

Lojban

COMP 4446 / 5046 Natural Language Processing

Lecture 7: Dependency Parsing



[zo'o ta jitfa .i .e'o xu do
pendo mi]

/

[That was a joke. Really.
Wanna be friends with me?]

Dr. Jonathan Kay Kummerfeld

Semester 1, 2023

School of Computer Science,
University of Sydney

[menti.com](https://www.menti.com/join/77864905) 7786 4905

Source: <https://xkcd.com/191/>

0

Announcements

- Virtual tutorials
- Feedback survey on Canvas
- Assignment 1 marks out - complete the form if you think there is an error
- Assignment 2 being finalised
- Office hours more variable in April

Course topics

What will you learn in this course?

Week 1: Introduction to Natural Language Processing (NLP)

Week 2: Word Embeddings

Week 3: Word Classification with Machine Learning I

Week 4: Word Classification with Machine Learning II

*NLP and
Machine
Learning*

Week 5: Fundamentals of Language

Week 6: Part of Speech Tagging

Week 7: Dependency Parsing

Week 8: Language Models

Week 9: [ANZAC Day]

Week 10: Named Entity Recognition and Coreference Resolution

Week 11: Question Answering

Week 12: Machine Translation

*NLP
Tasks*

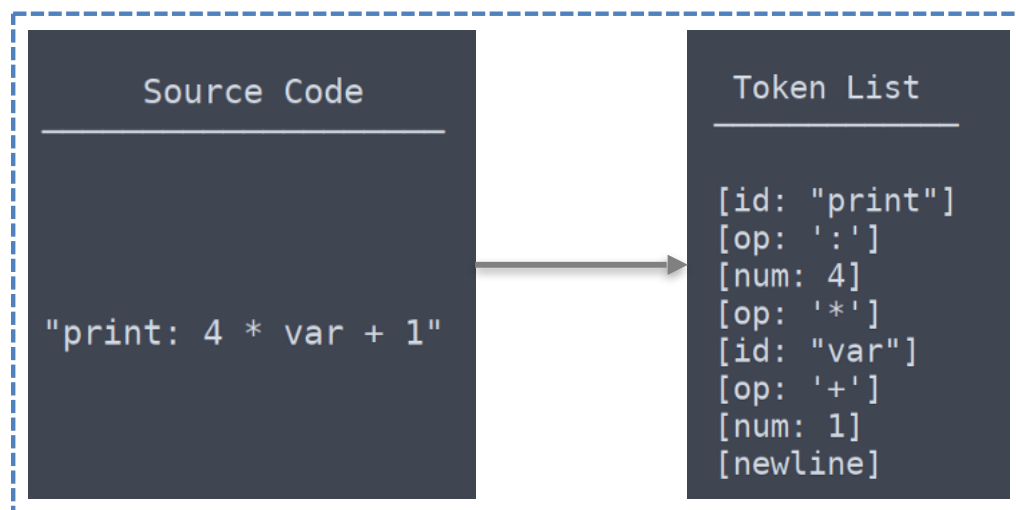
Week 13: Future of NLP and Exam Review

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

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!

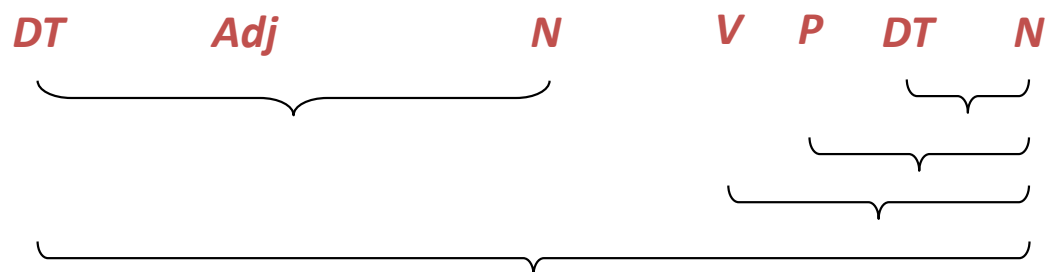
Why?

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

1 Linguistic Structure

Natural Language: Linguistic Structure

The expensive computer is on the table



However, Language is **more than just a “bag of words”**.

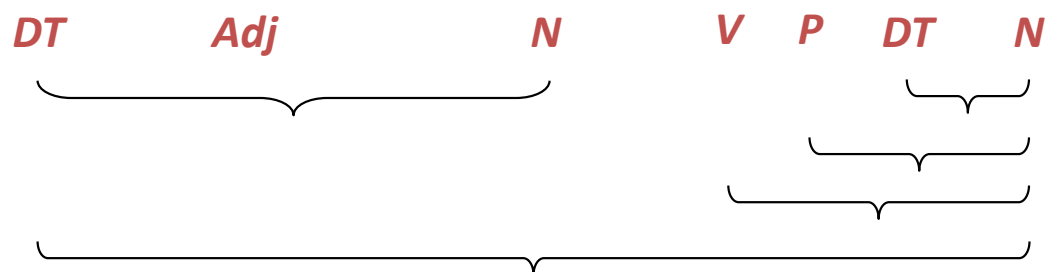
Grammatical rules apply to combine:

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

1 Linguistic Structure

Natural Language: Linguistic Structure

The expensive computer is on the table



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*.

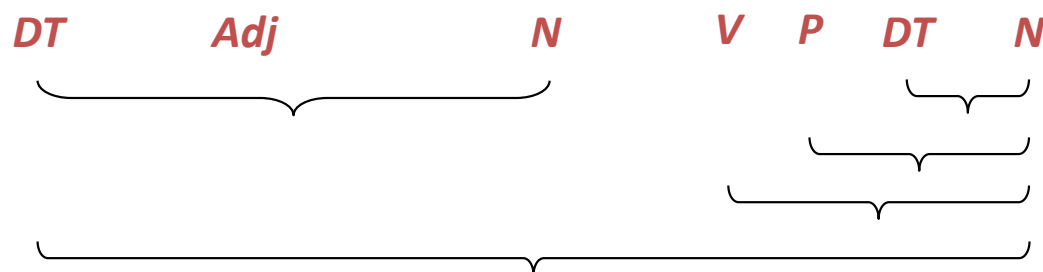
The *subject* is noun phrase and the *predicate* is a verb phrases.

1 Linguistic Structure

Natural Language: Linguistic Structure

Phrase Structure Grammar = Context-free Grammar (CFG)

The expensive computer is on the table



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*.

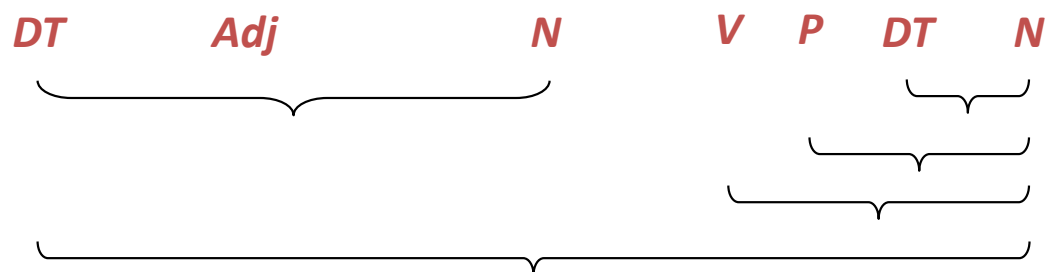
The *subject* is noun phrase and the *predicate* is a verb phrases.

1 Linguistic Structure

Natural Language: Linguistic Structure

Phrase Structure Grammar = Context-free Grammar (CFG)

The expensive computer is on the table



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

1 Linguistic Structure

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!**

1 Linguistic Structure

Syntactic Ambiguities

Grammars are declarative

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

Ambiguity

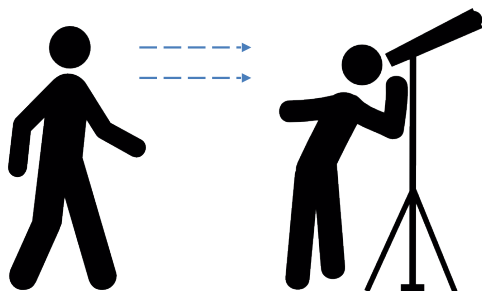
1. *Prepositional Phrase (PP) attachment ambiguity*
2. *Coordination Scope*
3. *Gaps*
4. *Particles or Prepositions*
5. *Gerund or adjective*
6. *More!*

1 Linguistic Structure

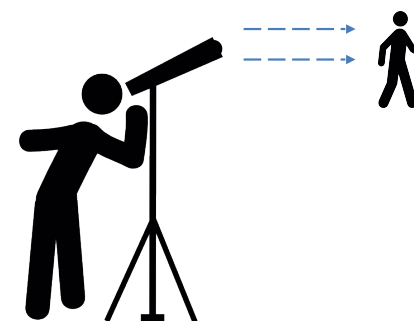
Syntactic Ambiguities – PP attachment Ambiguity

I saw the man with the telescope

I saw the man with the telescope



I saw the man with the telescope



1 Linguistic Structure

Syntactic Ambiguities – Coordination Scope Ambiguity

I ate red apples and bananas



1 Linguistic Structure

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 – Particles or Prepositions

Classification of Verb-Particle Constructions with the Google Web1T Corpus

Jonathan K. Kummerfeld and **James R. Curran**

School of Information Technologies

University of Sydney

NSW 2006, Australia

{jkum0593, james}@it.usyd.edu.au

1 Linguistic Structure

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*

Linguistic Structure

Two main views of linguistic structure

Constituency Grammar (a.k.a., *phrase structure grammar*)

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

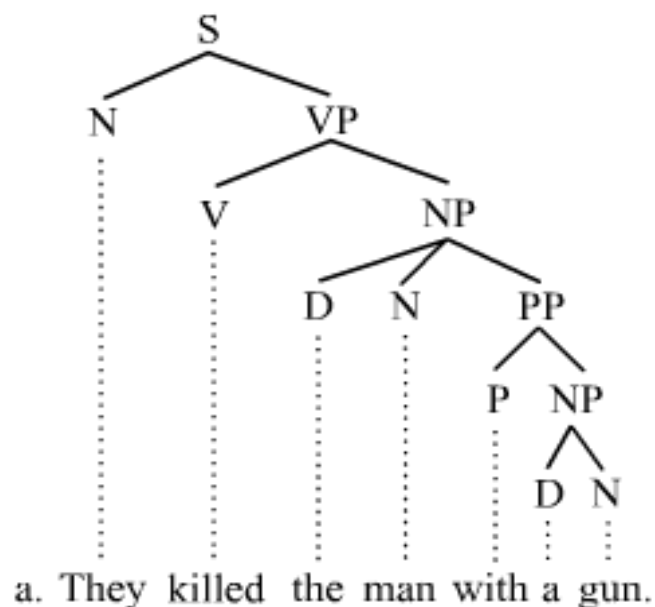
Dependency Grammar

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

1 Linguistic Structure

Two main views of linguistic structure

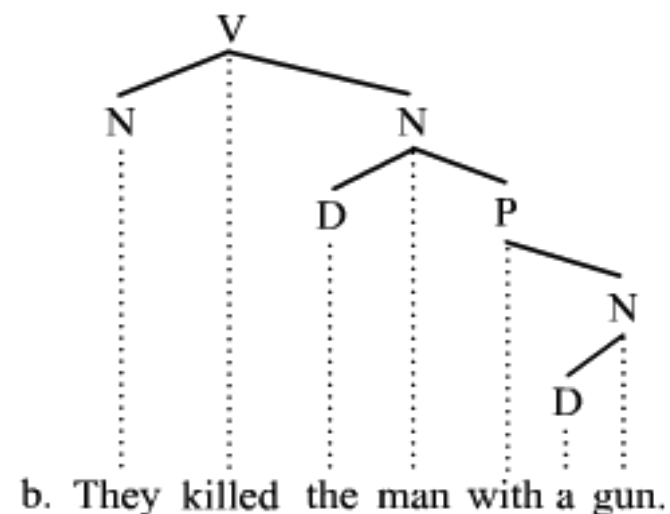
Constituency Parsing



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



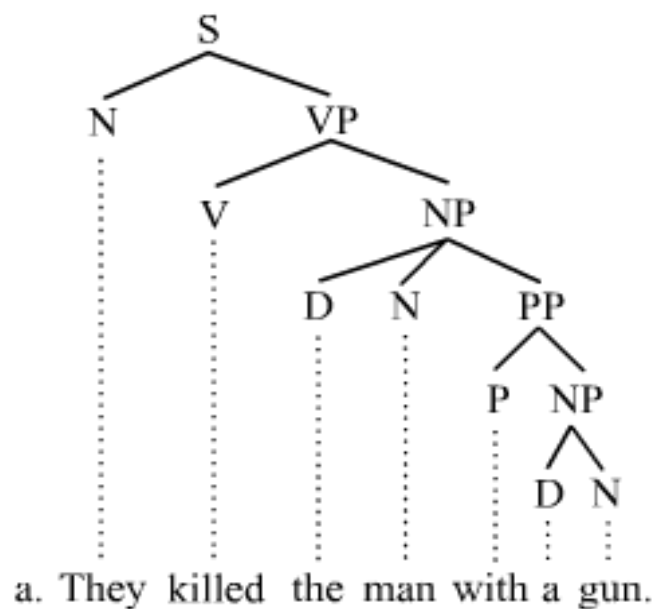
Dependency grammars

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

1 Linguistic Structure

Two main views of linguistic structure

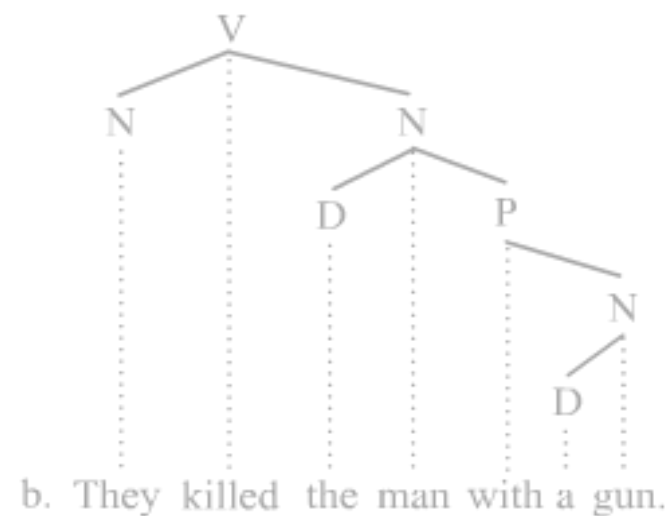
Constituency Parsing



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

1 Linguistic Structure

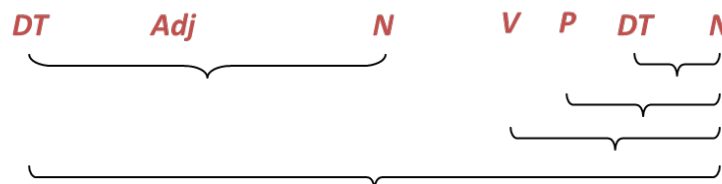
Constituency Grammar

- 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.

