CS 6501 Natural Language Processing

Dependency Parsing

Yangfeng Ji

October 17, 2019

Department of Computer Science
University of Virginia

University of Virginia | ENGINEERING

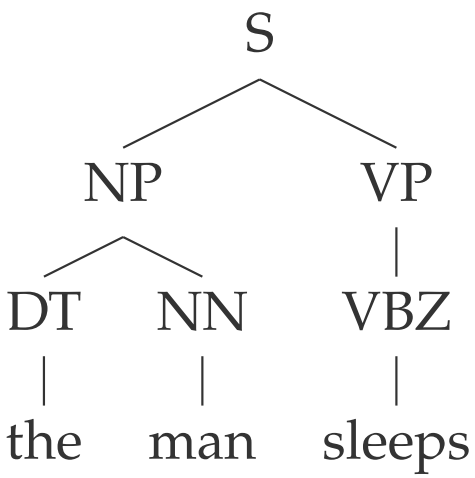
Overview

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

Dependency Grammars

Context-free Grammars

A example of CFG:



Or

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow DT\ NN \\
VP &\rightarrow VBZ \\
DT &\rightarrow \text{the} \\
NN &\rightarrow \text{man} \\
VBZ &\rightarrow \text{sleeps}
\end{aligned}
$$

Dependency Grammars

Given two words w_i and w_j from a sentence:

$$w_i \xrightarrow{\textsc{Relation Type}} w_j \tag{1}$$

Basic components in a dependency relation

- Head: w_i
- Dependent/modifier: w_j
- A relation type associated with the head and the dependent

Examples

Relations	head and dependent
NSUBJ	We booked her the first flight to Miami

Dependency relations:

- NSUBJ: nominal subject

Examples

Relations	head and dependent
NSUBJ	We booked her the first flight to Miami
DOBJ	We booked her the first flight to Miami

Dependency relations:

- NSUBJ: nominal subject
- DOBJ: direct object

Examples

Relations	head and dependent
NSUBJ	We booked her the first flight to Miami
DOBJ	We booked her the first flight to Miami
IOBJ	We booked her the first flight to Miami

Dependency relations:

- NSUBJ: nominal subject
- DOBJ: direct object
- IOBJ: indirect object

Examples

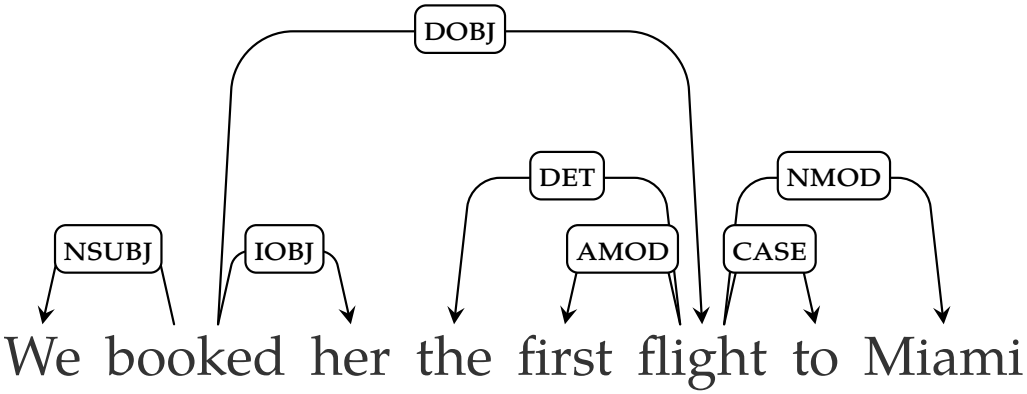
Relations	head and dependent
NSUBJ	We booked her the first flight to Miami
DOBJ	We booked her the first flight to Miami
IOBJ	We booked her the first flight to Miami
AMOD	We booked her the first flight to Miami

Dependency relations:

- NSUBJ: nominal subject
- DOBJ: direct object
- IOBJ: indirect object
- AMOD: adjectival modifier

