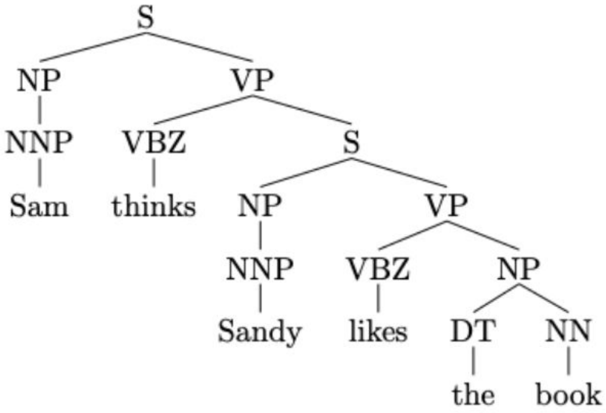
Midterm Review: CFGs, Parsing, and Neural Networks

Linguistic Structure

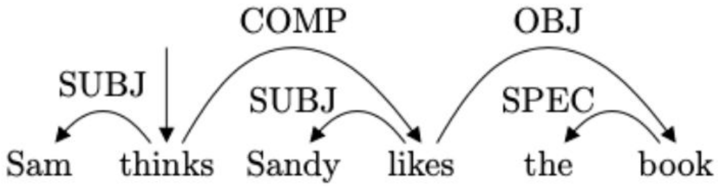
Constituency

- “Groups of words can behave as single units (constituents)”
- Based on Context Free Grammars (CFGs)



Dependency

- “Syntactic structure of a sentence is described solely in terms of relations between the words”



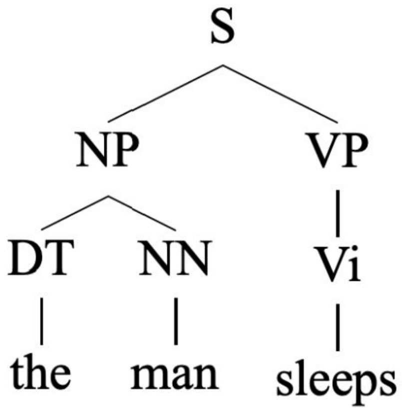
Constituency Parsing

Context-Free Grammars (CFGs)

A formal system for modeling constituent structure in natural language

Consists of:

- Non-terminals *N*
  - E.g. {S, NP, VP, DT, NN, Vi}
- Terminals Σ
  - E.g. {the, man, sleeps}
- Rules *R* (grammar, lexicon)
  - E.g. {S -> NP VP, NP -> DT NN, VP -> Vi, DT -> the, NN -> man, Vi -> sleeps}
- Start symbol *S* (picked from *N*)
  - E.g. S

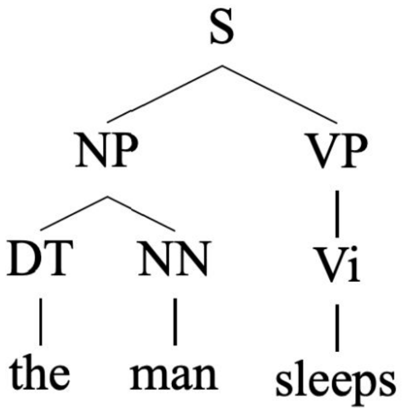


Deriving Parses with CFGs

Given a CFG, we want to get from a starting string s’ to a target string s

(i.e. we want the **derivation** of s starting from s’)

Represented as parse tree



Probabilistic Context-Free Grammars (PCFGs)

CFG + **Probabilities for each rule**

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

$R =$

S	→	NP	VP
VP	→	Vi	
VP	→	Vt	NP
VP	→	VP	PP
NP	→	DT	NN
NP	→	NP	PP
PP	→	IN	NP

Grammar

Vi	→	sleeps
Vt	→	saw
NN	→	man
NN	→	woman
NN	→	telescope
NN	→	dog
DT	→	the
IN	→	with
IN	→	in

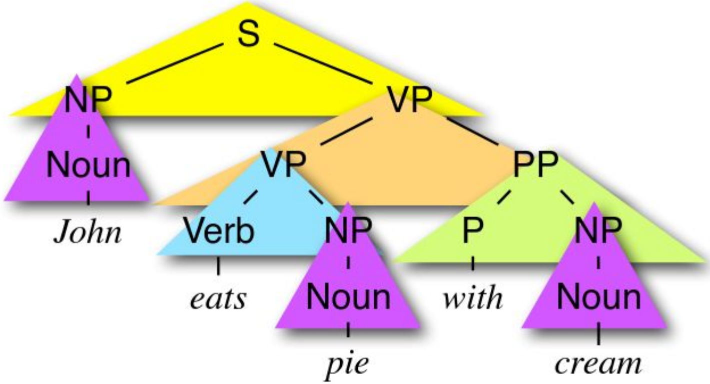
Lexicon

S	→	NP	VP	1.0
VP	→	Vi		0.3
VP	→	Vt	NP	0.5
VP	→	VP	PP	0.2
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

Vi	→	sleeps	1.0
Vt	→	saw	1.0
NN	→	man	0.1
NN	→	woman	0.1
NN	→	telescope	0.3
NN	→	dog	0.5
DT	→	the	1.0
IN	→	with	0.6
IN	→	in	0.4

Calculating Probability of a Parse Tree

The probability of a tree τ is the product of the probabilities of all its rules:



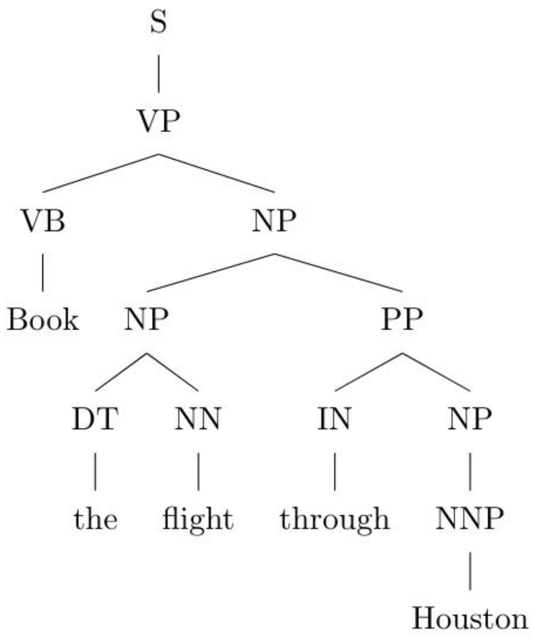
S	→	NP VP	0.8
S	→	S conj S	0.2
NP	→	Noun	0.2
NP	→	Det Noun	0.4
NP	→	NP PP	0.2
NP	→	NP conj NP	0.2
VP	→	Verb	0.4
VP	→	Verb NP	0.3
VP	→	Verb NP NP	0.1
VP	→	VP PP	0.2
PP	→	P NP	1.0

