Tutorial

Dependency Parsing

Teaching Assistant
Yash Bhartia
2019A7PS0151G

Dependency Parsing

- Formalizing dependency trees
- Transition-based dependency parsing
  - Shift-reduce parsing
  - Transition system
  - Oracle
  - Learning/predicting parsing actions
  - Graph Based systems

Definition

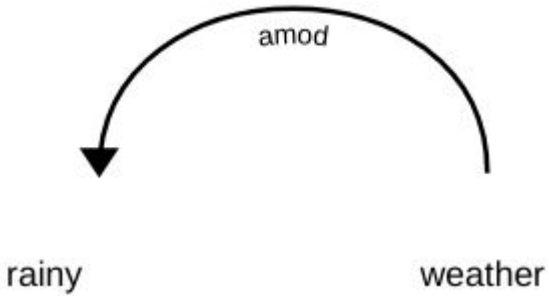
Dependency Parsing is the process to analyze the grammatical structure in a sentence and find out related words as well as the type of the relationship between them.

Each relationship has :

- Has one head and a dependent that modifies the head.
- Is labeled according to the nature of the dependency between the head and the dependent. These labels can be found at Universal Dependency Relations.

Definition - Example

In the phrase ‘rainy weather,’ the word rainy modifies the meaning of the noun weather. Therefore, a dependency exists from the weather -> rainy in which the weather acts as the head and the rainy acts as dependent or child. This dependency is represented by amod tag, which stands for the adjectival modifier.



Universal dependencies

The Universal Dependencies (UD) project provides Universal Dependencies an inventory of dependency relations that are linguistically motivated, computationally useful, and cross-linguistically applicable.

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 Some of the Universal Dependency relations (de Marneffe et al., 2014).

Dependency Tree

Dependency tree is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs.
2. With the exception of the root node, each vertex has exactly one incoming arc.
3. There is a unique path from the root node to each vertex in V.

Taken together, these constraints ensure that each word has a single head, that the dependency structure is connected, and that there is a single root node from which one can follow a unique directed path to each of the words in the sentence

Methods

2 dominant approaches: transition-based parsing and graph-based parsing:

Shift-Reduce parsing

- Predict from left-to-right
- Fast (linear), but slightly less accurate?

