COMP5046
Natural Language Processing

Lecture 7: Dependency Parsing

Semester 1, 2020

School of Computer Science
The University of Sydney, Australia

THE UNIVERSITY OF SYDNEY

Dr Caren Han
Caren.Han@sydney.edu.au

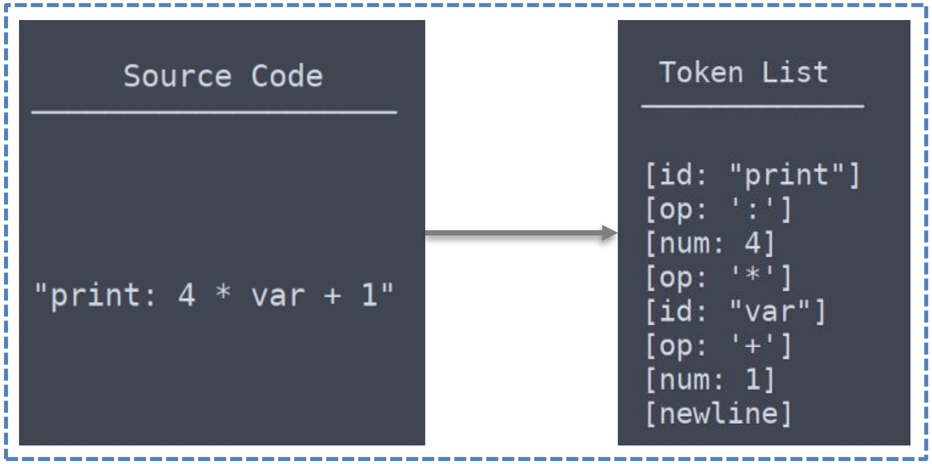
0 LECTURE PLAN

Lecture 7: Parsing

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

1 Linguistic Structure

Computer Language



Tokenisation

A token can be a variable or function name, an operator or a number.

1 Linguistic Structure

Parsing Computer Language

Parsing

The parser turns a list of tokens into a tree of nodes – logical order

Can we apply this in a human (natural) language?

Parsing Natural Language (Human Language)

Q: Can we apply this in a human (natural) language?
A: Possible! But it is much more difficult than parsing computer language!

Why?

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

Natural Language: Linguistic Structure

Let’s try to categorise the given words (Part of Speech Tags)

The expensive computer is on the table

DT Adj N V P DT N

However, Language is **more than just a “bag of words”.**
Grammatical rules apply to combine:

- words **into phrases**
- phrases into bigger phrases

Natural Language: Linguistic Structure

Let's try to categorise the given words (Part of Speech Tags)

The expensive computer is on the table

DT Adj N V P DT N

However, Language is **more than just a "bag of words".**
Grammatical rules apply to combine:

- words **into phrases**
- phrases into bigger phrases

Example*: a sentence includes* ***a subject*** *and* ***a predicate****.*
The ***subject*** *is noun phrase and the* ***predicate*** *is a verb phrases.*

1 Linguistic Structure

Natural Language: Linguistic Structure

Phrase Structure Grammar = Context-free Grammar (CFG)

The expensive computer is on the table

DT Adj N V P DT N

However, Language is **more than just a "bag of words".**
Grammatical rules apply to combine:

- words **into phrases**
- phrases into bigger phrases

Example*: a sentence includes* ***a subject*** *and* ***a predicate****.*
The ***subject*** *is noun phrase and the* ***predicate*** *is a verb phrases.*

Natural Language: Linguistic Structure

Phrase Structure Grammar = Context-free Grammar (CFG)

The expensive computer is on the table

DT Adj N V P DT N

However, Language is **more than just a "bag of words".**
Grammatical rules apply to combine:

- words **into phrases**
- phrases into bigger phrases

Parsing

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

will talk about this soon!

Parsing Natural Language (Human Language)

Q: Can we apply this in a human (natural) language?
A: Possible! But 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

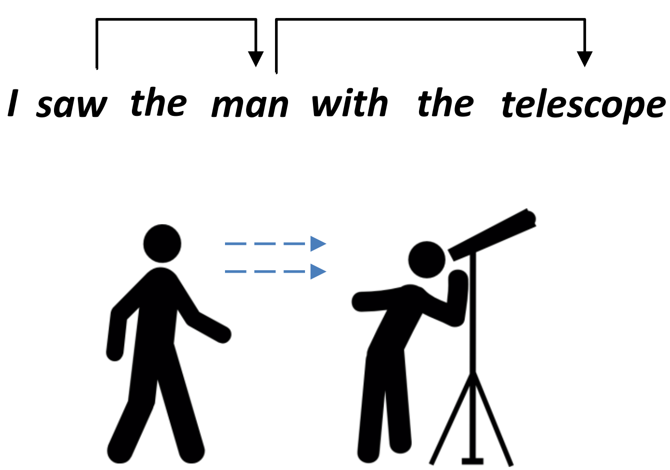
1. *Prepositional Phrase (PP) attachment ambiguity*
2. *Coordination Scope*
3. *Gaps*
4. *Particles or Prepositions*
5. *Gerund or adjective*

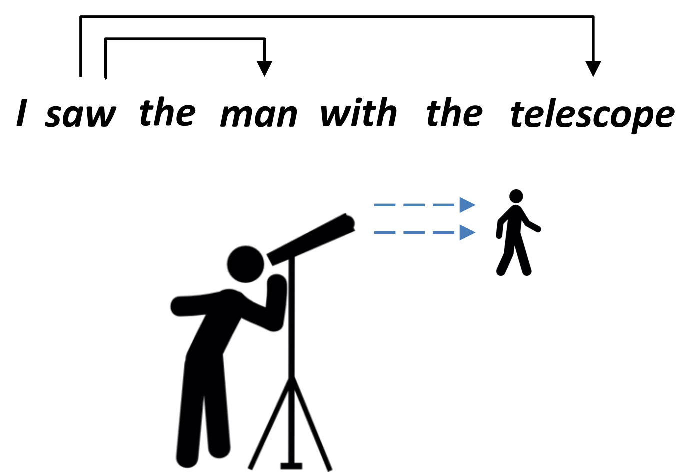
There are many more ambiguities…

1 Linguistic Structure

Syntactic Ambiguities – PP attachment Ambiguity

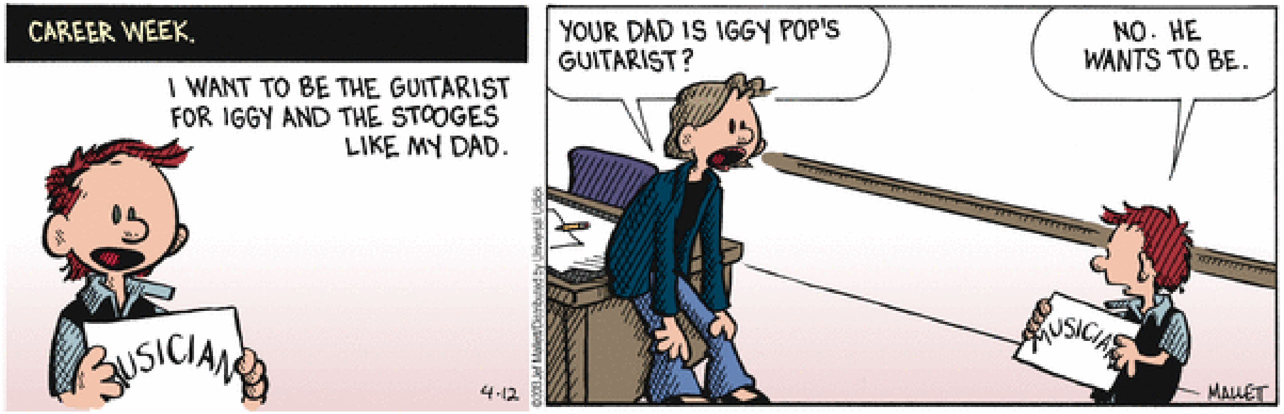
I saw the man with the telescope





Syntactic Ambiguities – PP attachment Ambiguity Multiply

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



THE UNIVERSITY OF SYDNEY

Syntactic Ambiguities – Coordination Scope Ambiguity

I ate red apples and bananas

1 Linguistic Structure

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

She ran up a large hill

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.*

1 Linguistic Structure

Syntactic Ambiguities – Gerund or Adjective

Dancing shoes can provide nice experience

Gerund

Adjective

When and Where do we use Parsing?

Syntactic Analysis *checks whether the generated tokens form a meaningful expression*

- *Humans communicate complex ideas by composing words together into bigger units to convey complex meanings*
- *We need to understand sentence structure in order to be able to interpret language correctly*

- *Grammar Checking*
- *Question Answering*
- *Machine Translation*
- *Information Extraction*
- *Text Classification*
- *Chatbot*

