



PARSING

PCFG, Recursive Neural Network

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

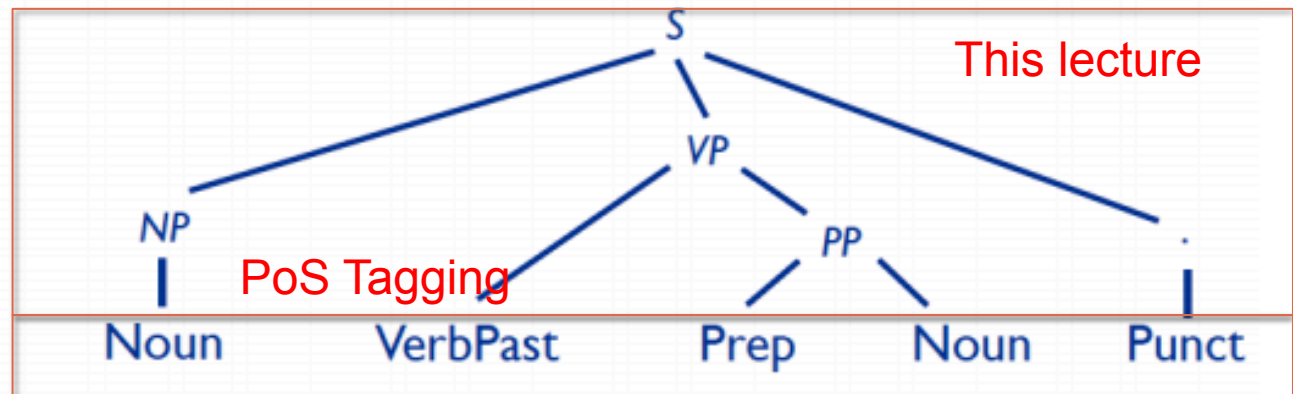
Words

Morphology

Characters

CommunicationEvent(e) SpeakerContext(s)
Agent(e,Alice) TemporalBefore(e, s)
Recipient(e, Bob)

Word embeddings



Alice talked to Bob .

Language modeling

↑ Tokenization

talk -ed [VerbPast]

Alice talked to Bob .

Use Syntax to create sentence level meanings!

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

Words

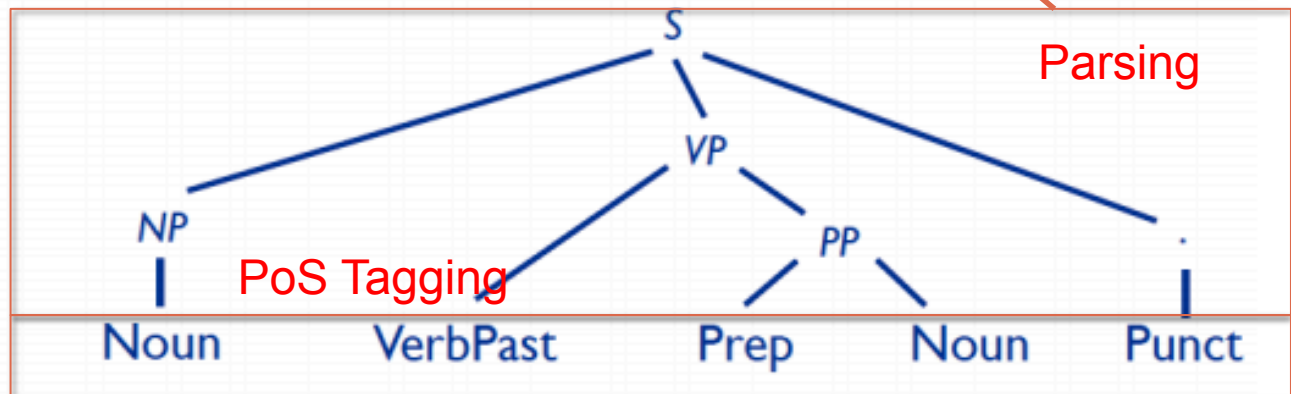
Morphology

Characters

Ref: Prof. Brendan O'Connor, CS 885 Intro to NLP, @UMass

CommunicationEvent(e) SpeakerContext(s)
Agent(e,Alice) TemporalBefore(e, s)
Recipient(e, Bob)

Word embeddings



PoS Tagging

Alice talked to Bob .

Language modeling

Tokenization

talk -ed [VerbPast]

Alice talked to Bob .

Semantic embeddings of several words

- Compositionality
- We know how to create a dense vector representation for a word
 - What about larger linguistic units? (e.g. phrase, sentence)
- We can combine smaller units into a larger unit

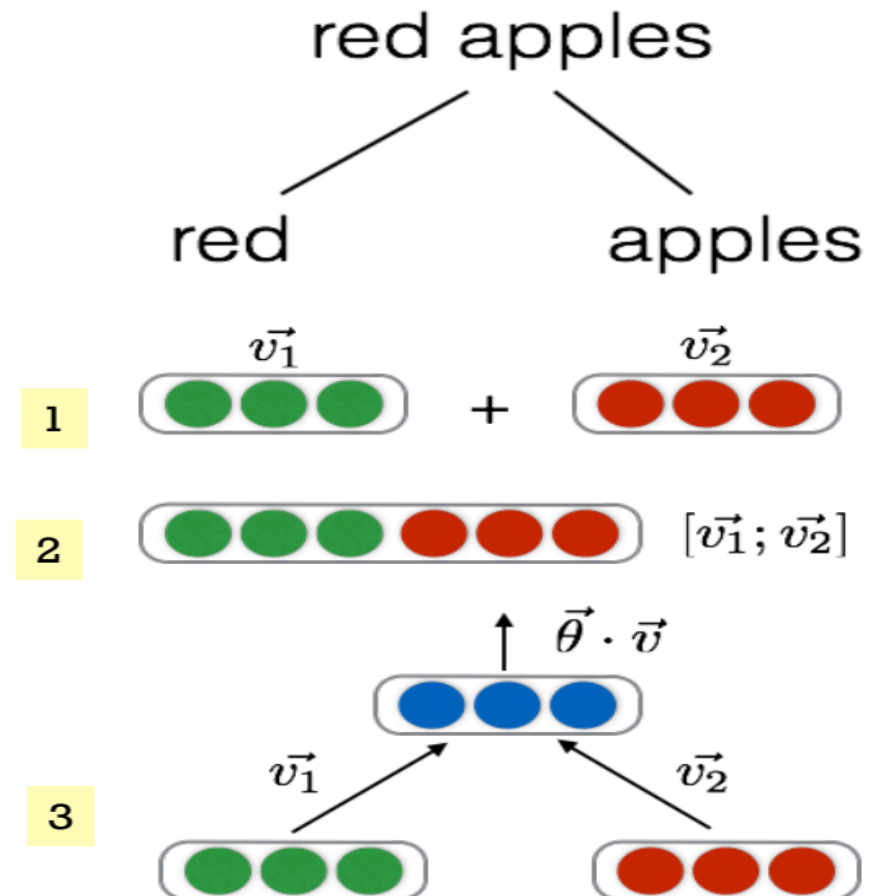
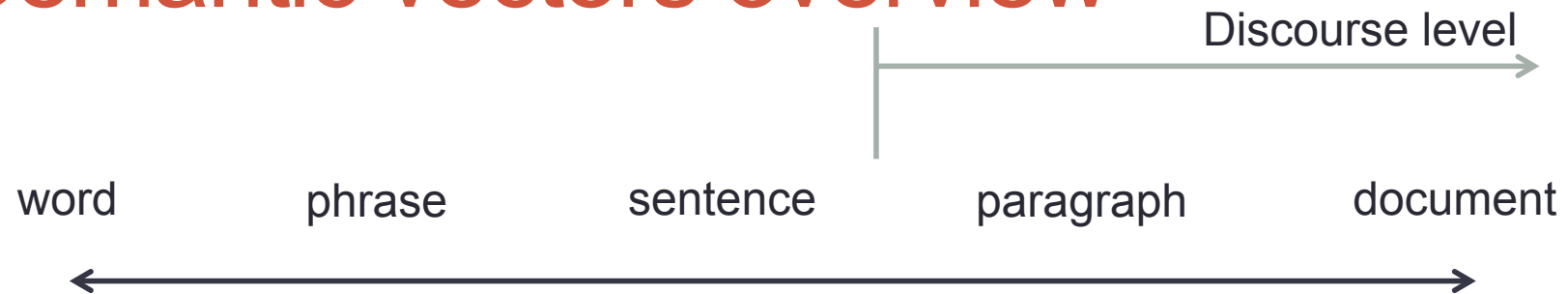


Image ref: Prof. Regina Barzilay , NLP@MIT

Semantic vectors overview



- Word vectors
 - Co-occurrence
 - PPMI
 - TFIDF
 - Word2vec
 - CBoW
 - Skip-gram

- Doc vectors
 - Term-document
 - Bag of words model
- Recurrent networks (HW3 ToDo#6)

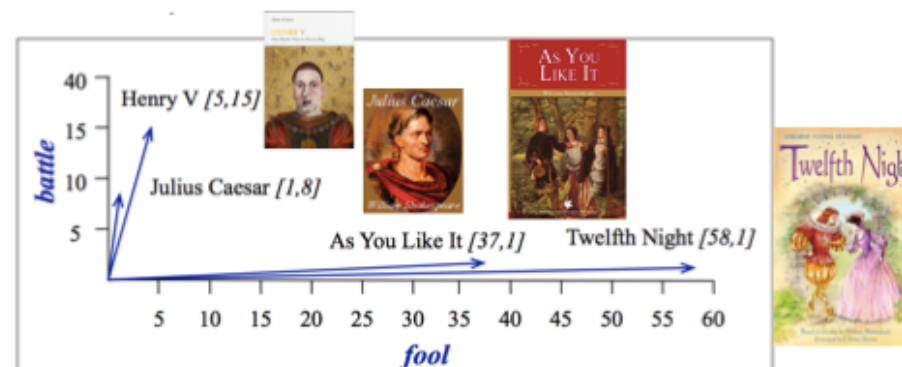


Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Reference: Jurafsky, Dan, and James H. Martin. Speech and language processing. 3rd edition draft, <https://web.stanford.edu/~jurafsky/slp3/>, August 2017

Semantic vectors overview

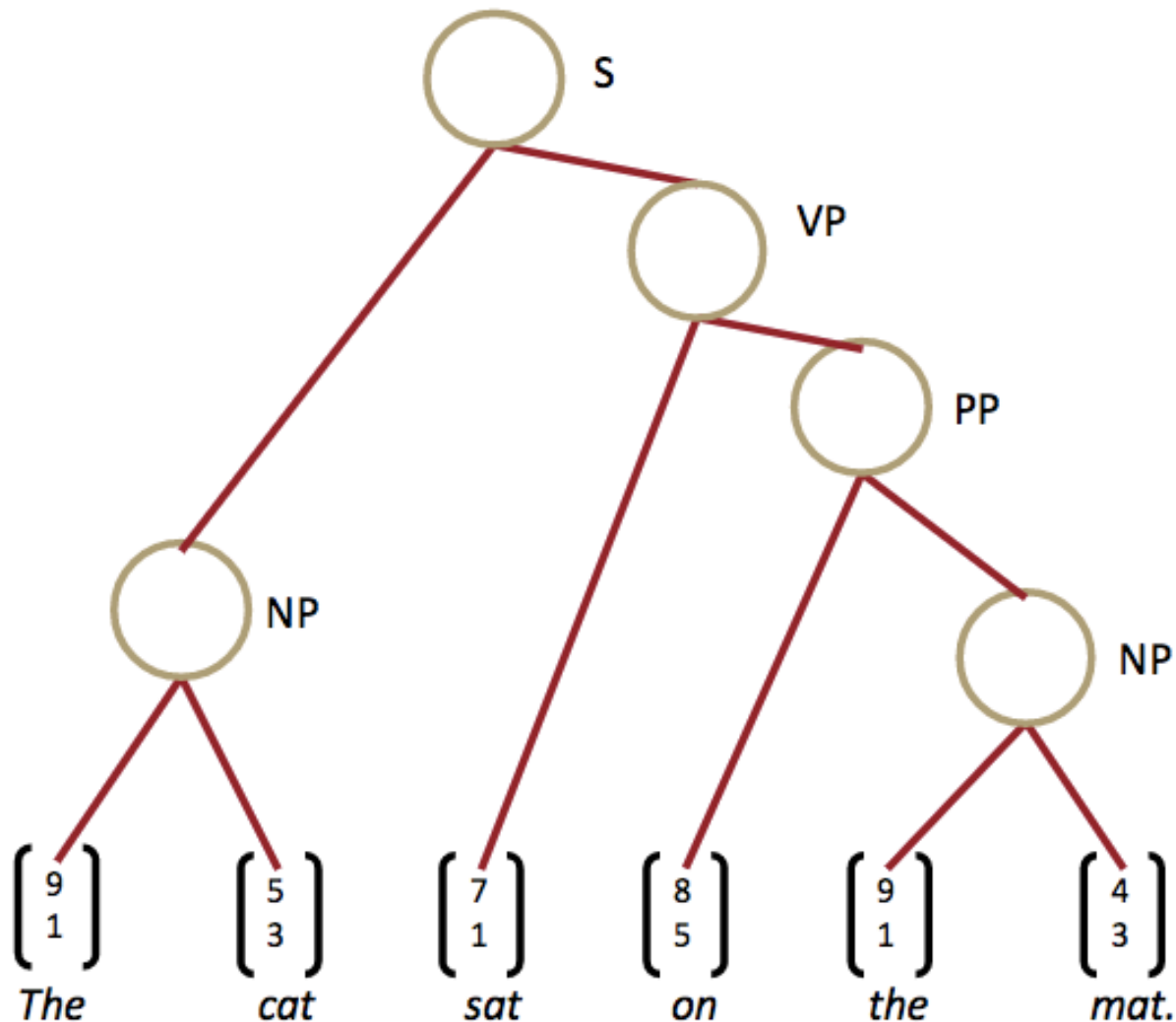


How do we represent things in this level? Without ignoring word order (bag of words)

