Dependency Parsing

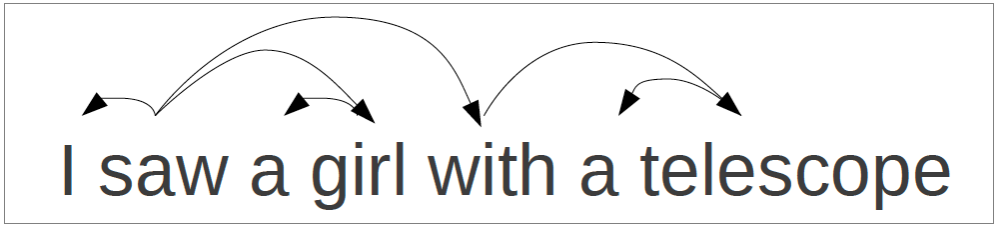
Phạm Quang Nhật Minh
Aimesoft JSC
minhpham0902@gmail.com

Lecture Outline

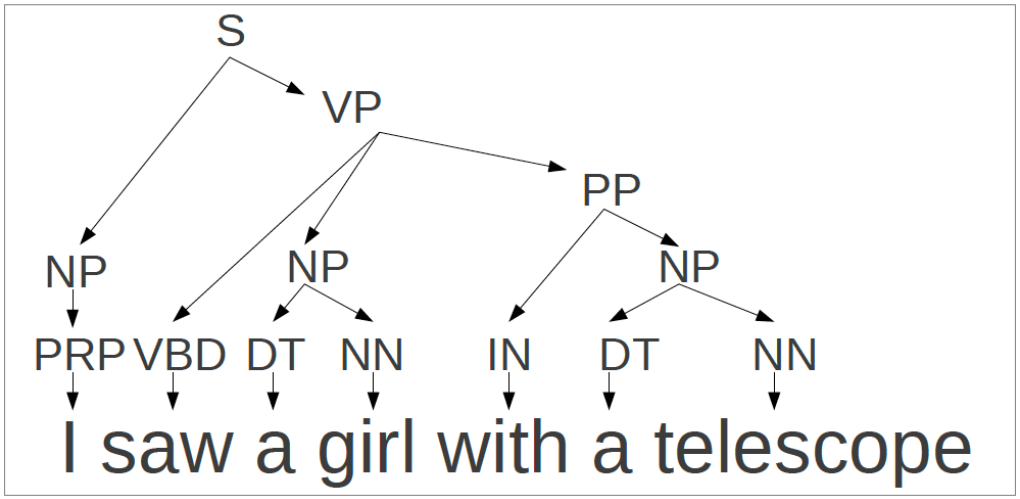
- Introduction and Formalisms
- Dependency Treebanks
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Evaluation
- Summary

Two Types of Parsing

- **Dependency**: focuses on relations between words



- **Phrase structure**: focuses on identifying phrases and their recursive structure



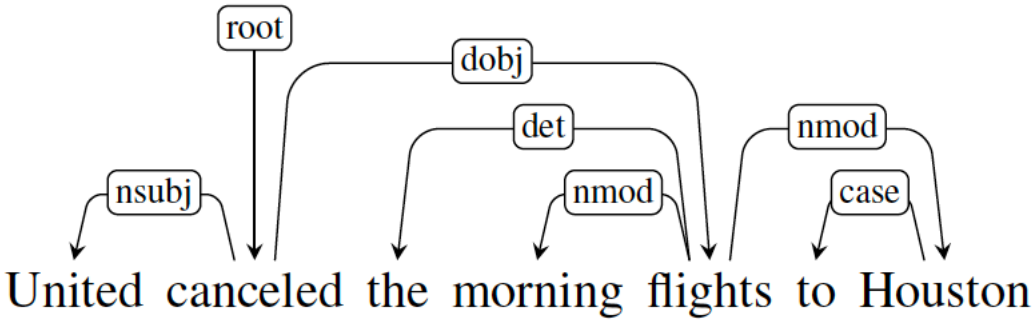
Advantages of Dependency Parsing

- Dependency grammar is more appropriate to languages that have a relatively free word order
  - Czech, Japanese
- Dependency relations are close to semantic relations between predicates/arguments
  - Useful for other NLP tasks: coreference resolution, question answering, information extraction

