Amirkabir University of Technology
(Tehran Polytechnic)

Natural Language Processing

Lecture 15: Dependency Parsing

Amirkabir University of Technology
Dr Momtazi

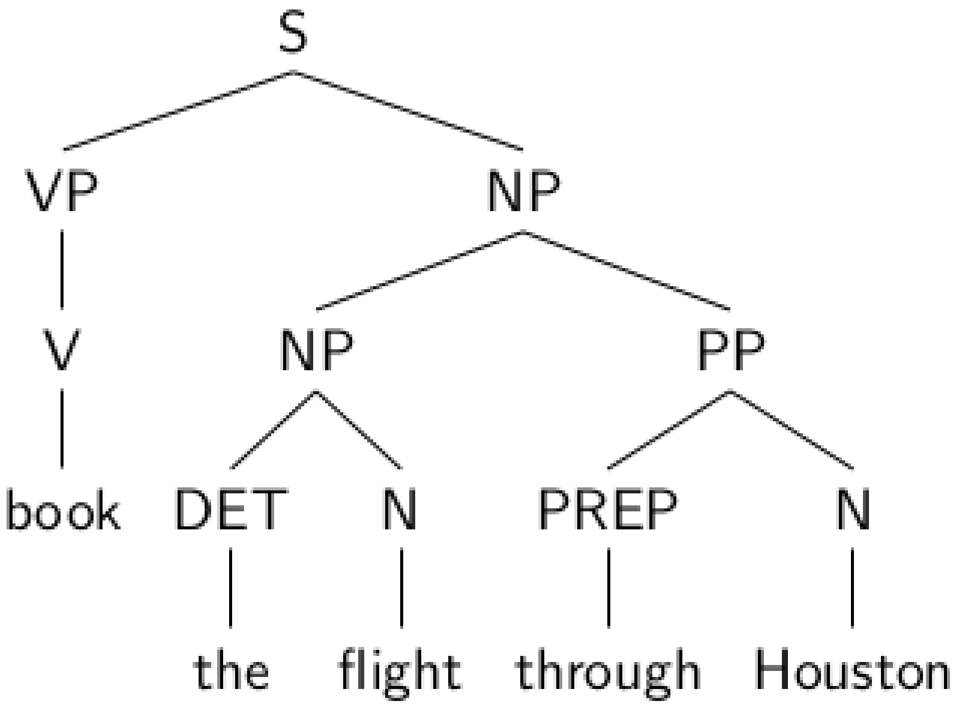
Parsing

- Finding structural relationship between words in a sentence

Outline

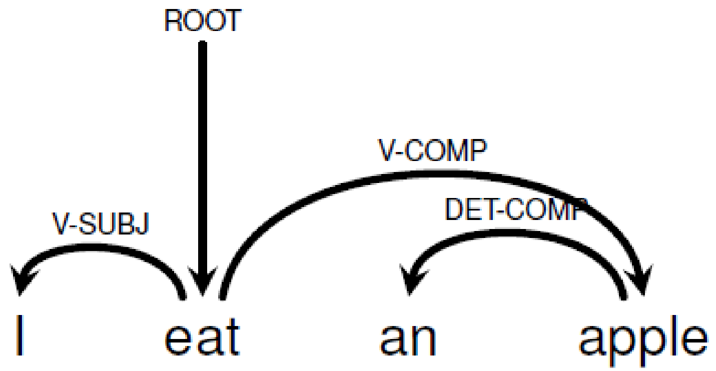
- **Constituency vs. Dependency**
- Dependency Relations and Formalisms
- Dependency Parsing
- Evaluation

Constituency

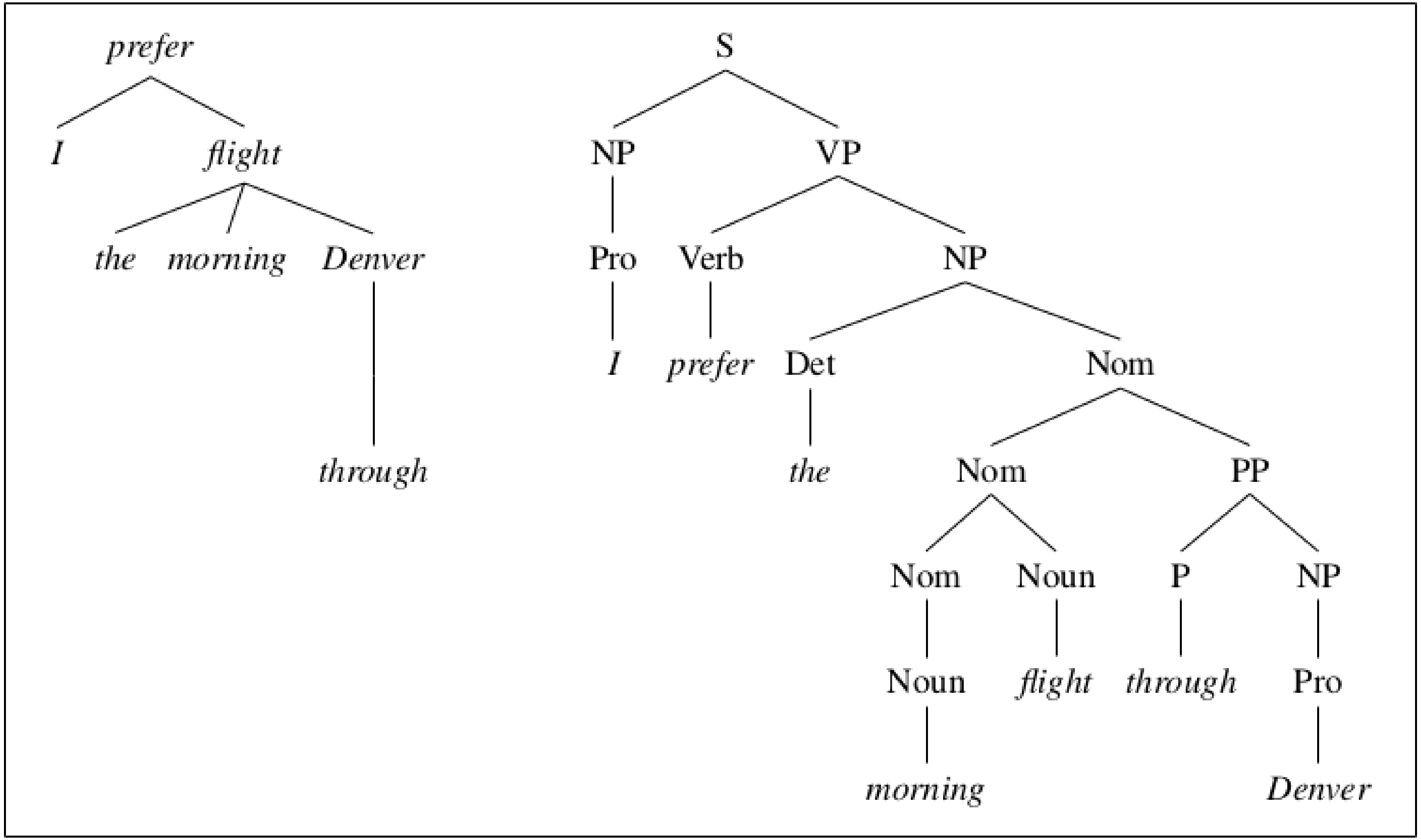


Dependency

- Identifying which words depend on (modify or arguments of) which other words



Constituency vs. Dependency

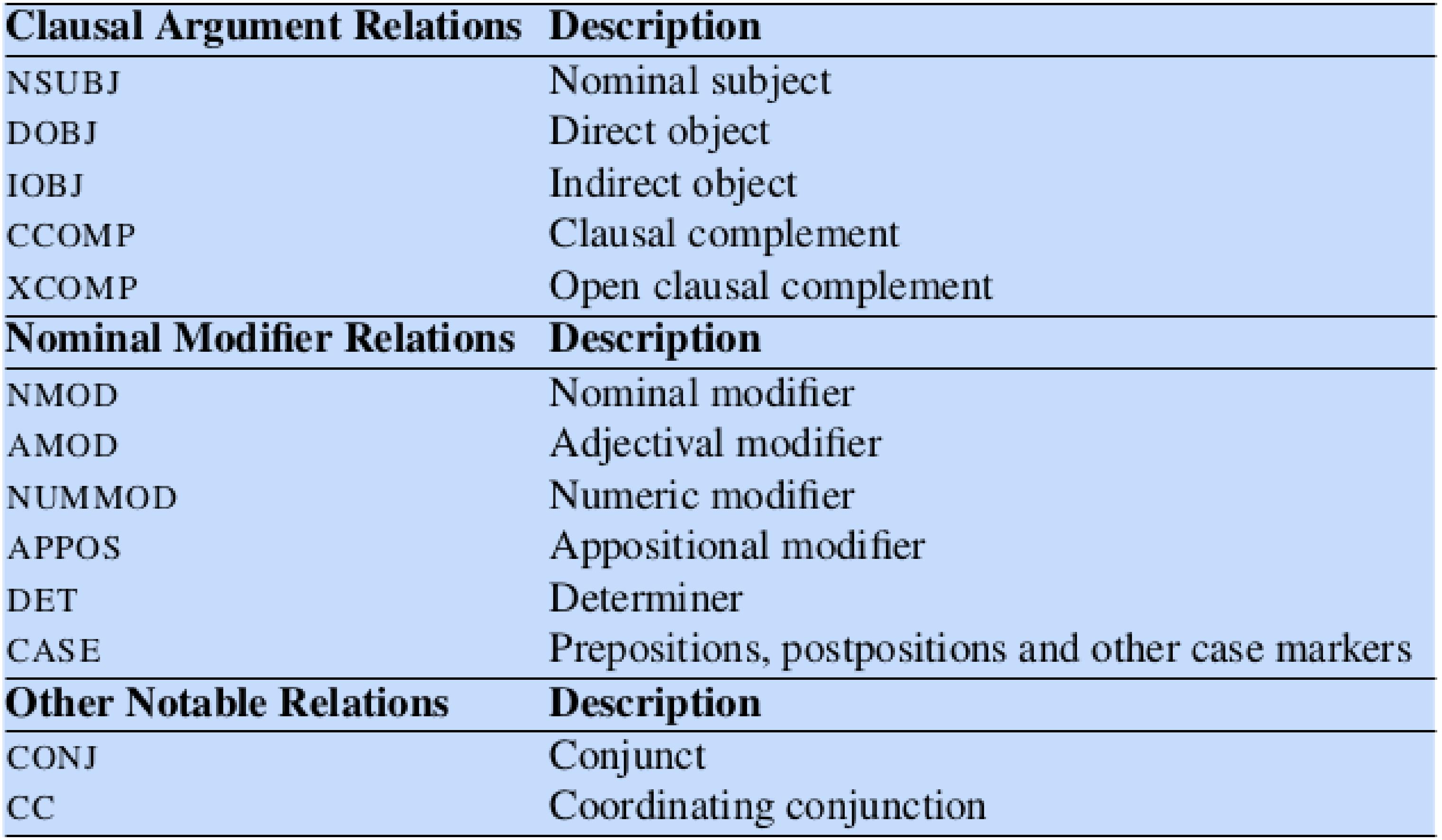


Outline

- Constituency vs. Dependency
- **Dependency Relations and Formalisms**
- Dependency Parsing
- Evaluation

