Natural Language Processing

Info 159/259

Lecture 15: Review (March 10, 2020)

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 and word representations
    - Distributional hypothesis
    - Distributed representations
    - Subword embedding models
    - Contextualized word representations (ELMO)
    - BERT
- Evaluation metrics (accuracy, precision, recall, F score, perplexity, parseval)
- Sequence labeling
    - POS, NER
    - Methods: HMM, MEMM, CRF, RNN, BiRNN, BERT
- 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?

Midterm

- Take-home exam — you'll have 80 continuous minutes to take the midterm anytime within the 24-hour period (3:30pm 3/12—2:59pm 3/13).
- Mix of multiple choice, short answer, long answer
- Covers all material from lectures and readings

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
$\mathcal{Y}$ = {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	{postive, negative, neutral, mixed}

Bayes' Rule

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

Likelihood of the data
given that Y=y

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

Posterior belief that Y=y given that X=x

Bayes' Rule

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

Likelihood of "really really the worst movie ever"
given that Y= 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=positive given that X="really really the worst movie ever"

This sum ranges over
y=positive + y=negative
(so that it sums to 1)

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
BIAS	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
BIAS	-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.

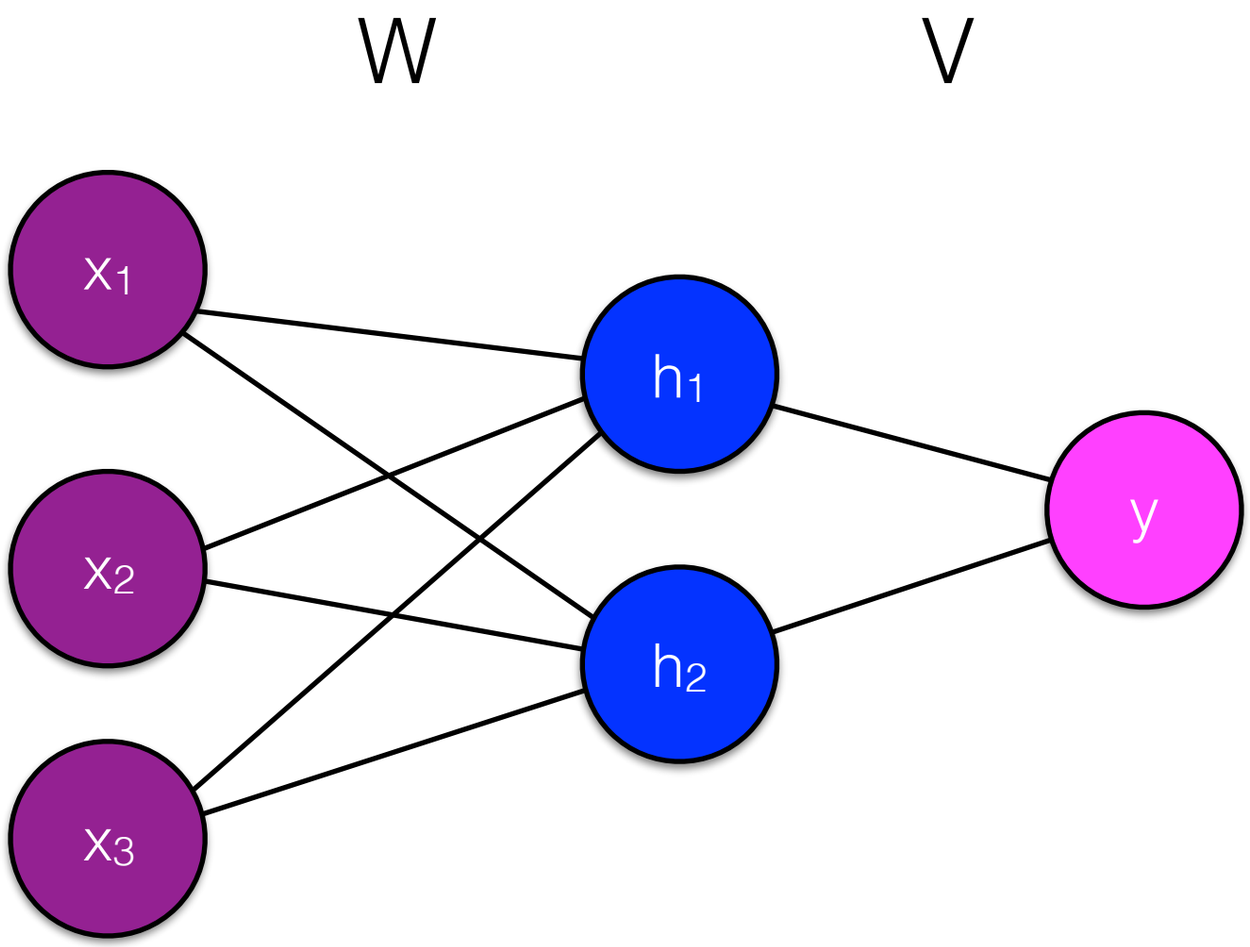
Algorithm 2 Logistic regression stochastic gradient descent

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

L2 regularization

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

- 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)



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

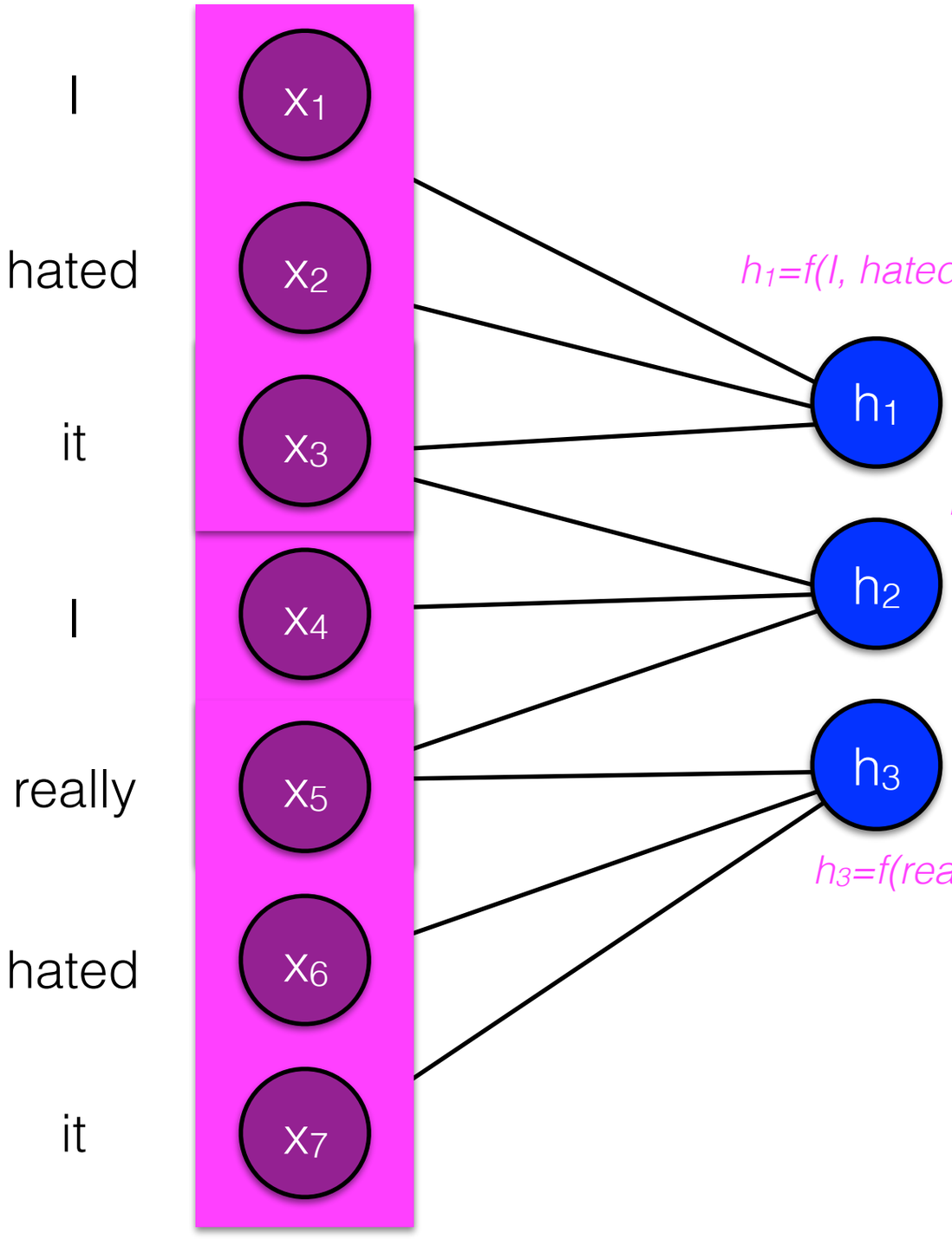
we can express y as a function only of the input x and the weights W and V

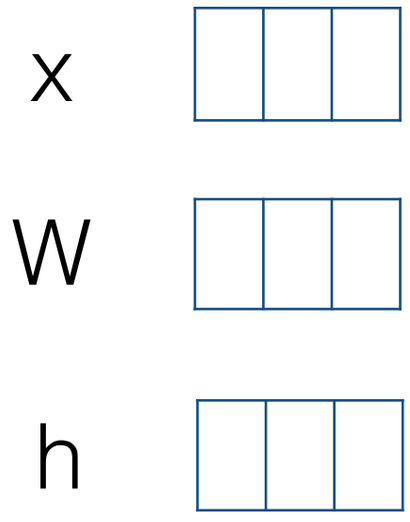
$$\hat{y} = \sigma\left[V_1 \underbrace{\left(\sigma\left(\sum_i^F x_i W_{i,1}\right)\right)}_{h_1} + V_2 \underbrace{\left(\sigma\left(\sum_i^F x_i W_{i,2}\right)\right)}_{h_2} \right]$$

