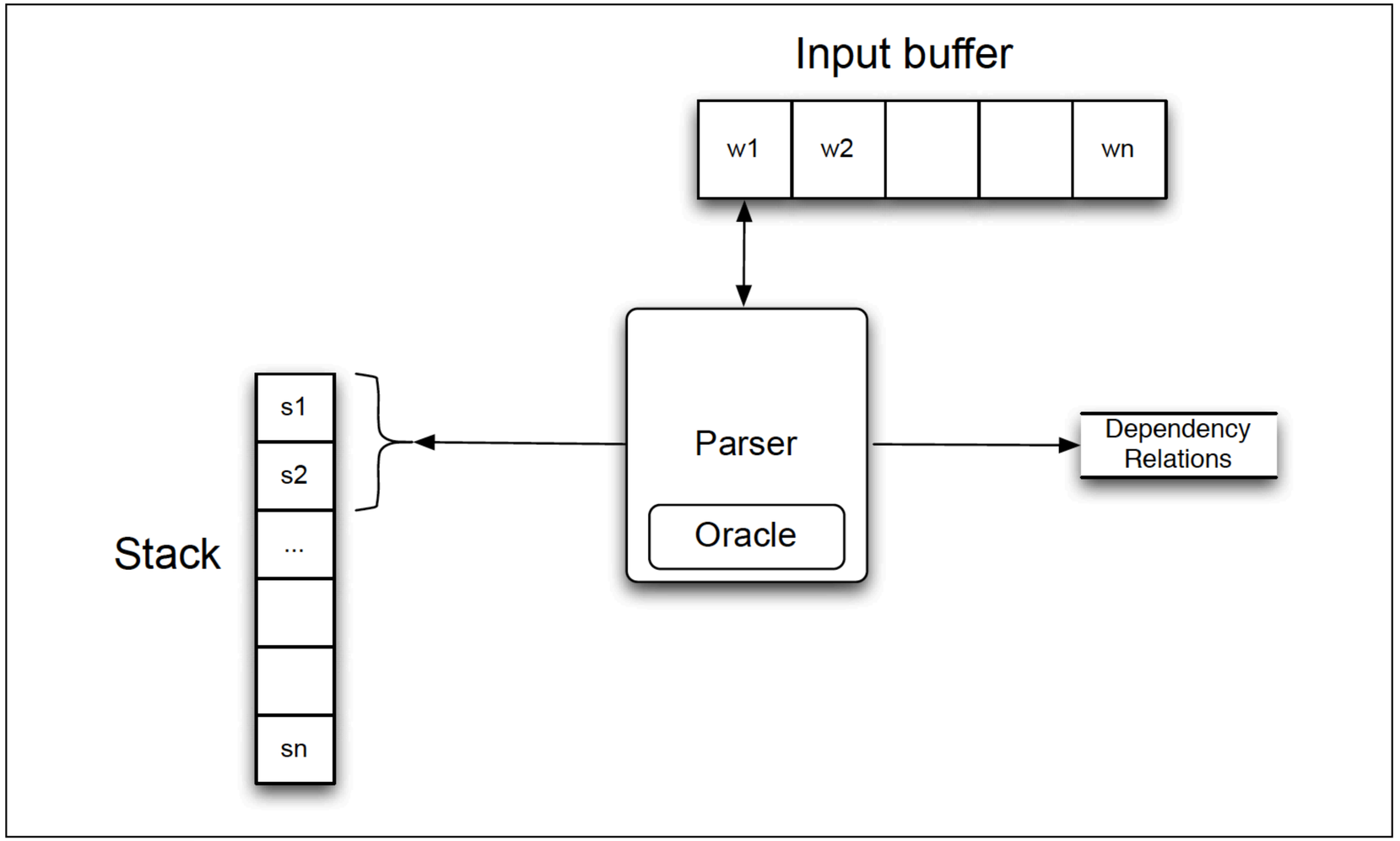


Figure 15.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

The notion of a "configuration":
a **stack**, an i**nput buffer** of words or tokens,
a set of relations representing
a dependency tree

