



Natural Language Processing

Info 159/259

Lecture 15: Review (Oct 12, 2017)

David Bamman, UC Berkeley

Big ideas

- Classification
 - Naive Bayes, Logistic regression, feedforward neural networks, CNN.
- Where does NLP data come from?
 - Annotation process
 - Interannotator agreement
- Language modeling
 - Markov assumption, featurized, neural
- Probability/statistics in NLP
 - Chain rule of probability, independence, Bayes' rule

Big ideas

- Lexical semantics
 - Distributional hypothesis
 - Distributed representations
- Evaluation
 - Metrics (accuracy, precision, recall, F score, perplexity, parseval)
 - Hypothesis tests
- Sequence labeling
 - POS, NER
 - Methods: HMM, MEMM, CRF, RNN, BiRNN
- Trees
 - Phrase-structure parsing, CFG, PCFG
 - CKY for recognition, parsing

Big ideas

- What defines the models we've seen so far? What formally distinguishes an HMM from an MEMM? How do we train those models?
- For all of the problems we've seen (sentiment analysis, POS tagging, phrase structure parsing), how do we evaluate the performance of different models?
- If faced with a new NLP problem, how would you decide between the alternatives you know about? How would you adapt an MEMM, for example, to a new problem?



Classification

A mapping h from input data x (drawn from instance space $\mathcal{X}$) to a label (or labels) y from some enumerable output space $\mathcal{Y}$

$\mathcal{X}$ = set of all documents

$\mathcal{Y} = \{\text{english, mandarin, greek, ...}\}$

x = a single document

y = ancient greek

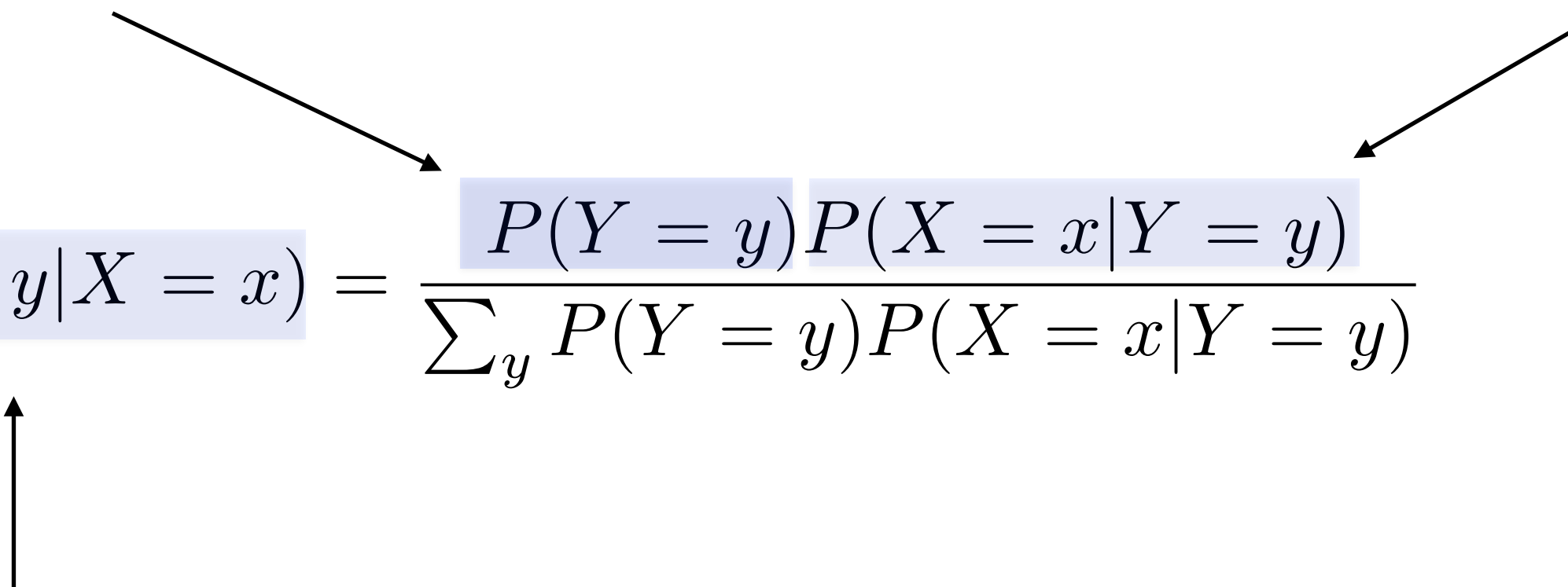
Text categorization problems

task	x	y
language ID	text	{english, mandarin, greek, ...}
spam classification	email	{spam, not spam}
authorship attribution	text	{jk rowling, james joyce, ...}
genre classification	novel	{detective, romance, gothic, ...}
sentiment analysis	text	{positive, negative, neutral, mixed}

Bayes' Rule

Prior belief that $Y = y$
(before you see any data)

Likelihood of the data
given that $Y=y$



The diagram illustrates the components of Bayes' Rule. An arrow points from the text 'Prior belief that Y = y (before you see any data)' to the term $P(Y = y)$ in the numerator. Another arrow points from the text 'Likelihood of the data given that Y=y' to the term $P(X = x|Y = y)$ in the numerator. A third arrow points from the text 'Posterior belief that Y=y given that X=x' to the entire fraction on the left side of the equation.

$$P(Y = y|X = x) = \frac{P(Y = y)P(X = x|Y = y)}{\sum_y P(Y = y)P(X = x|Y = y)}$$

Posterior belief that $Y=y$ given that $X=x$

Bayes' Rule

Prior belief that $Y = \text{positive}$
(before you see any data)

Likelihood of “really really the
worst movie ever”
given that $Y = \text{positive}$

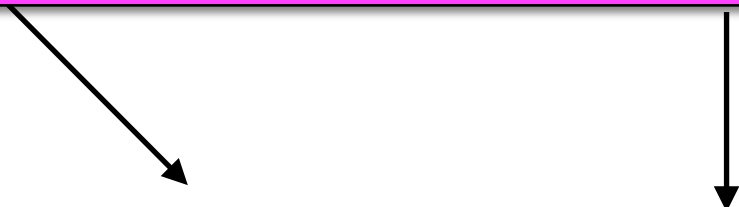
$$P(Y = y | X = x) = \frac{P(Y = y)P(X = x | Y = y)}{\sum_y P(Y = y)P(X = x | Y = y)}$$

Posterior belief that $Y = \text{positive}$ given that
 $X = \text{“really really the worst movie ever”}$

This sum ranges over
 $y = \text{positive} + y = \text{negative}$
(so that it sums to 1)

Naive Bayes training

Training a Naive Bayes classifier consists of estimating these two quantities from training data for all classes y



$$P(Y = y|X = x) = \frac{P(Y = y)P(X = x|Y = y)}{\sum_y P(Y = y)P(X = x|Y = y)}$$

At test time, use those estimated probabilities to calculate the posterior probability of each class y and select the class with the highest probability

Independence Assumption

really really the worst movie ever



We will assume the features are independent:

$$P(x_1, x_2, x_3, x_4, x_6, x_7 \mid c) = P(x_1 \mid c)P(x_2 \mid c) \dots P(x_7 \mid c)$$

$$P(x_1 \dots x_n \mid c) = \prod_{i=1}^N P(x_i \mid c)$$

Logistic regression

$$P(y = 1 \mid x, \beta) = \frac{1}{1 + \exp \left(- \sum_{i=1}^F x_i \beta_i \right)}$$

output space

$$\mathcal{Y} = \{0, 1\}$$

X = feature vector

Feature	Value
the	0
and	0
bravest	0
love	0
loved	0
genius	0
not	0
fruit	1
<i>BIAS</i>	1

β = coefficients

Feature	β
the	0.01
and	0.03
bravest	1.4
love	3.1
loved	1.2
genius	0.5
not	-3.0
fruit	-0.8
<i>BIAS</i>	-0.1

Features

- As a discriminative classifier, logistic regression doesn't assume features are independent like Naive Bayes does.
- Its power partly comes in the ability to create richly expressive features with out the burden of independence.
- We can represent text through features that are not just the identities of individual words, but any feature that is scoped over **the entirety of the input**.

features
contains like
has word that shows up in positive sentiment dictionary
review begins with "I like"
at least 5 mentions of positive affectual verbs (like, love, etc.)

Stochastic g.d.

- Batch gradient descent reasons over every training data point for each update of β . This can be slow to converge.
- Stochastic gradient descent updates β after each data point.

Algorithm 2 Logistic regression stochastic gradient descent

```
1: Data: training data  $x \in \mathbb{R}^F, y \in \{0, 1\}$ 
2:  $\beta = 0^F$ 
3: while not converged do
4:   for  $i = 1$  to  $N$  do
5:      $\beta_{t+1} = \beta_t + \alpha (y_i - \hat{p}(x_i)) x_i$ 
6:   end for
7: end while
```

L2 regularization

$$\ell(\beta) = \underbrace{\sum_{i=1}^N \log P(y_i | x_i, \beta)}_{\text{we want this to be high}} - \underbrace{\eta \sum_{j=1}^F \beta_j^2}_{\text{but we want this to be small}}$$

- We can do this by changing the function we're trying to optimize by adding a penalty for having values of β that are high
- This is equivalent to saying that each β element is drawn from a Normal distribution centered on 0.
- η controls how much of a penalty to pay for coefficients that are far from 0 (optimize on development data)

$$\frac{P(y \mid x, \beta)}{1 - P(y \mid x, \beta)} = \exp(x_0 \beta_0) \exp(x_1 \beta_1)$$

Let's increase the value of x by 1 (e.g., from 0 → 1)

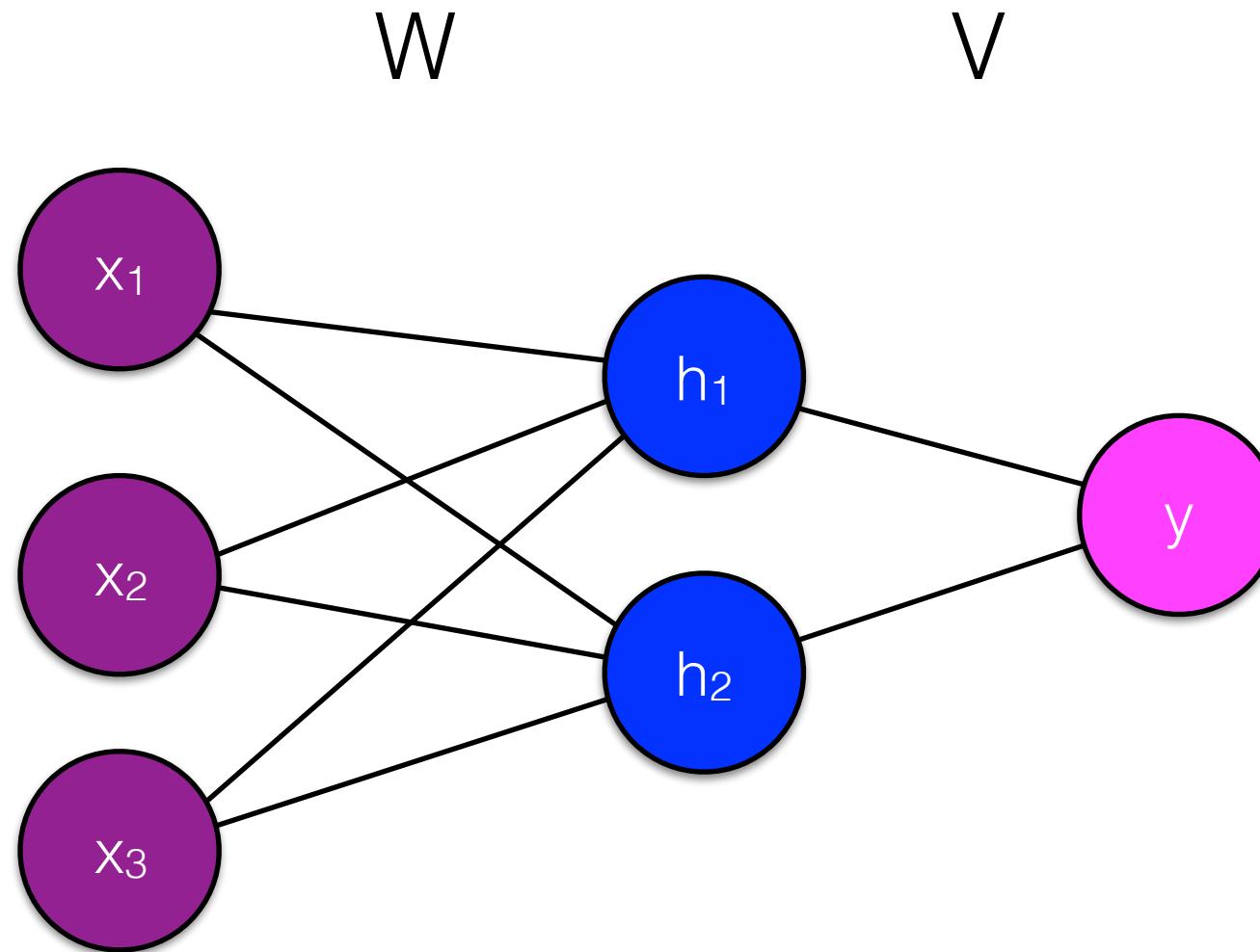
$$\exp(x_0 \beta_0) \exp((x_1 + 1) \beta_1)$$

$$\exp(x_0 \beta_0) \exp(x_1 \beta_1 + \beta_1)$$

$$\exp(x_0 \beta_0) \exp(x_1 \beta_1) \exp(\beta_1)$$

$\exp(\beta)$ represents the factor by which the **odds** change with a 1-unit increase in x

$$\frac{P(y \mid x, \beta)}{1 - P(y \mid x, \beta)} \exp(\beta_1)$$



$$\hat{y} = \sigma \left[V_1 \left(\sigma \left(\sum_i^F x_i W_{i,1} \right) \right) + V_2 \left(\sigma \left(\sum_i^F x_i W_{i,2} \right) \right) \right]$$

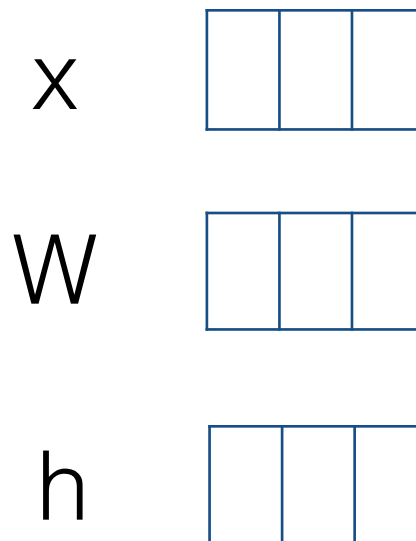
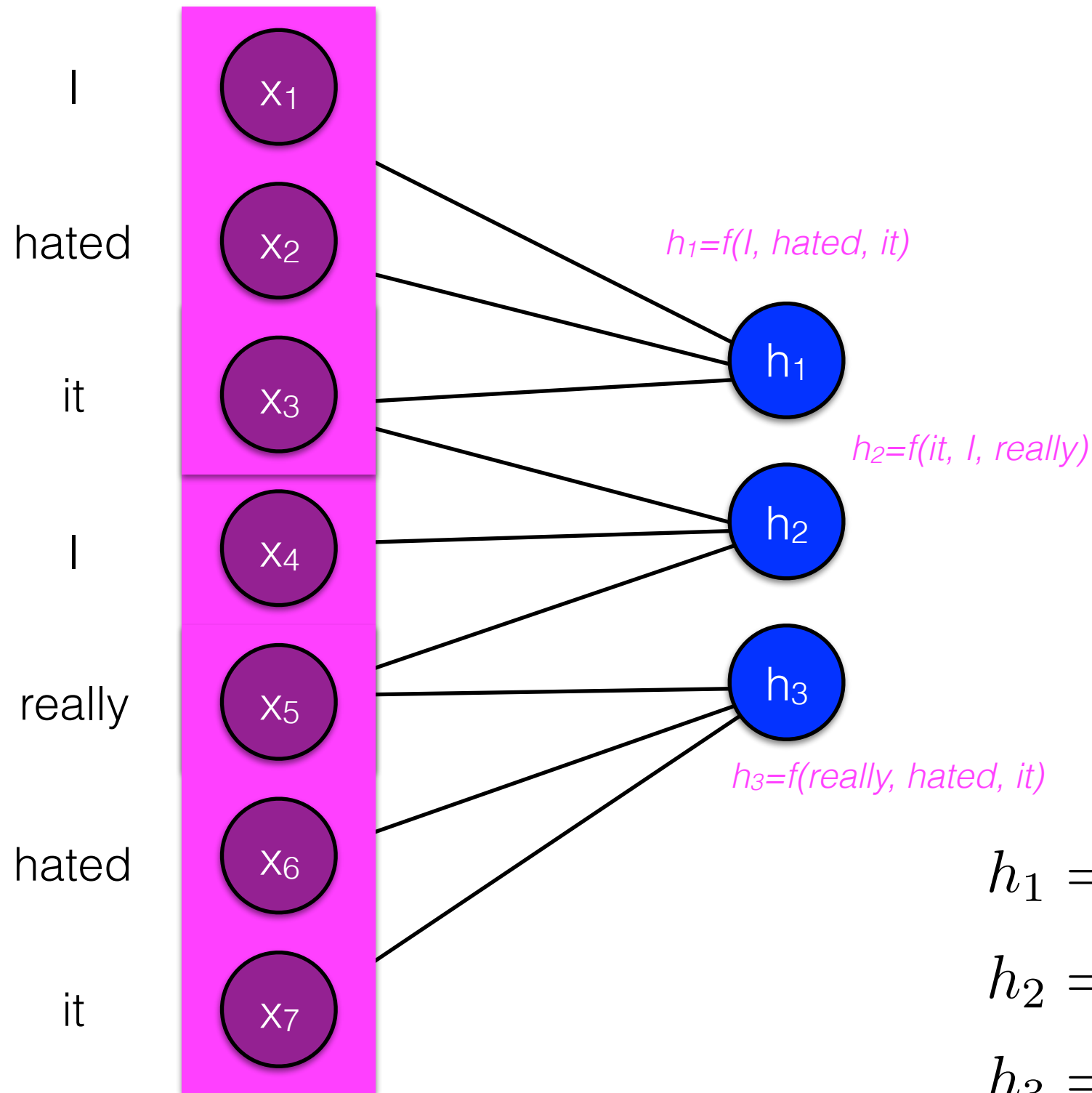
we can express y as a function only of the input x and the weights W and V

$$\hat{y} = \sigma \left[V_1 \underbrace{\left(\sigma \left(\sum_i^F x_i W_{i,1} \right) \right)}_{h_1} + V_2 \underbrace{\left(\sigma \left(\sum_i^F x_i W_{i,2} \right) \right)}_{h_2} \right]$$

This is hairy, but differentiable

Backpropagation: Given training samples of $\langle x, y \rangle$ pairs, we can use stochastic gradient descent to find the values of W and V that minimize the loss.