This is hairy, but differentiable

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

Convolutional networks

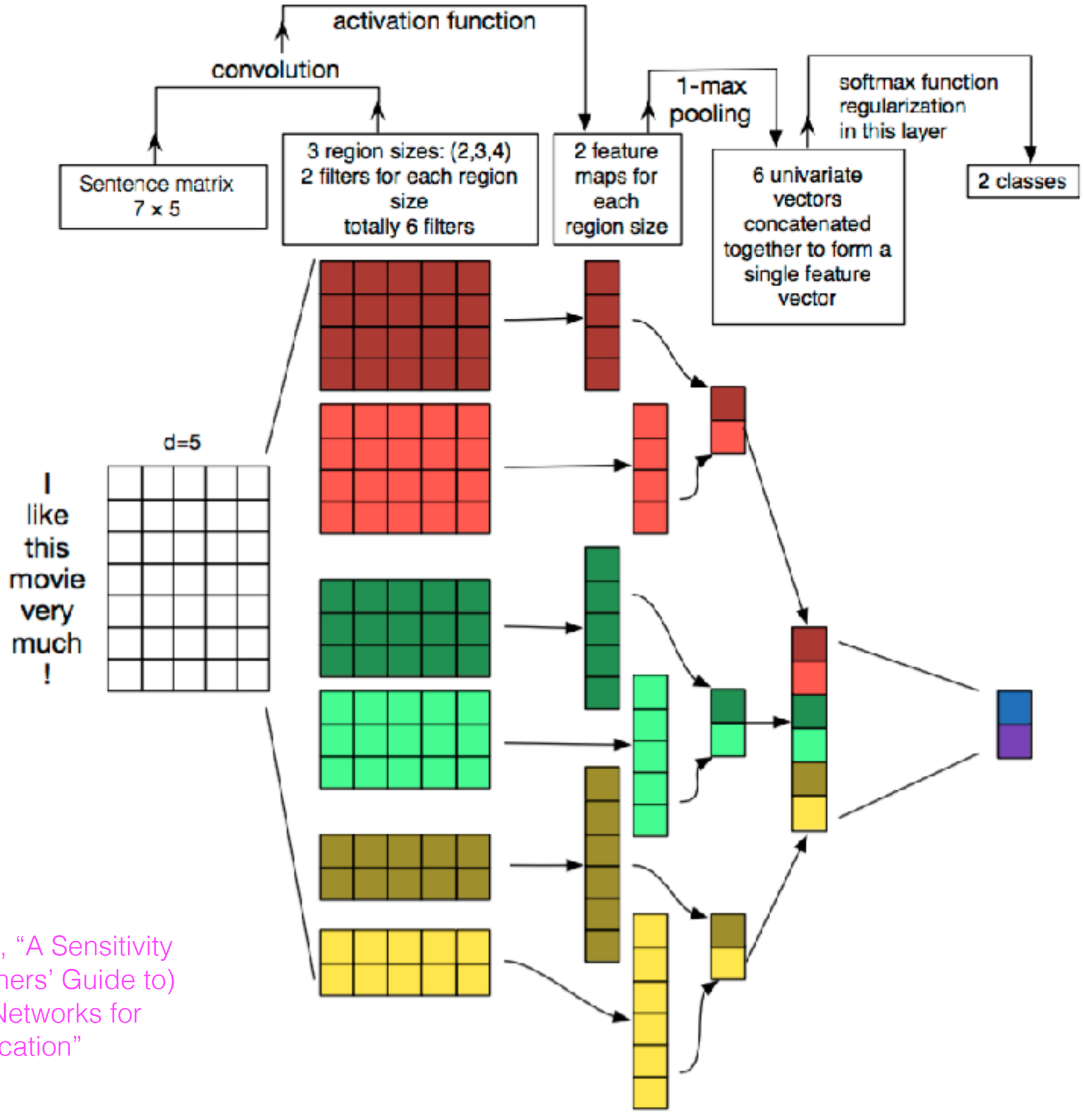




$$h_1 = \sigma(x_1 W_1 + x_2 W_2 + x_3 W_3)$$

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

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

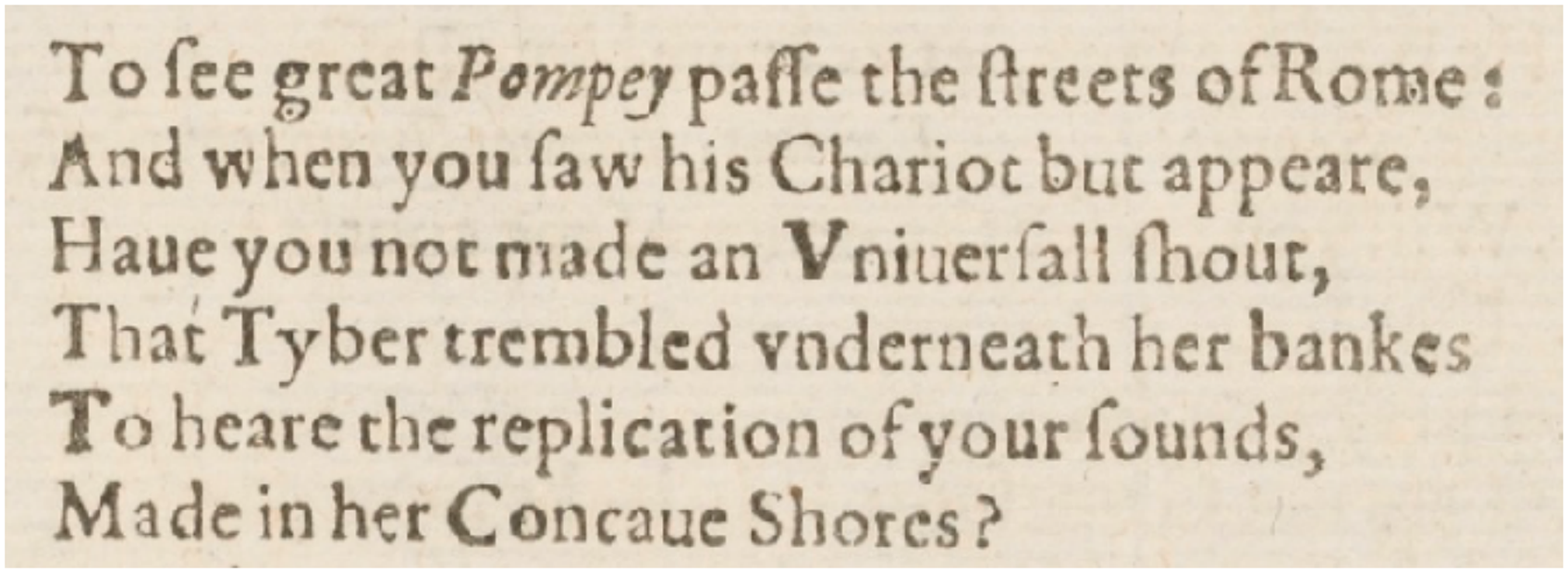


Zhang and Wallace 2016, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification”

Language Model

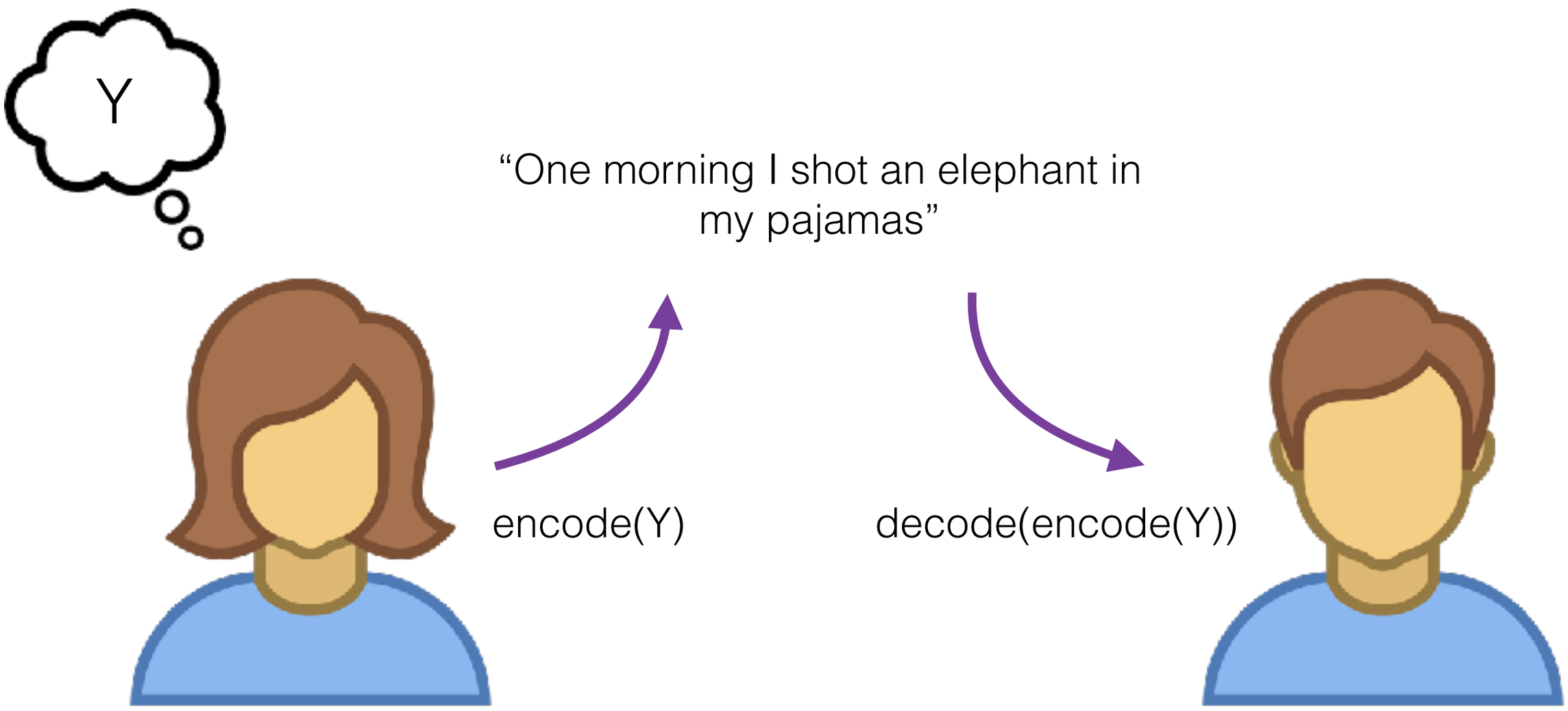
- Language models provide us with a way to quantify the likelihood of sequence — i.e., plausible sentences.