$$P(\tau) = 0.8 \times 0.3 \times 0.2 \times 1.0 \times 0.2^3$$

$$= \mathbf{0.00384}$$

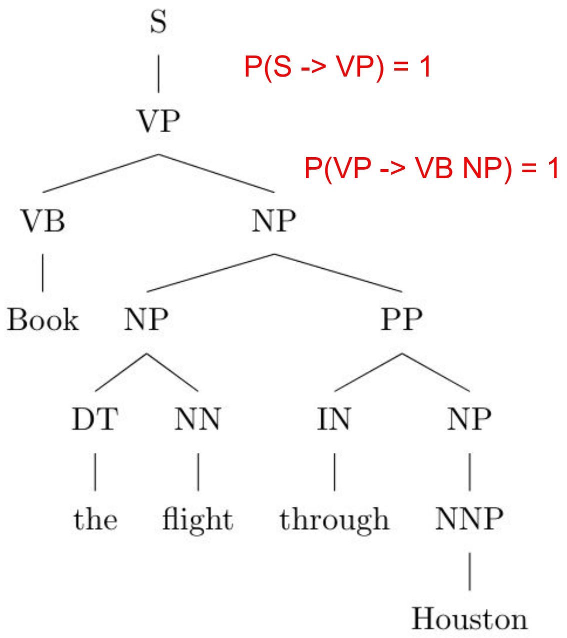
Calculating Probability of a Parse Tree

R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



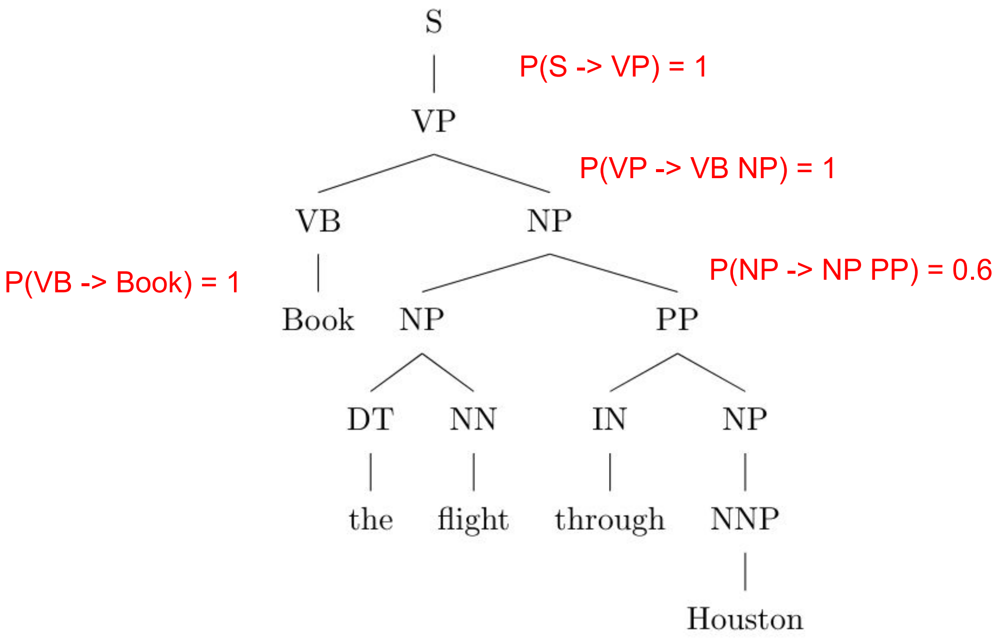
Calculating Probability of a Parse Tree

R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



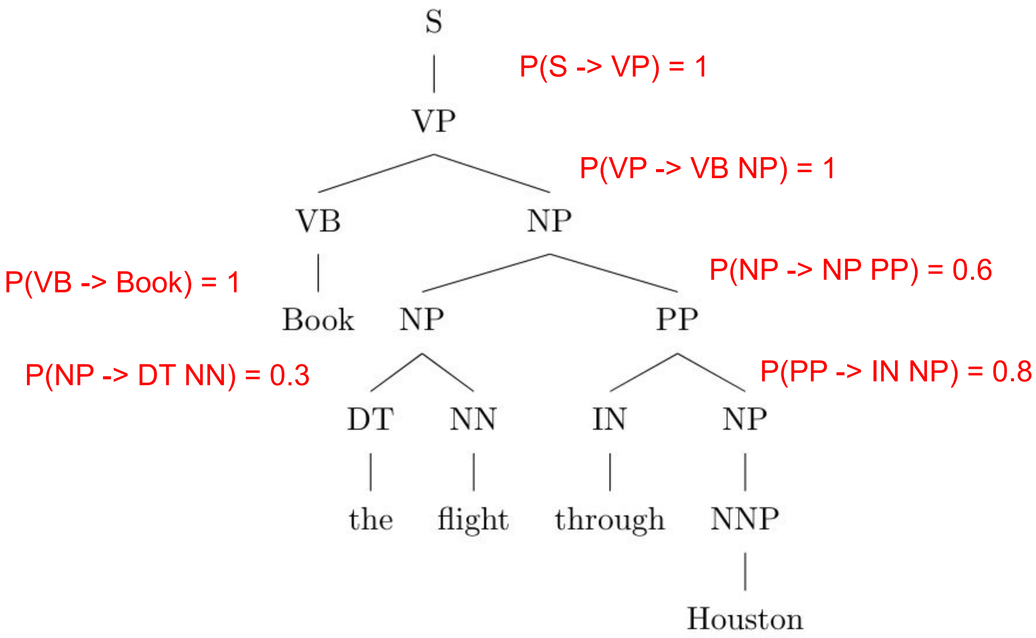
Calculating Probability of a Parse Tree

R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



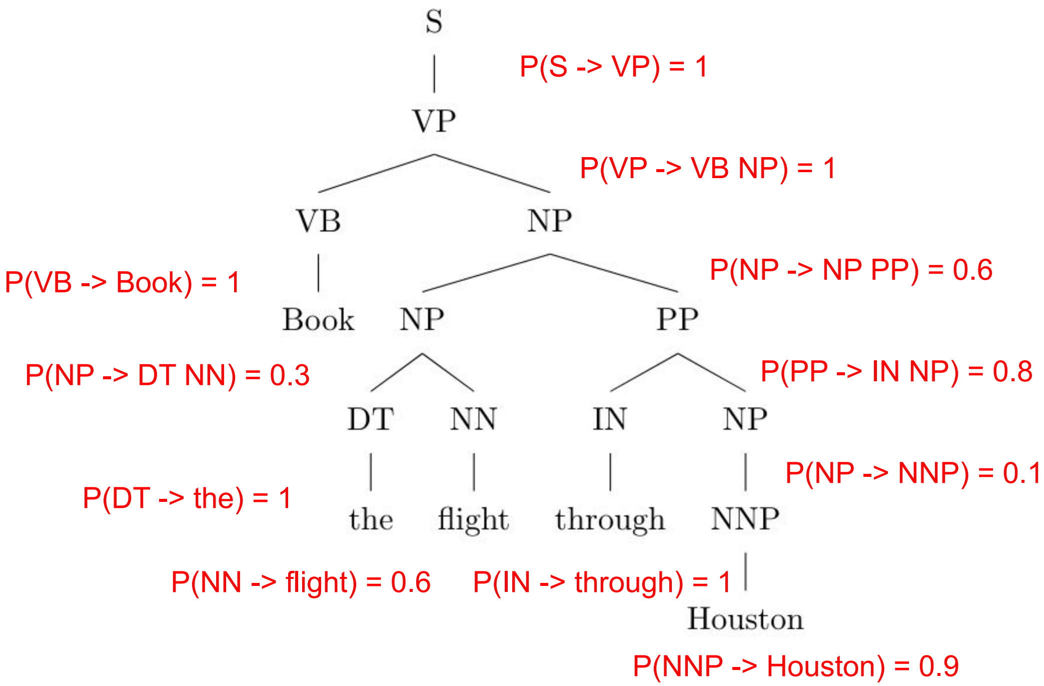