Dependency Relations

- Consists of head and dependent (head and modifier)
  - the head word is a central of a larger constituent
    - E.g., the primary noun in a noun phrase, or verb in a verb phrase

- Dependency relations can be categorized into grammatical functions
  - E.g., subject, direct object and indirect object



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

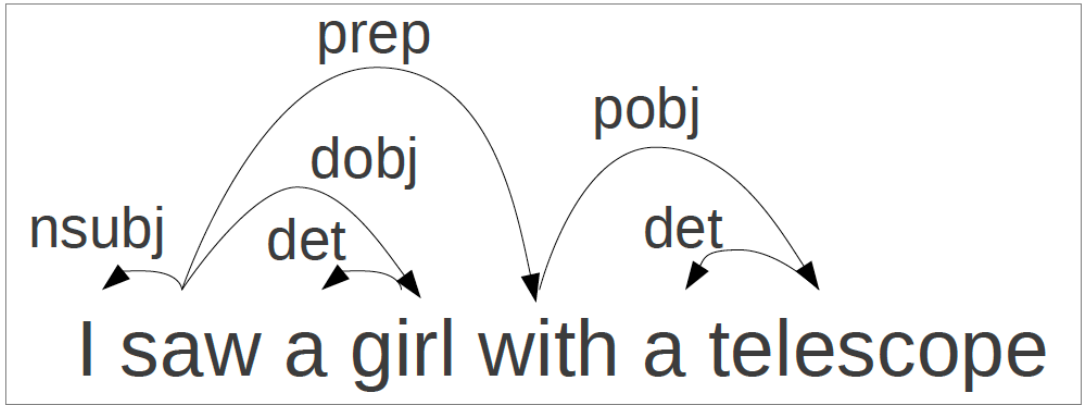
Selected dependency relations from the Universal Dependency set.

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*.

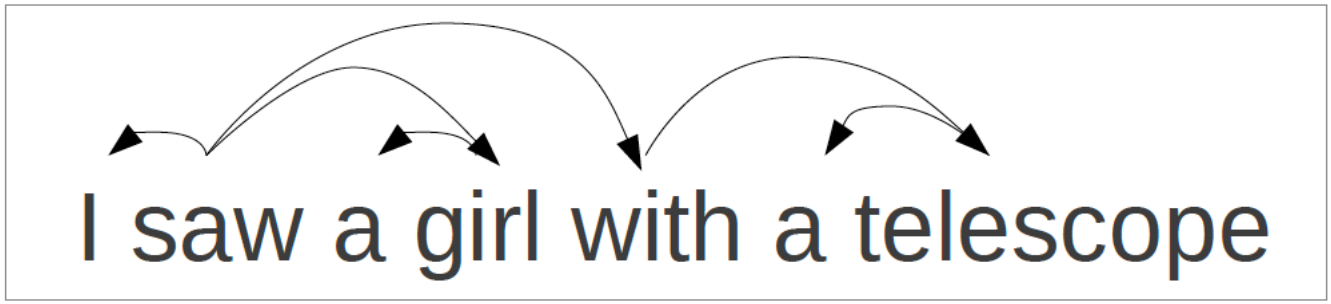
Examples of core Universal Dependency relations.

Dependencies

- Typed: Label indicating relationship between words



- Untyped: Only which words depend



Dependency Formalisms

- Represent dependency structure as a directed graph $G = (V, A)$
  - V: set of vertices
  - A: set of ordered pairs

- A dependency tree is a directed graph
  - There is a single designated root node that has no incoming arcs.
  - Except for the root node, each vertex has exactly one incoming arc.
  - There is a unique path from the root node to each vertex in *V*.