Chapter 15.4 in Jurafsky and Martin textbook:
https://web.stanford.edu/~jurafsky/slp3/15.pdf

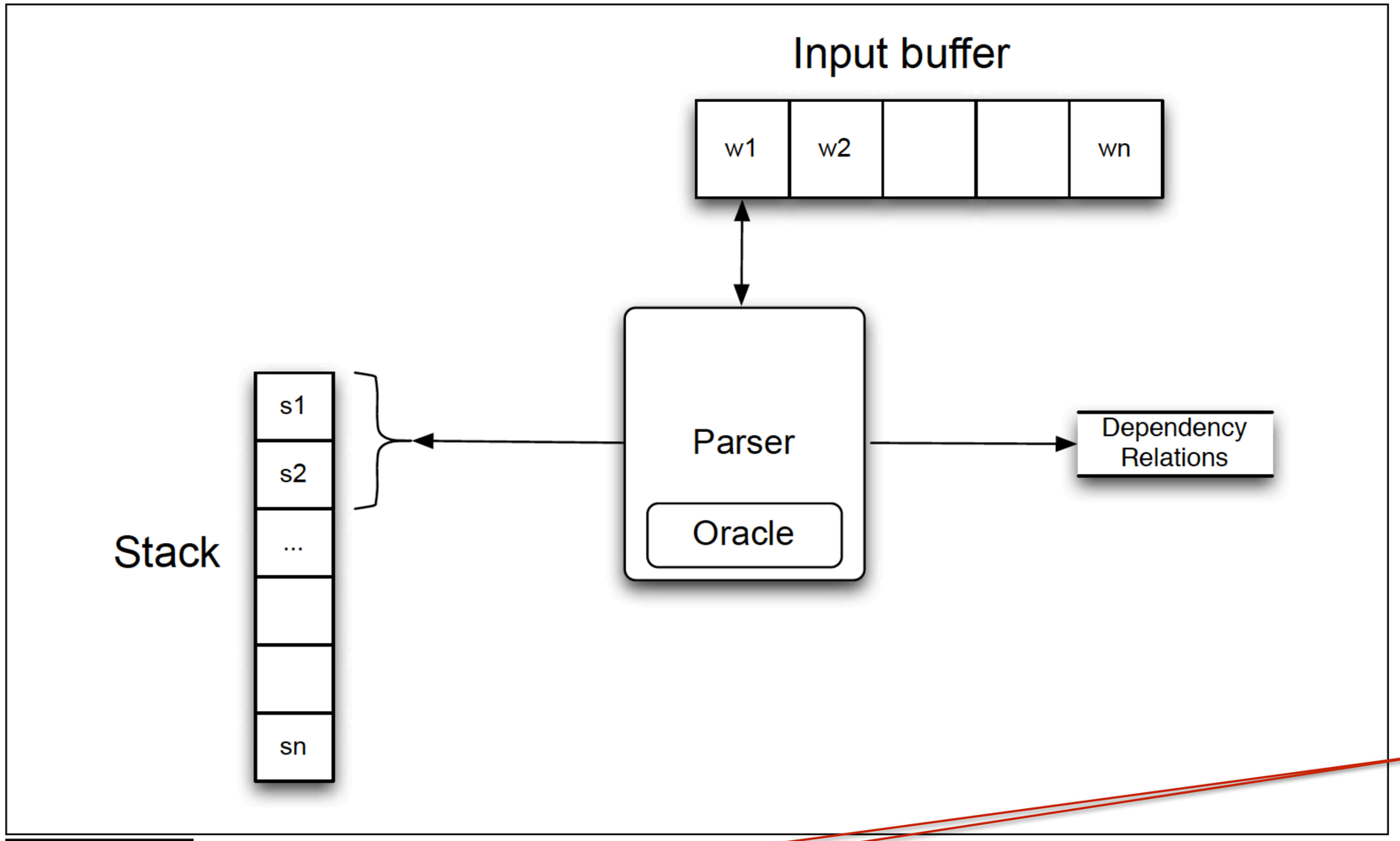


Figure 15.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

The notion of a "configuration":
a **stack**, an i**nput buffer** of words or tokens,
a set of relations representing
a dependency tree