Calculating Probability of a Parse Tree

R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



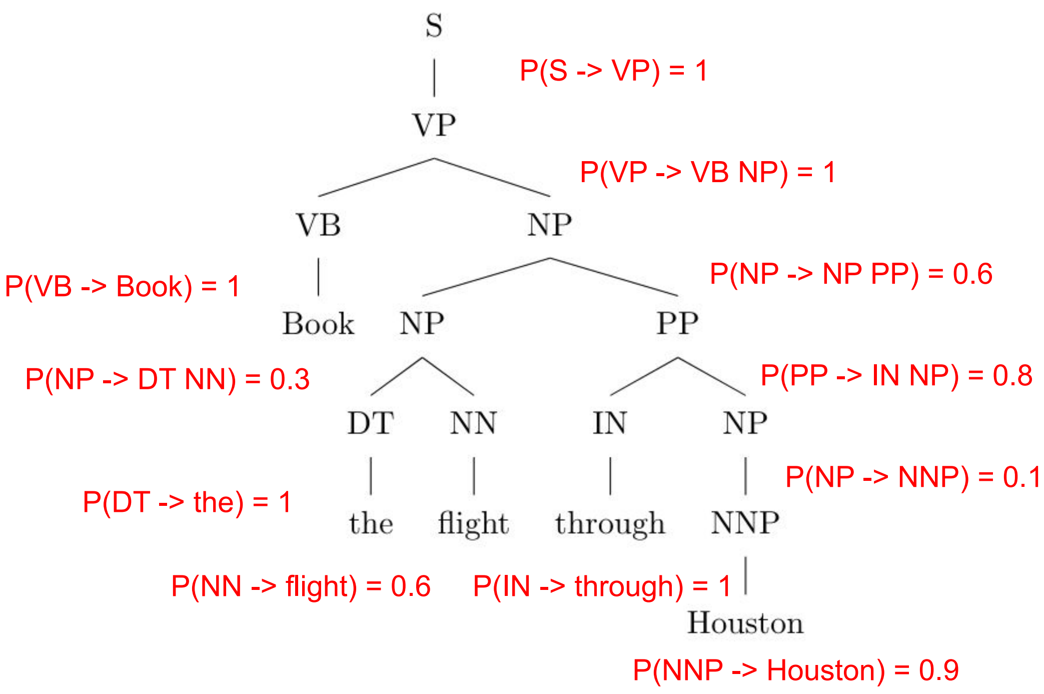
Calculating Probability of a Parse Tree

R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



Calculating Probability of a Parse Tree

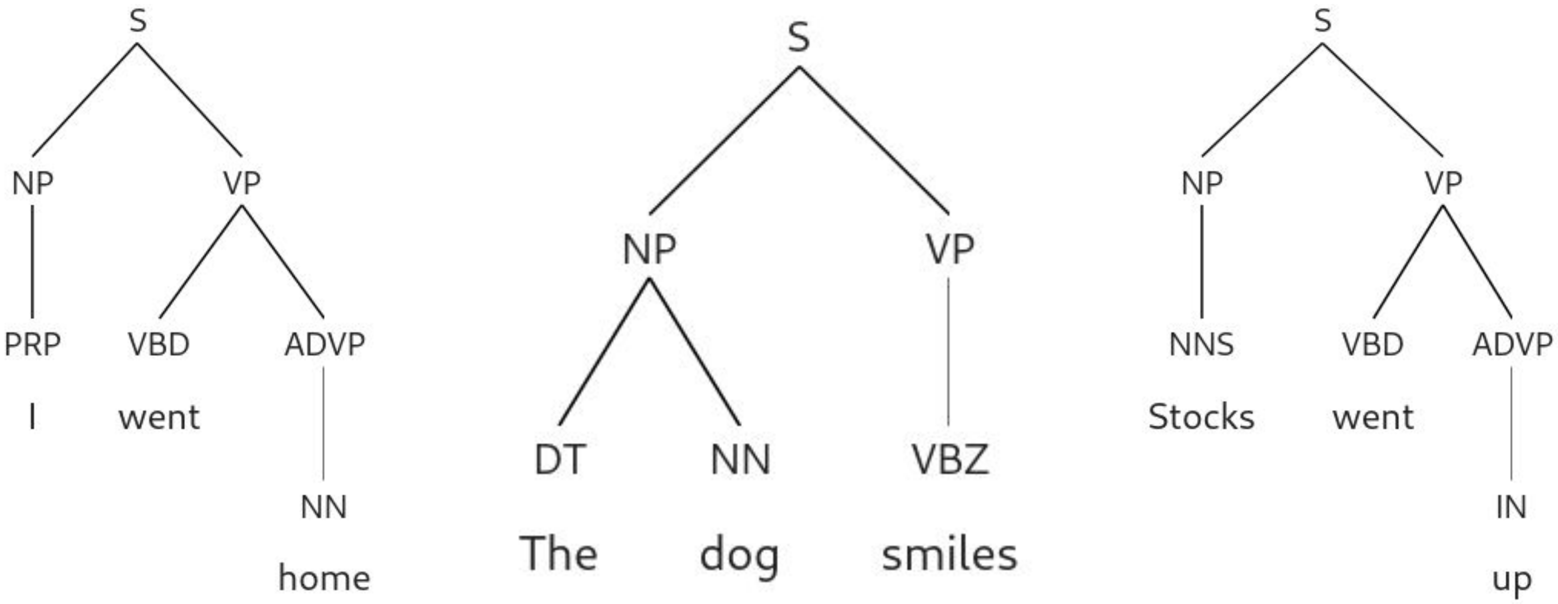
R	$q(R)$
S → VP	1
VP → VB NP	1
NP → NP PP	.6
NP → DT NN	.3
NP → NNP	.1
PP → IN NP	.8
PP → VB NP	.2
NN → flight	.6
NN → train	.4
IN → through	1
NNP → Houston	.9
NNP → France	.1
DT → the	1
VB → Book	1



