CS 6501 Natural Language Processing

Dependency Parsing

Yangfeng Ji

September 24, 2018

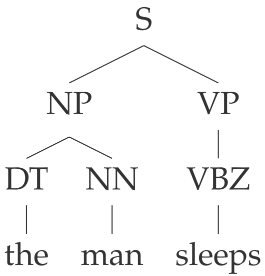
Department of Computer Science
University of Virginia

Overview

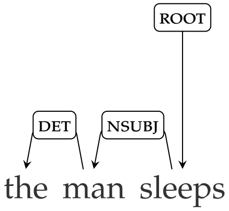
1. Dependency Grammars
2. Transition-Based Dependency Parsing
3. How to Build a Parser?
4. Comments on Dependency Grammars

Dependency Grammars

CFGs

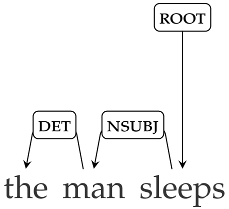


Dependency Grammars



- DET: Determiner
- NSUBJ: Nominal subject

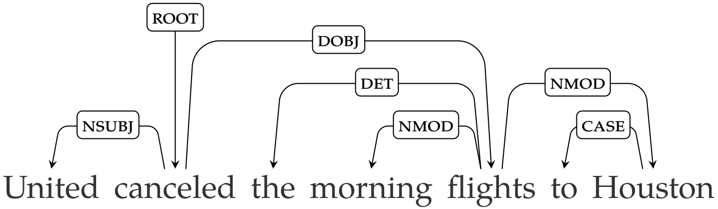
Dependency Trees



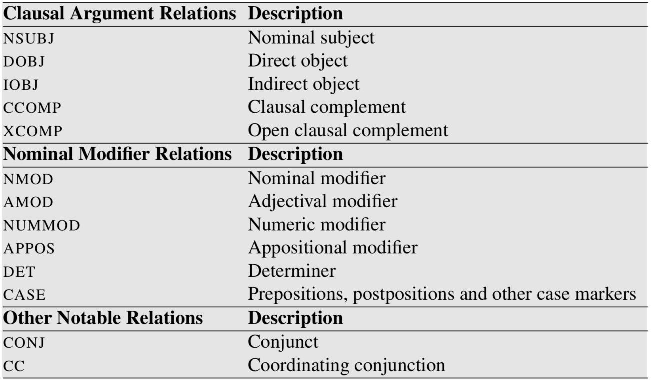
Direct graph $G = (V, A)$: a set of vertices V, and a set of ordered pairs of vertices A,

- root node has no incoming arcs
- each vertex has exactly one incoming arc, except the root node
- a unique path from the root node to each vertex

Components on Dependency Trees

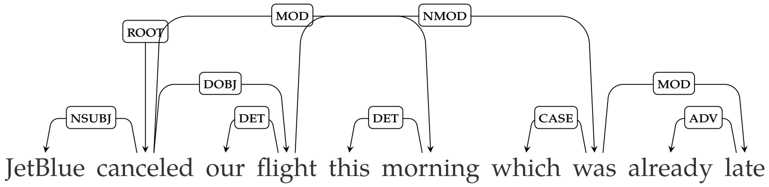


Dependency Relations



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

Projectivity

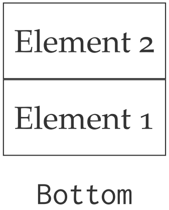


ROOT
MOD
NMOD
DOBJ
MOD
NSUBJ
DET
DET
CASE
ADV
JetBlue canceled our flight this morning which was already late

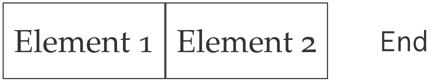
Transition-Based Dependency Parsing

Configuration

- A stack

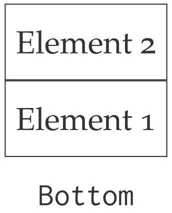


- A queue



- A set of relations representing dependency trees

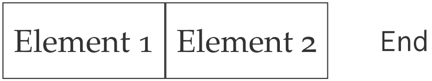
Stack



Basic operations

- Pop one element from the top
- Push one element from the top

Queue

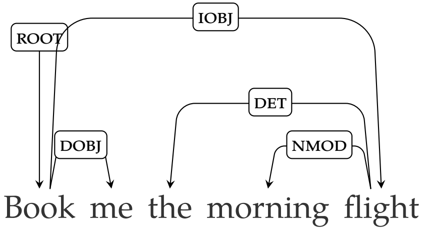


Basic operations

- Enqueue: append one element to the end
- Dequeue: remove one element from the head

Parsing Setup

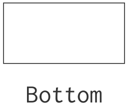
- Input: `Book me the morning flight`
- Output:



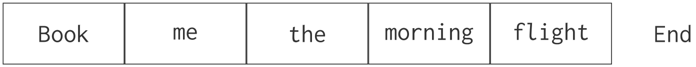
- Containers: a stack, and a queue
- Parser — produce parsing actions to manipulate the stack and queue

Initial State

- Stack: empty



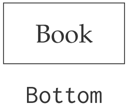
- Queue: contain all the words



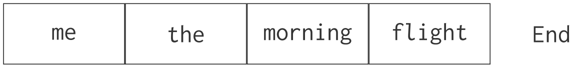
Parsing Actions (I): Shift

Shift one word from the queue to the stack

- Stack:



- Queue:



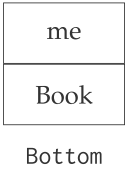
- Tree:

