

# COMP5046

# Natural Language Processing

## Lecture 7: Dependency Parsing

Semester 1, 2019

School of Computer Science

The University of Sydney, Australia



THE UNIVERSITY OF  
**SYDNEY**

**Caren Han**

Caren.Han@sydney.edu.au

## Lecture 7: Parsing

1. **Linguistic Structure**
2. Dependency Structure
3. Dependency Parsing Algorithms
4. Transition-based Dependency Parsing
5. Deep Learning-based Dependency Parsing

## What is Syntax and Parsing

- Language is *more than just a “bag of words”*
- Grammatical rules *apply to categories and groups of words*, not individual words.
- **Example** – a sentence includes *a subject* and *a predicate*.  
The *subject* is noun phrase and the *predicate* is a verb phrases.
  - Noun phrase: The lecture, Caren, He
  - Verb phrase: started, had dinner, went away

### **Parsing**

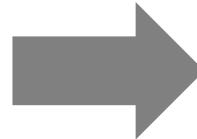
- Associating tree structures to a sentence, given a grammar  
(Context Free Grammar or Dependency Grammar)

## Parsing Computer Language

Source Code

---

```
"print: 4 * var + 1"
```



Abstract Syntax Tree

---

```
[id: "print"]
└─ [op: '+']
   └─ [op: '*']
      └─ [num: 4]
      └─ [id: "var"]
   └─ [num: 1]
```

# Linguistic Structure

## Parsing Natural Language (Human Language)

*It is much more difficult than parsing computer language*

*Why?*

- No **types** for words
- No **brackets** around phrases
- **Ambiguity!**

## Syntactic Ambiguities

### ***Grammars are declarative***

- *They don't specify how the parse tree will be constructed*

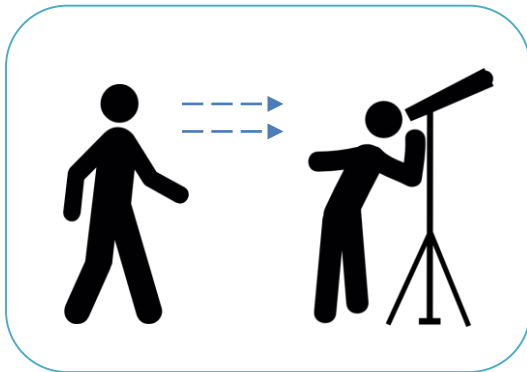
### ***Ambiguity***

1. *Prepositional Phrase (PP) attachment ambiguity*
2. *Verb attachment ambiguity*
3. *Coordination Scope*
4. *Gaps*
5. *Particles or Prepositions*
6. *Gerund or adjective*

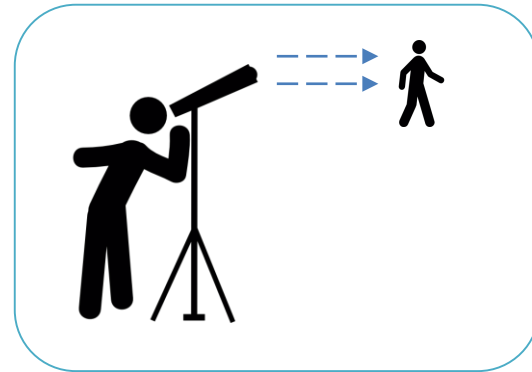
## Syntactic Ambiguities – PP attachment Ambiguity

*I saw the man with the telescope*

*I saw the man with the telescope*

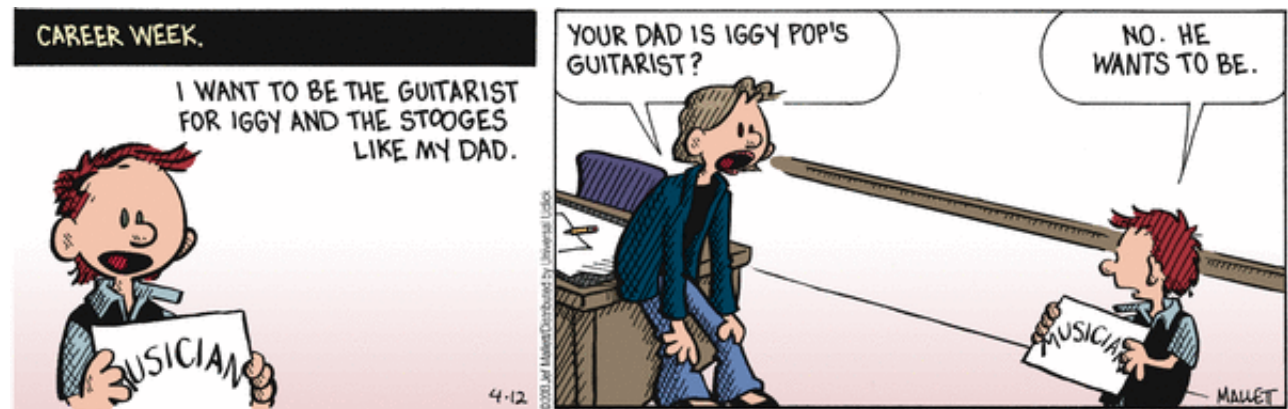


*I saw the man with the telescope*



## Syntactic Ambiguities – PP attachment Ambiguity Multiply

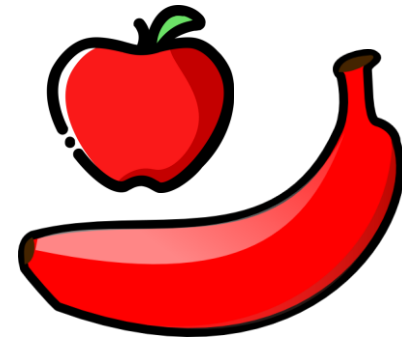
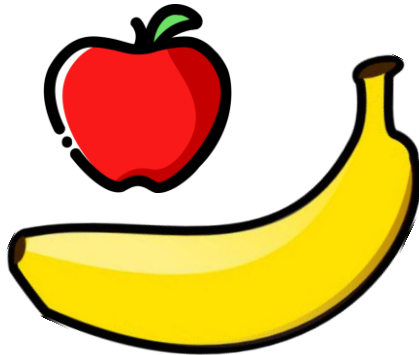
- A key parsing decision is how we ‘attach’ various constituents
  - *PPs, adverbial or participial phrases, infinitives, coordinations*





## Syntactic Ambiguities – Coordination Scope Ambiguity

*I ate red apples and bananas*



## Syntactic Ambiguities - Gaps

*She never saw a dog and did not smile*



## Syntactic Ambiguities – Particles or Prepositions

- Some verbs are followed by adverb particles.  
*(e.g. put on, take off, give away, bring up, call in)*

***She **ran up** a large bill***

- Some particles are detached from the verb and put after the object.

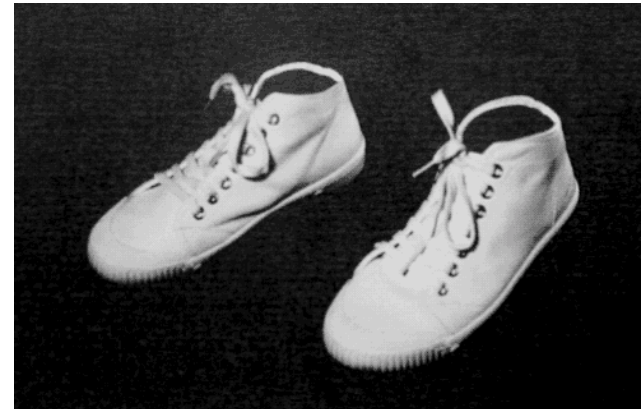
***He **called** the doctor **in*****

***Difference between an **adverb particle** and a **preposition**.***

- the **particle** is closely tied to its verb to form idiomatic expressions
- the **preposition** is closely tied to the noun or pronoun it modifies.