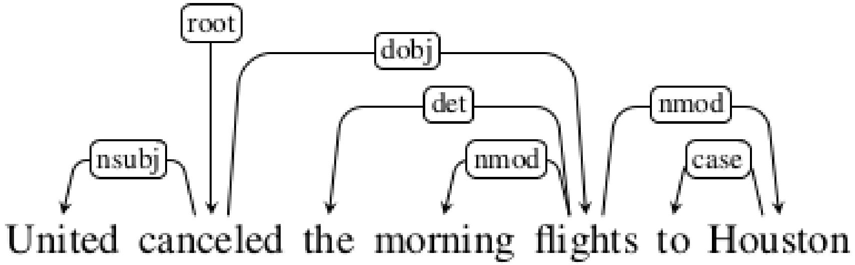
Dependency Relations

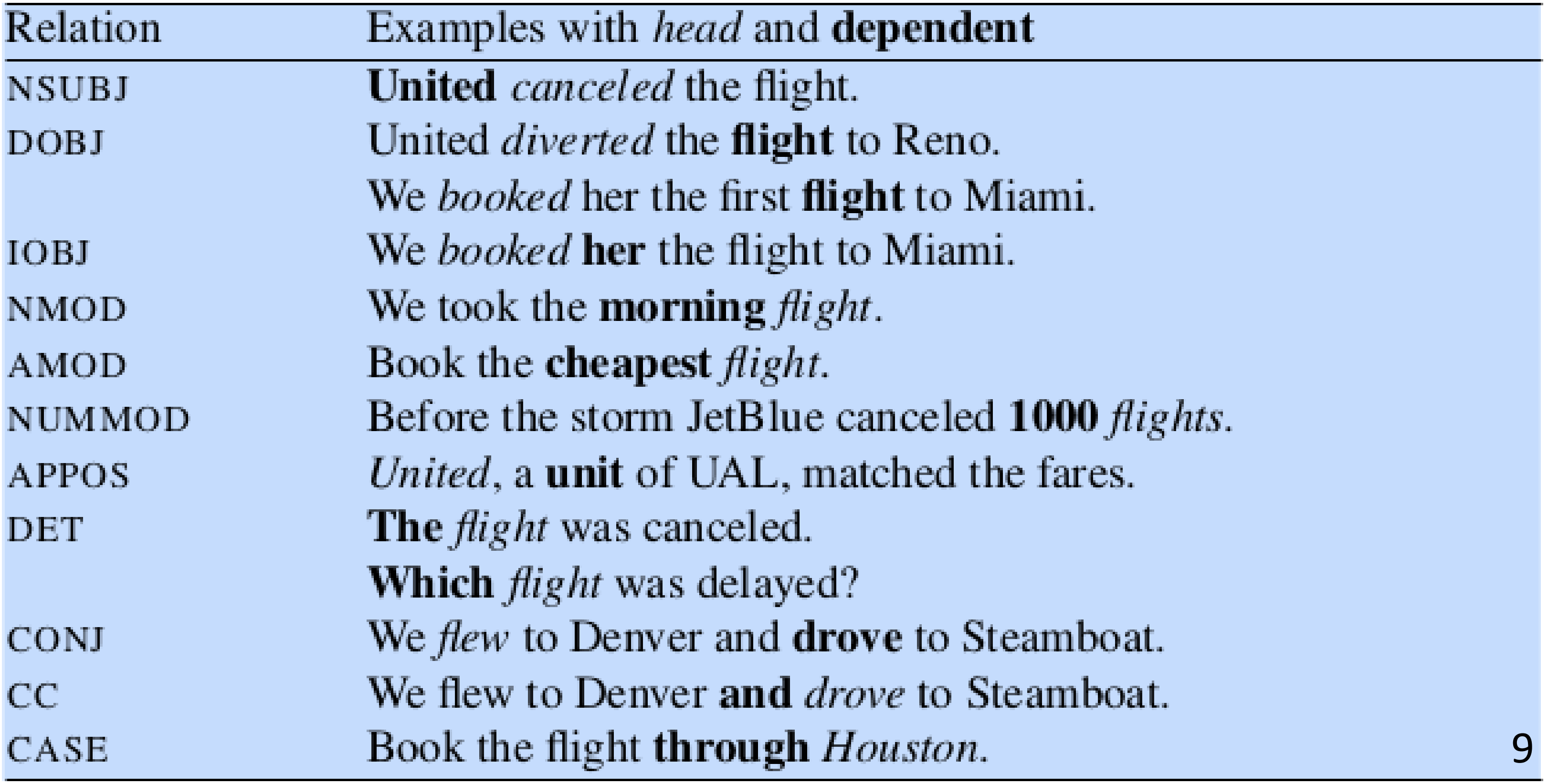
- The traditional linguistic notion of grammatical relation provides the basis for the binary relations that comprise these dependency structures.
- The arguments to these relations consist of a head and a dependent.



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

Dependency Relations





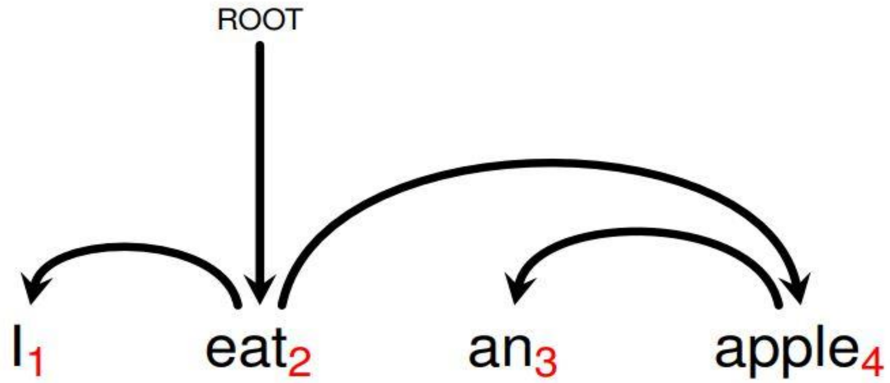
Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
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 Formalisms

- A dependency structure can be represented as a directed graph G=(V;A), consisting of a set of vertices V, and a set of ordered pairs of vertices A (arcs).
  - The set of vertices, V, corresponds exactly to the set of words in a given sentence.
  - The set of arcs, A, captures the head dependent and grammatical function relationships between the elements in V.

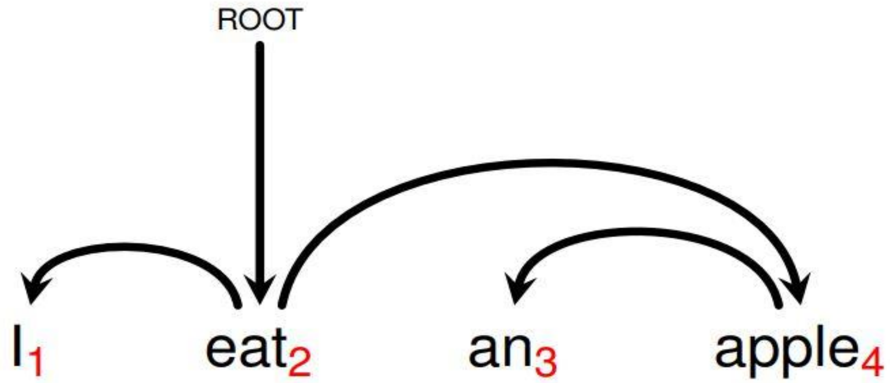
Unlabeled Dependency Parses

- A dependency tree is a directed graph that satisfies the following constraints:
  - There is a single designated root node that has no incoming arcs. It is used to point the head of the sentence
  - With the exception of the root node, each vertex has exactly one incoming arc. (Each dependency connect a head word h to a modifier m)
  - There is a unique path from the root node to each vertex in V.



Unlabeled Dependency Parses

- Dependencies in the following example are as follows:
  - (0,2)
  - (2,1)
  - (2,4)
  - (4,3)

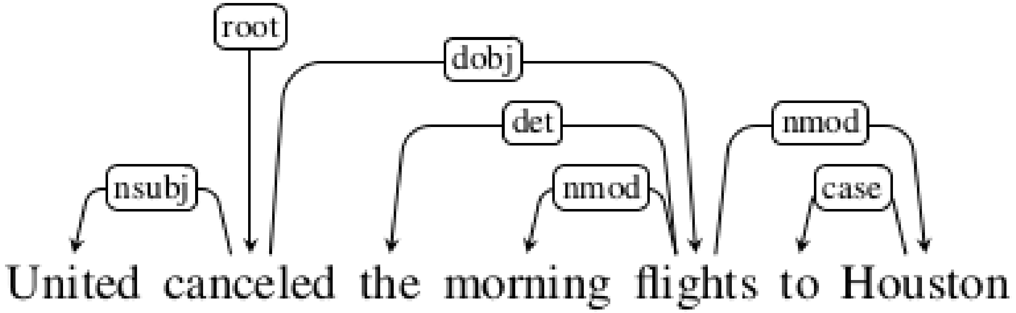


Projectivity

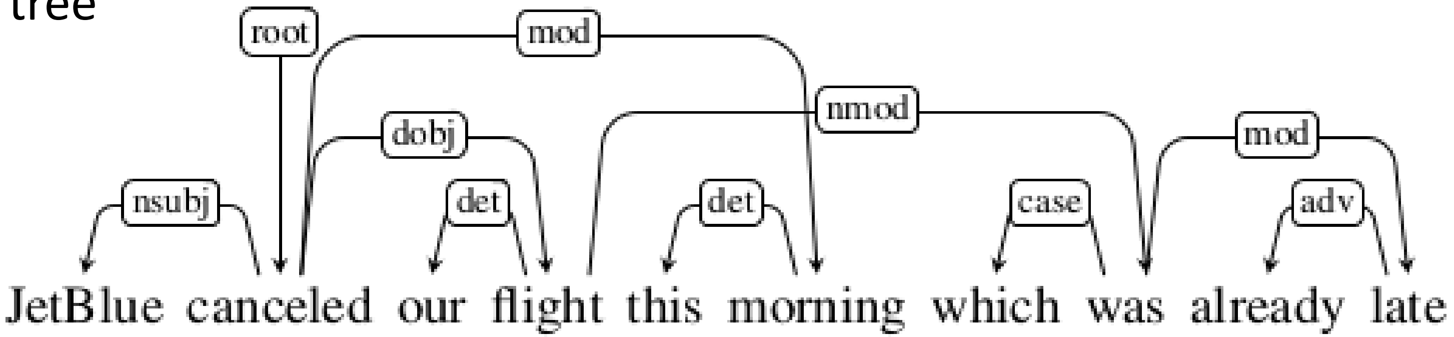
- The notion of projectivity imposes an additional constraint that is derived from the order of the words in the input.
- An arc from a head to a dependent is said to be projective if there is a path from the head to every word that lies between the head and the dependent in the sentence.
- A dependency tree is then said to be projective if all the arcs that make it up are projective (There is no crossing dependencies).