The expensive computer is on the table



A sample constituency 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 constituency 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 constituency grammar

I, prefer, a, morning, flight

PRP VBP DT NN NN

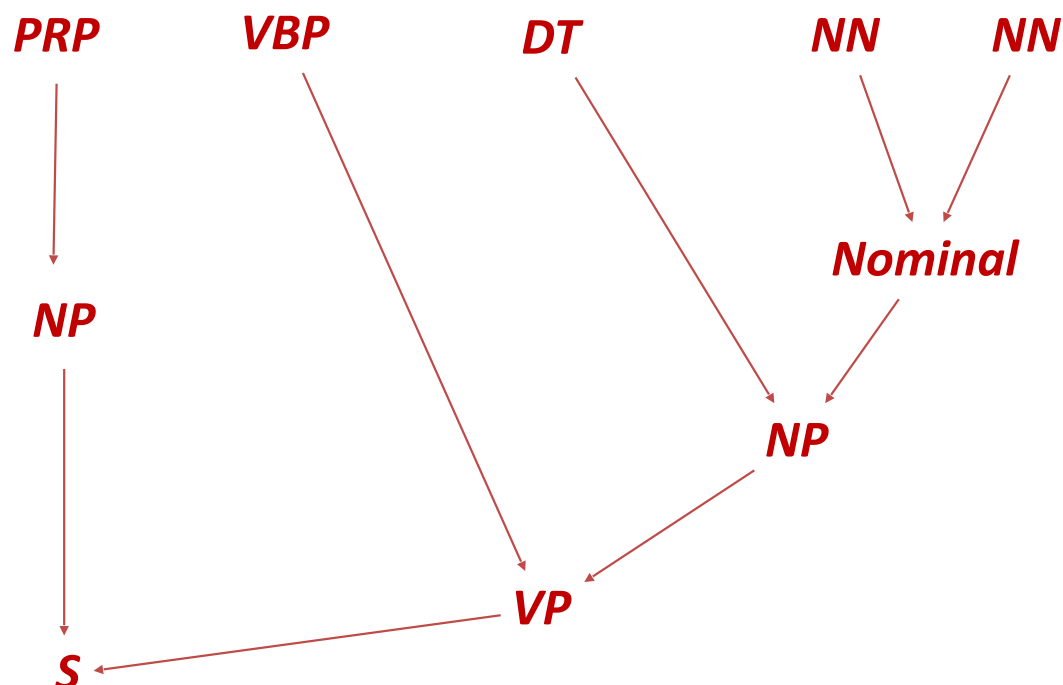
Grammar rule	Example
$S \rightarrow NPVP$	I + want a morning flight
$NP \rightarrow \text{Pronoun}$	I
$NP \rightarrow \text{Proper-Noun}$	Sydney
$NP \rightarrow \text{Det Nominal}$	a flight
$\text{Nominal} \rightarrow \text{Nominal Noun}$	morning flight
$\text{Nominal} \rightarrow \text{Noun}$	flights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

1 Linguistic Structure

A sample constituency grammar

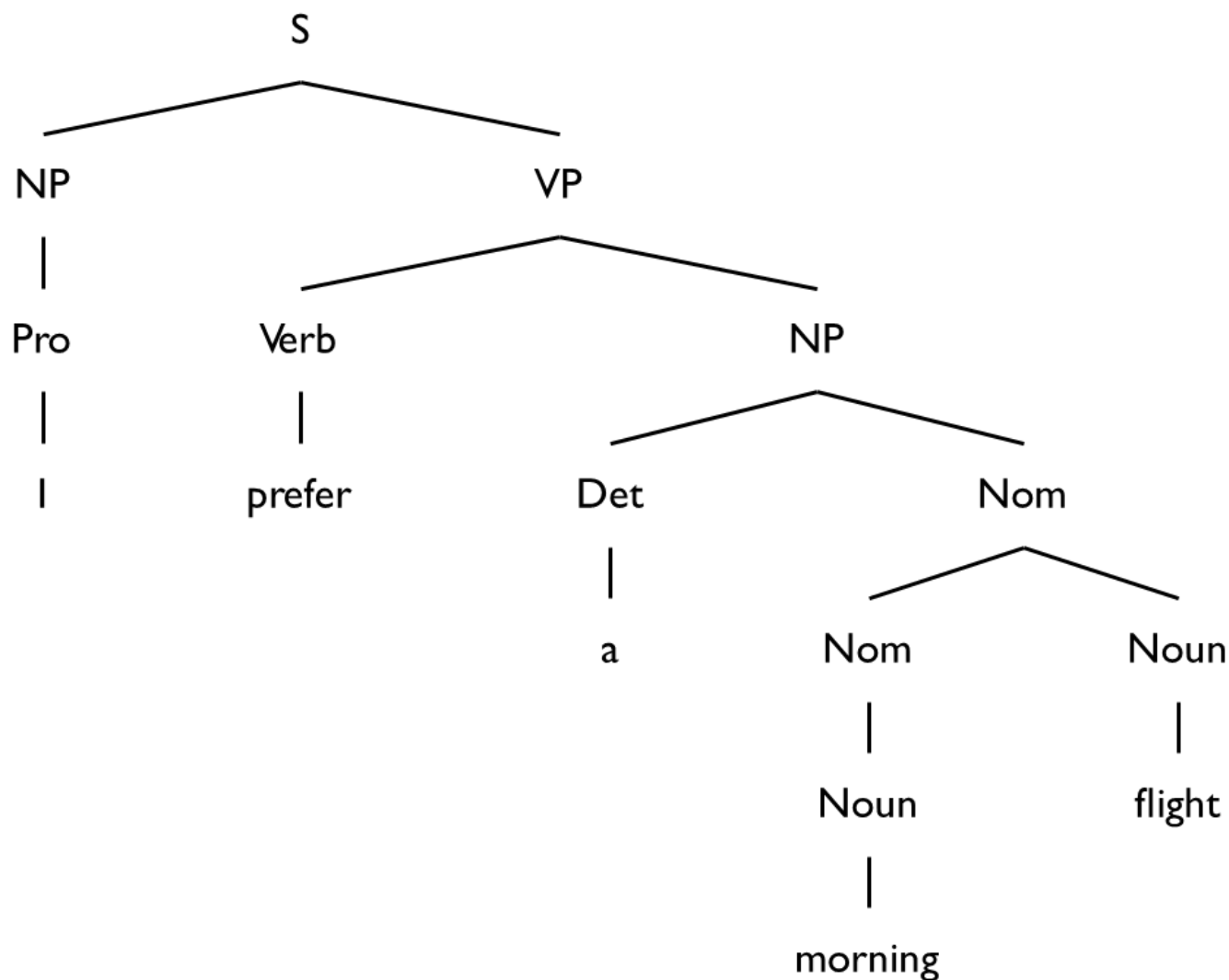
1. Starting unit: words are given a category (part-of-speech)
2. Combining words into **phrases with categories**
3. Combining phrases into **bigger phrases** recursively

I, prefer, a, morning, flight



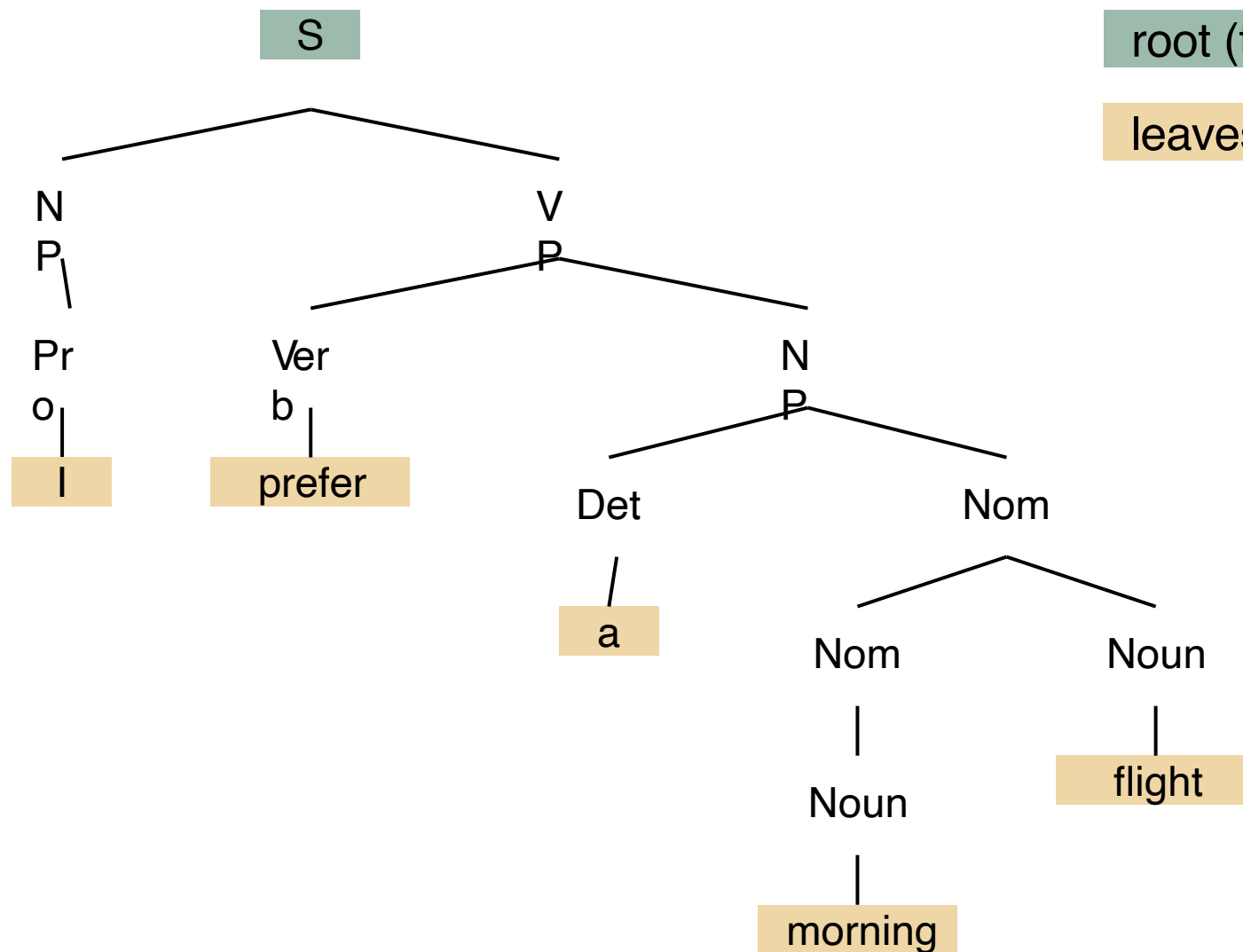
1 Linguistic Structure

A sample constituency grammar



1 Linguistic Structure

A sample constituency grammar



root (top)

leaves (bottom)