## Syntactic Ambiguities – Gerund or Adjective

*Dancing shoes* can provide nice experience



## When and Where do we use Parsing?

### ***Syntactic Analysis*** checks

*whether the generated tokens form a meaningful expression*

- *We need to understand sentence structure in order to be able to interpret language correctly*
- *Humans communicate complex ideas by composing words together into bigger units to convey complex meanings*
- *We need to know what is connected to what*
- *Grammar Checking*
- *Question Answering*
- *Machine Translation*
- *Information Extraction*
- *Chatbot*
- *... and many others*

## Two main views of linguistic structure

### ***Constituency Grammar*** (a.k.a context-free grammar, phrase structure grammar)

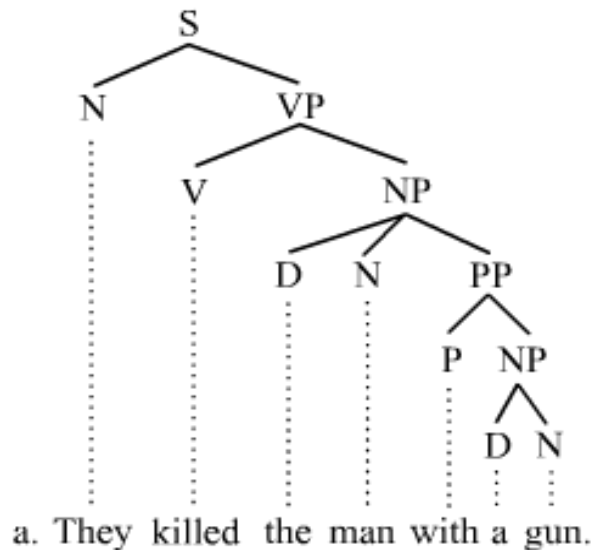
- *Noam Chomsky (1928 - )*
- *Immediate constituent analysis*
- *Insists on classification and distribution*

### ***Dependency Grammar***

- *Lucien Tesniere (1893 – 1954)*
- *Functional dependency relations*
- *Emphasises the relations between syntactic units, thus adding meaningful links (semantics)*

## Two main views of linguistic structure

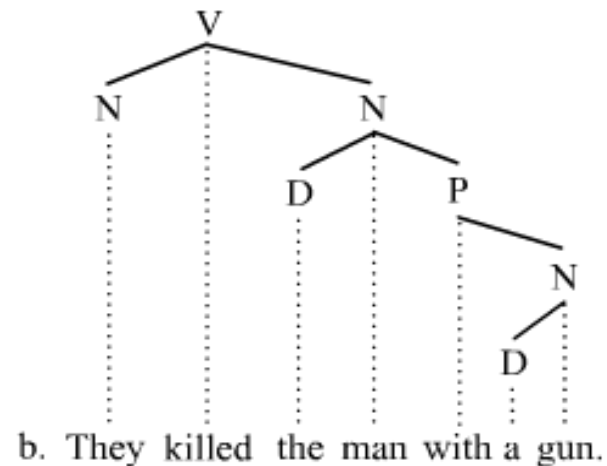
### Constituency Parsing



#### Constituency grammars

*One-to-one-or-more correspondence. For every word in a sentence, there is at least one node in the syntactic structure that corresponds to that word.*

### Dependency Parsing



#### Dependency grammars

*one-to-one relation; for every word in the sentence, there is exactly one node in the syntactic structure that corresponds to that word*

## Constituency Grammar

- A ***basic observation about syntactic structure*** is that groups of words can act as single units
- Such groups of words are called ***constituents***
- Constituents tend to have ***similar internal structure***, and behave similarly with respect to other units

### ***Examples***

- ***noun phrases (NP)***
  - she, the house, Robin Hood and his merry men, etc.
- ***verb phrases (VP)***
  - blushed, loves Mary, was told to sit down and be quiet, lived happily ever after
- ***prepositional phrases (PP)***
  - on it, with the telescope, through the foggy dew, etc.



A sample context-free grammar

***I prefer a morning flight***

1. Starting unit: words are *given a category (part-of-speech)*
2. Combining words into *phrases with categories*
3. Combining phrases into *bigger phrases* recursively

## A sample context-free grammar

1. Starting unit: words are *given a category (part-of-speech)*
2. Combining words into phrases with categories
3. Combining phrases into bigger phrases recursively

*I, prefer, a, morning, flight*

*PRP*

*VBP*

*DT*

*NN*

*NN*

## A sample context-free grammar

*I, prefer, a, morning, flight*

*PRP*

*VBP*

*DT*

*NN*

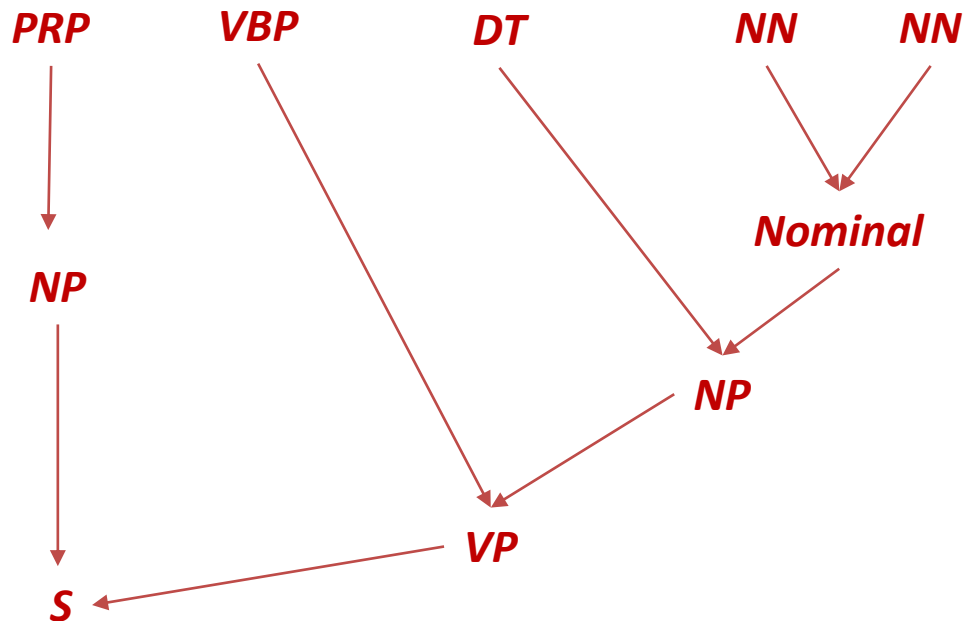
*NN*

Grammar rule	Example
$S \rightarrow \text{NPVP}$	I + want a morning flight
$\text{NP} \rightarrow \text{Pronoun}$	I
$\text{NP} \rightarrow \text{Proper-Noun}$	Sydney
$\text{NP} \rightarrow \text{Det Nominal}$	a flight
$\text{Nominal} \rightarrow \text{Nominal Noun}$	morning flight
$\text{Nominal} \rightarrow \text{Noun}$	flights
$\text{VP} \rightarrow \text{Verb}$	do
$\text{VP} \rightarrow \text{Verb NP}$	want + a flight
$\text{VP} \rightarrow \text{Verb NPPP}$	leave + Melbourne + in the morning
$\text{VP} \rightarrow \text{VerbPP}$	leaving + on Thursday
$\text{PP} \rightarrow \text{Preposition NP}$	from + Sydney

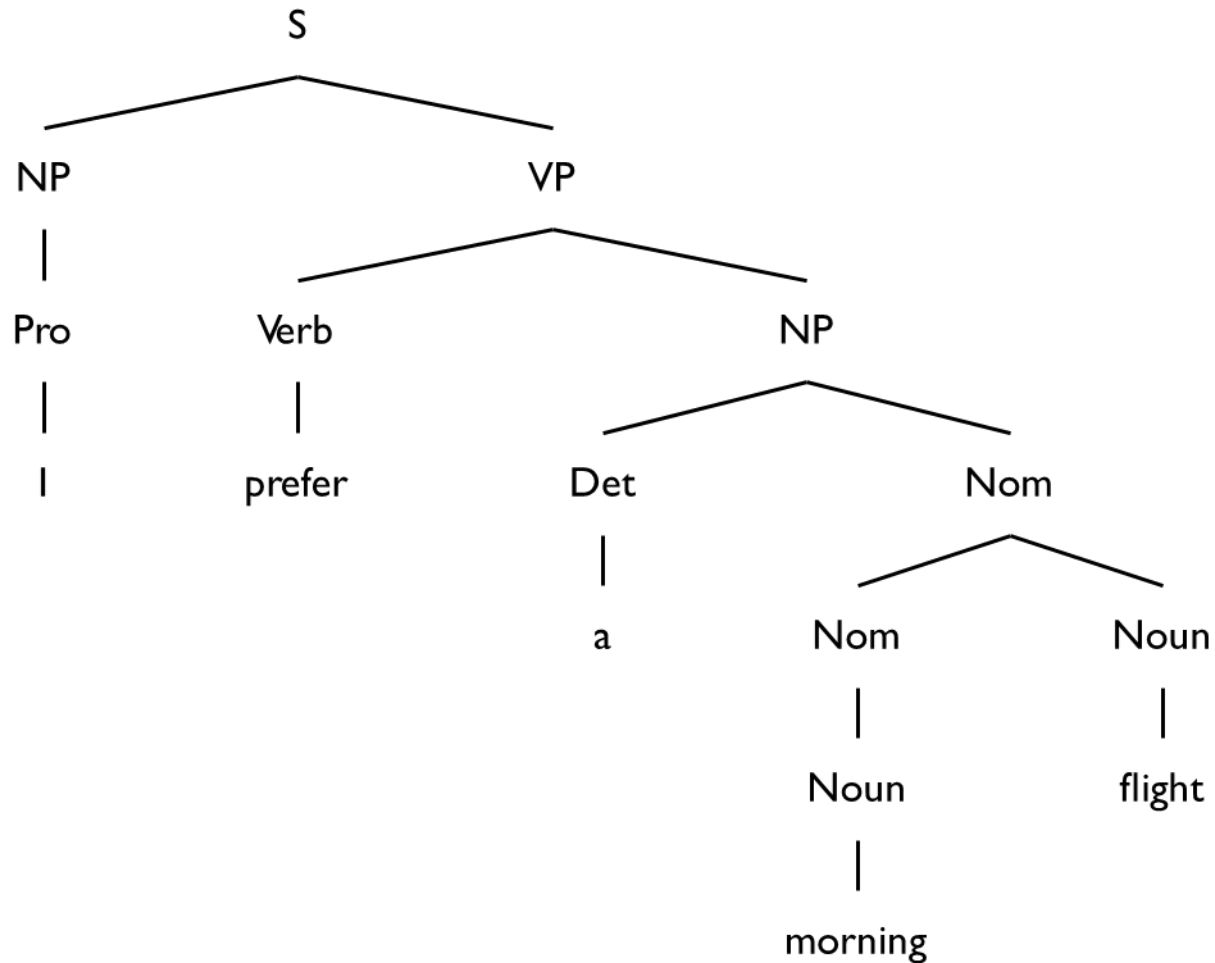
## A sample context-free grammar

1. Starting unit: words are given a category (part-of-speech)
2. Combining words into **phrases with categories**
3. Combining phrases into **bigger phrases** recursively