… and many others

1 Linguistic Structure

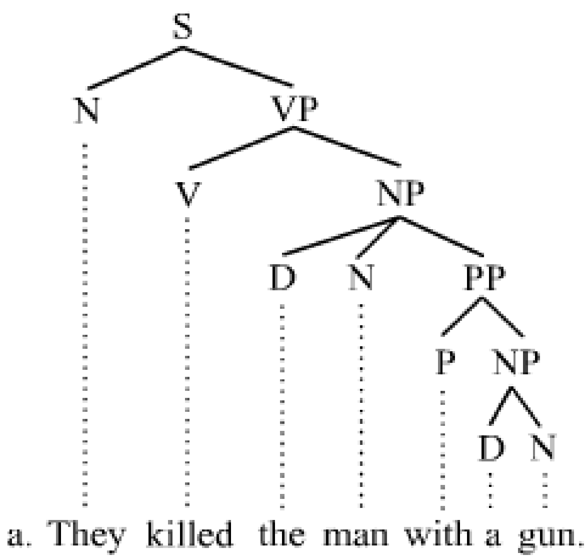
Two main views of linguistic structure

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

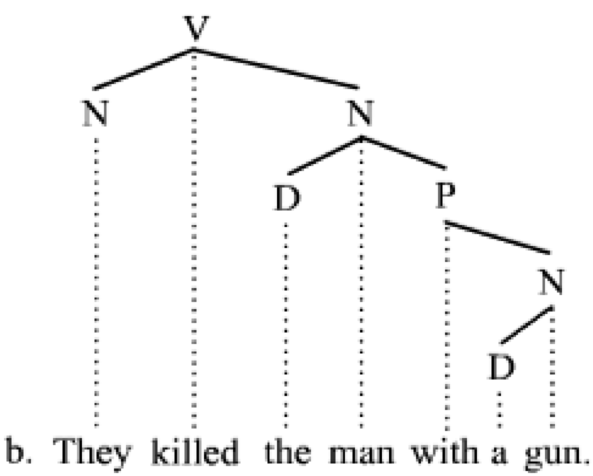
Constituency Parsing



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



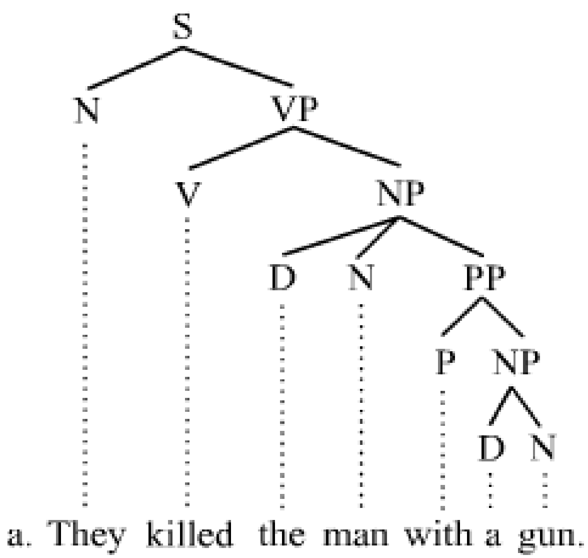
Dependency grammars

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

1 Linguistic Structure

Two main views of linguistic structure

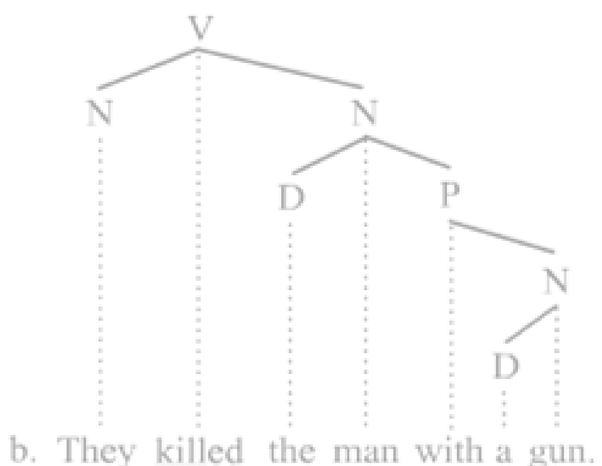
Constituency Parsing



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

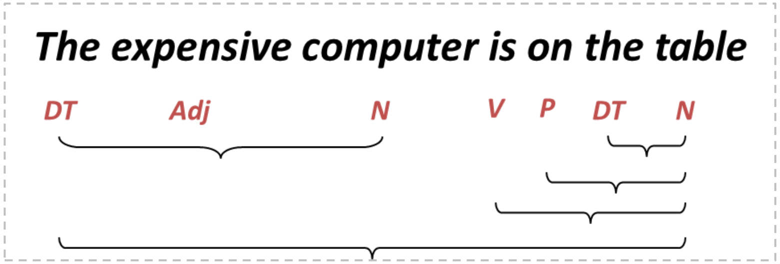
one-to-one relation; for every word in the sentence, there is exactly one node in the syntactic structure that corresponds to that word

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 similarity 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.*



1 Linguistic Structure

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*

1 Linguistic Structure

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

1 Linguistic Structure

A sample context-free grammar

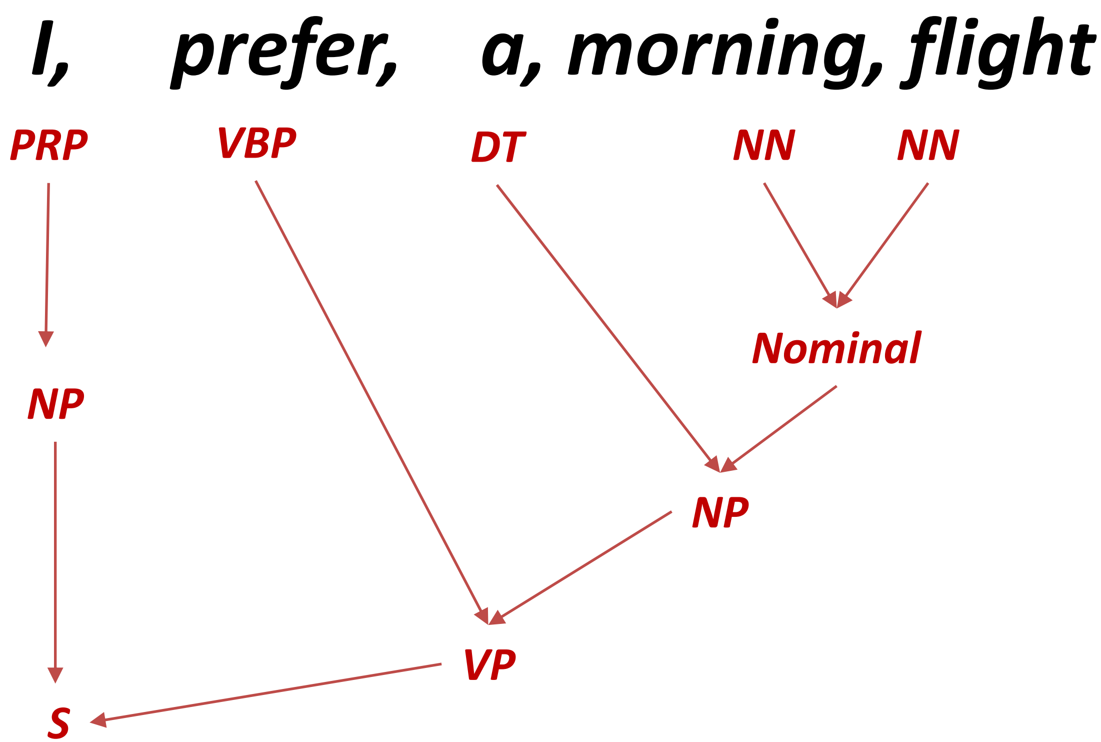
I, prefer, a, morning, flight

PRP ***VBP*** ***DT*** ***NN*** ***NN***

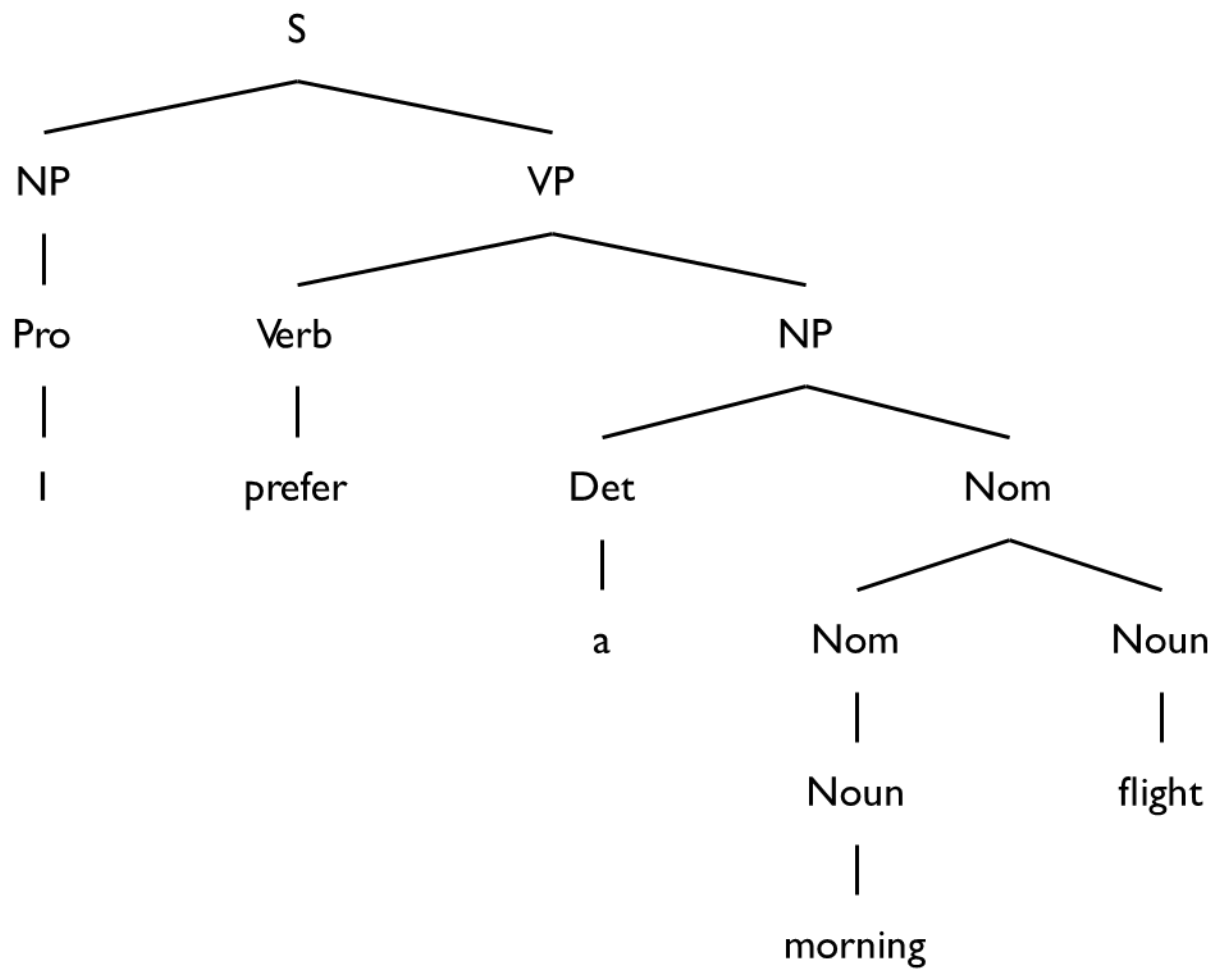
Grammar rule	Example
S → NPVP	I + want a morning flight
NP → Pronoun	I
NP → Proper-Noun	Sydney
NP → Det Nominal	a flight
Nominal → Nominal Noun	morning flight
Nominal → Noun	flights
VP → Verb	do
VP → Verb NP	want + a flight
VP → Verb NP PP	leave + Melbourne + in the morning
VP → Verb PP	leaving + on Thursday
PP → 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*