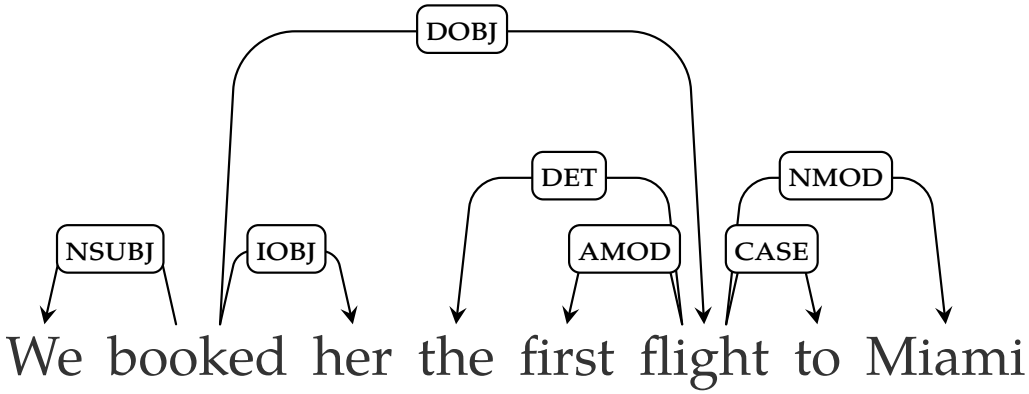
Dependency Trees

All dependency relations in the previous example can be unified into a tree structure, called **dependency tree**



Dependency Trees

All dependency relations in the previous example can be unified into a tree structure, called **dependency tree**



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

- 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

Dependency Relations

Some example relations

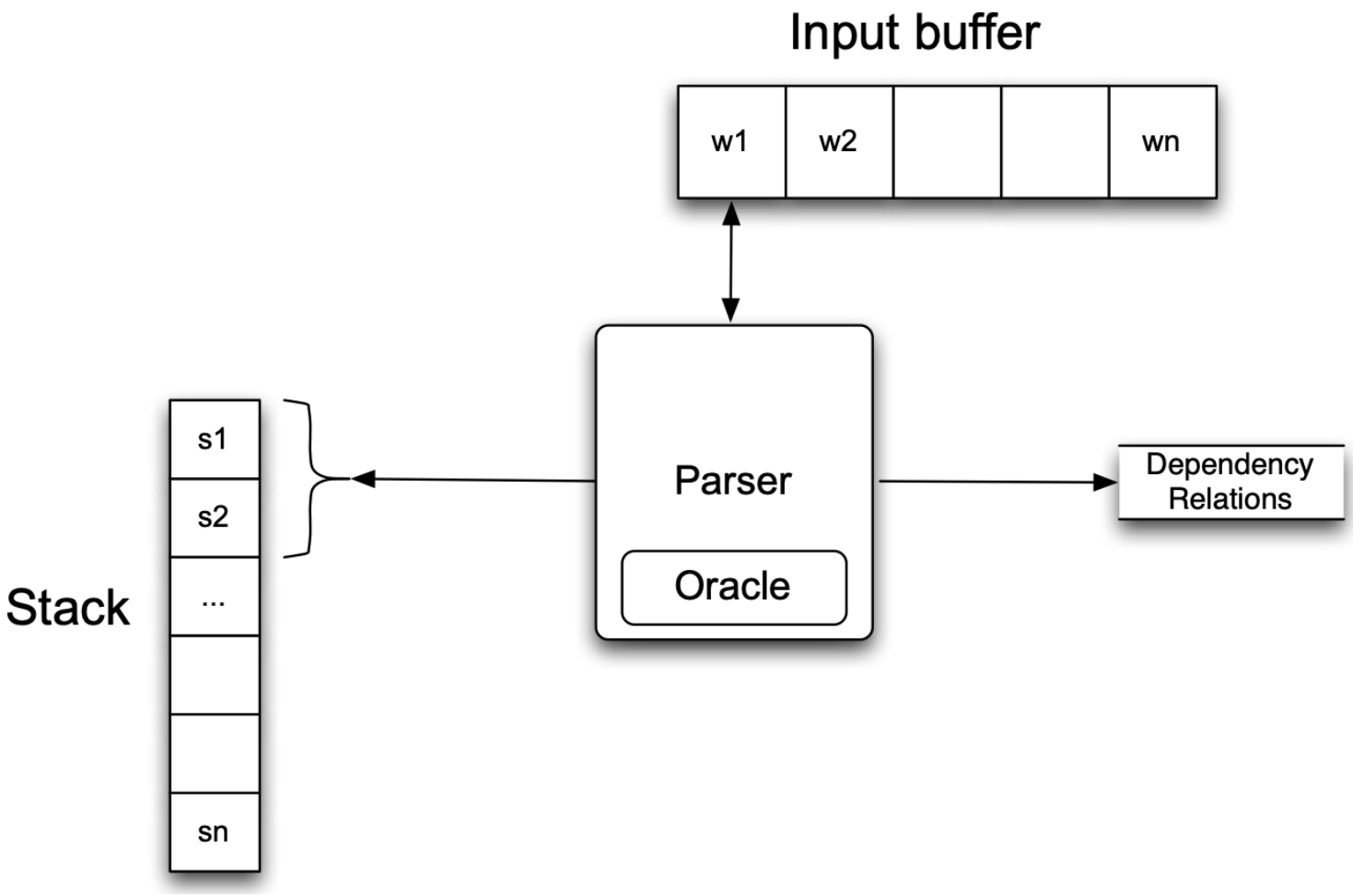
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET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