Probability: 0.6 * 0.8 * 0.3 * 0.1 * 0.6 * 0.9 = .0077

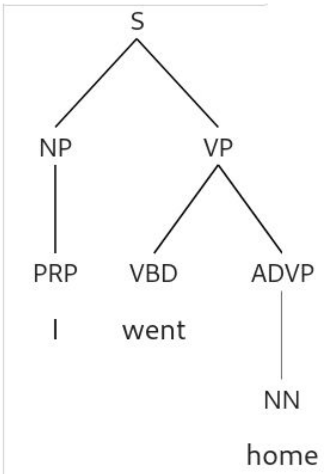
Treebanks

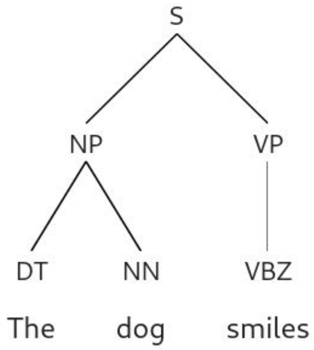
Dataset of sentences + associated parse trees

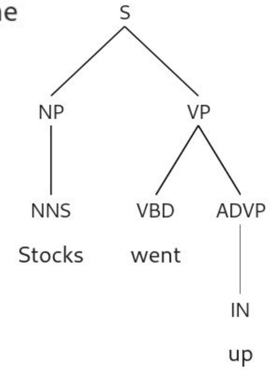


PCFG from Treebank

1) Get *N*, Σ, *S*, *R*
   a) *N* = All non-terminals
   b) Σ = All terminals
   c) *S* = Root of trees
   d) *R* = For each node, get all children
2) To construct probabilities *q*:
   a) For each non-terminal:
      i) Count all parent -> children relationship
      ii) Divide by number of occurrences of the non-terminal







CKY Algorithm

For a string with multiple parses, we want the highest probability one

Inputs:

- PCFG given by *N*, Σ, *S, R, q*, where *R* is in CNF (all nodes have either 1 terminal child, or 2 non-terminal children)
- A sentence $X = (x_1, x_2, \ldots, x_n)$

Outputs:

- The parse of *X* with highest probability

CKY Example

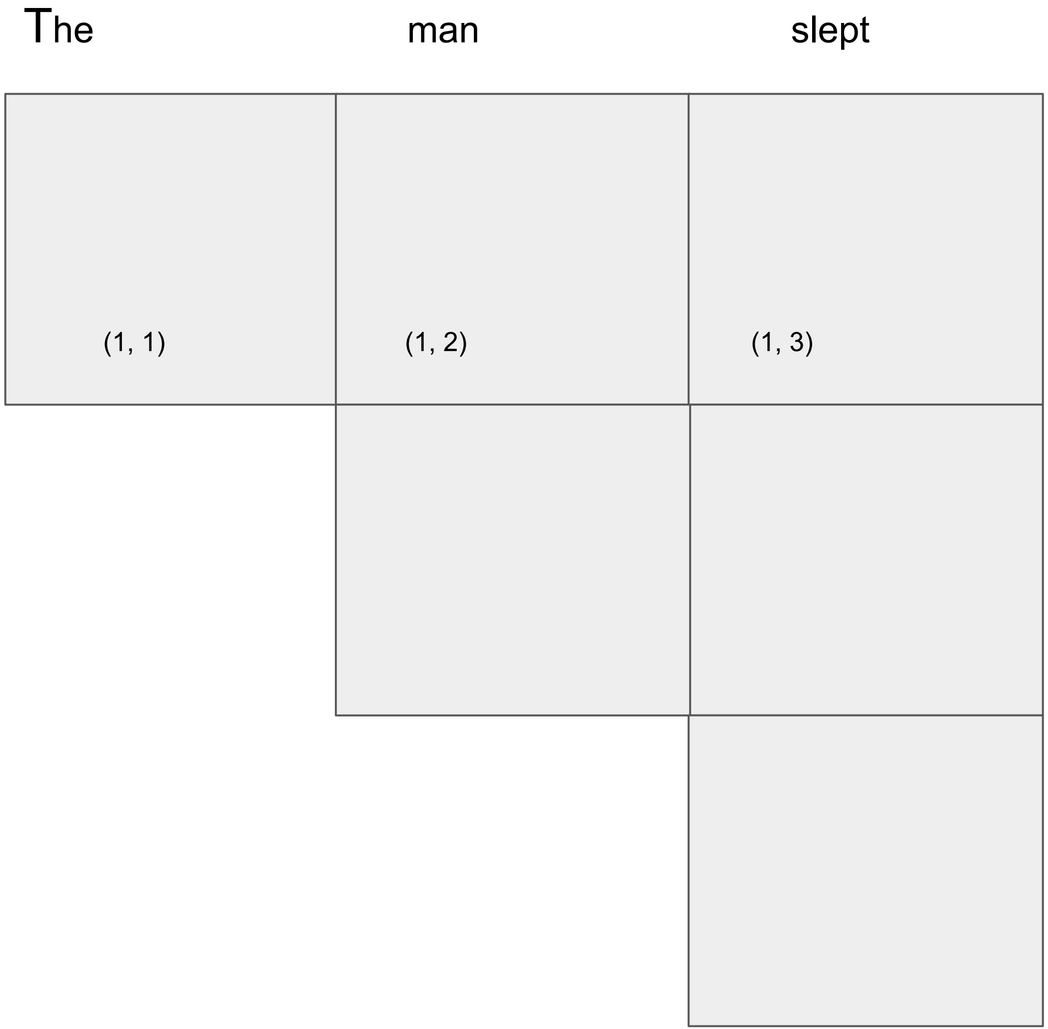
Sentence: The man slept

R	$q(R)$
S → NP VP	1
NP → DT NN	.6
NP → NP VP	.4
DT → The	1
NN → man	1
VP → slept	1