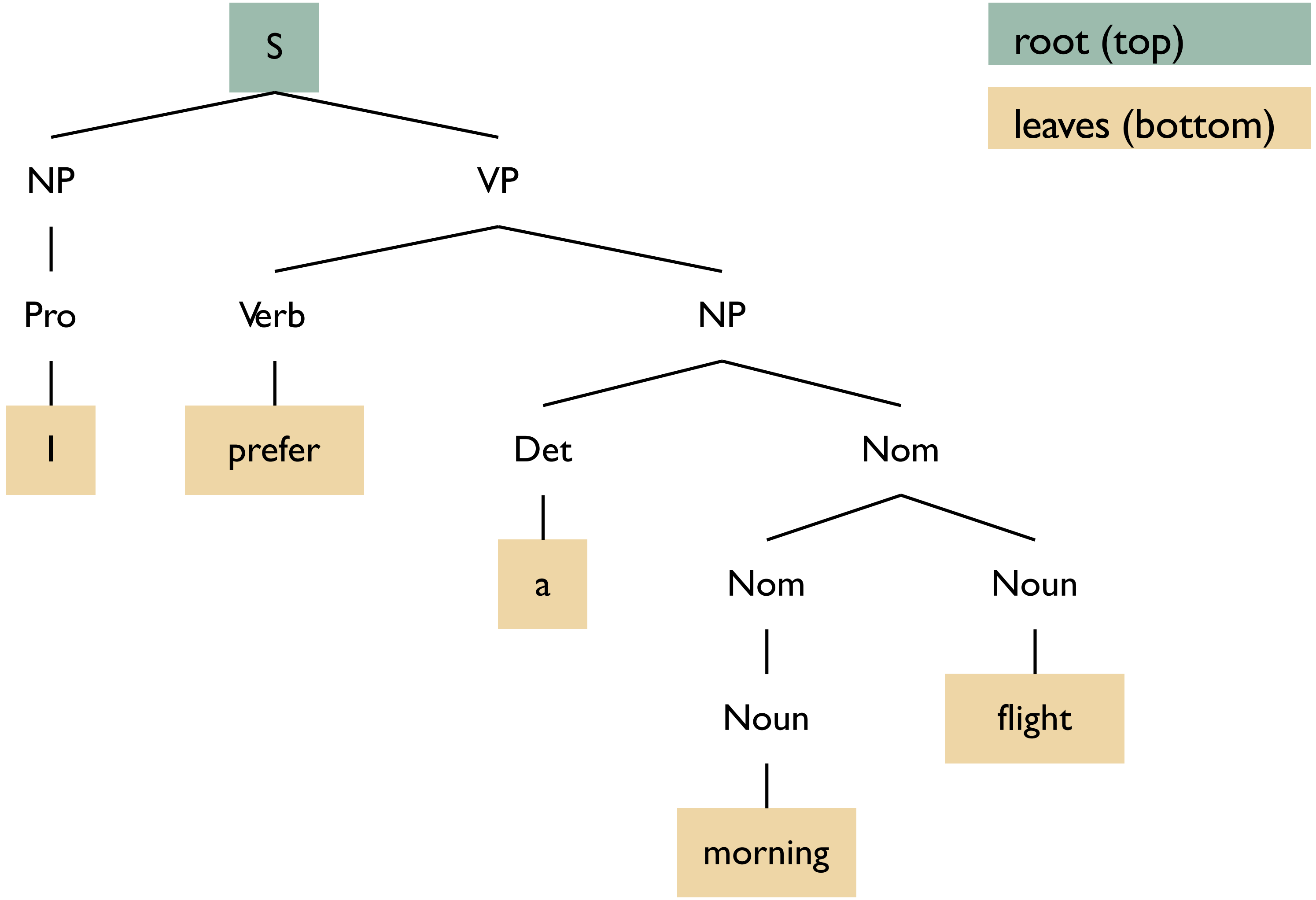


A sample context-free grammar



1 Linguistic Structure

A sample context-free grammar



1
Linguistic Structure

Treebanks

Corpora where each sentence is annotated with a parse tree

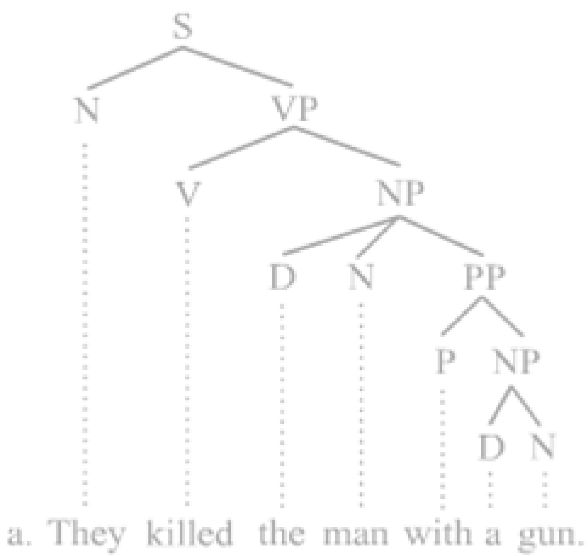
- *Treebanks are generally created by*
  - *parsing texts with an existing parser*
  - *having human annotators correct the result*
- *This requires detailed annotation guidelines for annotating different grammatical constructions*
- *Penn Treebank is a popular treebank for English (Wall Street Journal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

1 Linguistic Structure

Two main views of linguistic structure

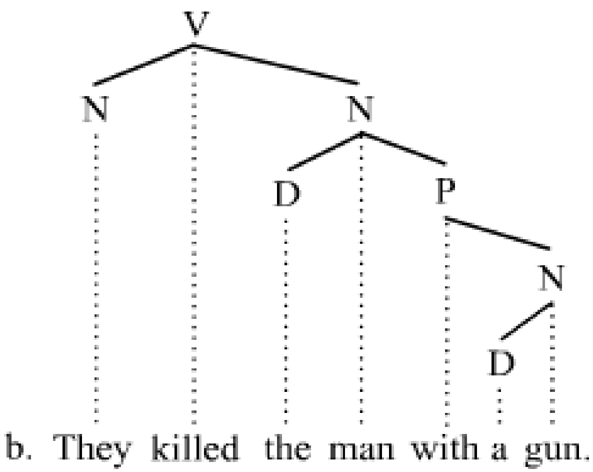
Constituency Parsing



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

0 LECTURE PLAN

Lecture 7: Parsing

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

2 Dependency Structure

Dependency Structure

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

dependencies

compound

Red **Hat**

modifier ***head***

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*

2 Dependency Structure

Dependency Parsing

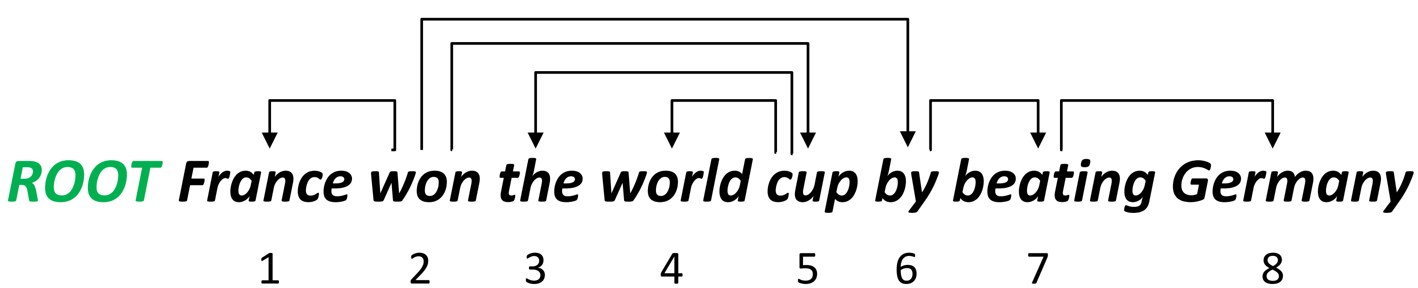
Represents Lexical/syntactic dependencies between words

- 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 → B, B → A



2 Dependency Structure

Dependency Grammar/Parsing History

Panini's grammar (4th century BCE)
The notion of dependencies between grammatical units

Ibn Maḍāʾ (12th century)
The first grammarian to use the term dependency in the grammatical sense

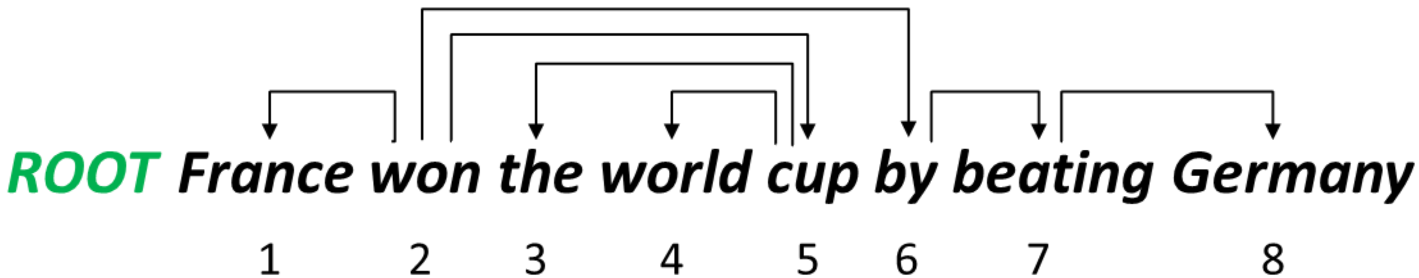
Sámuel Brassai, Franz Kern, Heimann Hariton Tiktin (1800 - 1930)
The dependency seems to have coexisted side by side with that of phrase structure

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

2 Dependency Structure

Dependency Grammar/Parsing



Some people draw the arrows one way; some the other way!