Projectivity

- Example
  - Projective dependency tree



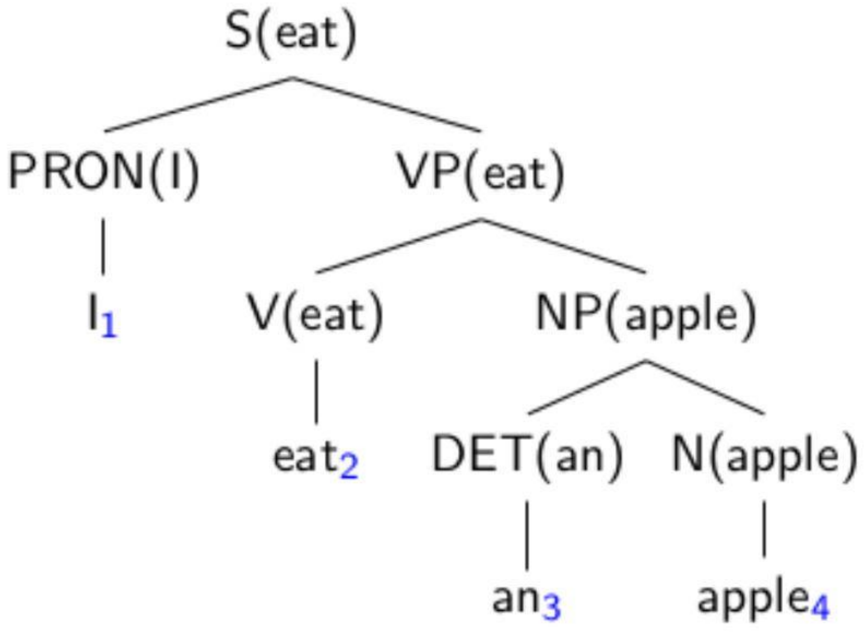
  - Non-projective dependency tree


root
mod
nmod
dobj
mod
nsubj
det
det
case
adv
JetBlue canceled our flight this morning which was already late

Dependency Treebanks

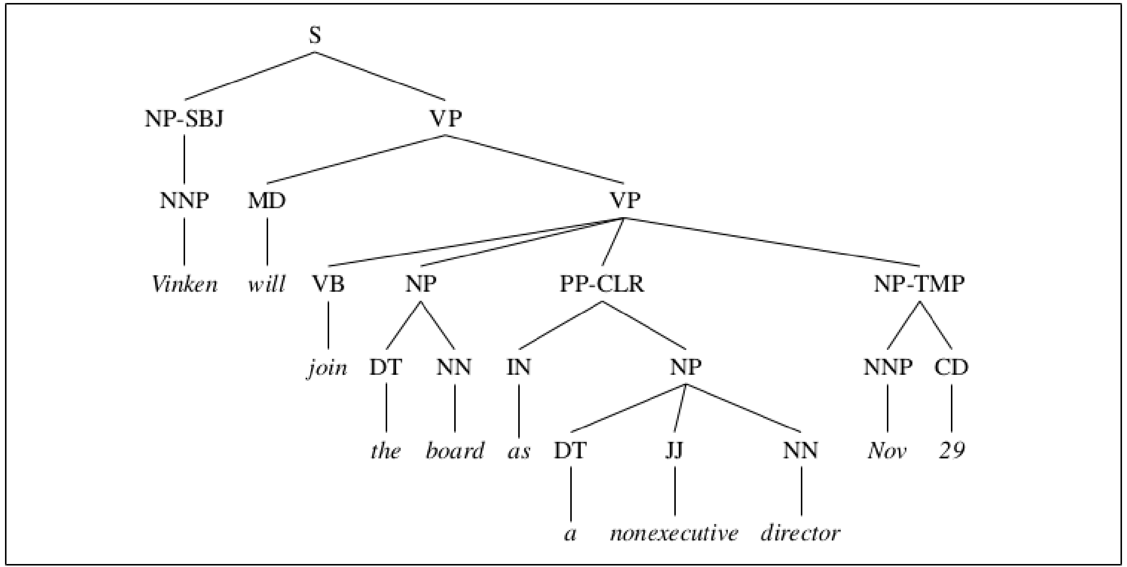
- For some languages, there exist “Dependency banks”; e.g., Czech
- If no dependency bank is available for a language, it is possible to extract it from a constituency-based treebank:
  - Mark the head child of each node in a phrase structure, using the appropriate head rules.
  - In the dependency structure, make the head of each non-head child depend on the head of the head-child.

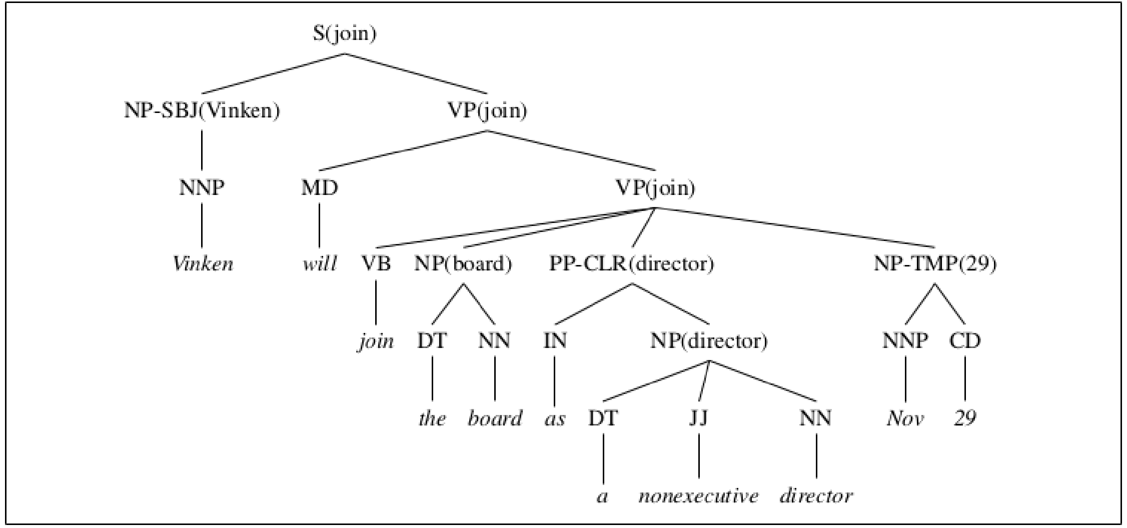
Dependency Treebanks



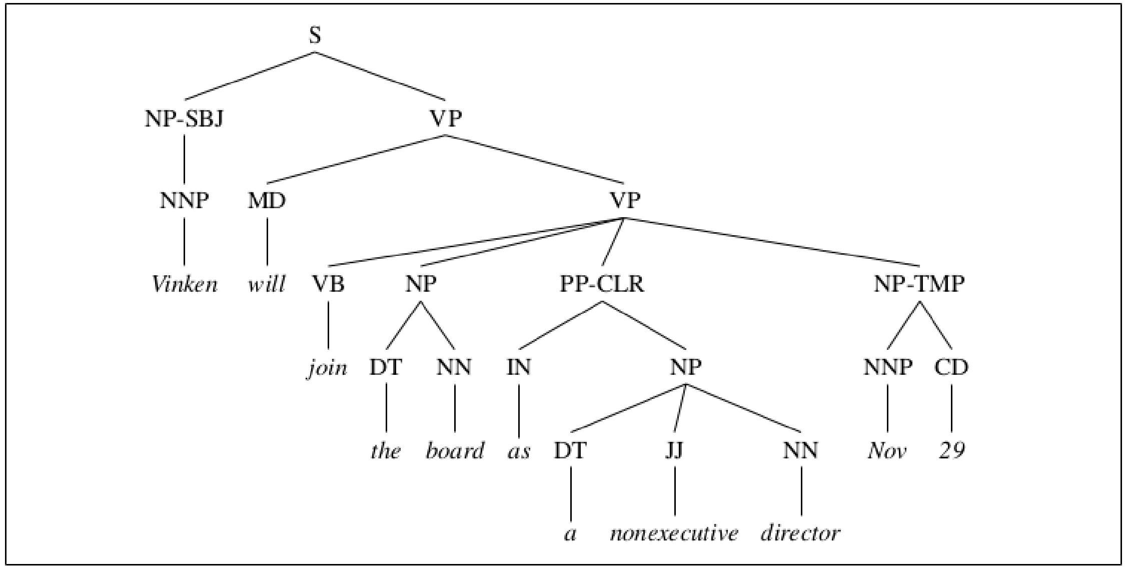
- Unlabeled Dependencies:
  - (0,2) root → eat
  - (2,1) eat → I
  - (2,4) eat → apple
  - (4,3) apple → an