[Jurafsky and Martin, 2019]

Transition-Based Parsing

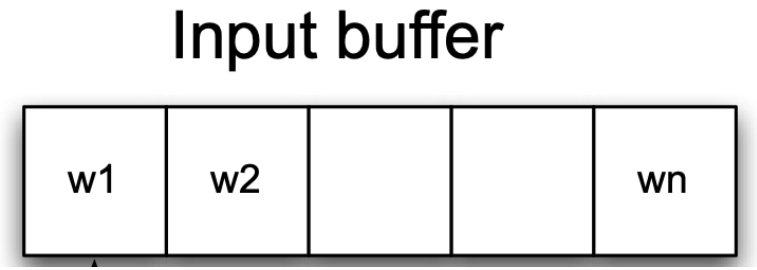
Configuration

Basic framework of transition-based dependency parsing



Input Buffer/Queue

Used to keep all the input tokens

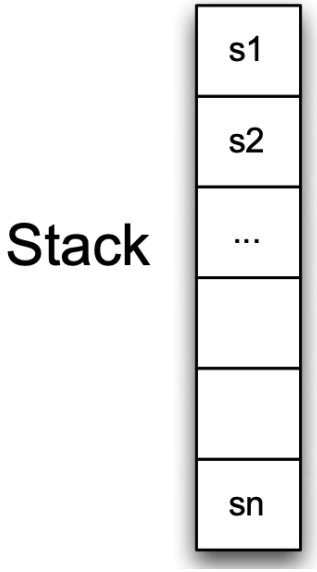


Basic operations

- Enqueue: append one element to the end (will *not* be used)
- Dequeue: remove one element from the head