- *Tesnière had them point from head to dependent…*

Usually add a fake ROOT so every word is a dependent of precisely 1 other node

Projectivity vs Non-Projectivity

- There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words
- Dependencies parallel to a CFG tree must be projective
  - Forming dependencies by taking 1 child of each category as head
- But dependency theory normally does allow non-projective structures to account for displaced constituents

2 Dependency Structure

Dependency Relations

- *The following list shows the 37 universal **syntactic relations** used in Universal Dependencies v2.*

  - `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

***Universal Dependency Relations:** http://universaldependencies.org/u/dep/index.html*

Dependency Structure

Dependency Relations with annotations

- 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

2 Dependency Structure

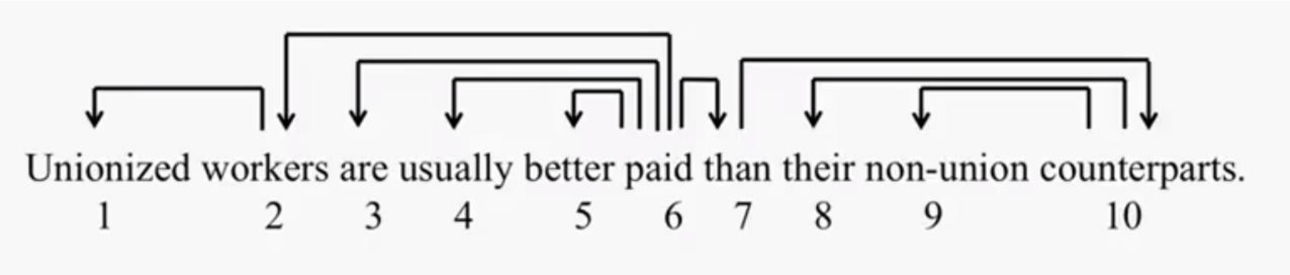
Dependency Parsing

Exercise – Let's do it together!

- Simpler to parse than context-free grammars

ROOT ***I prefer a morning flight***

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



LECTURE PLAN

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 Parsing Approaches

Methods of Dependency Parsing

- Dynamic programming
  Extension of the CYK algorithm to dependency parsing (Eisner, 1996)

- Constraint Satisfaction (Karlsson, 1990)
  word(pos(x)) = DET → (label(X)=NMOD, word(mod(x))=NN, pos(x) < mod(x))
  A determiner (DET) modifies a noun (NN) on the right with the label NMOD.

- ***Graph-based Dependency Parsing***
  Create a ***Minimum Spanning Tree*** for a sentence
  McDonald et al.'s (2005) MSTParser scores dependencies independently using an Machine Learning classifier

- ***Transition-based Dependency Parsing (Nivre 2008)***
- ***Neural Dependency Parsing***

CYK: Cocke–Younger–Kasami algorithm

3 Dependency Parsing Approaches

Graph-based dependency parsers

MST Parser (McDonald et al., 2015)

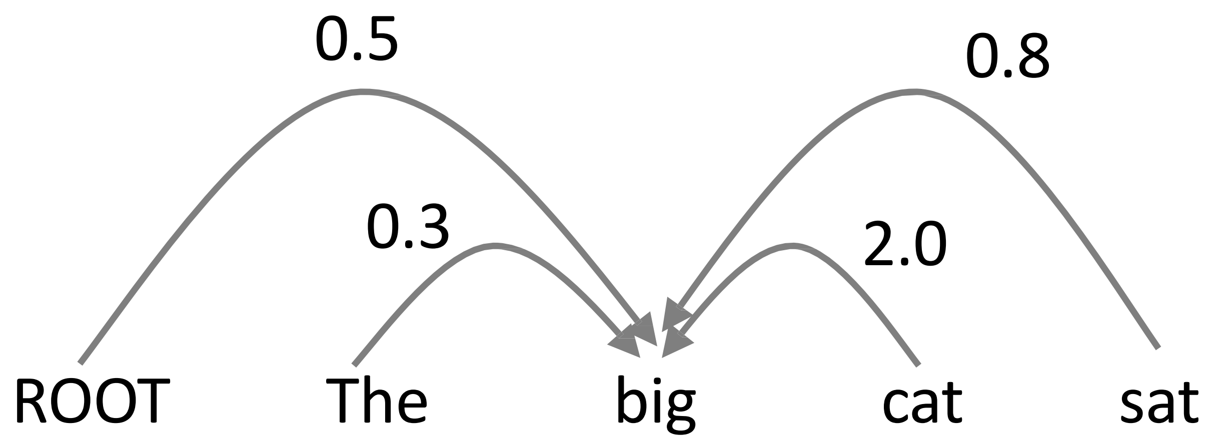
- **Projectivity**
  - English dependency trees are mostly projective (can be drawn without crossing dependencies)
  - Other languages are not

- **Idea**
  - Dependency parsing is equivalent to search for a maximum spanning tree in a directed graph
  - Chu and Liu (1965) and Edmonds (1967) give an efficient algorithm for finding MST for directed graphs

3 Dependency Parsing Approaches

Graph-based dependency parsers

- Compute a score for every possible dependency for each edge

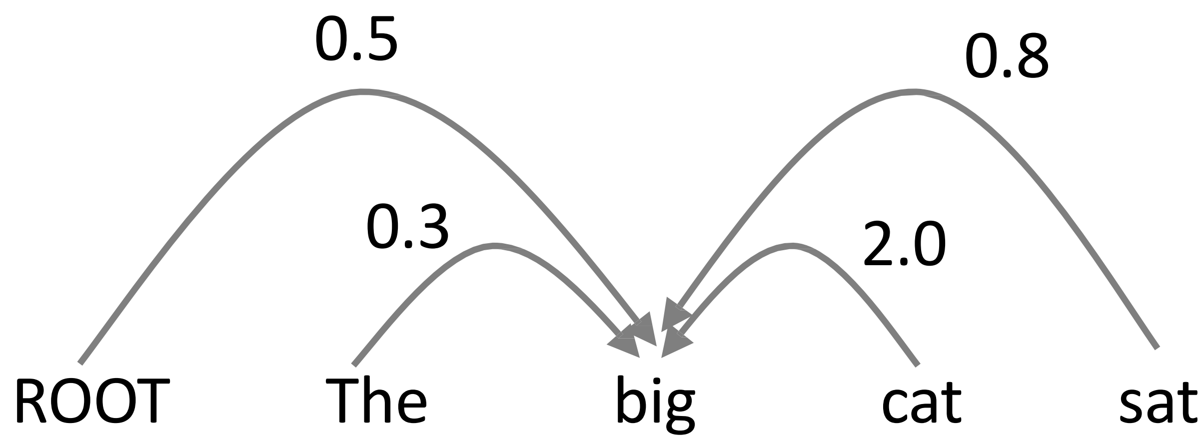


e.g., picking the head for "big"

Dependency Parsing Approaches

Graph-based dependency parsers

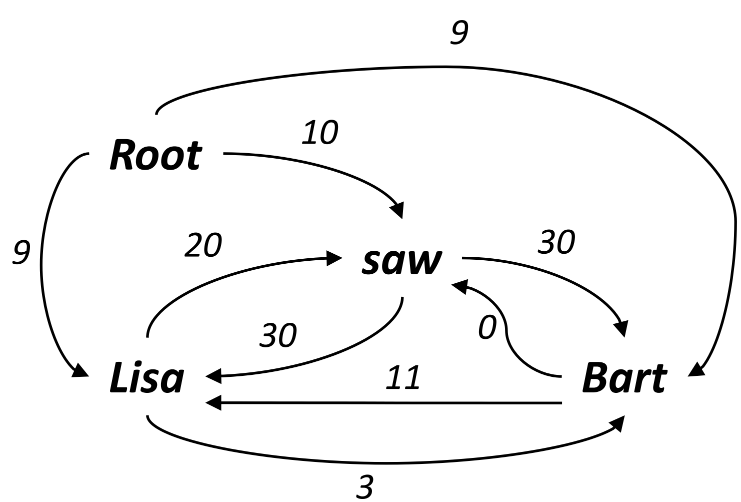
- Compute a score for every possible dependency for each edge
  - Then add an edge from each word to its highest-scoring candidate head
  - And repeat the same process for each other word

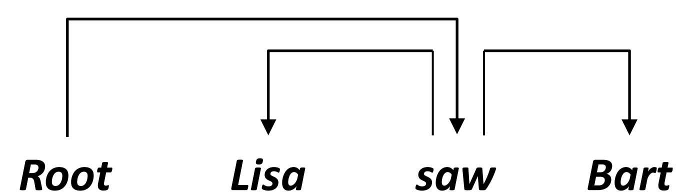


e.g., picking the head for “big”

Graph-based dependency parsers

- Consider the sentence “Lisa saw Bart”





3 Dependency Parsing Approaches

Methods of Dependency Parsing

greedy algorithm

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

Transition-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

0 LECTURE PLAN

Lecture 7: Parsing

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

4 Transition-based Parsing

Greedy transition-based parsing (Nivre 2008)

- A simple form of greedy discriminative dependency parser

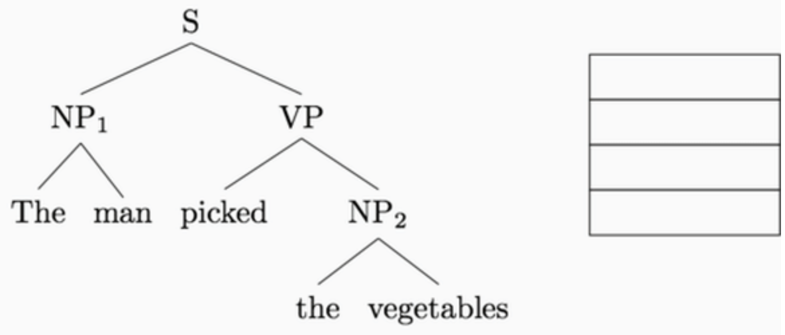
- **Transition**: an operation that searches for a dependency relation between each pair of words (e.g. Left-Arc, Shift, etc.)
- Design a dumber but really fast algorithm and let the machine learning(deep learning) do the rest.