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

Information theoretic view



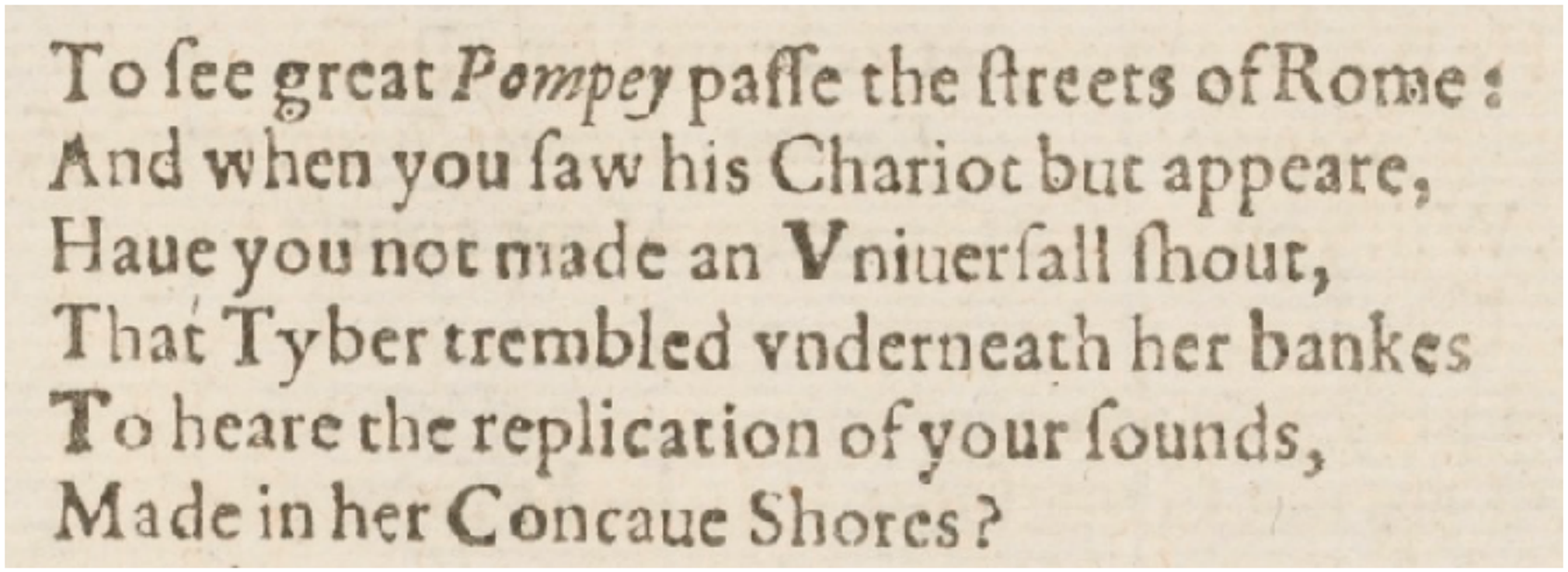
Shannon 1948

Noisy Channel

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

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

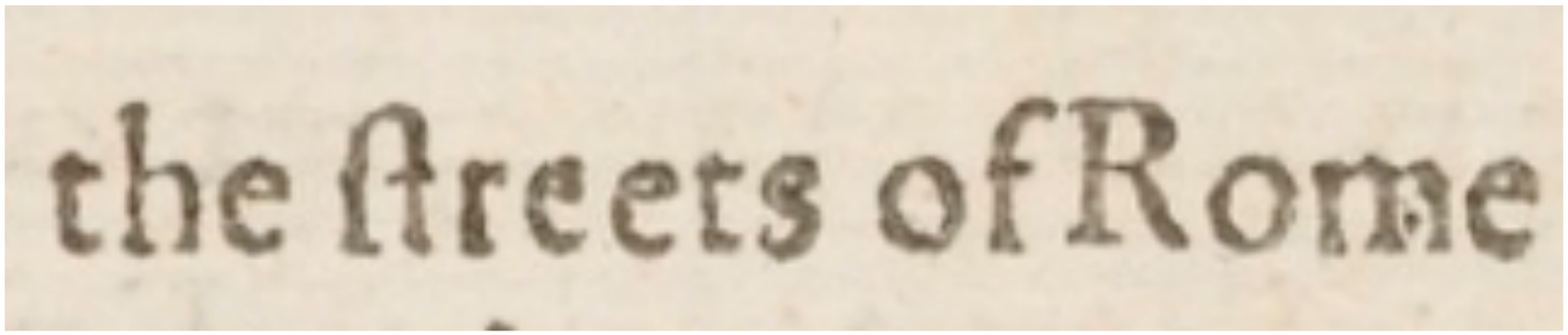
OCR



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?

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

OCR



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

Language Model

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

P(“It was the best of times, it was the worst of times”)

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

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

second-order features

first-order features

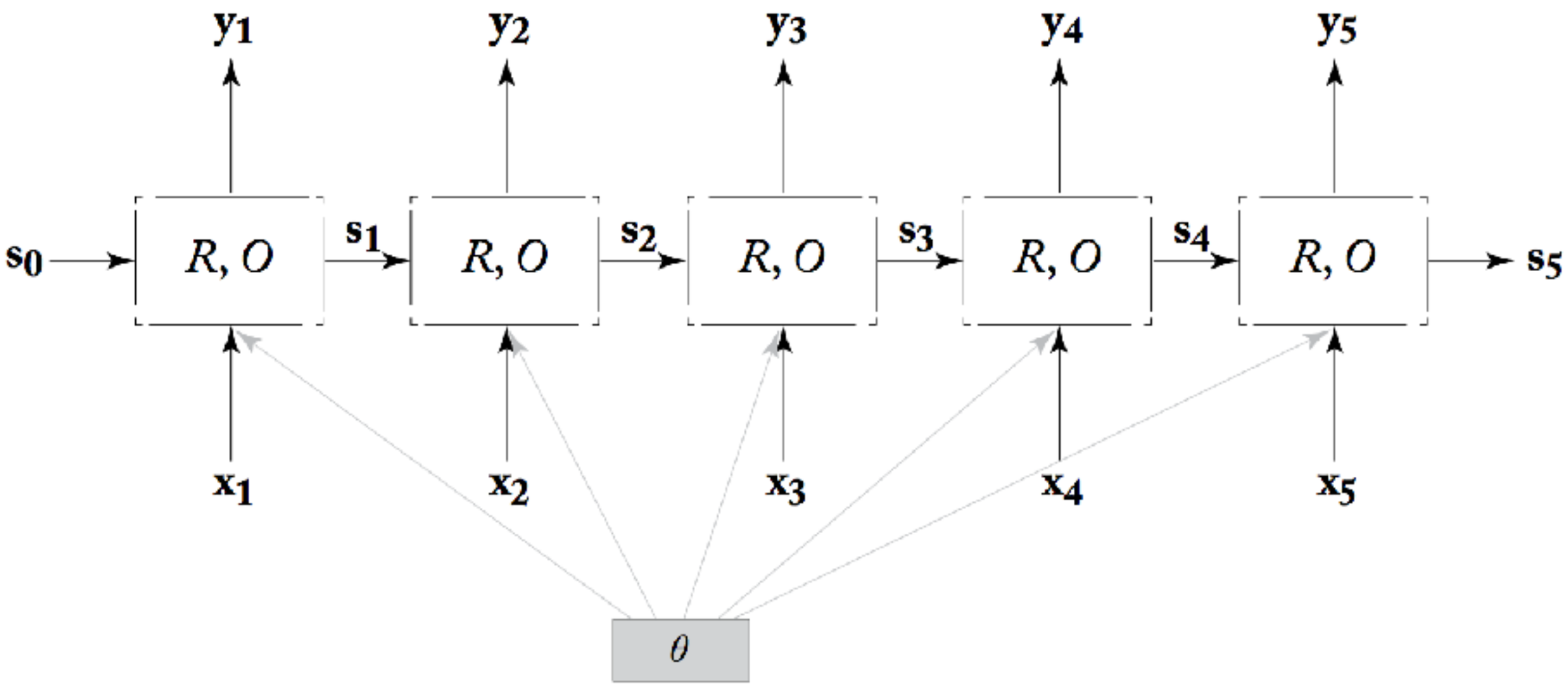
Feature	Value
w_{i-2}=the ^ w_{i-1}=the	0
w_{i-2}=and ^ w_{i-1}=the	1
w_{i-2}=bravest ^ w_{i-1}=the	0
w_{i-2}=love ^ w_{i-1}=the	0
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=bravest	0
w_{i-1}=love	0
BIAS	1

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.

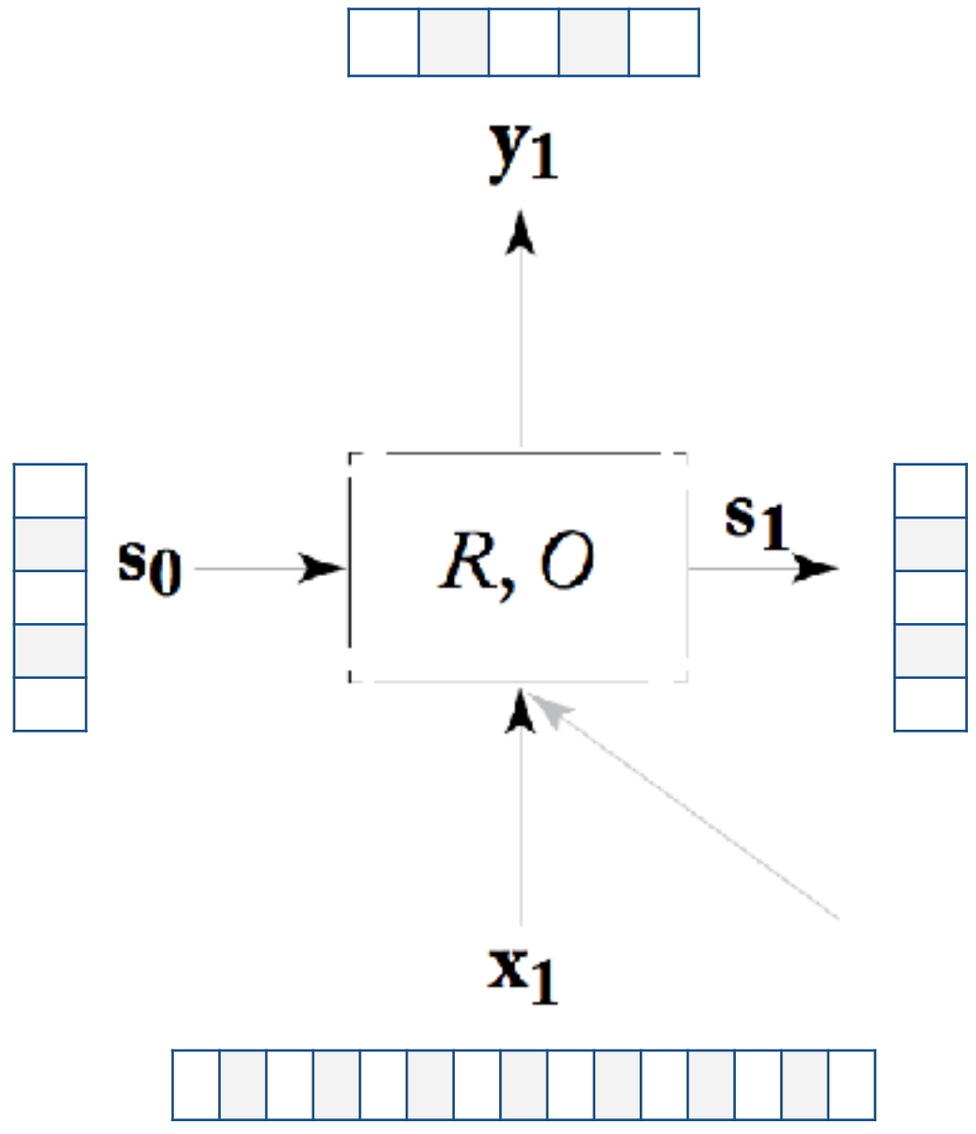
Recurrent neural network



Goldberg 2017

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

- 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).