Stack

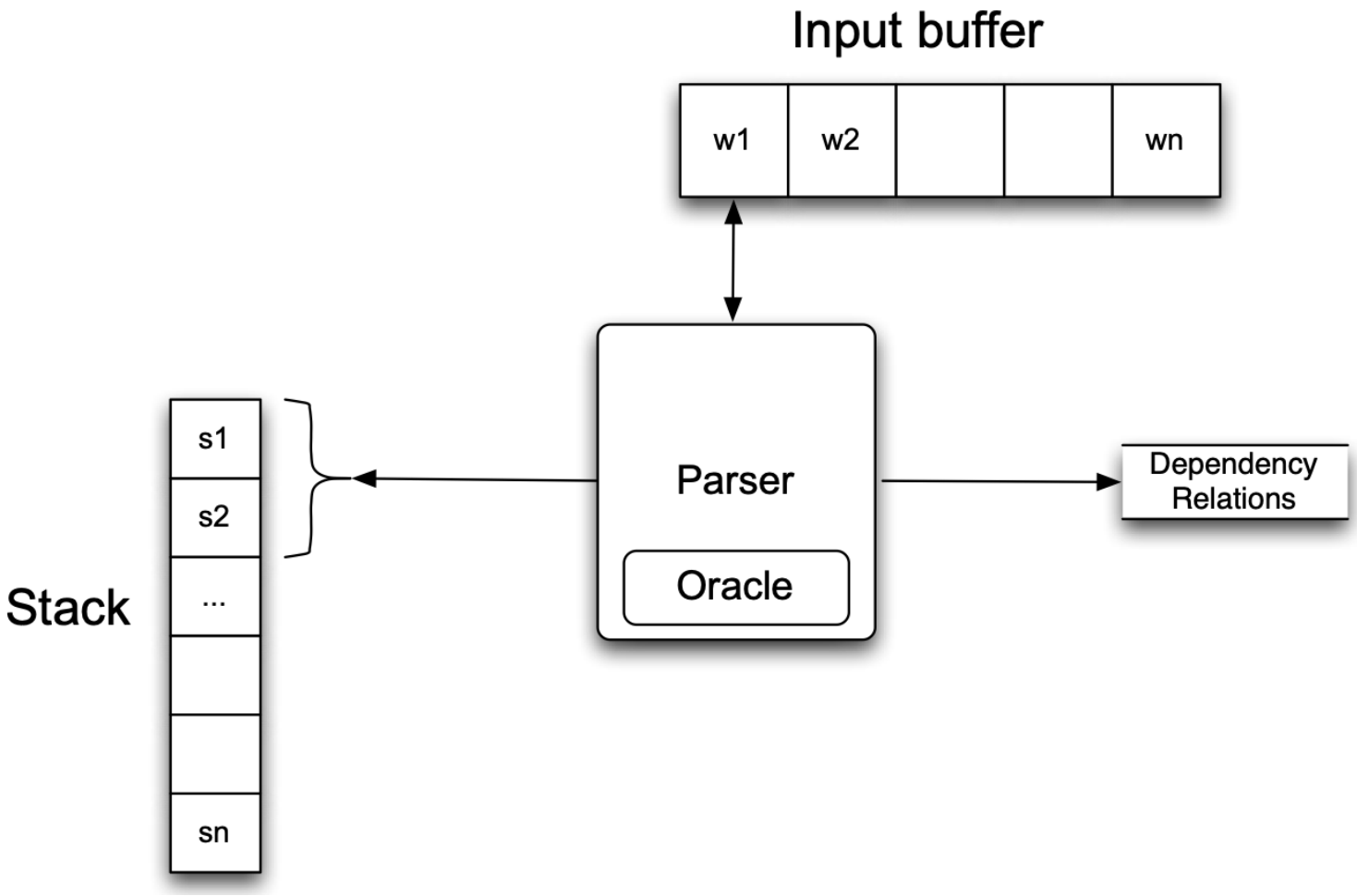
Used to keep all the intermediate results during parsing



Basic operations

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

Oracle



Oracle: given the status of input buffer and stack, generate the next parsing action.