- 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

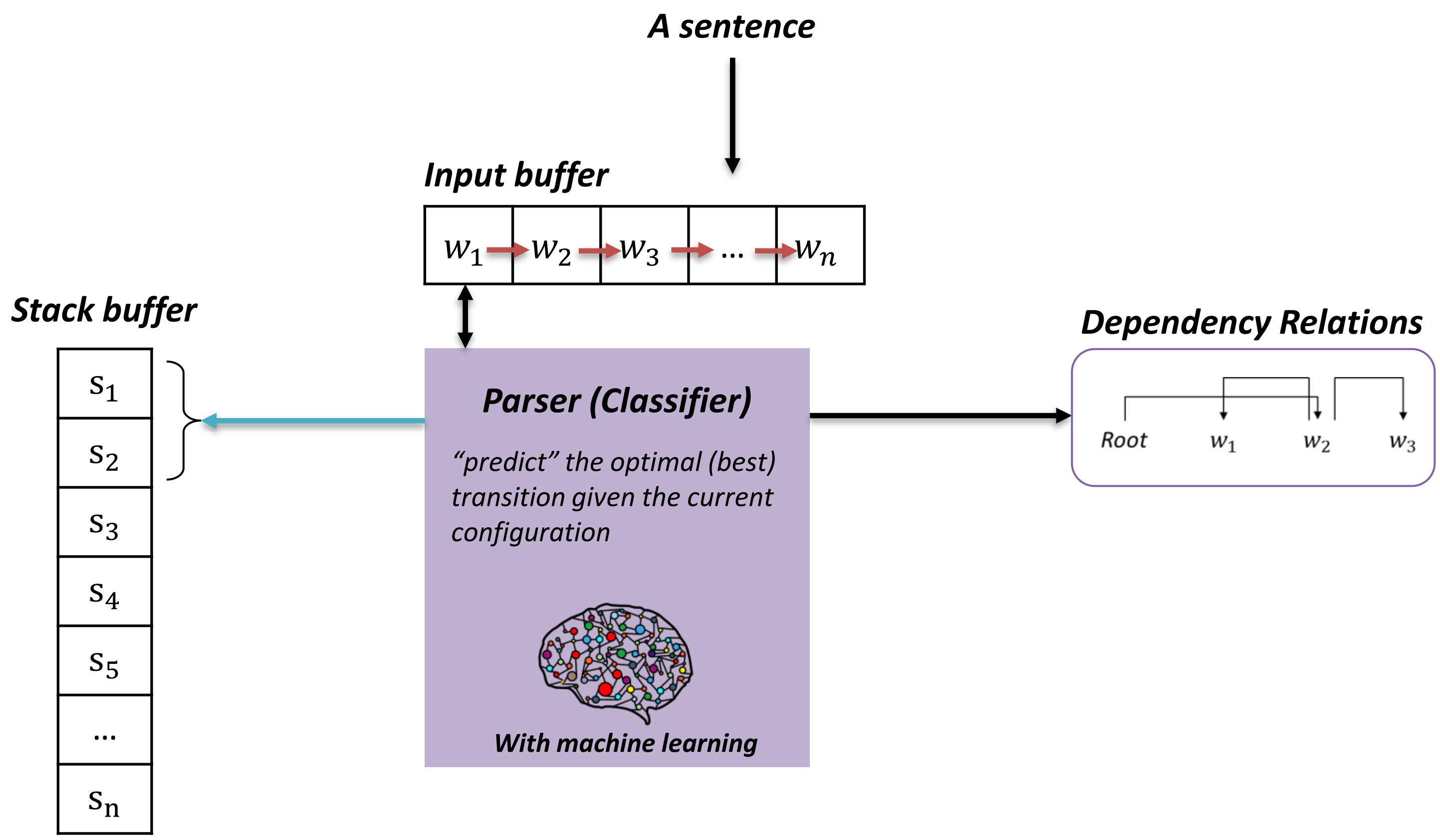
Transition-based parsing – The arc-standard algorithm

- 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**. The arc-standard algorithm is a simple algorithm for transition-based dependency parsing.

Transition-based parsing



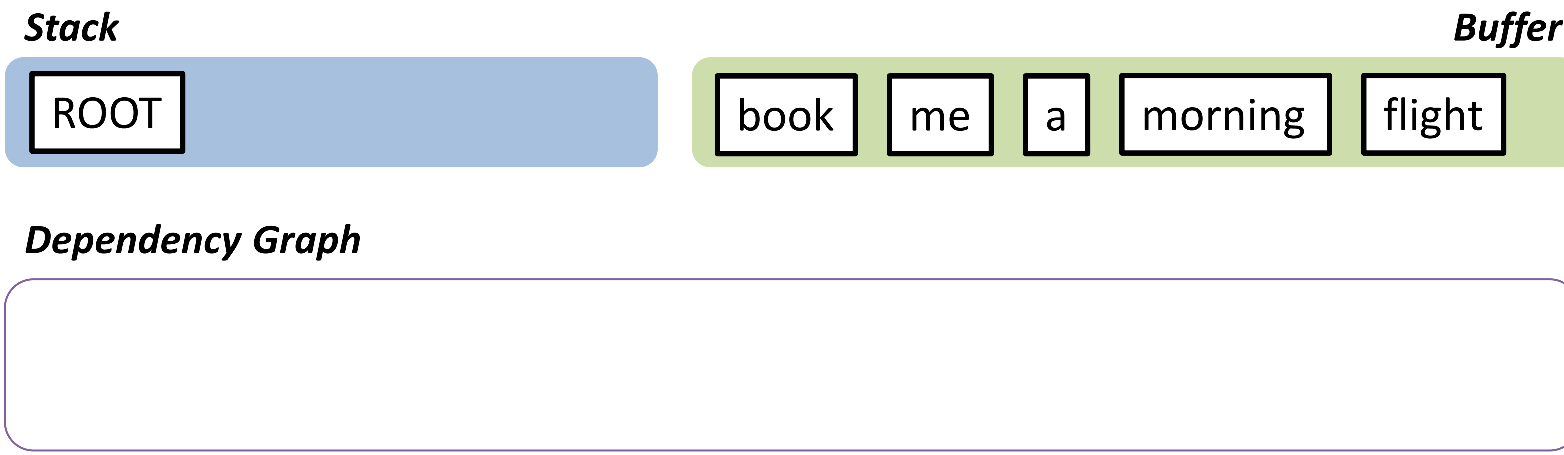
Transition-based parsing – The arc-standard algorithm

Stack

Buffer

Dependency Graph

Transition-based parsing – The arc-standard algorithm



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

4 Transition-based Parsing

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-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

ROOT cannot have incoming arc

Stack

ROOT

Buffer

book | me | a | morning | flight

Dependency Graph

Left Arc and Right Arc require 2 elements in stack to be applied

Possible Transaction

Shift

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

Left-Arc

- *Add an arc* ***from the topmost word to the 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

Stack

ROOT | book

Buffer

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

Stack

ROOT | book | me

Buffer

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 2nd-topmost word** on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc **from the 2nd-topmost word to the topmost word** on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

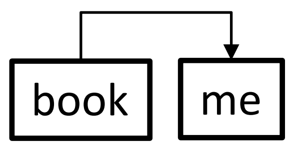
Stack

ROOT | book | me

Buffer

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

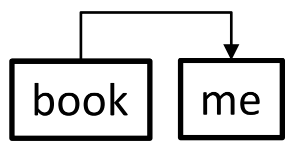
Stack

ROOT | book | a

Buffer

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

Stack

ROOT | book | a | morning

Buffer

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

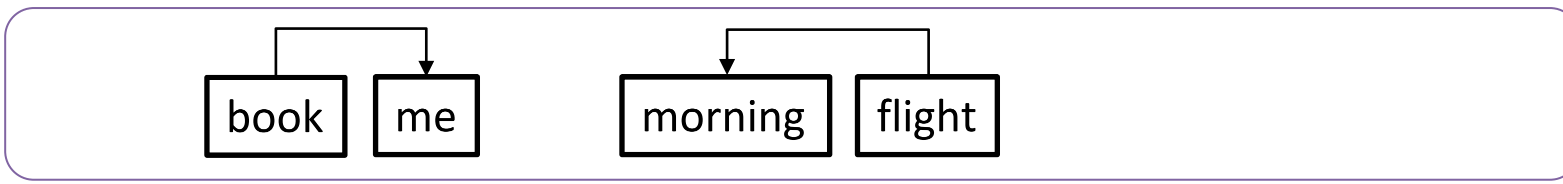
Transition-based parsing – The arc-standard algorithm

Stack

ROOT | book | a | morning | flight

Buffer

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 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

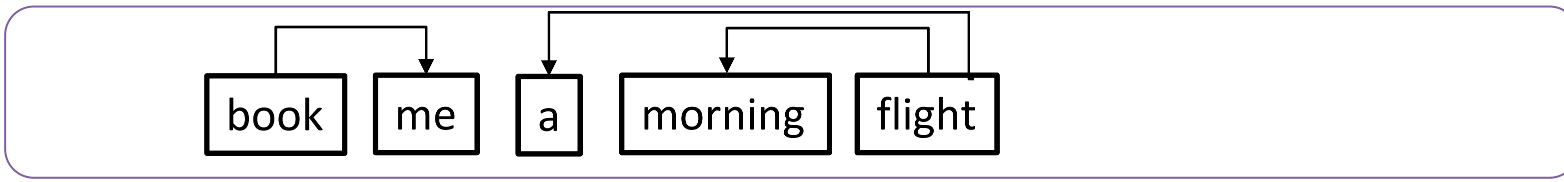
Transition-based parsing – The arc-standard algorithm