Convolutional networks

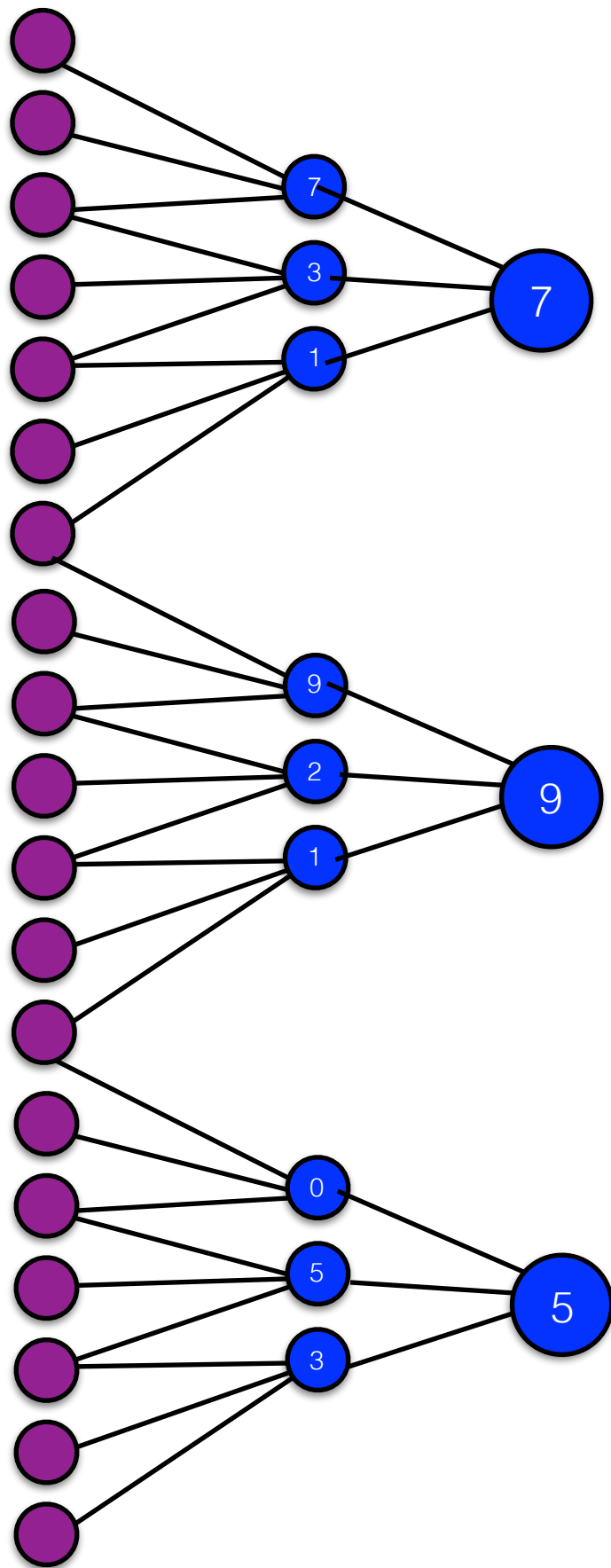


$$h_1 = \sigma(x_1 W_1 + x_2 W_2 + x_3 W_3)$$

$$h_2 = \sigma(x_3 W_1 + x_4 W_2 + x_5 W_3)$$

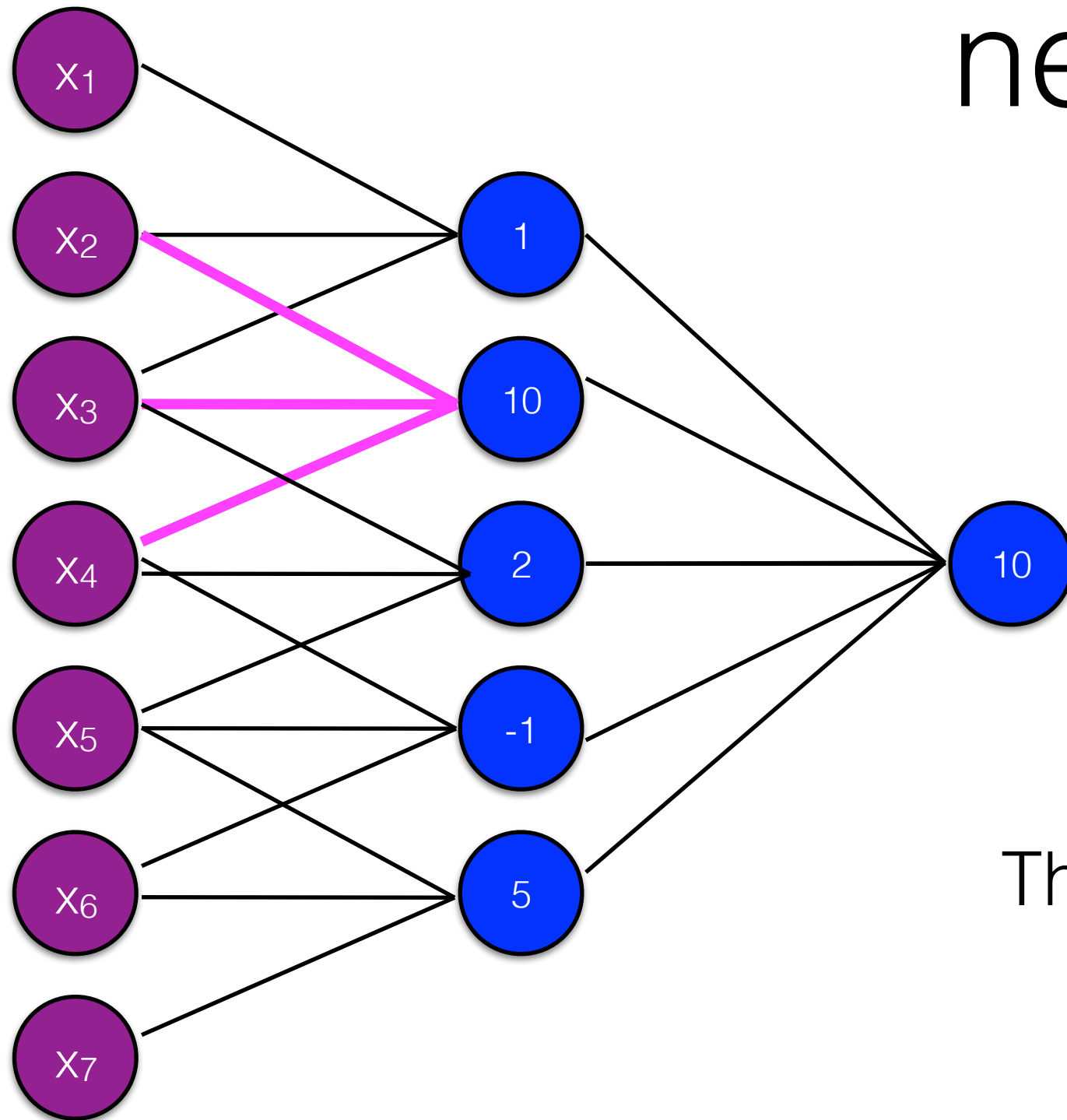
$$h_3 = \sigma(x_5 W_1 + x_6 W_2 + x_7 W_3)$$

Pooling



- Down-samples a layer by selecting a single point from some set
- **Max-pooling** selects the largest value

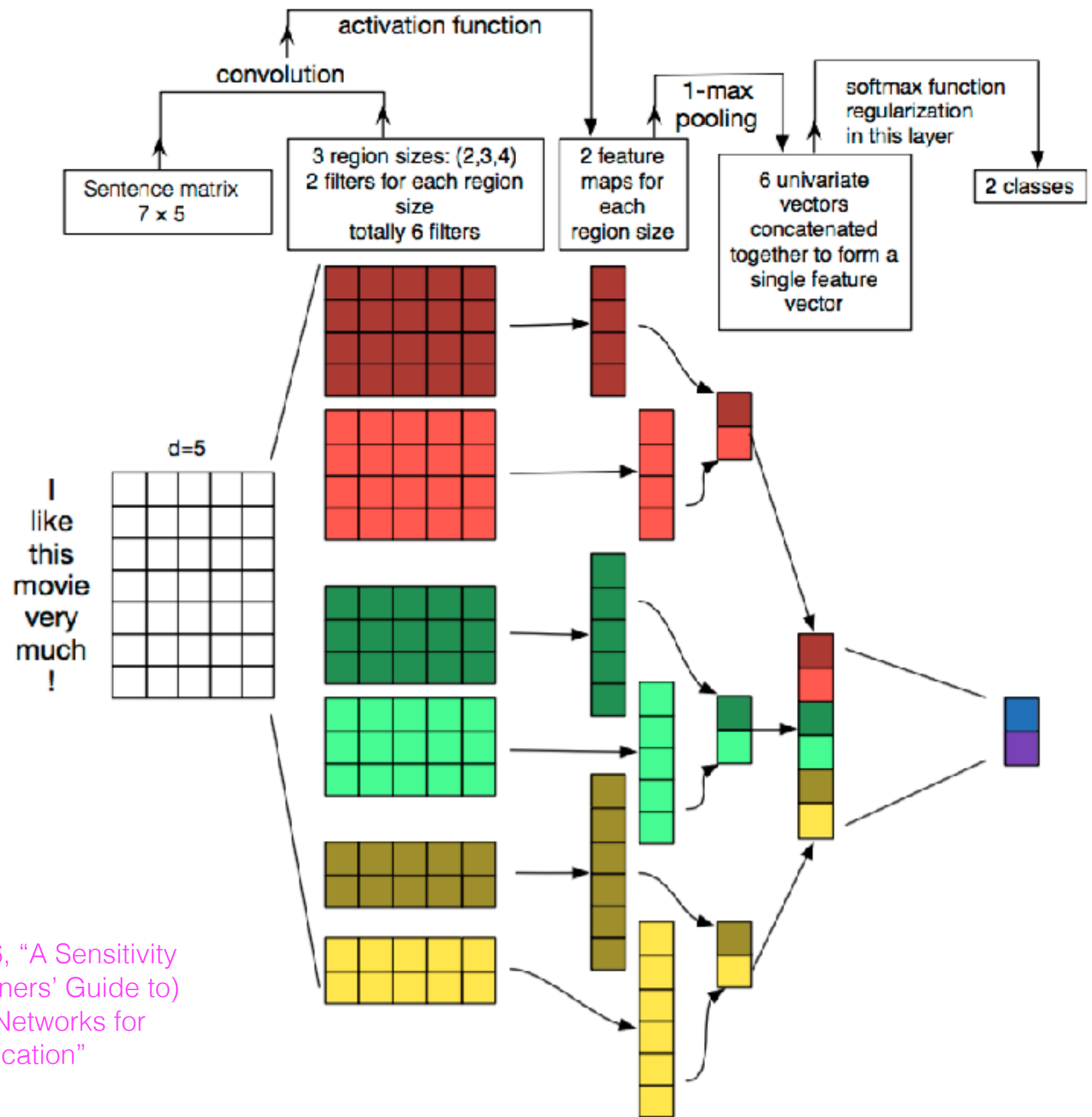
Convolutional networks



This defines one *filter*.

convolution

max pooling



Zhang and Wallace 2016, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification"

Language Model

- Language models provide us with a way to quantify the likelihood of sequence — i.e., **plausible** sentences.

OCR

To see great *Pompey* passe the streets of Rome :
And when you saw his Chariot but appeare,
Haue you not made an Vniuersall shout,
That Tyber trembled vnderneath her bankes
To heare the replication of your sounds,
Made in her Concaue Shores?

- to fee great Pompey passe the Areets of Rome:
- to see great Pompey passe the streets of Rome:

Machine translation

Italian - detected ▾  	English ▾  
Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura, ché la diritta via era smarrita. Edit	In the middle of the walk of our lives I found myself in a dark forest, as the straight way was lost.

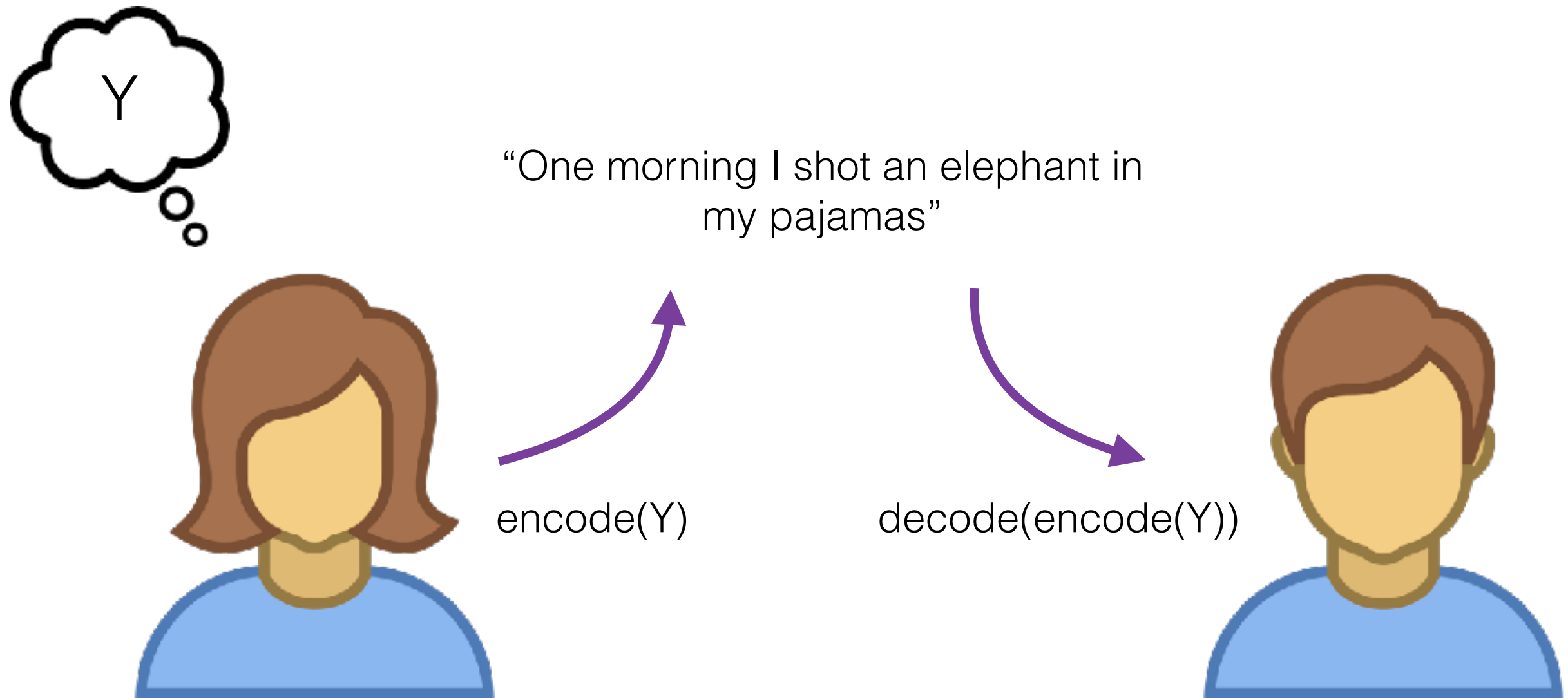
- Fidelity (to source text)
- Fluency (of the translation)

Speech Recognition



- 'Scuse me while I kiss the sky.
- 'Scuse me while I kiss this guy
- 'Scuse me while I kiss this fly.
- 'Scuse me while my biscuits fry

Information theoretic view



Noisy Channel

	X	Y
ASR	speech signal	transcription
MT	target text	source text
OCR	pixel densities	transcription

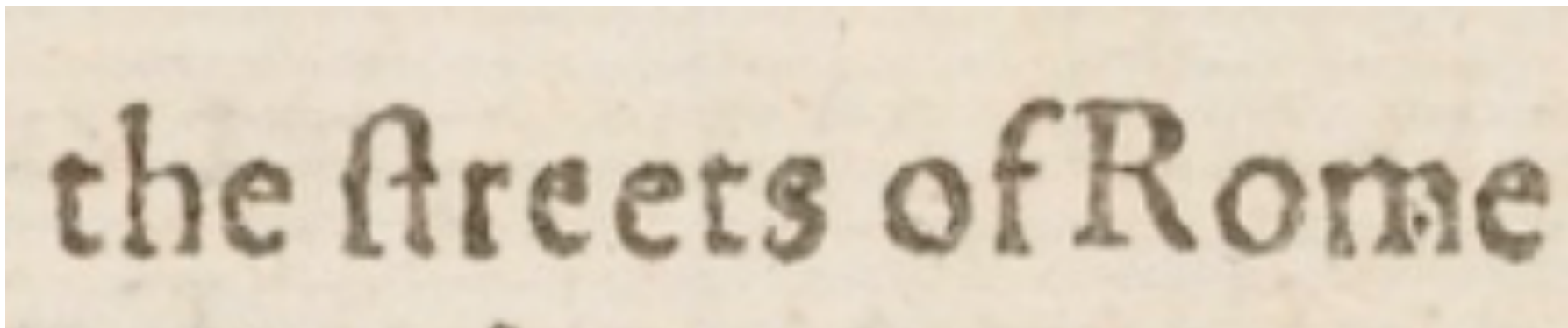
$$P(Y | X) \propto \underbrace{P(X | Y)}_{\text{channel model}} \underbrace{P(Y)}_{\text{source model}}$$

OCR

To see great *Pompey* passe the streets of Rome :
And when you saw his Chariot but appeare,
Haue you not made an Vniuersall shout,
That Tyber trembled vnderneath her bankes
To heare the replication of your sounds,
Made in her Concaue Shores?

$$P(Y | X) \propto \underbrace{P(X | Y)}_{\text{channel model}} \underbrace{P(Y)}_{\text{source model}}$$

OCR



$$P(Y | X) \propto \underbrace{P(X | Y)}_{\text{channel model}} \underbrace{P(Y)}_{\text{source model}}$$

Language Model

- Language modeling is the task of estimating $P(w)$
- Why is this hard?

$P(\text{"It was the best of times, it was the worst of times"})$

Chain rule (of probability)

$$\begin{aligned} P(x_1, x_2, x_3, x_4, x_5) &= P(x_1) \\ &\times P(x_2 \mid x_1) \\ &\times P(x_3 \mid x_1, x_2) \\ &\times P(x_4 \mid x_1, x_2, x_3) \\ &\times P(x_5 \mid x_1, x_2, x_3, x_4) \end{aligned}$$

Markov assumption

bigram model
(first-order markov)

$$\prod_i^n P(w_i \mid w_{i-1}) \times P(\text{STOP} \mid w_n)$$

trigram model
(second-order markov)

$$\prod_i^n P(w_i \mid w_{i-2}, w_{i-1}) \\ \times P(\text{STOP} \mid w_{n-1}, w_n)$$

Smoothing LM

- Additive smoothing; Laplace smoothing
- Interpolating LMs of different orders
- Kneser-Ney
- Stupid backoff

Featurized LMs

- We can use multi class logistic regression for language modeling by treating the vocabulary as the output space

$$\mathcal{Y} = \mathcal{V}$$

Featurized LMs

$$P(w_i = \text{dog} \mid w_{i-2} = \text{and}, w_{i-1} = \text{the})$$

second-order
features

first-order
features

Feature	Value
$w_{i-2}=\text{the} \wedge w_{i-1}=\text{the}$	0
$w_{i-2}=\text{and} \wedge w_{i-1}=\text{the}$	1
$w_{i-2}=\text{bravest} \wedge w_{i-1}=\text{the}$	0
$w_{i-2}=\text{love} \wedge w_{i-1}=\text{the}$	0
$w_{i-1}=\text{the}$	1
$w_{i-1}=\text{and}$	0
$w_{i-1}=\text{bravest}$	0
$w_{i-1}=\text{love}$	0
<i>BIAS</i>	1

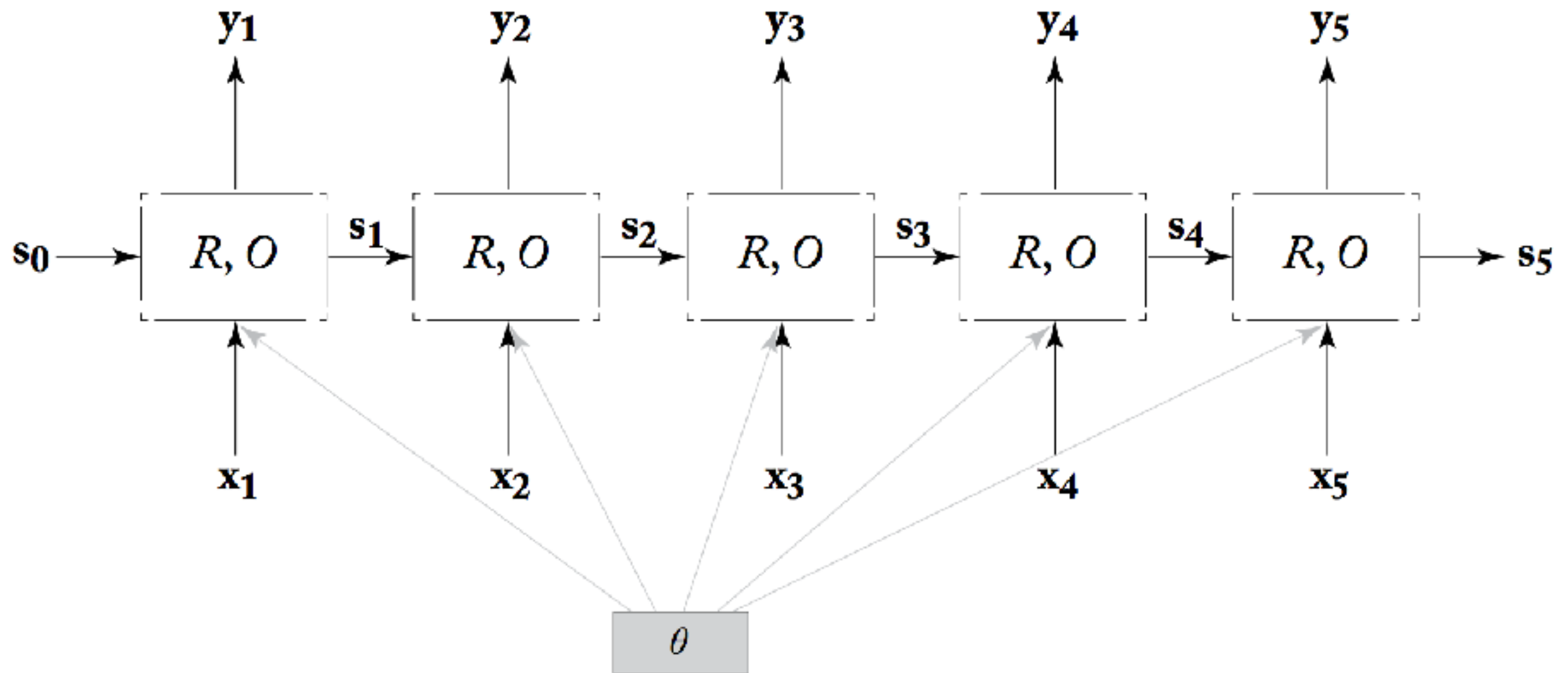
Richer representations

- Log-linear models give us the flexibility of encoding richer representations of the **context** we are conditioning on.
- We can reason about any observations from the entire history and not just the local context.