Parsing Action (I)

SHIFT action:

1. Dequeue one token from the input buffer
2. Push it to the top of the stack

Stack	Input buffer	Action
(book)	(me, the, morning, flight)	Shift

Parsing Action (I)

Shift action:

1. Dequeue one token from the input buffer
2. Push it to the top of the stack

Stack	Input buffer	Action
(book)	(me, the, morning, flight)	Shift
(book, me)	(the, morning, flight)	

Parsing Action (II)

RightArc action:

1. Pop the top two words from the stack
2. Create a right-facing arc between the these two words
3. Push the top second word back to the stack

Stack	Input buffer	Action
(book, me)	(the, morning, flight)	RightArc

Parsing Action (II)

RightArc action:

1. Pop the top two words from the stack
2. Create a right-facing arc between the these two words
3. Push the top second word back to the stack

Stack	Input buffer	Action
(book, me)	(the, morning, flight)	RightArc
(book)	(the, morning, flight)	

Parsing Action (III)

LeftArc action:

1. Pop the top two words from the stack
2. Create a left-facing arc between the these two words
3. Push the top first word back to the stack

Stack	Input buffer	Action
(book, me, the, morning, flight)	()	LeftArc

Parsing Action (III)

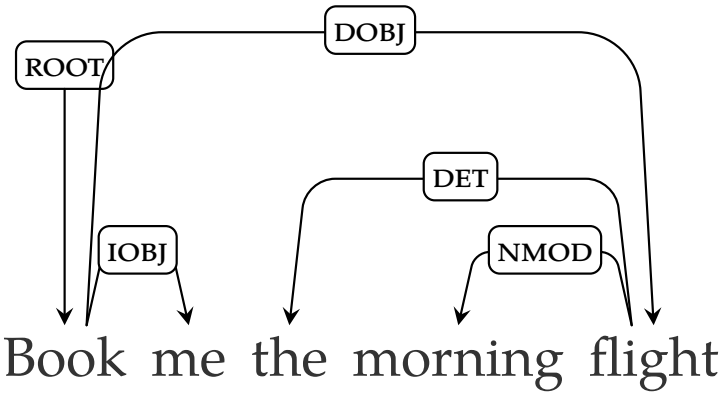
LeftArc action:

1. Pop the top two words from the stack
2. Create a left-facing arc between the these two words
3. Push the top first word back to the stack

Stack	Input buffer	Action
(book, me, the, morning, flight)	()	LeftArc
(book, me, the, flight)	()	

Overview: Parsing setup

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



- Containers: an input buffer and a stack
- Oracle: produce parsing actions to manipulate the input buffer and the stack

Initial State

- Stack: empty
- Input Buffer: contain all the words

Stack	Input buffer	Action
()	(book, me, the, morning, flight)	Shift

End State

- Stack: only contain one word
- Input Buffer: empty

Stack	Input buffer	Action
(book)	()	

A Complete Example

Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT

Book me the morning flight

Dependency relations are ignored from the parsing actions in the table.

A Complete Example

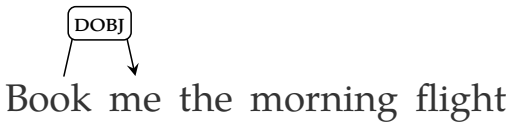
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT

Book me the morning flight

Dependency relations are ignored from the parsing actions in the table.

A Complete Example

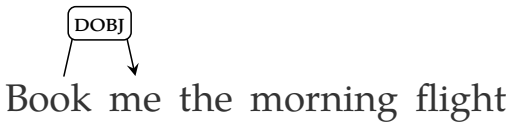
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

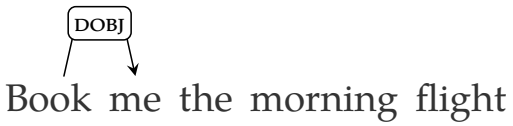
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

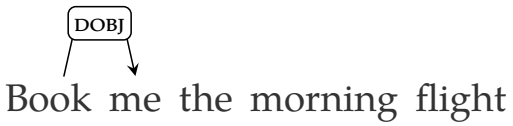
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

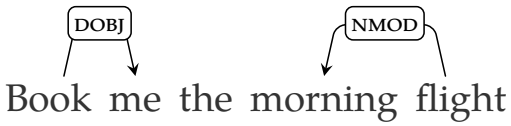
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT
(book, the, morning)	(flight)	SHIFT