Dependency Parsing Methods

- Shift-reduce
  - Predict from left-to-right
  - Fast (linear), but slightly less accurate?
  - MaltParser
- Spanning tree
  - Calculate full tree at once
  - Slightly more accurate, slower
  - MSTParser, Eda (Japanese)
- Cascaded chunking
  - Chunk words into phrases, find heads, delete nonheads, repeat
  - CaboCha (Japanese)

Shift-Reduce

- **Configuration**
  - Queue: of unprocessed words in input
  - Stack: of partially processed words
  - A set of relations
- At each point choose
  - SHIFT: move one word from queue to stack
  - REDUCE LEFT: top word on stack is head of second word, remove the lower word
  - REDUCE RIGHT: second word on stack is head of top word, remove the top word

Shift-Reduce Example

- Initial configuration

Step	Stack	Queue	Action	Relation Added
0	[root]	[book, me, the, morning, flight]		

- We rely on a trained **oracle** to choose the action

※ oracle: An oracle is a definition for an entity who knows the correct answer to all questions.

- https://bit.ly/3I6kLni

Step	Stack	Queue	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]		

Step	Stack	Queue	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]	reduce right	book → me
3	[root, book]	[the, morning, flight]		

Step	Stack	Queue	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]	reduce right	(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]	[]		

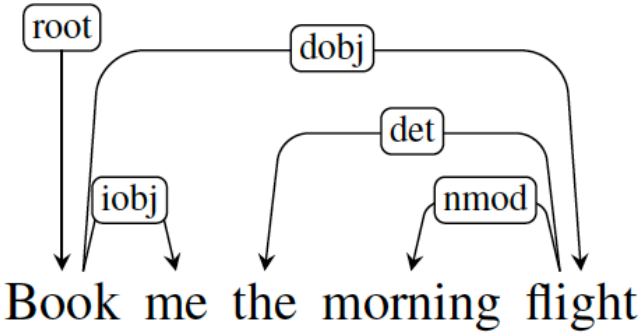
Step	Stack	Queue	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	shift	
1	[root, book]	[me, the, morning, flight]	shift	
2	[root, book, me]	[the, morning, flight]	reduce right	(book → me)
3	[root, book]	[the, morning, flight]	shift	
4	[root, book, the]	[morning, flight]	shift	
5	[root, book, the, morning]	[flight]	shift	
6	[root, book, the, morning, flight]	[]	reduce left	(morning ← flight)
7	[root, book, the, flight	[]		

Step	Stack	Queue	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]	reduce right	(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]	[]	reduce left	(morning ← flight)
7	[root, book, the, flight]	[]	reduce left	(the ← flight)
8	[root, book, flight]	[]		

Step	Stack	Queue	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]	reduce right	(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]	[]	reduce left	(morning ← flight)
7	[root, book, the, flight]	[]	reduce left	(the ← flight)
8	[root, book, flight]	[]	reduce right	(book → flight)
9	[root, book]	[]	reduce right	(root → book)
10	[root]	[]	Done	

Shift-Reduce Example

- Resulting tree



- To produce labels for dependency relations: use actions with labels
  - REDUCE-LEFT[NSUBJ]
  - REDUCE-RIGHT[DOBJ]
  - …

Shift-Reduce Analysis

- Predict from left-to-right
- Fast (linear complexity)
- Greedy (single choice at each point)

Creating an Oracle

- Given a configuration

stack	queue
[root, book, me]	[the, morning, flight]

- Which action do we choose?

shift

reduce left

reduce right

- We need a classifier!