Dense vectors from prediction

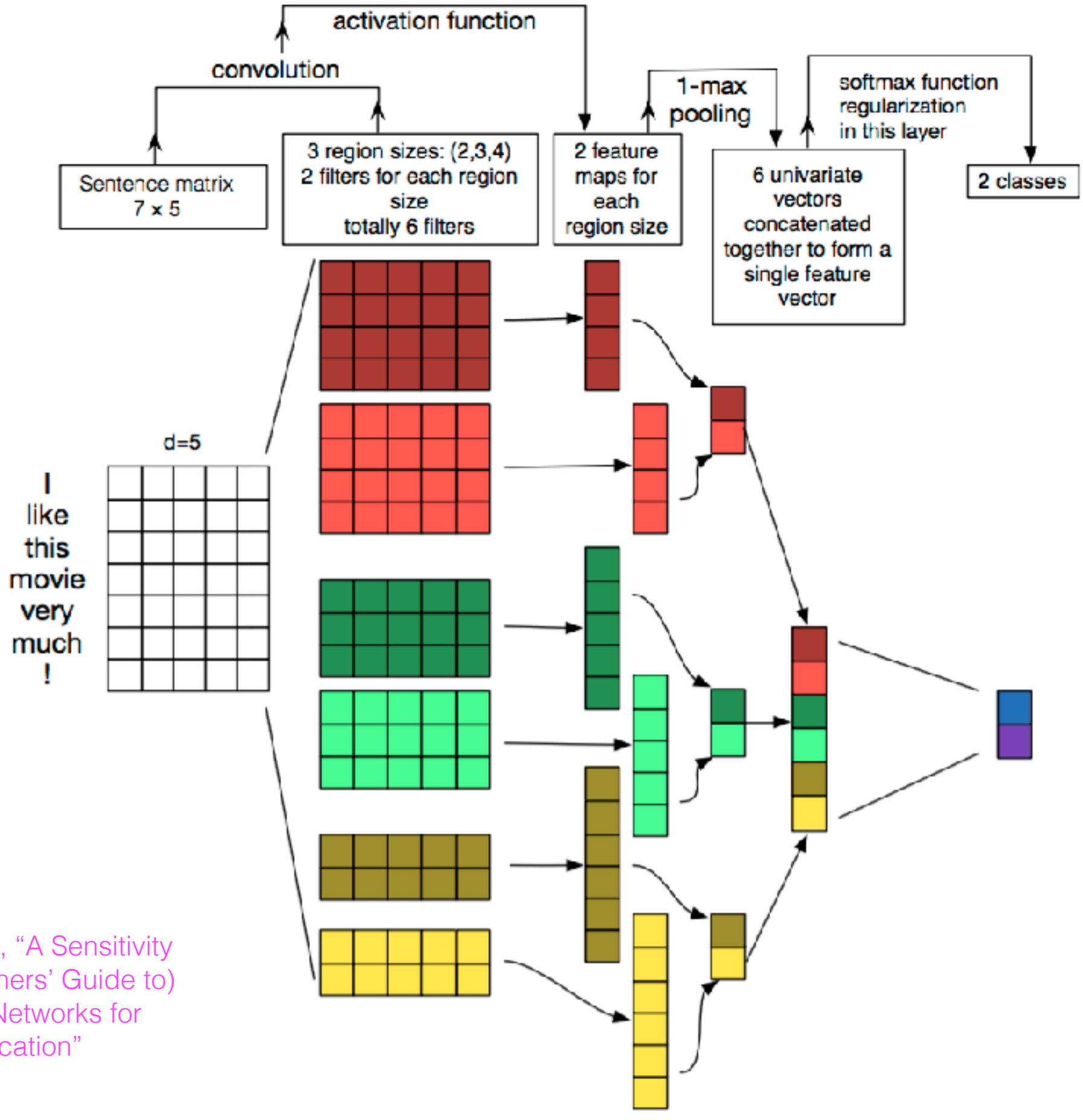
- Learning low-dimensional representations of words by framing a predicting task: using context to predict words in a surrounding window

- Transform this into a supervised prediction problem; similar to language modeling but we're ignoring order within the context window

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”

Subword models

- Rather than learning a single representation for each word type w, learn representations z for the set of ngrams $\mathcal{G}_w$ that comprise it [Bojanowski et al. 2017]

- The word itself is included among the ngrams (no matter its length).

- A word representation is the sum of those ngrams

$$w = \sum_{g \in \mathcal{G}_w} z_g$$

FastText

e(where) =

e(<wh)
+ e(whe)
+ e(her)
+ e(ere)
+ e(re>)

3-grams

+ e(<whe)
+ e(wher)
+ e(here)
+ e(ere>)

4-grams

+ e(<wher)
+ e(where)
+ e(here>)

5-grams

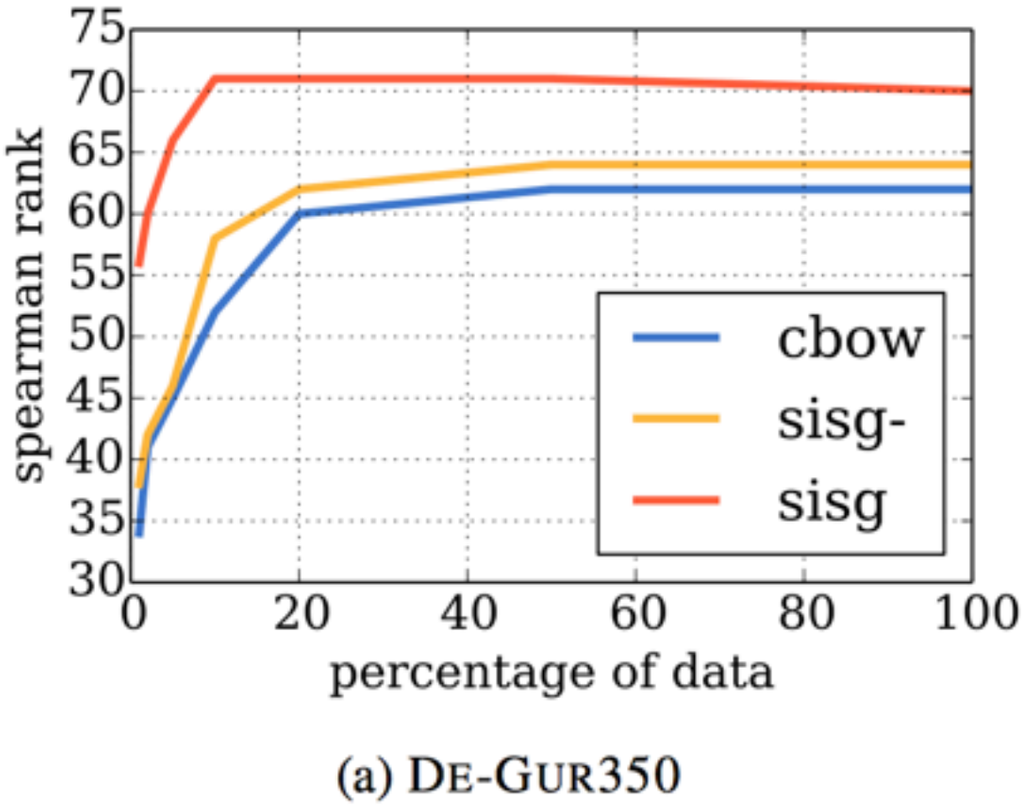
+ e(<where)
+ e(where>)

6-grams

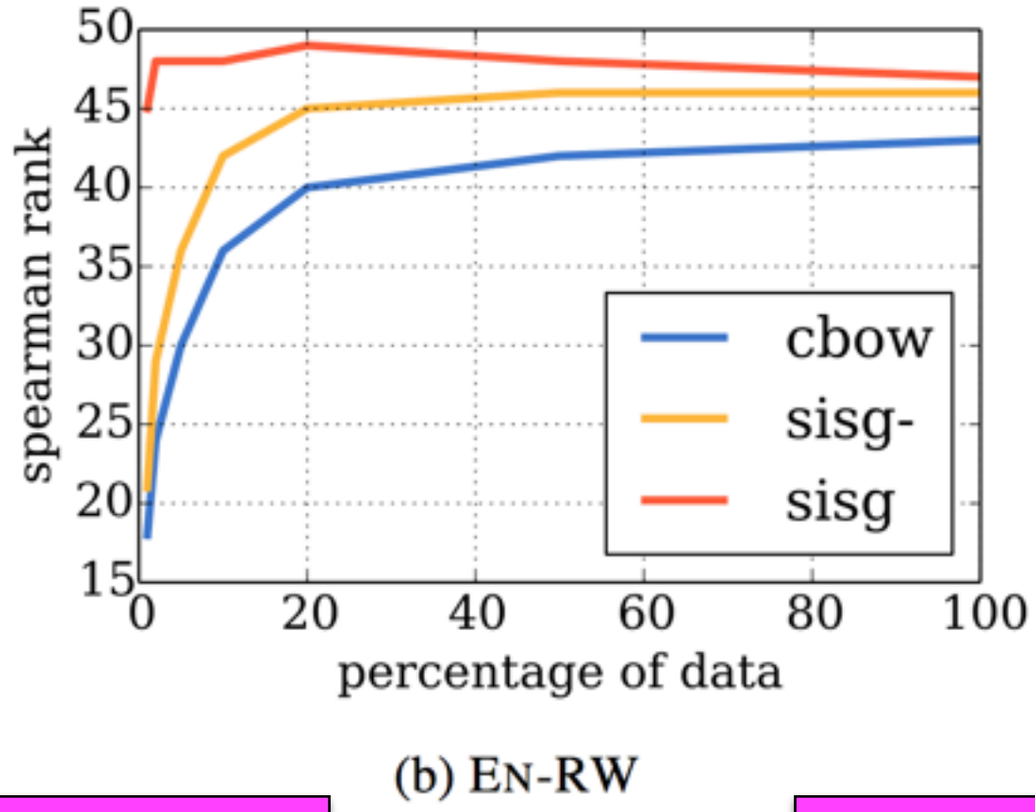
+ e(<where>)