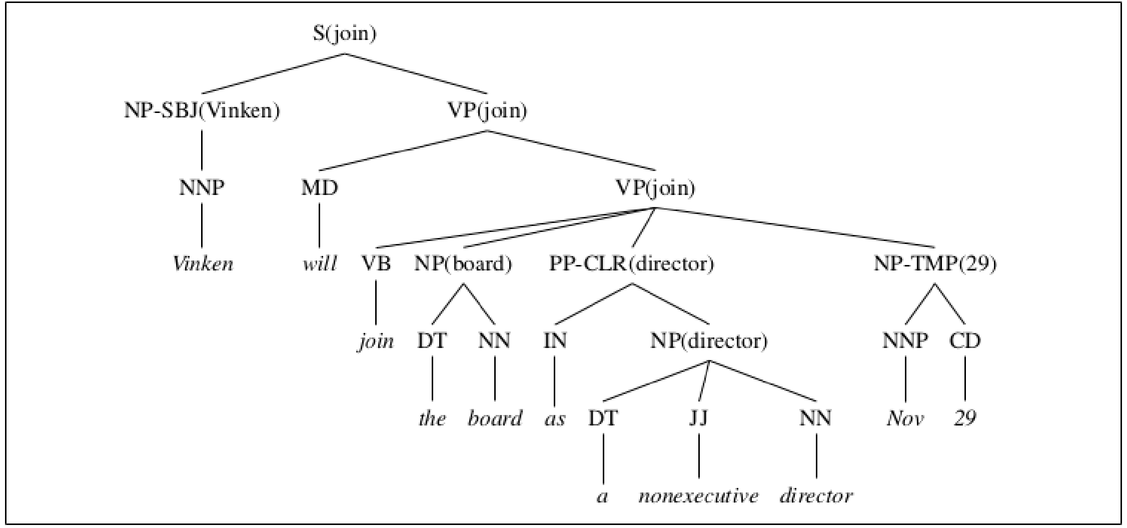
Dependency Treebanks

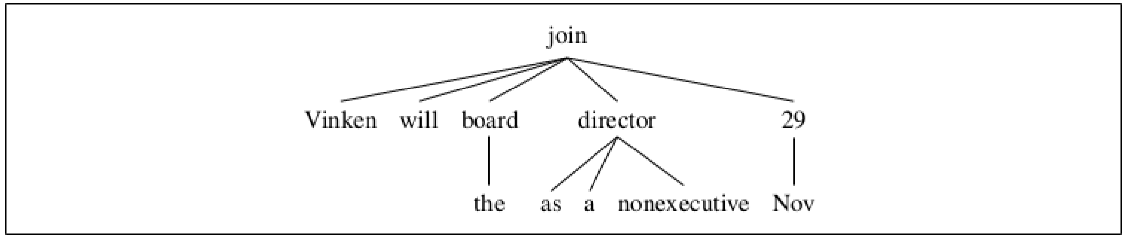




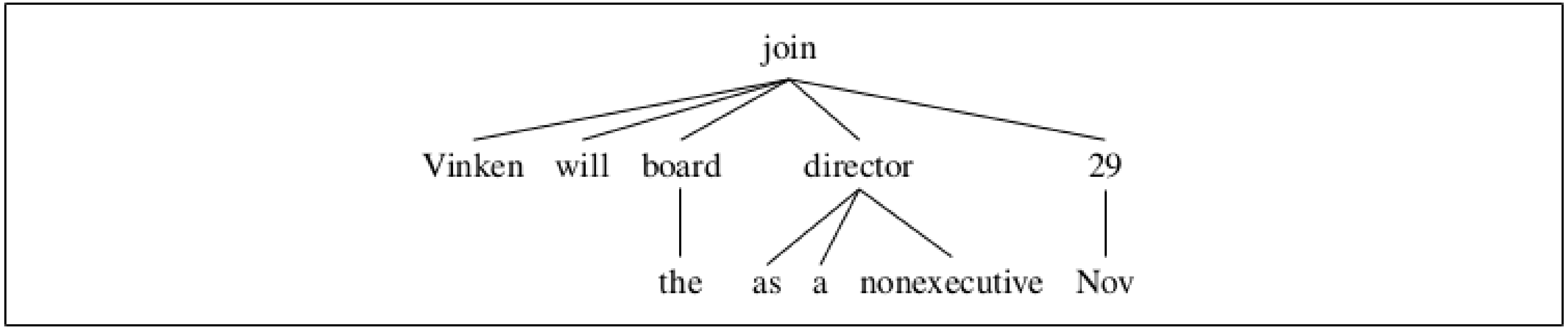
Dependency Treebanks



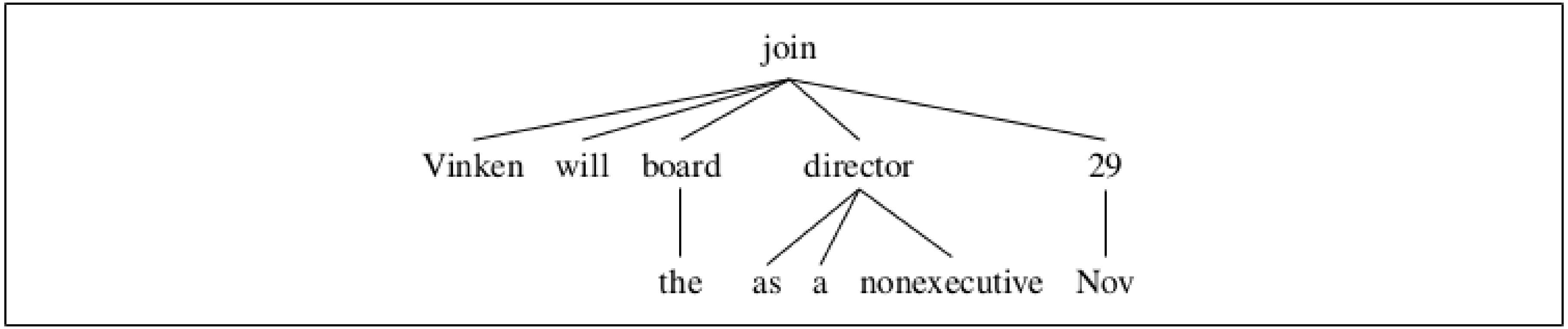


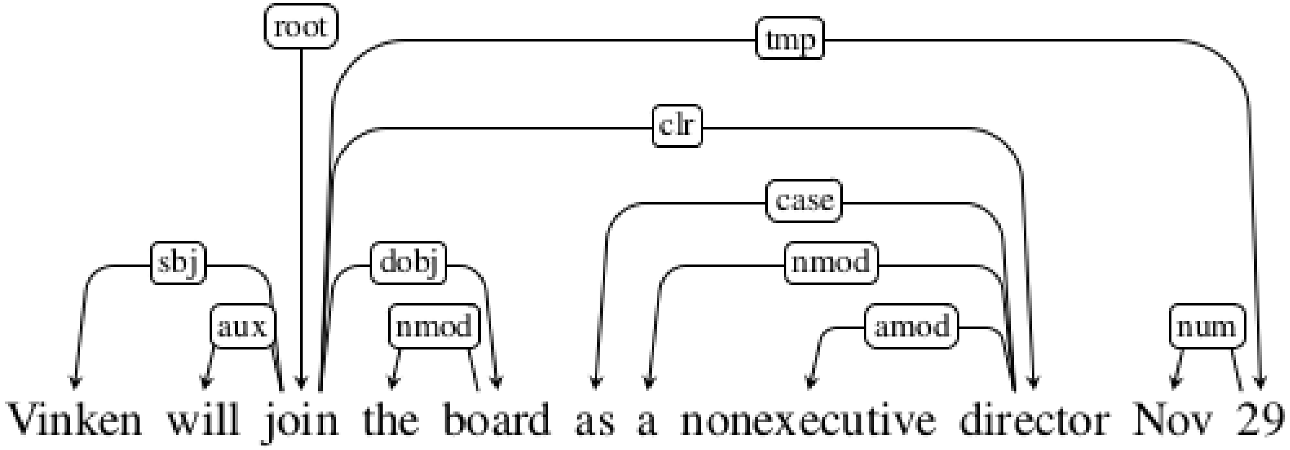


Dependency Treebanks



Dependency Treebanks



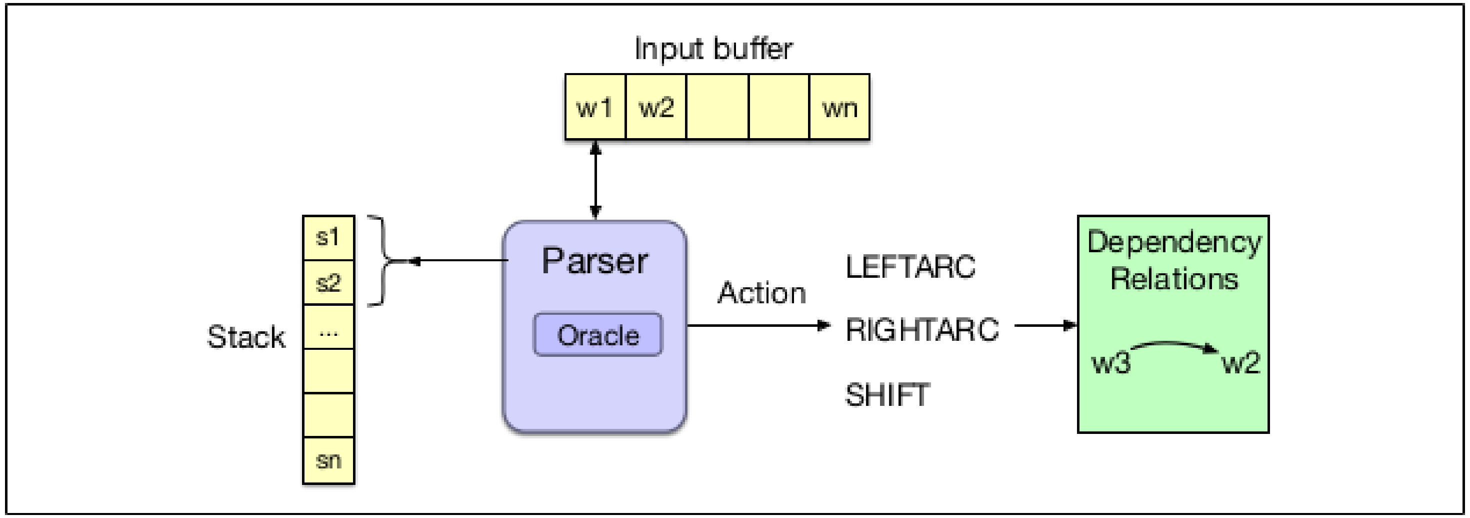

root
tmp
clr
case
sbj
dobj
nmod
aux
nmod
amod
num
Vinken will join the board as a nonexecutive director Nov 29

Outline

- Constituency vs. Dependency
- Dependency Relations and Formalisms
- **Dependency Parsing**
    - **Transition-based Dependency Parsing**
    - Sequence Modeling
- Evaluation

Transition-based Dependency Parsing

- Using on shift-reduce parsing, a paradigm originally developed for compilers
- Main elements:
  - A stack on which we build the parse
  - A buffer of tokens to be parsed
  - A parser which takes actions on the parse via a predictor called an oracle