Book me the morning flight

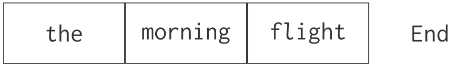
Parsing Actions (I): Shift

Shift one word from the queue to the stack

- Stack:



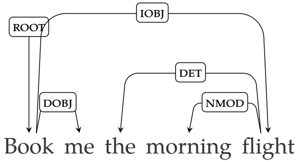
- Queue:



- Tree:

Book me the morning flight

Next Step?



me
Book

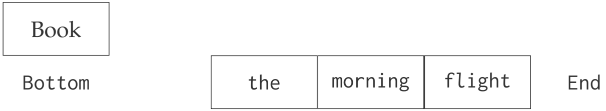
Bottom

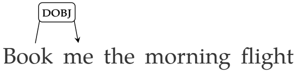
the	morning	flight

End

Parsing Actions: RightArc

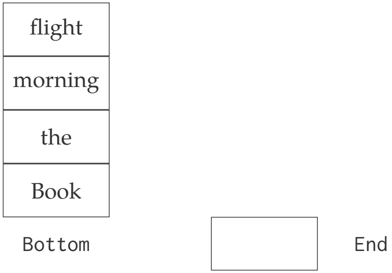
RightArc: assert a head-dependent relation between the second and the top words; remove the top word from the stack

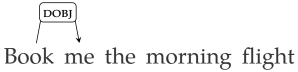




Parsing Actions: LeftArc

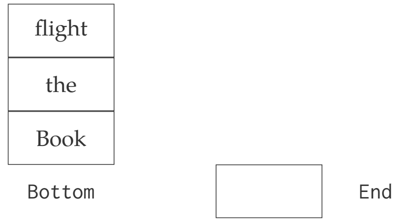
LeftArc: assert a head-dependent relation between the top and the second words; remove the second word from the stack

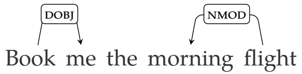




Parsing Actions: LeftArc

LeftArc: assert a head-dependent relation between the top and the second words; remove the second word from the stack





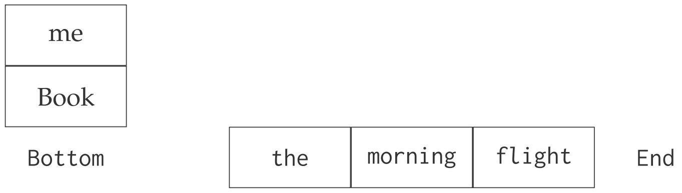
Complexity

- Greedy
- Time complexity $\mathcal{O}(n)$, $2n - 1$ parsing actions to be accurate
- Space complexity $\mathcal{O}(n)$

where n is the length of the sentence

How to Build a Parser?

How to Pick an Action?



- If there is no ground truth

Parsing as Classification

$$\hat{y}_t = \arg\max_{y'_t} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}_t, y'_t) \quad (1)$$

Parsing as Classification

$$\hat{y}_t = \arg\max_{y'_t} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}_t, y'_t) \tag{1}$$

$\boldsymbol{x}_t$?

- Top two elements from the stack
- The head element from the queue

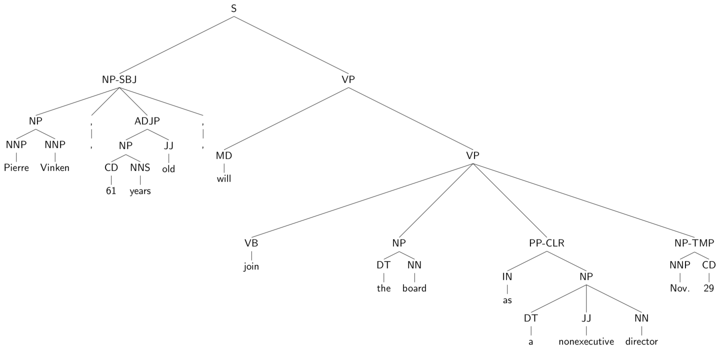
Parsing Actions

How many parsing actions in total?

- Three basic parsing actions
- N dependency relations

Total: $2N + 1$ actions (labels for classification)

Training Corpus: Penn Treebank

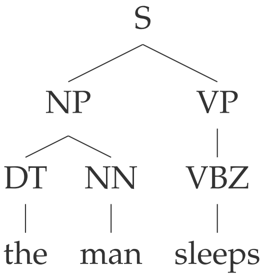


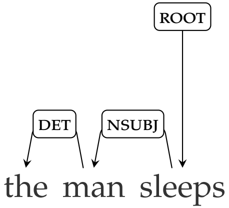
From CFGs to Dependency Trees

The rule of finding the head of a noun phrase:

- If the last word is tagged POS, return last-word.
- Else search from right to left for the first child which is an NN,NNP,NNPS,NX,POS, or JJR.
- Else search from left to right for the first child which is an NP.
- Else search from right to left for the first child which is a$, ADJP, or PRN.
- Else search from right to left for the first child which is a CD.
- Else search from right to left for the first child which is a JJ, JJS, RB or QP.
- Else return the last word

Example





From a Tree to Parsing Actions

How to recover parsing actions from a dependency tree?

Comments on Dependency Grammars

Advantage of Dependency Grammars

Relations with CFGs

Summary

1. Dependency Grammars
2. Transition-Based Dependency Parsing
3. How to Build a Parser?
4. Comments on Dependency Grammars

Reference