word

e(*) = embedding for *



(a) DE-GUR350



(b) EN-RW

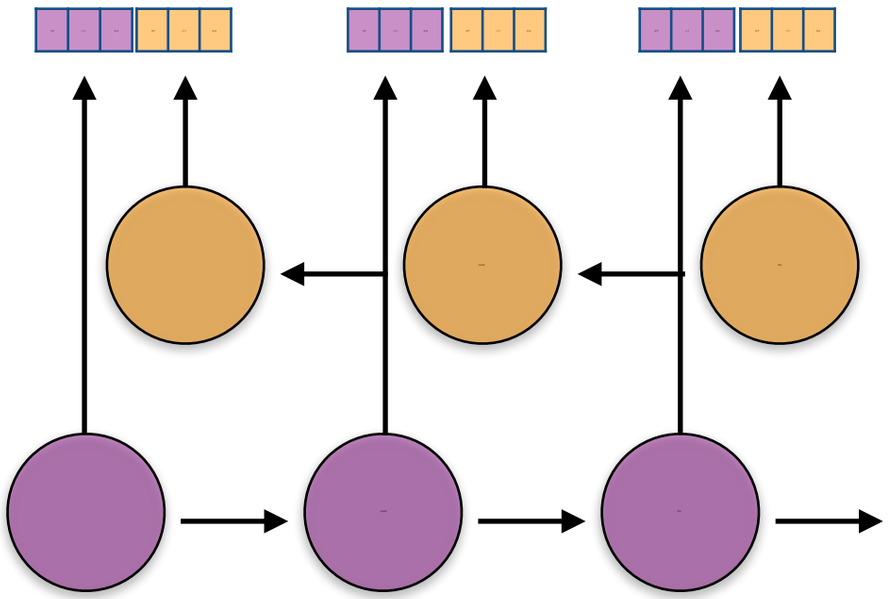
1% =
~20M tokens

100% =
~1B tokens

- Subword models need less data to get comparable performance.

ELMo

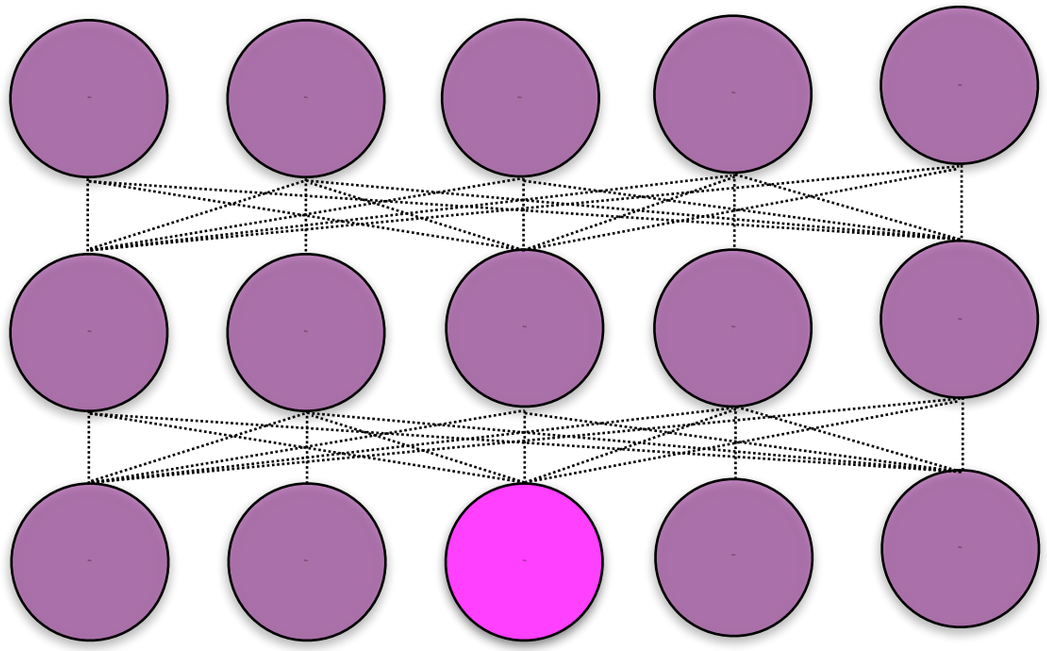
Stacked BiRNN trained to predict next word in language modeling task



Peters et al. 2018

BERT

Transformer-based model to predict masked word using bidirectional context + next sentence prediction.



Devlin et al. 2019

Attention

- Let's incorporate structure (and parameters) into a network that captures which elements in the input we should be attending to (and which we can ignore).

$$v \in \mathcal{R}^H$$

2.7	3.1	-1.4	-2.3	0.7

Define v to be a vector to be learned; think of it as an "important word" vector. The dot product here measures how similar each input vector is to that "important word" vector

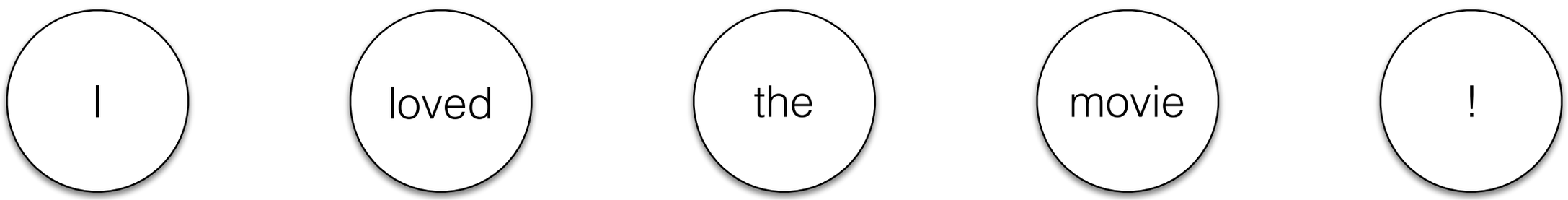
2.7	3.1	-1.4	-2.3	0.7

-0.7	-0.8	-1.3	-0.2	-0.9

2.3	1.5	1.1	1.4	1.3

-0.9	-1.5	-0.7	0.9	0.2

-0.1	-0.7	-1.6	0.2	0.6



$v \in \mathscr{R}^H$

2.7	3.1	-1.4	-2.3	0.7

-3.4 2.4 -0.8 -1.2 1.7

$r_1 = v^\top x_1$ $r_2 = v^\top x_2$ $r_3 = v^\top x_3$ $r_4 = v^\top x_4$ $r_5 = v^\top x_5$