1 Linguistic Structure

Treebanks

Corpora where each sentence is annotated with a parse tree

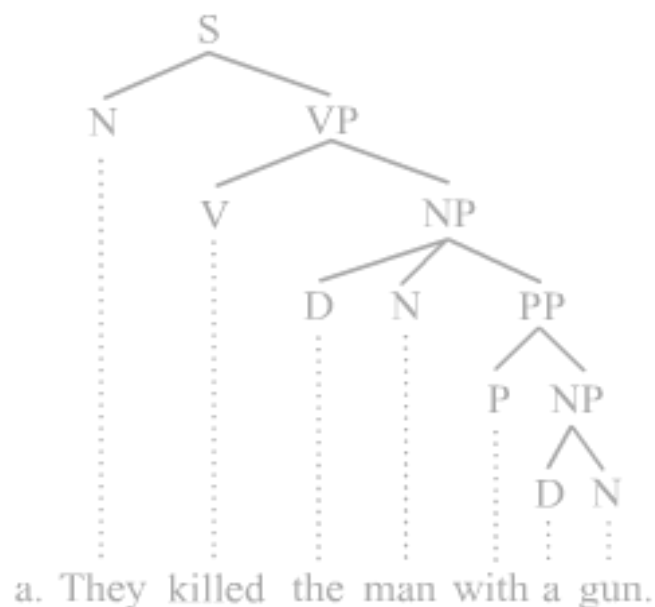
- *Treebanks are generally created by*
 - *parsing texts with an existing parser*
 - *having human annotators correct the result*
- *This requires detailed annotation guidelines for annotating different grammatical constructions*
- *The Penn Treebank is a popular treebank for English*

```
( (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

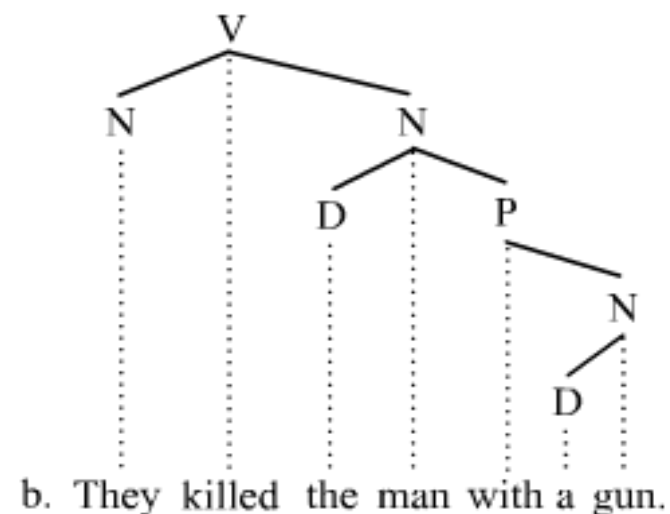
Constituency Parsing



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

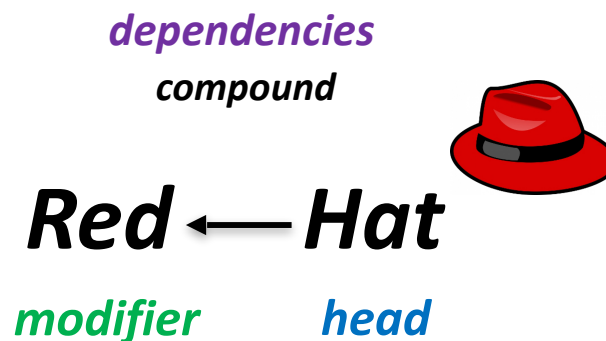
Lecture 7: Parsing

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



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

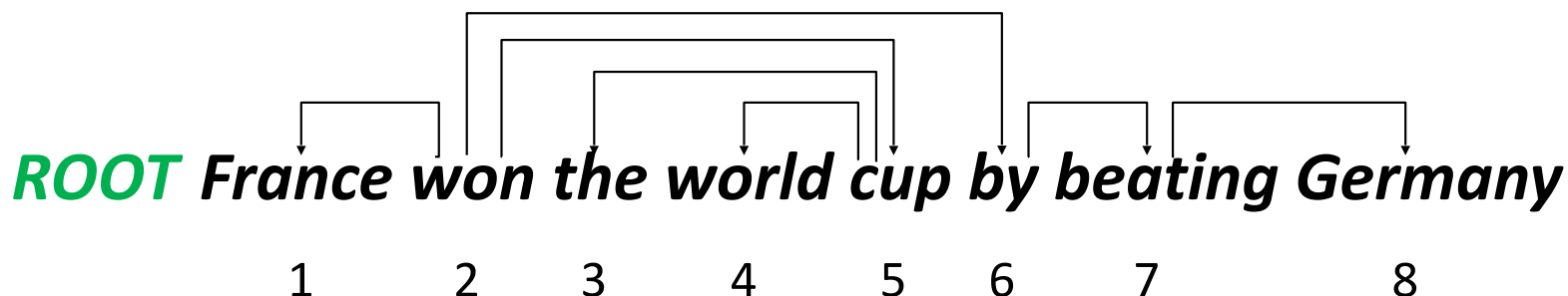
- A sentence is parsed by choosing for each word what other word (including ROOT) it is a dependent of

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 $A \rightarrow B, B \rightarrow A$



2 Dependency Structure

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, ...)

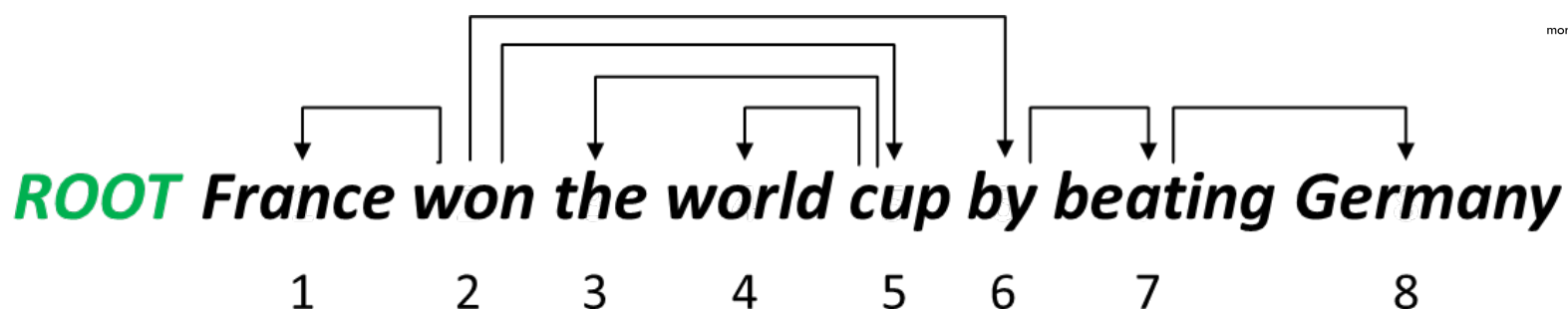
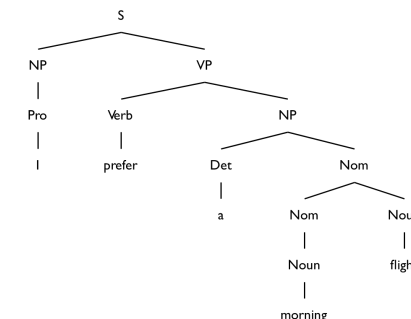
Good for free-er word order languages

David Hays (1962)

Great development surrounding dependency-based theories in 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
- But dependency theory normally does allow non-projective structures to account for displaced constituents

2 Dependency Structure

Dependency Grammar/Parsing

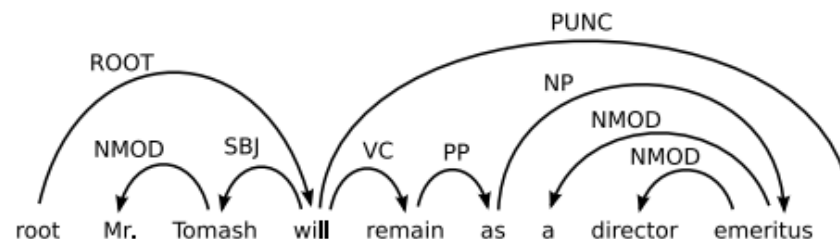


Figure 1: A projective dependency graph.

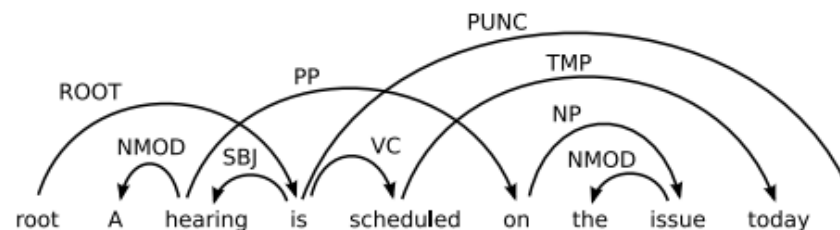


Figure 2: Non-projective dependency graph.

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
- 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

2 Dependency Structure