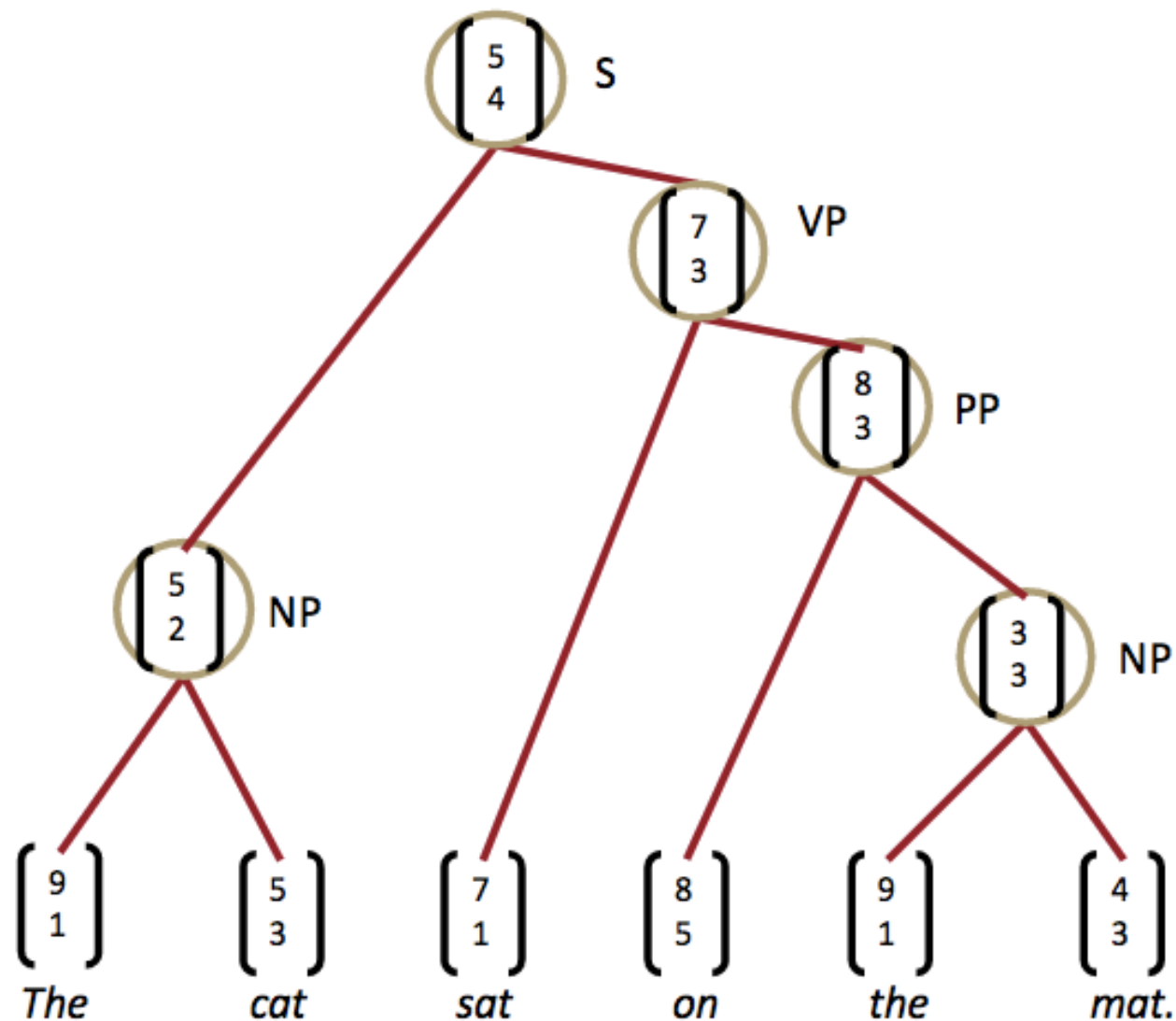
- Word vectors
 - Co-occurrence
 - PPMI
 - TFIDF
 - Word2vec
 - CBoW
 - Skip-gram

- Doc vectors
 - Term-document
 - Bag of words model
 - Recurrent networks (HW3 ToDo#6)

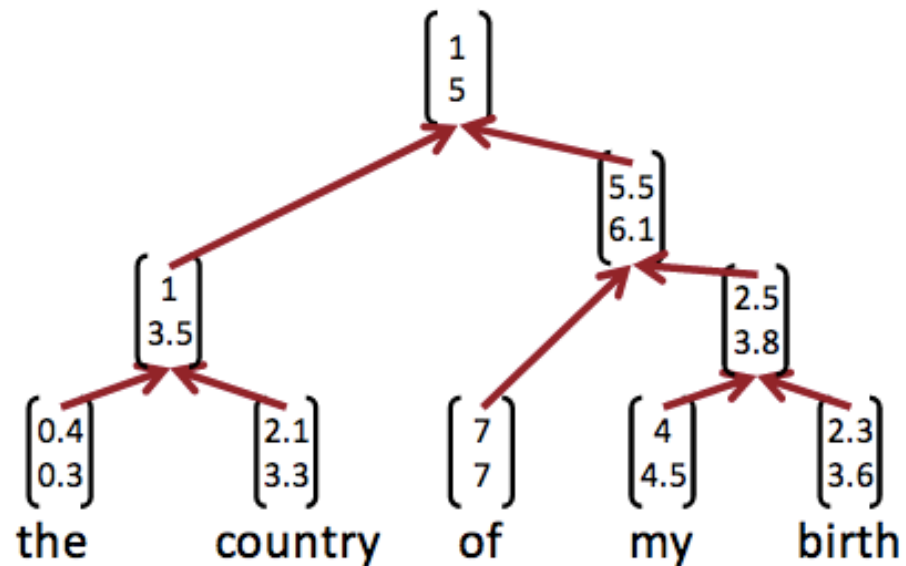
Parsing and representation big picture



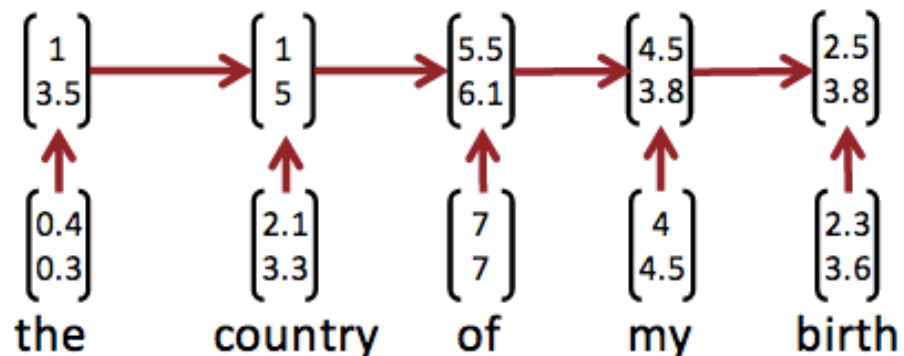
Parsing and representation big picture



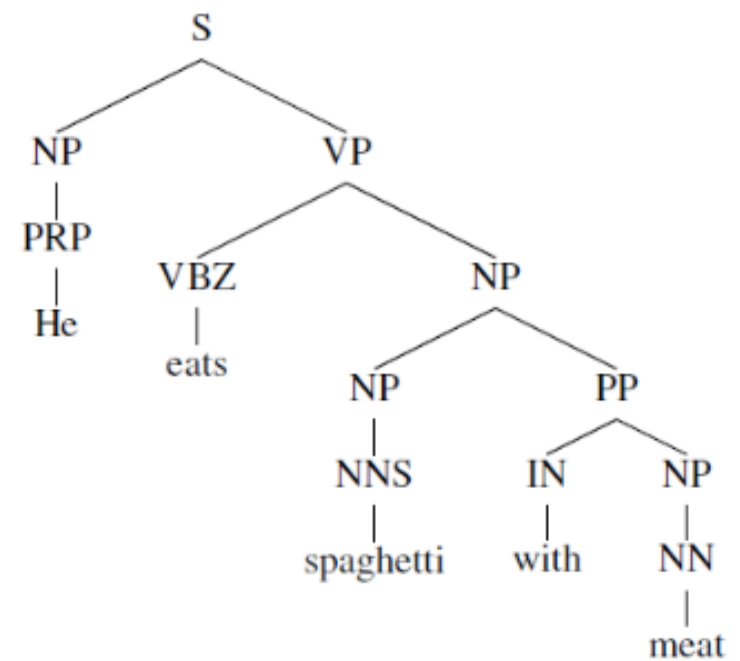
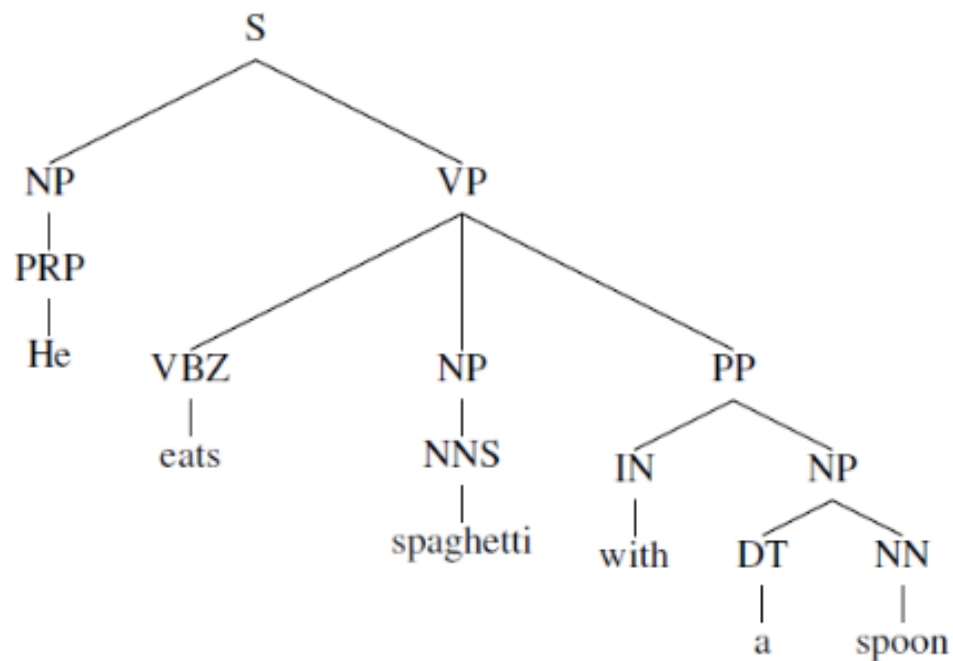
Recursive vs Recurrent representation



Which to use is still up for debate

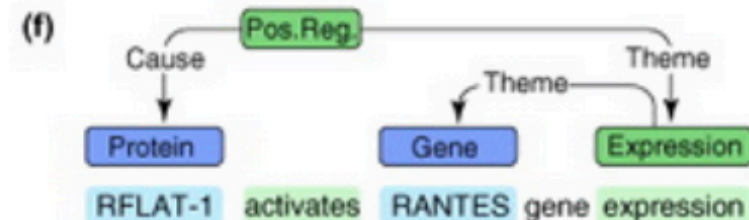
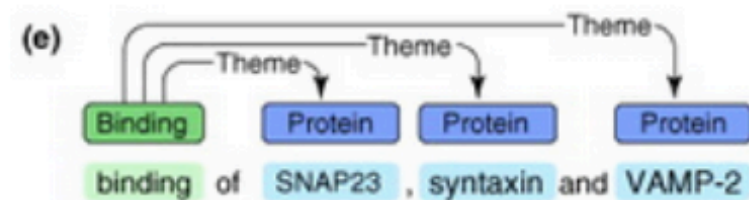
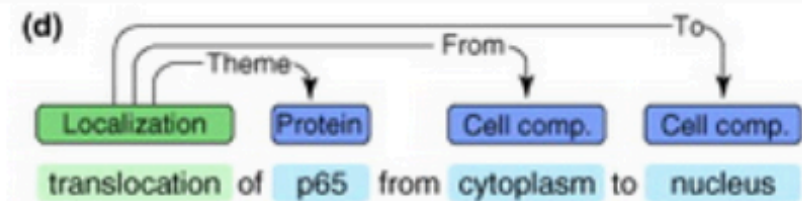
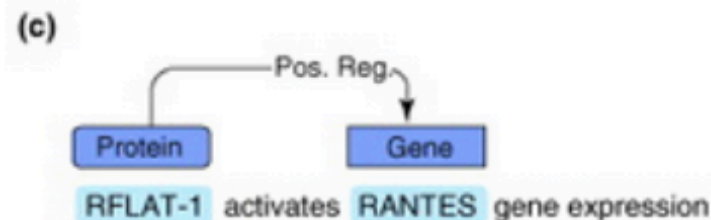
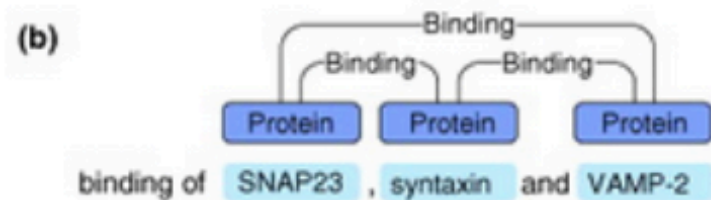
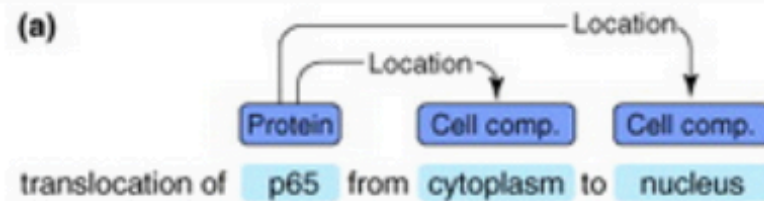


Cases where recursive might be better?

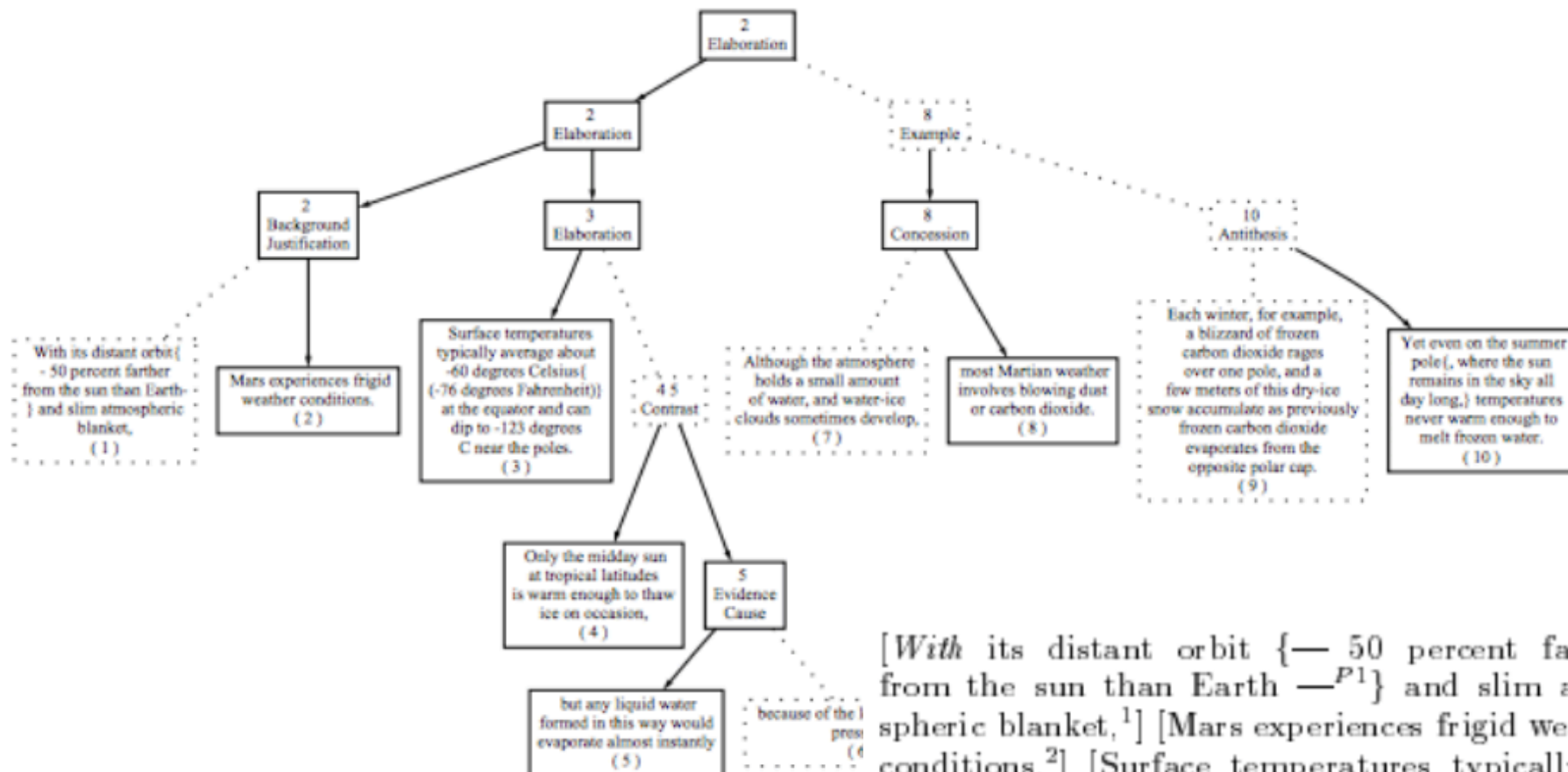


But recurrent structures might be able to learn this too...