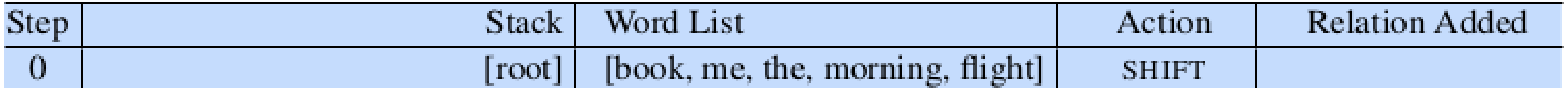


Transition-based Dependency Parsing

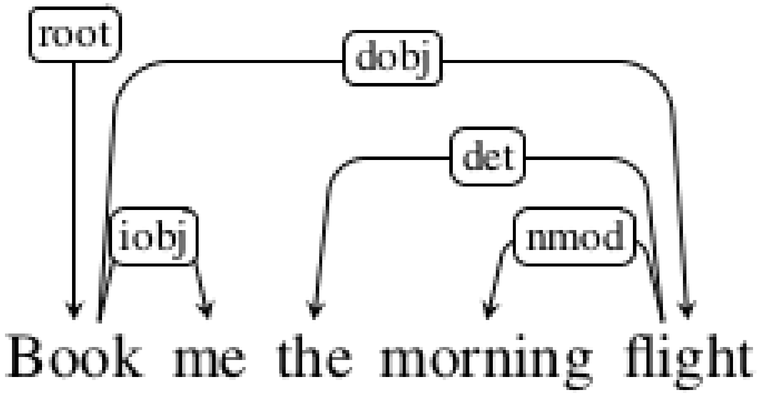
- The parser walks through the sentence left-to-right, successively shifting items from the buffer onto the stack.
- At each point we examine the top two elements on the stack, and the oracle makes a decision about what transition to apply to build the parse.
- The possible transitions:
  - LEFTARC: Assigning the current word as the head of some previously seen word
  - RIGHTARC: Assigning some previously seen word as the head of the current word
  - SHIFT: Postponing to deal with the current word, storing it for later processing

- LEFTARC and RIGHTARC are also known as reduce operations, based on a metaphor from shift-reduce parsing

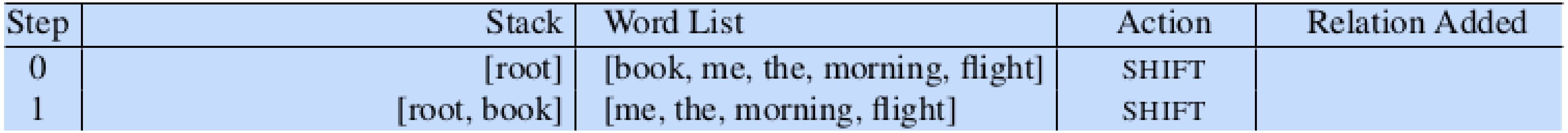
Transition-based Dependency Parsing



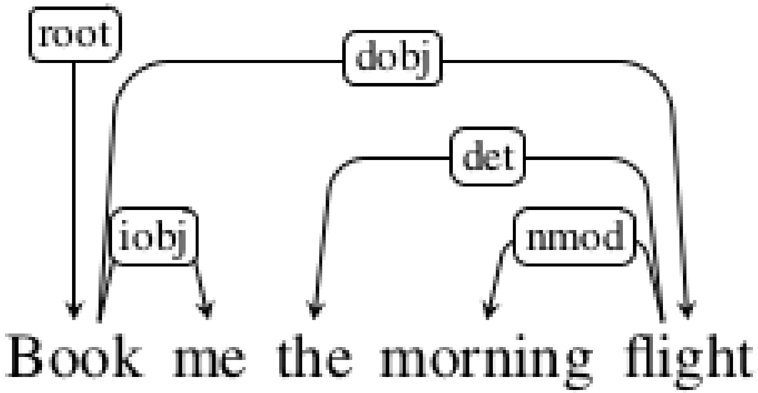
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	



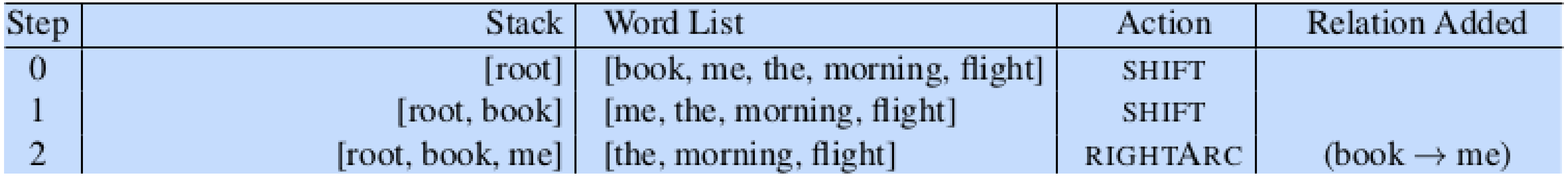
Transition-based Dependency Parsing



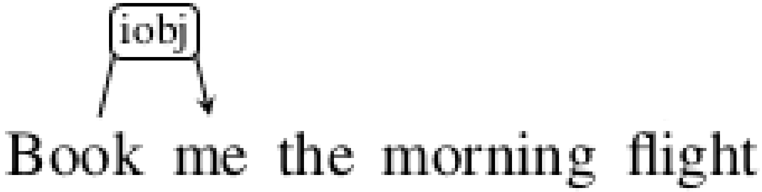
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

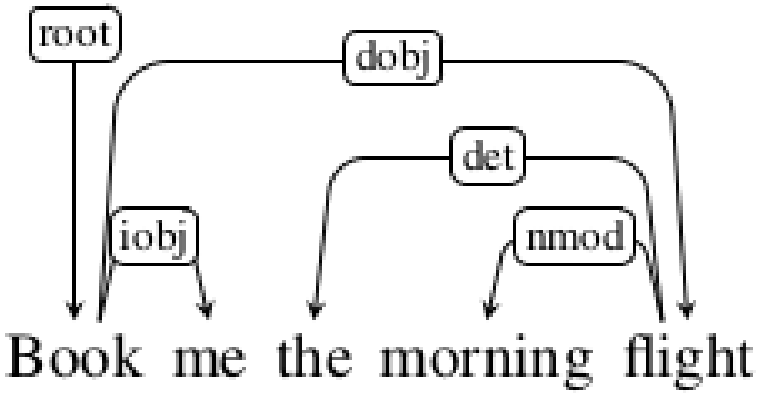


Transition-based Dependency Parsing

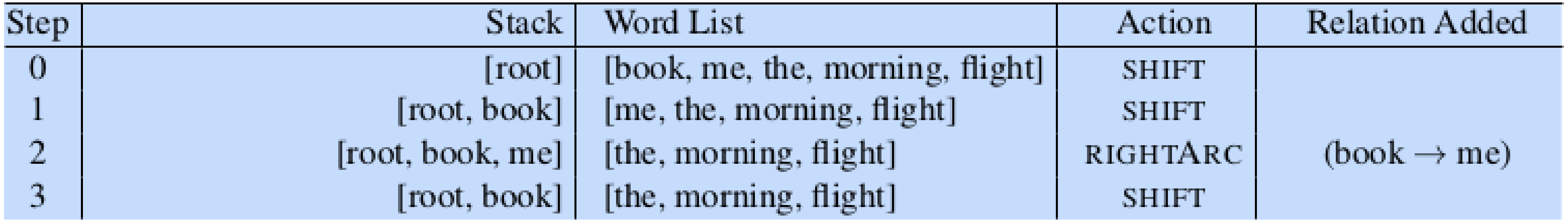


Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)

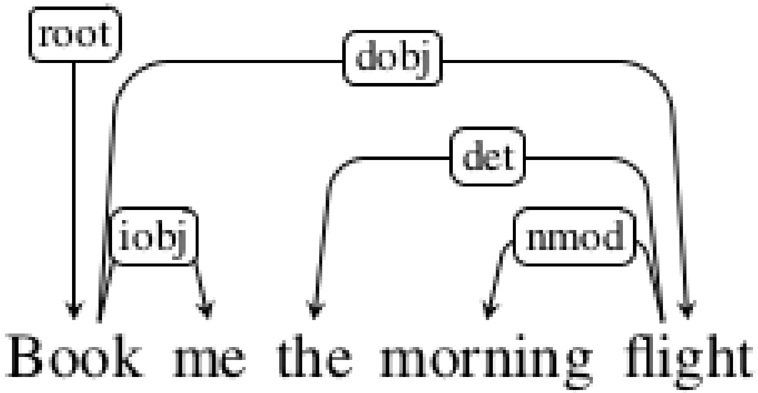




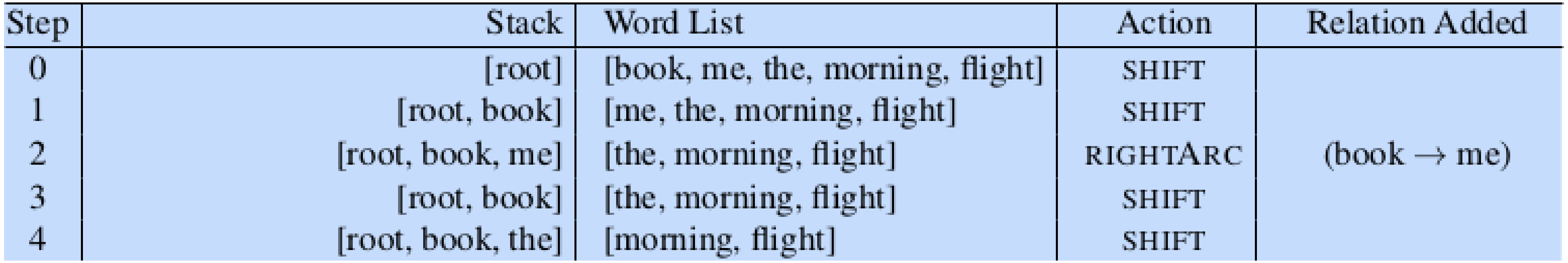
Transition-based Dependency Parsing



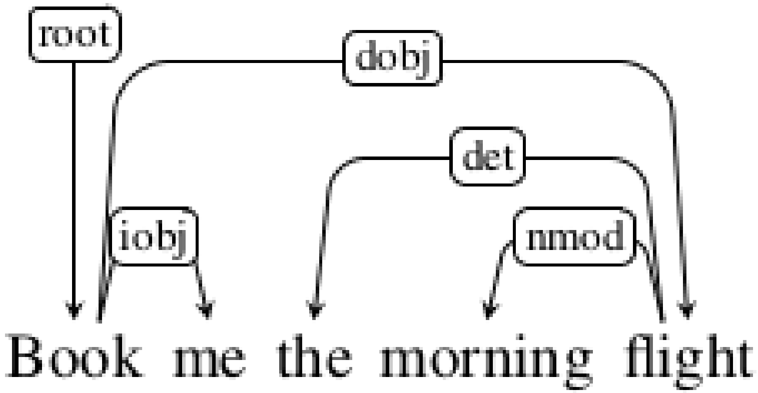
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	