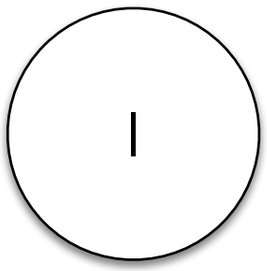
2.7	3.1	-1.4	-2.3	0.7

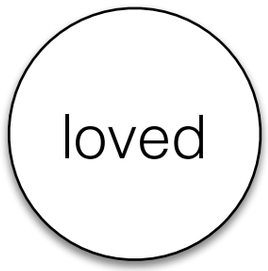
-0.7	-0.8	-1.3	-0.2	-0.9

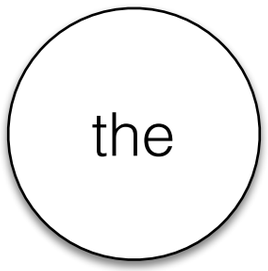
2.3	1.5	1.1	1.4	1.3

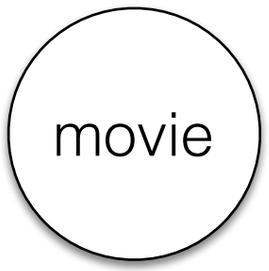
-0.9	-1.5	-0.7	0.9	0.2

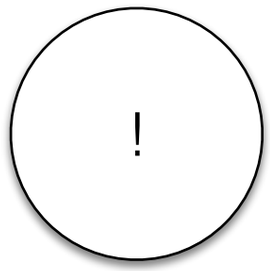
-0.1	-0.7	-1.6	0.2	0.6









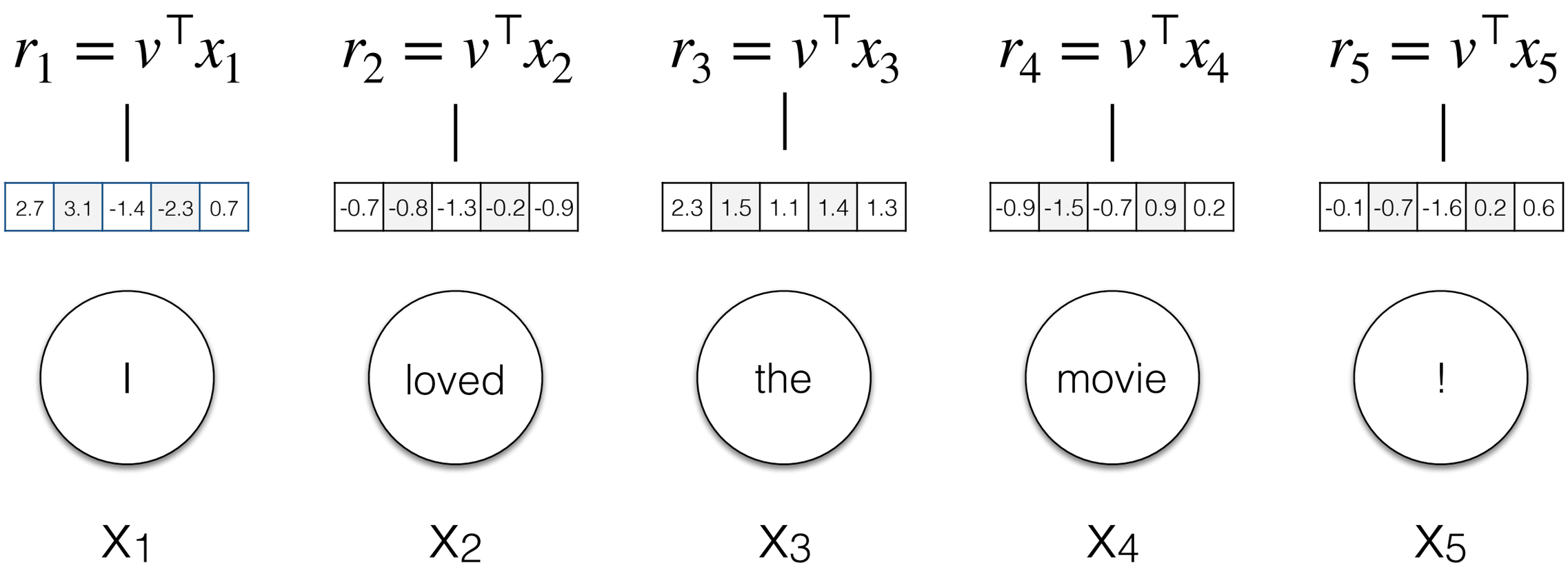


x_1 x_2 x_3 x_4 x_5

Convert r into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$

a	0	0.64	0.02	0.02	0.32
r	-3.4	2.4	-0.8	-1.2	1.7



y

1.9 | -0.2 | -1.1 | -0.2 | -0.7

$$x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5$$

weighted sum

2.7 | 3.1 | -1.4 | -2.3 | 0.7

-0.7 | -0.8 | -1.3 | -0.2 | -0.9

2.3 | 1.5 | 1.1 | 1.4 | 1.3

-0.9 | -1.5 | -0.7 | 0.9 | 0.2

-0.1 | -0.7 | -1.6 | 0.2 | 0.6

I

loved

the

movie

!

Attention

- Lots of variations on attention:
  - Linear transformation of x into before dotting with v
  - Non-linearities after each operation.
  - “Multi-head attention”: multiple v vectors to capture different phenomena that can be attended to in the input.
  - Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).

Parts of speech

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

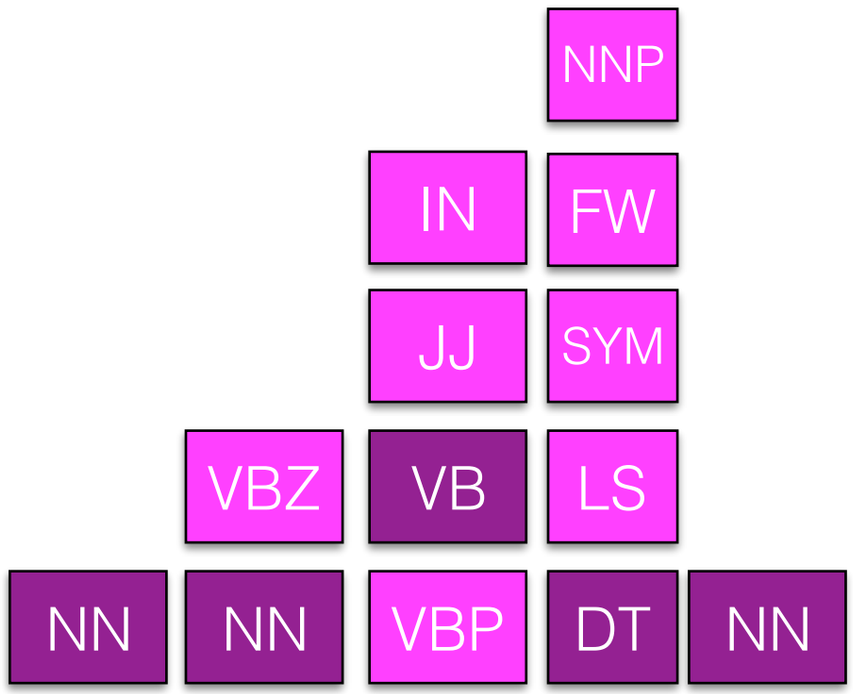
	-s	-ed	-ing
walk	walks	walked	walking
slice	slices	sliced	slicing
believe	believes	believed	believing
of	*ofs	*ofed	*ofing
red	*reds	*redded	*reding

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

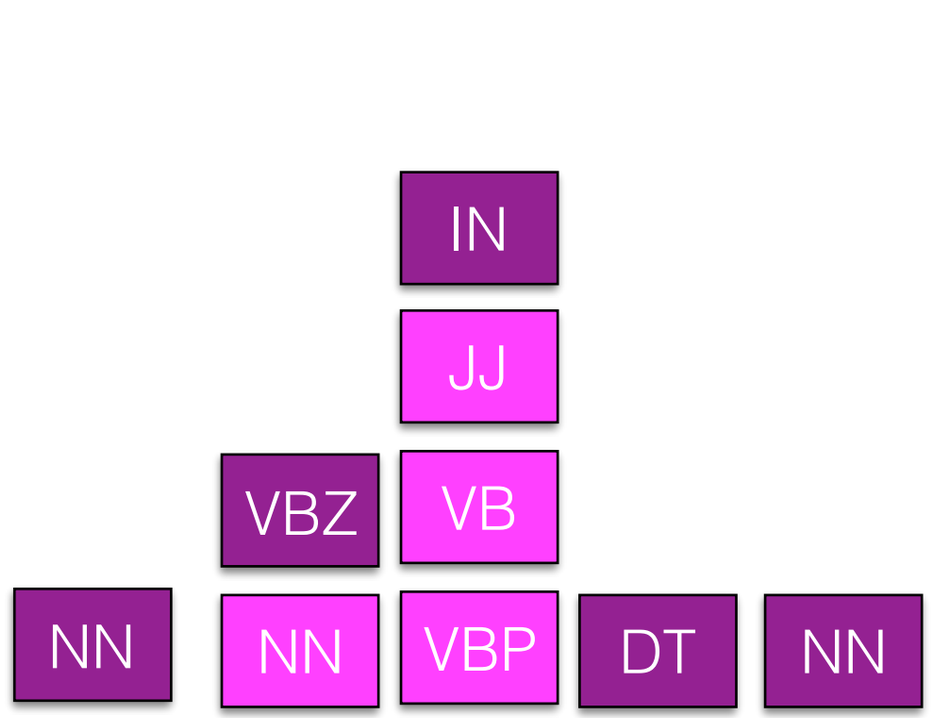
Open class	Nouns	fax, affluenza, subtweet, bitcoin, cronut, emoji, listicle, mocktail, selfie, skort
	Verbs	text, chillax, manspreading, photobomb, unfollow, google
	Adjectives	crunk, amazeballs, post-truth, woke
	Adverbs	hella, wicked
Closed class	Determiner	
	Pronouns	
	Prepositions	English has a new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

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!)

Sequence labeling

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

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

- For a set of inputs x with n sequential time steps, one corresponding label y_i for each x_i
- Model correlations in the labels y.

HMM

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

Hidden Markov Model

Prior probability of label sequence

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

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

- 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

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

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

- 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

$$P(y_t \mid y_{t-1})$$

$$\frac{c(y_1, y_2)}{c(y_1)}$$

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

$$P(x_t \mid y_t)$$

$$\frac{c(x, y)}{c(y)}$$

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

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**
    $$\textit{viterbi}[s,t] \leftarrow \max_{s'=1}^{N} \textit{viterbi}[s',t-1] * a_{s',s} * b_s(o_t)$$
    $$\textit{backpointer}[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \textit{viterbi}[s',t-1] * a_{s',s}$$
$$\textit{viterbi}[q_F,T] \leftarrow \max_{s=1}^{N} \textit{viterbi}[s,T] * a_{s,q_F} \quad \text{; termination step}$$
$$\textit{backpointer}[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} \textit{viterbi}[s,T] * a_{s,q_F} \quad \text{; termination step}$$
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.

MEMM

General maxent form

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

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

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

Features

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

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

feature	example
x_i = man	1
t_{i-1} = 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

will to fight

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

Because of this local normalization, P(TO | context) will always be 1 if x="to"

Toutanova et al. 2003

Label bias

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

Toutanova et al. 2003

Conditional random fields

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

MEMM

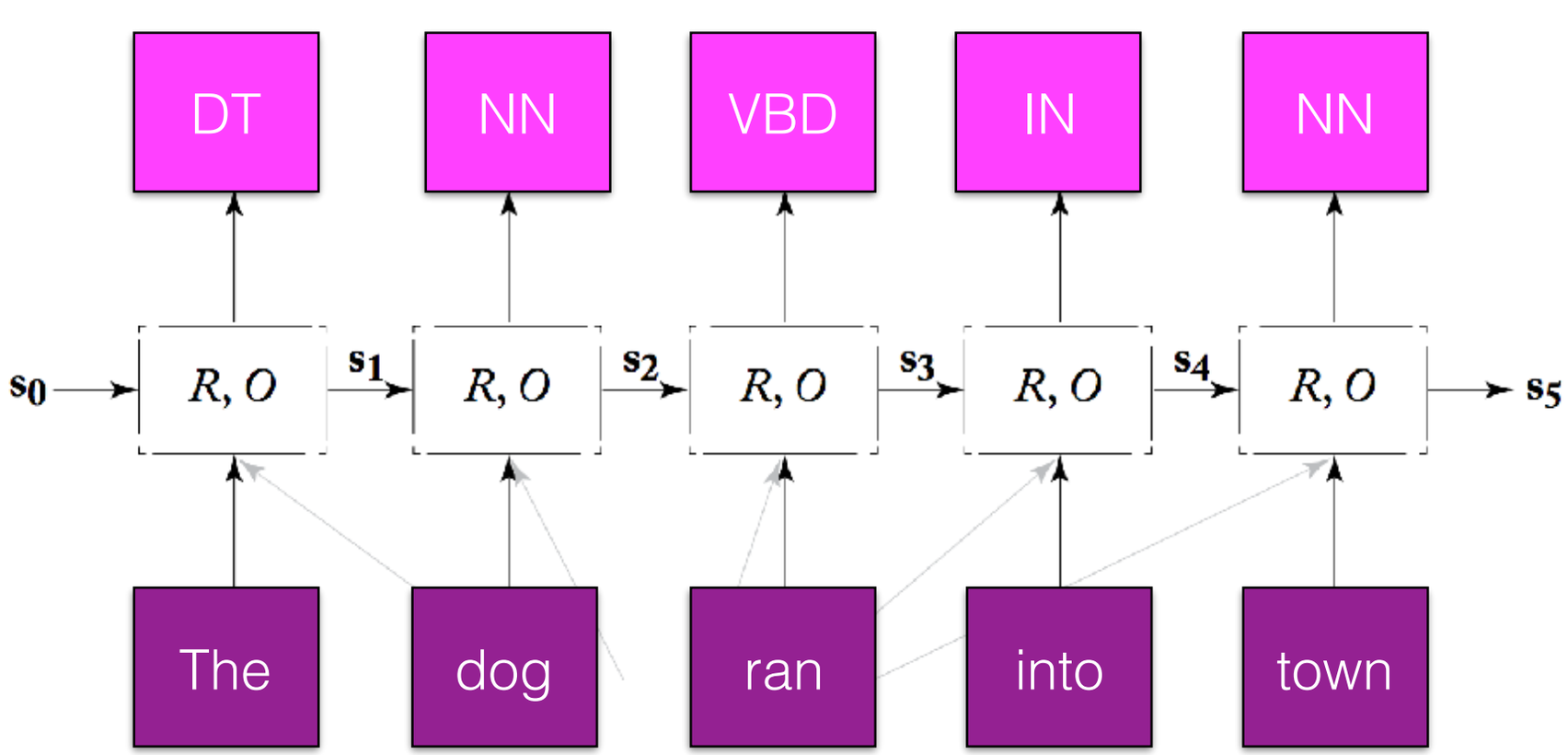
$$P(y \mid x, \beta) = \prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