“Hillary Clinton seemed to add Benghazi to her already-long list of culprits to blame for her upset loss to Donald _____”

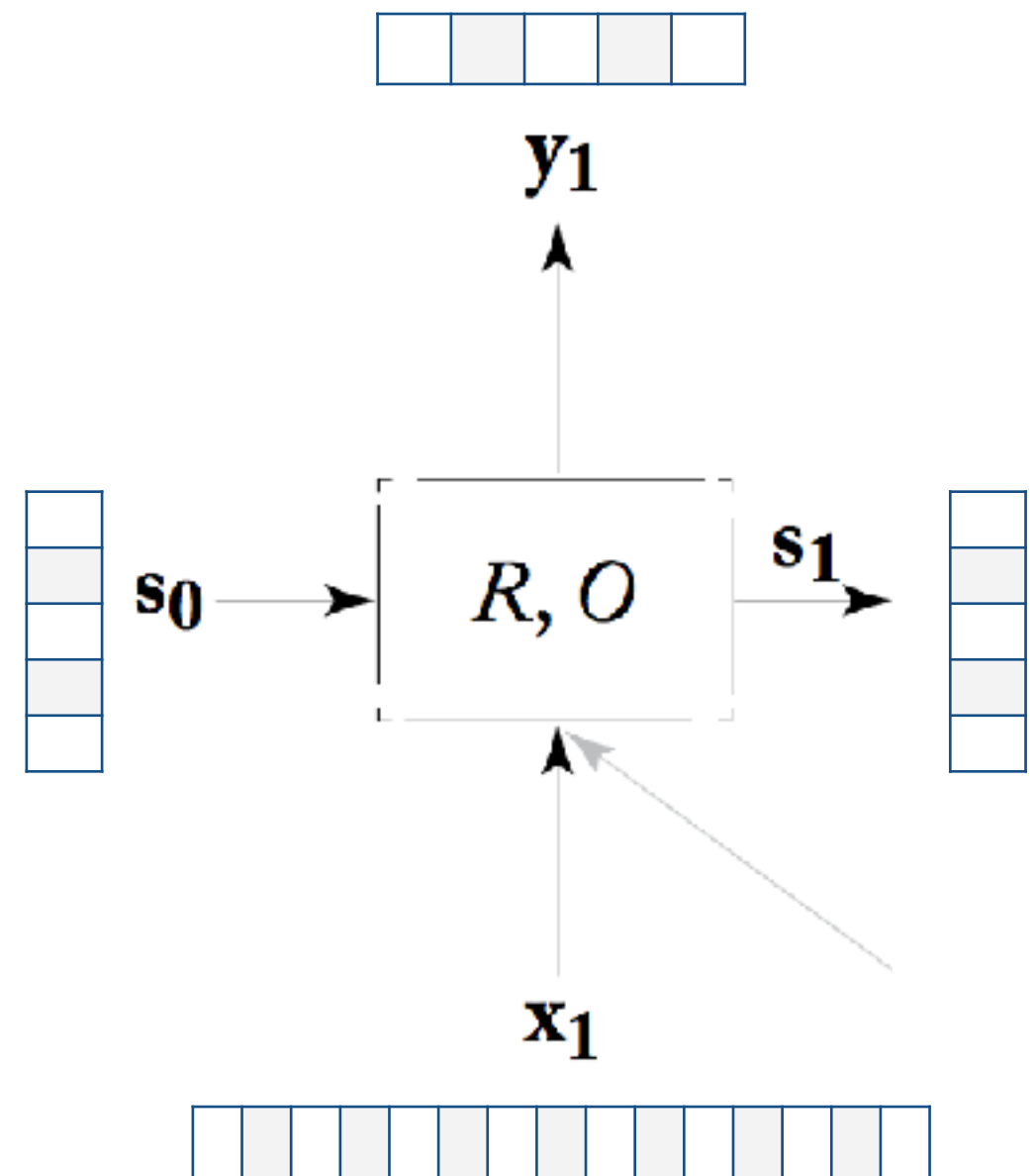
feature classes	example
ngrams (w_{i-1} , $w_{i-2}:w_{i-1}$, $w_{i-3}:w_{i-1}$)	w_{i-2} ="to", w_i ="donald"
gappy ngrams	w_1 ="hillary" and w_{i-1} ="donald"
spelling, capitalizaiton	w_{i-1} is capitalized and w_i is capitalized
class/gazetteer membership	w_{i-1} in list of names and w_i in list of names

Recurrent neural network



Recurrent neural network

- Each time step has two inputs:
 - x_i (the observation at time step i); one-hot vector, feature vector or distributed representation.
 - s_{i-1} (the output of the previous state); base case: $s_0 = 0$ vector



Distributed representations

...	...
the	1
a	0
an	0
for	0
in	0
on	0
dog	0
cat	0
...	...

the is a point in V-dimensional space

the

4.1
-0.9

the is a point in 2-dimensional space

Distributed representations

- Low-dimensional, dense word representations are extraordinarily powerful (and are arguably responsible for much of gains that neural network models have in NLP).
- Lets your representation of the input share statistical strength with words that behave similarly in terms of their distributional properties (often **synonyms** or words that belong to the same **class**).

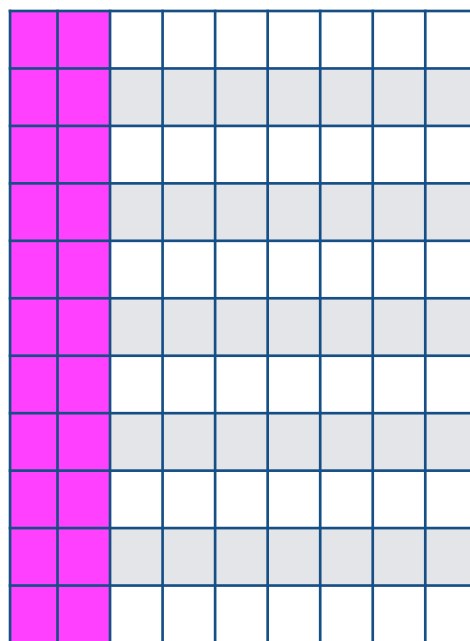
Term-term matrix

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Figure 15.4 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

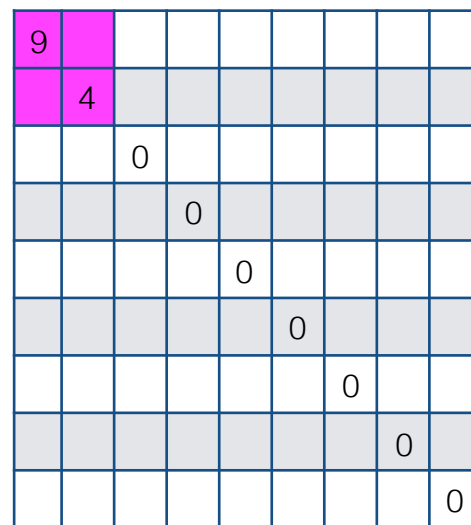
Singular value decomposition

- We can **approximate** the full matrix by only considering the leftmost k terms in the diagonal matrix (the k largest singular values)



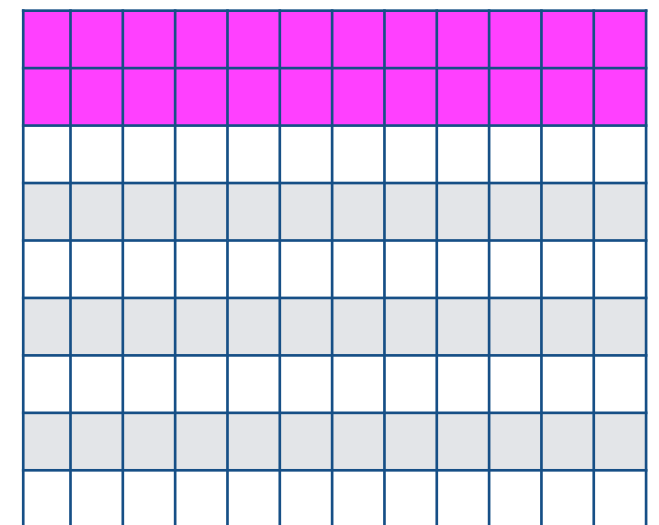
$n \times m$

$\times$



$m \times m$

$\times$



$m \times p$

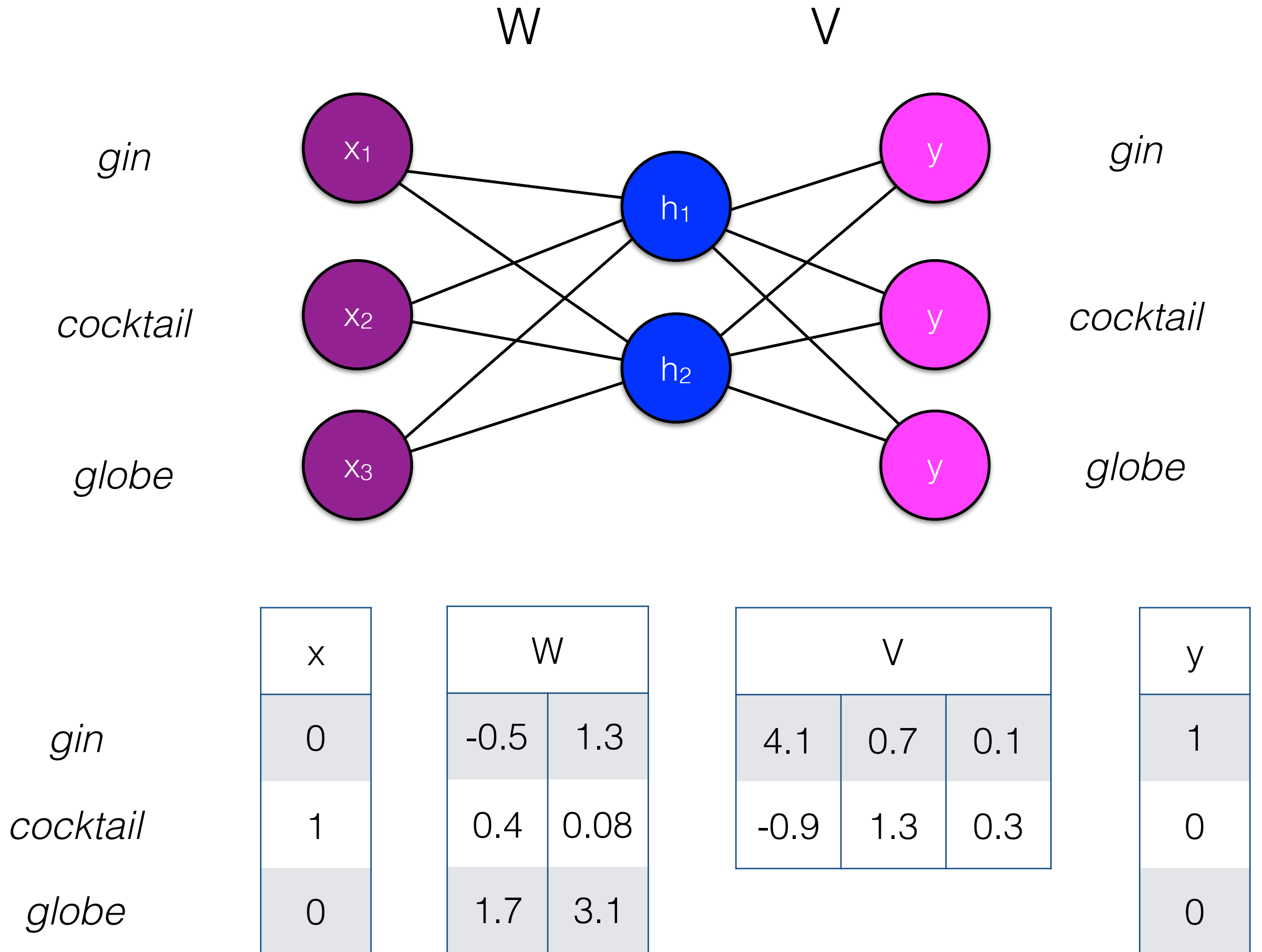
	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear
knife	1	1	4	2		2		2
dog	2		6	6		2		12
sword	17	2	7	12		2		17
love	64		135	63		12		48
like	75	38	34	36	34	41	27	44

knife		
dog		
sword		
love		
like		

Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear

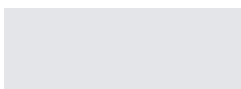
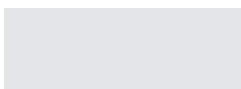
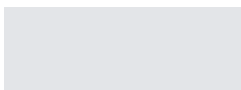
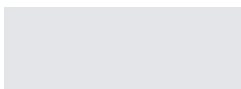
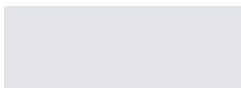
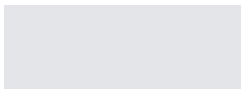
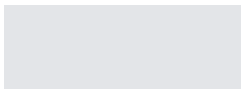
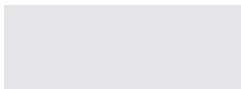
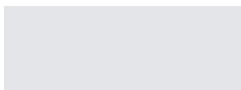
Dense vectors from prediction

- Learning low-dimensional representations of words by framing a predicting task: using context to predict words in a surrounding window
- Transform this into a supervised prediction problem; similar to language modeling but we're ignoring order within the context window



X

W



0.13	0.56
-1.75	0.07

0.80	1.19
-0.11	1.38

-0.62	-1.46
-1.16	-1.24

0.99	-0.26
-1.46	-0.85

0.79	0.47
0.06	-1.21

-0.31	0.00
-1.01	-2.52

-1.50	-0.14
-0.14	0.01

-0.13	-1.76
-1.08	-0.56

-0.17	-0.74
0.31	1.03

-0.24	-0.84
-0.79	-0.18

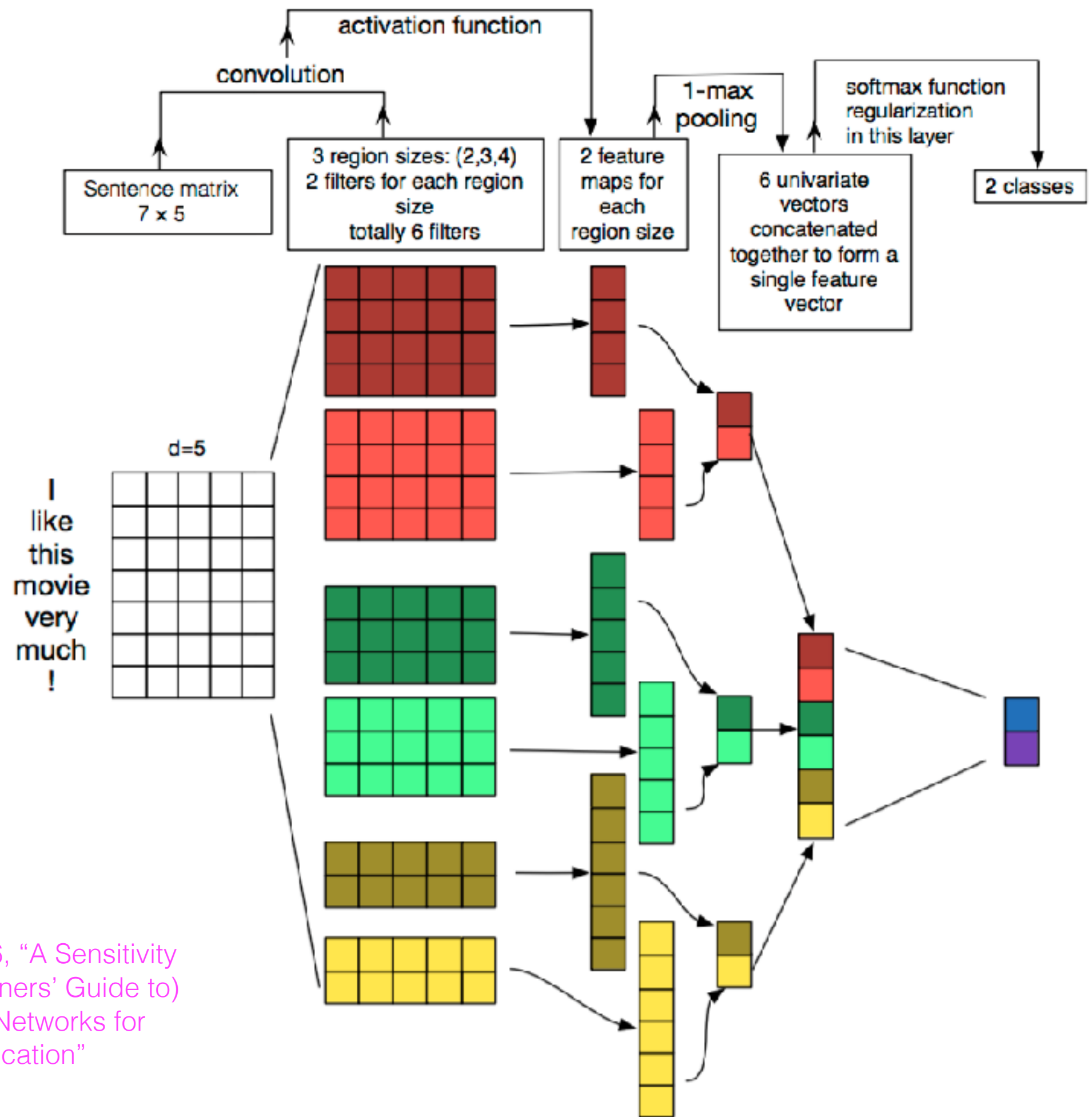
$$x^{\top}W =$$

-1.01	-2.52
-------	-------

This is the
embedding of the
context

Using dense vectors

- In neural models (CNNs, RNNs, LM), replace the V -dimensional sparse vector with the much smaller K -dimensional dense one.
- Can also take the derivative of the loss function with respect to those representations to optimize for a particular task.



Zhang and Wallace 2016, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification"

Parts of speech

- Parts of speech are categories of words defined **distributionally** by the morphological and syntactic contexts a word appears in.