Application of parse trees: info extraction



A side note: discourse parsing



[With its distant orbit {— 50 percent farther from the sun than Earth —^{P1}} and slim atmospheric blanket,¹] [Mars experiences frigid weather conditions.²] [Surface temperatures typically average about -60 degrees Celsius (-76 degrees Fahrenheit) at the equator and can dip to -123 degrees C near the poles.³] [Only the midday sun at tropical latitudes is warm enough to thaw ice

Overview

- Types of grammars
 - Context Free Grammar
 - Probabilistic Context Free Grammar
 - CYK parser
 - Dependency Grammar
 - Transition-based parsing
 - Recursive neural networks

Constituents

- Groups of words behaving as a single units

- Ex: Noun phrase

Harry the Horse

The reason he comes into the house

They

A high-class spot such as Mindy's

- Checking for constituents

- See if they can appear in similar syntactic environments

They sit...

The reason he comes into the house is...

Context-Free Grammar (CFG)

- A grammar specifies what kind of parse tree can be generated.
- CFG or Phrase-Structure Grammars assumes the grammar is context-free
 - Most forms of natural language are context-free
 - Thai and English are typically CFG languages
 - Used in many programming languages
- CFG is based on constituents structures

A context-free grammar example

A CFG is defined by
 $G = (N, S, \Sigma, R)$

- $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$ Nonterminals
- $S = S$ Starting symbol
- $\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$

Terminals

$R =$

$S \Rightarrow NP VP$	$Vi \Rightarrow \text{sleeps}$
$VP \Rightarrow Vi$	$Vt \Rightarrow \text{saw}$
$VP \Rightarrow Vt NP$	$NN \Rightarrow \text{man}$
$VP \Rightarrow VP PP$	$NN \Rightarrow \text{woman}$
$NP \Rightarrow DT NN$	$NN \Rightarrow \text{telescope}$
$NP \Rightarrow NP PP$	$DT \Rightarrow \text{the}$
$PP \Rightarrow IN NP$	$IN \Rightarrow \text{with}$
	$IN \Rightarrow \text{in}$

Production rules

$X \rightarrow Y_1 \dots Y_n$

Y_i can be terminal or nonterminal

The rule only relies on X (no context)
 (vs context-sensitive)

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

Generation using left-most derivation strategy

- Recursive strategy

```
left_most_derivation(A):  
    if terminal(A):  
        return A          Return if terminal  
    rule = choose (rules(A))  
    rhs = right_hand_side(rule)  
    return concatenate([left_most_derivation(A') for A' in rhs])
```

$R =$

S	\Rightarrow	NP	VP
VP	\Rightarrow	Vi	
VP	\Rightarrow	Vt	NP
VP	\Rightarrow	VP	PP
NP	\Rightarrow	DT	NN
NP	\Rightarrow	NP	PP
PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

- To generate we call `left_most_derivation(S)` (top down)
- A string belongs to the language of a CFG if there exist a sequence of left-most derivation that can generate the string

$$L = \{s \in \Sigma^* | s = \text{left_most_derivation}(S)\}$$

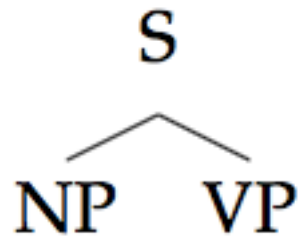
Example

- The woman saw the man with the telescope

$R =$	S	\Rightarrow	NP	VP
	VP	\Rightarrow	Vi	
	VP	\Rightarrow	Vt	NP
	VP	\Rightarrow	VP	PP
	NP	\Rightarrow	DT	NN
	NP	\Rightarrow	NP	PP
	PP	\Rightarrow	IN	NP
	Vi	\Rightarrow	sleeps	
	Vt	\Rightarrow	saw	
	NN	\Rightarrow	man	
	NN	\Rightarrow	woman	
	NN	\Rightarrow	telescope	
	DT	\Rightarrow	the	
	IN	\Rightarrow	with	
	IN	\Rightarrow	in	

Example

- The woman saw the man with the telescope

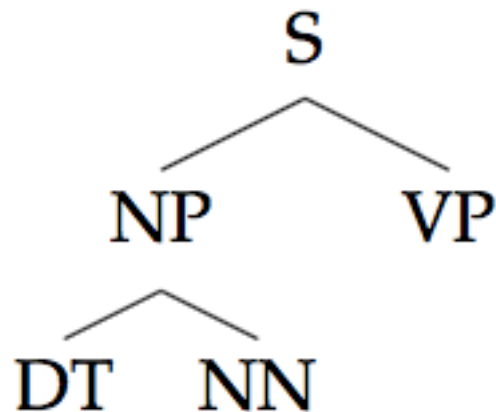


$R =$	S	\Rightarrow	NP	VP
	VP	\Rightarrow	Vi	
	VP	\Rightarrow	Vt	NP
	VP	\Rightarrow	VP	PP
	NP	\Rightarrow	DT	NN
	NP	\Rightarrow	NP	PP
	PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

Example

- The woman saw the man with the telescope



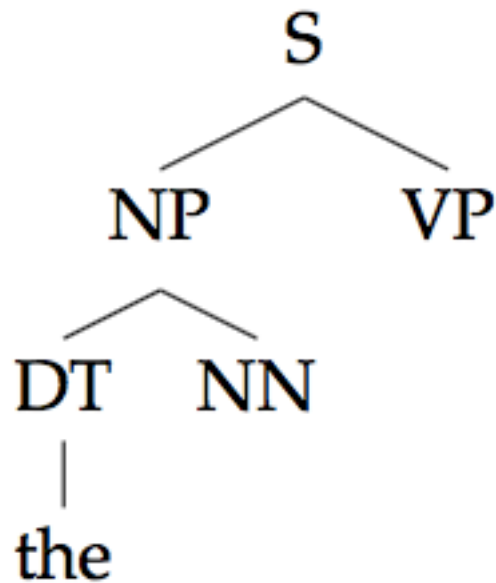
$R =$

S	\Rightarrow	NP	VP
VP	\Rightarrow	Vi	
VP	\Rightarrow	Vt	NP
VP	\Rightarrow	VP	PP
NP	\Rightarrow	DT	NN
NP	\Rightarrow	NP	PP
PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

Example

- The woman saw the man with the telescope



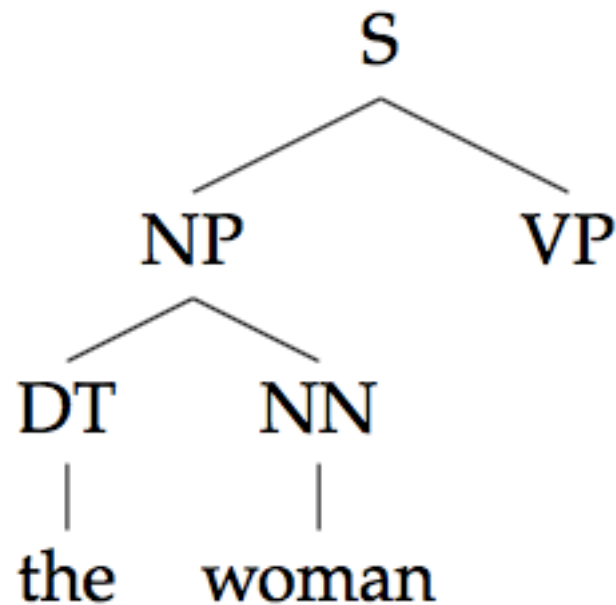
$R =$

S	\Rightarrow	NP	VP
VP	\Rightarrow	Vi	
VP	\Rightarrow	Vt	NP
VP	\Rightarrow	VP	PP
NP	\Rightarrow	DT	NN
NP	\Rightarrow	NP	PP
PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

Example

- The woman saw the man with the telescope



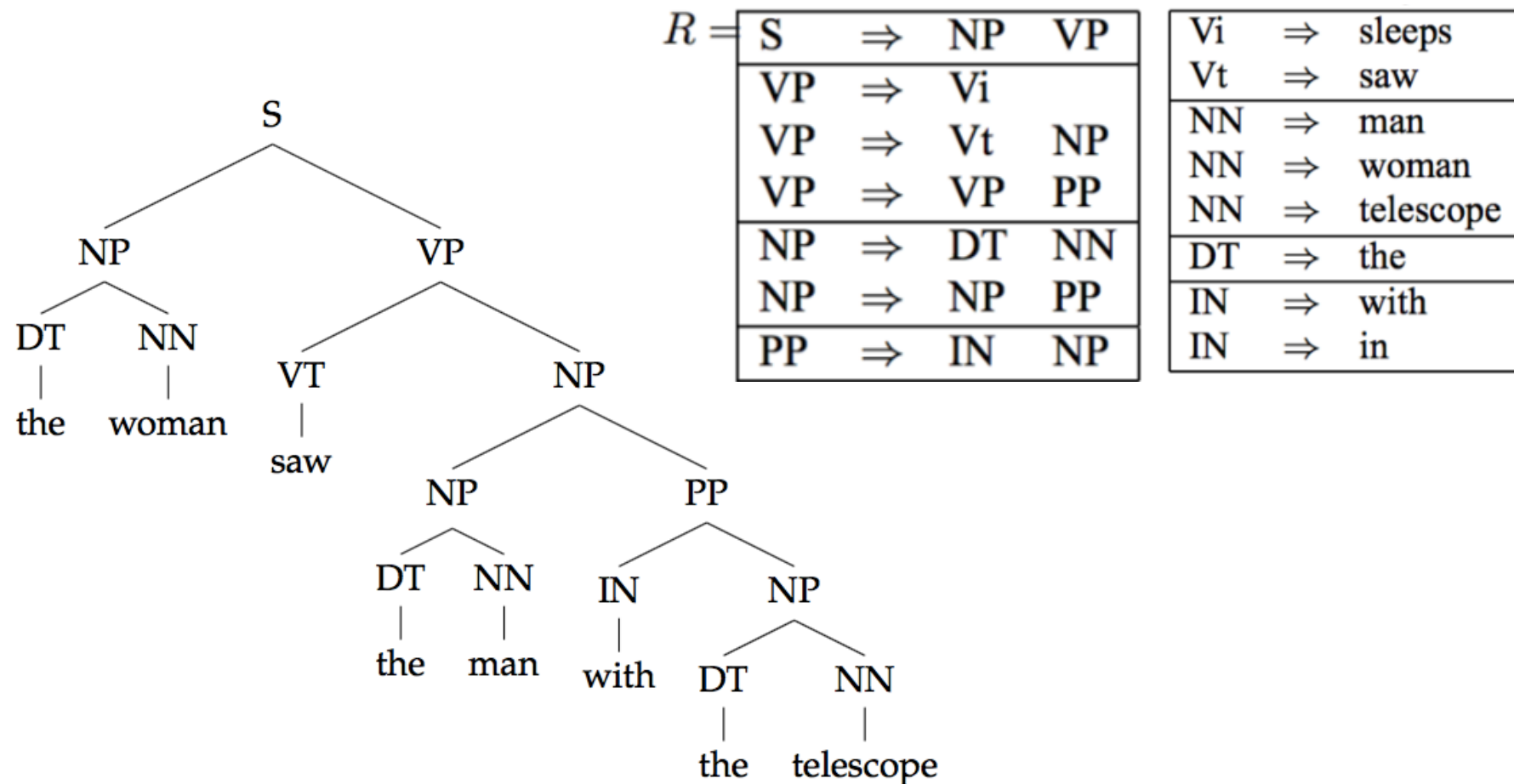
$R =$

S	\Rightarrow	NP	VP
VP	\Rightarrow	Vi	
VP	\Rightarrow	Vt	NP
VP	\Rightarrow	VP	PP
NP	\Rightarrow	DT	NN
NP	\Rightarrow	NP	PP
PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

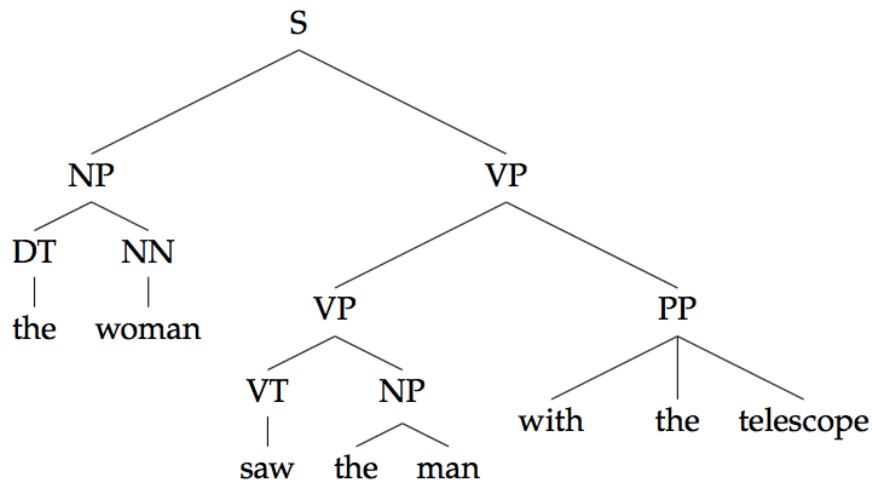
Example

- The woman saw the man with the telescope

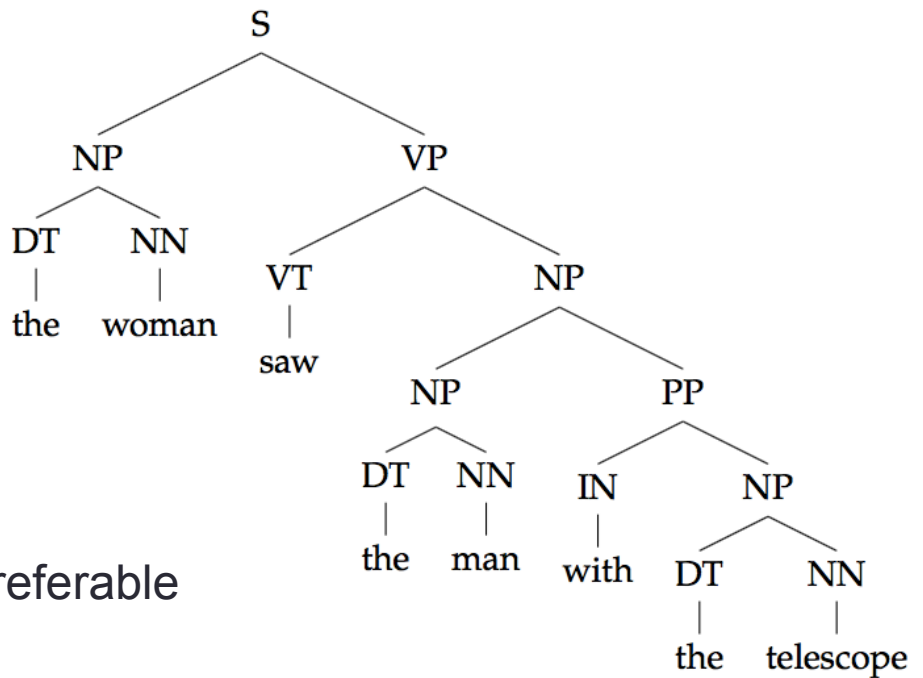


Ambiguities

- There can be multiple derivations for the same string
- These sentences are ambiguous as each parse represents a different meaning



Need a way to say which one is more preferable
Add probabilities!