Maximal Spanning tree

- Calculate full tree at once
- Slightly more accurate, slower

Transition-based Parser

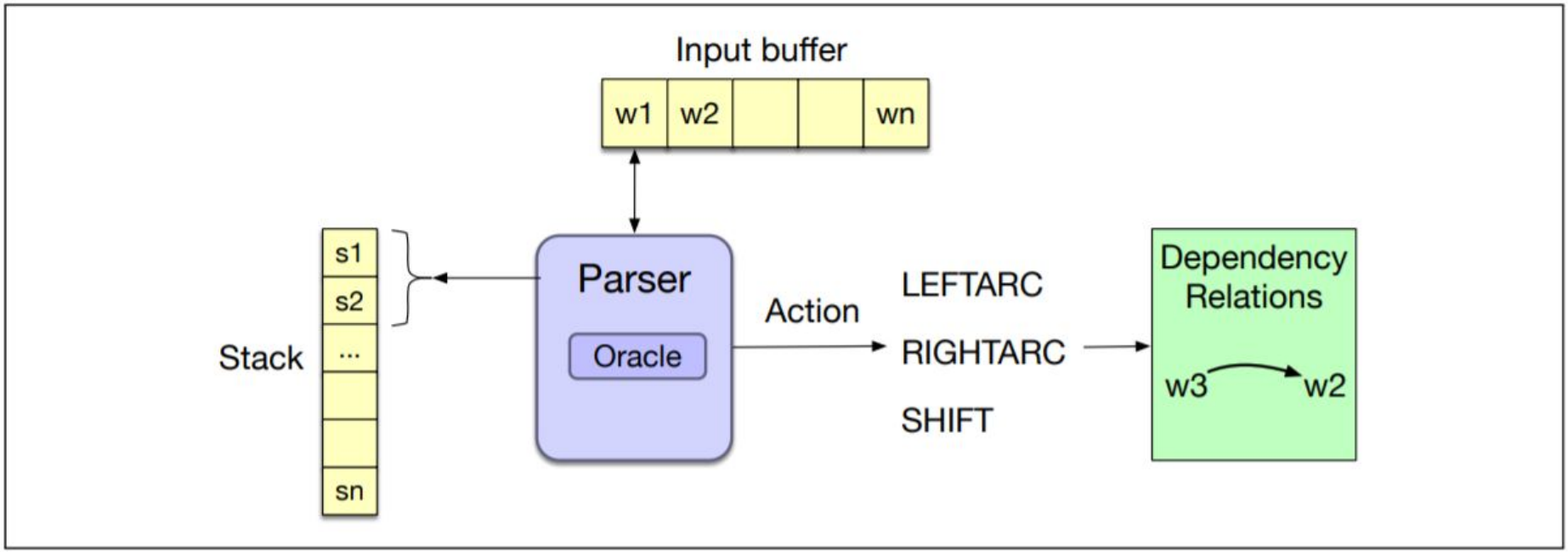


Figure 14.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action by consulting an oracle that examines the current configuration.

Transition-based Parser

- Assume an oracle
- Parsing complexity
- Linear in sentence length!
- Greedy algorithm