	-s	-ed	-ing
walk	walks	walked	walking
slice	slices	sliced	slicing
believe	believes	believed	believing
of	*ofs	*ofed	*ofing
red	*reds	*redded	*reding

Kim saw the	elephant	before we did
	dog	
	idea	
	*of	
	*goes	

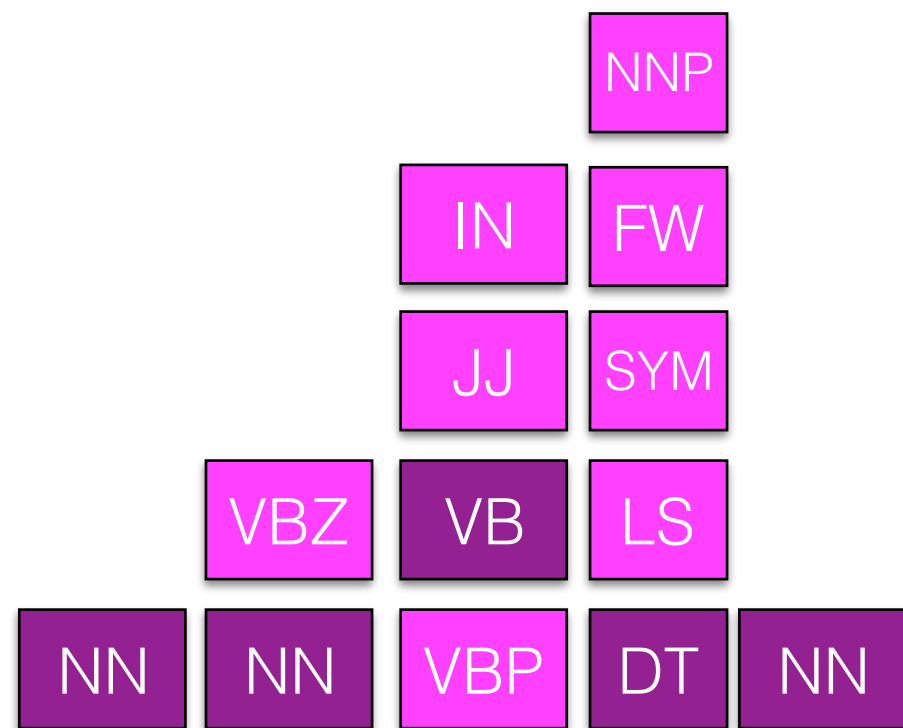
Open class

Nouns	fax, affluenza, subtweet, bitcoin, cronut, emoji, listicle, mocktail, selfie, skort
Verbs	text, chillax, manspreading, photobomb, unfollow, google
Adjectives	crunk, amazeballs, post-truth, woke
Adverbs	hella, wicked
Determiner	
Pronouns	
Prepositions	English has a new preposition, because internet [Garber 2013; Pullum 2014]
Conjunctions	

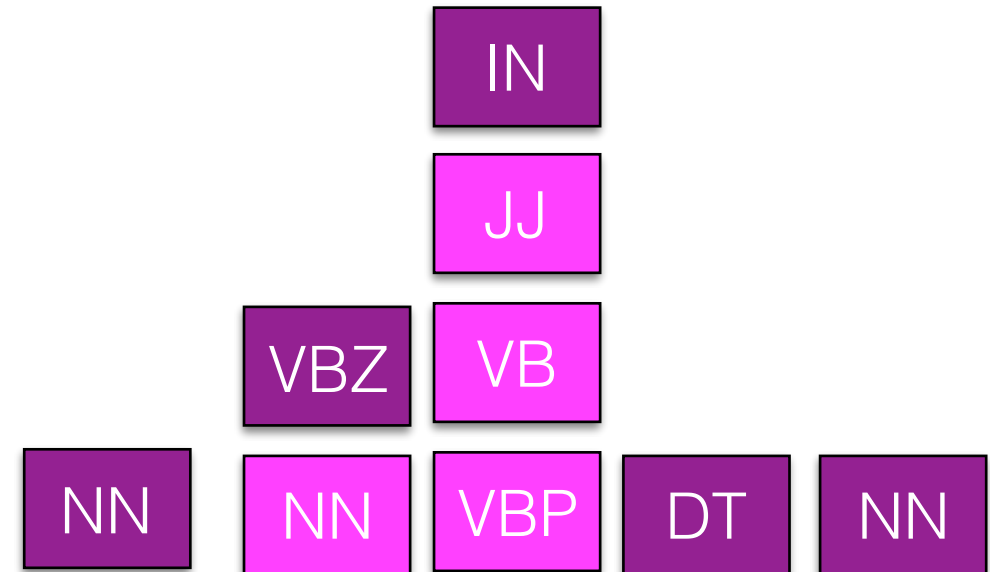
Closed class

POS tagging

Labeling the tag that's correct
for the context.



Fruit flies like a banana



Time flies like an arrow

(Just tags in evidence within the Penn Treebank — more are possible!)

Why is part of speech tagging useful?

Sequence labeling

$$x = \{x_1, \dots, x_n\}$$

$$y = \{y_1, \dots, y_n\}$$

- For a set of inputs x with n sequential time steps, one corresponding label y_i for each x_i

Majority class

- Pick the label each word is seen most often with in the training data

fruit	flies	like	a	banana
NN 12	VBZ 7	VB 74	FW 8	NN 3
	NNS 1	VBP 31	SYM 13	
		JJ 28	LS 2	
		IN 533	JJ 2	
			IN 1	
			DT 25820	
			NNP 2	

Naive Bayes

- Treat each prediction as independent of the others

$$P(y \mid x) = \frac{P(y)P(x \mid y)}{\sum_{y' \in \mathcal{Y}} P(y')P(x \mid y')}$$

$$P(\text{VBZ} \mid \text{flies}) = \frac{P(\text{VBZ})P(\text{flies} \mid \text{VBZ})}{\sum_{y' \in \mathcal{Y}} P(y')P(\text{flies} \mid y')}$$

Logistic regression

- Treat each prediction as independent of the others but condition on much more expressive set of features

$$P(y \mid x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

$$P(\text{VBZ} \mid \text{flies}) = \frac{\exp(x^\top \beta_{\text{VBZ}})}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

Sequences

- Models that make independent predictions for elements in a sequence can reason over expressive representations of the **input** x (including correlations among inputs at different time steps x_i and x_j).
- But they don't capture another important source of information: correlations in the **labels** y .

HMM

$$P(x_1, \dots, x_n, y_1, \dots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1}) \prod_{i=1}^n P(x_i \mid y_i)$$

Hidden Markov Model

Prior probability of label sequence

$$P(y) = P(y_1, \dots, y_n)$$

$$P(y_1, \dots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1})$$

- We'll make a first-order Markov assumption and calculate the joint probability as the product the individual factors conditioned **only on the previous tag**.

Hidden Markov Model

$$\begin{aligned} P(y_1, \dots, y_n) &= P(y_1) \\ &\times P(y_2 \mid y_1) \\ &\times P(y_3 \mid y_1, y_2) \\ &\dots \\ &\times P(y_n \mid y_1, \dots, y_{n-1}) \end{aligned}$$

- Remember: a Markov assumption is an approximation to this **exact** decomposition (the chain rule of probability)

Hidden Markov Model

$$P(x \mid y) = P(x_1, \dots, x_n \mid y_1, \dots, y_n)$$

$$P(x_1, \dots, x_n \mid y_1, \dots, y_n) \approx \prod_{i=1}^N P(x_i \mid y_i)$$

- Here again we'll make a strong assumption: the probability of the word we see at a given time step is only dependent on its label

Parameter estimation

$$P(y_t \mid y_{t-1}) \qquad \frac{c(y_1, y_2)}{c(y_1)}$$

MLE for both is just counting
(as in Naive Bayes)

$$P(x_t \mid y_t) \qquad \frac{c(x, y)}{c(y)}$$

Transition probabilities

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 10.5 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

Emission probabilities

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 10.6 Observation likelihoods B computed from the WSJ corpus without smoothing.

function VITERBI(*observations* of len T , *state-graph* of len N) **returns** *best-path*

create a path probability matrix $viterbi[N+2, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$viterbi[s, 1] \leftarrow a_{0,s} * b_s(o_1)$

$backpointer[s, 1] \leftarrow 0$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$viterbi[s, t] \leftarrow \max_{s'=1}^N viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

$backpointer[s, t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s', t-1] * a_{s',s}$

$viterbi[q_F, T] \leftarrow \max_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

$backpointer[q_F, T] \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

return the backtrace path by following backpointers to states back in time from $backpointer[q_F, T]$

Figure 10.8 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence. Note that states 0 and q_F are non-emitting.

Decoding

- Greedy: proceed left to right, committing to the best tag for each time step (given the sequence seen so far)

Fruit flies like a banana

NN	VB	IN	DT	NN
----	----	----	----	----

MEMM

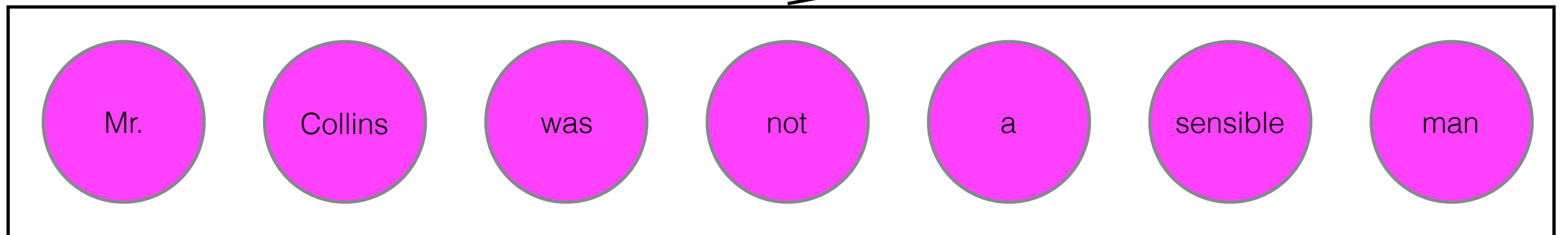
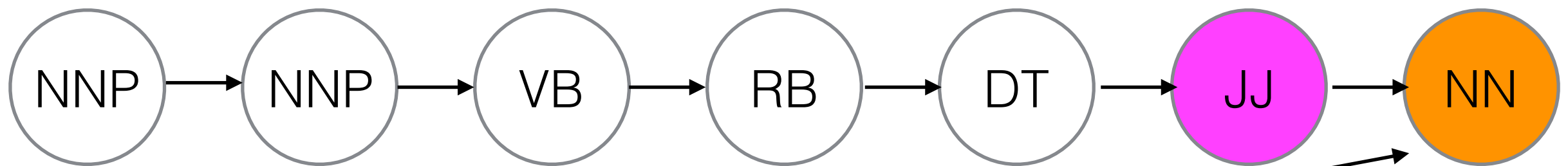
General maxent form

$$\arg \max_y P(y \mid x, \beta)$$

Maxent with first-order Markov
assumption: Maximum Entropy
Markov Model

$$\arg \max_y \prod_{i=1}^n P(y_i \mid y_{i-1}, x)$$

MEMM



Features

$$f(t_i, t_{i-1}; x_1, \dots, x_n)$$

Features are scoped over
the previous predicted
tag and the entire
observed input

feature	example
$x_i = \text{man}$	1
$t_{i-1} = \text{JJ}$	1
$i=n$ (last word of sentence)	1
x_i ends in -ly	0

Viterbi decoding

Viterbi for HMM: max joint probability

$$P(y)P(x \mid y) = P(x, y)$$

$$v_t(y) = \max_{u \in \mathcal{Y}} [v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u)P(x_t \mid y_t = y)]$$

Viterbi for MEMM: max conditional probability

$$P(y \mid x)$$

$$v_t(y) = \max_{u \in \mathcal{Y}} [v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u, x, \beta)]$$

MEMM Training

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

Locally normalized — at each time step,
each conditional distribution sums to 1

Label bias

NN TO VB

will to fight

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

Because of this **local normalization**, $P(\text{TO} \mid \text{context})$ will always be 1 if $x = \text{"to"}$

Label bias

NN TO VB

will to fight

That means our prediction for *to* can't help us disambiguate *will*. We lose the information that MB + TO sequences rarely happen.

Conditional random fields

- We can solve this problem using global normalization (over the entire sequences) rather than locally normalized factors.

MEMM

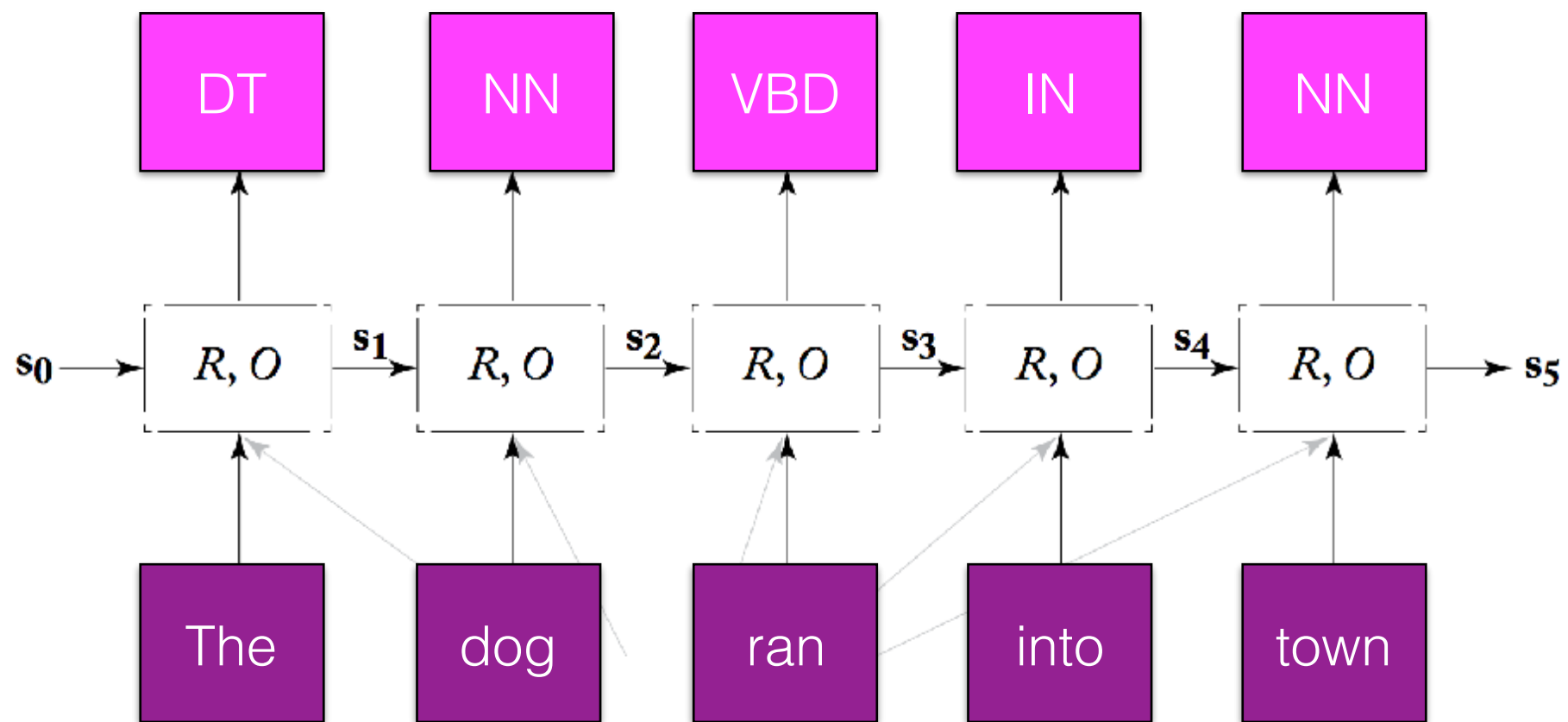
$$P(y \mid x, \beta) = \prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

CRF

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

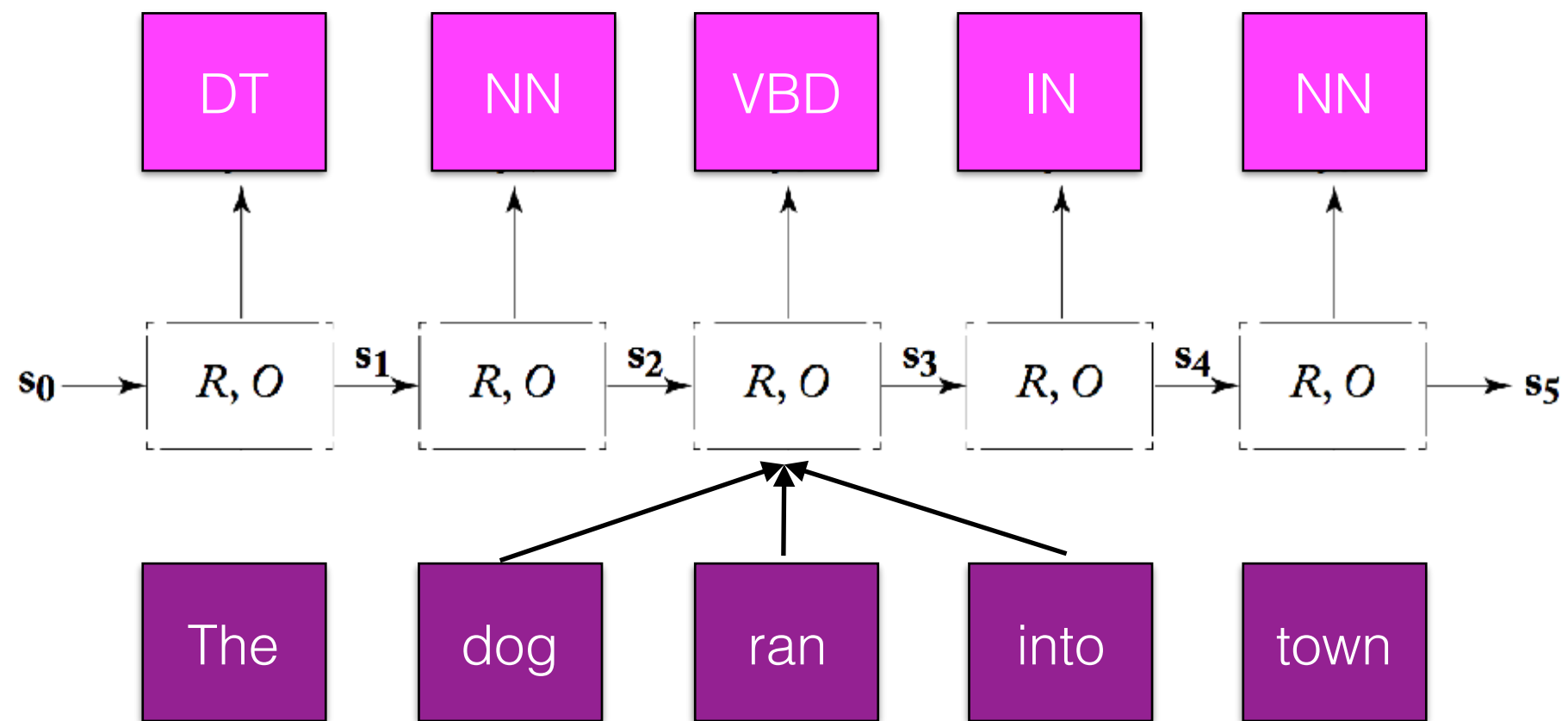
Recurrent neural network

- For POS tagging, predict the **tag** from ***y*** conditioned on the context



Recurrent neural network

- The simplest thing we could do is condition on a window of inputs



Bidirectional RNN

- A powerful alternative is make predictions conditioning both on the **past** and the **future**.
- Two RNNs
 - One running left-to-right
 - One right-to-left
- Each produces an output vector at each time step, which we concatenate