Probabilistic Context-Free Grammar

- Production rules now have **probabilities**

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

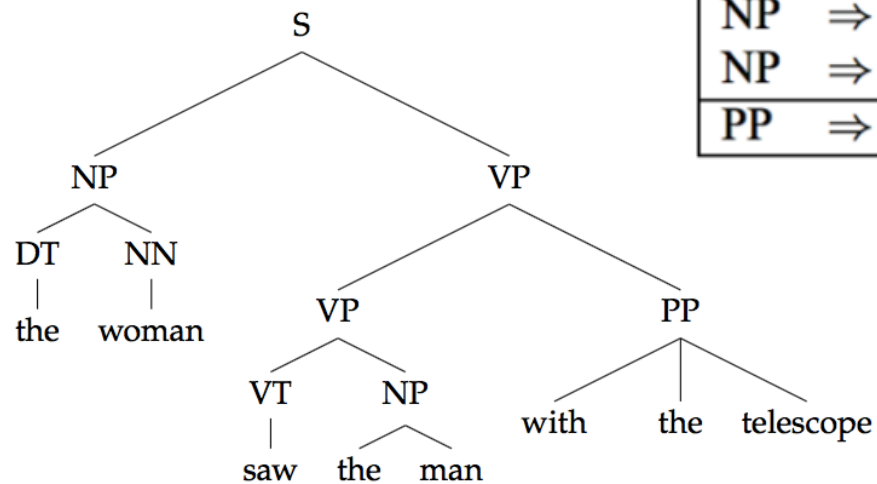
The probability of a (sentence, parse tree) pair is
$$p(S, T) = \prod_{i=1}^m p(\alpha_i \rightarrow \beta_i | \alpha_i)$$

m is the number of transitions

$$P(\text{NN} \rightarrow \text{man} \mid \text{NN}) = 0.7$$

P(S,T)

• P(S,T) =



S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

Estimating transition probabilities

- Counts from training set!

$$P(S \rightarrow NP VP | S) = \frac{\text{count}(S \rightarrow NP VP)}{\text{count}(S)}$$

PCFG related questions

- What is the most likely parse? (**parsing task**)
 - $\operatorname{argmax}_T P(T,S)$
- What is the probability of the sentence, $P(S)$? (**Language modeling task**)
 - $P(S) = \sum_T P(T,S)$

How?

Dynamic programming

(CYK algorithm –

Cocke–Younger–Kasami algorithm)

Chomsky Normal Form

- CYK can be used if the CFG is in Chomsky Normal Form
- A CFG is in Chomsky Normal Form if each rule either converts to **two nonterminals** or a **single terminal**.

$$X \rightarrow Y_1 Y_2$$

Uppercase letters mean nonterminal
Lowercase letters mean terminal

$$X \rightarrow y$$

- Any CFG can be converted to CNF

- For example $NP \rightarrow DT, ADJ, NN$
 - Turns into two rules
 - $NP \rightarrow DT, ADJP$
 - $ADJP \rightarrow ADJ, NN$

• $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$ **Nonterminals**

• $S = S$ **Starting symbol**

• $\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$

Terminals

$$R =$$

S	\Rightarrow	NP	VP
VP	\Rightarrow	Vi	
VP	\Rightarrow	Vt	NP
VP	\Rightarrow	VP	PP
NP	\Rightarrow	DT	NN
NP	\Rightarrow	NP	PP
PP	\Rightarrow	IN	NP

Vi	\Rightarrow	sleeps
Vt	\Rightarrow	saw
NN	\Rightarrow	man
NN	\Rightarrow	woman
NN	\Rightarrow	telescope
DT	\Rightarrow	the
IN	\Rightarrow	with
IN	\Rightarrow	in

Production rules

$X \rightarrow Y_1 \dots Y_n$

Y_i can be terminal or nonterminal

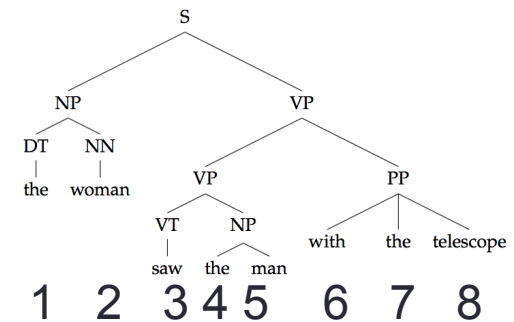
The rule only relies on X (no context)
(vs. context-sensitive)

CYK algorithm for parsing

3 dimensional

- $\pi(i, j, N)$ probability that words i to j can be generated by nonterminal N

- Base case: $\pi(i, i, N) = P(N \rightarrow w_i | N)$



- Inductive case:

k is the index that splits the subsentence

$$\pi(i, j, N) = \max_{k, P, Q} Pr(N \rightarrow P Q | N) \cdot \pi(\underline{i, k}, P) \cdot \pi(\underline{k+1, j}, Q)$$

where $k \in \{i, \dots, j-1\}$, $P \in \mathcal{N}$, and $Q \in \mathcal{N}$.

Saves the rule that gave max probability to backtrack

CYK algorithm for language modeling

- $\pi(i, j, N)$ probability that words i to j can be generated by nonterminal N

- Base case: $\pi(i, i, N) = P(N \rightarrow w_i | N)$

- Inductive case: $\pi(i, j, N) = \max_{k, P, Q} Pr(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$
 $\text{argmax}_T P(T, S) \text{ vs } P(S) = \sum_T P(T, S)$

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(\underline{i, k}, P) \cdot \pi(\underline{k + 1, j}, Q)$$

where $k \in \{i, \dots, j - 1\}$, $P \in \mathcal{N}$, and $Q \in \mathcal{N}$.

Language modeling example CYK with PCFG

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$\pi(i, j, N)$

A

a			
b			
c			

B

a			
b			
c			

Base case

- $N = \{A, B\}$
 - $\Sigma = \{a, b, c\}$
 - $S = \{A\}$
- Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, i, N) = P(N \rightarrow w_i | N)$$

A

a	0.2		
b		0	
c			0

B

a	0		
b		0.1	
c			0.2

1st step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2		
b		0	
c			0

B

a	0		
b		0.1	
c			0.2

Only consider i, j where $i < j$

1st step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2	0.016	
b		0	
c			

$$\begin{aligned}
 \pi[1, 2, A] &= P(A \rightarrow AA) \cdot P(A \rightarrow a | A) \cdot P(A \rightarrow b | A) \\
 &\quad + P(A \rightarrow AB) \cdot P(A \rightarrow a | A) \cdot P(B \rightarrow b | B) \\
 &\quad + P(A \rightarrow BA) \cdot P(B \rightarrow a | B) \cdot P(A \rightarrow b | A) \\
 &\quad + P(A \rightarrow BB) \cdot P(B \rightarrow a | B) \cdot P(B \rightarrow b | B) \\
 &= 0 + P(A \rightarrow AB) \cdot \pi[1, 1, A] \cdot \pi[2, 2, B] + 0 + 0 \\
 &= 0.8 \cdot 0.2 \cdot 0.1 = 0.016
 \end{aligned}$$

1st step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2	0.016	
b		0	
c			0

B

a	0	0	
b		0.1	
c			0.2

2nd step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2	0.016	
b		0	0
c			0

B

a	0	0	
b		0.1	0.014
c			0.2

Finish

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2	0.016	0.0048
b		0	0
c			0

B

a	0	0	0
b		0.1	0.014
c			0.2

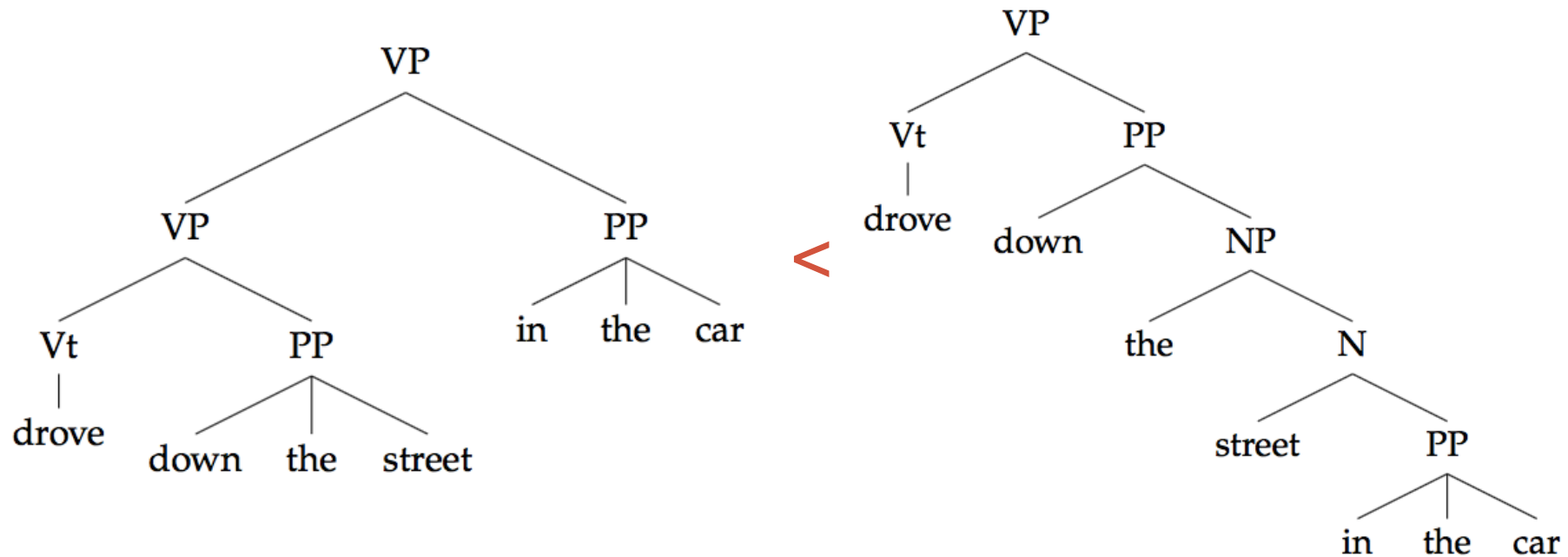
Fill order

PCFG weakness

- Lack of sensitivity to lexical info (does not consider semantics)
- Lack of sensitivity to structural frequency

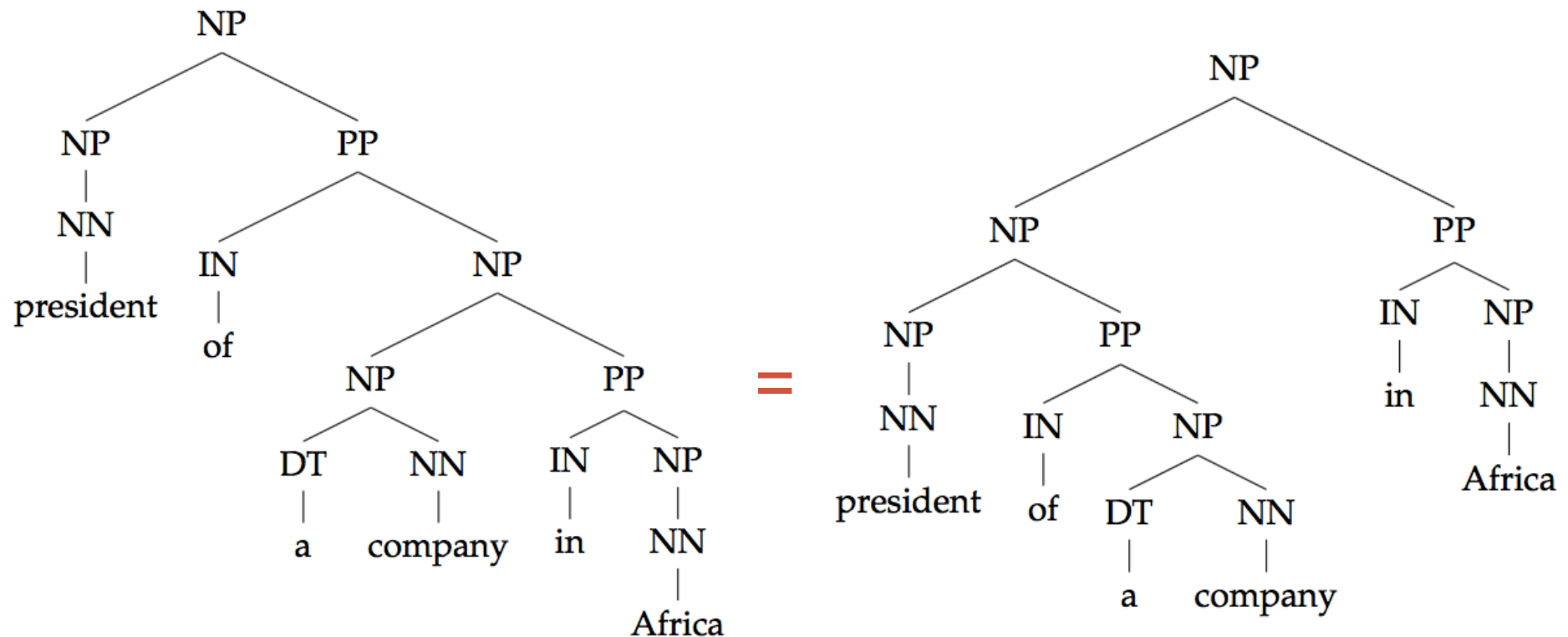
Lack of lexical info

The probabilities only see terminals and expansions



The street in the car

Lack of sensitivity to structural frequency



Both trees have same expansion and so same probability

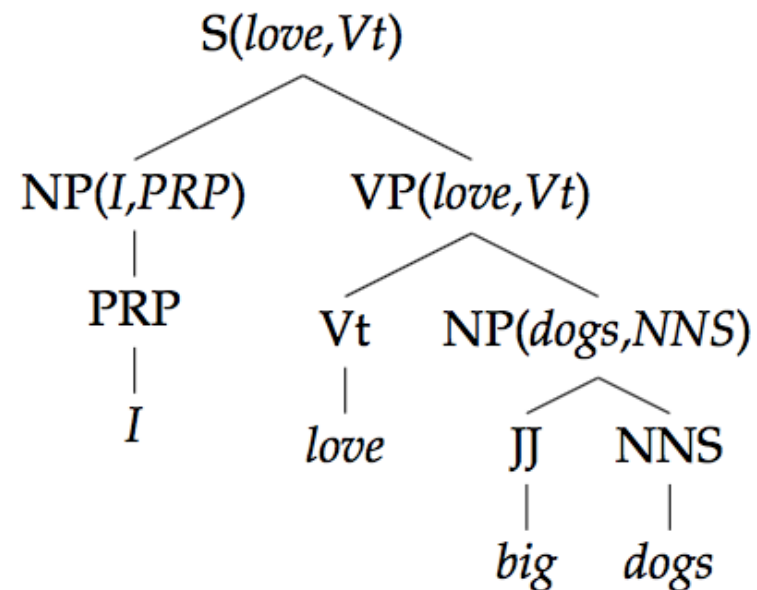
Turns out, left hand side structure appears more frequently in the training set