Training Data for Shift-Reduce

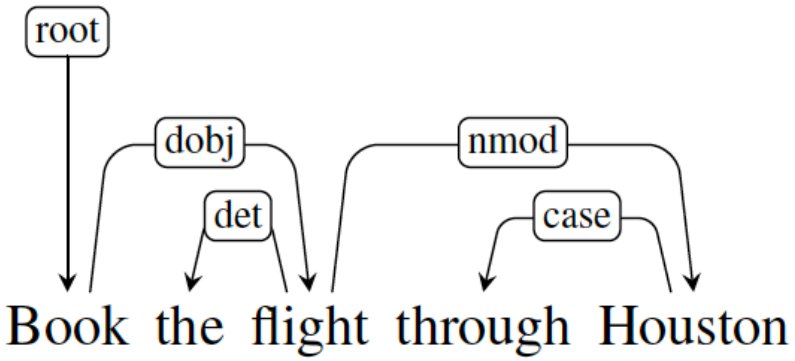
- We need to train the classifier for shift-reduce
- Training data contains configurations (stack, queue) paired with action label

stack	queue	label
[root]	[book, me, the, morning, flight]	shift
[root, book]	[me, the, morning, flight]	shift
[root, book, me]	[the, morning, flight]	reduce right
…		

Generating Training Data for Shift-Reduce

- Dependency Treebank includes sentences with annotaed dependency parses
  - Not directly provides our required training data
- We simulate shift-reduce parsing for sentences given their reference parses
- Given a reference parse and a configuration
  - Choose reduce left if it produces a correct head-dependent relation
  - Otherwise, choose reduce right if
    - (1) it produces a correct head-dependent and
    - (2) all the dependents of the word at the top of the stack have already been assigned
  - Otherwise, choose shift

Step	Stack	Queue	Action
0	[root]	[book, the, flight, through, houston]	shift
1	[root, book]	[the, flight, through, houston]	shift
2	[root, book, the]	[flight, through, houston]	shift
3	[root, book, the, flight]	[through, houston]	reduce left
4	[root, book, flight]	[through, houston]	shift
5	[root, book, flight, through]	[houston]	shift
6	[root, book, flight, through, houston]	[]	reduce left
7	[root, book, flight, houston]	[]	reduce right
8	[root, book, flight]	[]	reduce right
9	[root, book]	[]	reduce right
10	[root]	[]	Done



Features for Shift-Reduce

- Features should generally cover at least the last stack entries and first queue entry

Stack	Queue
[root, canceled, flights]	[to Houston]

- Use Word, POS information

	stack[1]	stack[2]	queue[1]
Word	flights	canceled	to
POS	NNS	VBD	TO

Features for Shift-Reduce

Stack	Queue
[root, canceled, flights]	[to Houston]

	stack[1]	stack[2]	queue[1]
Word	flights	canceled	to
POS	NNS	VBD	TO

- Some features
  - s1.w=flights
  - s2.w=canceled
  - s1.pos=NNS
  - s2.pos=VBD
  - q1.w=to
  - q1.pos=TO
  - s1.wpos=flights-NNS
  - …

Training

- Multinomial logistic regression
- Support Vector Machines
- Neural Networks (Chen and Manning, 2014)

CoNLL File Format

- Standard format for dependencies
- Tab-separated columns, sentences separated by space

```
# sent_id = train-s1
# text = mảnh đất của đạn bom không còn người nghèo.
1       mảnh    mảnh    NOUN    Nc      _       2       compound        _       _
2       đất     đất     NOUN    N       _       7       nsubj   _       _
3       của     của     ADP     E       _       4       case    _       _
4       đạn     đạn     NOUN    N       _       2       nmod    _       _
5       bom     bom     NOUN    N       _       4       compound        _       _
6       không   không   X       R       Polarity=Neg    7       advmod  _       _
7       còn     còn     VERB    V       _       0       root    _       _
8       người   người   NOUN    N       _       7       obj     _       _
9       nghèo   nghèo   ADJ     A       _       8       amod    _       SpaceAfter=No
10      .       .       PUNCT   .       _       7       punct   _       _
```

Reference: https://github.com/UniversalDependencies/UD_Vietnamese-VTB/blob/master/vi_vtb-ud-train.conllu