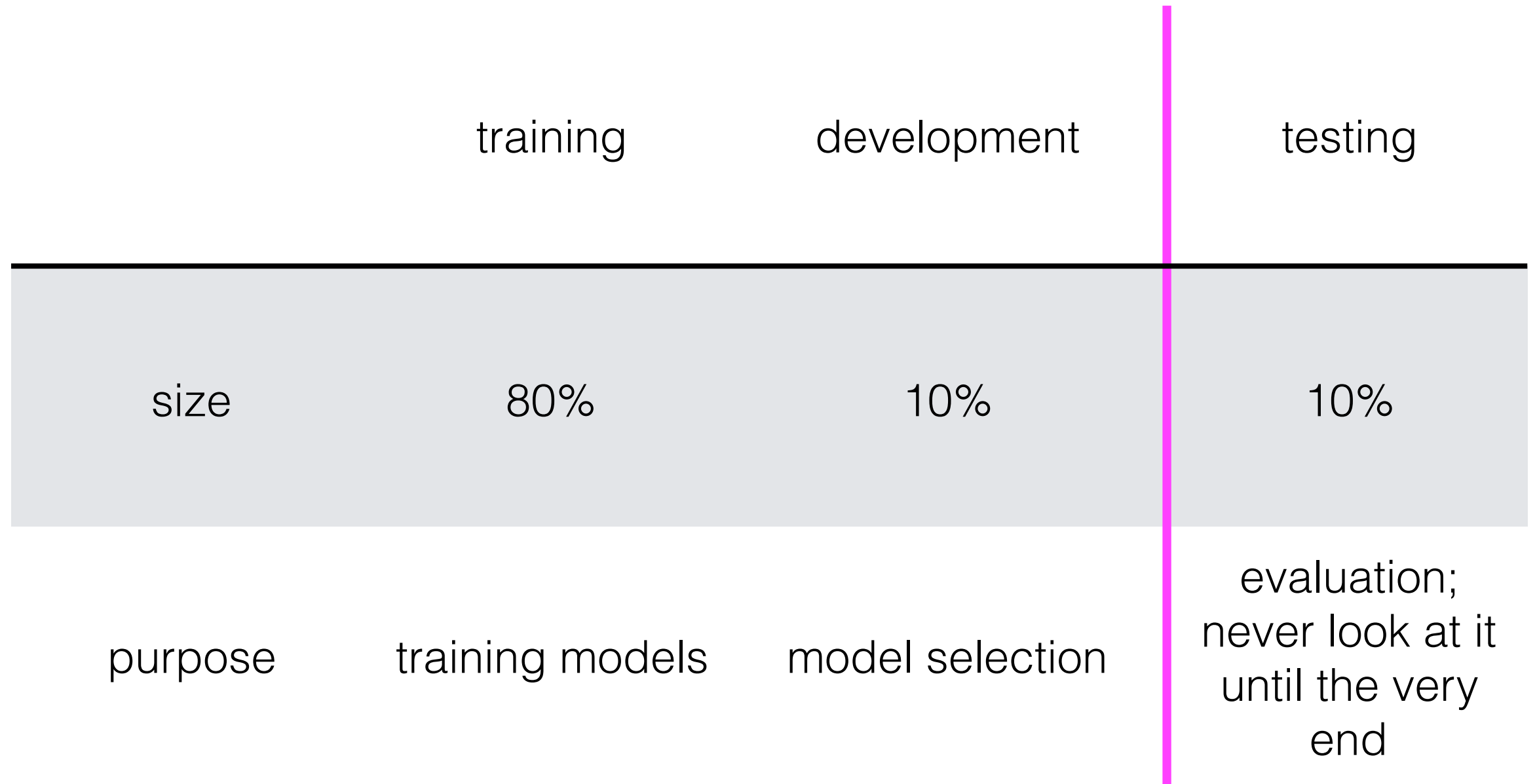
Significance in NLP

- You develop a new POS tagging algorithm; is it better than what comes before?
- You're developing a new model; should you include feature X? (when there is a cost to including it)
- You're developing a new model; does feature X reliably predict outcome Y?

Evaluation

- A critical part of development new algorithms and methods and demonstrating that they work

Experiment design



Accuracy

$$\frac{1}{N} \sum_{i=1}^N I[\hat{y}_i = y_i]$$

Predicted ($\hat{y}$)

NN

VBZ

JJ

$$I[x] \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

True (y)

	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

Precision

Precision(NN) =

$$\frac{\sum_{i=1}^N I(y_i = \hat{y}_i = \text{NN})}{\sum_{i=1}^N I(\hat{y}_i = \text{NN})}$$

Precision: proportion of predicted class that are actually that class.

True (y)

Predicted ($\hat{y}$)

	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

Recall

$$\text{Recall}(\text{NN}) = \frac{\sum_{i=1}^N I(y_i = \hat{y}_i = \text{NN})}{\sum_{i=1}^N I(y_i = \text{NN})}$$

Recall: proportion of true class that are predicted to be that class.

True (y)

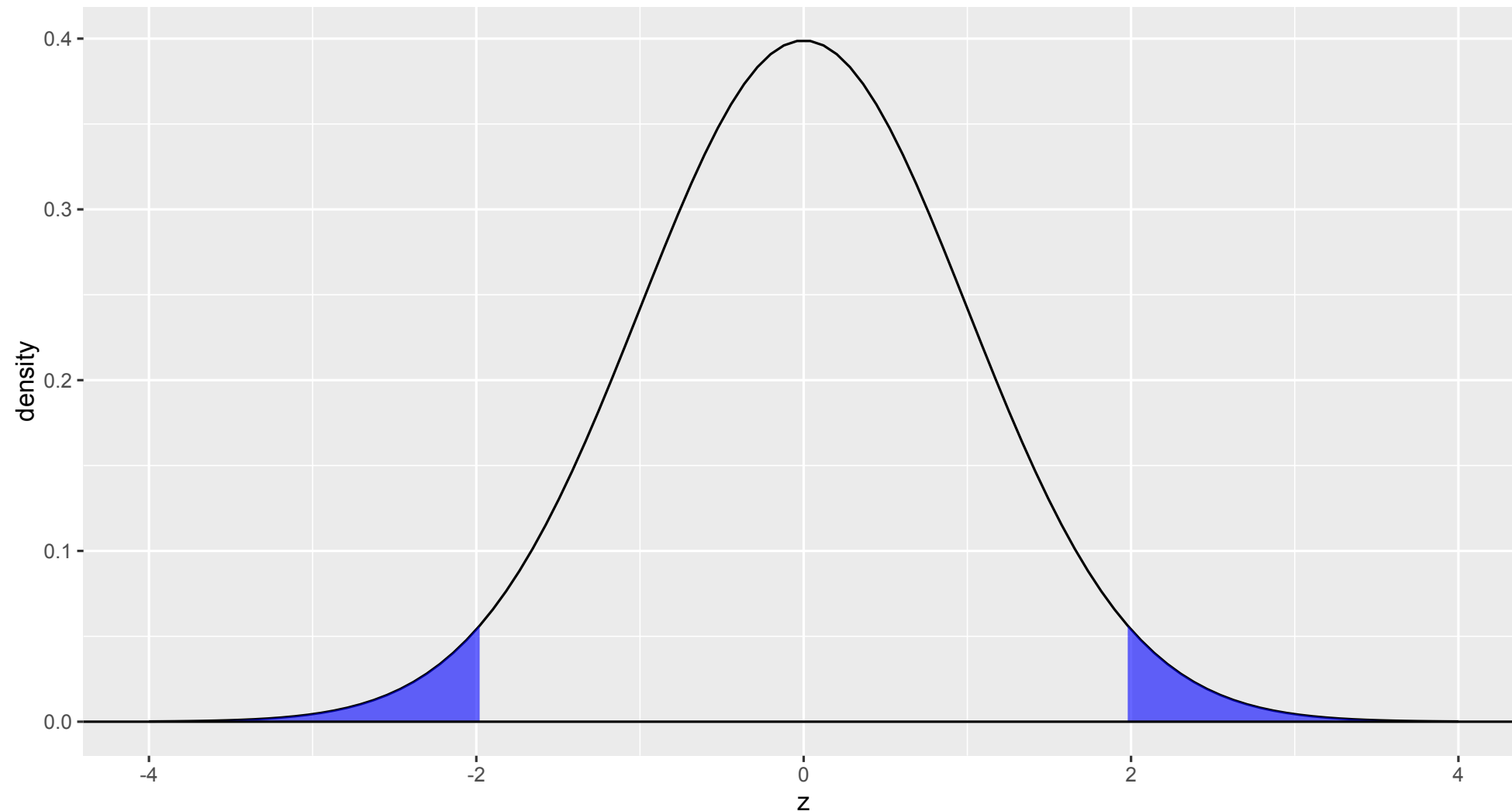
Predicted ($\hat{y}$)

	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

F score

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

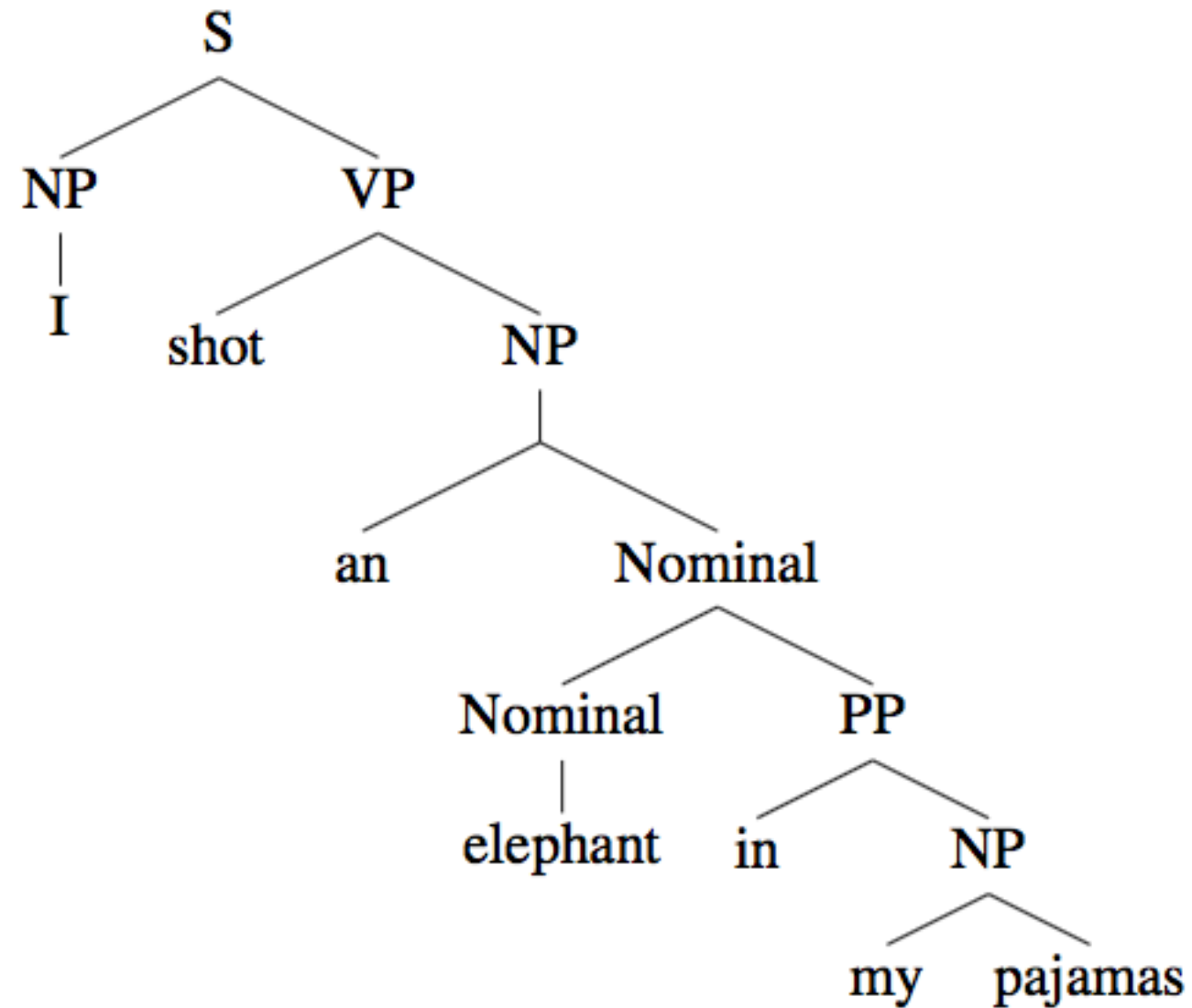
Hypothesis tests



Hypothesis tests measure how (un)likely an observed statistic is under the null hypothesis

Nonparametric tests

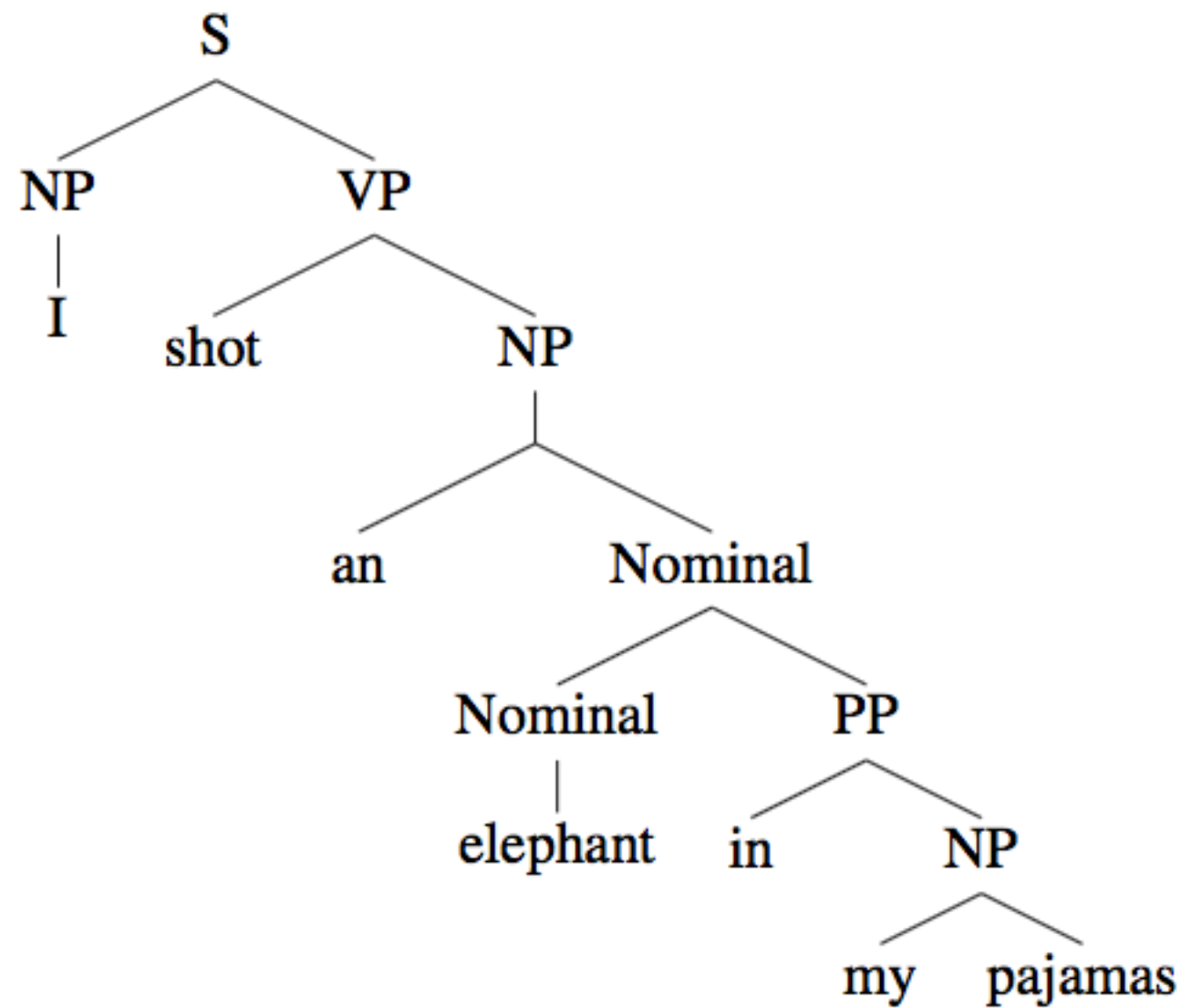
- Many hypothesis tests rely on parametric assumptions (e.g., normality)
- Alternatives that don't rely on those assumptions:
 - permutation test
 - the bootstrap



PRP VBD DT NN IN PRP\$ NNS

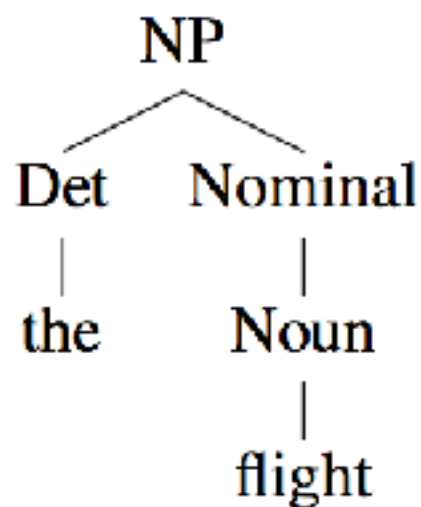
I shot an elephant in my pajamas

Why is syntax important?

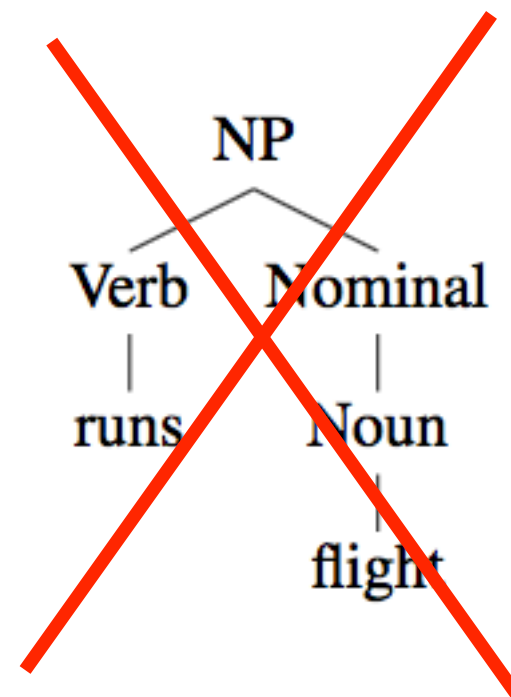


Context-free grammar

- A CFG gives a formal way to define what meaningful constituents are and exactly how a constituent is formed out of other constituents (or words). It defines **valid structure** in a language.