Dependency Relations with annotations

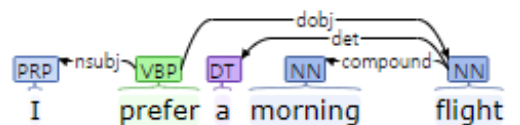
- The idea of dependency structure goes back a long way
- *Current Main effort - Universal Dependencies:*
<http://universaldependencies.org/> ;
- Starting off, building a treebank seems a lot slower and less useful than building a grammar
- But a treebank gives us many things
 - Reusability of the labor
 - 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

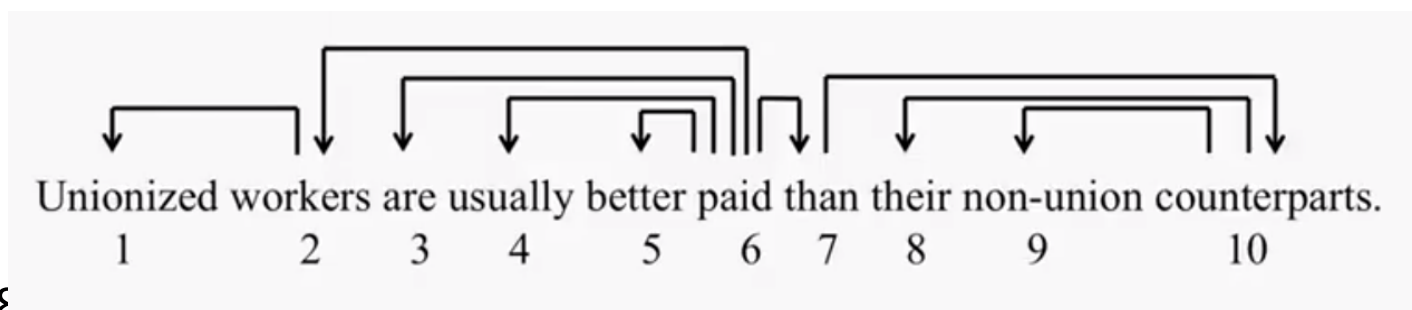
Dependency Parsing

Exercise – Let's do it together!

ROOT *I prefer a morning flight*

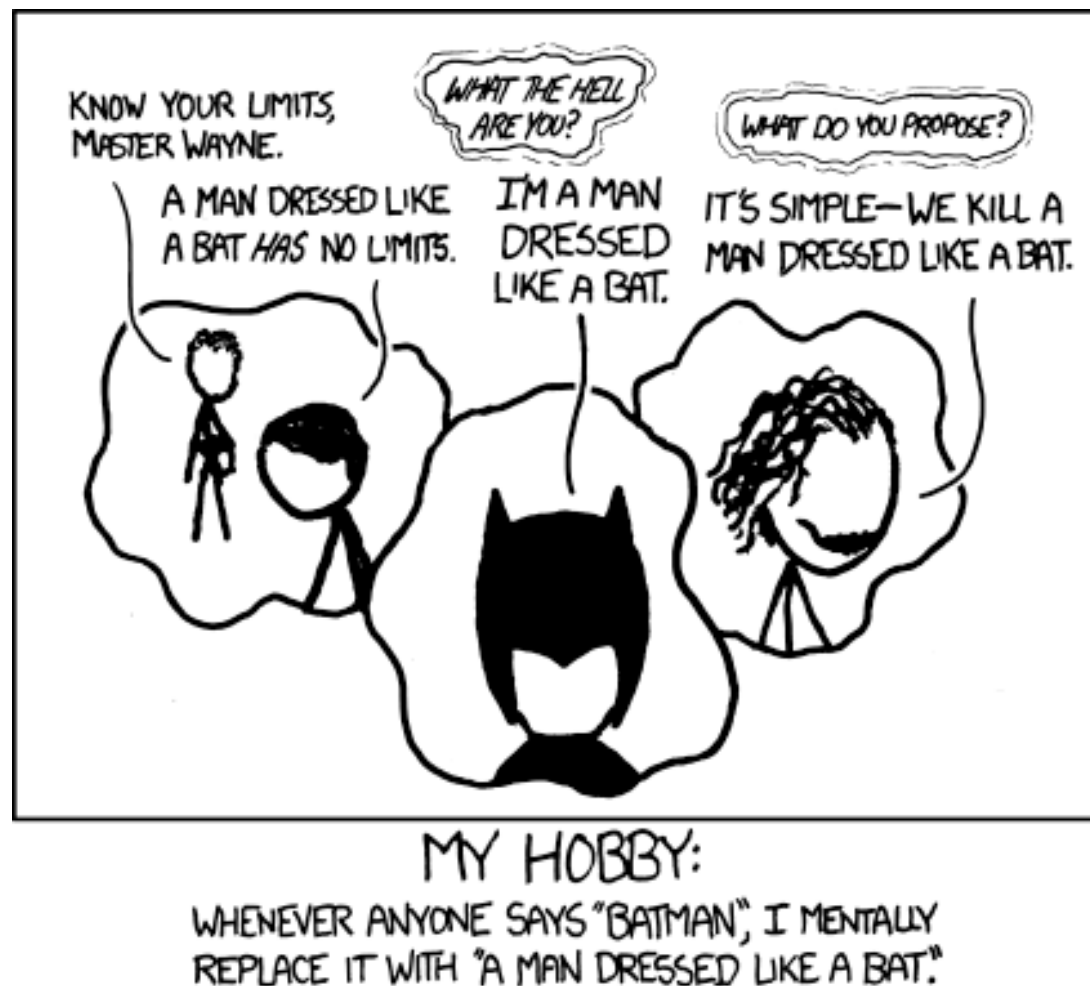


ROOT *Unionised workers are usually better paid than their non-union counterparts*



Break

Batman



[I'm really worried Christopher Nolan will kill a man dressed like a bat in his next movie. (The man will be dressed like a bat, I mean. Christopher Nolan won't be, probably.)]

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)
 $word(pos(x)) = DET \rightarrow (label(X)=NMOD, word(mod(x))=NN, pos(x) < mod(x))$
A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
- **Graph-based Dependency Parsing**
Create a **Maximum Spanning Tree** for a sentence
(MSTParser, McDonald et al. 2005)
- **Transition-based Dependency Parsing (Nivre 2008)**
- **Neural Dependency Parsing**

Dependency Parsing Approaches

Graph-based dependency parsers

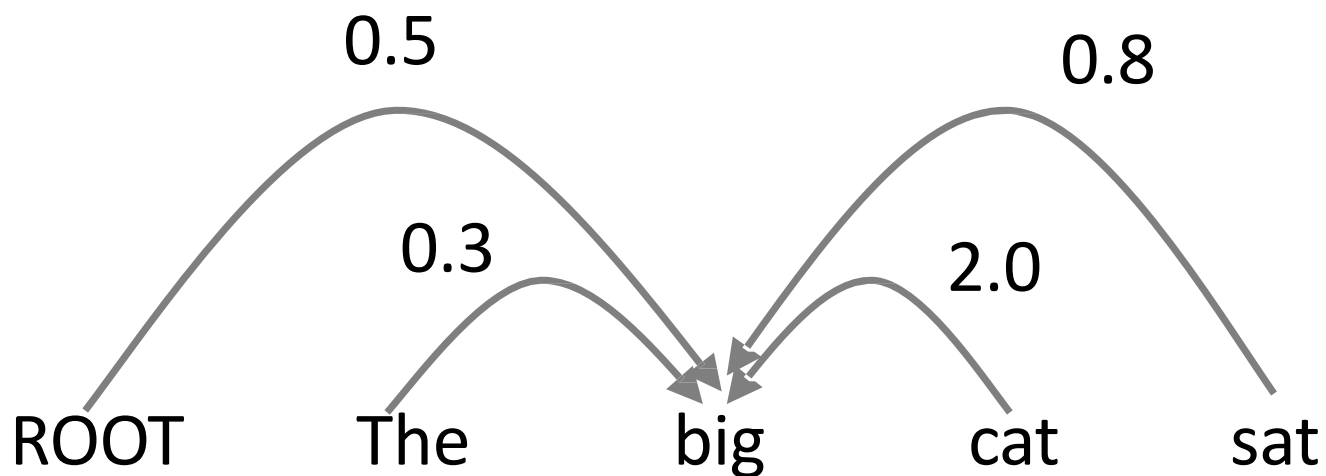
MST Parser (McDonald et al., 2005)

- 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

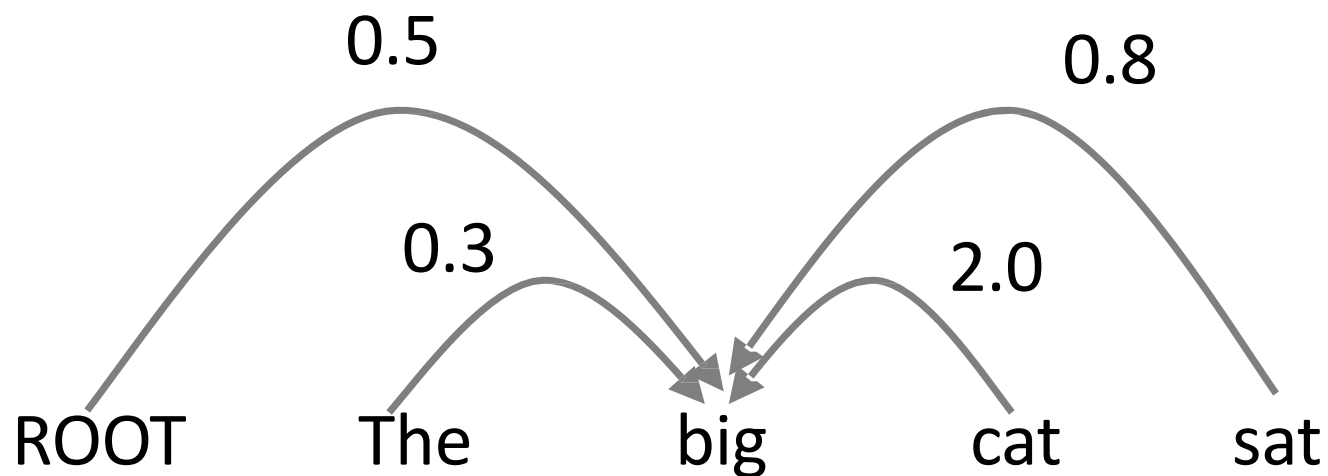


e.g., picking the head for “big”

3 Dependency Parsing Approaches

Graph-based dependency parsers

- Compute a score for every possible dependency for each edge

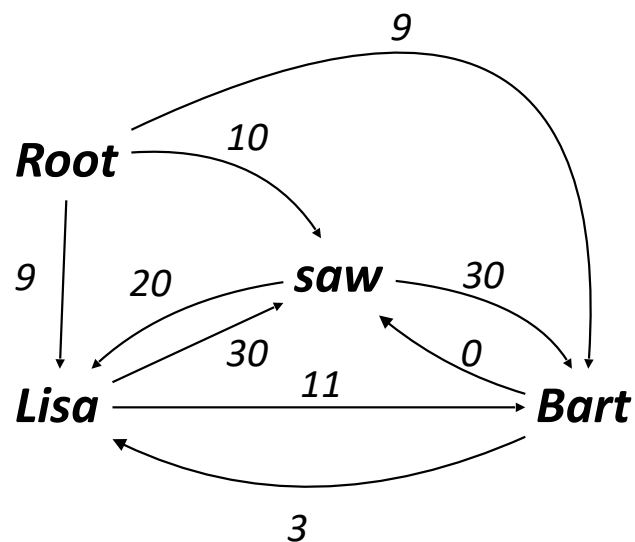


e.g., picking the head for "big"

3 Dependency Parsing Approaches

Graph-based dependency parsers