CKY Example

Sentence: The man slept

R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1



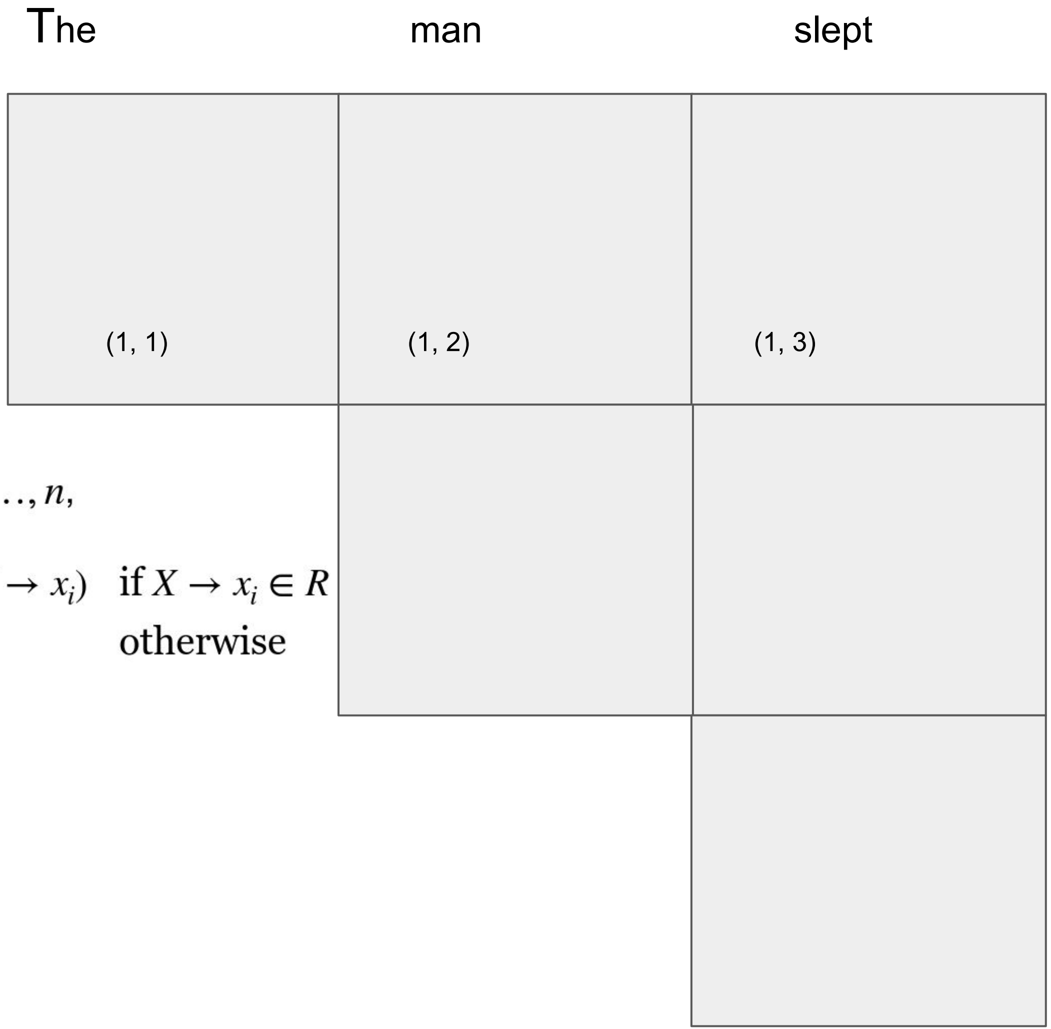
CKY Example

Sentence: The man slept

R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1

Initially, for $i = 1,2,\ldots,n$,

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$



CKY Example

Sentence: The man slept

R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1

Initially, for $i = 1, 2, \ldots, n$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

The	man	slept
π(1,1,DT)=1 (1, 1)	(1, 2)	(1, 3)
	π(2,2,NN)=1	
		π(3,3,VP)=1

CKY Example

Sentence: The man slept

R	$q(R)$
$S \to NP\ VP$	1
$NP \to DT\ NN$	.6
$NP \to NP\ VP$	.4
$DT \to The$	1
$NN \to man$	1
$VP \to slept$	1

	The	man	slept
	$\pi(1,1,DT)=1$ (1, 1)	(1, 2)	(1, 3)
		$\pi(2,2,NN)=1$	
			$\pi(3,3,VP)=1$

Initially, for $i = 1,2,\ldots,n$,

$$\pi(i,i,X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

For all (i,j) such that $1 \le i < j \le n$ for all $X \in N$,

$$\pi(i,j,X) = \max_{X \to YZ \in R, i \le k < j} q(X \to YZ) \times \pi(i,k,Y) \times \pi(k+1,j,Z)$$

CKY Example

Sentence: The man slept

R	$q(R)$
$S \rightarrow NP\ VP$	1
$NP \rightarrow DT\ NN$	.6
$NP \rightarrow NP\ VP$	.4
$DT \rightarrow The$	1
$NN \rightarrow man$	1
$VP \rightarrow slept$	1

The	man	slept
$\pi(1,1,DT)=1$	$\pi(1, 2, NP)=.6$ (1, 2)	(1, 3)
	$\pi(2,2,NN)=1$	
		$\pi(3,3,VP)=1$

We only need to consider k = 1 (i.e. π(1,1,Y) and π(2,2,Z))

$\pi(1,2, S) = 0$
$\pi(1, 2, NP) = .6$

Initially, fo…

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

For all (i, j) such that $1 \leq i < j \leq n$ for all $X \in N$,

$$\pi(i,j,X) = \max_{X \rightarrow YZ \in R, i \leq k < j} q(X \rightarrow YZ) \times \pi(i,k,Y) \times \pi(k+1,j,Z)$$

CKY Example

Sentence: The man slept

R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1

The	man	slept
π(1,1,DT)=1	π(1, 2, NP)=.6 (1, 2)	(1, 3)
	π(2,2,NN)=1	
		π(3,3,VP)=1

Add backreferences

Initially, for $i = 1,2,\ldots,n$,

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

For all (i,j) such that $1 \leq i < j \leq n$ for all $X \in N$,

$$\pi(i,j,X) = \max_{X \rightarrow YZ \in R, i \leq k < j} q(X \rightarrow YZ) \times \pi(i,k,Y) \times \pi(k+1,j,Z)$$

CKY Example

Sentence: The man slept

R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1

Initially, for $i = 1,2,\ldots,n$,

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{other…} \end{cases}$$

For all (i,j) such that $1 \leq i < j \leq$ …

$$\pi(i,j,X) = \max_{X \rightarrow YZ \in R, i \leq k < j} q(X \rightarrow \ldots$$

	The	man	slept
	π(1,1,DT)=1 (1, 1)	π(1, 2, NP)=.6 (1, 2)	(1, 3)
		π(2,2,NN)=1	π(2, 3, X) = 0
			π(3,3,VP)=1

We only need to consider k = 2 (i.e. π(2,2,Y) and π(3,3,Z))

π(2, 3, S) = 0
π(2, 3, NP) = 0

CKY Example

Sentence: The man slept

R	$q(R)$
S → NP VP	1
NP → DT NN	.6
NP → NP VP	.4
DT → The	1
NN → man	1
VP → slept	1

Initially, fo

$\pi(i, i, X$

For all $(i,$

$\pi(i, j$

The | man | slept

$\pi(1,1,DT)=1$ (1, 1)

$\pi(1, 2, NP)=.6$ (1, 2)

$\pi(1, 3, S)=.6$ (1, 3)