Stack

ROOT | book | a | flight

Buffer

Dependency Graph



Possible Transaction

Shift

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

Left-Arc

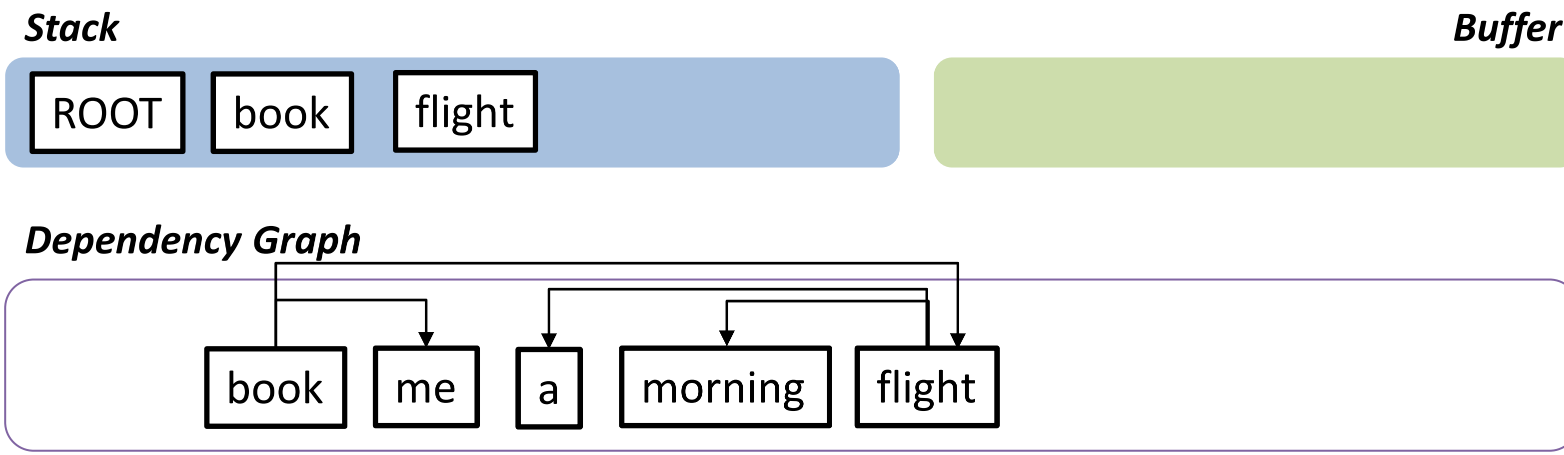
- *Add an arc **from the topmost word to the 2nd-topmost word** on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc **from the 2nd-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



Possible Transaction

Shift

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

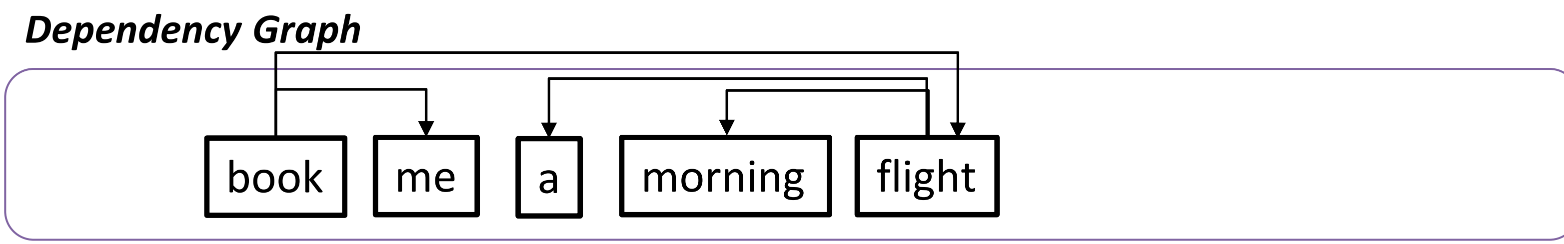
Left-Arc

- *Add an arc* ***from the topmost word to the 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm



Possible Transaction

Shift

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

Left-Arc

- *Add an arc* ***from the topmost word to the 2nd-topmost word*** *on the stack*
- *Remove 2nd word from stack*

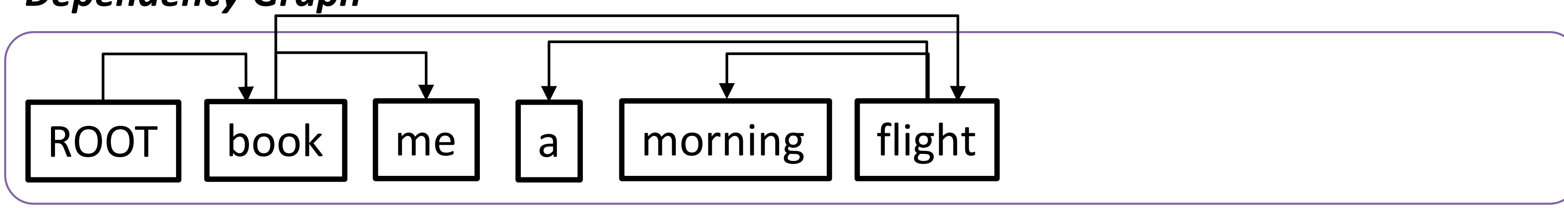
Right-Arc

- *Add an arc* ***from the 2nd-topmost word to the topmost word*** *on the stack*
- *Remove the topmost word from stack*

Transition-based parsing – The arc-standard algorithm

Stack ***Buffer***

Dependency Graph



Possible Transaction

Shift

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

Left-Arc

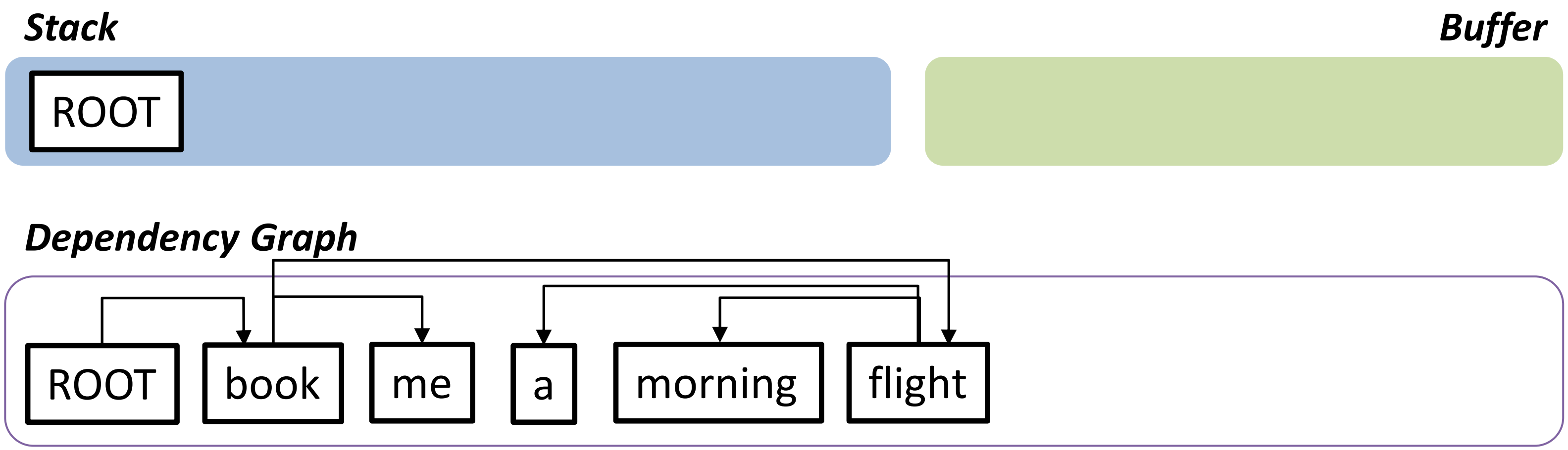
- *Add an arc **from the topmost word to the 2nd-topmost word** on the stack*
- *Remove 2nd word from stack*

Right-Arc

- *Add an arc **from the 2nd-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