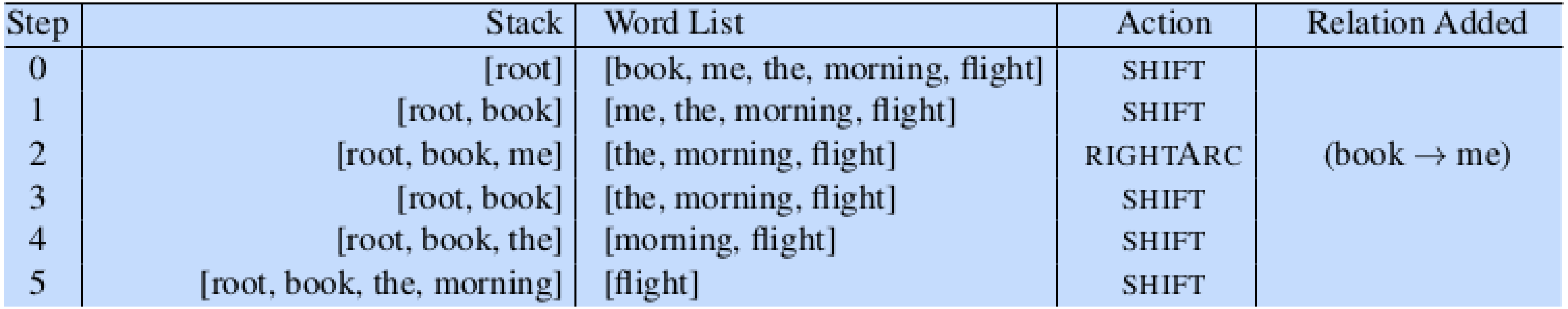
Transition-based Dependency Parsing



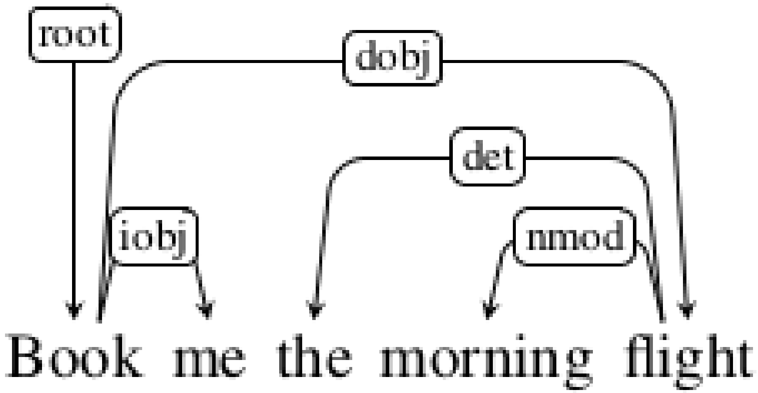
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	



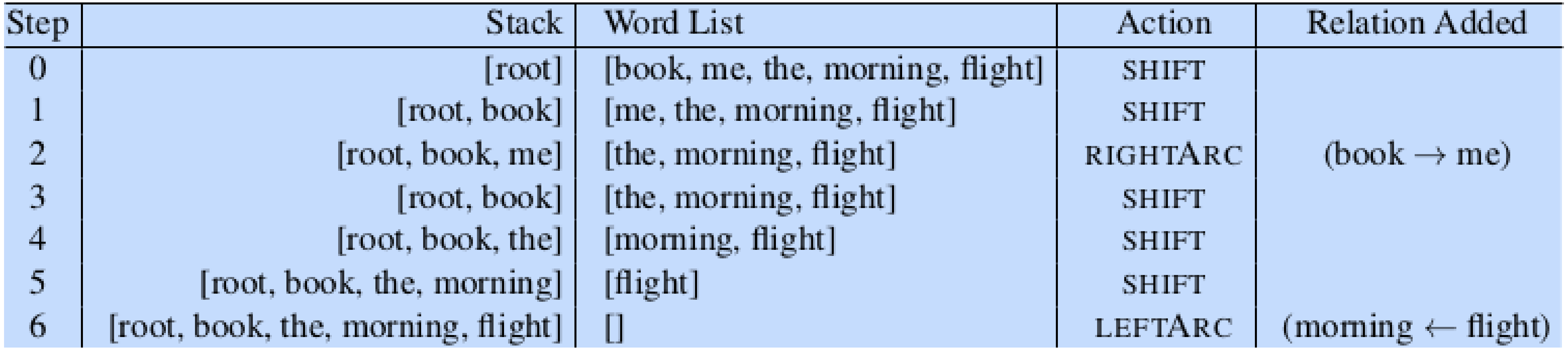
Transition-based Dependency Parsing



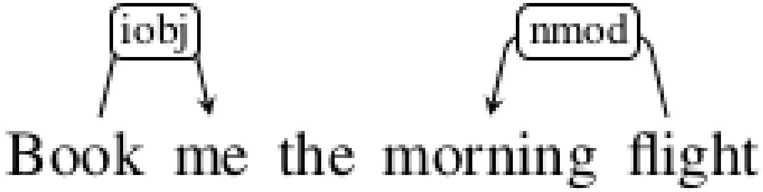
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	

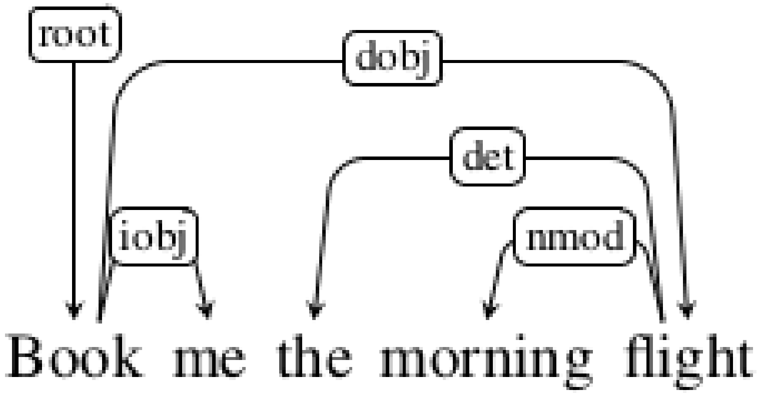


Transition-based Dependency Parsing

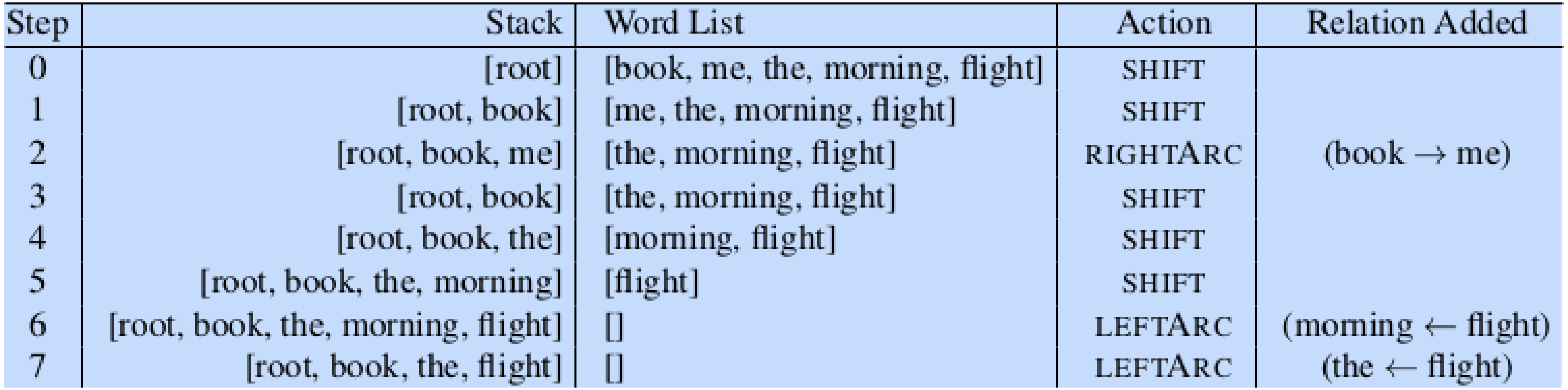


Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[moming, flight]	SHIFT	
5	[root,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