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)         ; choose a transition operator to apply
    state ← APPLY(t, state)   ; apply it, creating a new state
  return state
```

Figure 14.6 A generic transition-based dependency parser

Shift Reduce Parsing

Process words one-by-one left-to-right

Two data structures

- Queue: of unprocessed words
- Stack: of partially processed words

At each point choose

- shift: move one word from queue to stack
- reduce left: top word on stack is head of second word
- reduce right: second word on stack is head of top word

Learn how to choose each action with a classifier

Shift Reduce Parsing

Defines 3 transition operators [Covington, 2001; Nivre 2003]

LEFT-ARC:

- create head-dependent rel. between word at top of stack and 2nd word (under top)
- remove 2nd word from stack

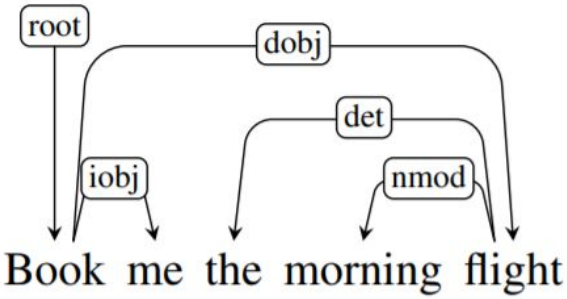
RIGHT-ARC:

- Create head-dependent rel. between word on 2nd word on stack and word on top
- Remove word at top of stack

SHIFT

- Remove word at head of input buffer
- Push it on the stack

Shift Reduce Parsing - Example



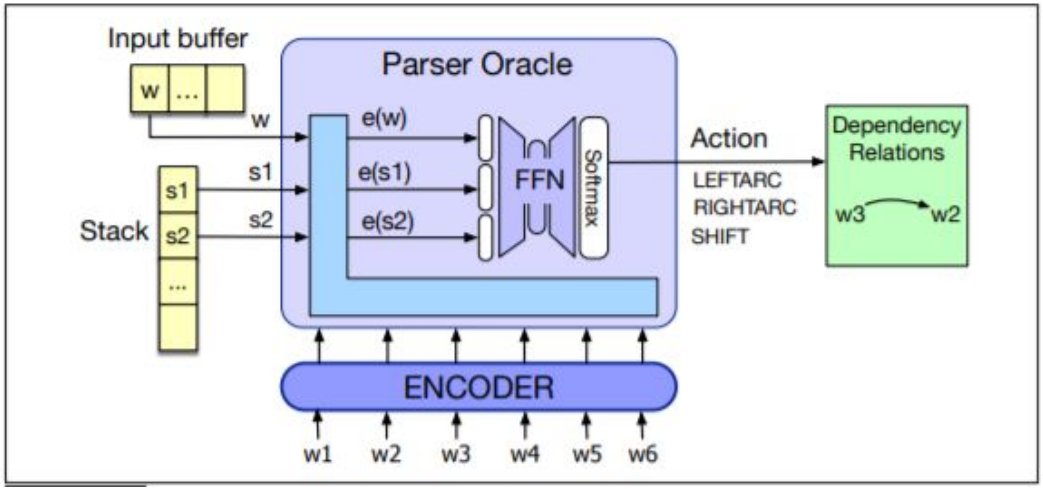
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, 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 14.7 Trace of a transition-based parse.

Oracle

Classic feature based algorithm and **Neural classifier** using embedding features.

- Featured-based classifiers generally use the same features we've seen with part of-speech tagging and partial parsing: Word forms, lemmas, parts of speech, the head, and the dependency relation to the head.
- Neural:



Graph-Based Dependency Parsing

Each dependency is an edge in a directed graph

- Assign each edge a score (with machine learning)
- Keep the tree with the highest score

Graph-Based Dependency Parsing

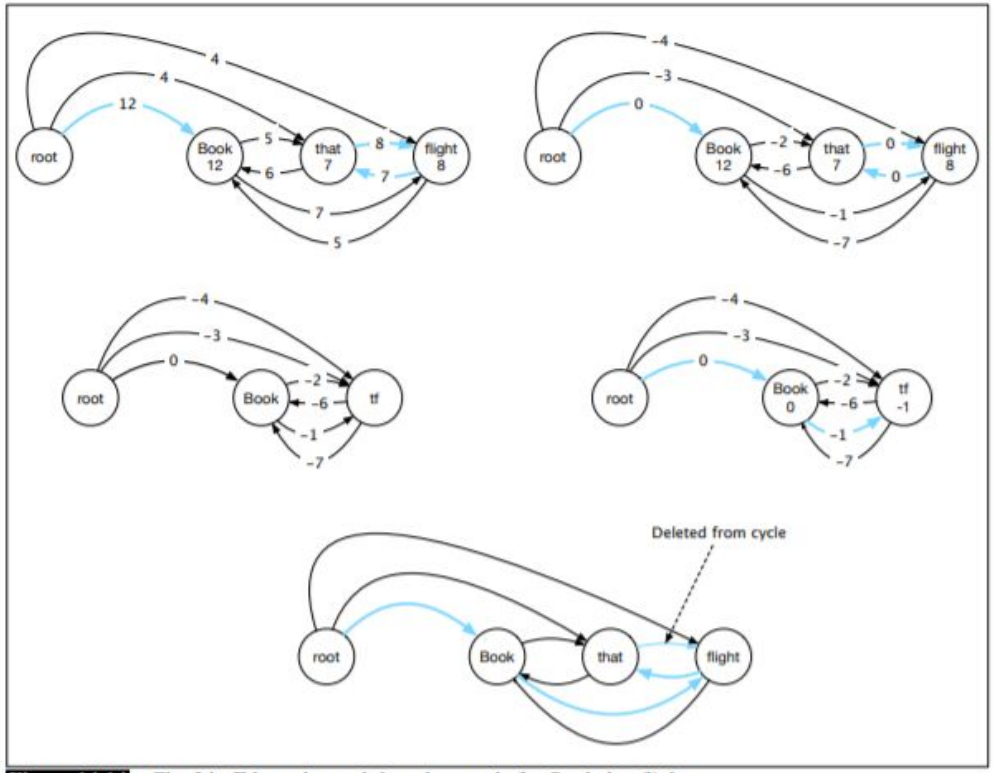


Figure 14.14 Chu-Liu-Edmonds graph-based example for *Book that flight*