$\pi(2, 3, X) = 0$

$\pi(3,3,VP)=1$

We now must consider k=1,2

π(1, 3, S) = max{

q(S -> NP VP) π(1,1,NP) π(2, 3, VP) = 0,

q(S -> NP VP) π(1,2,NP) π(3, 3, VP) = .6

}

π(1, 3, NP) = max{

q(NP -> NP VP) π(1,1,NP) π(2, 3, VP) = 0,

q(NP -> NP VP) π(1,2,NP) π(3, 3, VP) = .24,

q(NP -> DT NN) π(1,1,DT) π(2, 3, NN) = 0,

q(NP -> NP VP) π(1,2,DT) π(3, 3, NN) = 0,

}

CKY Example

Sentence: The man slept

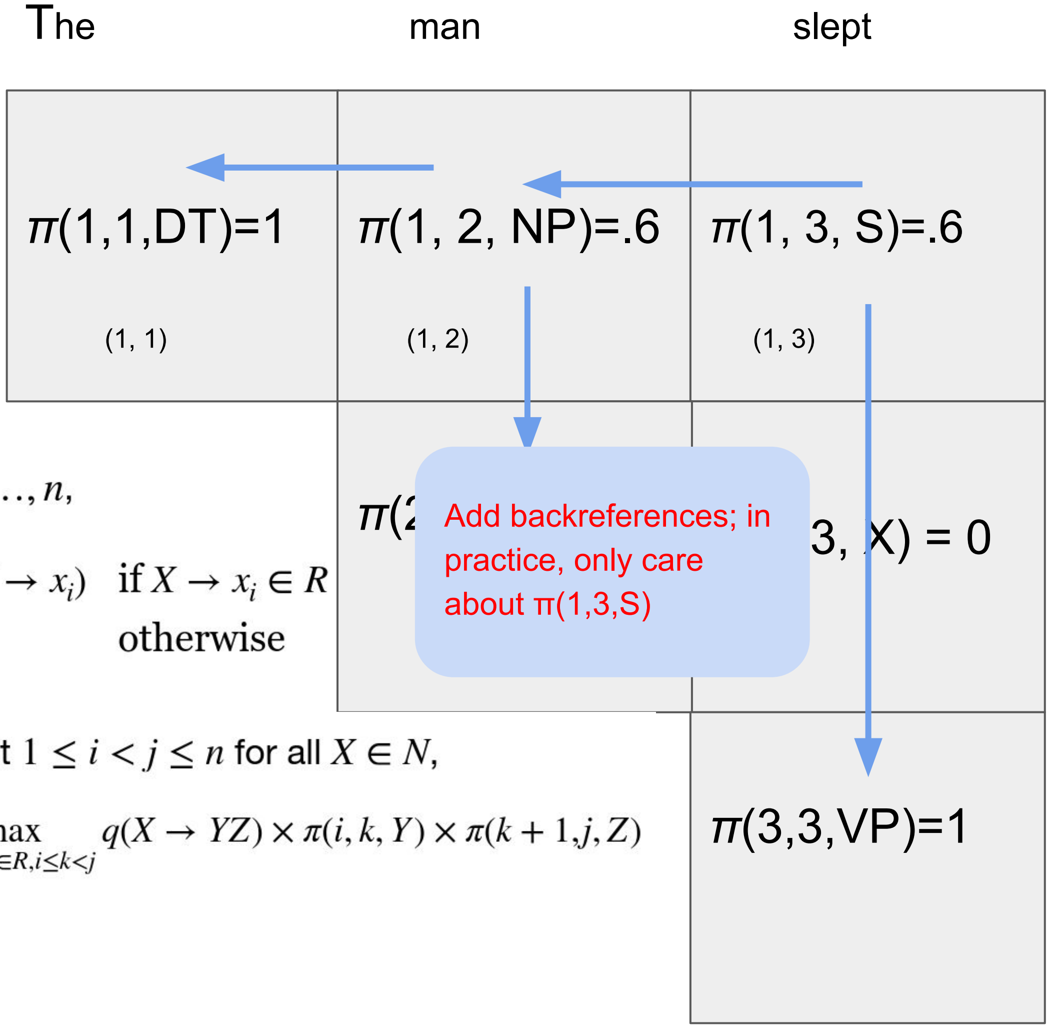
R	$q(R)$
S $\rightarrow$ NP VP	1
NP $\rightarrow$ DT NN	.6
NP $\rightarrow$ NP VP	.4
DT $\rightarrow$ The	1
NN $\rightarrow$ man	1
VP $\rightarrow$ slept	1

Initially, for $i = 1,2,\ldots,n,$

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

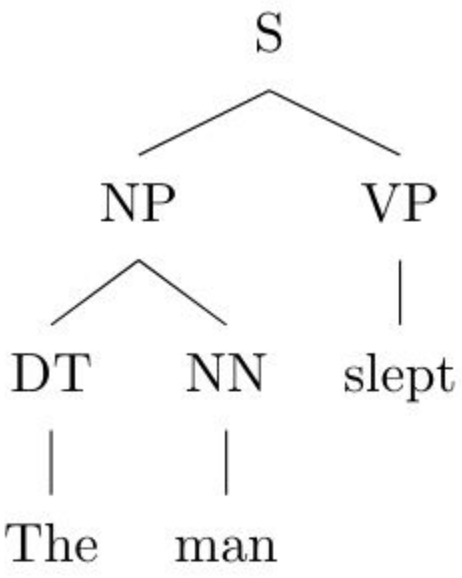
For all (i,j) such that $1 \leq i < j \leq n$ for all $X \in N$,

$$\pi(i,j,X) = \max_{X \rightarrow YZ \in R, i \leq k < j} q(X \rightarrow YZ) \times \pi(i,k,Y) \times \pi(k+1,j,Z)$$