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

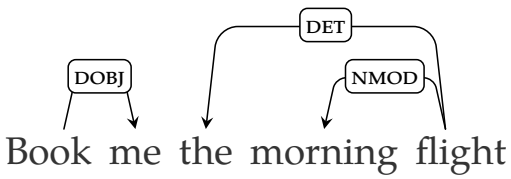
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT
(book, the, morning)	(flight)	SHIFT
(book, the, morning, flight)	()	LEFTARC



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

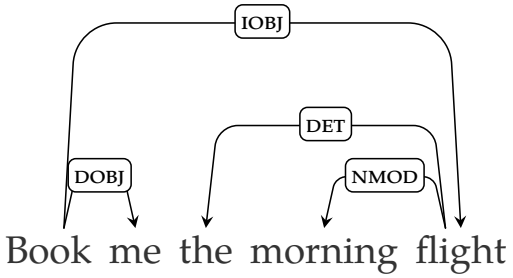
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT
(book, the, morning)	(flight)	SHIFT
(book, the, morning, flight)	()	LEFTARC
(book, the, flight)	()	LEFTARC



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

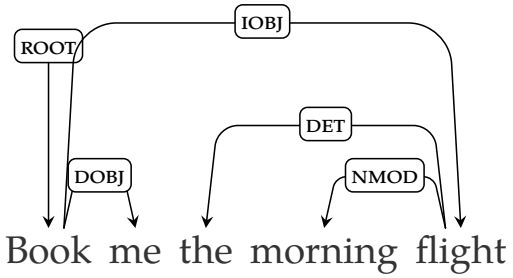
Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT
(book, the, morning)	(flight)	SHIFT
(book, the, morning, flight)	()	LEFTARC
(book, the, flight)	()	LEFTARC
(book, flight)	()	RIGHTARC



Dependency relations are ignored from the parsing actions in the table.

A Complete Example

Stack	Input Buffer	Action
()	(book, me, the, morning, flight)	SHIFT
(book)	(me, the, morning, flight)	SHIFT
(book, me)	(the, morning, flight)	RIGHTARC
(book)	(the, morning, flight)	SHIFT
(book, the)	(morning, flight)	SHIFT
(book, the, morning)	(flight)	SHIFT
(book, the, morning, flight)	()	LEFTARC
(book, the, flight)	()	LEFTARC
(book, flight)	()	RIGHTARC
(book)	()	



Dependency relations are ignored from the parsing actions in the table.

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?

How to select a parsing action, if there is no oracle:

Stack	Input buffer	Action
(book, me)	(the, morning, flight)	?

How to Pick an Action?

How to select a parsing action, if there is no oracle:

	Stack	Input buffer	Action
	(book, me)	(the, morning, flight)	?
Option 1	(book, me, the)	(morning, flight)	SHIFT
Option 2	(book)	(the, morning, flight)	RIGHTARC
Option 3	(me)	(the, morning, flight)	LEFTARC

Parsing as Classification

Formulate the parsing action selection as a classification problem

$$\hat{y}_t = \operatorname*{argmax}_{y_t} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}_t, y_t) \tag{2}$$

where $\boldsymbol{x}_t$ represents the status of the input buffer and the stack, and y_t is a parsing action

Parsing as Classification

Formulate the parsing action selection as a classification problem

$$\hat{y}_t = \underset{y_t}{\operatorname{argmax}}\, \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}_t, y_t) \tag{2}$$

where $\boldsymbol{x}_t$ represents the status of the input buffer and the stack, and y_t is a parsing action

What is the minimal requirement of $\boldsymbol{x}_t$?

- Top two words from the stack
- The first words from the input buffer

Stack	Input buffer	Action
(book, me)	(the, morning, flight)	?