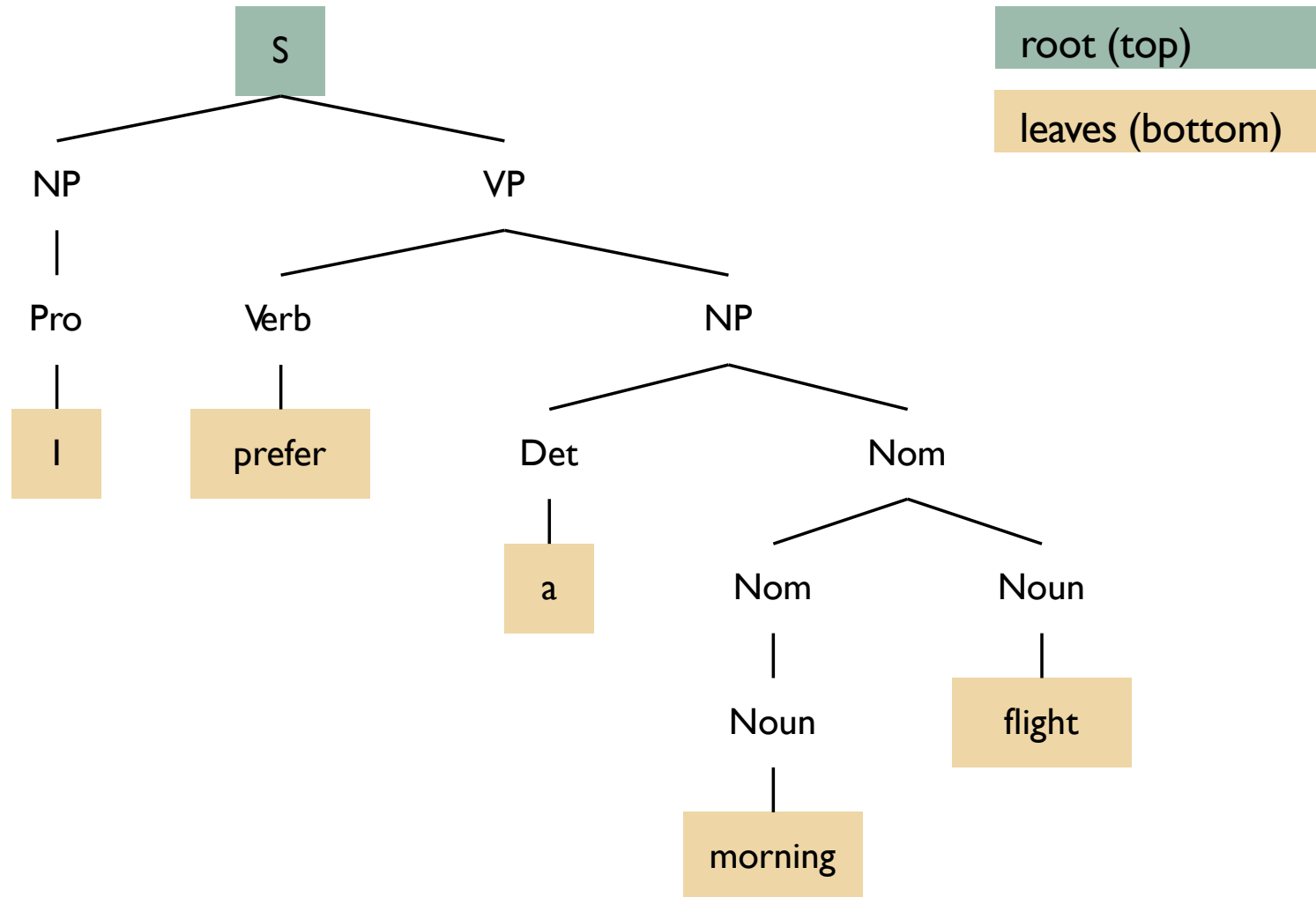
*I, prefer, a, morning, flight*



## A sample context-free grammar



## A sample context-free grammar



## Treebanks

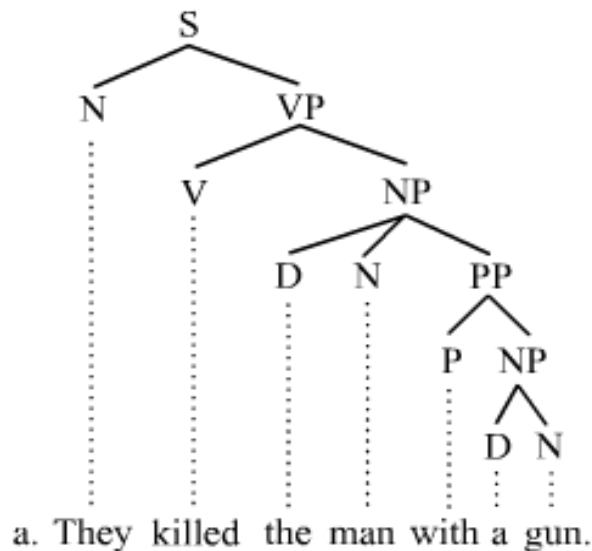
*Corpora where each sentence is annotated with a parse tree*

- *Treebanks are generally created by*
  - *parsing texts with an existing parser*
  - *having human annotators correct the result*
- *This requires detailed annotation guidelines for annotating different grammatical constructions*
- *Penn Treebank is a popular treebank for English (Wall Street Journal section)*

```
( (S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken) )
    (, ,)
    (ADJP
      (NP (CD 61) (NNS years) )
      (JJ old) )
    (, ,) )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board) )
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) ))
      (NP-TMP (NNP Nov.) (CD 29) )))
  (. .) ))
```

## Two main views of linguistic structure

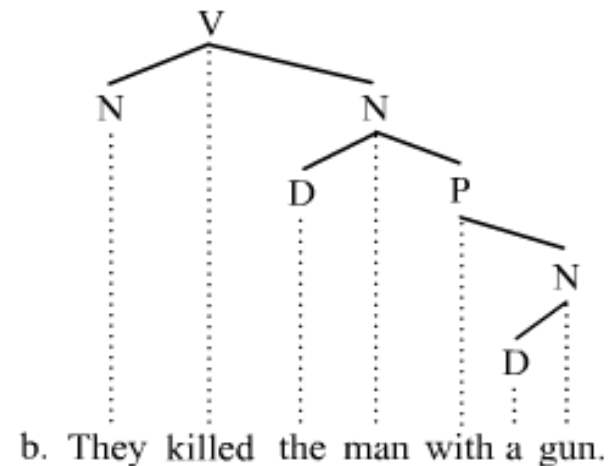
### Constituency Parsing



#### Constituency grammars

*One-to-one-or-more correspondence. For every word in a sentence, there is at least one node in the syntactic structure that corresponds to that word.*

### Dependency Parsing



#### Dependency grammars

*one-to-one relation; for every word in the sentence, there is exactly one node in the syntactic structure that corresponds to that word*

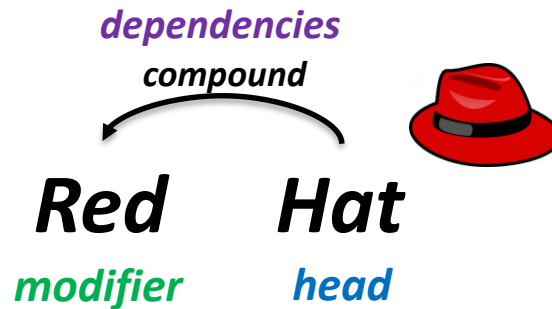


## Lecture 7: Parsing

1. Linguistic Structure
2. **Dependency Structure**
3. Dependency Parsing Algorithms
4. Transition-based Dependency Parsing
5. Deep Learning-based Dependency Parsing