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 $\boldsymbol{y}$ conditioned on the context



Bidirectional RNN

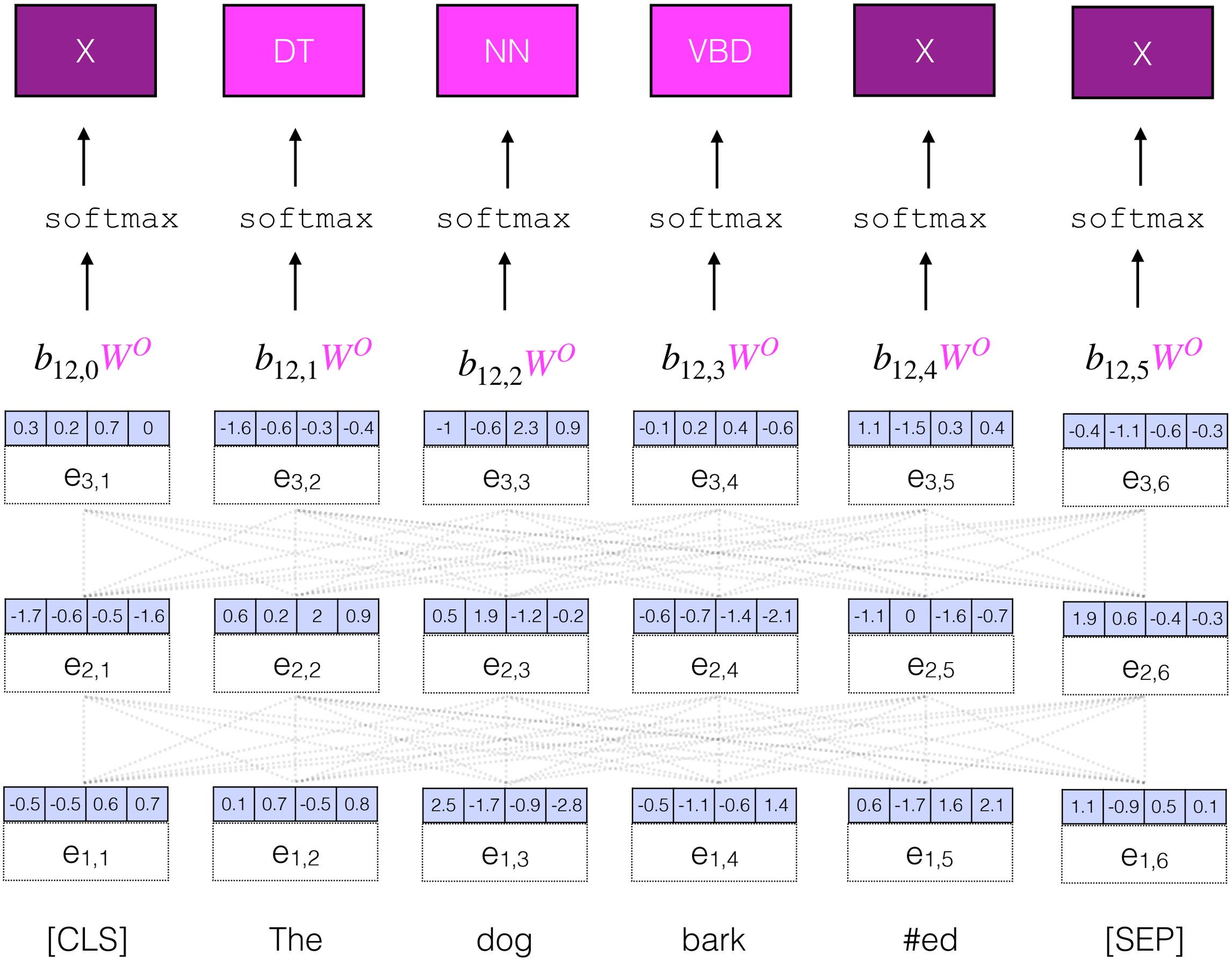
- A powerful alternative is make predictions conditioning both on the past and the future.

- Two RNNs
    - One running left-to-right
    - One right-to-left

- Each produces an output vector at each time step, which we concatenate

BERT

- BERT can be used not only as a language model to generate contextualized word representations, but also as a predictive model whose parameters are fine-tuned to a task.


X
DT
NN
VBD
X
X
softmax
softmax
softmax
softmax
softmax
softmax
$b_{12,0}W^O$
$b_{12,1}W^O$
$b_{12,2}W^O$
$b_{12,3}W^O$
$b_{12,4}W^O$
$b_{12,5}W^O$
0.3 0.2 0.7 0
-1.6 -0.6 -0.3 -0.4
-1 -0.6 2.3 0.9
-0.1 0.2 0.4 -0.6
1.1 -1.5 0.3 0.4
-0.4 -1.1 -0.6 -0.3
$e_{3,1}$
$e_{3,2}$
$e_{3,3}$
$e_{3,4}$
$e_{3,5}$
$e_{3,6}$
-1.7 -0.6 -0.5 -1.6
0.6 0.2 2 0.9
0.5 1.9 -1.2 -0.2
-0.6 -0.7 -1.4 -2.1
-1.1 0 -1.6 -0.7
1.9 0.6 -0.4 -0.3
$e_{2,1}$
$e_{2,2}$
$e_{2,3}$
$e_{2,4}$
$e_{2,5}$
$e_{2,6}$
-0.5 -0.5 0.6 0.7
0.1 0.7 -0.5 0.8
2.5 -1.7 -0.9 -2.8
-0.5 -1.1 -0.6 1.4
0.6 -1.7 1.6 2.1
1.1 -0.9 0.5 0.1
$e_{1,1}$
$e_{1,2}$
$e_{1,3}$
$e_{1,4}$
$e_{1,5}$
$e_{1,6}$
[CLS]
The
dog
bark
#ed
[SEP]

Sequence labeling models

model	form	label dependency	rich features?
Hidden Markov Models	$\prod_{i=1}^{N} P(x_i \mid y_i)\, P(y_i \mid y_{i-1})$	Markov assumption	no
MEMM	$\prod_{i=1}^{N} P(y_i \mid y_{i-1}, x, \beta)$	Markov assumption	yes
CRF	$P(y \mid x, \beta)$	pairwise through entire sequence	yes
RNN	$\prod_{i=1}^{N} P(y_i \mid x_{1:i}, \beta)$	none	distributed

Evaluation

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

Experiment design

	training	development	testing
size	80%	10%	10%
purpose	training models	model selection	evaluation; never look at it until the very end

Accuracy

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

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

Predicted (ŷ)

True (y)	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

Precision

Precision(NN) =

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

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

True (y) \ Predicted (ŷ)	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

Recall

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

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

Predicted (ŷ)

True (y)

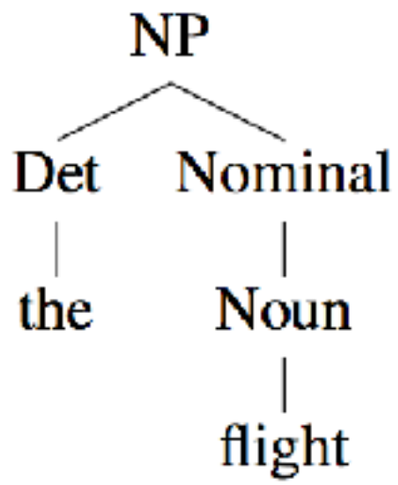
	NN	VBZ	JJ
NN	100	2	15
VBZ	0	104	30
JJ	30	40	70

F score

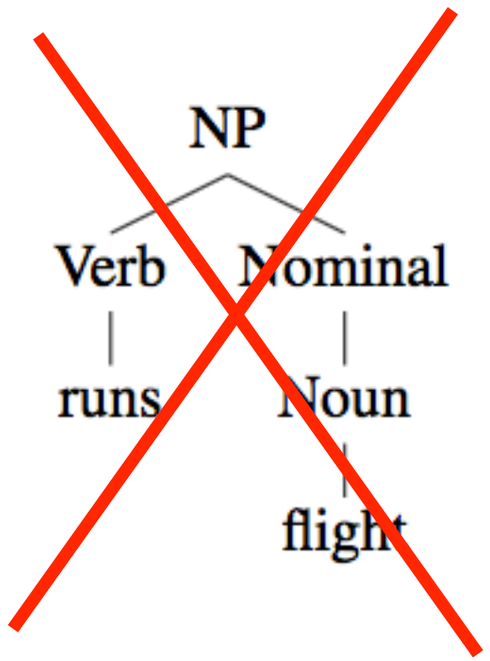
$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

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.