1. LeftArc
2. RightArc
3. Shift

Chapter 15.4 in Jurafsky and Martin textbook:
https://web.stanford.edu/~jurafsky/slp3/15.pdf

- LEFTARC: Assert a head-dependent relation between the word at the top of the stack and the word directly beneath it; remove the lower word from the stack.
- RIGHTARC: Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack;
- SHIFT: Remove the word from the front of the input buffer and push it onto the stack.

arc standard approach

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Figure 15.7 Trace of a transition-based parse.

- LEFTARC: Assert a head-dependent relation between the word at the top of the stack and the word directly beneath it; remove the lower word from the stack.
- RIGHTARC: Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack;
- SHIFT: Remove the word from the front of the input buffer and push it onto the stack.

arc standard approach

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Figure 15.7 Trace of a transition-based parse.

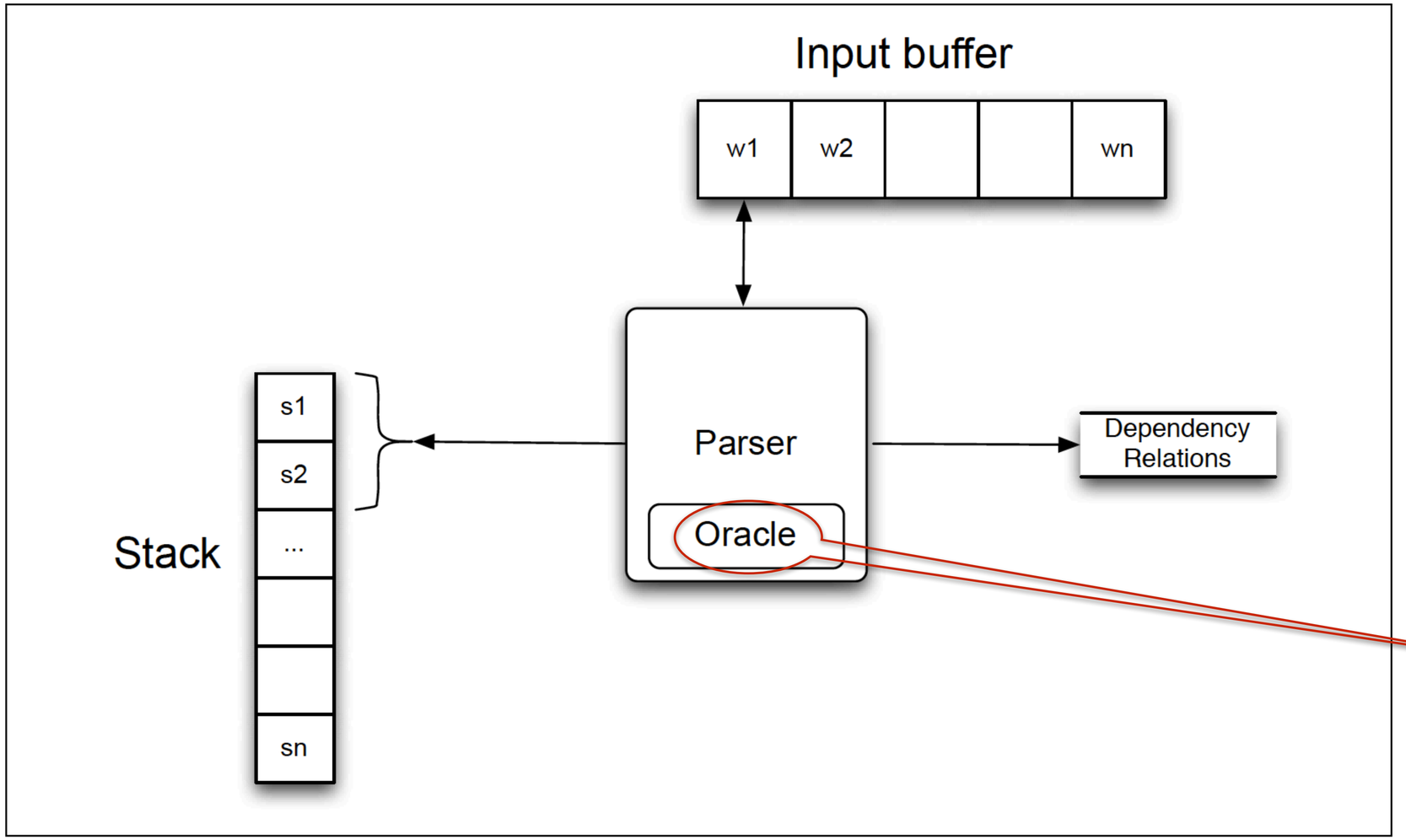


Figure 15.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

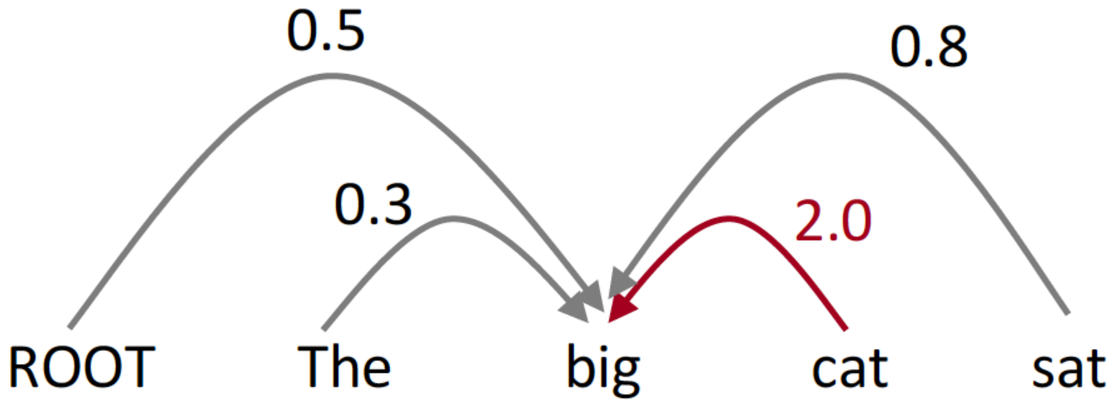
The notion of a "configuration":
a **stack**, an i**nput buffer** of words or tokens,
a set of relations representing
a dependency tree

classifiers based on supervised learning algorithm (logistic regression, SVM, deep neural networks, etc.)

Chapter 15.4 in Jurafsky and Martin textbook:
https://web.stanford.edu/~jurafsky/slp3/15.pdf