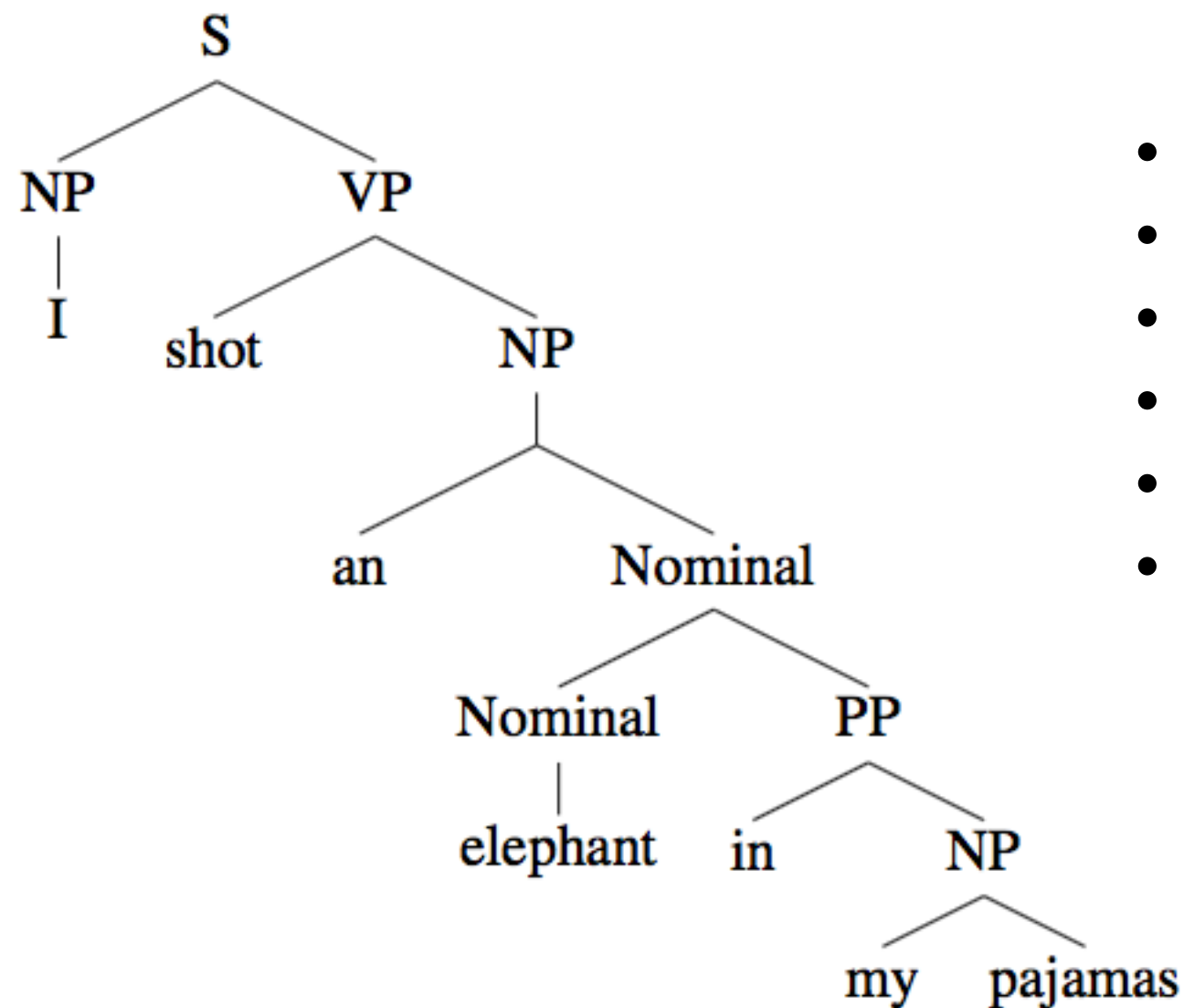


NP → Det Nominal



NP → Verb Nominal

Constituents



Every internal node is a phrase

- my pajamas
- in my pajamas
- elephant in my pajamas
- an elephant in my pajamas
- shot an elephant in my pajamas
- I shot an elephant in my pajamas

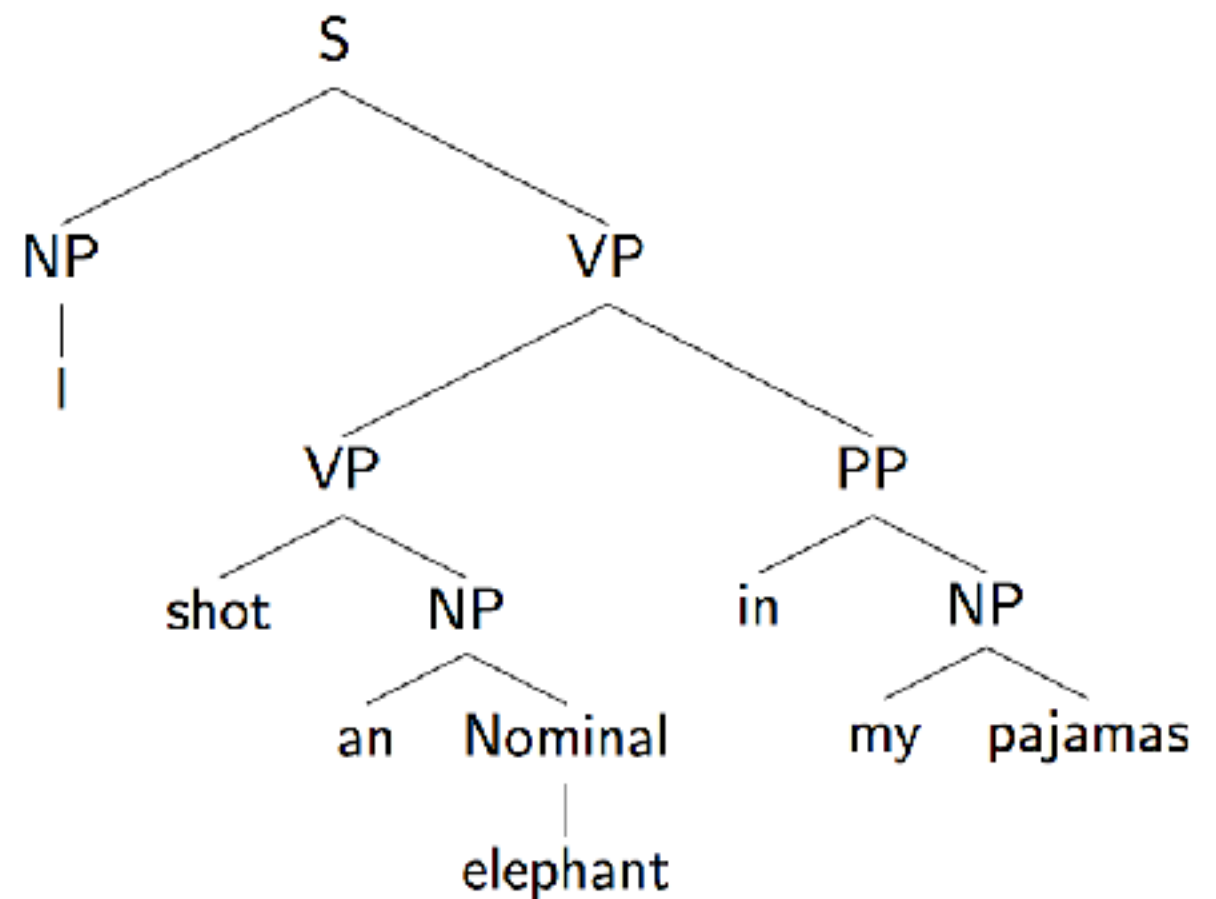
Each phrase could be replaced by another of the same type of constituent

Evaluation

Parseval (1991):
Represent each tree as a
collection of tuples:

$\langle l_1, i_1, j_1 \rangle, \dots, \langle l_n, i_n, j_n \rangle$

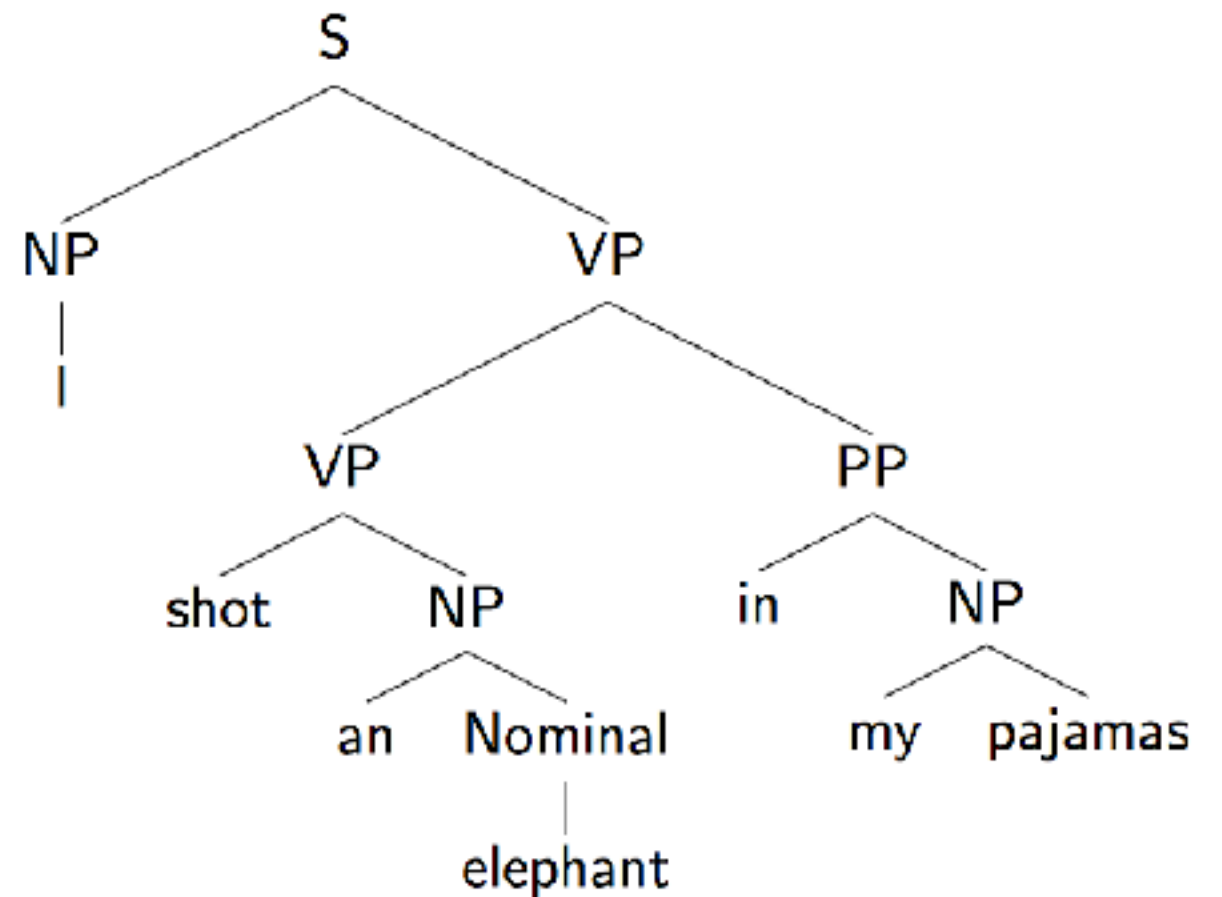
- l_k = label for k th phrase
- i_k = index for first word in z th phrase
- j_k = index for last word in k th phrase



Evaluation

I₁ shot₂ an₃ elephant₄ in₅ my₆ pajamas₇

- <S, 1, 7>
- <NP, 1, 1>
- <VP, 2, 7>
- <VP, 2, 4>
- <NP, 3, 4>
- <Nominal, 4, 4>
- <PP, 5, 7>
- <NP, 6, 7>



Evaluation

I_1 shot $_2$ an $_3$ elephant $_4$ in $_5$ my $_6$ pajamas $_7$

- $\langle S, 1, 7 \rangle$
- $\langle NP, 1, 1 \rangle$
- $\langle VP, 2, 7 \rangle$
- $\langle VP, 2, 4 \rangle$
- $\langle NP, 3, 4 \rangle$
- $\langle Nominal, 4, 4 \rangle$
- $\langle PP, 5, 7 \rangle$
- $\langle NP, 6, 7 \rangle$

- $\langle S, 1, 7 \rangle$
- $\langle NP, 1, 1 \rangle$
- $\langle VP, 2, 7 \rangle$
- $\langle NP, 3, 7 \rangle$
- $\langle Nominal, 4, 7 \rangle$
- $\langle Nominal, 4, 4 \rangle$
- $\langle PP, 5, 7 \rangle$
- $\langle NP, 6, 7 \rangle$

Evaluation

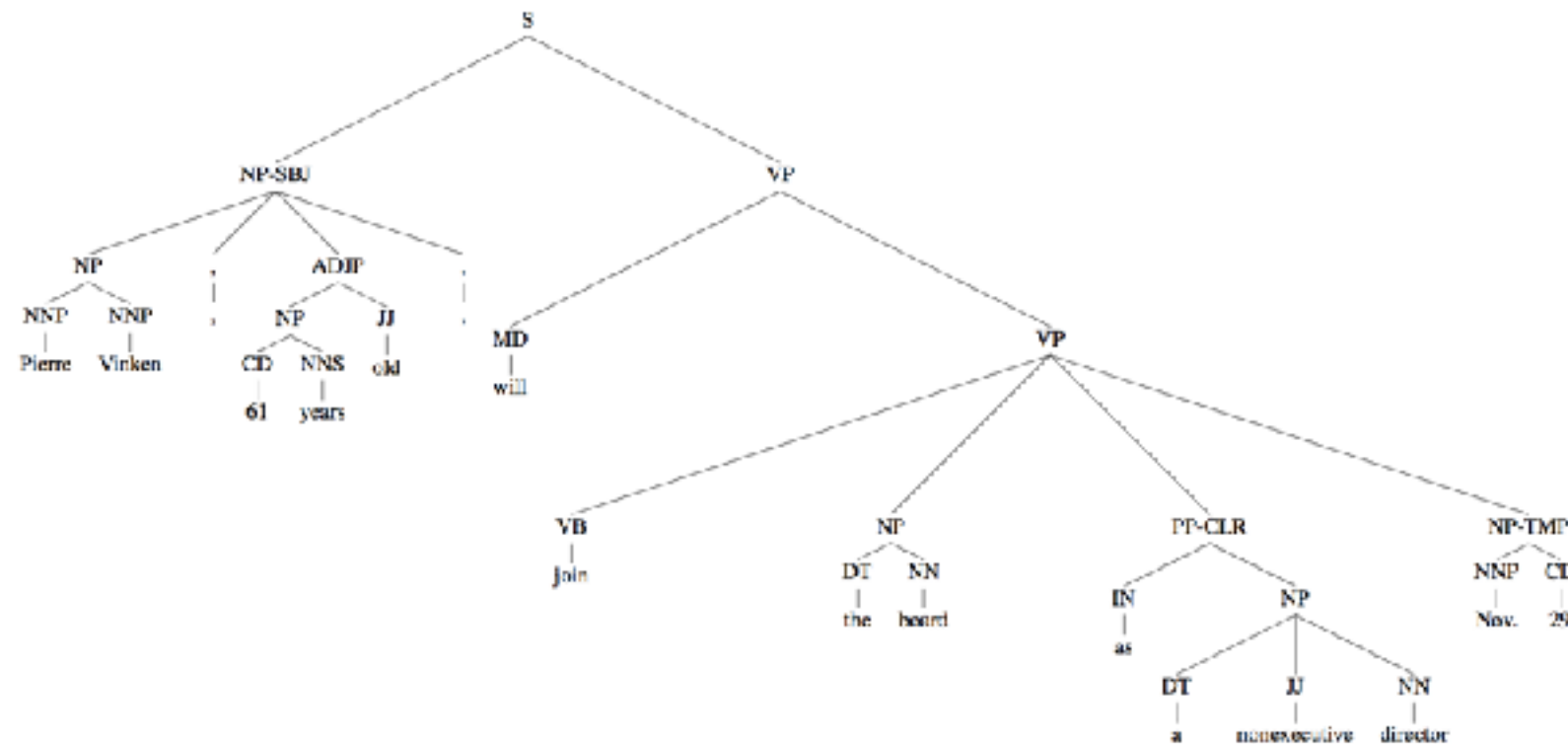
Calculate precision, recall, F1 from these collections of tuples

- Precision: number of tuples in predicted tree also in gold standard tree, divided by number of tuples in **predicted tree**
- Recall: number of tuples in predicted tree also in gold standard tree, divided by number of tuples in **gold standard tree**

Treebanks

- Rather than create the rules by hand, we can annotate sentences with their syntactic structure and then extract the rules from the annotations
- Treebanks: collections of sentences annotated with syntactic structure

Penn Treebank



NP	→	NNP NNP
NP-SBJ	→	NP , ADJP ,
S	→	NP-SBJ VP
VP	→	VB NP PP-CLR NP-TMP

Example rules extracted from this single annotation

PCFG

- Probabilistic context-free grammar: each production is also associated with a probability.
- This lets us calculate the probability of a parse for a given sentence; for a given parse tree T for sentence S comprised of n rules from R (each $A \rightarrow \beta$):

$$P(T, S) = \prod_i^n P(\beta \mid A)$$

Estimating PCFGs

$$\sum_{\beta} P(\beta \mid A) = \frac{C(A \rightarrow \beta)}{\sum_{\gamma} C(A \rightarrow \gamma)}$$

(equivalently)

$$\sum_{\beta} P(\beta \mid A) = \frac{C(A \rightarrow \beta)}{C(A)}$$

A		β	$P(\beta \mid \text{NP})$
NP	→	NP PP	0.092
NP	→	DT NN	0.087
NP	→	NN	0.047
NP	→	NNS	0.042
NP	→	DT JJ NN	0.035
NP	→	NNP	0.034
NP	→	NNP NNP	0.029
NP	→	JJ NNS	0.027
NP	→	QP -NONE-	0.018
NP	→	NP SBAR	0.017
NP	→	NP PP-LOC	0.017
NP	→	JJ NN	0.015
NP	→	DT NNS	0.014
NP	→	CD	0.014
NP	→	NN NNS	0.013
NP	→	DT NN NN	0.013
NP	→	NP CC NP	0.013

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]						
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

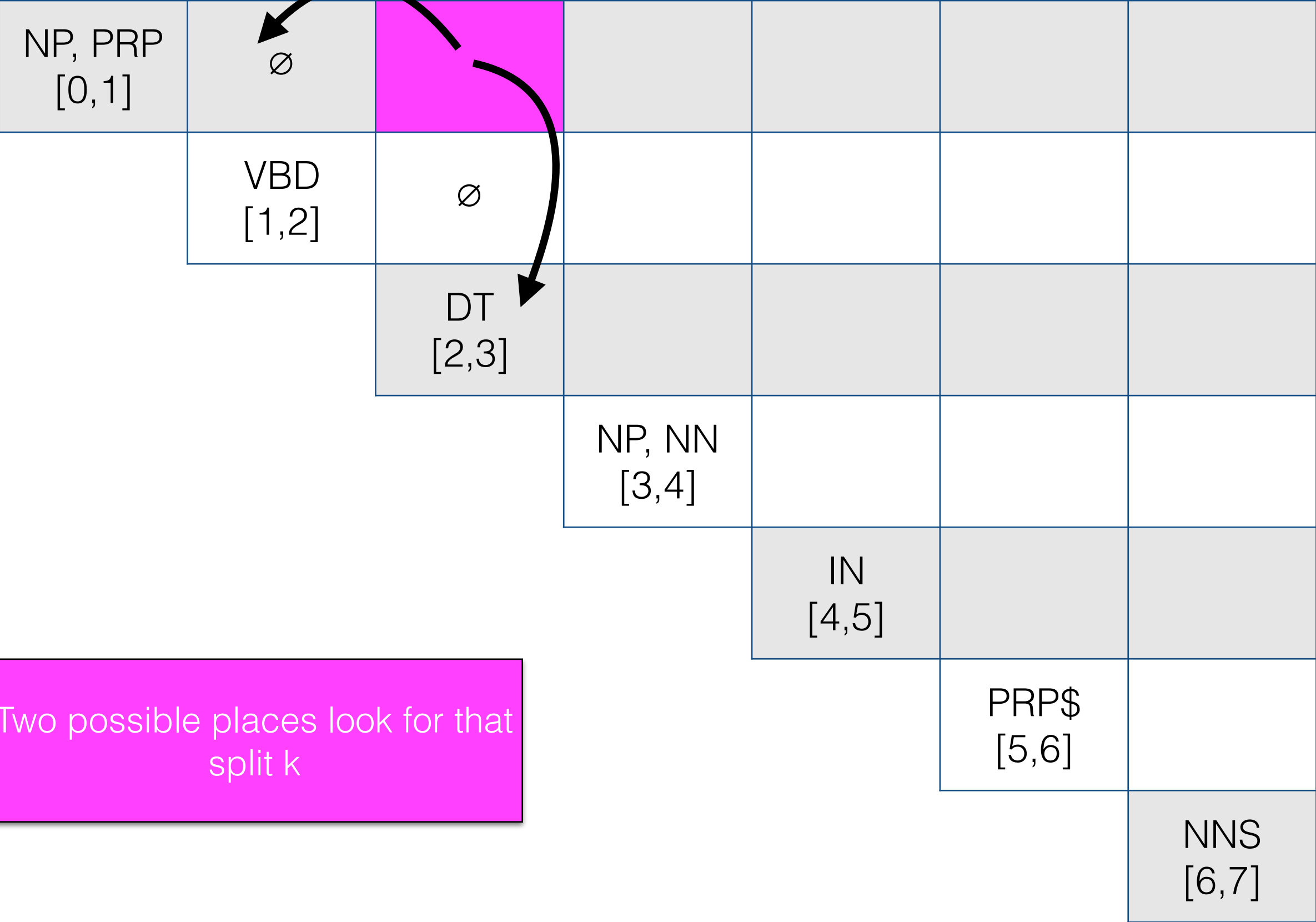
Does any rule generate PRP
VBD?