Graph-Based Dependency Parsing

- Search for the tree that maximizes some score

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}}\, score(t, S)$$

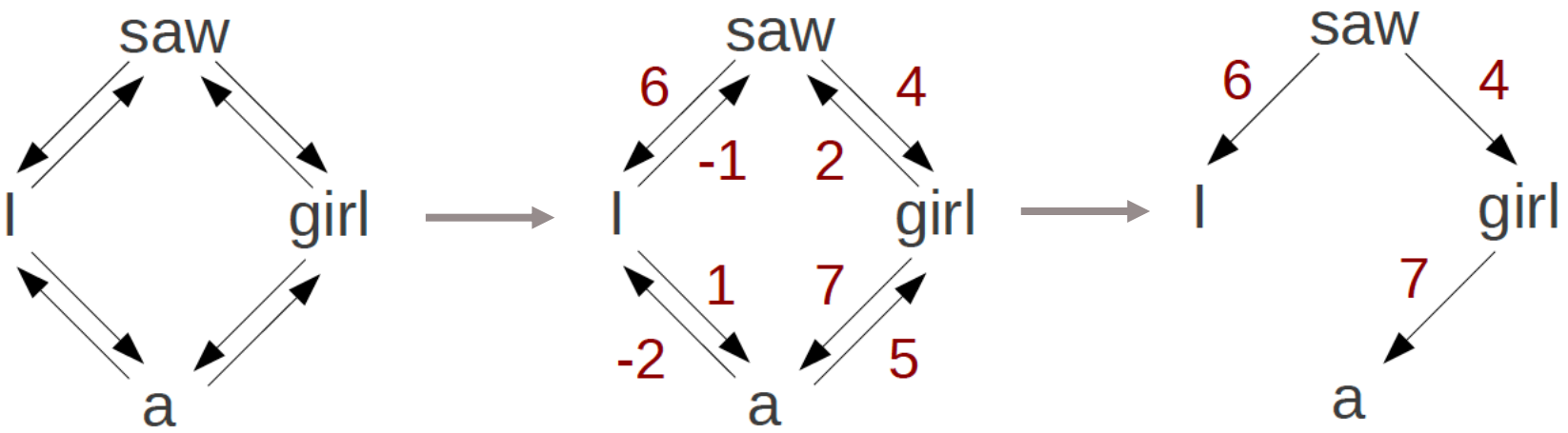
- **Edge-factored:** score for a tree is based on scores of its edges

$$score(t, S) = \sum_{t \in T} score(e)$$

- We assign scores for edges with machine learning

Parsing by Maximum Spanning Tree (MST)

- A spanning tree of a graph G
  - A subgraph of G
  - includes all the vertices of *G*.