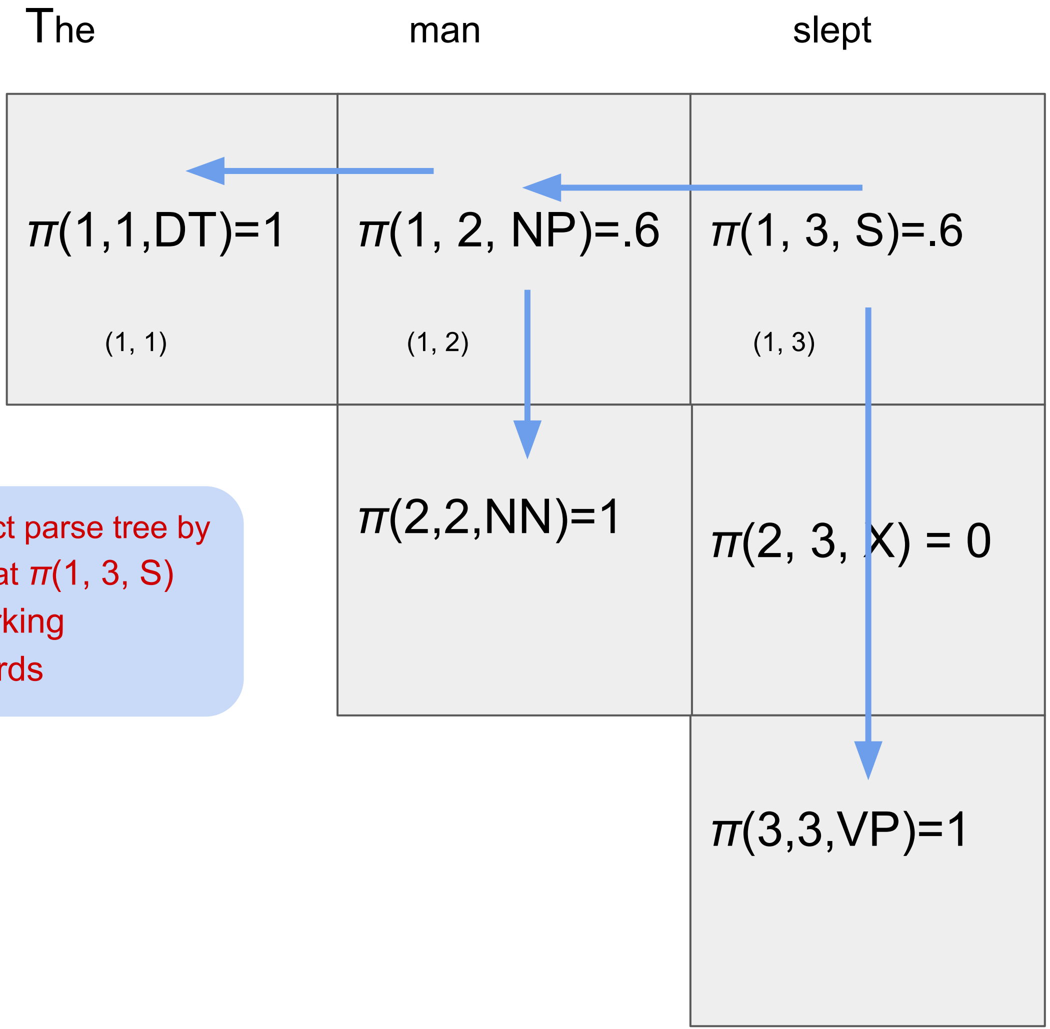


CKY Example

Sentence: The man slept



Construct parse tree by starting at $\pi(1, 3, S)$ and working backwards



Dependency Parsing

The Arc-standard algorithm

- Given: a sentence of $w_1, w_2, \ldots, w_n$
- The parsing process is modeled as a sequence of transitions
- A configuration (current state of parse) consists of a stack s, a buffer b and a set of dependency arcs A: $c = (s, b, A)$
- Initially, $s = [\text{ROOT}]$, $b = [w_1, w_2, \ldots, w_n]$, $A = \varnothing$
- A configuration is terminal if $s = [\text{ROOT}]$ and $b = \varnothing$
- Three types of transitions: SHIFT, LEFT-ARC (l), RIGHT-ARC (r)

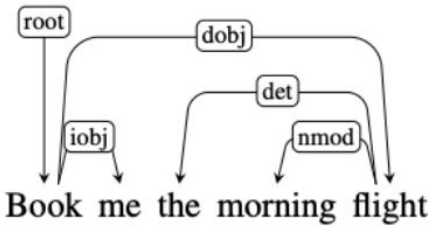
Arc-standard

Want to build a dependency parse for a sentence

- Three types of transitions: SHIFT, LEFT-ARC (r), RIGHT-ARC (r)

Arc-standard system: three operations

- Shift: top of buffer -> top of stack
- Left-Arc: $\sigma | w_{-2}, w_{-1} \rightarrow \sigma | w_{-1}$, w_{-2} is now a child of w_{-1}
- Right-Arc $\sigma | w_{-2}, w_{-1} \rightarrow \sigma | w_{-2}$, w_{-1} is now a child of w_{-2}



"Book me the morning flight"

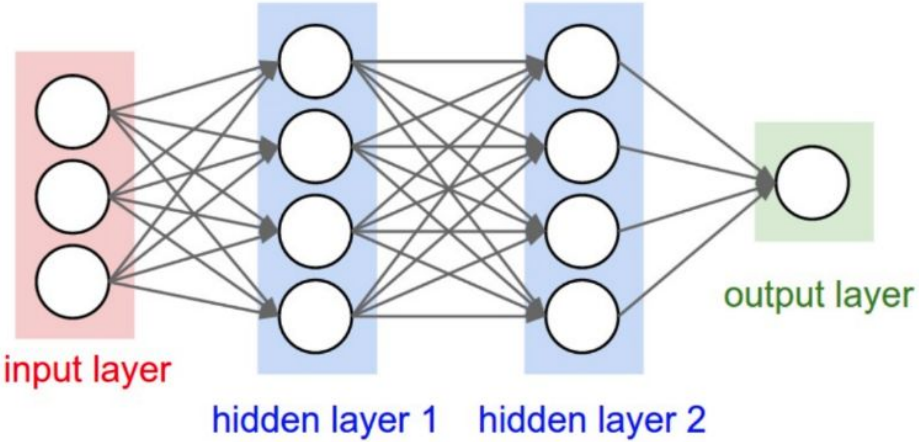
A running example

	stack	buffer	action	added arc
0	[ROOT]	[Book, me, the, morning, flight]	SHIFT	
1	[ROOT, Book]	[me, the, morning, flight]	SHIFT	
2	[ROOT, Book, me]	[the, morning, flight]	RIGHT-ARC(iobj)	(Book, iobj, me)
3	[ROOT, Book]	[the, morning, flight]	SHIFT	
4	[ROOT, Book, the]	[morning, flight]	SHIFT	
5	[ROOT, Book, the, morning]	[flight]	SHIFT	
6	[ROOT, Book, the,morning,flight]	[]	LEFT-ARC(nmod)	(flight,nmod,morning)
7	[ROOT, Book, the, flight]	[]	LEFT-ARC(det)	(flight,det,the)
8	[ROOT, Book, flight]	[]	RIGHT-ARC(dobj)	(Book,dobj,flight)
9	[ROOT, Book]	[]	RIGHT-ARC(root)	(ROOT,root,Book)
10	[ROOT]	[]		

Neural Networks

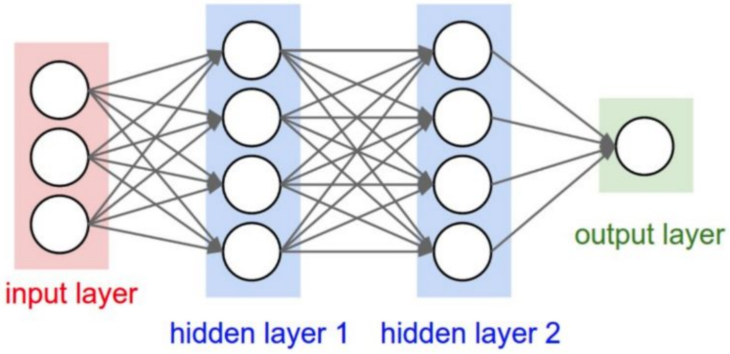
Feed-forward NNs

- The units are connected with no cycles
- The outputs from units in each layer are passed to units in the next higher layer
- No outputs are passed back to lower layers



Fully-connected (FC) layers:
All the units from one layer are fully connected to every unit of the next layer.