Methods to improve PCFG

- Add lexical information to the tree
 - Label each node with associate words
 - Lexicalized trees
 - Latent variable grammars
- Look at bigger portions of the tree at a time

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5



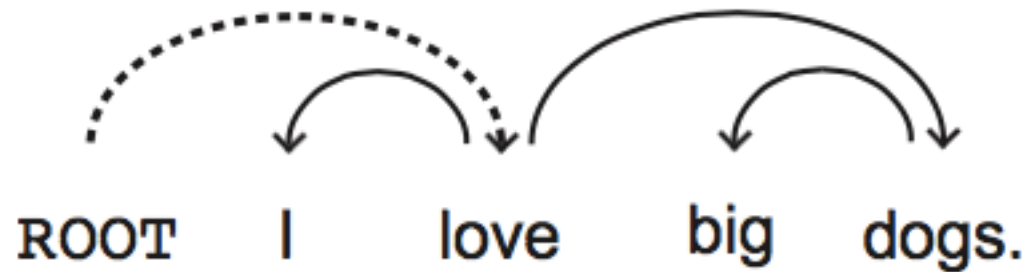
Lexicalized tree

Dependency grammar

- CFG is based on constituency relation
- In dependency grammar the structure is composed of lexical items (words) linked by edges to form a tree
- Assumptions
 - Each words in a sentence is related or modifies another word
 - All words have a direct or indirect relation to the main verb

Example

- Add ROOT node as the root of the tree
- The main verb always point to ROOT



A -> B means
A governs B or
B depends on A
A is the head of B

- Each arc can have a category for the relationship.
- Each word can have a PoS label

Constituency structures vs dependency structures

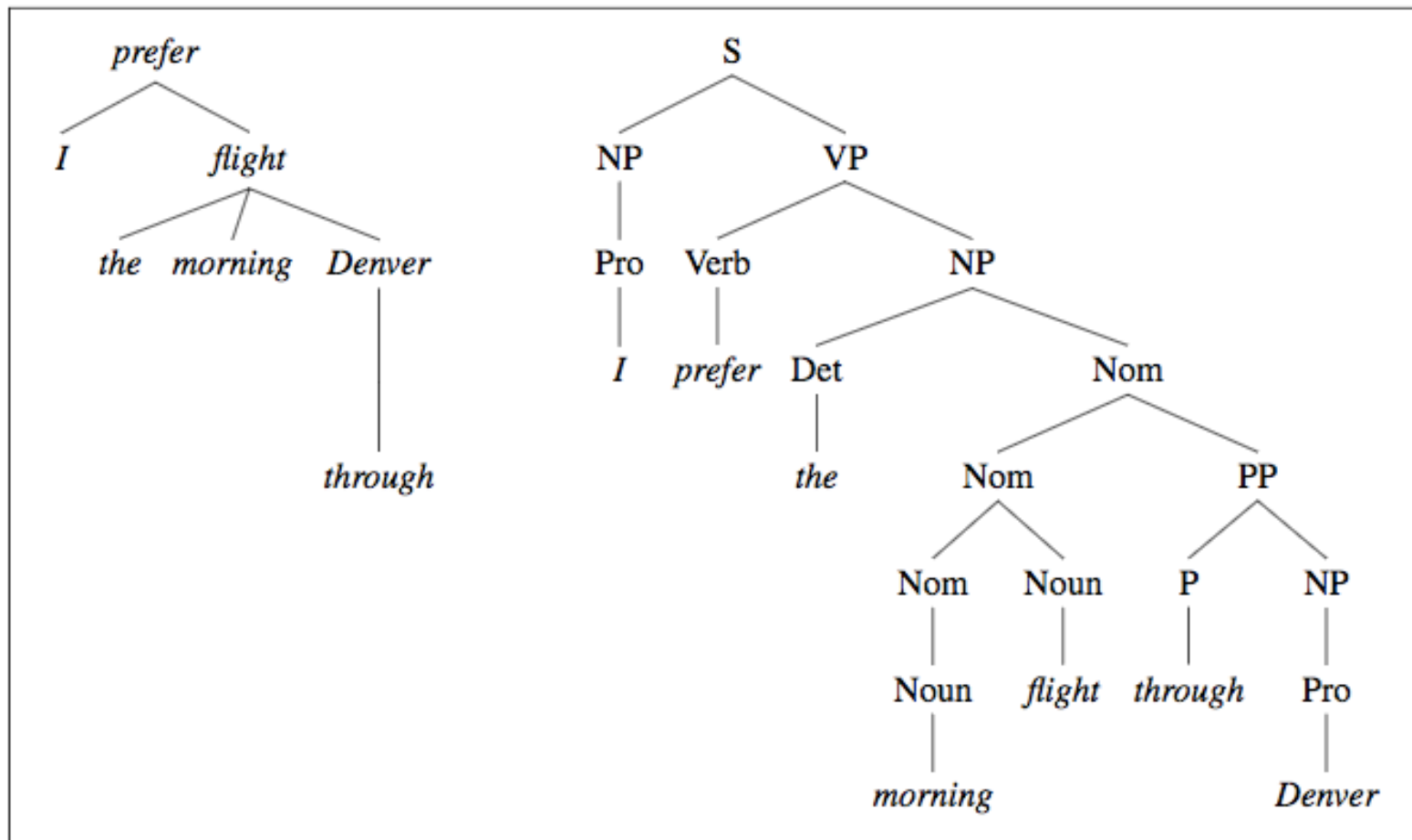


Figure 14.1 A dependency-style parse alongside the corresponding constituent-based analysis for *I prefer the morning flight through Denver*.

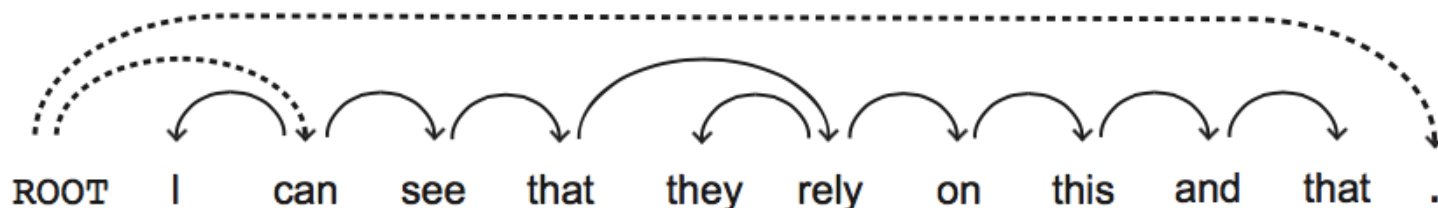
Constituency structures vs dependency structures

- Constituency structures use more nodes to represent sentences at different levels.
- Constituency structures explicitly label non-terminal nodes (NP vs VP)
- Constituency structures encode more info than dependency structures
- You can convert constituency structures to dependency structures
 - Dependency parsers trained on this is usually better than dependency parsers trained on original dependency structures

Criteria for heads (basics)

- Head, H. Dependent D
- D modifies H
 - Big (D) dogs (H), willow (D) tree (H)
- H can often replace D
 - I love big (D) dogs (H) -> I love dogs
- H is obligatory while D sometimes is optional
- H determines whether D is obligatory
 - Sarah sneezed (H) vs George kicks (H) the chair (D)
- More criterias! **Mostly depends on corpus**

Ex



Dependency and meaning

- Scientists study whales from space
- Scientists study whales from space

Sample Thai sentences

- วัน หนึ่ง เกิด เหตุ ที่ ไม่ คาดคิด ขึ้น มา
- การ ที่ เรา จะ รู้ ถึง สิ่ง เหล่านั้น ได้ ก็ ต้อง อาศัย ปัจจัย หลาย ๆ อย่าง ประกอบ เข้า ด้วยกัน

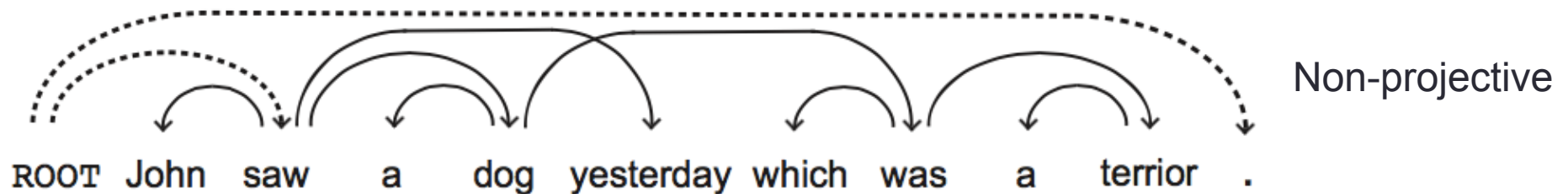
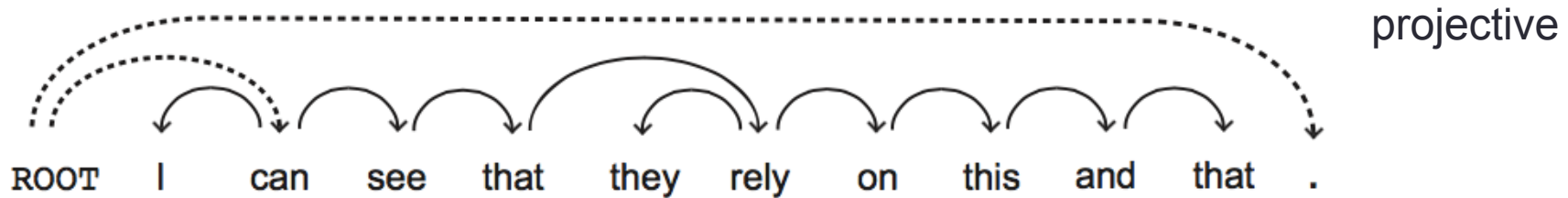
Note these criteria depends on the corpus convention

Dependency graph requirements

- Syntactic structure is complete (connectedness, spanning)
- Hierarchical (acyclic)
- Every word has a single head

Projectivity

- A dependency graph is projective if the arcs do not cross



English and Thai are mostly projective.

Some languages are more non-projective than others,
for example German, Dutch, Czech.

When picking a parser algo, check whether it assumes projectivity

Transition-based parsing (Nivre 2007)

- Use a stack and buffer data structure and sequentially add edges
- Characteristics
 - Greedy algo. Only goes left to right. No backtracking
 - Fast $O(n)$
 - Requires projective
 - The algo is closely related to how human parse sentences (left to right one word at a time instead of looking at the sentence as a whole)

Arc-standard Transition-based parsing

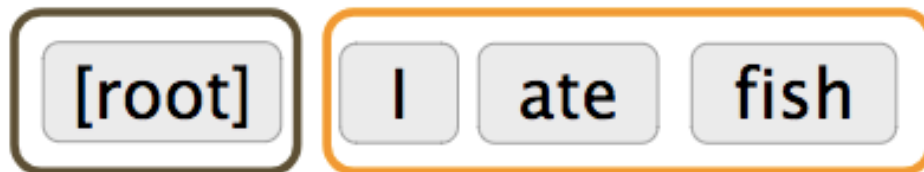
- A stack σ , written with top of the stack to the right
 - Starts with the ROOT symbol
- A buffer β , written with top to the left
 - Starts with the input sentence
- A set of dependency arcs A
 - Starts of empty

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

- A set of actions
 1. Shift $\sigma, w_i | \beta, A \Rightarrow \sigma | w_i, \beta, A$ Move buffer to stack
 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 $\sigma | w_i | w_j, \beta, A \Rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$
- Finishes when β becomes empty