## Dependency Structure

**Syntactic structure:** lexical items linked by binary asymmetrical relations (“arrows”) called **dependencies**



**Red** – **modifier**, dependent, child, subordinate

**Hat** - **head**, governor, parent, regent

**Compound** – **dependency relations** (e.g. subject, prepositional object, etc)

\***Head** determines the syntactic/semantic category of the construct

\*The arrows are commonly typed with the name of **grammatical relations**

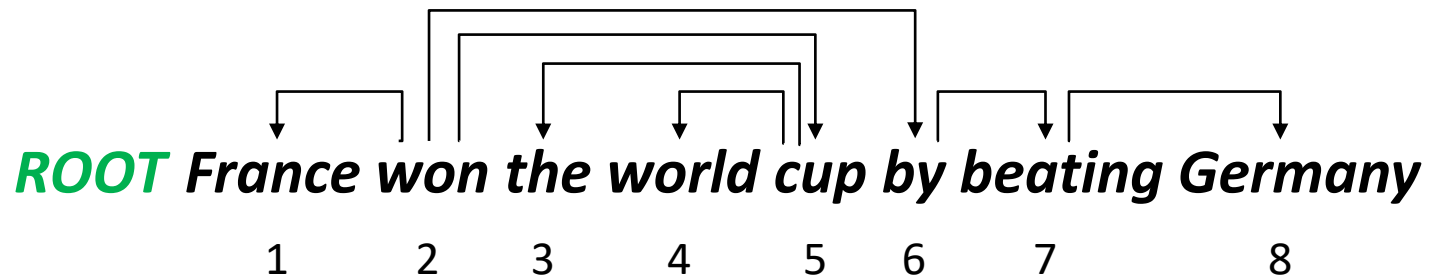
## Dependency Relations

*The following list shows the 37 universal **syntactic relations** used in UD v2. (de Marneffe et al. 2014).*

- [acl](#): clausal modifier of noun (adjectival clause)
- [advcl](#): adverbial clause modifier
- [advmod](#): adverbial modifier
- [amod](#): adjectival modifier
- [appos](#): appositional modifier
- [aux](#): auxiliary
- [case](#): case marking
- [cc](#): coordinating conjunction
- [ccomp](#): clausal complement
- [clf](#): classifier
- [compound](#): compound
- [conj](#): conjunct
- [cop](#): copula
- [csubj](#): clausal subject
- [dep](#): unspecified dependency
- [det](#): determiner
- [discourse](#): discourse element
- [dislocated](#): dislocated elements
- [expl](#): expletive
- [fixed](#): fixed multiword expression
- [flat](#): flat multiword expression
- [goeswith](#): goes with
- [iobj](#): indirect object
- [list](#): list
- [mark](#): marker
- [nmod](#): nominal modifier
- [nsubj](#): nominal subject
- [nummod](#): numeric modifier
- [obj](#): object
- [obl](#): oblique nominal
- [orphan](#): orphan
- [parataxis](#): parataxis
- [punct](#): punctuation
- [reparandum](#): overridden disfluency
- [root](#): root
- [vocative](#): vocative
- [xcomp](#): open clausal complement

## Dependency Structure

*Example*

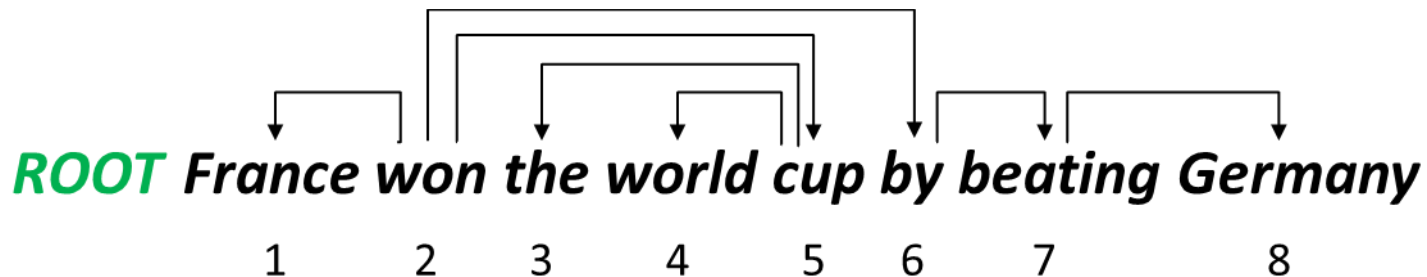


## Dependency Parsing


- Represents Lexical/syntactic dependencies between words
  - A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of

*Dependencies form a tree (connected, acyclic, single-head)*

- *How to make the dependencies a tree - Constraints*
  - Only one word is a dependent of ROOT (the main predicate of a sentence)
  - Don't want cycles  $A \rightarrow B, B \rightarrow A$



## Dependency Grammar/Parsing History



***Panini's grammar*** (4th century BCE)

*The notion of dependencies between grammatical units*

***Ibn Maḍā'*** (12th century)

*The first grammarian to use the term dependency in the grammatical sense*

***Sámuel Brassai, Franz Kern, Heimann Hariton Tiktin*** (1800 - 1930)

*The dependency seems to have coexisted side by side with that of phrase structure*

***Lucien Tesnière*** (1959)

*Was dominant approach in "East" in 20th Century (Russia, China, ...)*

*Good for free-er word order languages*

***David Hays*** (1962)

*The great development surrounding dependency-based theories has come from computational linguistics*



## The rise of annotated data

- The idea of dependency structure goes back a long way
- *[Universal Dependencies: <http://universaldependencies.org/> ;*
- *cf. Marcus et al. 1993, The Penn Treebank, Computational Linguistics]*
- Starting off, building a treebank seems a lot slower and less useful than building a grammar
- But a treebank gives us many things
  - Reusability of the labor
    - Many parsers, part-of-speech taggers, etc. can be built on it
    - Valuable resource for linguistics
  - Broad coverage, not just a few intuitions
  - Frequencies and distributional information
  - A way to evaluate systems



## Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

- Bilexical affinities
- Dependency distance
- Intervening material
- Valency of heads

# Dependency Structure

## Dependency Parsing

*Exercise – Let's do it together!*

- Simpler to parse than context-free grammars

**ROOT** *I prefer the morning flight through Sydney*

**ROOT** *Unionised workers are usually better paid than their non-union counterparts*

## Lecture 7: Parsing

1. Linguistic Structure
2. Dependency Structure
3. **Dependency Parsing Algorithms**
4. Transition-based Dependency Parsing
5. Deep Learning-based Dependency Parsing

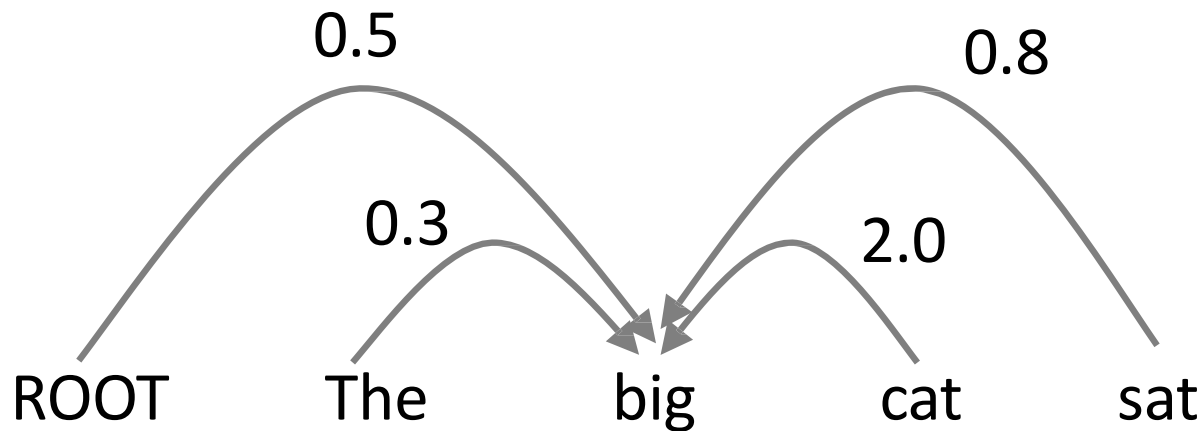
## Methods of Dependency Parsing

- Dynamic programming
  - Eisner (1996) gives a clever algorithm with complexity  $O(n^3)$ , by producing parse items with heads at the ends rather than in the middle
- Constraint Satisfaction
  - Edges are eliminated that don't satisfy hard constraints. Karlsson (1990)
- ***Graph-based Dependency Parsing***
  - Create a ***Minimum Spanning Tree*** for a sentence
  - McDonald et al.'s (2005) MSTParser scores dependencies independently using a Machine Learning classifier
- ***Transition-based Dependency Parsing***
- ***Neural Network-based Dependency Parsing***