Feed forward neural networks



*: *f* is applied element-wise

$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$

C: number of classes

d: input dimension, d_1, d_2: hidden dimensions

- Input layer: $\mathbf{x} \in \mathbb{R}^d$
- Hidden layer 1:

$$\mathbf{h}_1 = f(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \in \mathbb{R}^{d_1}$$

$$\mathbf{W}^{(1)} \in \mathbb{R}^{d_1 \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^{d_1}$$

- Hidden layer 2:

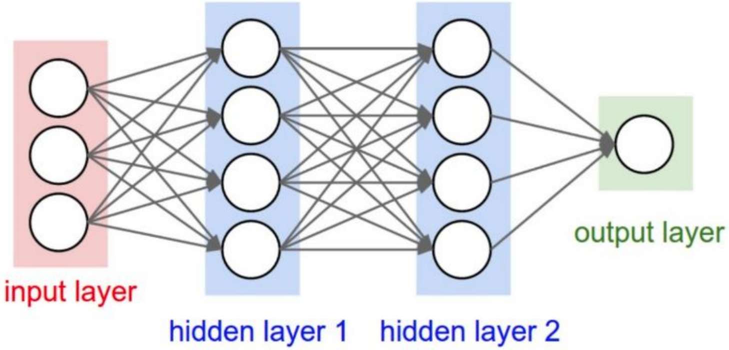
$$\mathbf{h}_2 = f(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)}) \in \mathbb{R}^{d_2}$$

$$\mathbf{W}^{(2)} \in \mathbb{R}^{d_2 \times d_1}, \mathbf{b}^{(2)} \in \mathbb{R}^{d_2}$$

- Output layer:

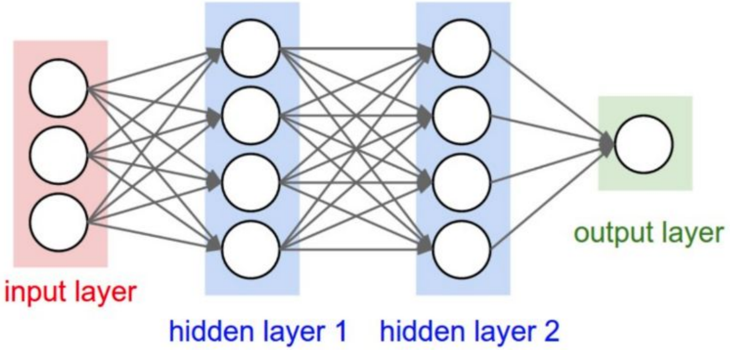
$$\mathbf{y} = \mathbf{W}^{(o)}\mathbf{h}_2, \mathbf{W}^{(o)} \in \mathbb{R}^{C \times d_2}$$

Feed forward neural networks



$$
\begin{aligned}
\mathbf{h} &= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \\
\mathbf{z} &= \mathbf{U}\mathbf{h} \\
\mathbf{y} &= \mathrm{softmax}(\mathbf{z})
\end{aligned}
$$

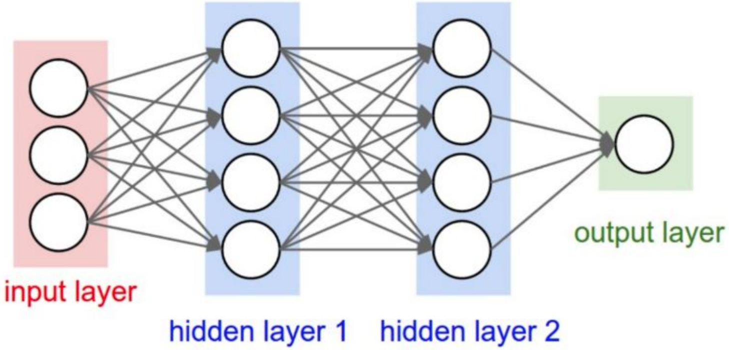
Feed forward neural networks



$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\mathbf{z} = \mathbf{U}\mathbf{h}$$
$$\mathbf{y} = \mathrm{softmax}(\mathbf{z})$$

Q: Suppose your input is of dimensionality *N*, hidden state is size *H,* and you are classifying for C classes. Suppose that your network has L hidden layers (all of size H). How many parameters does the model have?

Feed forward neural networks



$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\mathbf{z} = \mathbf{U}\mathbf{h}$$
$$\mathbf{y} = \text{softmax}(\mathbf{z})$$

Q: Suppose your input is of dimensionality *N*, hidden state is size *H,* and you are classifying for C classes. Suppose that your network has L hidden layers (all of size H). How many parameters does the model have?

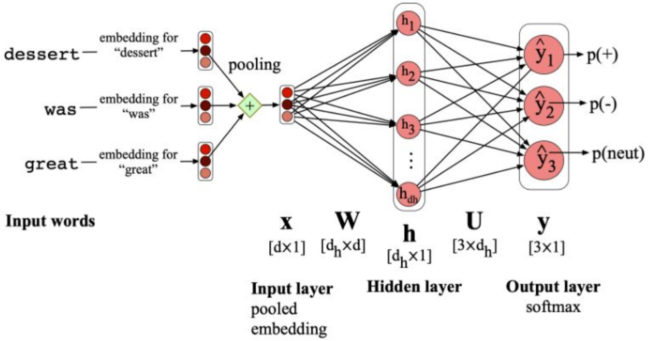
A: NH + H + (L-1)(H^2 + H) + CH

Neural bag-of-words models for text classification

- Want to train a feed forward network to classify text
- We need a way to get a feature vector **x** given a sentence $\mathbf{w}_1, \ldots, \mathbf{w}_n$
- Solutions:
  - Extract features manually from sentence
  - Use **word embeddings** to embed each word, and pool

$$x_{\text{mean}} = \frac{1}{K}\sum_{i=1}^{K} e(w_i)$$

pooling: sum, mean or max



- $\mathbf{x} = \frac{1}{K}\sum_{i=1}^{K} e(w_i)$
- $\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$
- $\mathbf{y} = \mathbf{U}\mathbf{h}$
- $\hat{\mathbf{y}} = \text{softmax}(\mathbf{y})$

How to train this model?

- Training data: $\{(d^{(1)}, y^{(1)}), \ldots, (d^{(m)}, y^{(m)})\}$
- Parameters: $\{\mathbf{W}, \mathbf{b}, \mathbf{U}\}$
- Optimize these parameters using gradient descent!

- Word embeddings can be treated as parameters too!

$$\mathbf{E} \in \mathbb{R}^{|V| \times d}$$

- $\mathbf{x} = \frac{1}{K}\sum_{i=1}^{K} e(w_i)$
- $\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$
- $\mathbf{y} = \mathbf{U}\mathbf{h}$
- $\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{y})$

Feedforward Neural Language Model

- Recap:

Language models: Given $x_1, x_2, \ldots, x_n \in V$, the goal is to model:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_1, \ldots, x_{i-1})$$

- N-gram models suffer from many issues:
  - Exponential scaling with context size
  - Sparse probabilities as context size increases

Feedforward Neural Language Model

- Solution: Can treat language modelling as V way classification task

  - Input layer (m= 5):

    $$\mathbf{x} = [e(\text{the}); e(\text{cat}); e(\text{sat}); e(\text{on}); e(\text{the})] \in \mathbb{R}^{md}$$

  - Hidden layer:

    $$\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^h$$

  - Output layer

    $$\mathbf{z} = \mathbf{U}\mathbf{h} \in \mathbb{R}^{|V|}$$

    $$P(w = i \mid \text{the cat sat on the}) = \text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$