I ate fish

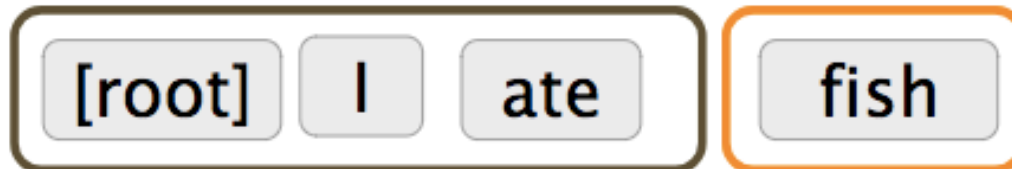
Start



Shift



Shift



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: $\beta = \emptyset$

I ate fish

Left Arc



Shift



Right Arc



Right Arc

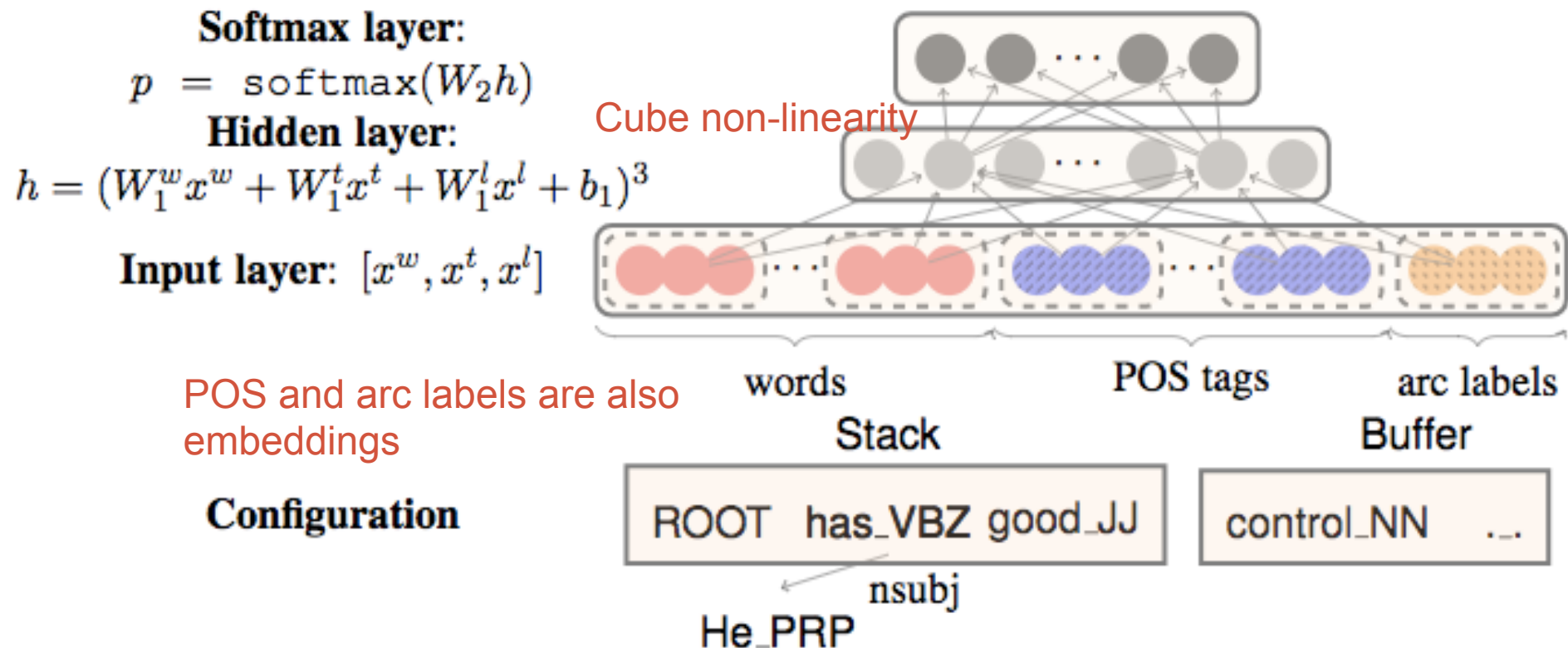


Discriminative parsing

- How to choose an action?
 - Shift, left-arc, right-arc
- Each action is predicted by a discriminative classifier (SVM, logistic regression, Neural networks) over legal moves
 - Features: top two word from stack, POS, children info; first word in buffer, POS, children info; etc.
- Greedy and no beamsearch
 - But you can include beamsearch (modern parsers do)

Discriminative parsing with neural networks

Shift, left, right (in actual, left/right + type of dependency)
So $2N+1$, where N = dependency types



Improvements

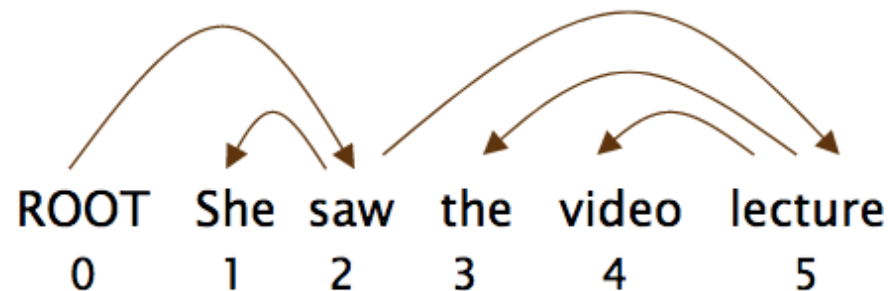
- Bigger networks
- Beam search
- Global inference over the sequence (CRF style)
- Lead to SyntaxNet and Parsey McParseFace model

<https://github.com/tensorflow/models/tree/master/research/syntaxnet>

<https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>

Parsing evaluation

- Labeled parsing accuracy (LAS)
- Unlabeled parsing accuracy (UAS)



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

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

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

Gold

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

Parsed

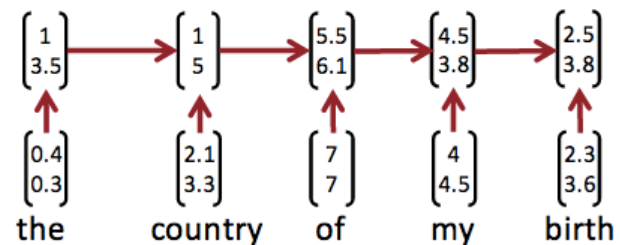
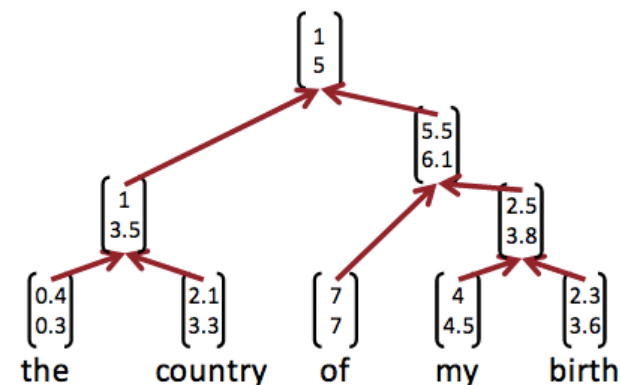
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

UAS of Parsey McParseFace

Model	News	Web	Questions
Martins et al. (2013)	93.10	88.23	94.21
Zhang and McDonald (2014)	93.32	88.65	93.37
Weiss et al. (2015)	93.91	89.29	94.17
Andor et al. (2016)*	94.44	90.17	95.40
Parsey McParseface	94.15	89.08	94.77

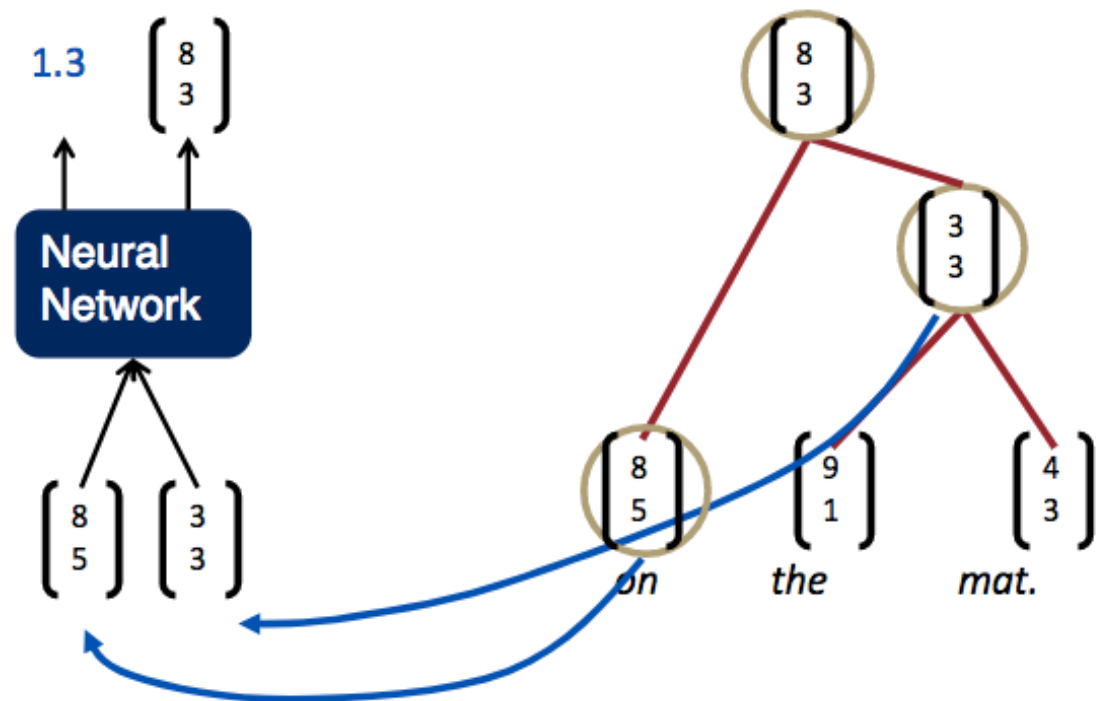
Recursive neural networks

- Not really used in parsing anymore but interesting concept
- Recursive vs Recurrent

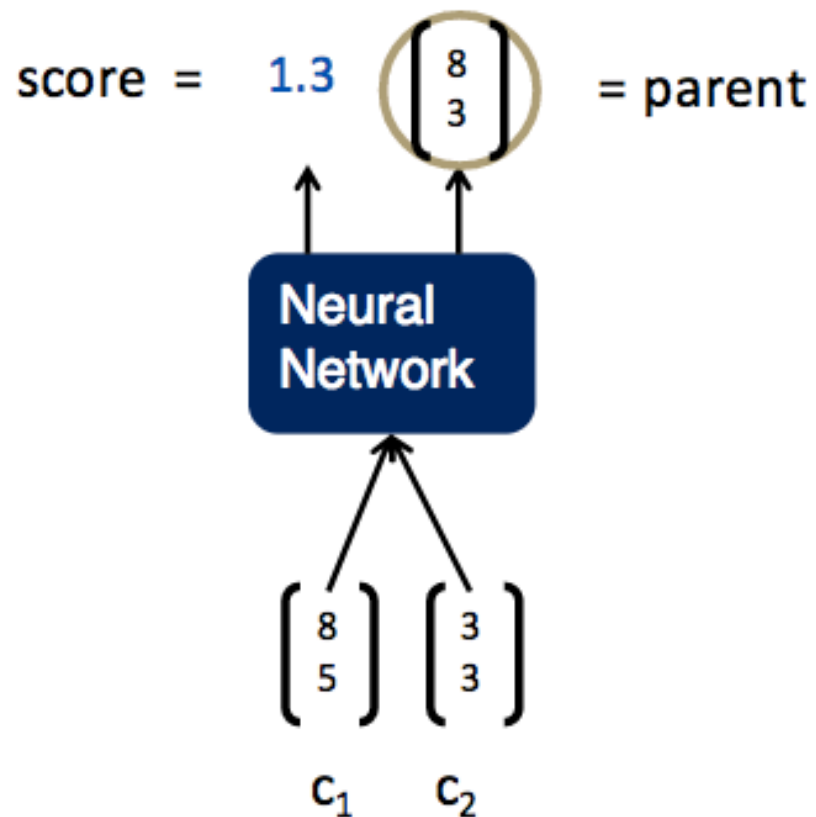


Recursive neural networks

- Concept: try different connections, see if which one gives the highest score (graph-based dependency parsers)
- Inputs: two candidate children representations
- Output:
 - Semantic representation of the parent
 - Score of new node



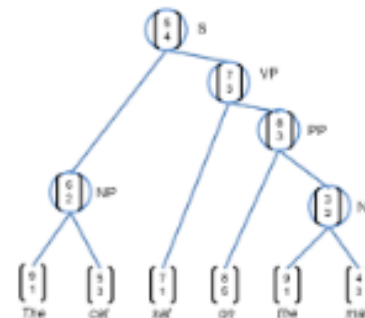
Shared recursive structure



$$\text{score} = U^T p$$

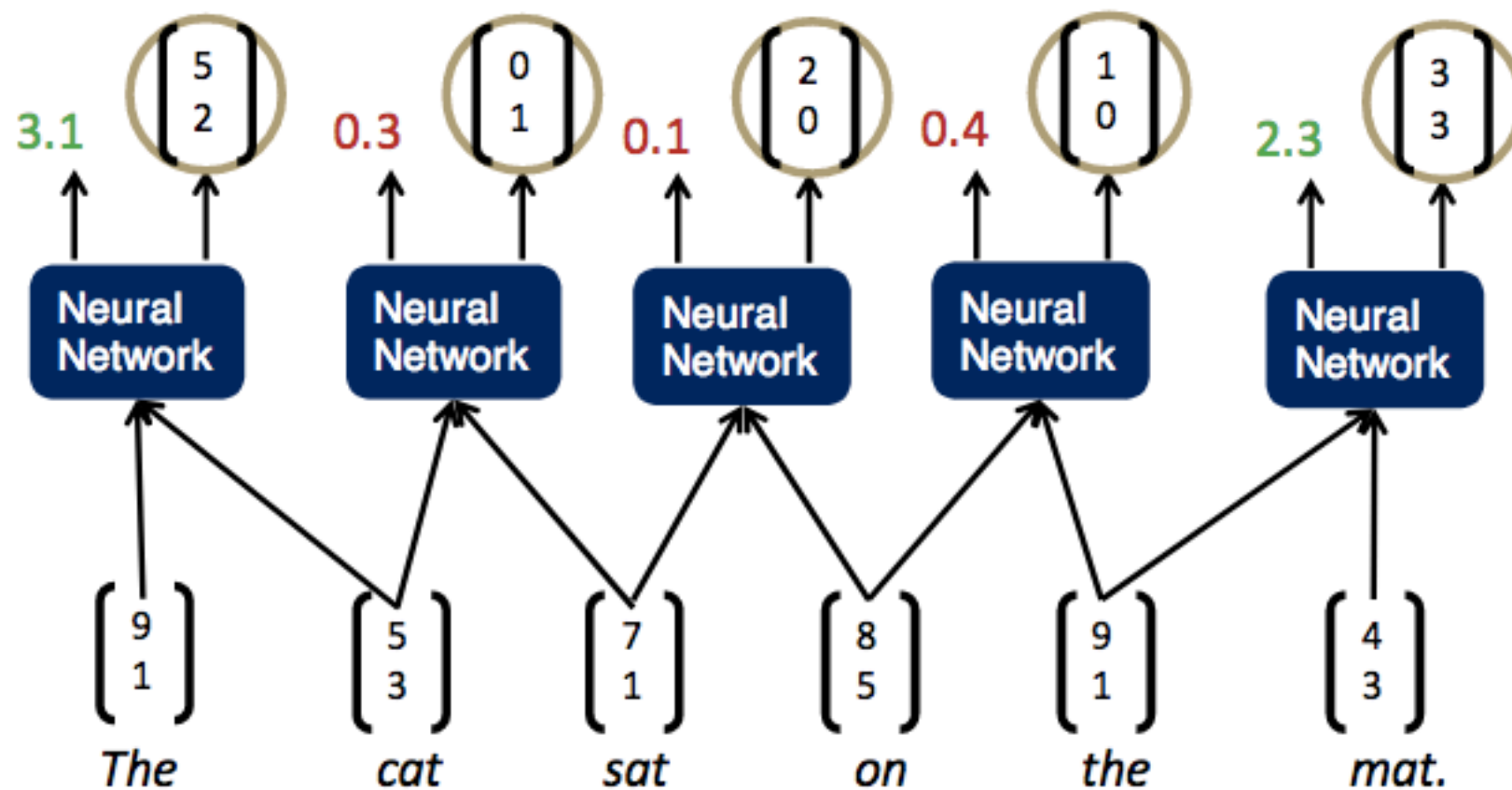
$$p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right),$$