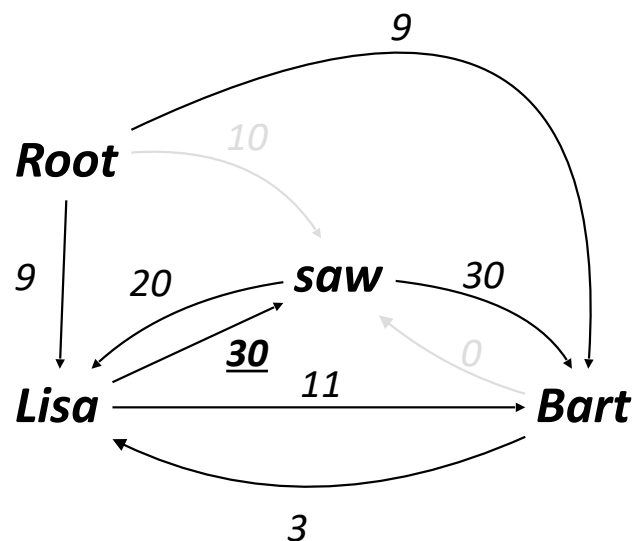
- Consider the sentence “Lisa saw Bart”



3 Dependency Parsing Approaches

Graph-based dependency parsers

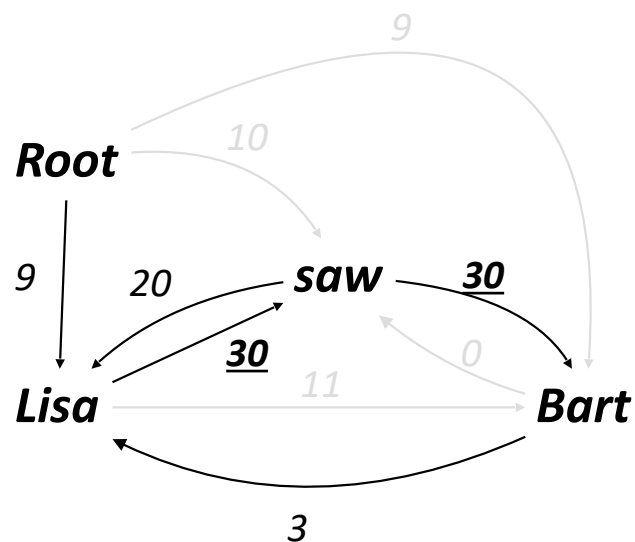
- Consider the sentence “Lisa saw Bart”



3 Dependency Parsing Approaches

Graph-based dependency parsers

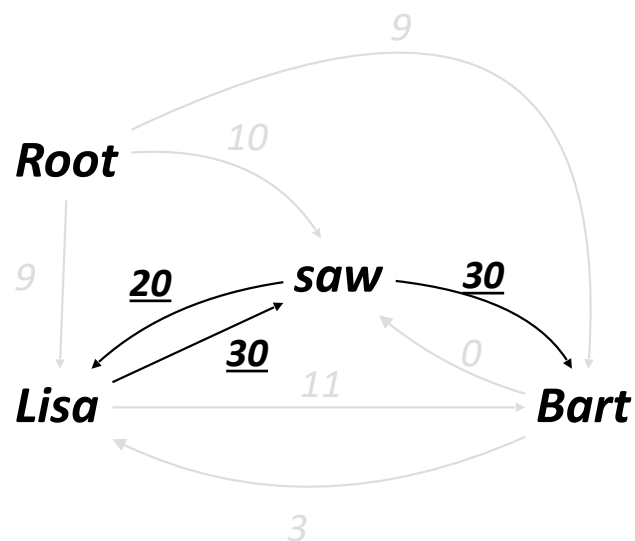
- Consider the sentence “Lisa saw Bart”



3 Dependency Parsing Approaches

Graph-based dependency parsers

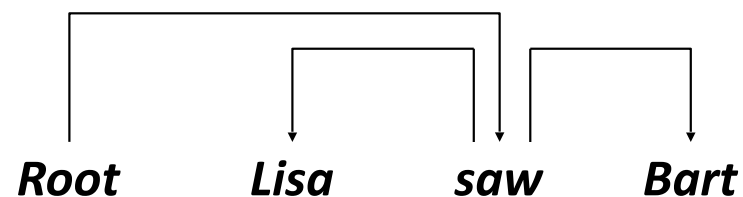
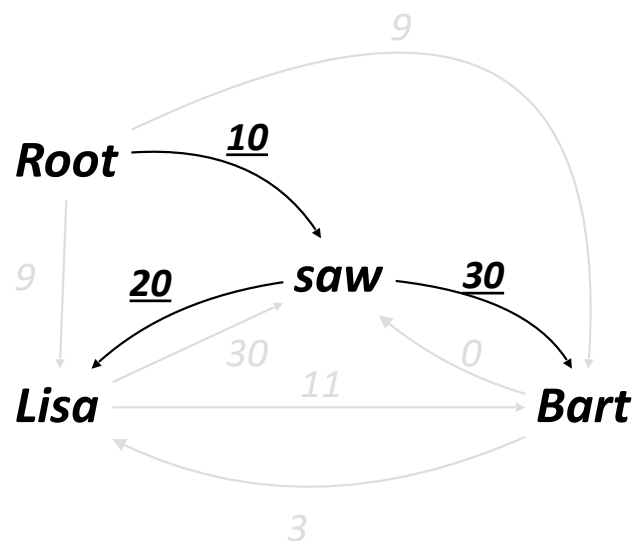
- Consider the sentence “Lisa saw Bart”



3 Dependency Parsing Approaches

Graph-based dependency parsers

- Consider the sentence “Lisa saw Bart”
- Chu and Liu (1965) and Edmonds (1967)



Dependency Parsing Approaches

Methods of Dependency Parsing

Graph-based Dependency Parsing

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

Transition-based Dependency Parsing

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

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

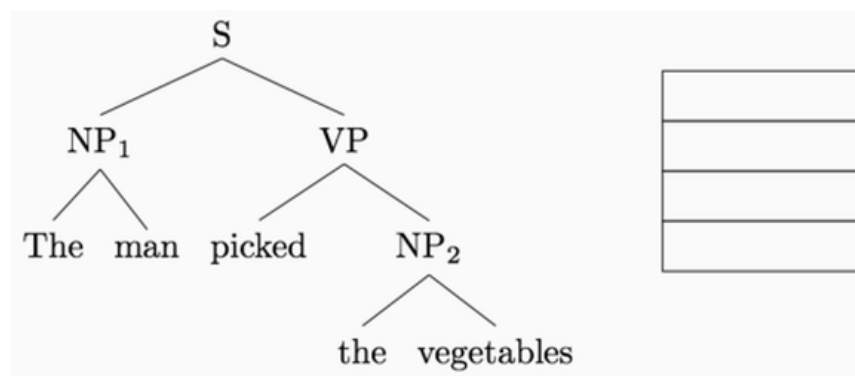
Transition-based parsing (Nivre 2008)

- **Transition:** an operation that searches for a dependency relation between each pair of words (e.g. Left-Arc, Shift, etc.)
- A transition-based dependency parser only builds one tree, in one left-to-right sweep over the input
- Nivre's approach is a greedy algorithm, but variants using beam search, A* search, and more exist.

4 Transition-based Parsing

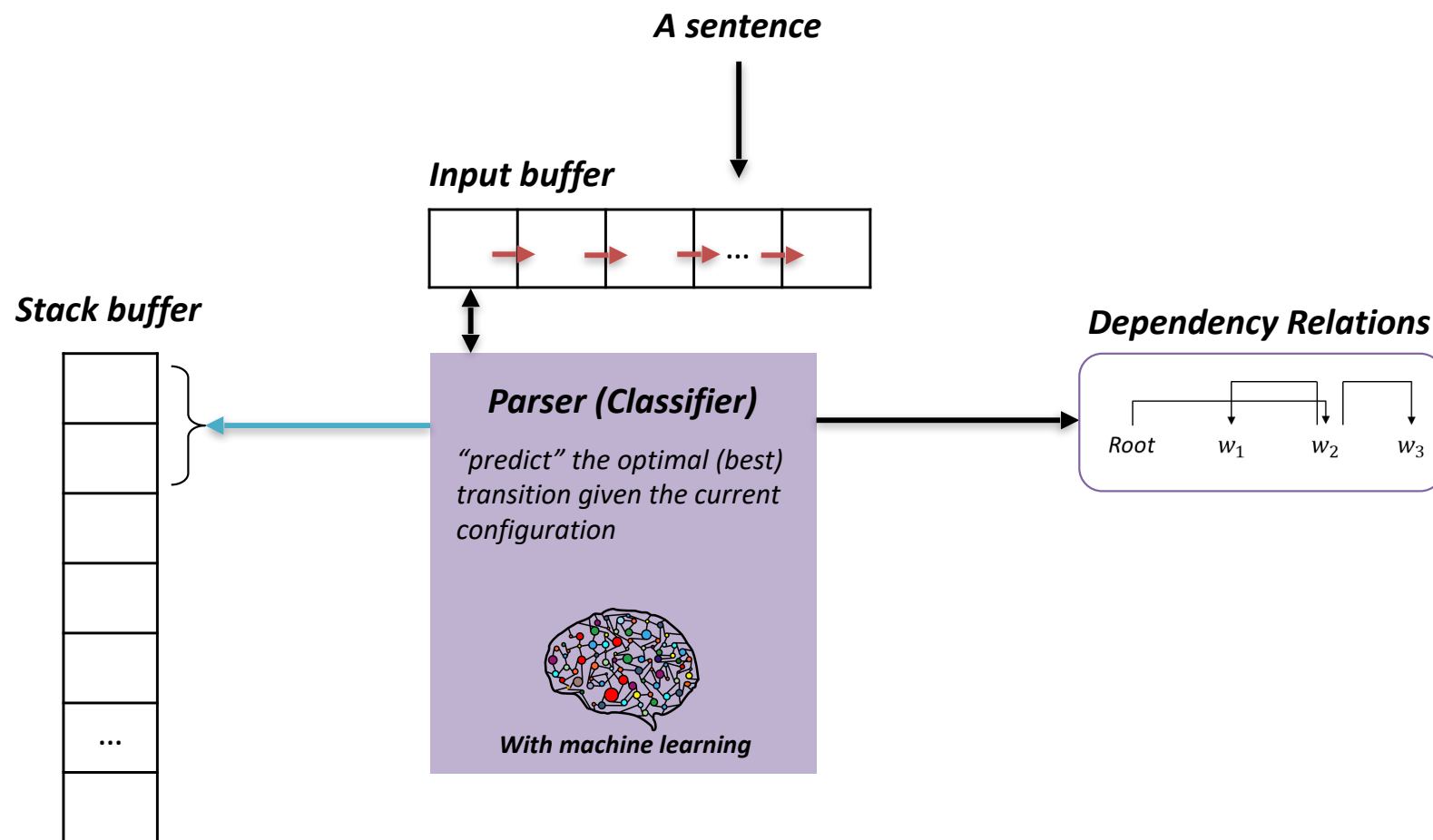
Note - the same idea can apply to constituency parsing

- A sequence of bottom up actions



4 Transition-based Parsing

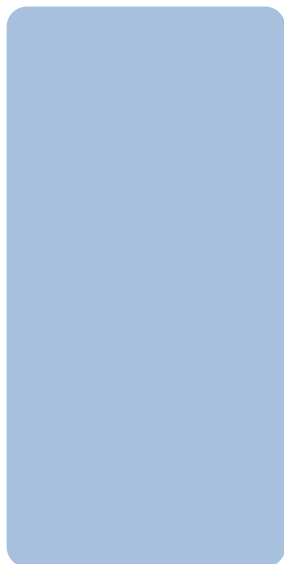
Transition-based parsing - Arc-Standard, MALT parse



4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm

Stack



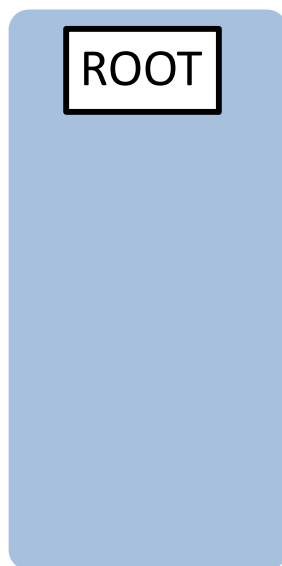
Buffer



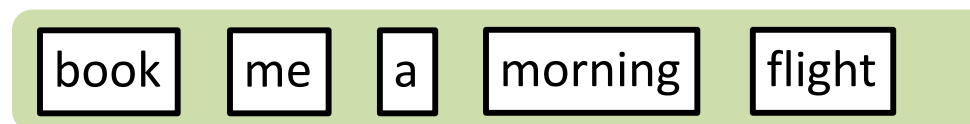