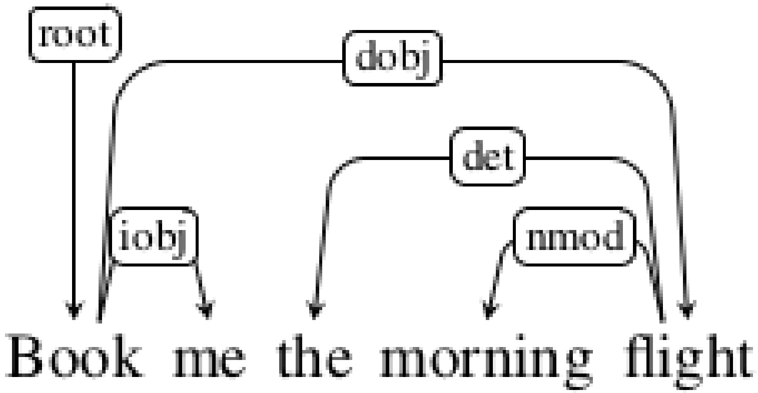




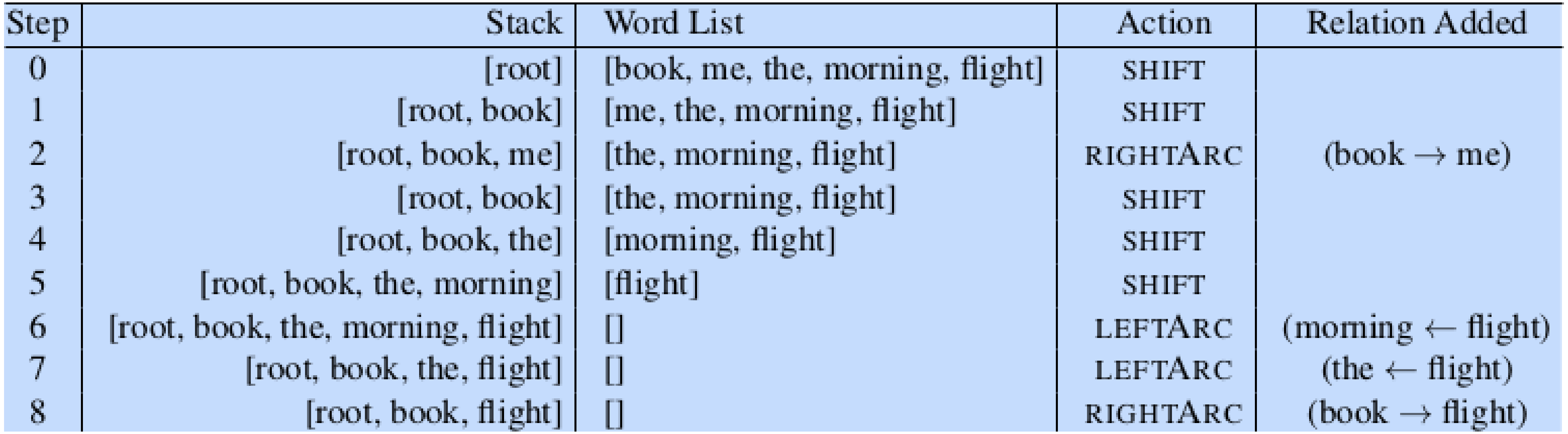
Transition-based Dependency Parsing



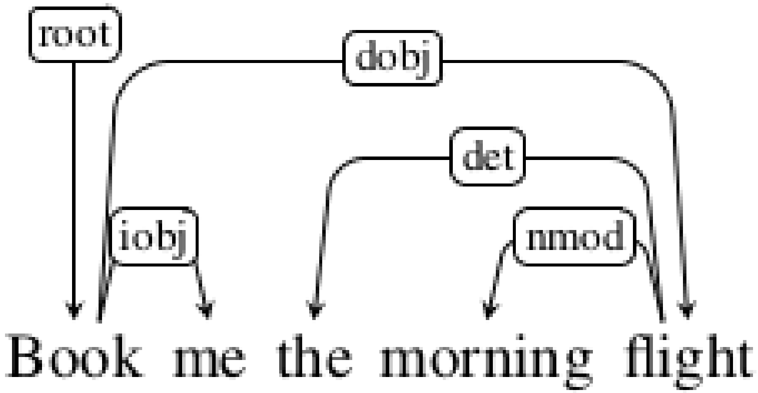
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root,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



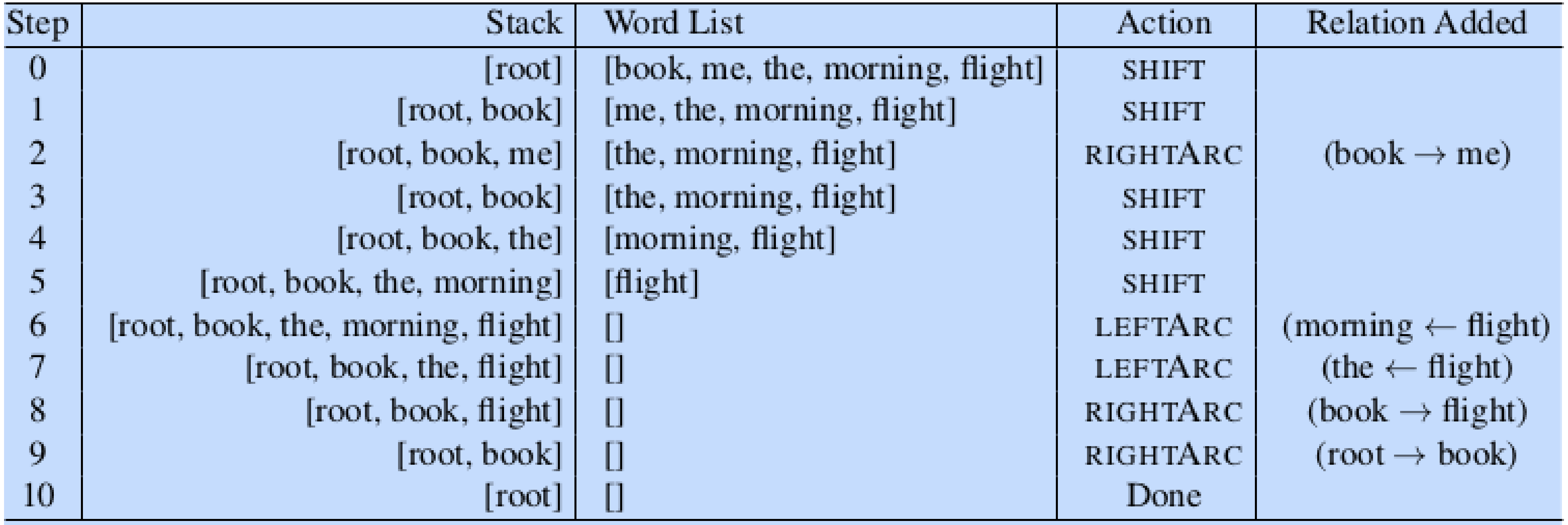
Transition-based Dependency Parsing



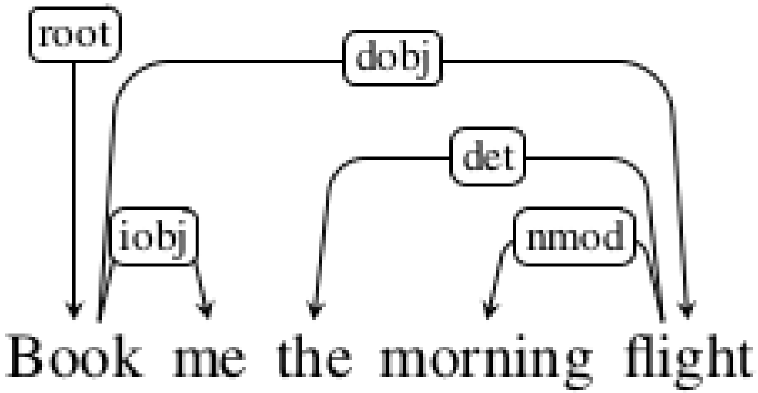
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[moming, flight]	SHIFT	
5	[root,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



Transition-based Dependency Parsing



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	



Parsing Process

- Generating training data
  - Simulating the operation of the parser by running the algorithm and relying on a new training oracle to give us correct transition operators for each successive configuration

- Training a classifier on the generated training data
  - Classic feature-based algorithm
  - Neural classifier using embedding features

Feature-based Classifier

- Using classical ML algorithms; e.g., logistic regression, SVM

- Featured-based classifiers generally use different features, such as
  - Word forms
  - Lemmas
  - Parts of speech
  - The head
  - The dependency relation to the head

- Other features may be relevant for some languages, such as
  - Morphosyntactic features like case marking on subjects or objects

Feature-based Classifier

- The features are extracted from the training configurations, which consist of the stack, the buffer and the current set of relations.

- Most useful are features are extracted from:
  - The top levels of the stack
  - The words near the front of the buffer
  - The dependency relations already associated with any of those elements

- Feature template

<s_1.w,op> <s_2.w,op>

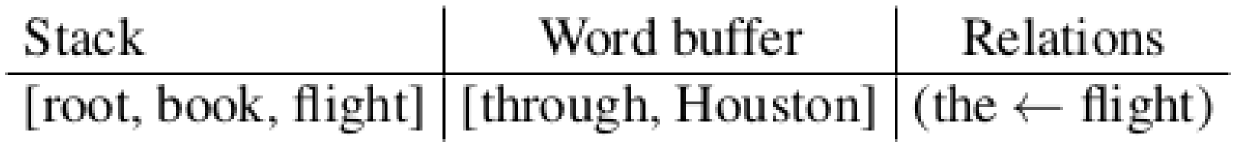
<s_1.t,op> <s_2.t,op>

<b_1.w,op> <b_1.t,op>

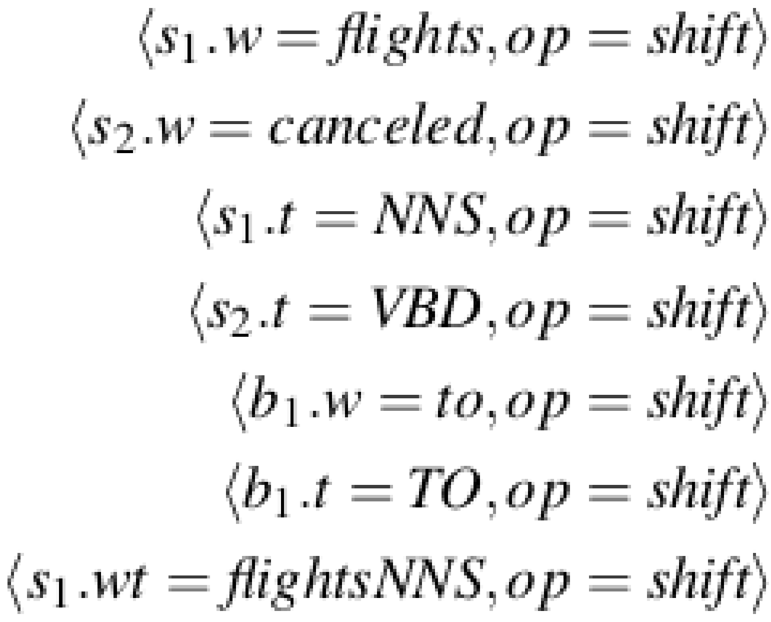
<s_1.wt,op>

Feature-based Classifier

- Example



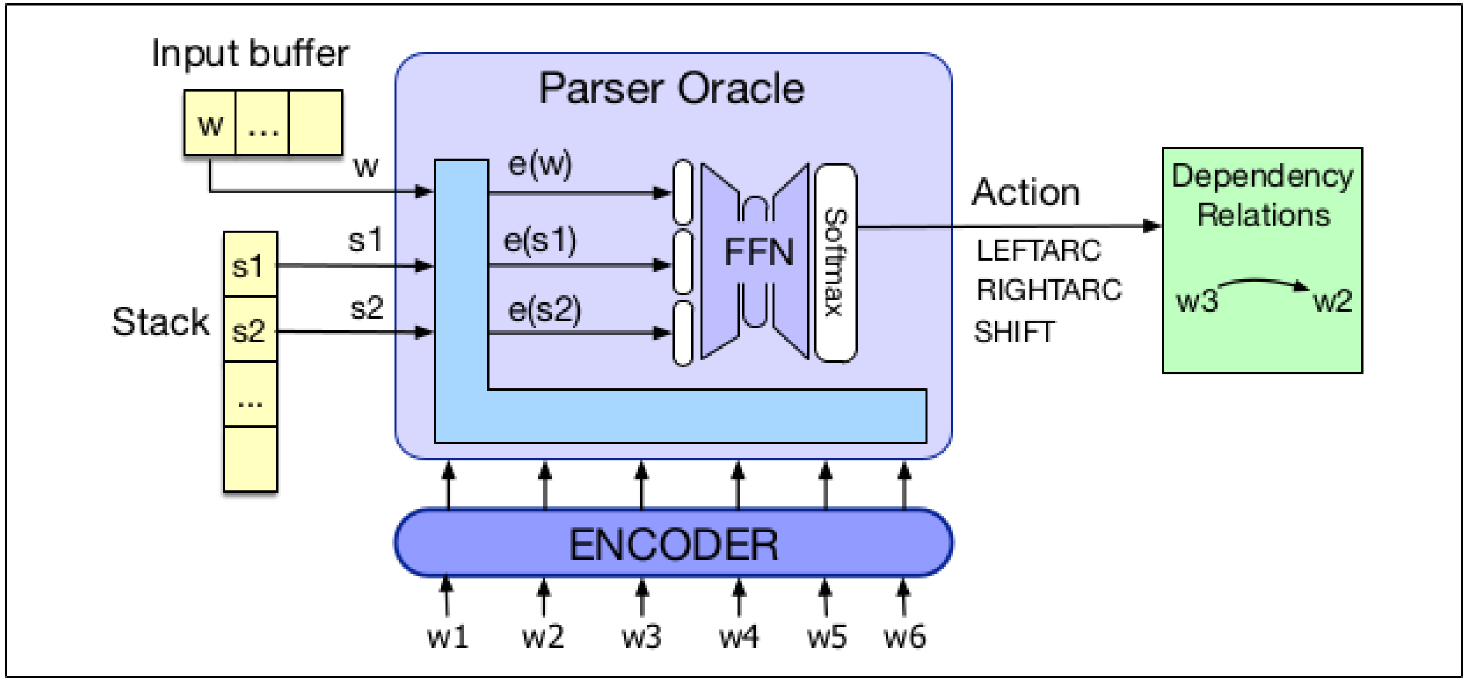
Stack	Word buffer	Relations
[root, book, flight]	[through, Houston]	(the ← flight)