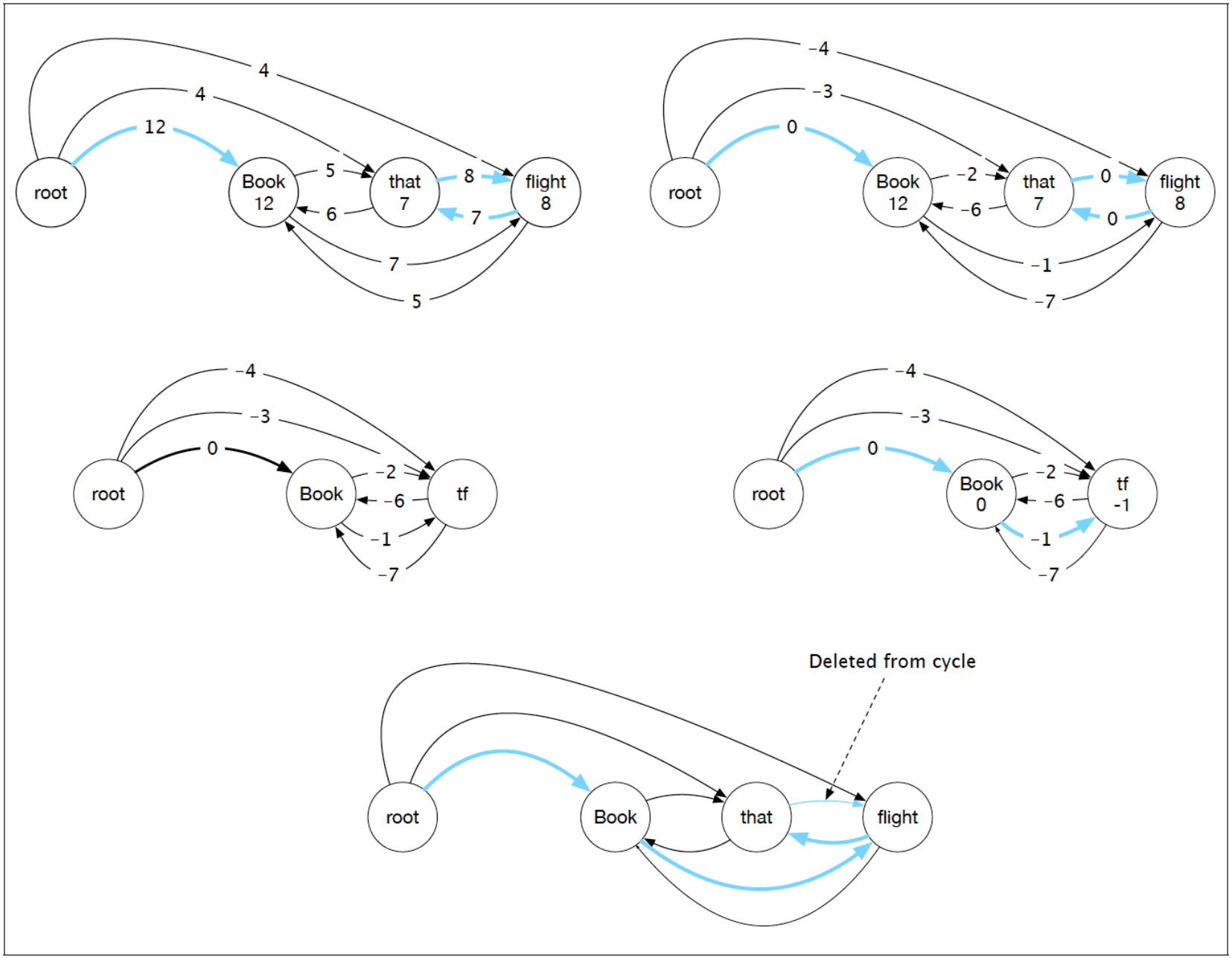


Figure 14.14 Chu-Liu-Edmonds graph-based example for *Book that flight*

Features and Training

- We need a model to compute edge score
- Use weighted sum of features

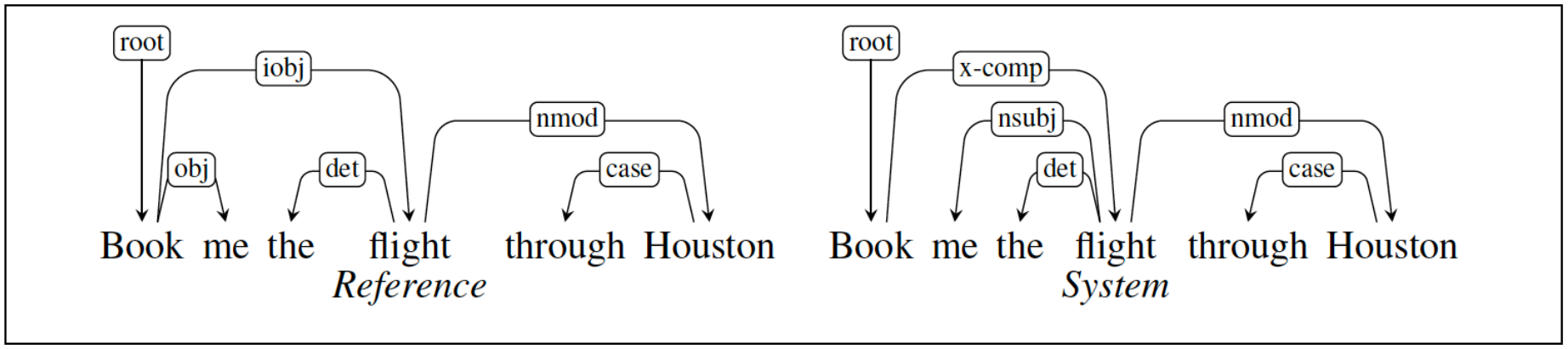
$$score(S, e) = w \cdot f$$

- Features are similar to features used to train the shift-reduce classifier
- State-of-the-art algorithms in multilingual parsing are based on recurrent neural networks (RNNs) (Zeman et al. 2017 , Dozat et al. 2017).

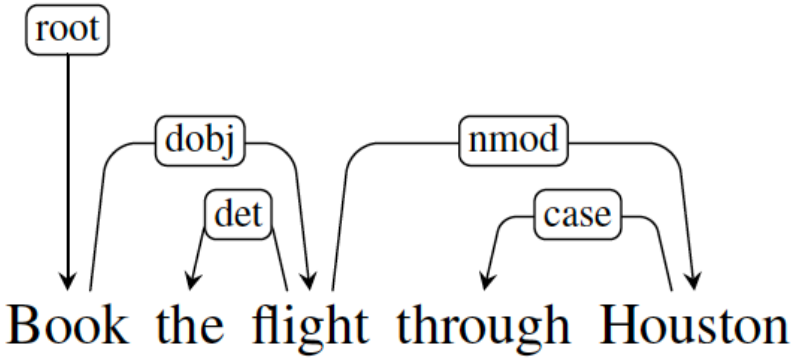
Evaluation Metrics