# Dependency Parsing Approaches

## Graph-based dependency parsers

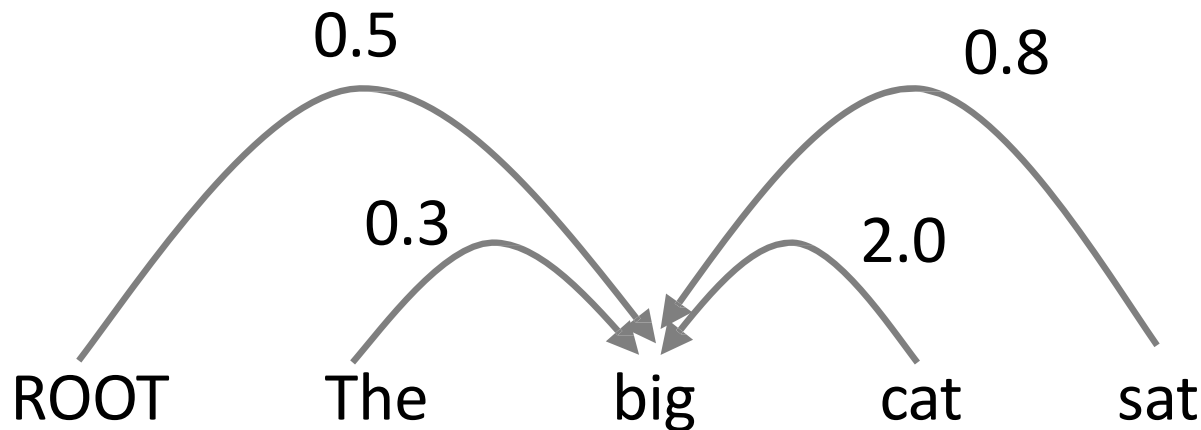
- Compute a score for every possible dependency for each edge



e.g., picking the head for "big"

## Graph-based dependency parsers

- Compute a score for every possible dependency for each edge
  - Then add an edge from each word to its highest-scoring candidate head
  - And repeat the same process for each other word



e.g., picking the head for “big”

## Methods of Dependency Parsing

### *Graph-based Dependency Parsing*

- Non-deterministic dependency parsing
- Build a complete graph with directed/weighted edges
- Find the highest scoring tree from a complete dependency graph

### *Transaction-based Dependency Parsing*

- Deterministic dependency parsing
- Build a tree by applying a sequence of transition actions
- Find the highest scoring action sequence that builds a legal tree

*greedy algorithm*



## Lecture 7: Parsing

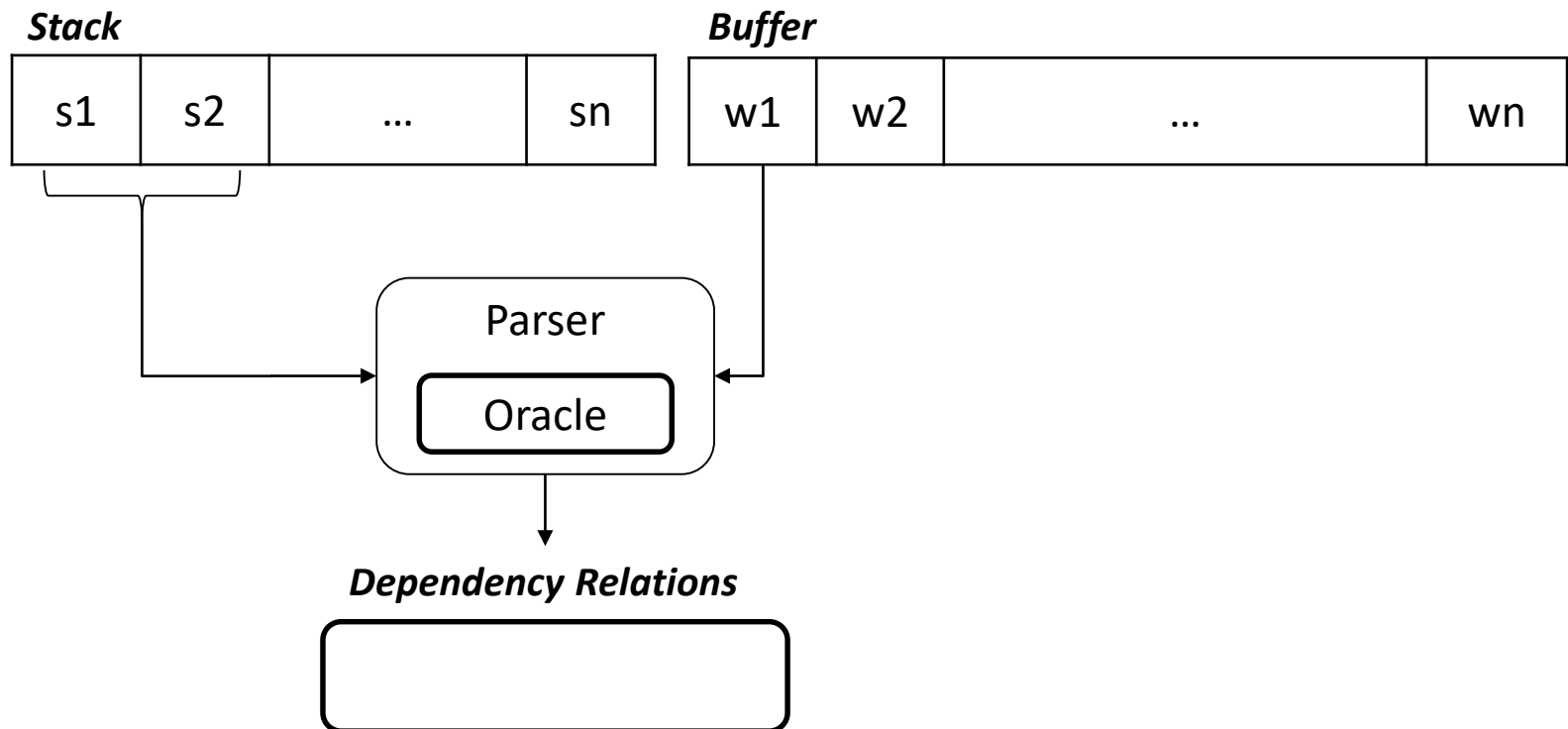
1. Linguistic Structure
2. Dependency Structure
3. Dependency Parsing Algorithms
4. **Transition-based Dependency Parsing**
5. Deep Learning-based Dependency Parsing



## Greedy transition-based parsing

- A simple form of greedy discriminative dependency parser
- Design a dumber but really fast algorithm and let the machine learning do the rest.
- Eisner's algorithm (Dynamic Programming-based Dependency Parsing) searches over many different dependency trees at the same time.
- A transition-based dependency parser only builds one tree, in one left-to-right sweep over the input

## Greedy transition-based parsing



## Transition-based parsing – The arc-standard algorithm

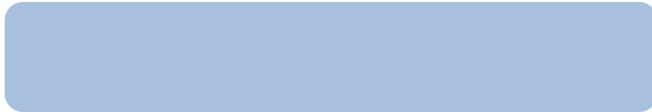
- The arc-standard algorithm is a simple algorithm for transition-based dependency parsing.
- A sequence of bottom up actions
  - Roughly like “shift” or “reduce” in a shift-reduce parser, but the “reduce” actions are specialized to create dependencies with head on left or right
- It is implemented in most practical transition-based dependency parsers, including MaltParser.

## 4

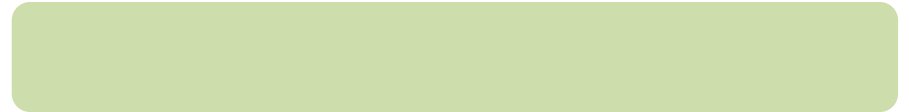
# Transition-based Parsing

## Transition-based parsing – The arc-standard algorithm

*Stack*



*Buffer*



*Dependency Graph*



## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT

*Buffer*

book

me

a

morning

flight

*Dependency Graph*

- **Initial configuration:**
  - All words are in the buffer.
  - The stack is empty or starts with the ROOT symbol
  - The dependency graph is empty.

## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT

*Buffer*

book

me

a

morning

flight

*Dependency Graph*

*Possible Transaction*

Shift

- **Push** the next word in buffer onto the stack

Left-Arc

- Add an arc **from the topmost word to the 2<sup>nd</sup>-topmost word** on the stack
- Remove 2<sup>nd</sup> word from stack

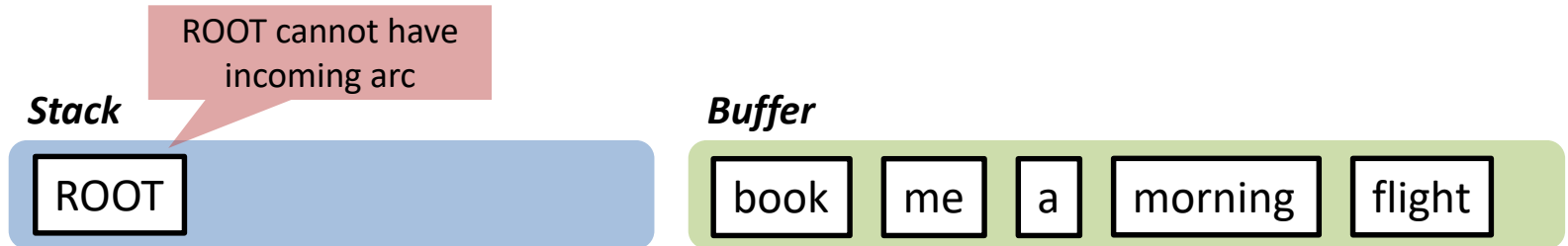
Right-Arc

- Add an arc **from the 2<sup>nd</sup>-topmost word to the topmost word** on the stack
- Remove the topmost word from stack

# 4

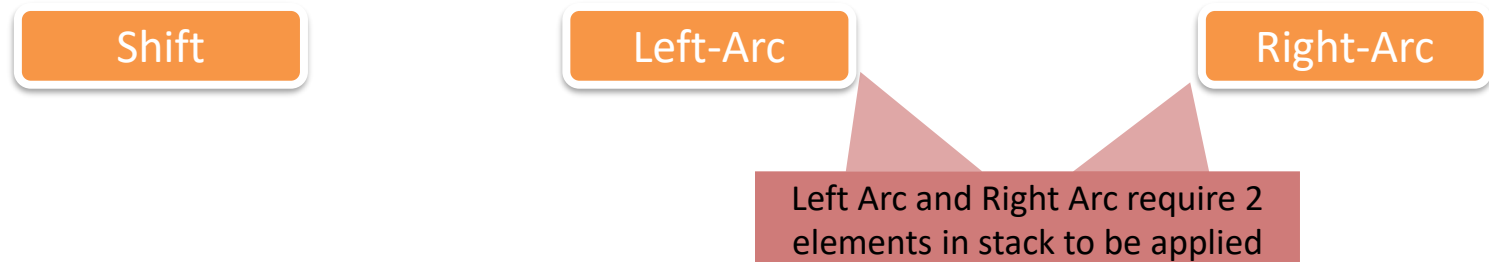
## Transition-based Parsing

### Transition-based parsing – The arc-standard algorithm



#### Dependency Graph

#### Possible Transaction



## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT

*Buffer*

book

me

a

morning

flight

*Dependency Graph*

*Possible Transaction*

Shift

Left-Arc

Right-Arc



## 4

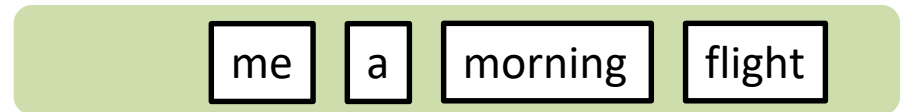
# Transition-based Parsing

## Transition-based parsing – The arc-standard algorithm

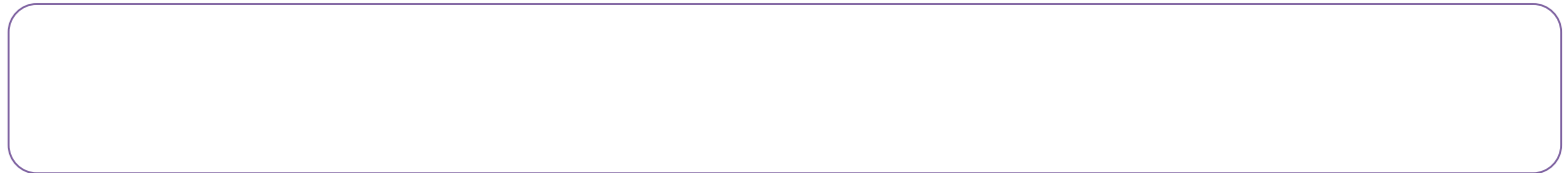
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

## 4

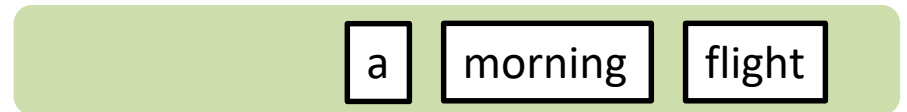
# Transition-based Parsing

## Transition-based parsing – The arc-standard algorithm

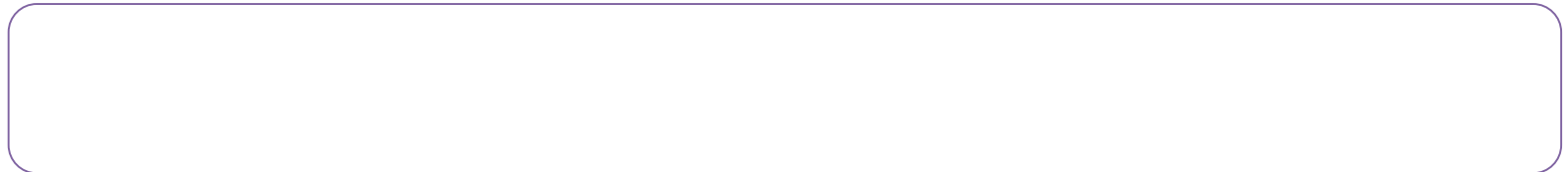
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

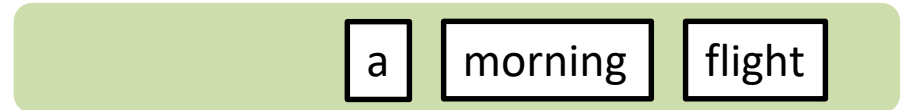
Right-Arc

## Transition-based parsing – The arc-standard algorithm

*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

# 4

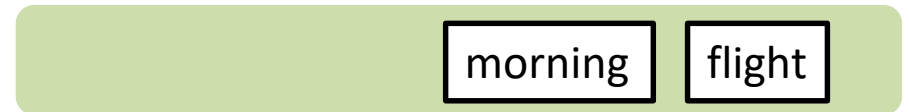
## Transition-based Parsing

### Transition-based parsing – The arc-standard algorithm

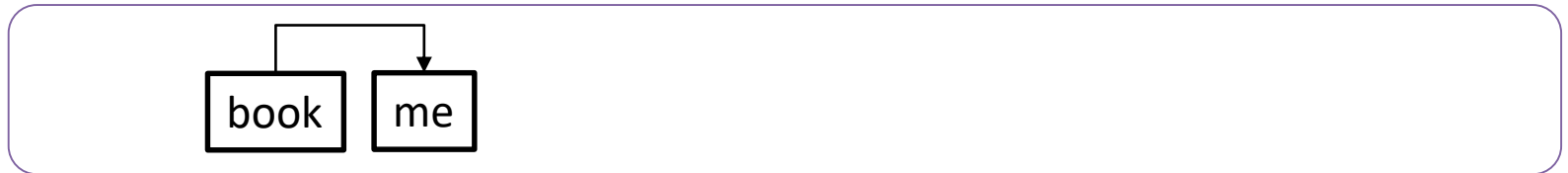
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

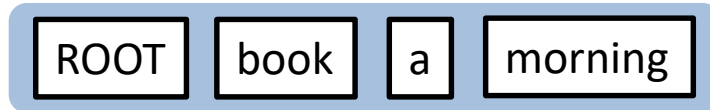
Right-Arc

## 4

# Transition-based Parsing

## Transition-based parsing – The arc-standard algorithm

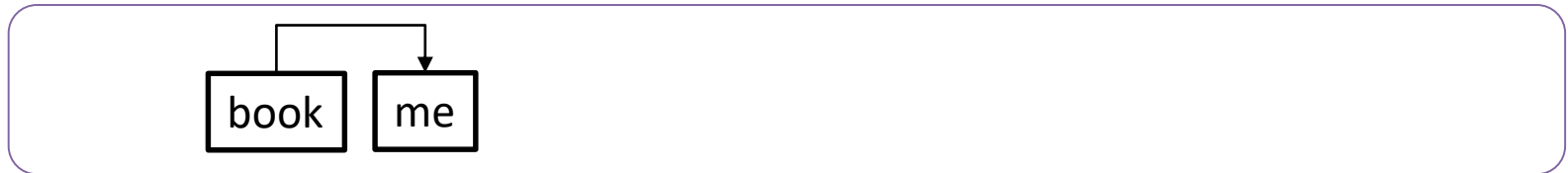
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