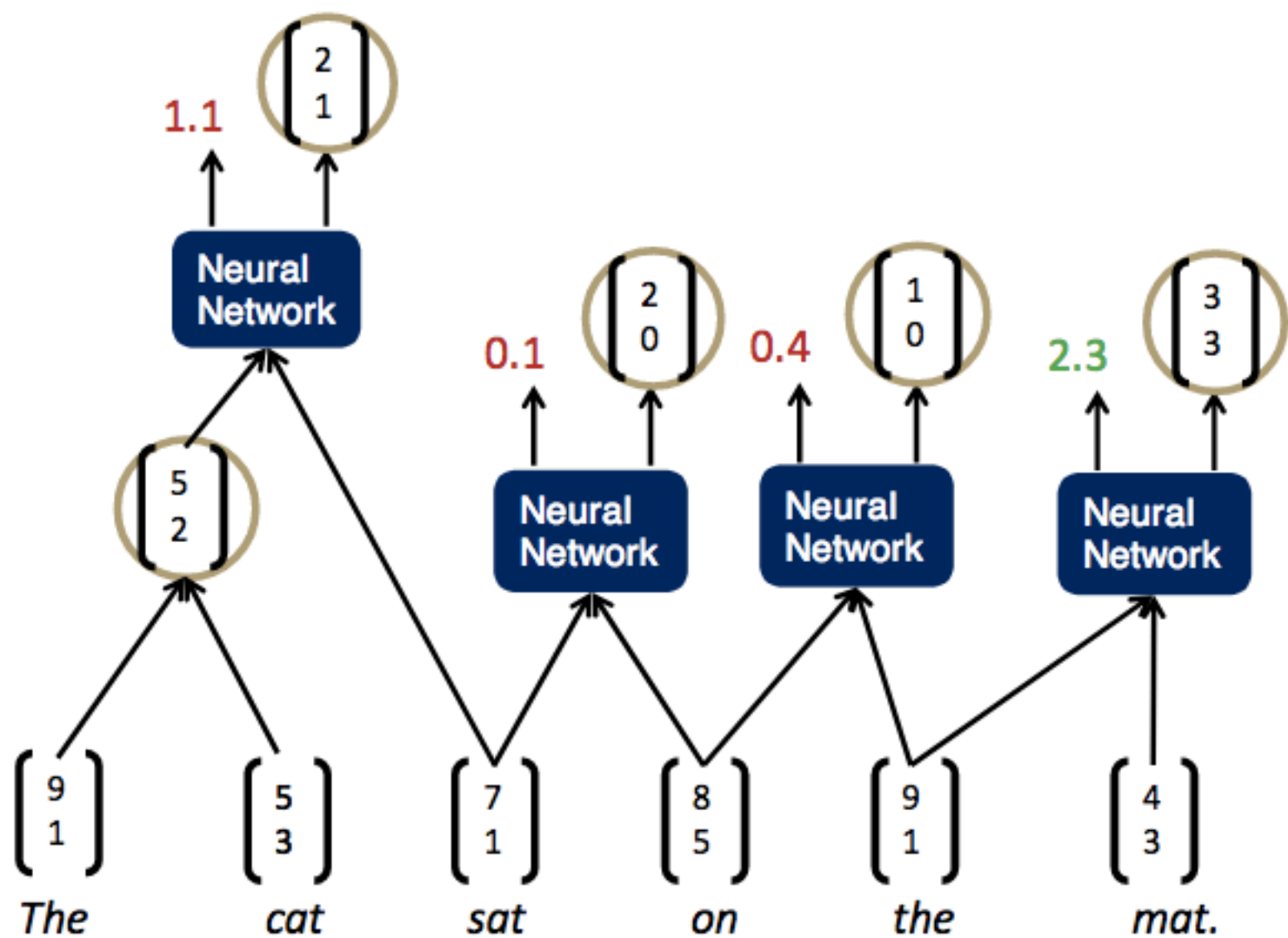
Same W parameters at all nodes of the tree