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

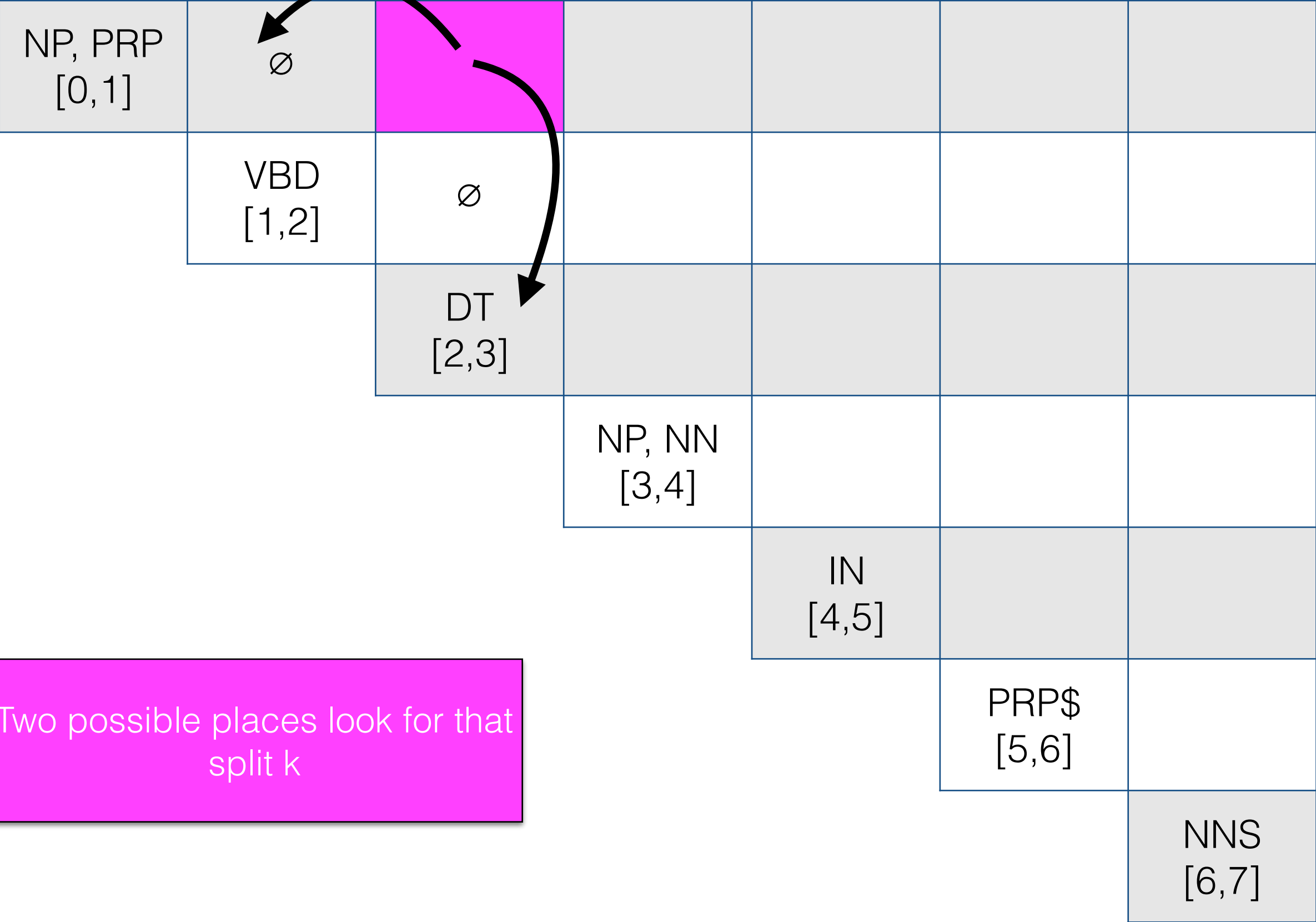
NP, PRP [0,1]	∅					
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

Does any rule generate
VBD DT?

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------



I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------



I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅				
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

Does any rule generate
DT NN?

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

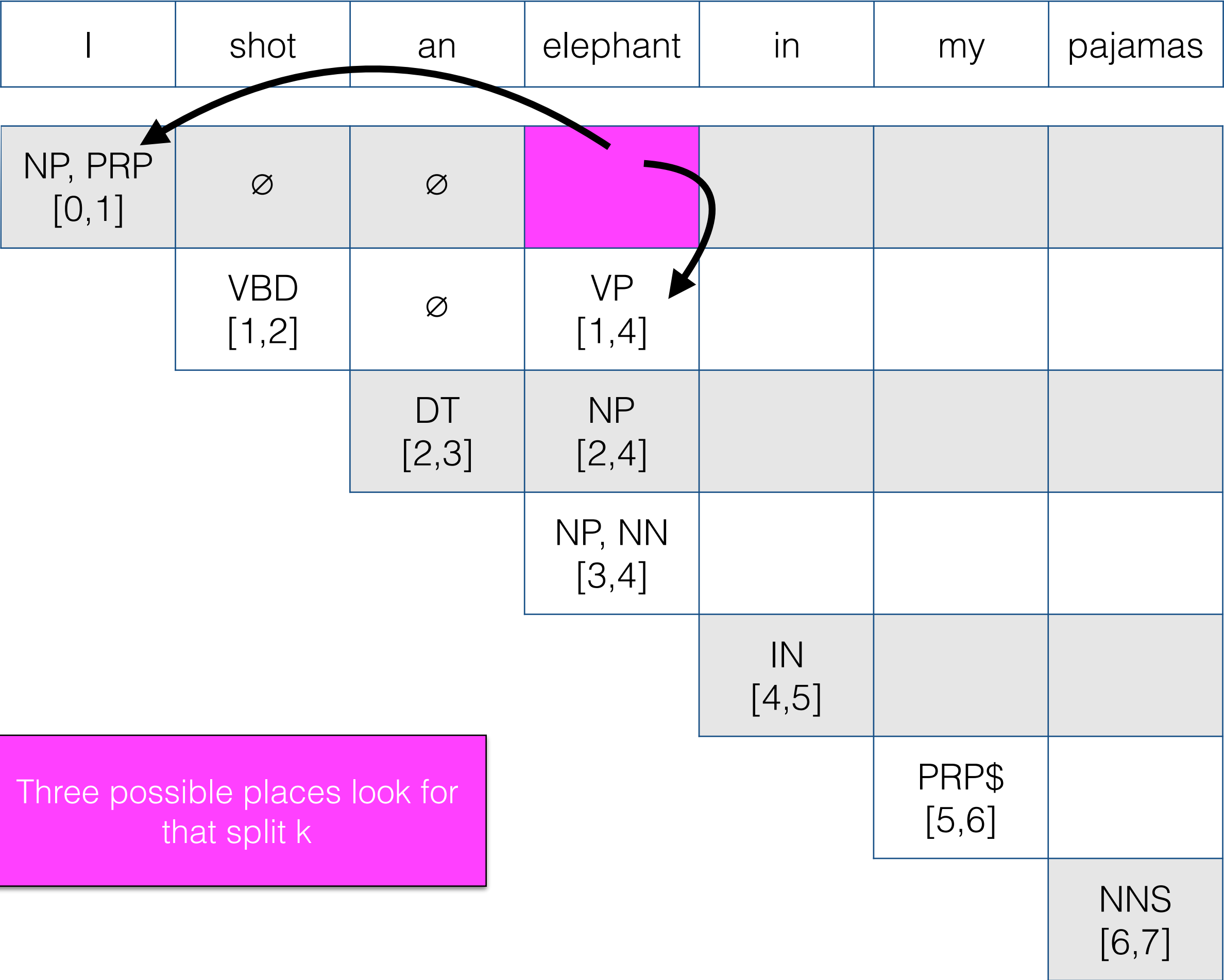
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅				
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

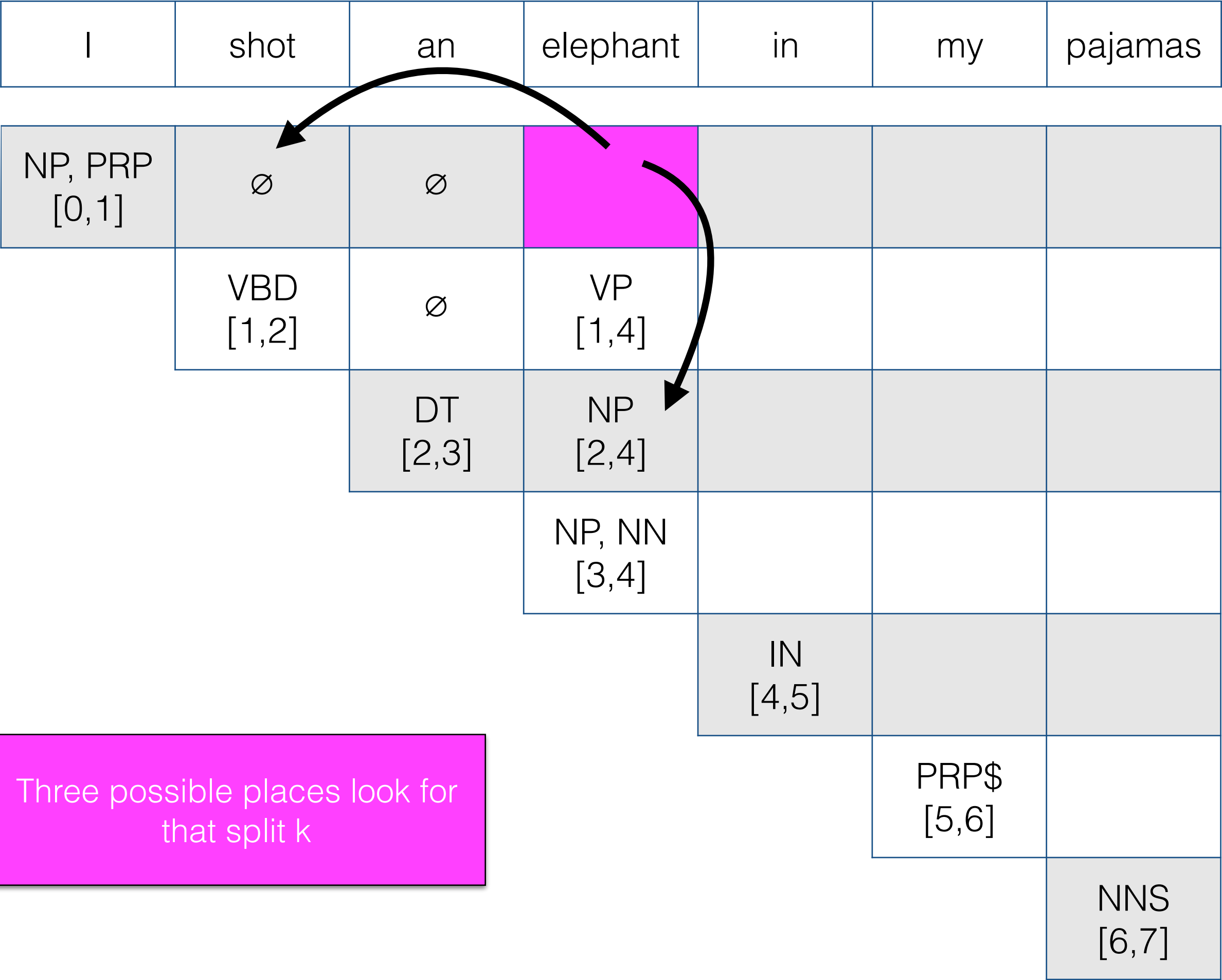
Two possible places look for that split k

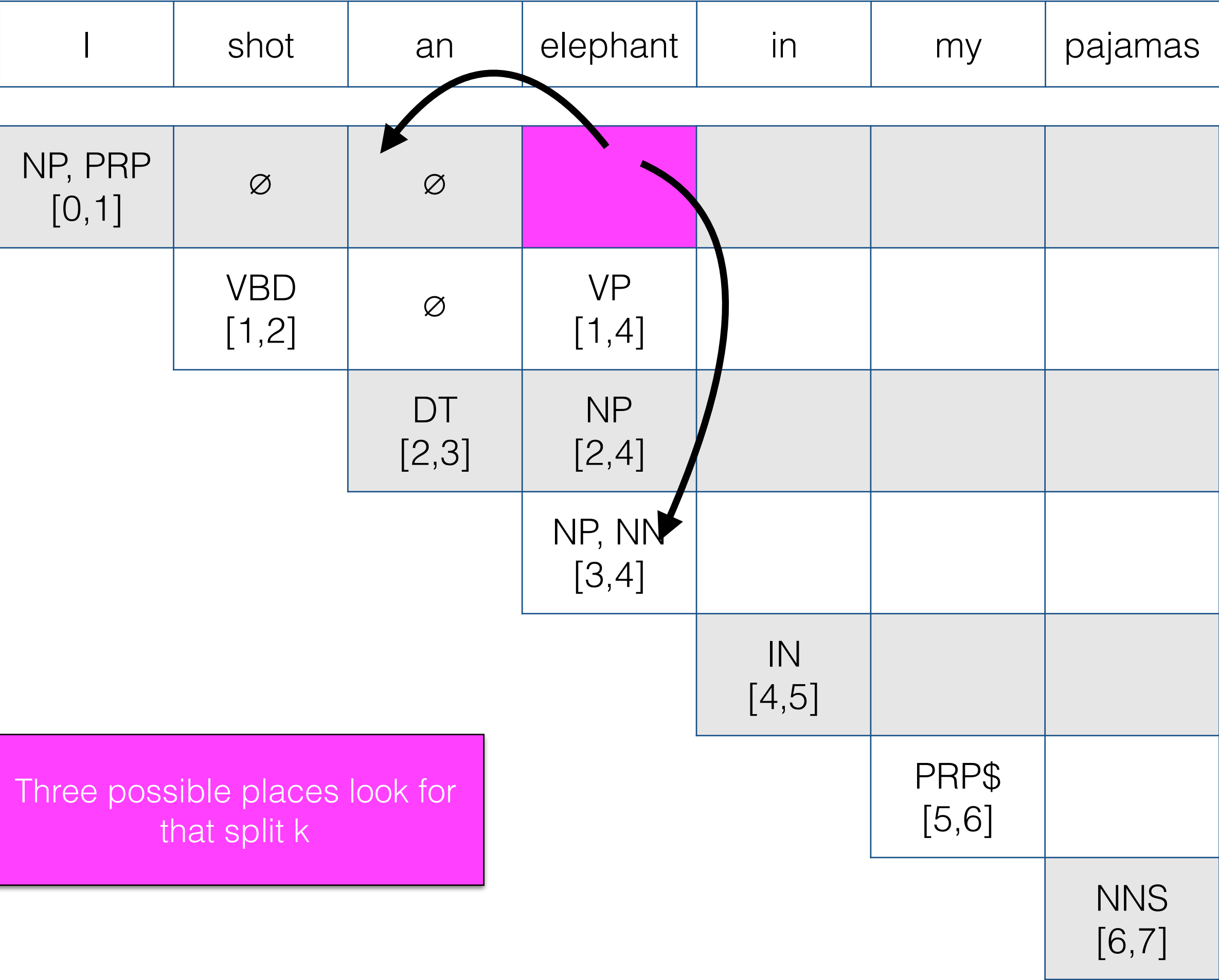
I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

Three possible places look for
that split k







I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]			
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP\$ [5,6]	
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	
			NP, NN [3,4]	∅	∅	
				IN [4,5]	∅	
					PRP\$ [5,6]	
						NNS [6,7]

- *elephant in
*an elephant in
*shot an elephant in
*I shot an elephant in
- *in my
*elephant in my
*an elephant in my
*shot an elephant in my
*I shot an elephant in my

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	
			NP, NN [3,4]	∅	∅	
				IN [4,5]	∅	
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	
			NP, NN [3,4]	∅	∅	
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

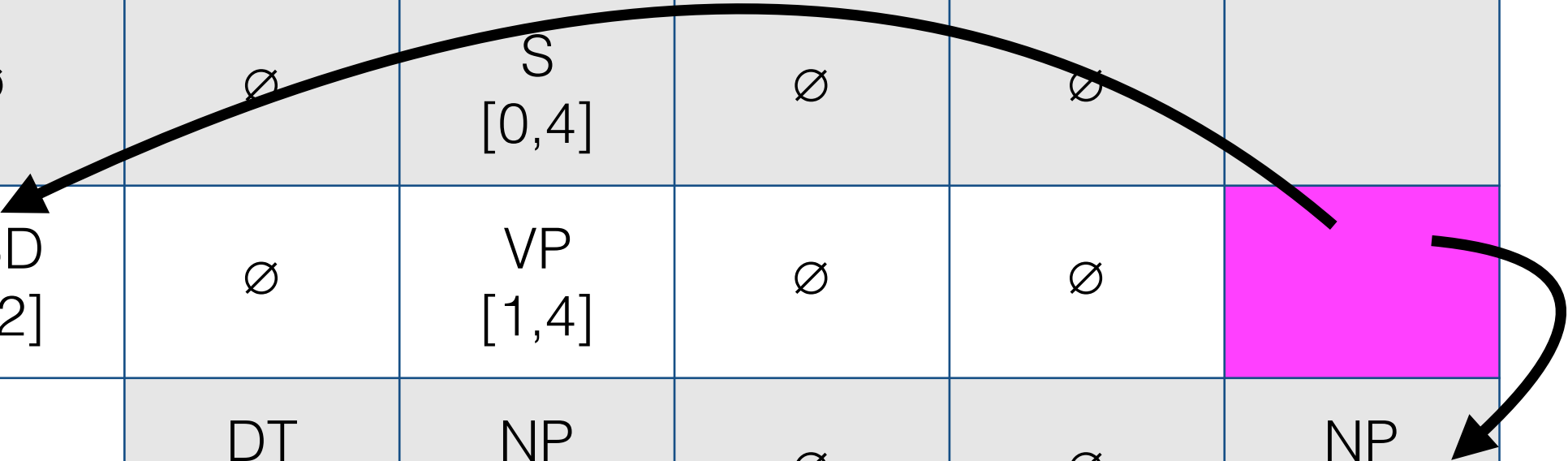
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