Parsing Actions

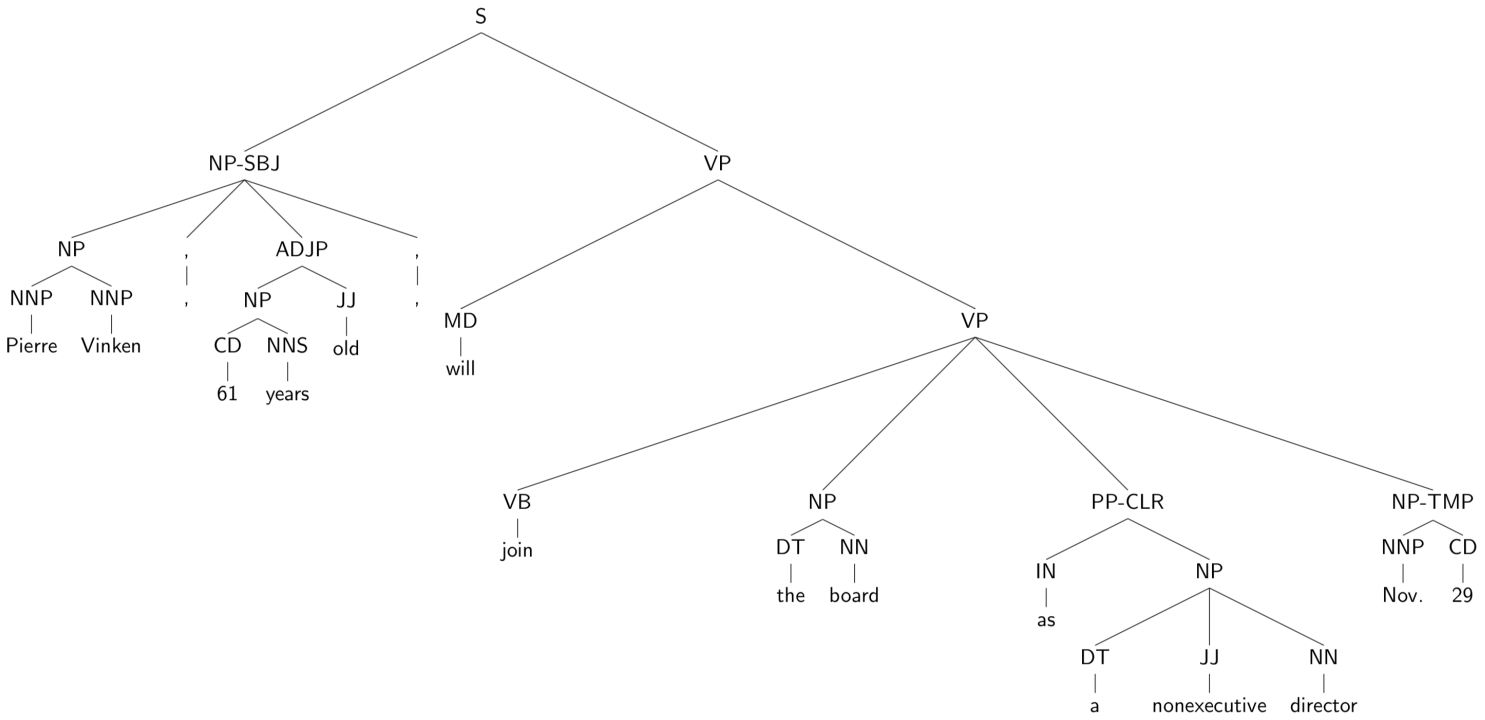
How many parsing actions in total?

- Three basic parsing actions
- N dependency relations

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

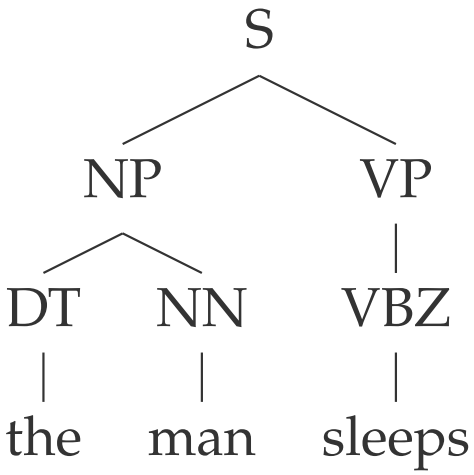
Stack	Input buffer	Action
(book, me)	(the, morning, flight)	RightArc-iobj