Graph-based dependency parsers

- Compute a score for every possible dependency for each edge
  - Then add an edge from each word to its highest-scoring candidate head
  - And repeat the same process for each other word



e.g., picking the head for “big”

slides from cs224n-2019, Manning, Stanford

A Neural graph-based dependency parser

[Dozat and Manning 2017; Dozat, Qi, and Manning 2017]

- Revived graph-based dependency parsing in a neural world
  - Design a biaffine scoring model for neural dependency parsing
    - Also using a neural sequence model, as we discuss next week
- Really great results!
  - But slower than simple neural transition-based parsers
    - There are n^2 possible dependencies in a sentence of length n

Method	UAS	LAS (PTB WSJ SD 3.3
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79
Dozat & Manning 2017	**95.74**	**94.08**

slides from cs224n-2019, Manning, Stanford

Evaluating Dependency Parsing

- UAS: unlabeled attachment score. Accuracy of choosing each word's parent (*n* decisions per sentence)
- LAS: additionally consider label for each edge
- Log-linear CRF parser, decoding with Eisner algorithm: 91 UAS
- Higher-order features from Koo parser: 93 UAS
- Best English results with neural CRFs (Dozat and Manning): 95-96 UAS

Takeaways

- Dependency formalism provides an alternative to constituency, particularly useful in how portable it is across languages
- Dependency parsing also has efficient dynamic programs for inference
- CRFs + neural CRFs (again) work well