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

Transition-based parsing – The arc-standard algorithm

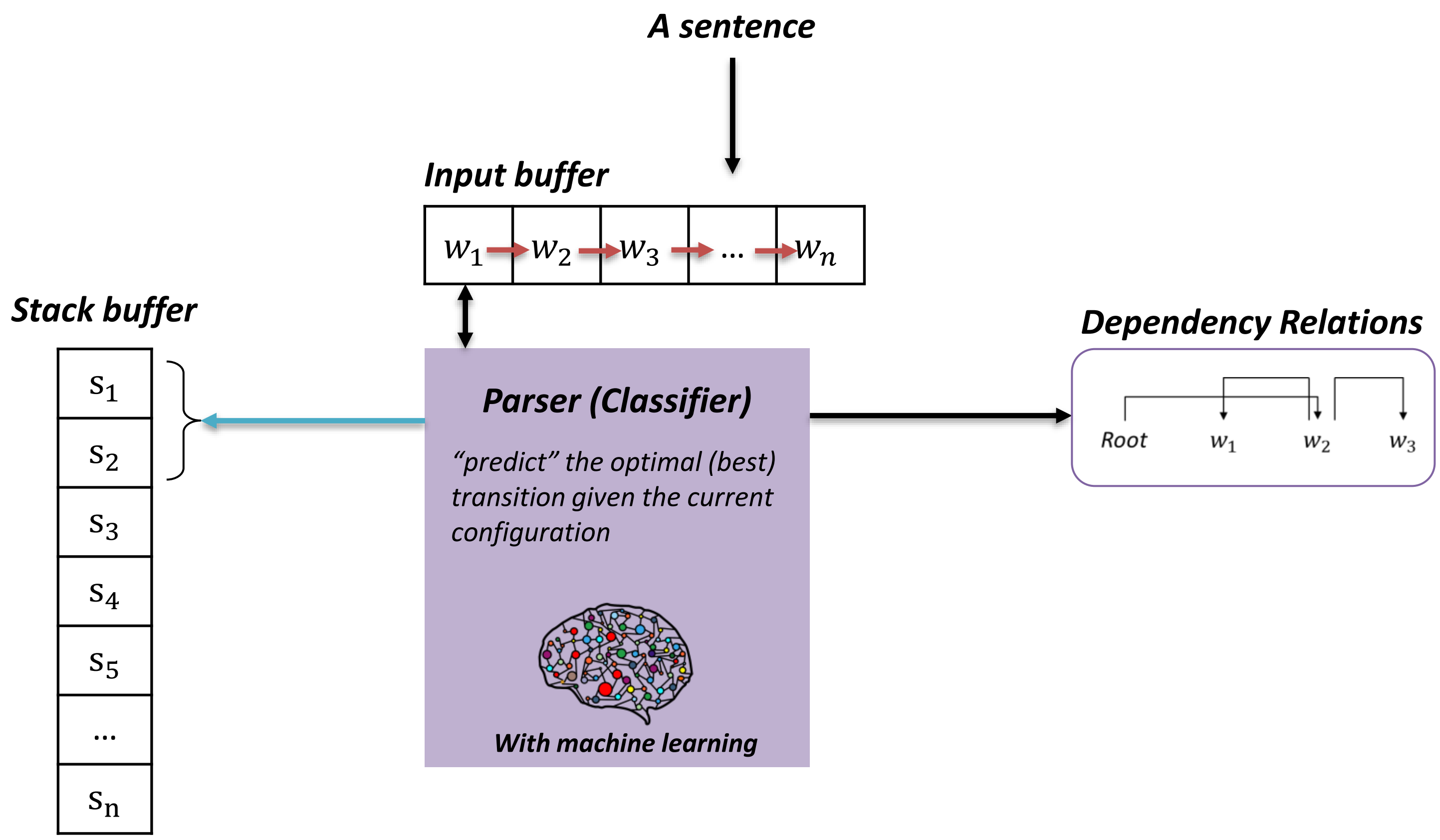
Start: σ = [ROOT], $\beta = w_1, \ldots, w_n$, A = Ø

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

Finish: $\sigma = [w]$, β = Ø

How to choose the next action?

Transition-based parsing



4 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?

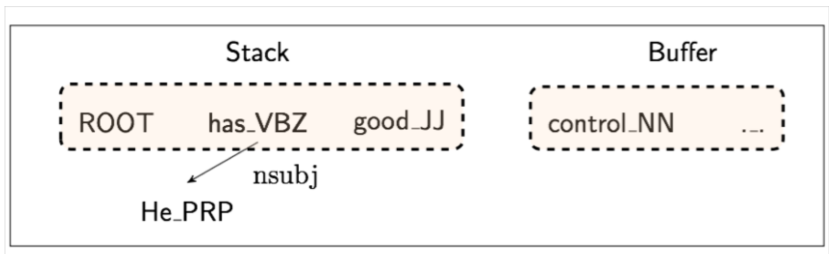
Transition-based Parsing

Feature Representation

- Define a set of features of configurations that you consider to be relevant for the task of predicting the next transition.

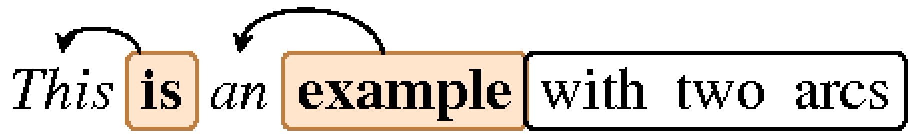
Example: word forms of the topmost two words on the stack and the next two words in the buffer

- Describe every configuration in terms of a feature vector.

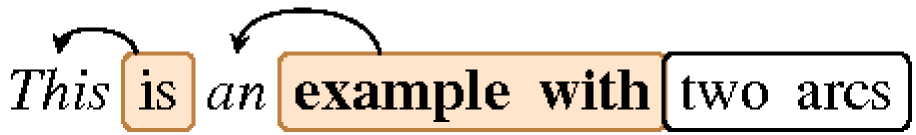


- In practical systems, we have thousands of features and hundreds of transitions.
- There are several machine-learning paradigms that can be used to train a guide for such a task
- Examples: perceptron, decision trees, support-vector machines, memory-based learning

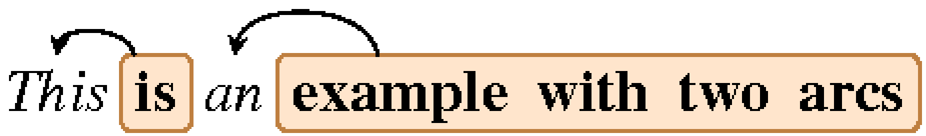
Transition-based parsing



(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).

4 Transition-based Parsing

Evaluation of Dependency Parsing

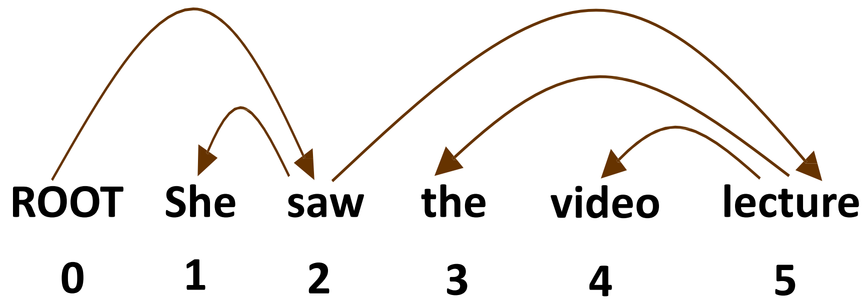
$$Accuracy = \frac{\#\ correct\ deps}{\#\ of\ deps}$$

Unlabeled attachment score (UAS) = head

Labeled attachment score (LAS) = head and label

4 Transition-based Parsing

Evaluation of Dependency Parsing



Gold Standard

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

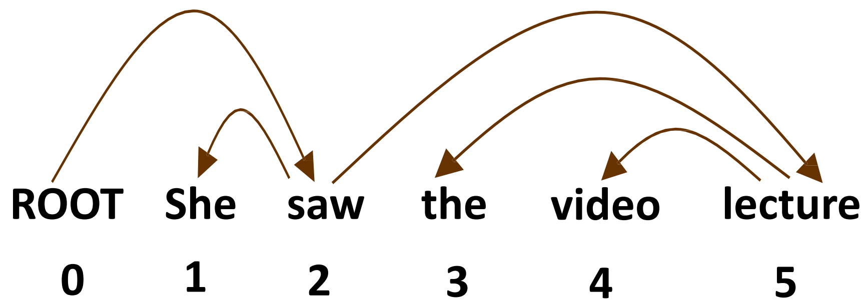
Parsed (assume this is what you classified)

1	2	she	nsubj
2	0	Saw	root
3	4	The	det
4	5	Video	nsubj
5	2	lecture	ccomp

4 Transition-based Parsing

THE UNIVERSITY OF SYDNEY

Evaluation of Dependency Parsing



Gold Standard

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

Parsed (assume this is what you classified)

1	2	she	nsubj
2	0	Saw	root
3	4	The	det
4	5	Video	nsubj
5	2	lecture	ccomp

Unlabeled attachment score (UAS) = 4 / 5= 80%

Labeled attachment score (LAS) = 2 / 5= 40%

0 LECTURE PLAN

Lecture 7: Parsing

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

5 Deep Learning-based Parsing

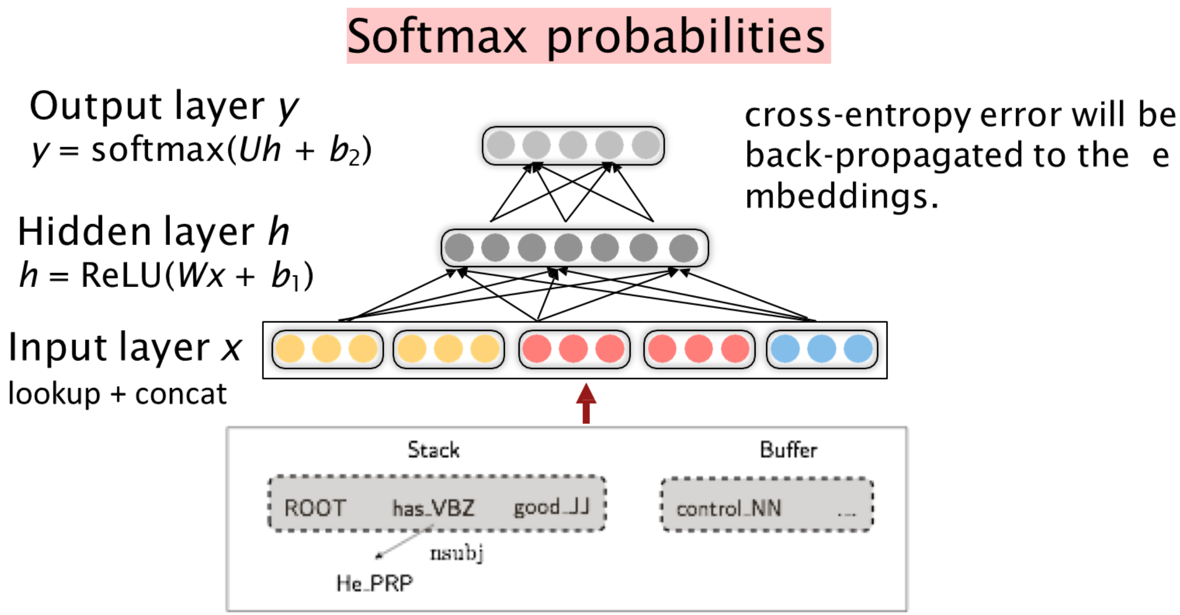
Distributed Representations

- 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 noguon).

- 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

5 Deep Learning-based Parsing

Neural Dependency Parsing (Chen & Manning, 2014)



Neural Dependency Parsing

Accuracy and parsing speed on PTB + Stanford dependencies.