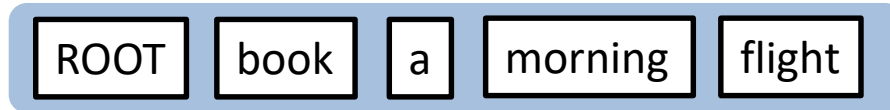
Right-Arc

## 4

# Transition-based Parsing

## Transition-based parsing – The arc-standard algorithm

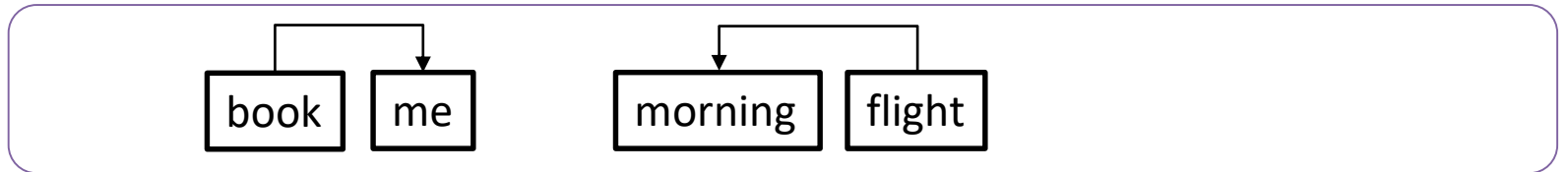
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

# 4

## Transition-based Parsing

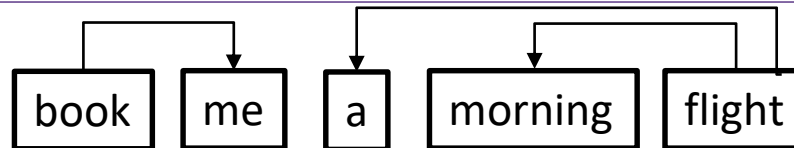
### Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT book a flight

*Buffer*

*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

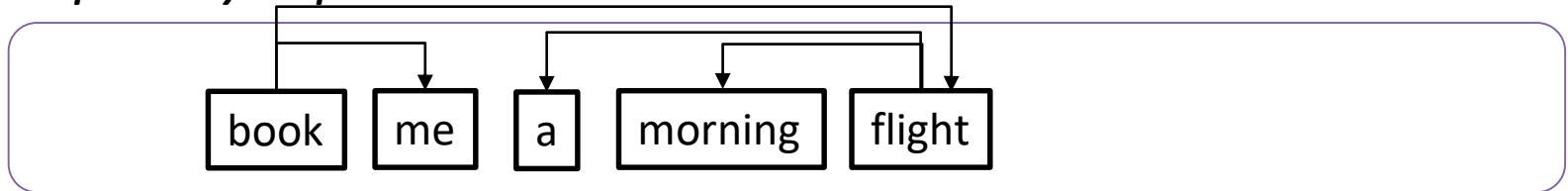
## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT book flight

*Buffer*

*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc



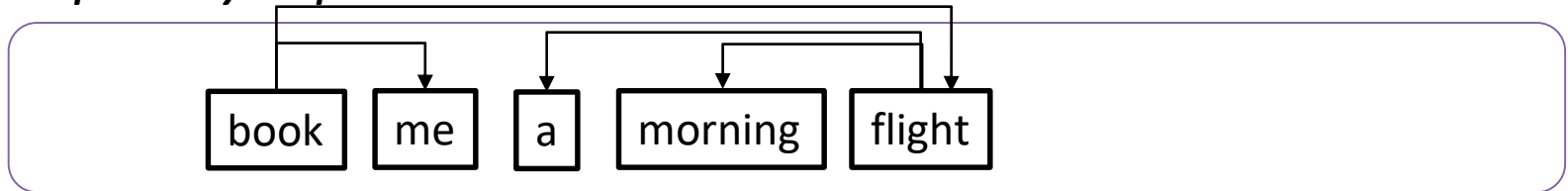
## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT book

*Buffer*

*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

# 4

## Transition-based Parsing

### Transition-based parsing – The arc-standard algorithm

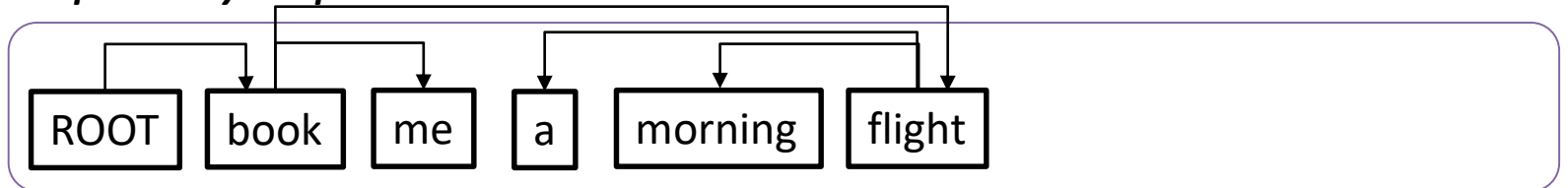
*Stack*



*Buffer*



*Dependency Graph*



*Possible Transaction*

Shift

Left-Arc

Right-Arc

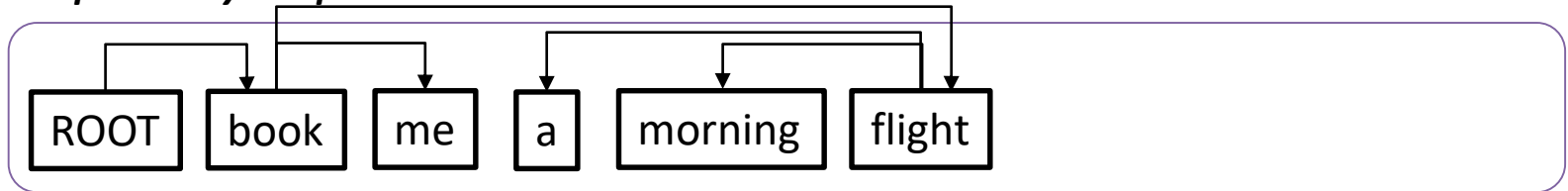
## Transition-based parsing – The arc-standard algorithm

*Stack*

ROOT

*Buffer*

*Dependency Graph*



- ***Terminal configuration:***
  - The buffer is empty.
  - The stack contains a single word.

## Transition-based parsing – The arc-standard algorithm

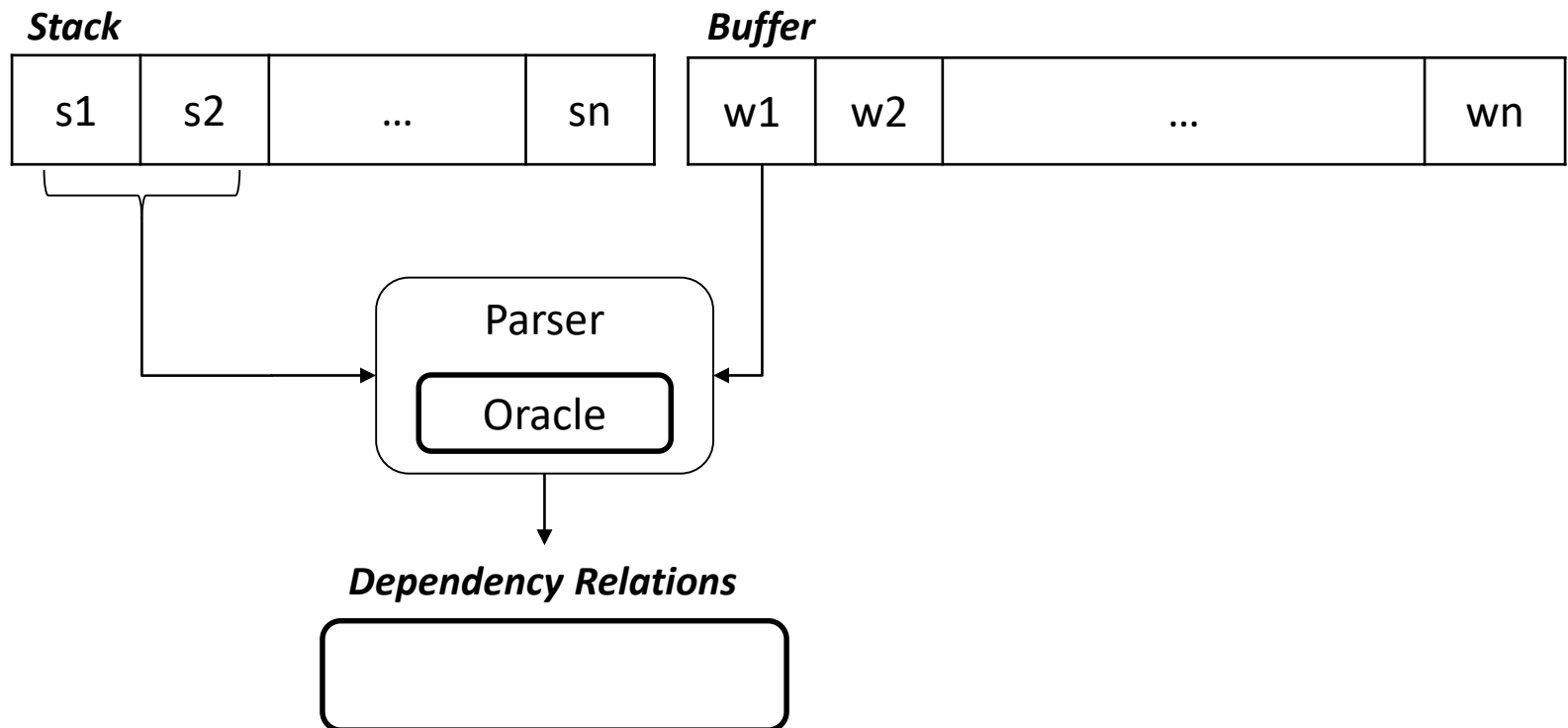
**Start:**  $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