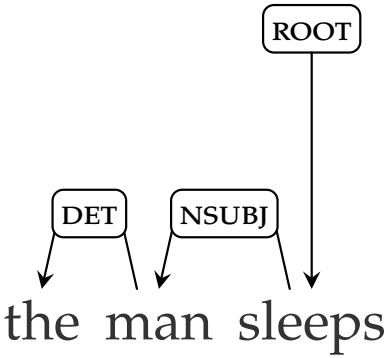
Training Corpus: Penn Treebank



Example

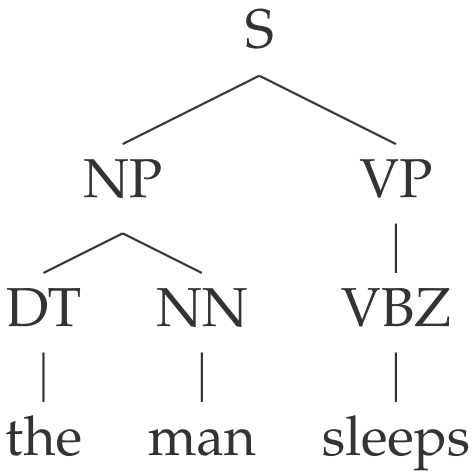
Convert a CFG tree to a dependency tree

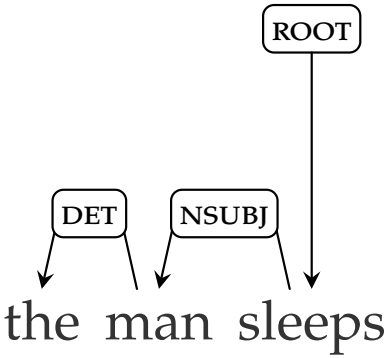




Example

Convert a CFG tree to a dependency tree





Question

Can we convert a dependency tree to a CFG tree?

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

Universal Dependencies

Universal Dependencies

Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD is an open community effort with over 200 contributors producing more than 100 treebanks in over 70 languages. If you're new to UD, you should start by reading the first part of the Short Introduction and then browsing the annotation guidelines.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

If you want to receive news about Universal Dependencies, you can subscribe to the UD mailing list. If you want to discuss individual annotation questions, use the Github issue tracker.

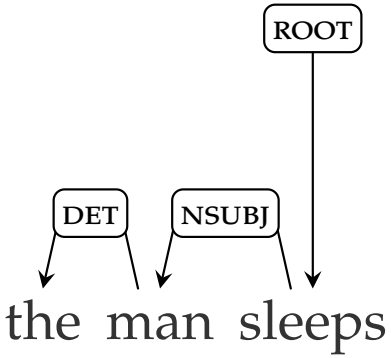
Current UD Languages

Information about language families (and genera for families with multiple branches) is mostly taken from WALS Online (IE = Indo-European).

Afrikaans	1	49K		IE, Germanic
Akkadian	1	1K		Afro-Asiatic, Semitic
Amharic	1	10K		Afro-Asiatic, Semitic
Ancient Greek	2	416K		IE, Greek
Arabic	3	1,042K	W	Afro-Asiatic, Semitic
Armenian	1	36K		IE, Armenian
Assyrian	1	<1K		Afro-Asiatic, Semitic
Bambara	1	13K		Mande
Basque	1	121K		Basque
Belarusian	1	13K		IE, Slavic
Breton	1	10K	W	IE, Celtic

Exercise: From a Dependency Tree to Parsing Actions

How to recover parsing actions from a dependency tree?



Summary

1. Dependency Grammars

2. Transition-Based Parsing

3. How to Build a Parser?

Reference

Jurafsky, D. and Martin, J. (2019).
Speech and language processing.