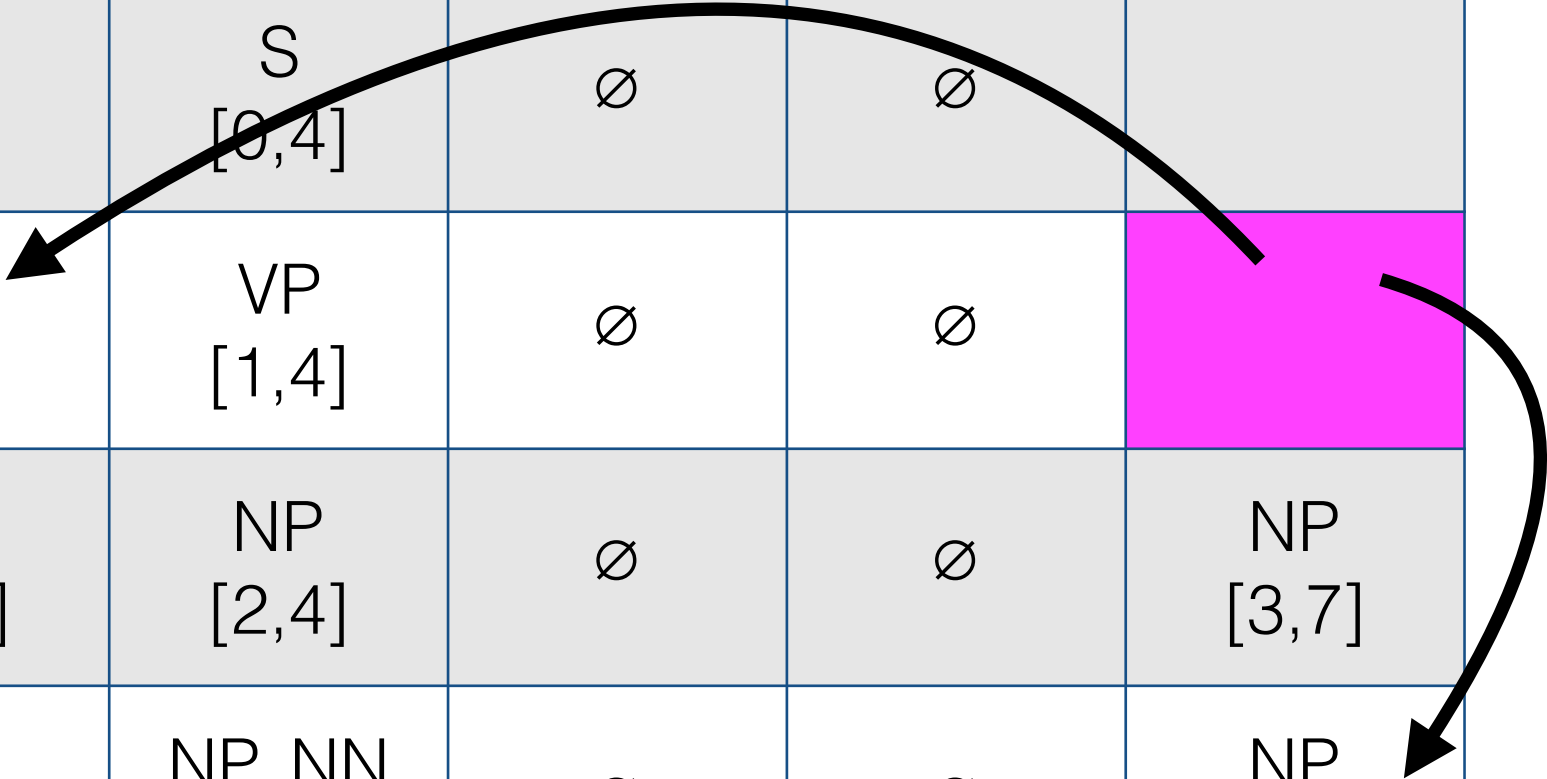
I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]



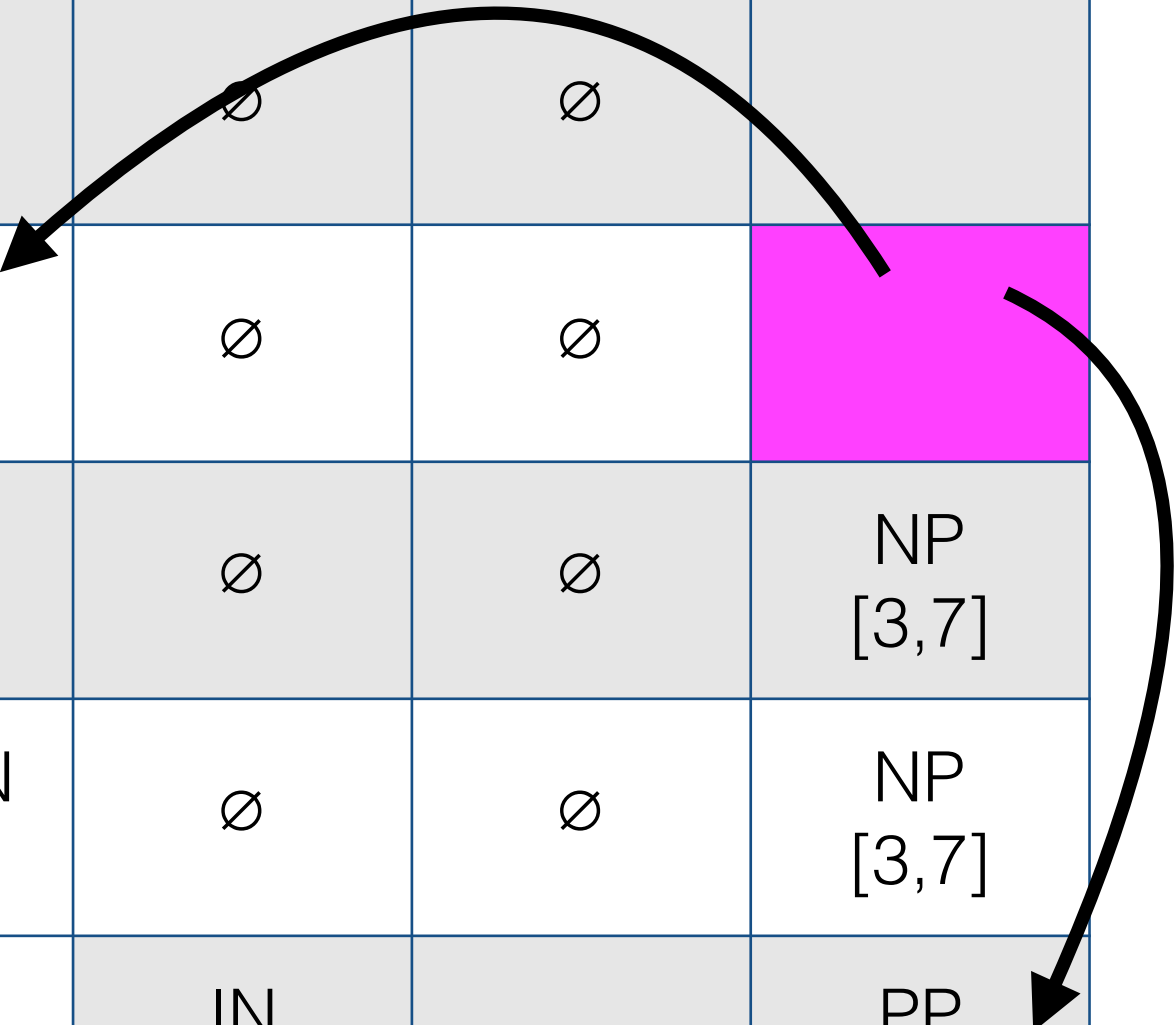
I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]



I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

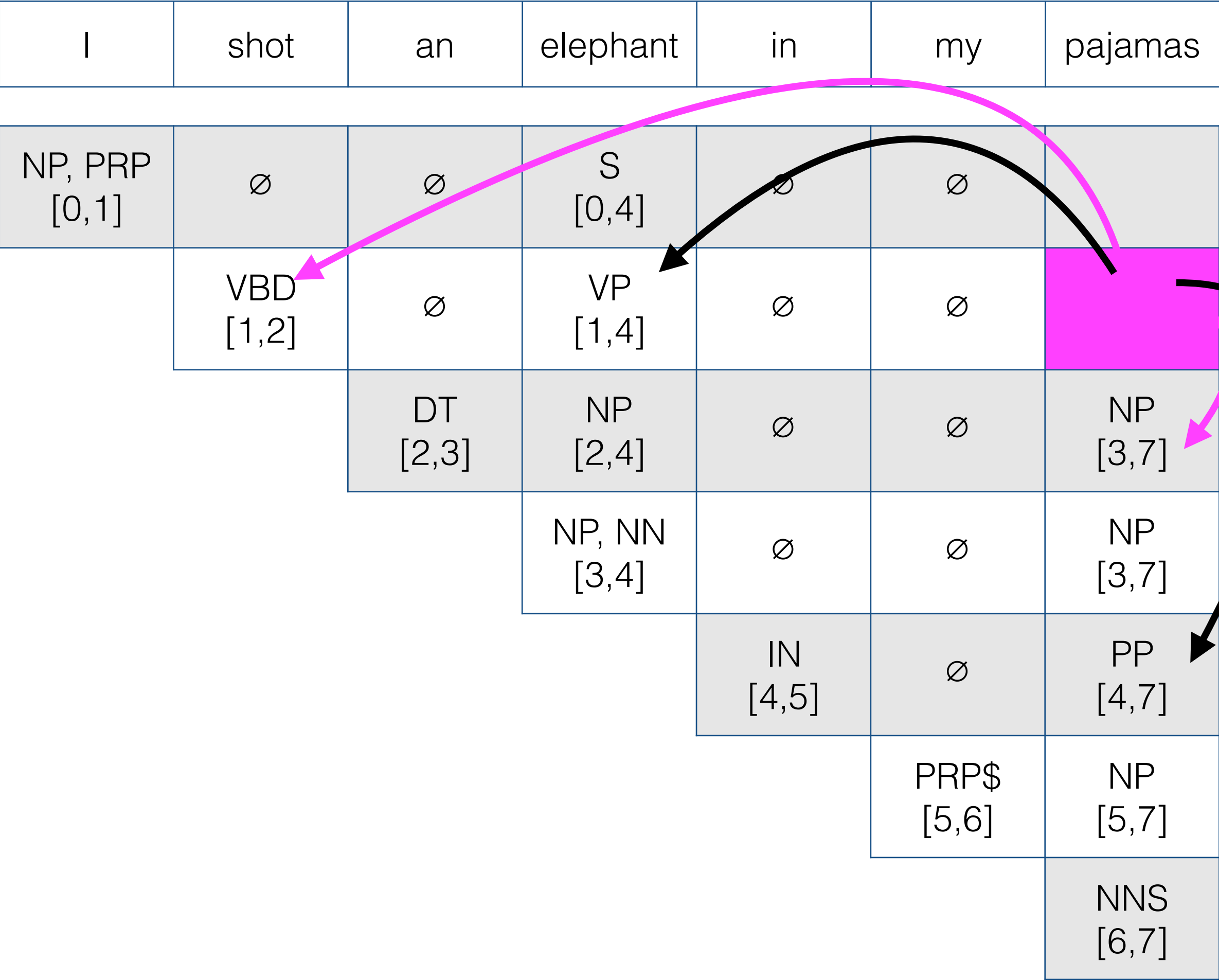


I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

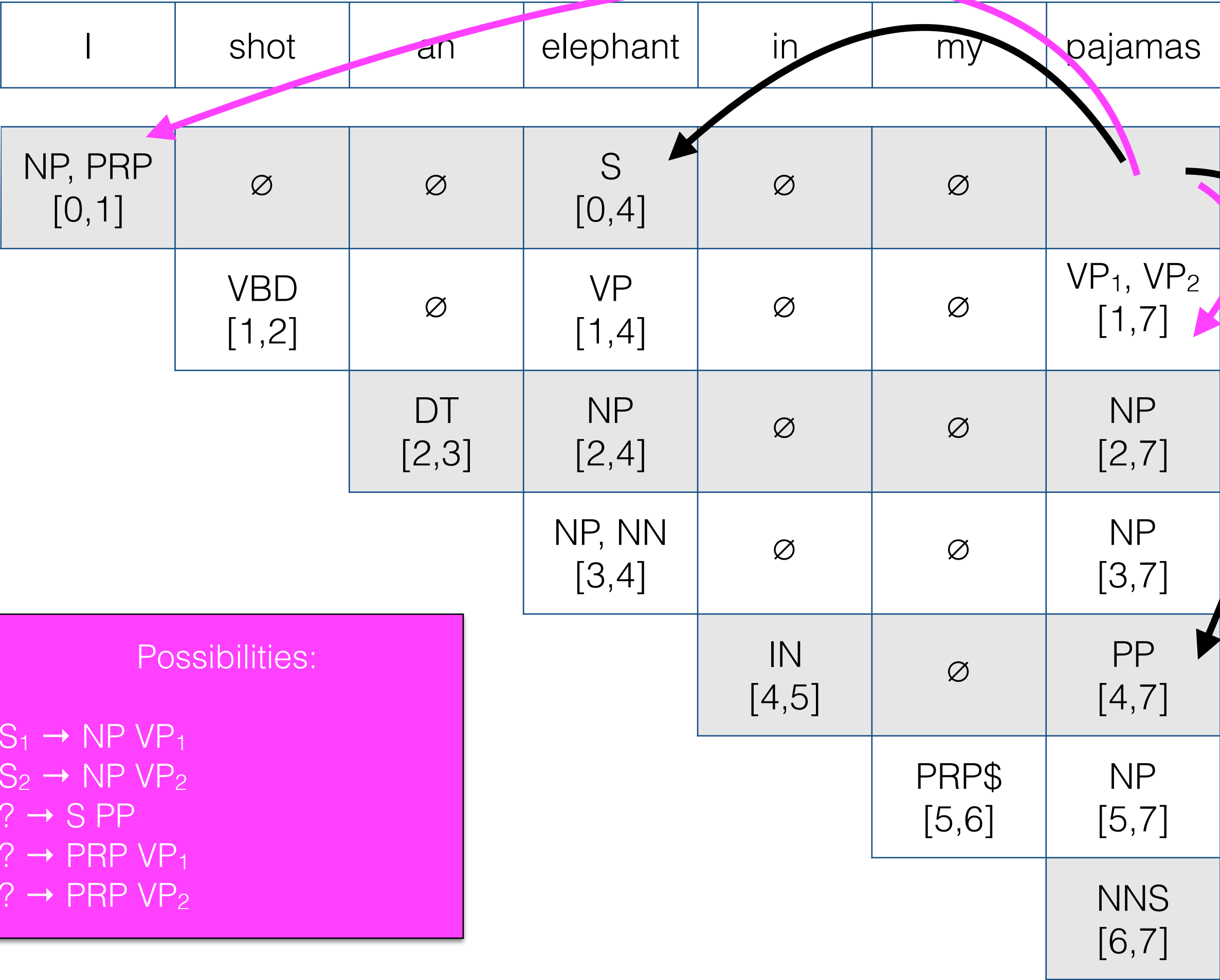
I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]



I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	VP ₁ , VP ₂ [1,7]
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

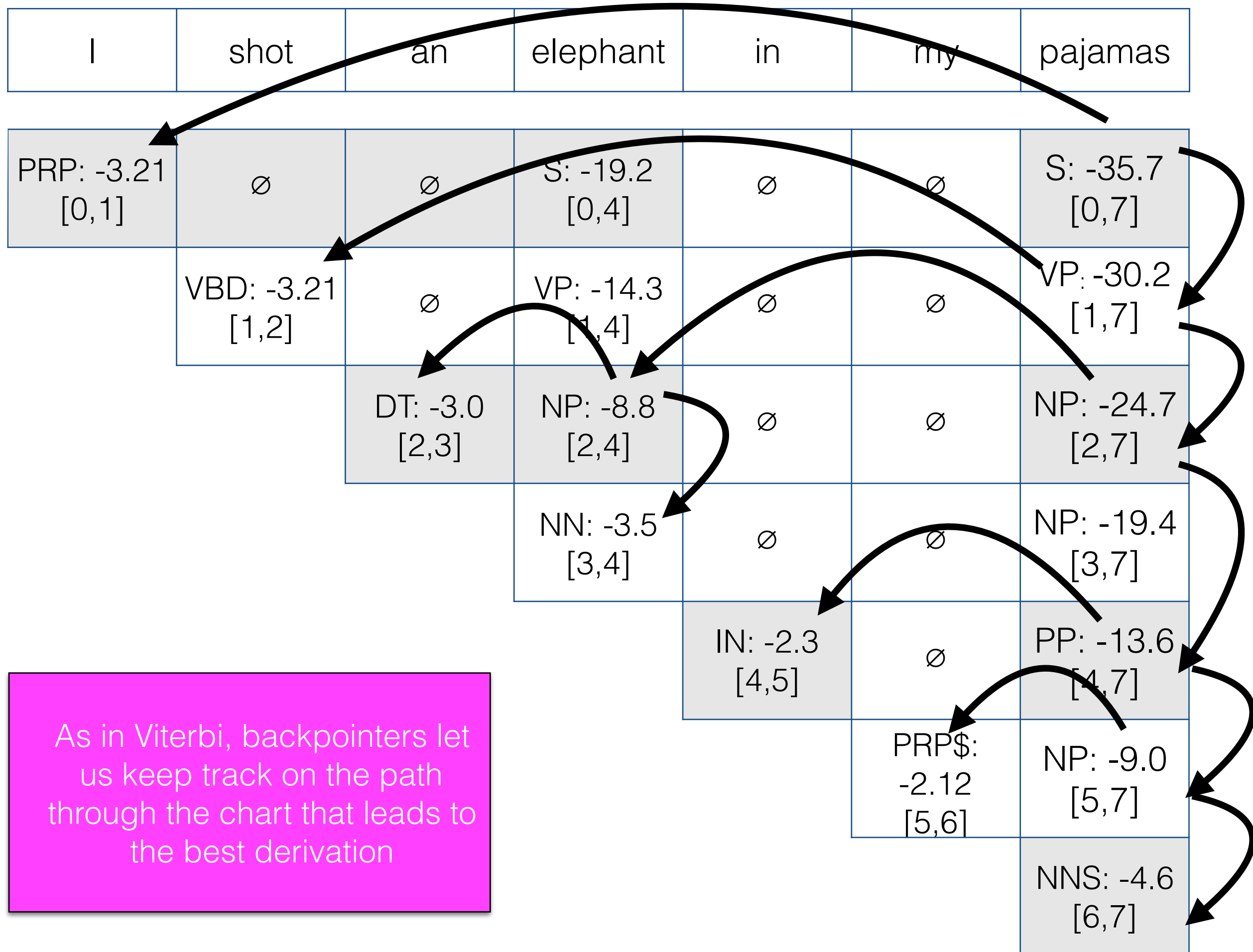


I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	S ₁ , S ₂ [0,7]
	VBD [1,2]	∅	VP [1,4]	∅	∅	VP ₁ , VP ₂ [1,7]
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP\$ [5,6]	NP [5,7]
						NNS [6,7]

Success! We've recognized a total of two valid parses

PCFGs

- A PCFG gives us a mechanism for assigning scores (here, probabilities) to different parses for the same sentence.
- But what we often care about is finding the single best parse with the highest probability.
- We calculate the max probability parse using CKY by storing the probability of each phrase within each cell as we build it up.



Multiple choice

John builds a system to detect lies in speeches by politicians. To evaluate his system, he halts development and runs every speech given by a national political candidate through the system, during one week. The system hypothesizes that ten statements are lies, which John has an expert human fact-checker check. Nine of the ten are found to be lies, and one is found to be true. “My system achieves 90% accuracy!” exclaims John. Your response:

- (A) Yes! Great job, John!
- (B) No, John, your system achieves 90% F -measure.
- (C) No, John, your system achieves 90% recall.
- (D) No, John, your system achieves 90% precision.

Multiple choice

Pointwise mutual information is a measure of association between:

- (A) Two random variables
- (B) A random variable and one of its values
- (C) A word and document label
- (D) Two values of two random variables

Short answer

- What is regularization and why is it important?

Short answer

- For sequence labeling problems like POS tagging and named entity recognition, what are two strengths of using a bidirectional LSTM over an HMM? What's one weakness?

Long answer

Here is a fragment of a PCFG:

Production	log P(Production)	Notes
S → NP VP	-0.70	
S → Noun VP	-2.30	
VP → Verb Noun	-0.70	
VP → Verb ProperNoun	-1.50	
VP → buffalo	-4.00	
NP → ProperNoun Noun	-0.70	
Verb → buffalo	-4.00	“to bully or harrass”
ProperNoun → buffalo	-2.30	A city in New York
Noun → buffalo	-3.00	bison

Table 1: Fragment of PCFG

- (a) Use that grammar and (optionally) the chart below to find the most probable parse of *buffalo buffalo buffalo*. Draw the parse tree and its probability.
- (b) If the sentence is ambiguous, list the interpretations, explaining their differences in plain English.
-

@kimkierkegaardashian is a Twitter account that mashes up the language of celebrity Kim Kardashian and philosopher Søren Kierkegaard.



Figure 2: @kimkierkegaardashian

- (a) Assume independent language models have been trained on the tweets of Kim Kardashian (generating language model $\mathcal{L}_{\text{Kim}}$) and the writings of Søren Kierkegaard (generating language model $\mathcal{L}_{\text{Søren}}$). How could you use $\mathcal{L}_{\text{Kim}}$ and $\mathcal{L}_{\text{Søren}}$ to create a new language model $\mathcal{L}_{\text{Kim+Søren}}$ to generate tweets like those above?
- (b) How would you control that model to sound more like Kierkegaard than Kardashian?
- (c) Assume you have access to the full Twitter archive of @kimkierkegaardashian. How could you choose the best way to combine $\mathcal{L}_{\text{Kim}}$ and $\mathcal{L}_{\text{Søren}}$? How would you operationalize “best”?