Parser	Dev UAS	Dev LAS	Test UAS	Test LAS	Speed (sent/s)
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	**92.0**	**89.7**	**91.8**	**89.6**	**654**

Accuracy and parsing speed on CTB

Parser	Dev UAS	Dev LAS	Test UAS	Test LAS	Speed (sent/s)
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	**84.0**	82.1	83.0	81.2	6
Our parser	**84.0**	**82.4**	**83.9**	**82.4**	**936**

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*

5 Deep Learning-based Parsing

Dependency Parsing Trends – Penn Treebank

RANK	METHOD	UAS	LAS	POS	PAPER TITLE	YEAR
1	CVT + Multi-Task	96.61	95.02	---	Semi-Supervised Sequence Modeling with Cross-View Training	2018
2	Stack-Pointer Network	95.87	94.19	---	Stack-Pointer Networks for Dependency Parsing	2018
3	Deep Biaffine	95.44	93.76	97.3	Deep Biaffine Attention for Neural Dependency Parsing	2016
4	Andor et al.	94.61	92.79	97.44	Globally Normalized Transition-Based Neural Networks	2016
5	jPTDP	94.51	92.87	97.97	An improved neural network model for joint POS tagging and dependency parsing	2018
6	Distilled neural FOG	94.26	92.06	97.3	Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser	2016
7	Weiss et al.	93.99	92.05	97.44	Structured Training for Neural Network Transition-Based Parsing	2015
8	BIST transition-based parser	93.9	91.9	97.3	Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations	2016
9	Arc-hybrid	93.56	91.42	97.3	Training with Exploration Improves a Greedy Stack-LSTM Parser	2016
10	BIST graph-based parser	93.1	91.0	97.3	Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations	2016

https://paperswithcode.com/sota/dependency-parsing-on-penn-treebank

5 Deep Learning-based Parsing

Semi-Supervised Sequence Modeling with Cross-View Training (Clark, 2018)

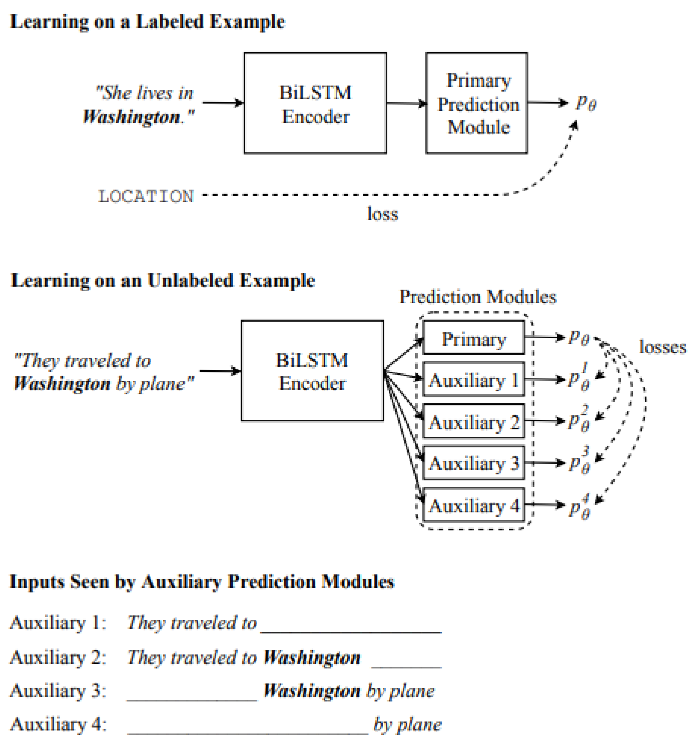


Figure 1: An overview of Cross-View Training. The model is trained with standard supervised learning on labeled examples. On unlabeled examples, auxiliary prediction modules with different views of the input

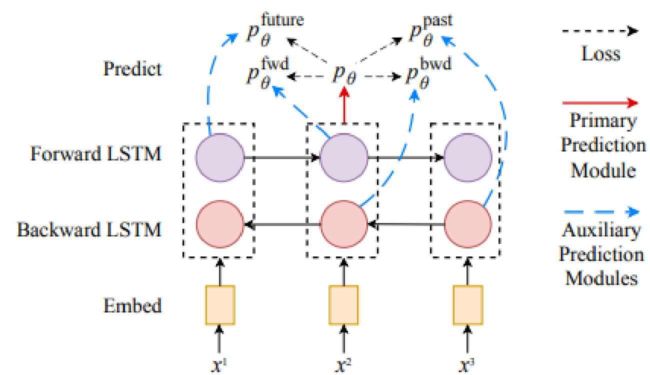


Figure 2: Auxiliary prediction modules for sequence tagging models. Each one sees a restricted view of the input. For example, the "forward" prediction module does not see any context to the right of the current token when predicting that token's label. For simplicity, we only show a one layer Bi-LSTM encoder and only show the model's predictions for a single time step.

https://arxiv.org/pdf/1809.08370v1.pdf

Stack-Pointer Network (Ma, 2018)

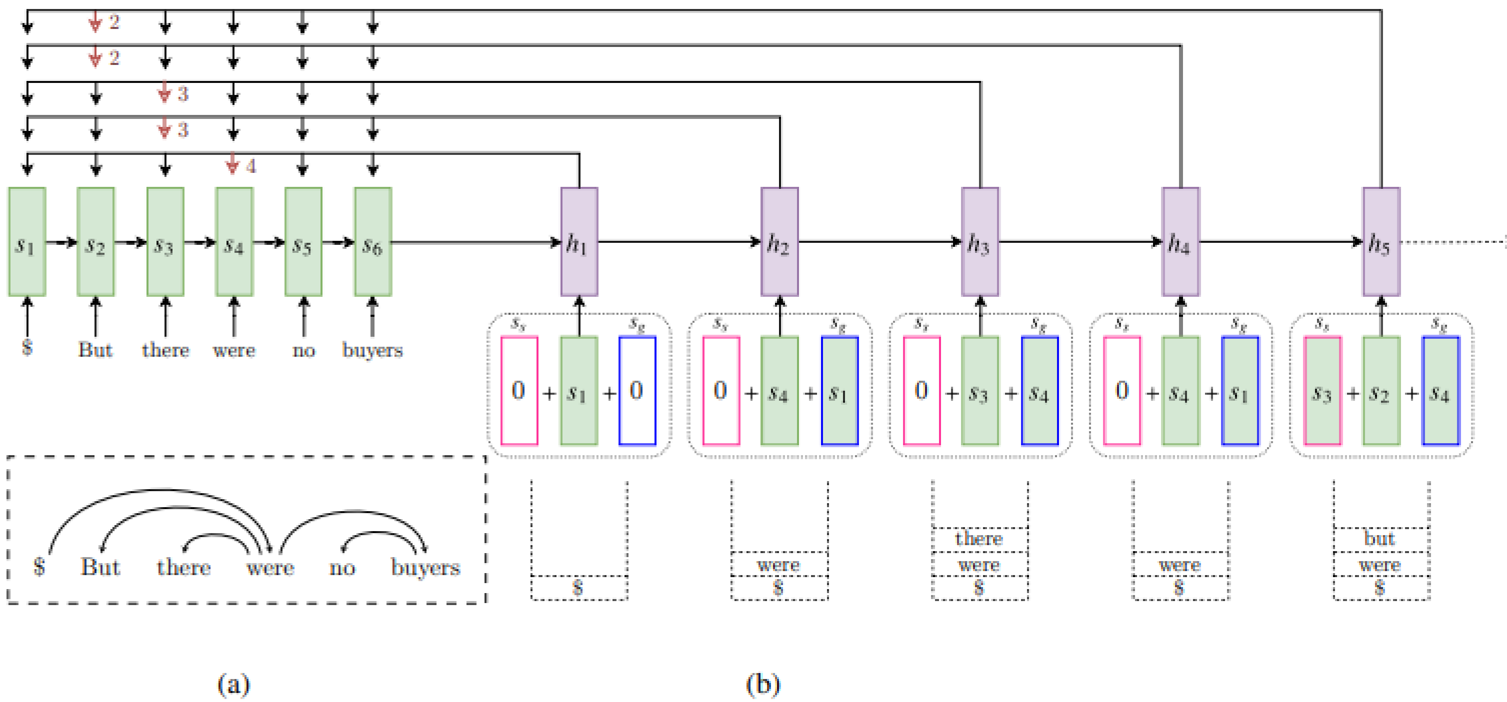


Figure 1: Neural architecture for the STACKPTR network, together with the decoding procedure of an example sentence. The BiRNN of the encoder is elided for brevity. For the inputs of decoder at each time step, vectors in red and blue boxes indicate the sibling and grandparent.

https://arxiv.org/pdf/1805.01087v1.pdf

5 Deep Learning-based Parsing

Dependency Parsing Trends – Penn Treebank

RANK	METHOD	UAS	LAS	POS	PAPER TITLE	YEAR
1	CVT + Multi-Task	96.61	95.02	---	Semi-Supervised Sequence Modeling with Cross-View Training	2018
2	Stack-Pointer Network	95.87	94.19	---	Stack-Pointer Networks for Dependency Parsing	2018
3	Deep Biaffine	95.44	93.76	97.3	Deep Biaffine Attention for Neural Dependency Parsing	2016
4	Andor et al.	94.61	92.79	97.44	Globally Normalized Transition-Based Neural Networks	2016
5	jPTDP	94.51	92.87	97.97	An improved neural network model for joint POS tagging and dependency parsing	2018
6	Distilled neural FOG	94.26	92.06	97.3	Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser	2016
7	Weiss et al.	93.99	92.05	97.44	Structured Training for Neural Network Transition-Based Parsing	2015
8	BIST transition-based parser	93.9	91.9	97.3	Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations	2016
9	Arc-hybrid	93.56	91.42	97.3	Training with Exploration Improves a Greedy Stack-LSTM Parser	2016
10	BIST graph-based parser	93.1	91.0	97.3	Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations	2016

https://paperswithcode.com/sota/dependency-parsing-on-penn-treebank

/ Final Exercise

VB PRP

Thank you!

VBP DT JJ NN

Have a great day!

/ Reference

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.