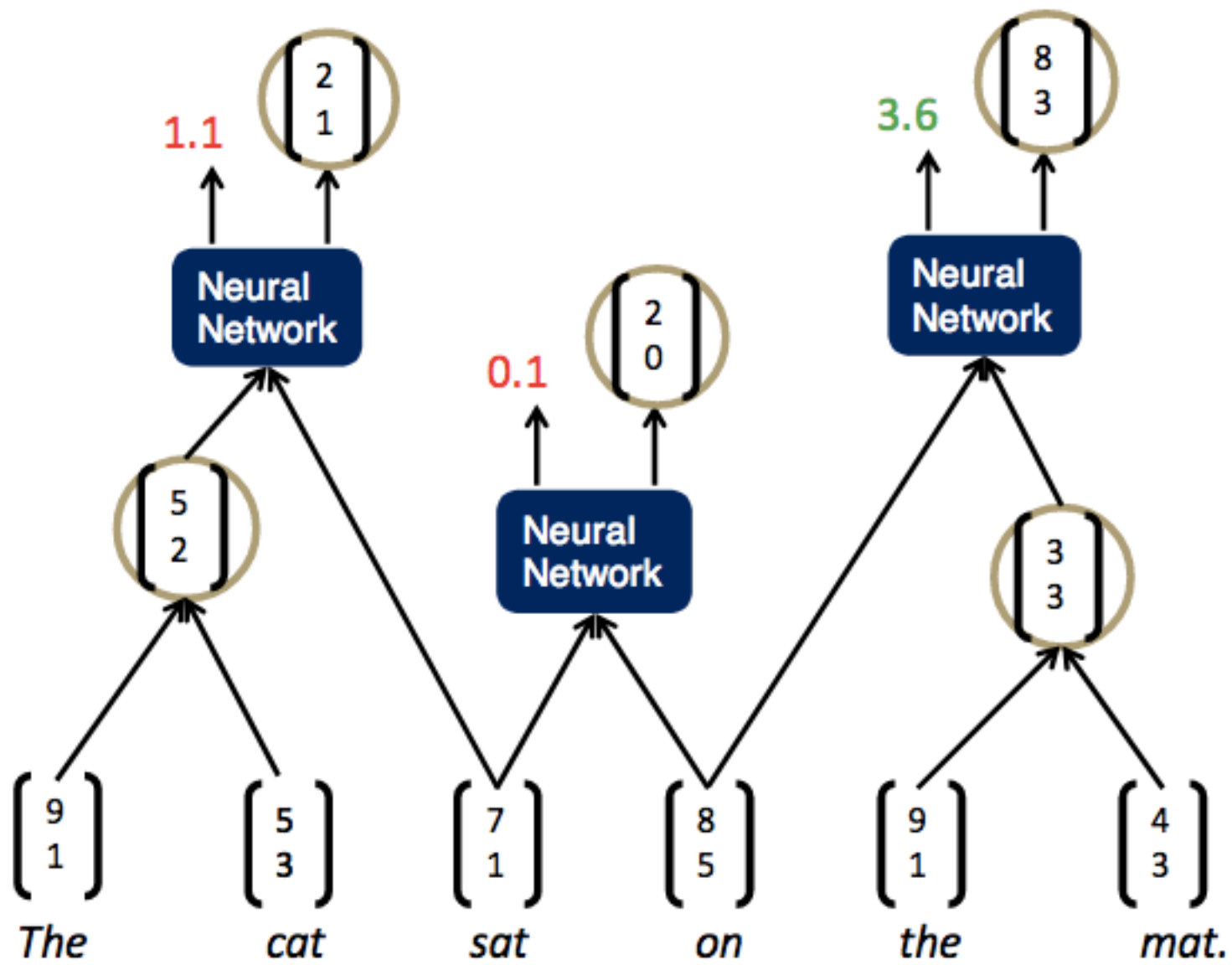


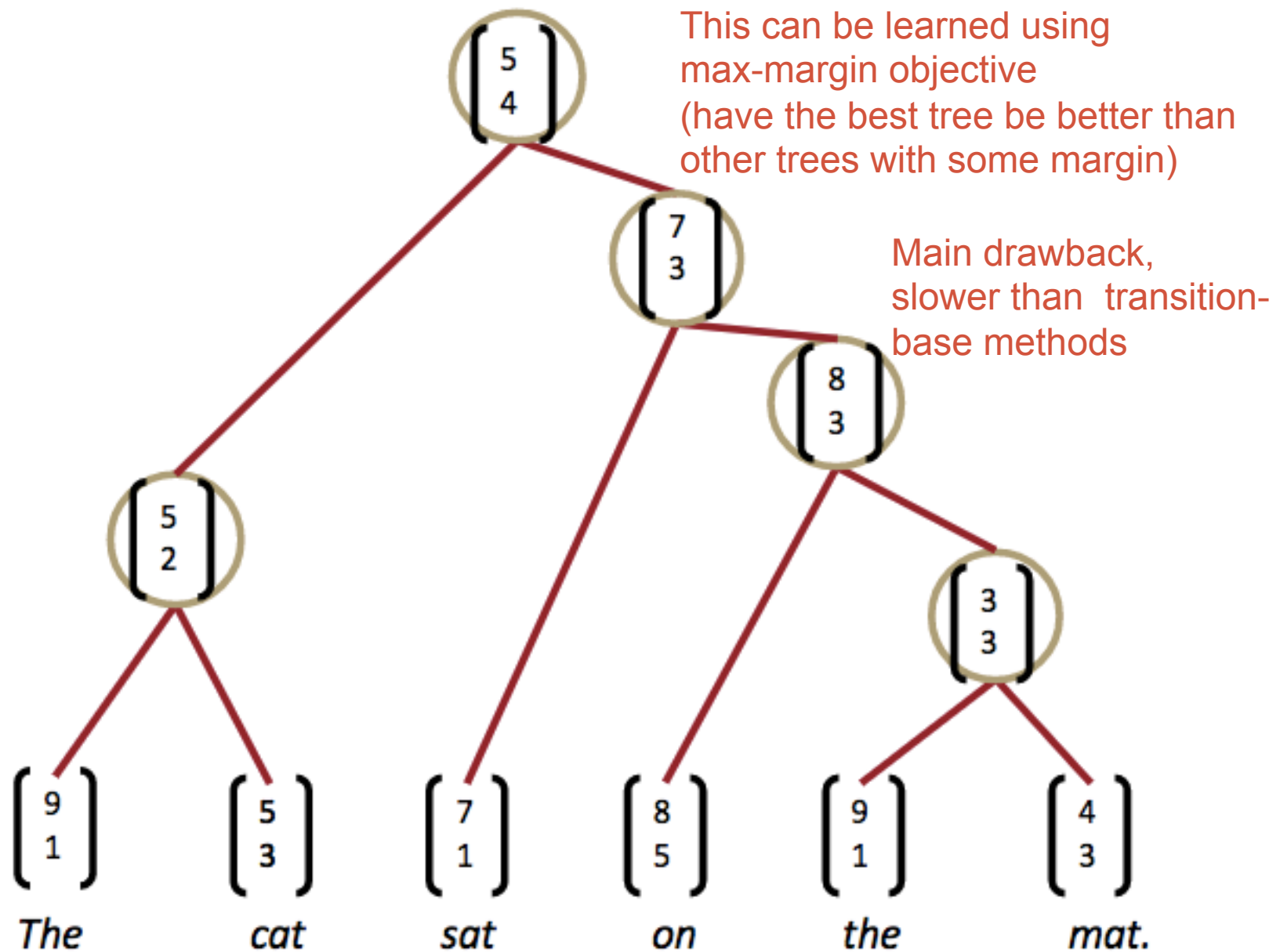
Parsing with recursive networks

Follow the highest scoring pair

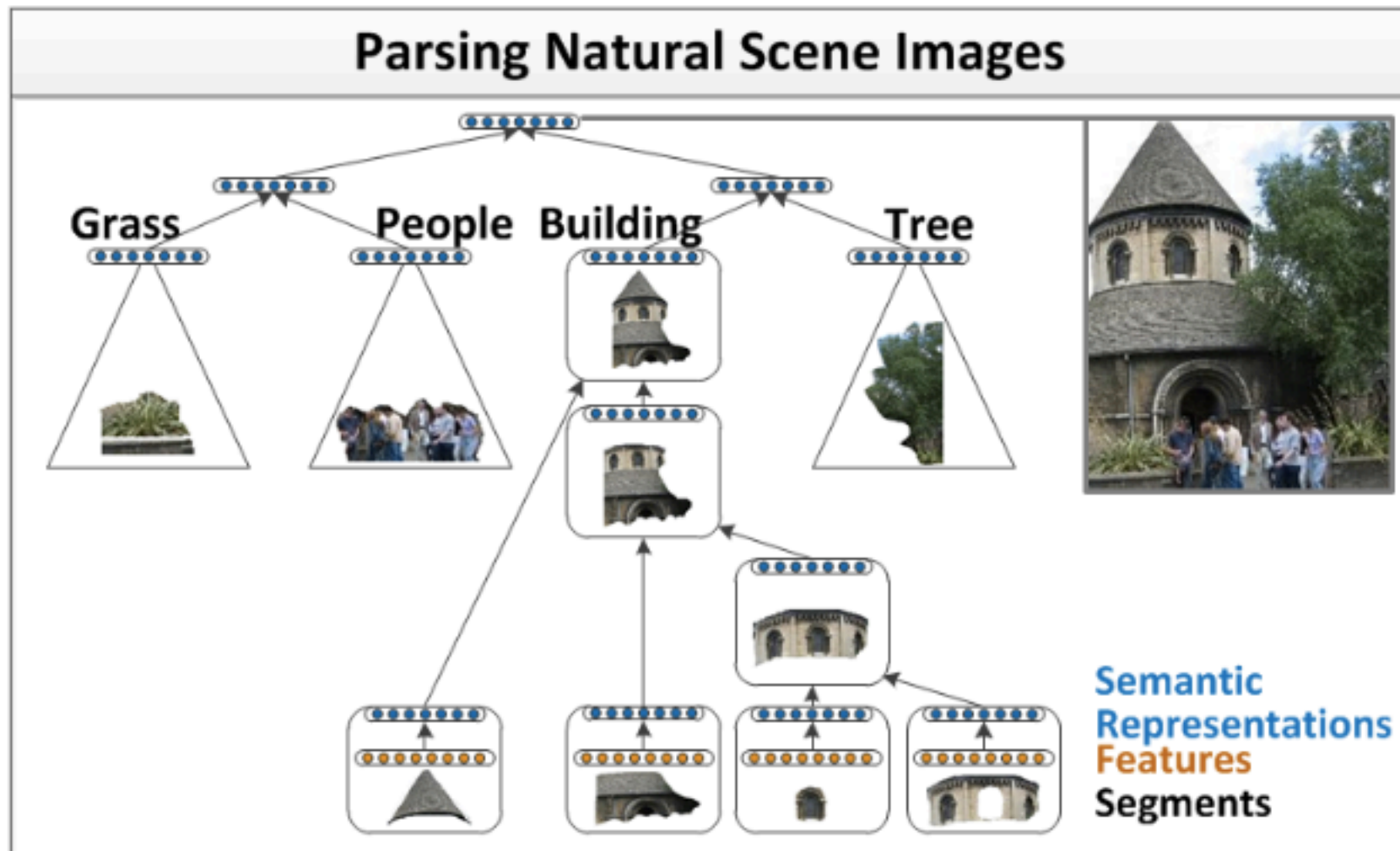










































Recursive neural networks for scene parsing

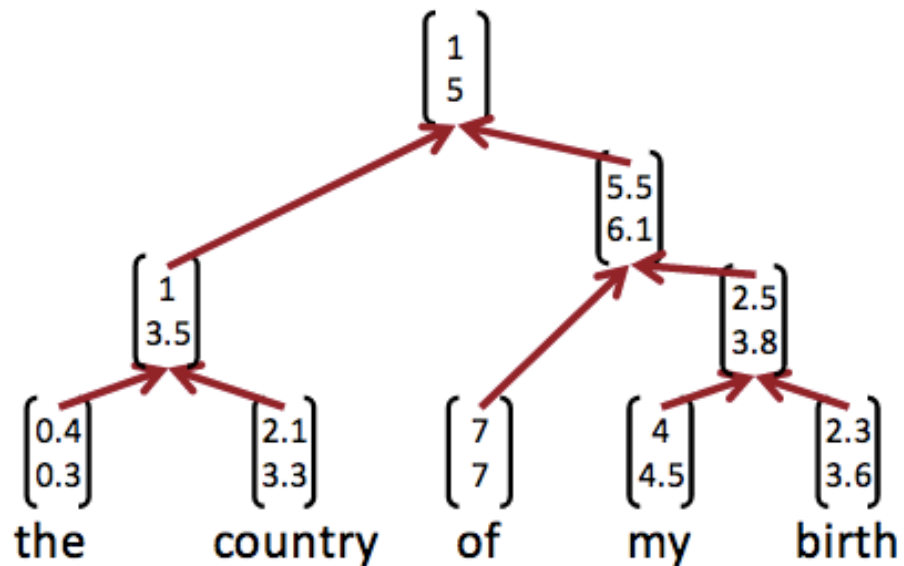


Resources for parsing

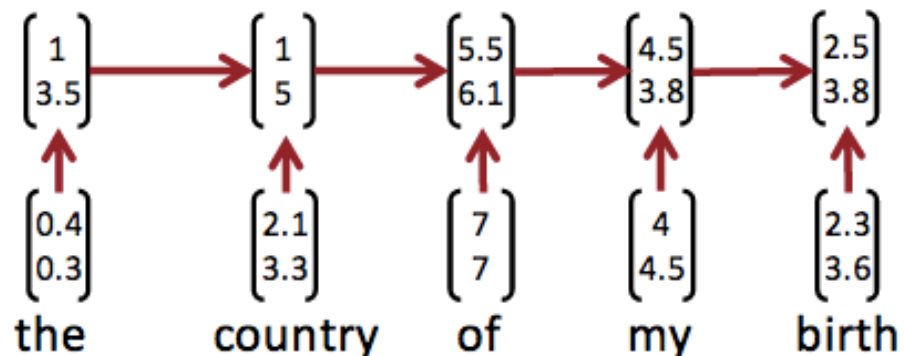
- Orchid & some corpus in development by NECTEC
- <http://universaldependencies.org/>

▶		Russian	3	1,226K		IE, Slavic
▶		Sanskrit	1	1K		IE, Indic
▶		Serbian	1	86K		IE, Slavic
▶		Slovak	1	106K		IE, Slavic
▶		Slovenian	2	170K		IE, Slavic
▶		Spanish	3	1,004K		IE, Romance
▶		Swedish	3	195K		IE, Germanic
▶		Swedish Sign Language	1	1K		Sign Language
▶		Tamil	1	9K		Dravidian, Southern
▶		Telugu	1	6K		Dravidian, South Central
▶		Thai	1	23K		Tai-Kadai
▶		Turkish	2	74K		Turkic, Southwestern
▶		Ukrainian	1	100K		IE, Slavic
▶		Upper Sorbian	1	10K		IE, Slavic
▶		Urdu	1	138K		IE, Indic
▶		Uyghur	1	15K		Turkic, Southeastern
▶		Vietnamese	1	43K		Austro-Asiatic

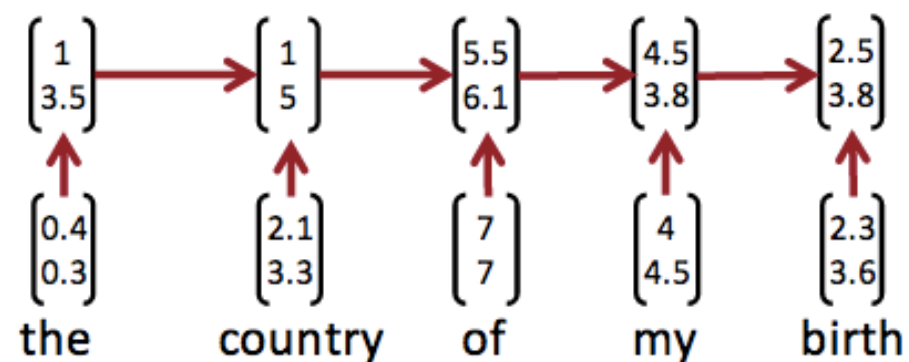
Recursive vs Recurrent representation



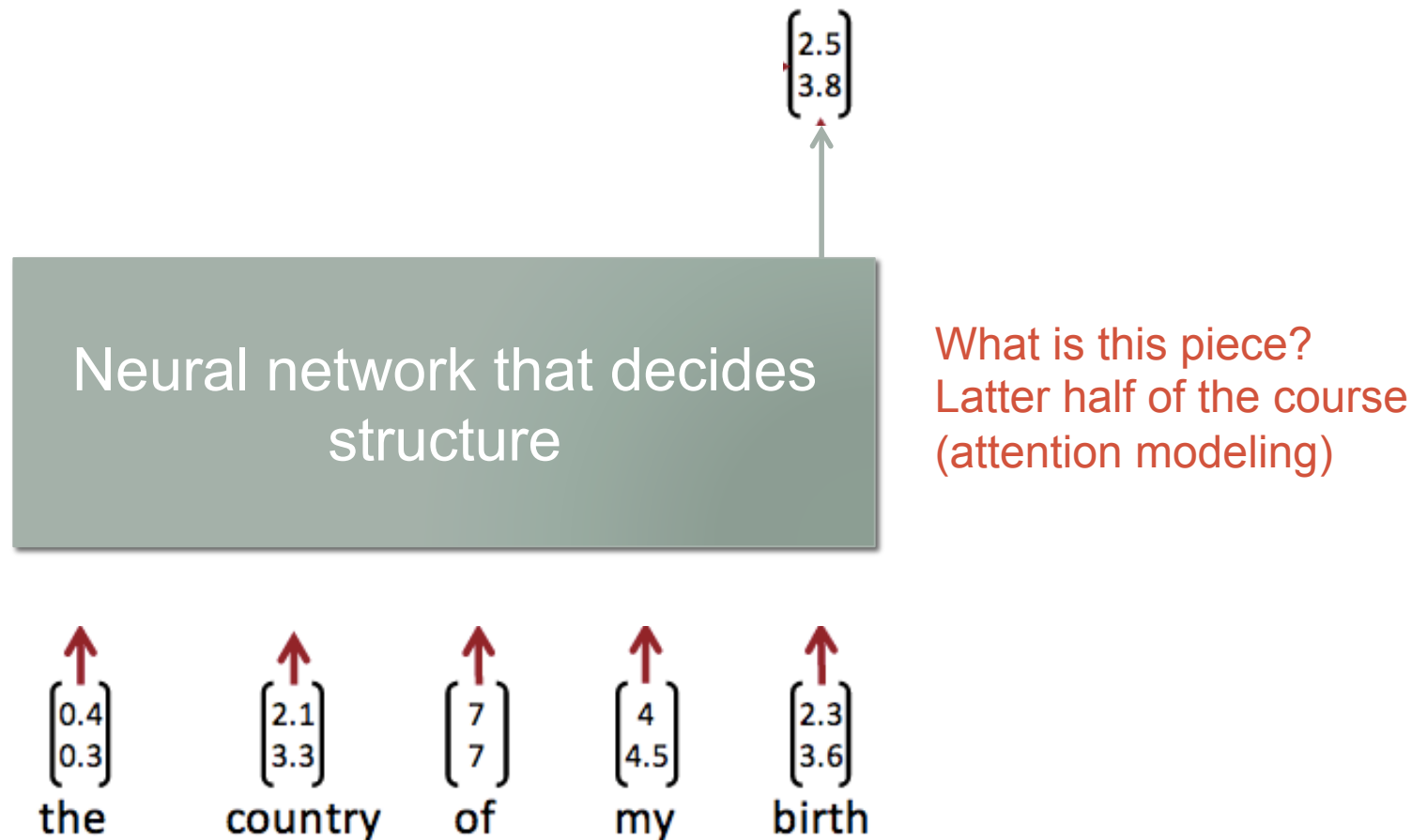
Which to use is still up for debate



Towards unsupervised dependency parsing



Towards unsupervised dependency parsing



Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

Words

Morphology

Characters

Ref: Prof. Brendan O'Connor, CS 885 Intro to NLP, @UMass

CommunicationEvent(e) SpeakerContext(s)
Agent(e,Alice) TemporalBefore(e, s)
Recipient(e, Bob)

Word embeddings

Typical flow

Parsing

PoS Tagging

Language model

Tokenization

Alice talked to Bob

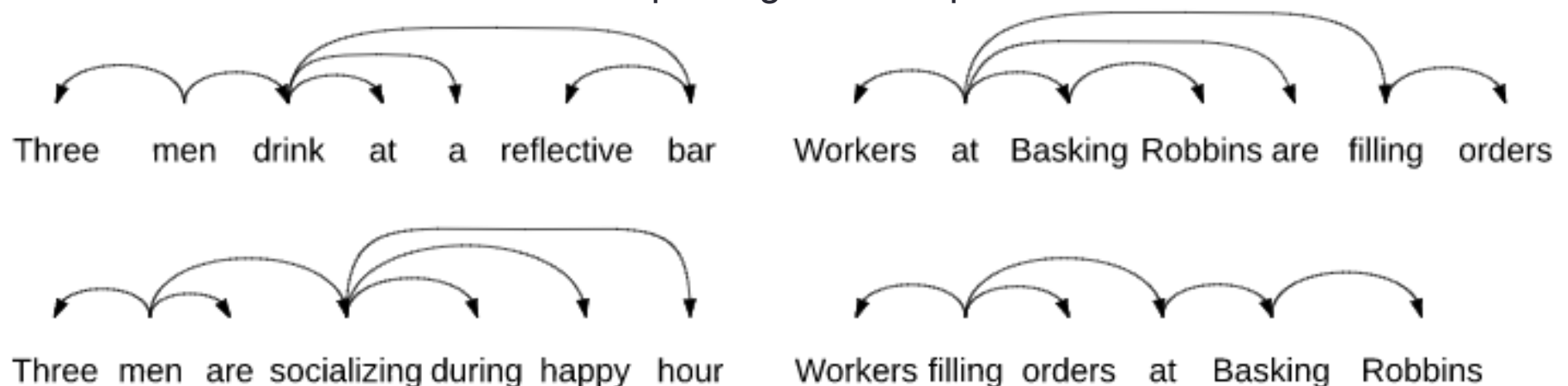
talk -ed [VerbPast]

Alice talked to Bob.

An top down end-to-end approach

- Input: text
- Output: some task, sentiment analysis score
- Automatically gets parse tree (without any treebank corpus)

Example of generated parse tree



<https://arxiv.org/pdf/1705.09207.pdf>

Can also use recursive neural networks to learn unsupervised parse trees

Example <https://arxiv.org/pdf/1707.02786.pdf>

Summary

- Parsing
- Types of grammars
 - Context Free Grammar
 - Probabilistic Context Free Grammar
 - CYK parser
 - Dependency Grammar
 - Transition-based parsing
 - Recursive Neural networks

Document classification, sentiment analysis, QA,
conversation agents, summarization, translation

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

Words

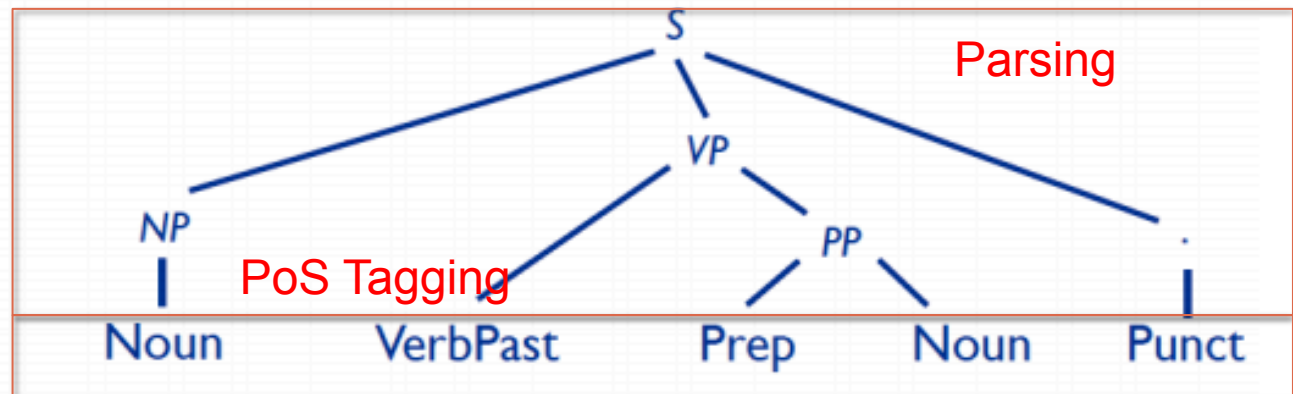
Morphology

Characters

Ref: Prof. Brendan O'Connor, CS 885 Intro to NLP, @UMass

CommunicationEvent(e) SpeakerContext(s)
Agent(e,Alice) TemporalBefore(e, s)
Recipient(e, Bob)

Word embeddings



Alice talked to Bob .

Language modeling

↑ Tokenization

talk -ed [VerbPast]

Alice talked to Bob .

Midterm

- Next Monday
 - Theory part (paper exam)
 - Closed book
 - One sheet of note (A4)
 - Calculator
 - Hands-on part (coding exam)
 - Two weeks (Kaggle style)
 - Open everything, except discussing to each other (please be honest)
 - Due Sunday March 11th Noon
- Material: everything till this lecture.
 - Expect some small hand calculations for some algorithms.
 - You will not be asked to do a dependency parse by hand.
 - You are not expected to use parsing on the coding exam.