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm

Stack

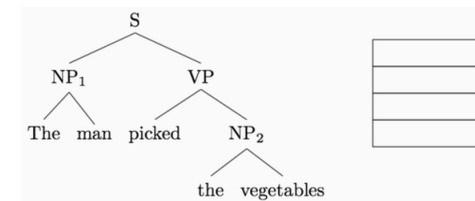


Buffer



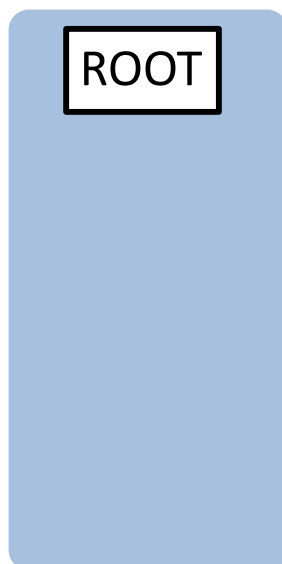
- ***Initial configuration:***
 - All words are in the buffer.
 - The stack is empty or starts with the ROOT symbol

4 Transition-based Parsing

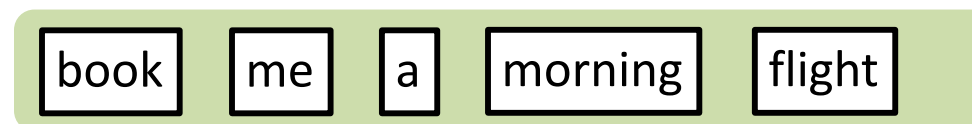


Transition-based parsing – The arc-standard algorithm

Stack



Buffer



Possible Transition

Shift

Push the next word in buffer onto the stack

Left-Arc

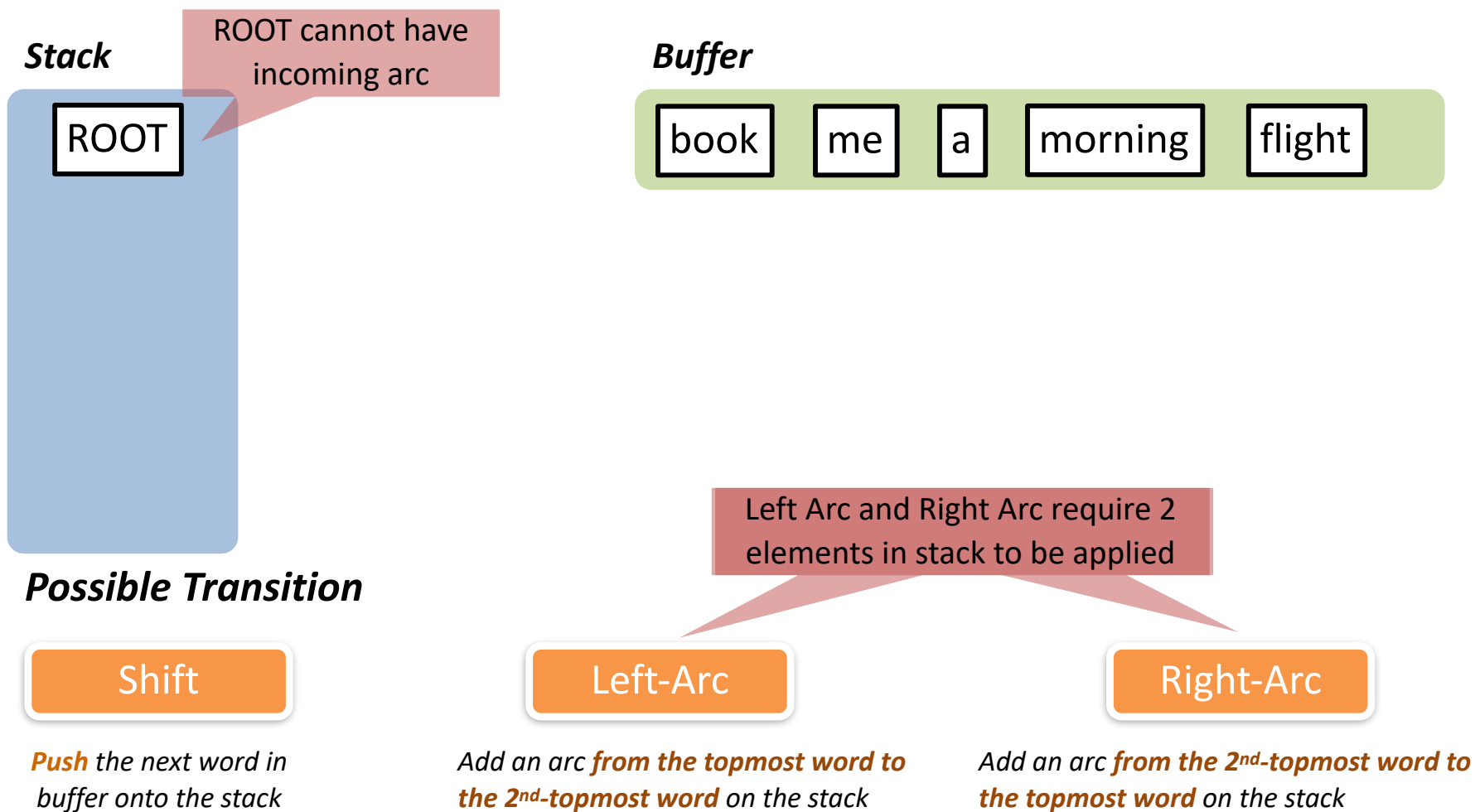
Add an arc *from the topmost word to the 2nd-topmost word* on the stack

Right-Arc

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

4 Transition-based Parsing

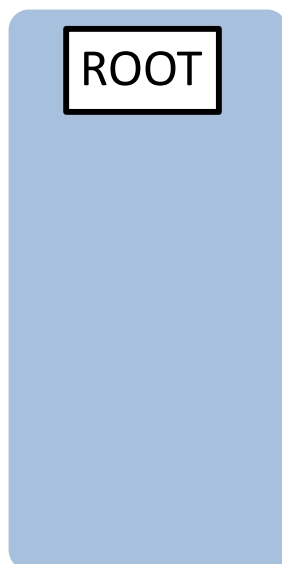
Transition-based parsing – The arc-standard algorithm



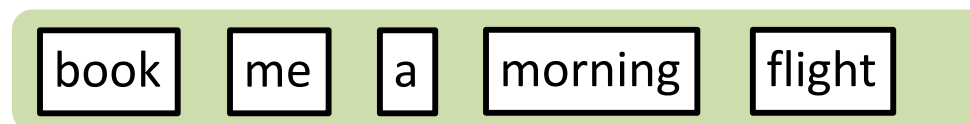
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm

Stack



Buffer



Possible Transition

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

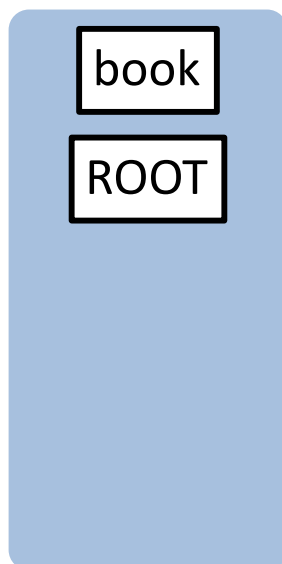
Right-Arc

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

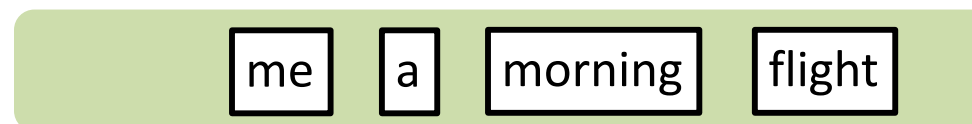
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm

Stack



Buffer



Possible Transition

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

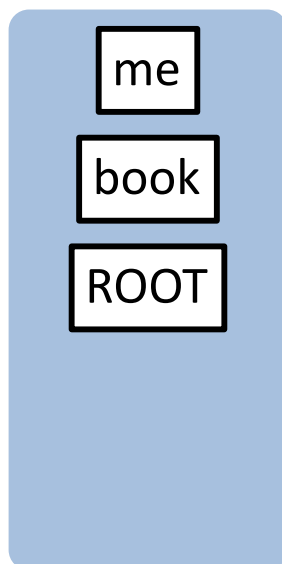
Right-Arc

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

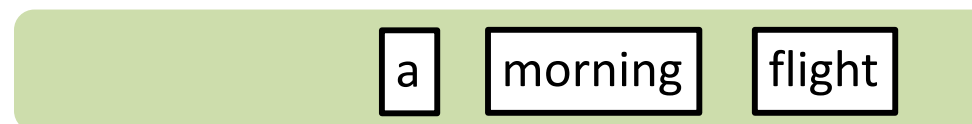
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm

Stack



Buffer



Possible Transition

Shift

Push the next word in buffer onto the stack

Left-Arc

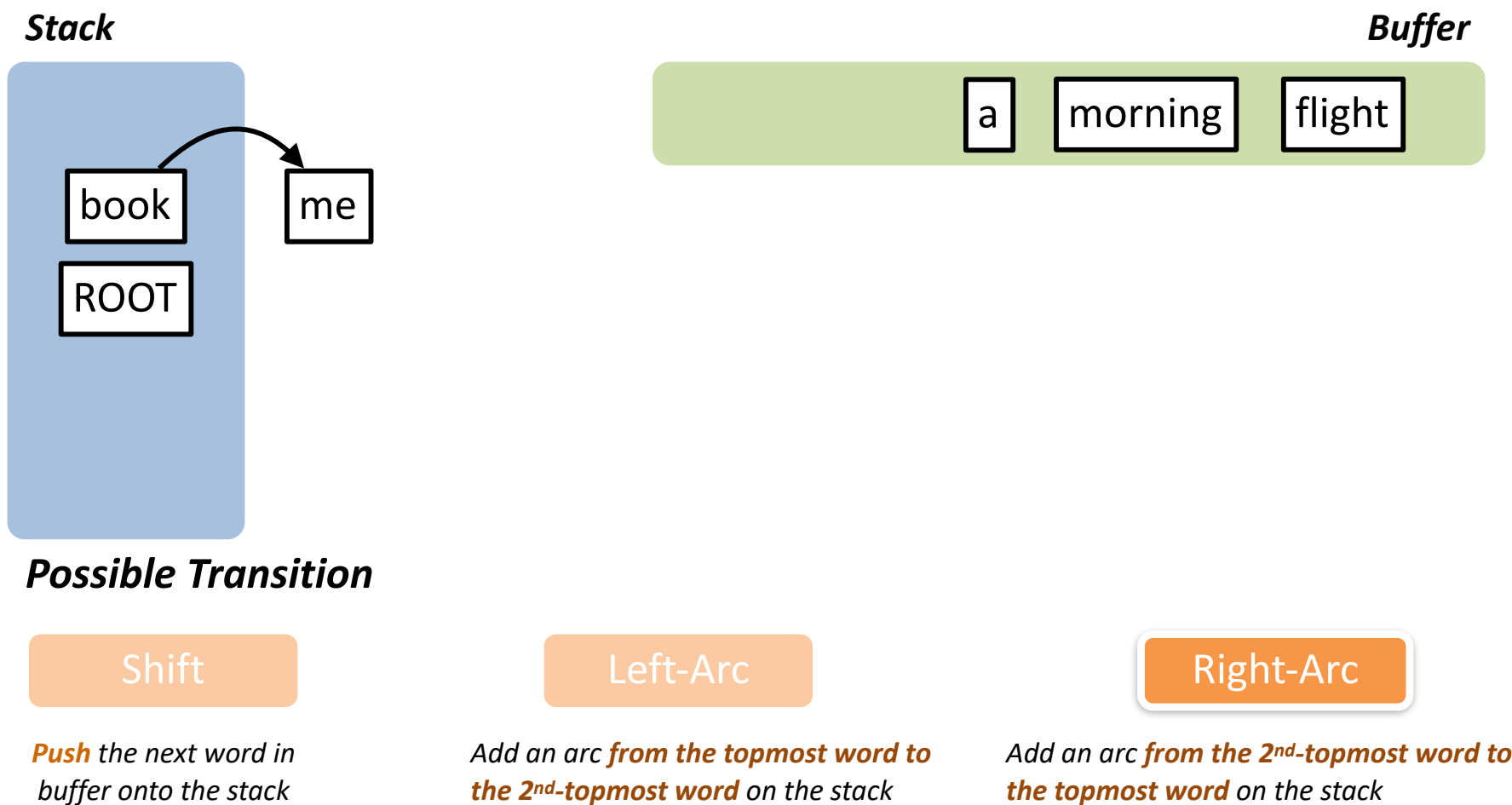
Add an arc *from the topmost word to the 2nd-topmost word* on the stack

Right-Arc

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

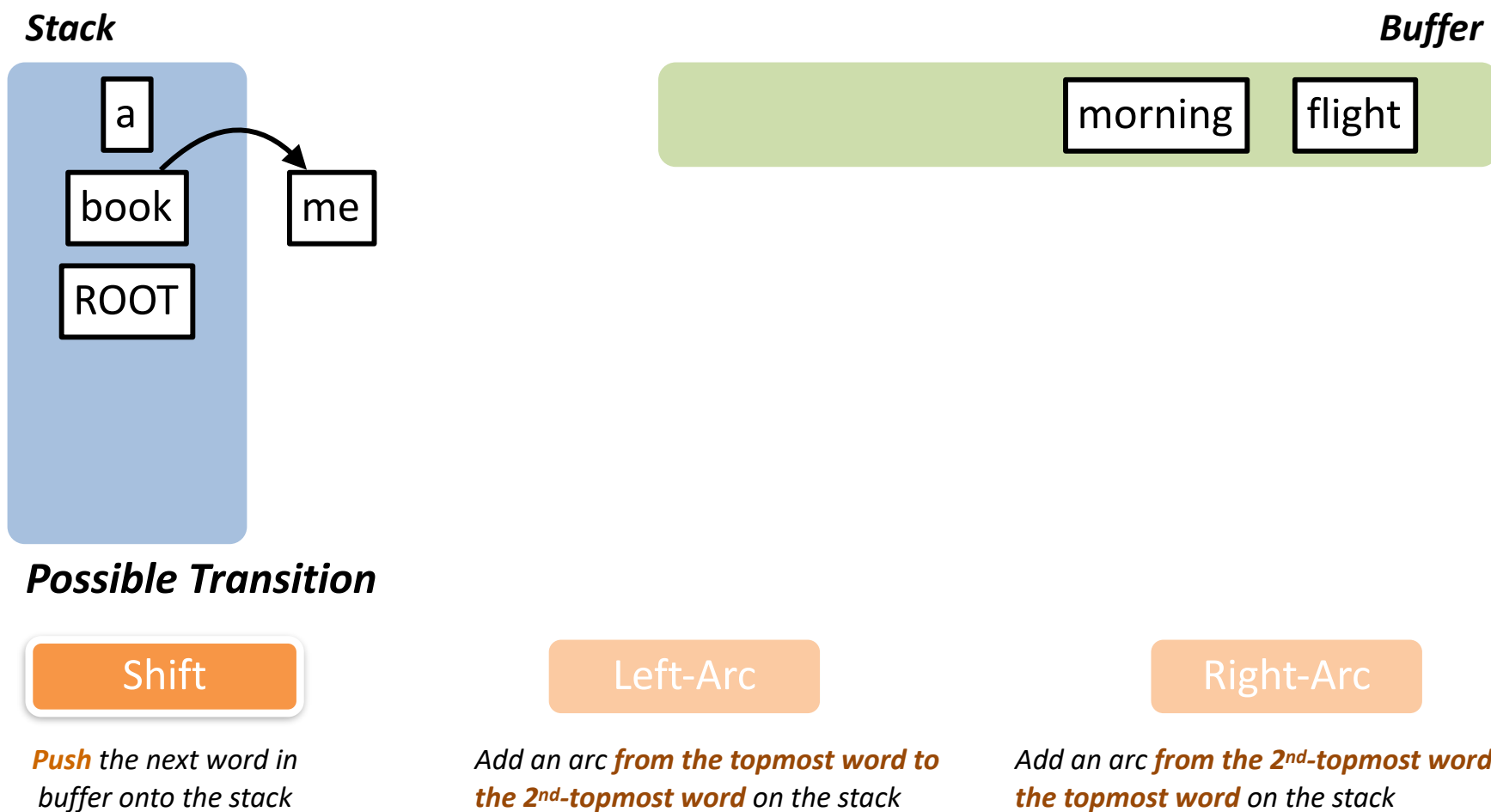