1. Shift  $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
2. Left-Arc<sub>r</sub>  $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$   $\sigma | w_i | w_j, \beta, A \rightarrow$
3. Right-Arc<sub>r</sub>  $\sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$

**Finish:**  $\sigma = [w], \beta = \emptyset$

***How to choose the next action?***

## Transition-based Parsing



## How to choose the next action?

**Stand back, You know machine learning!**

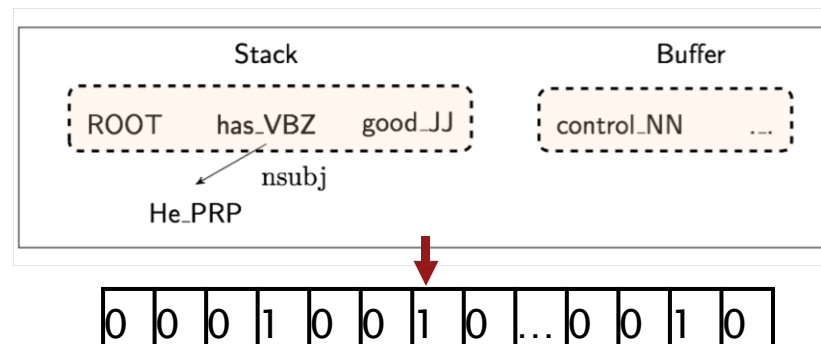
**Goal:** Predict the next transition (class), given the current configuration.

- We let the parser run on gold-standard trees.
- Every time there is a choice to make, we simply look into the tree and do ‘the right thing’.
- We collect all (configuration, transition) pairs and train a classifier on them.
- When parsing unseen sentences, we use the trained classifier as a guide.

***What if the number of pairs is far too large?***

## Feature Representation

- Define a set of features of configurations that you consider to be relevant for the task of predicting the next transition.
- Example: word forms of the topmost two words on the stack and the next two words in the buffer
- Describe every configuration in terms of a feature vector.



- In practical systems, we have thousands of features and hundreds of transitions.
- There are several machine-learning paradigms that can be used to train a guide for such a task
- Examples: perceptron, decision trees, support-vector machines, memory-based learning

## Transition-based parsing



(a) Arc-standard: *is* and *example* are eligible for arcs.



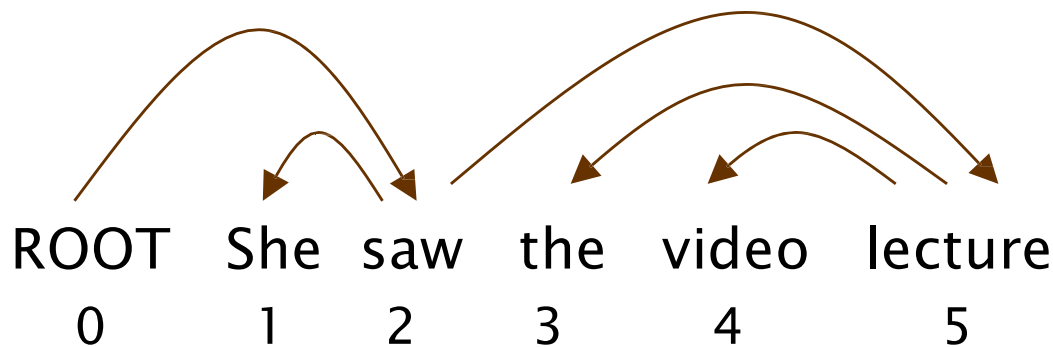
(b) Arc-eager: *example* and *with* are eligible for arcs.



(c) Easy-first: All unreduced tokens are active (bolded).



## Evaluation of Dependency Parsing



$$\text{Acc} = \frac{\text{\# correct deps}}{\text{\# of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

### Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

### Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

## Lecture 7: Parsing

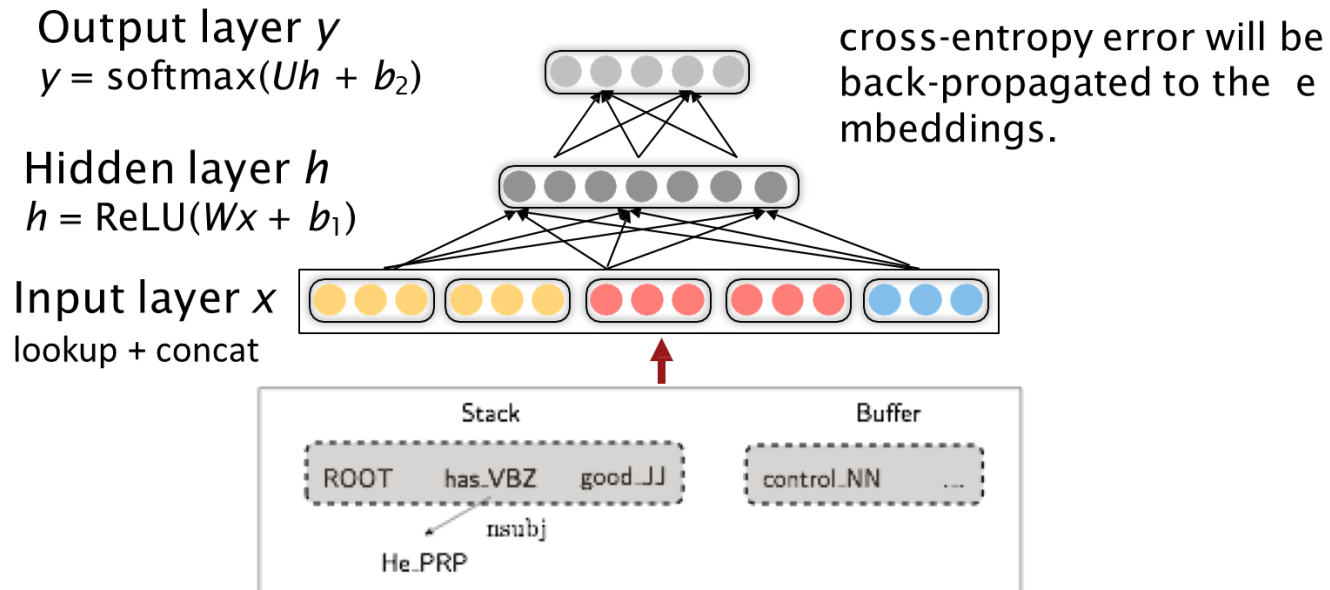
1. Linguistic Structure
2. Dependency Structure
3. Dependency Parsing Algorithms
4. Transition-based Dependency Parsing
5. **Deep Learning-based Dependency Parsing**

## Distributed Representations

- Represent each word as a d-dimensional dense vector (i.e., word embedding)
  - Similar words are expected to have close vectors.
  - NNS (plural noun) should be close to NN (singular noun).
- Meanwhile, part-of-speech tags (POS) and dependency labels are also represented as d-dimensional vectors.
- The smaller discrete sets also exhibit many semantical similarities

## Distributed Representations

### Softmax probabilities



## Neural Network Dependency Parsing

*Accuracy and parsing speed  
on PTB + Stanford dependencies.*

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	<b>92.0</b>	<b>89.7</b>	<b>91.8</b>	<b>89.6</b>	<b>654</b>

*Accuracy and parsing speed on CTB*

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	<b>84.0</b>	82.1	83.0	81.2	6
Our parser	<b>84.0</b>	<b>82.4</b>	<b>83.9</b>	<b>82.4</b>	<b>936</b>

Chen, D., & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 740-750).

- PTB: English Penn Treebank
- CTB: Chinese Penn Treebank



# Final Exercise

Thank you !

Any questions ?

## Reference for this lecture

- Deng, L., & Liu, Y. (Eds.). (2018). Deep Learning in Natural Language Processing. Springer.
- Rao, D., & McMahan, B. (2019). Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. " O'Reilly Media, Inc.".
- Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.
- Nivri, J (2016). Transition-based dependency parsing, lecture notes, Uppsala Universitet
- Manning, C 2017, Natural Language Processing with Deep Learning, lecture notes, Stanford University
- Chen, D., & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 740-750).
- Eisner, J. M. (1996, August). Three new probabilistic models for dependency parsing: An exploration. In Proceedings of the 16th conference on Computational linguistics-Volume 1 (pp. 340-345). Association for Computational Linguistics.