$$\langle s_1.w = flights, op = shift\rangle$$
$$\langle s_2.w = canceled, op = shift\rangle$$
$$\langle s_1.t = NNS, op = shift\rangle$$
$$\langle s_2.t = VBD, op = shift\rangle$$
$$\langle b_1.w = to, op = shift\rangle$$
$$\langle b_1.t = TO, op = shift\rangle$$
$$\langle s_1.wt = flightsNNS, op = shift\rangle$$

Neural Classifier

- Classification scenario:
  - Passing the sentence through an encoder
  - Taking the presentation of the top 2 words on the stack and the first word of the buffer
  - Concatenating the representations and presenting to a feedforward network that predicts the transition to take

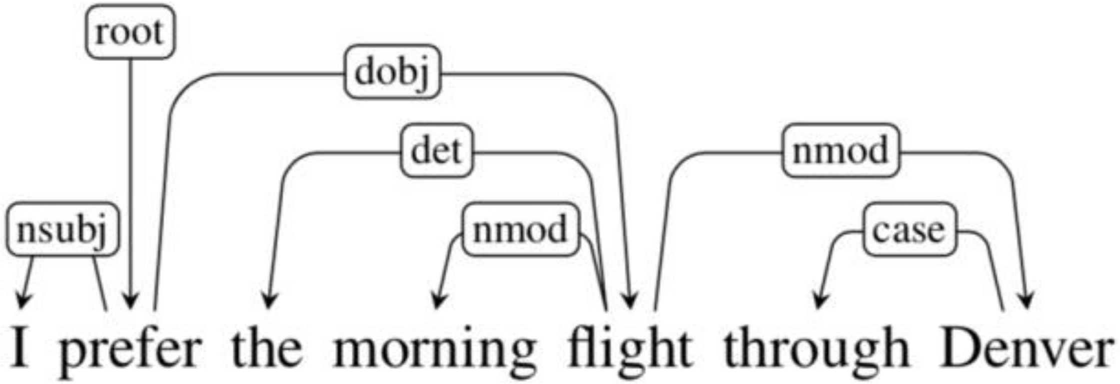


Outline

- Constituency vs. Dependency
- Dependency Relations and Formalisms
- **Dependency Parsing**
  - Transition-based Dependency Parsing
  - **Sequence Modeling**
- Evaluation

Sequence Modeling for Dependency Parsing

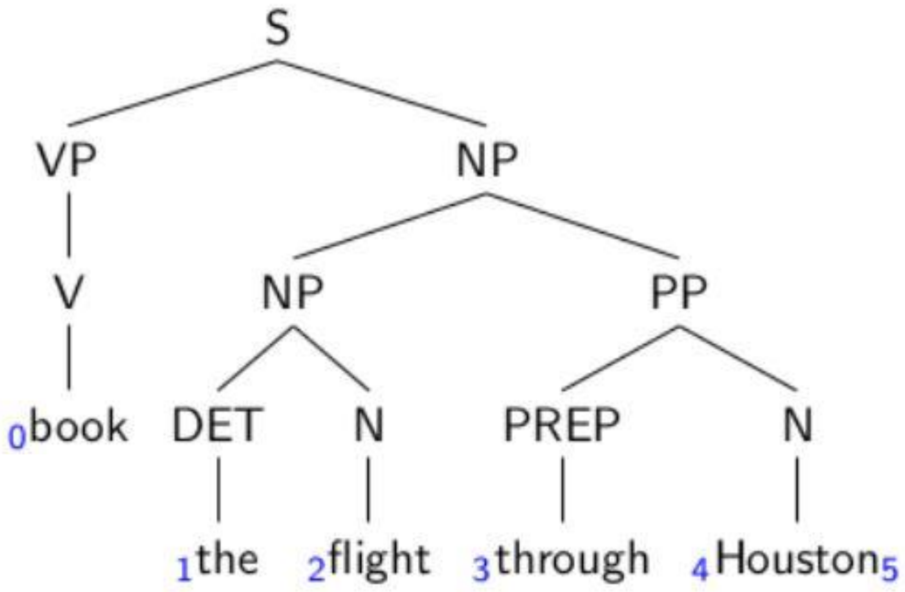
- Bi-directional LSTM and Label learning
- Predicting a label for each word
  - The head word of the target word
    - the: flight
    - I: prefer
    - flight: prefer
  - The path to the head word
    - the → 2 R
    - I → 1 R
    - flight → 3 L



Outline

- Constituency vs. Dependency
- Dependency Relations and Formalisms
- Dependency Parsing
- **Evaluation**

Dependencies in Parse Tree

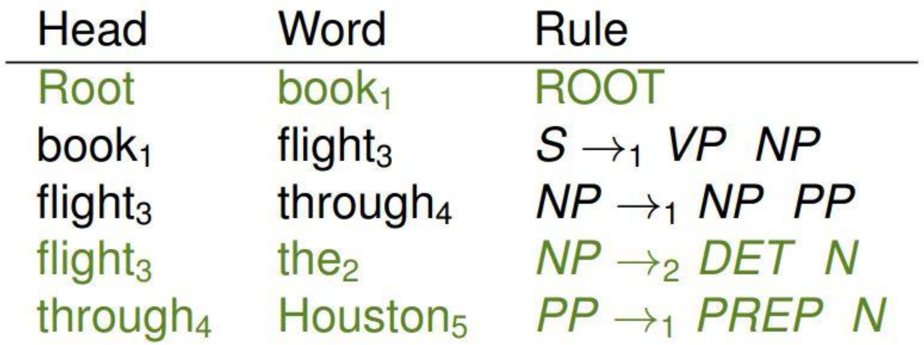


Head	Word	Rule
Root	book$_1$	ROOT
book$_1$	flight$_3$	$S \rightarrow_1 VP\ \ NP$
flight$_3$	through$_4$	$NP \rightarrow_1 NP\ \ PP$
flight$_3$	the$_2$	$NP \rightarrow_2 DET\ \ N$
through$_4$	Houston$_5$	$PP \rightarrow_1 PREP\ \ N$

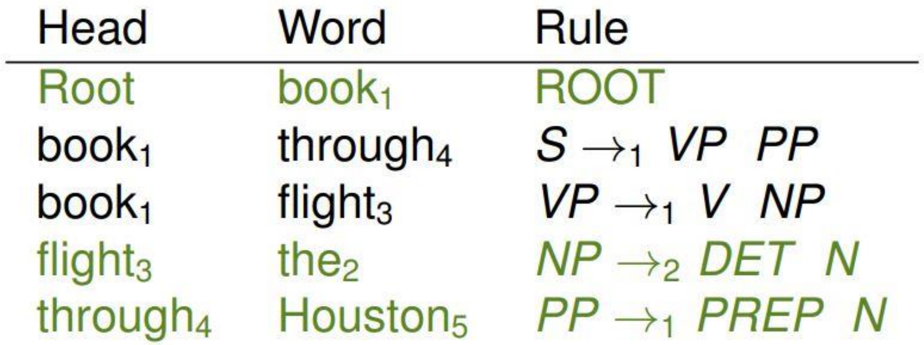
Dependency Accuracy

- The number of dependencies in both gold and guess parse trees are equal to the number of words
- Dependency accuracy:
  The number of dependencies matches in both trees

Dependency Accuracy



Head	Word	Rule
Root	book$_1$	ROOT
book$_1$	flight$_3$	$S \rightarrow_1 VP\ \ NP$
flight$_3$	through$_4$	$NP \rightarrow_1 NP\ \ PP$
flight$_3$	the$_2$	$NP \rightarrow_2 DET\ \ N$
through$_4$	Houston$_5$	$PP \rightarrow_1 PREP\ \ N$



Head	Word	Rule
Root	book$_1$	ROOT
book$_1$	through$_4$	$S \rightarrow_1 VP\ \ PP$
book$_1$	flight$_3$	$VP \rightarrow_1 V\ \ NP$
flight$_3$	the$_2$	$NP \rightarrow_2 DET\ \ N$
through$_4$	Houston$_5$	$PP \rightarrow_1 PREP\ \ N$

$$\text{Dependency Accuracy} = \frac{3}{5} = 0.6$$

Results

- Comparing LPCFG with Dependency model
- Available results on Penn treebank (Wall Street Journal)
  - LPCFG (Collins 1997): 91.4% Dependency Accuracy
  - Dependency parsing (McDonald 2005): 90.7%
    - using dynamic programming
- Efficiency
  - Dependency parsing is much more faster than constituency parsing

Behind Dependency Accuracy

- Precision and recall can still be used, but for a particular dependency type (e.g., NP $\rightarrow_1$ NP PP)

Behind Dependency Accuracy

- Precision and recall can still be used, but for a particular dependency type (e.g., NP $\rightarrow_1$ NP PP)

⇒ Can be used for error analysis

- subject-verb: above 95% recall and precision
- object-verb: above 92% recall and precision
- PP attachments ≈ 82% recall and precision
- Coordination ≈ 61% recall and precision

Further Reading

- Speech and Language Processing (3rd ed. draft)
  - Chapters 18