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



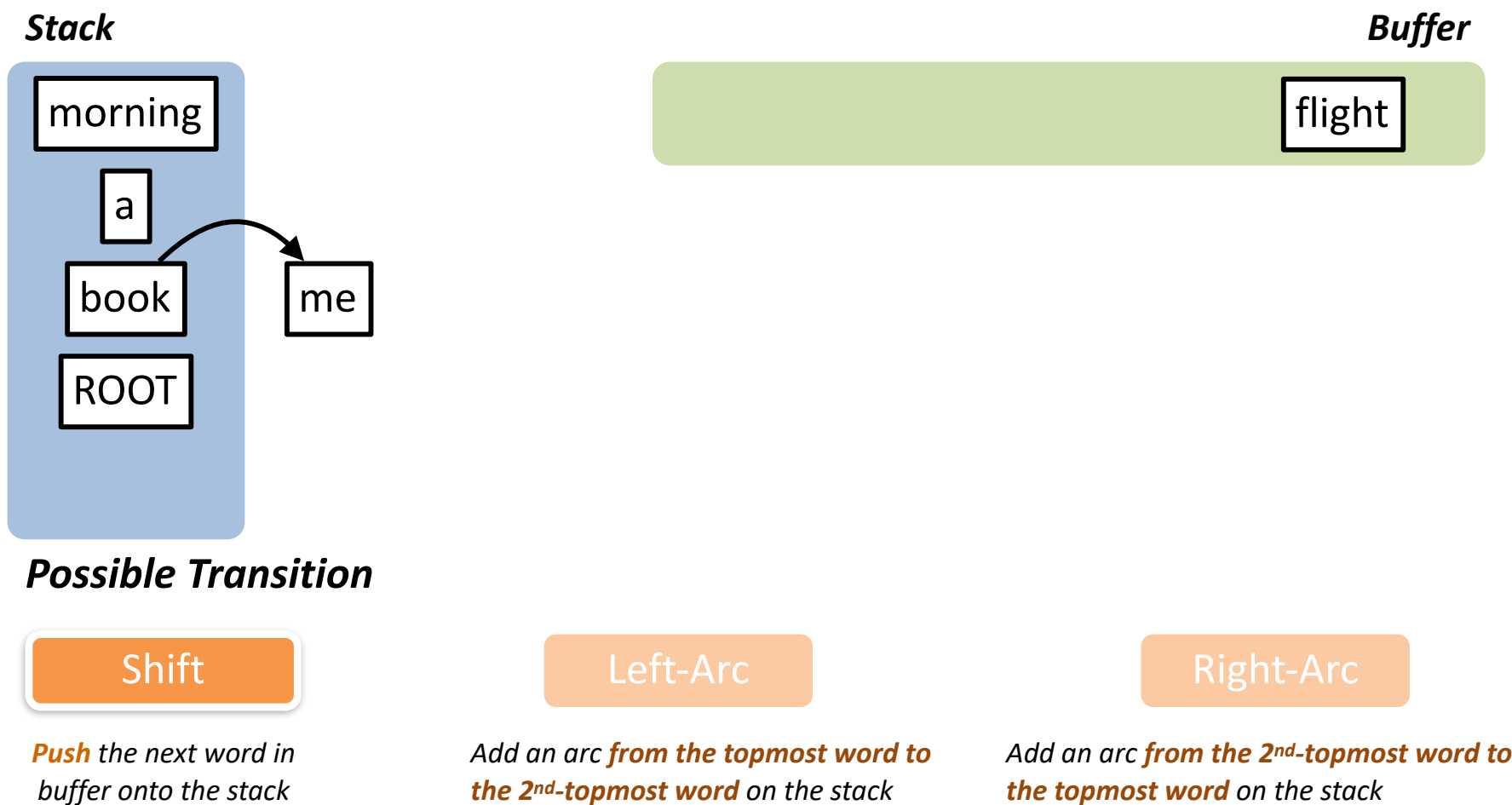
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



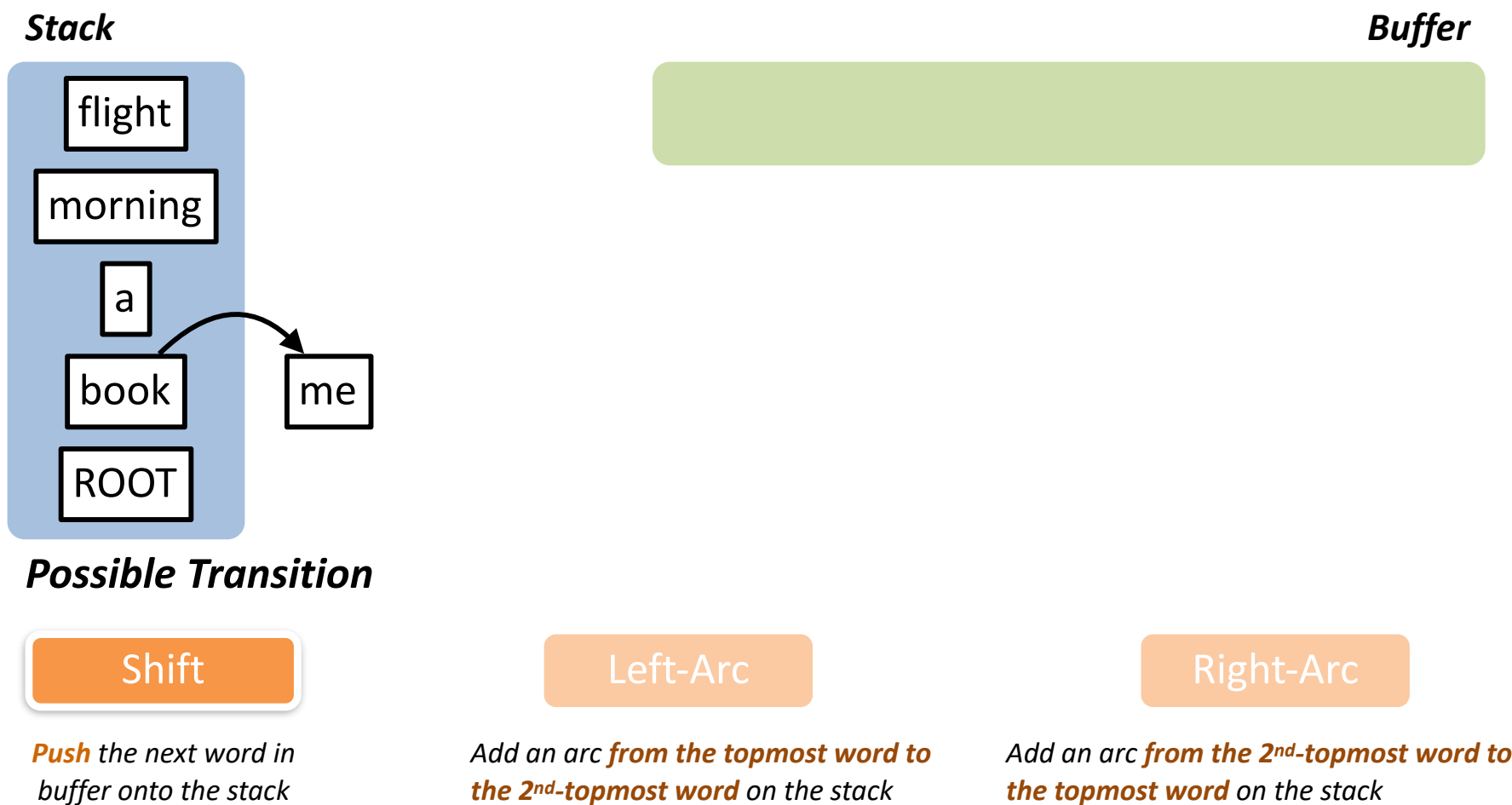
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



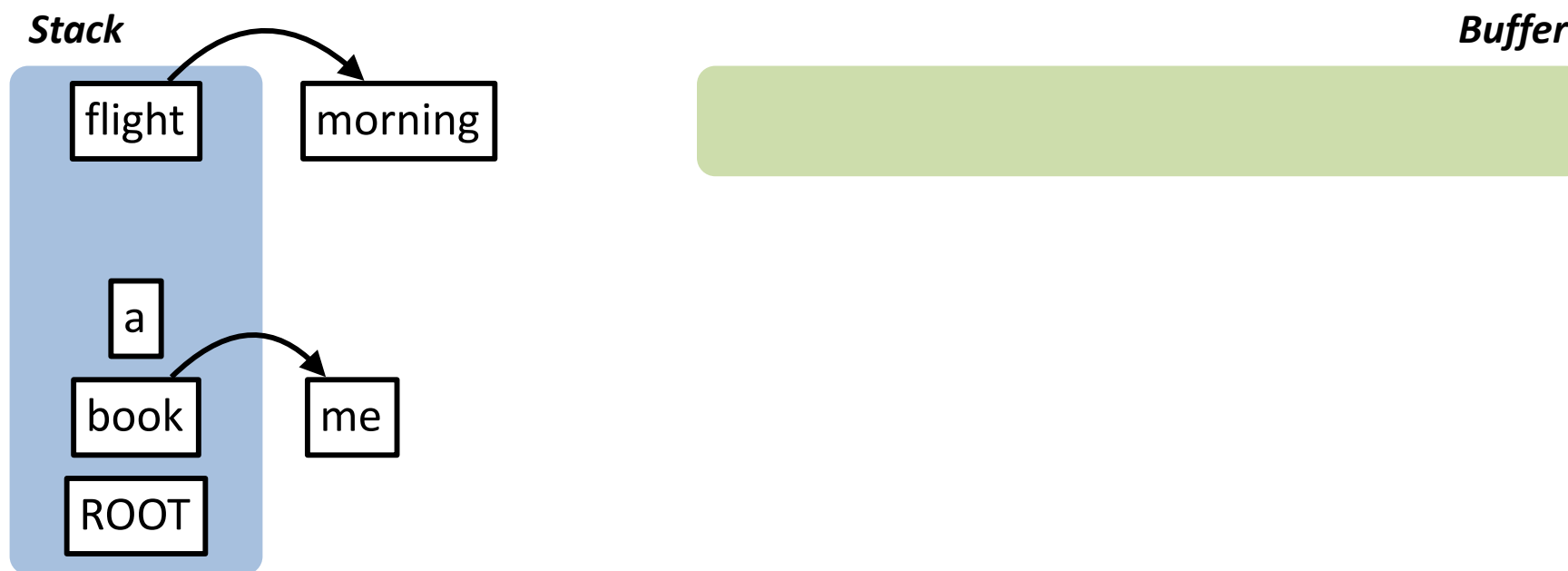
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



Possible Transition

Shift

Push the next word in buffer onto the stack

Left-Arc

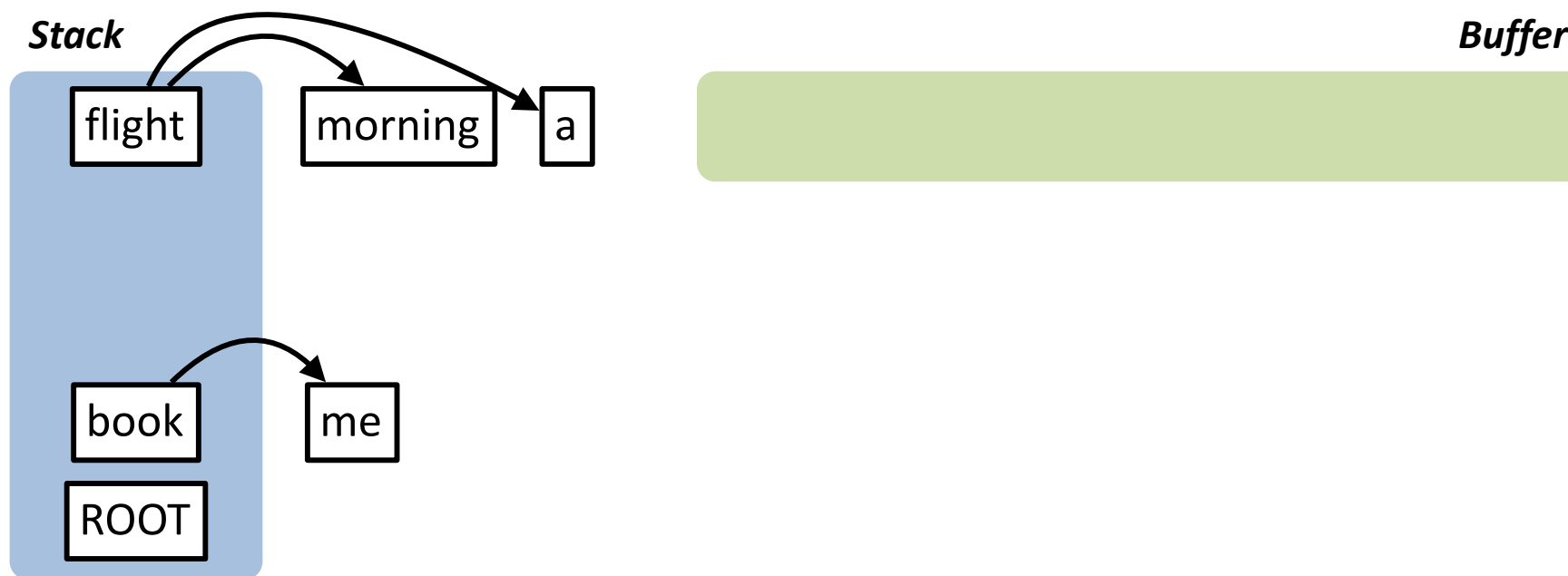
Add an arc *from the topmost word to the 2nd-topmost word* on the stack

Right-Arc

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

4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



Possible Transition

Shift

Push the next word in buffer onto the stack

Left-Arc

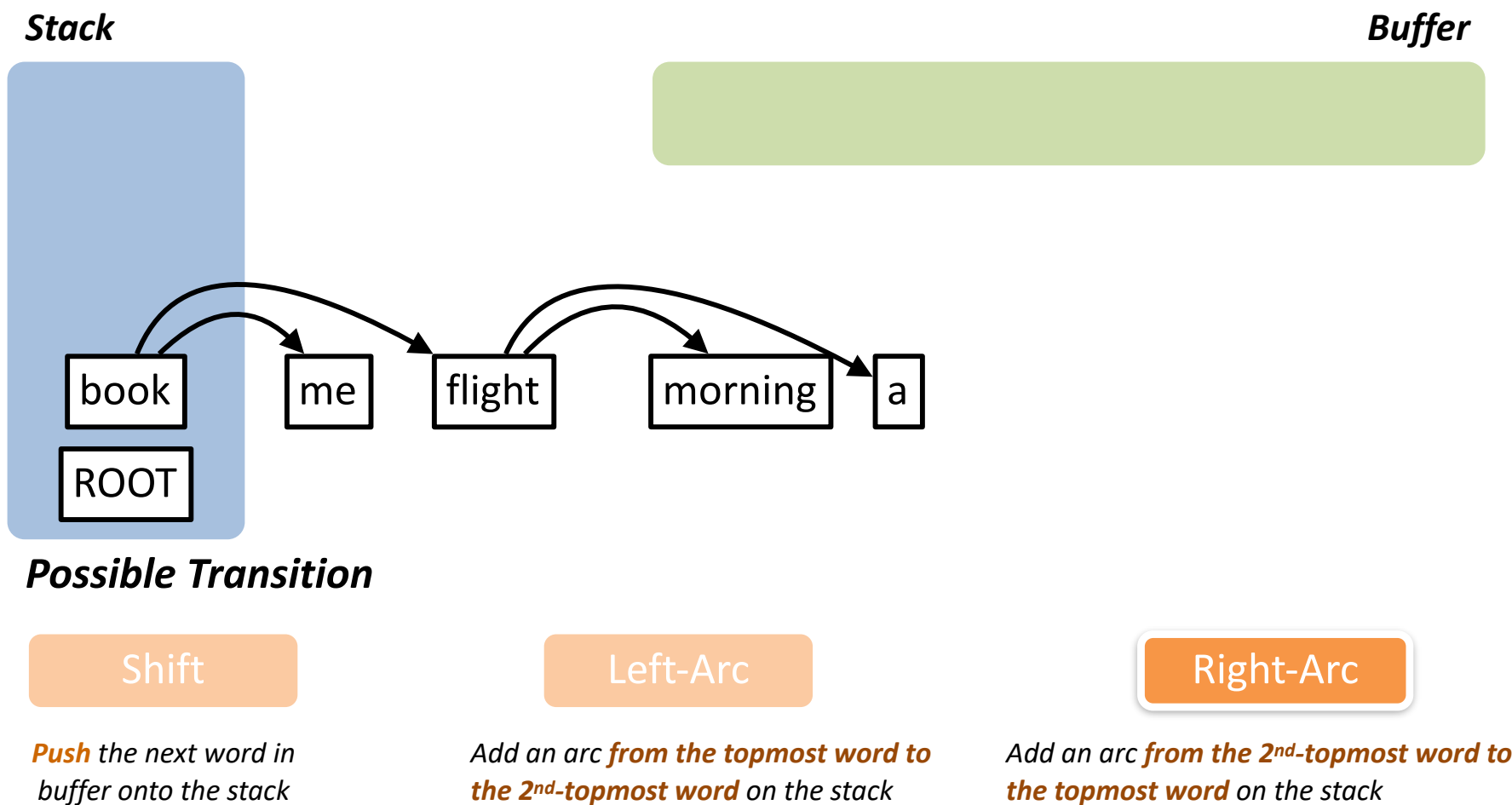
Add an arc *from the topmost word to the 2nd-topmost word* on the stack

Right-Arc

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

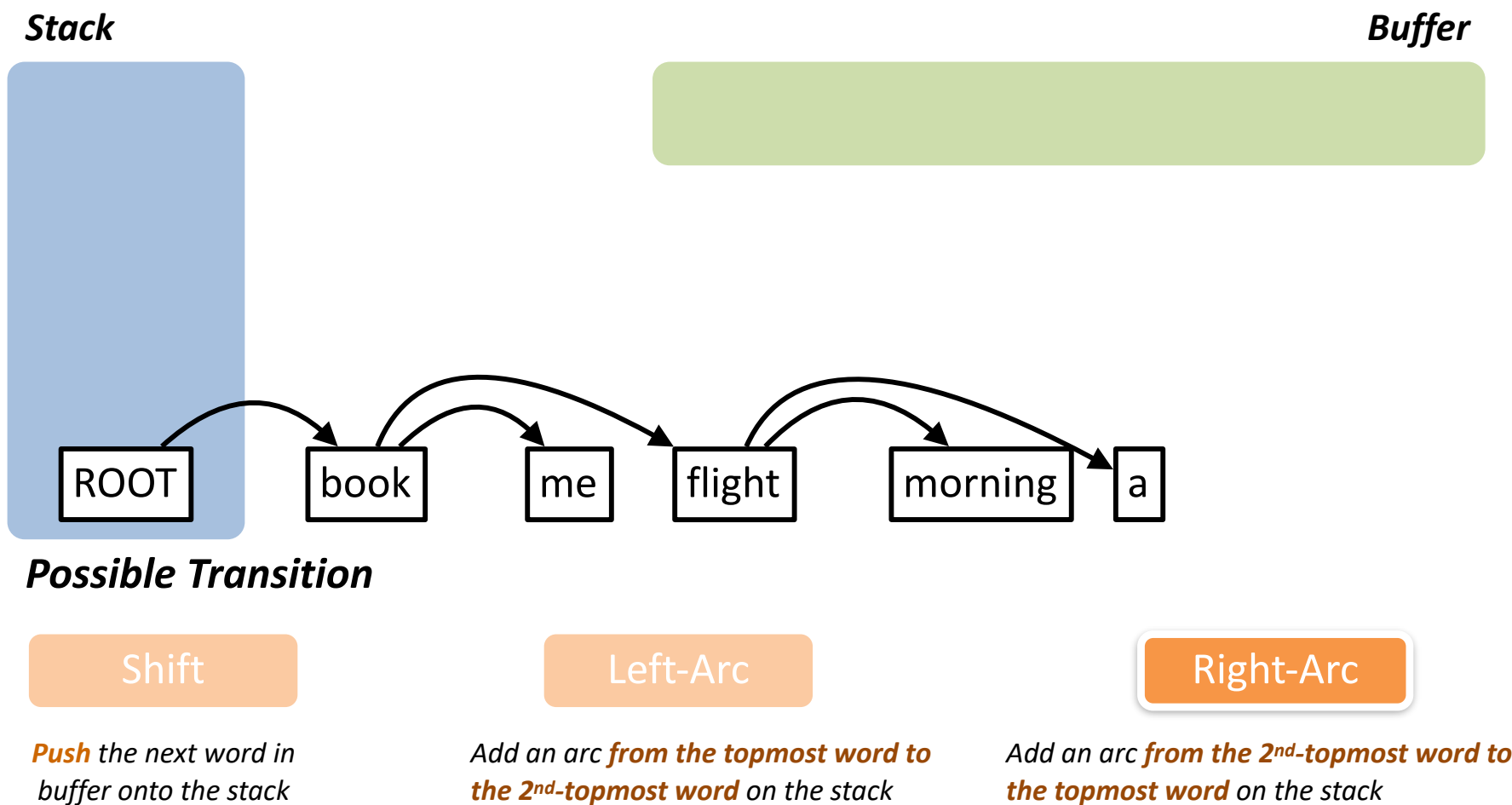
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



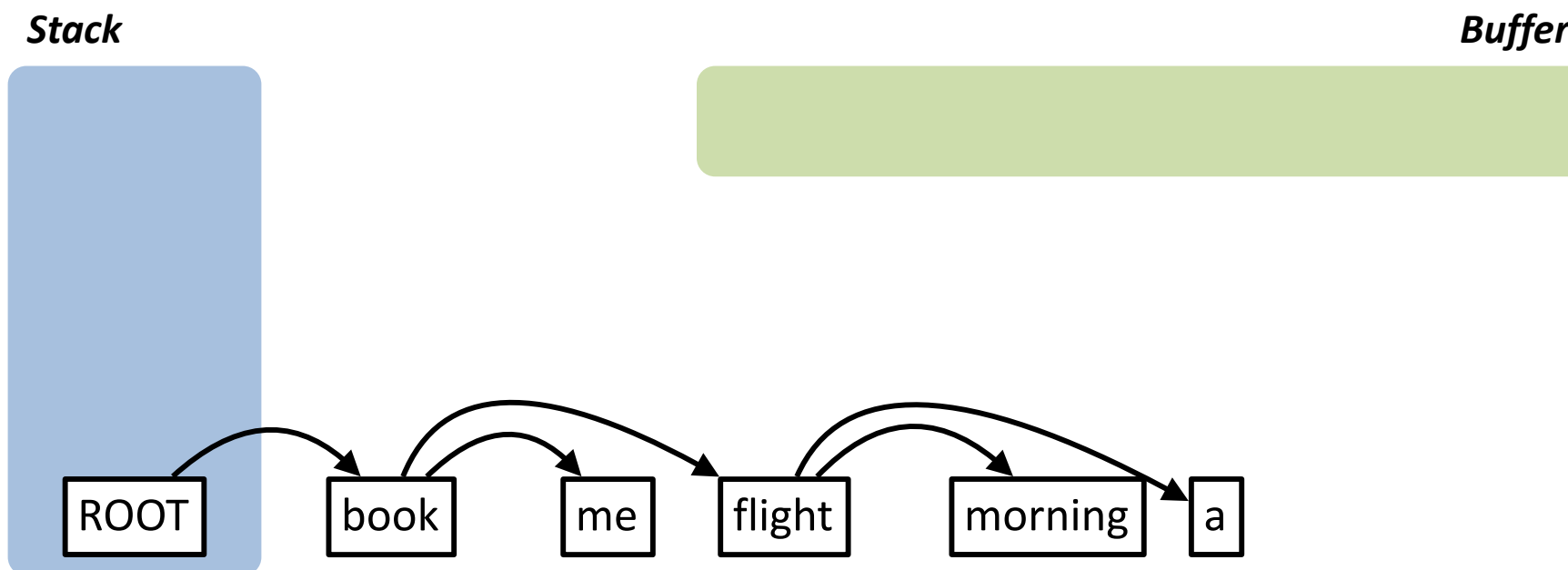
4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



4 Transition-based Parsing

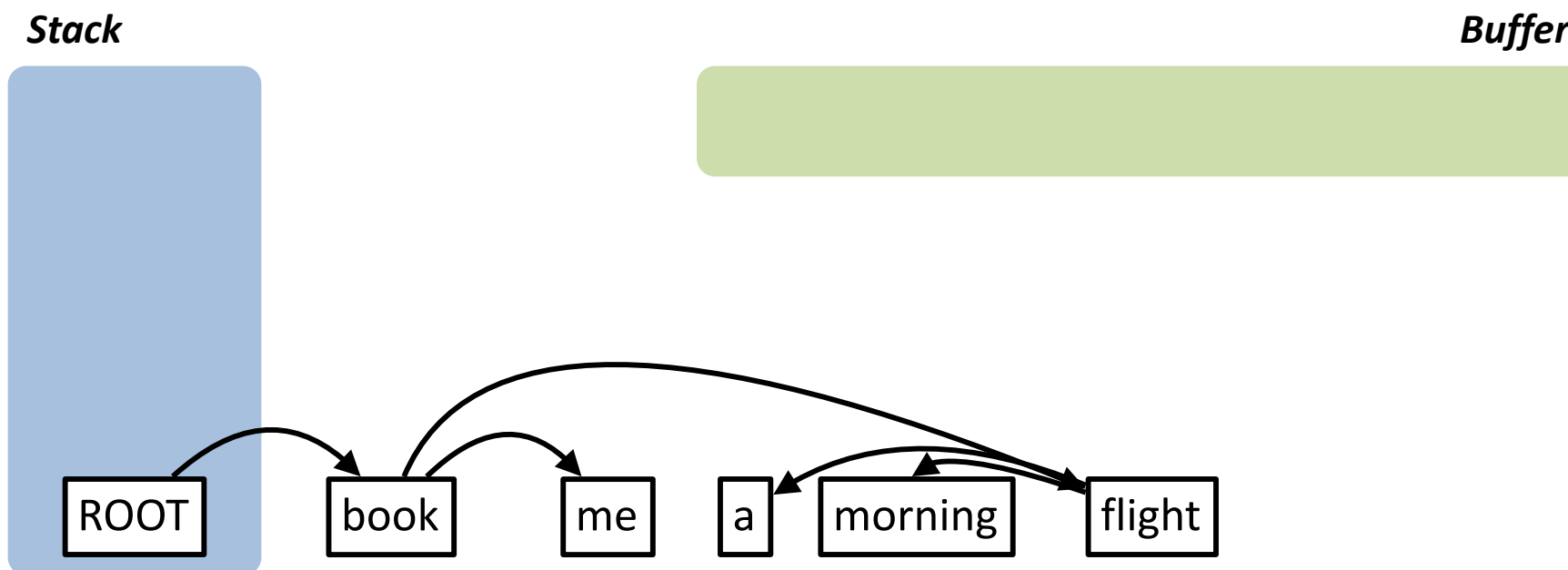
Transition-based parsing – The arc-standard algorithm



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

4 Transition-based Parsing

Transition-based parsing – The arc-standard algorithm



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

4 Transition-based Parsing

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

Transition-based parsing – The arc-standard algorithm

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

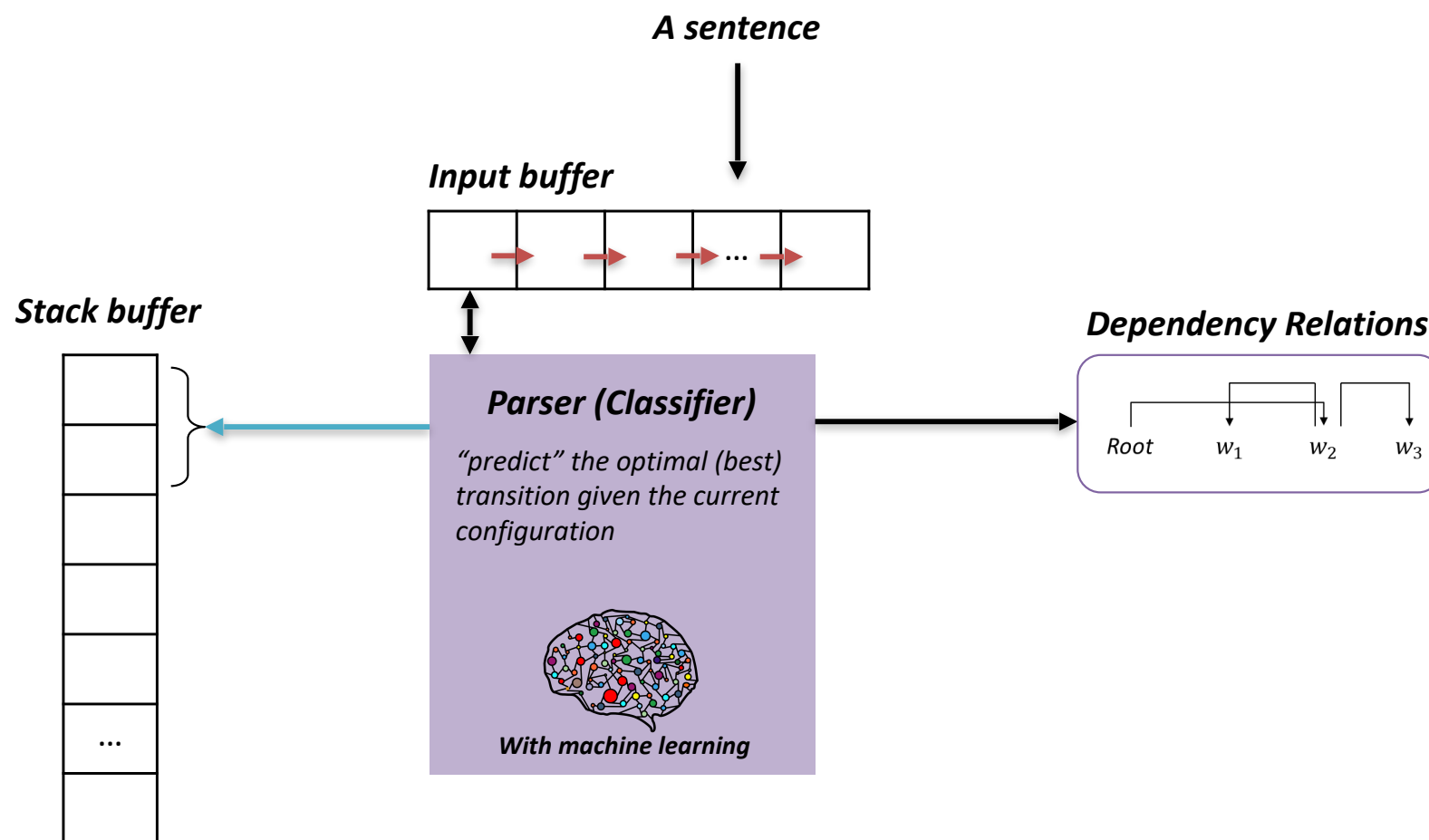
1. Shift $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
2. Left-Arc_r $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc_r $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$