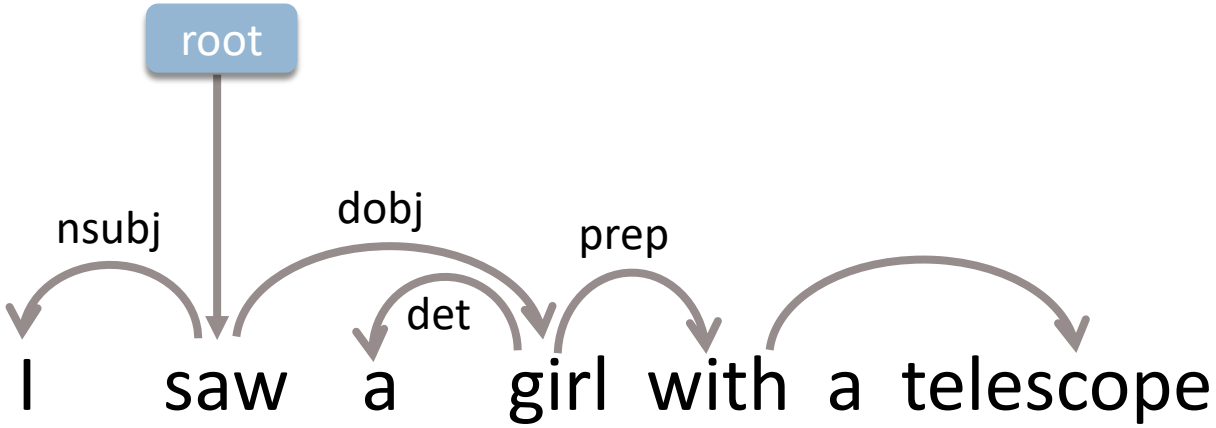
- Percentage of words in an input that are assigned the correct head with the correct relation.
- Two types
  - Labeled attachment score (LAS)
    - Consider labels of dependency relations
  - Unlabeled attachment score (UAS)
    - Ignore labels of dependency relations

Example of LAS and UAS



Reference and system parses for *Book me the flight through Houston*, resulting in an LAS of 2/3 and an UAS of 5/6.


root
dobj
nmod
det
case
Book the flight through Houston


root
nsubj
dobj
prep
det
I saw a girl with a telescope