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

How do we choose the next action?

4 Transition-based Parsing

Transition-based parsing



4 Transition-based Parsing

How to choose the next action?

Machine learning!

Goal: Predict the next transition (ie., a classification task), given the current configuration.

Training (simple idea - high r is more complex):

- 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?

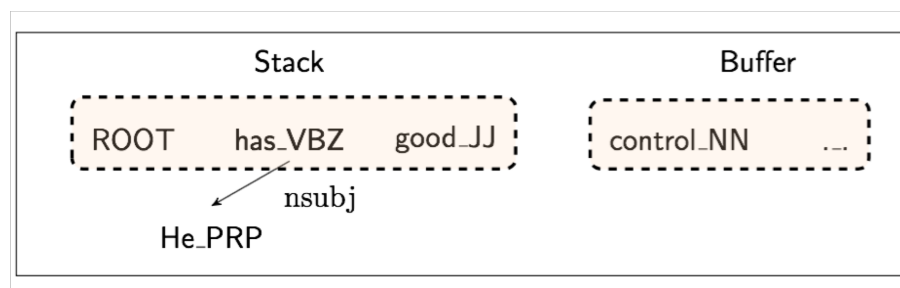
4 Transition-based Parsing

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

4 Transition-based Parsing

Evaluation of Dependency Parsing

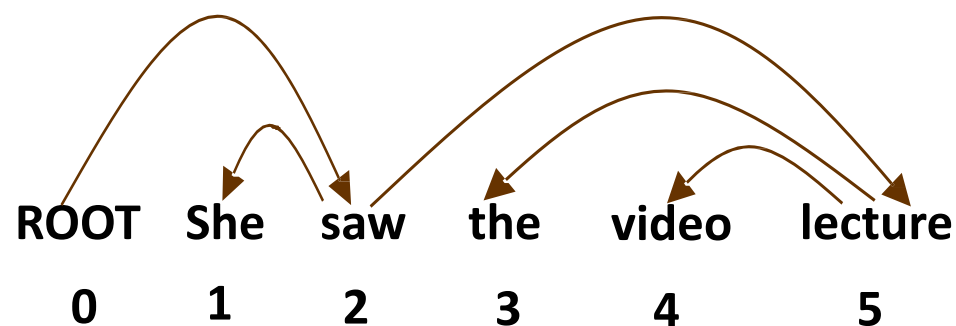
$$Accuracy = \frac{\# \text{ correct deps}}{\# \text{ of deps}}$$

Unlabeled attachment score (UAS) = head

Labeled attachment score (LAS) = head and label

4 Transition-based Parsing

Evaluation of Dependency Parsing



Gold Standard

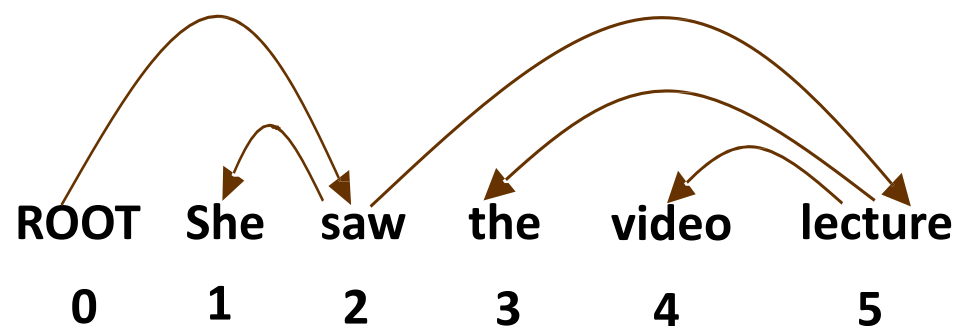
1	2	she	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

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

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%

Lecture 7: Parsing

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

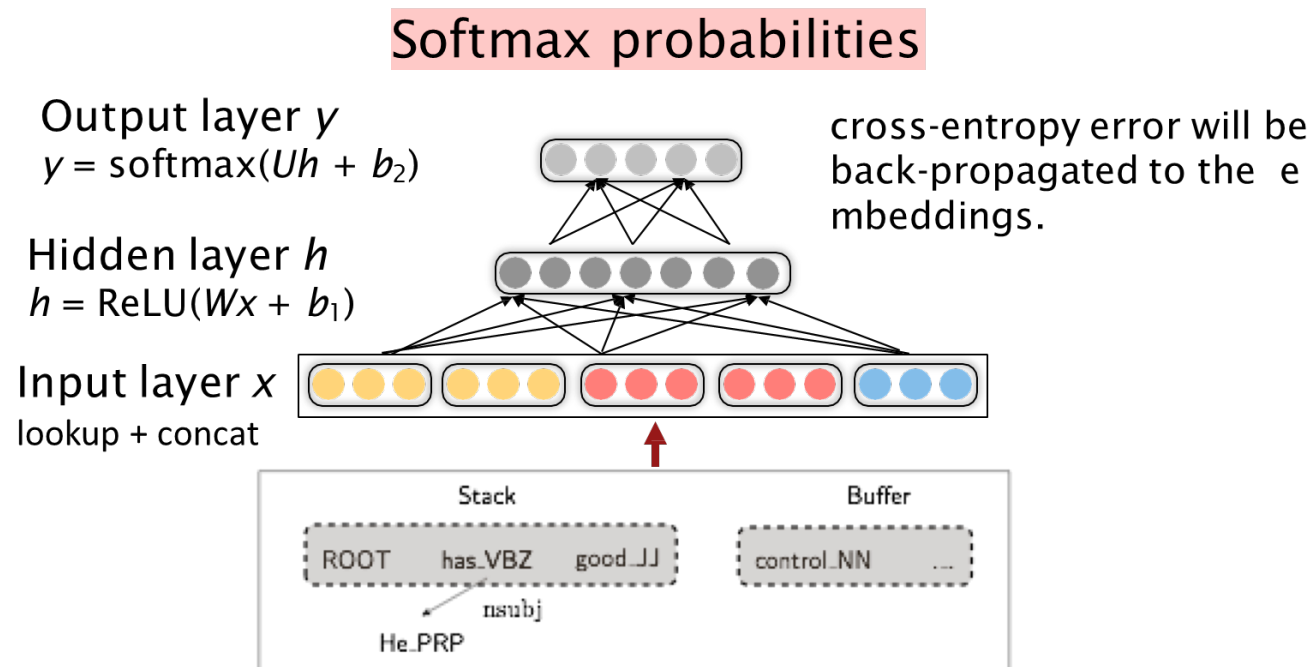
Deep Learning-based Parsing

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 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

5 Deep Learning-based Parsing

Neural Dependency Parsing (Chen & Manning, 2014)



5 Deep Learning-based Parsing

Neural 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	92.0	89.7	91.8	89.6	654

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	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).

/ Final Exercise

VB PRP
Thank you!

VBP DT JJ NN
Have a great night!

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.