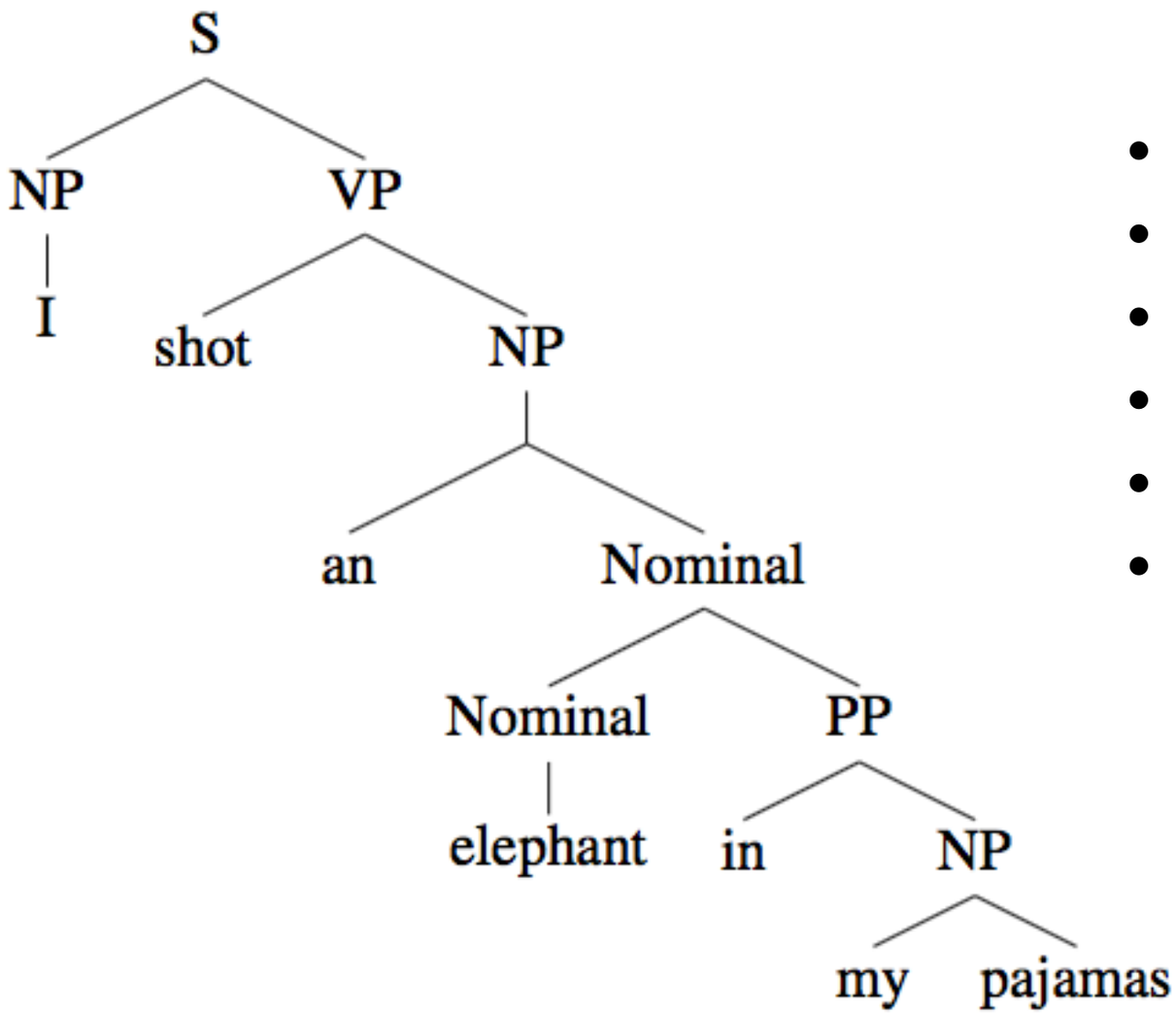


NP → Det Nominal



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

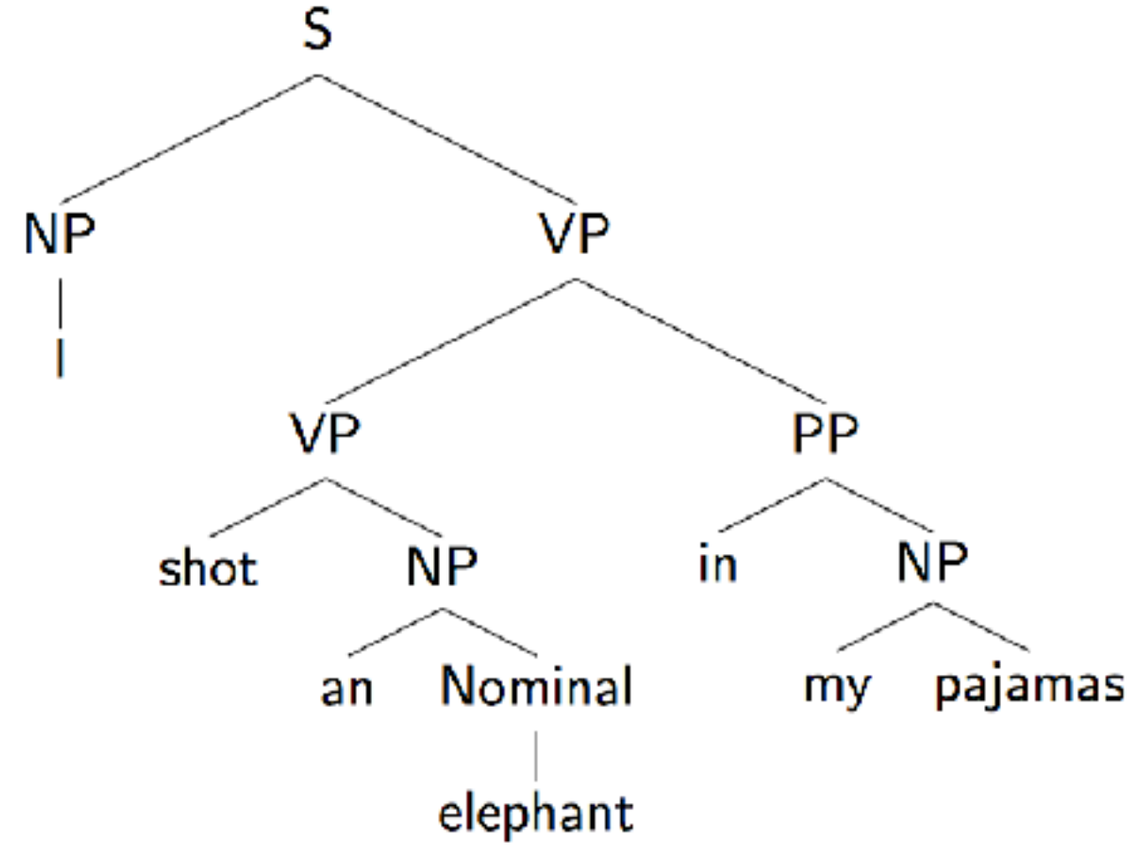
Each phrase could be replaced by another of the same type of constituent

Evaluation

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

<l_1, i_1, j_1>, …, <l_n, i_n, j_n>

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

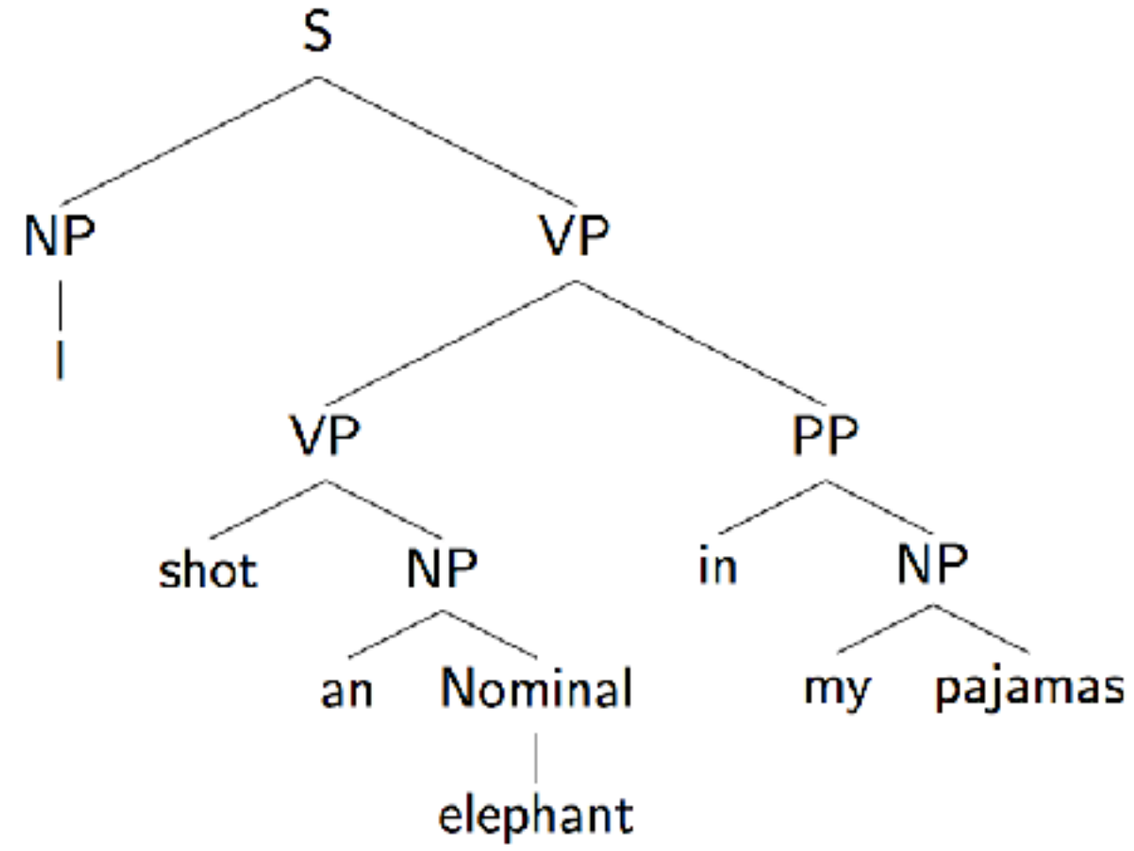


Smith 2017

Evaluation

I_1 $shot_2$ an_3 $elephant_4$ in_5 my_6 $pajamas_7$

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



Smith 2017

Evaluation

I_1 $shot_2$ an_3 $elephant_4$ in_5 my_6 $pajamas_7$

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

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

Smith 2017

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

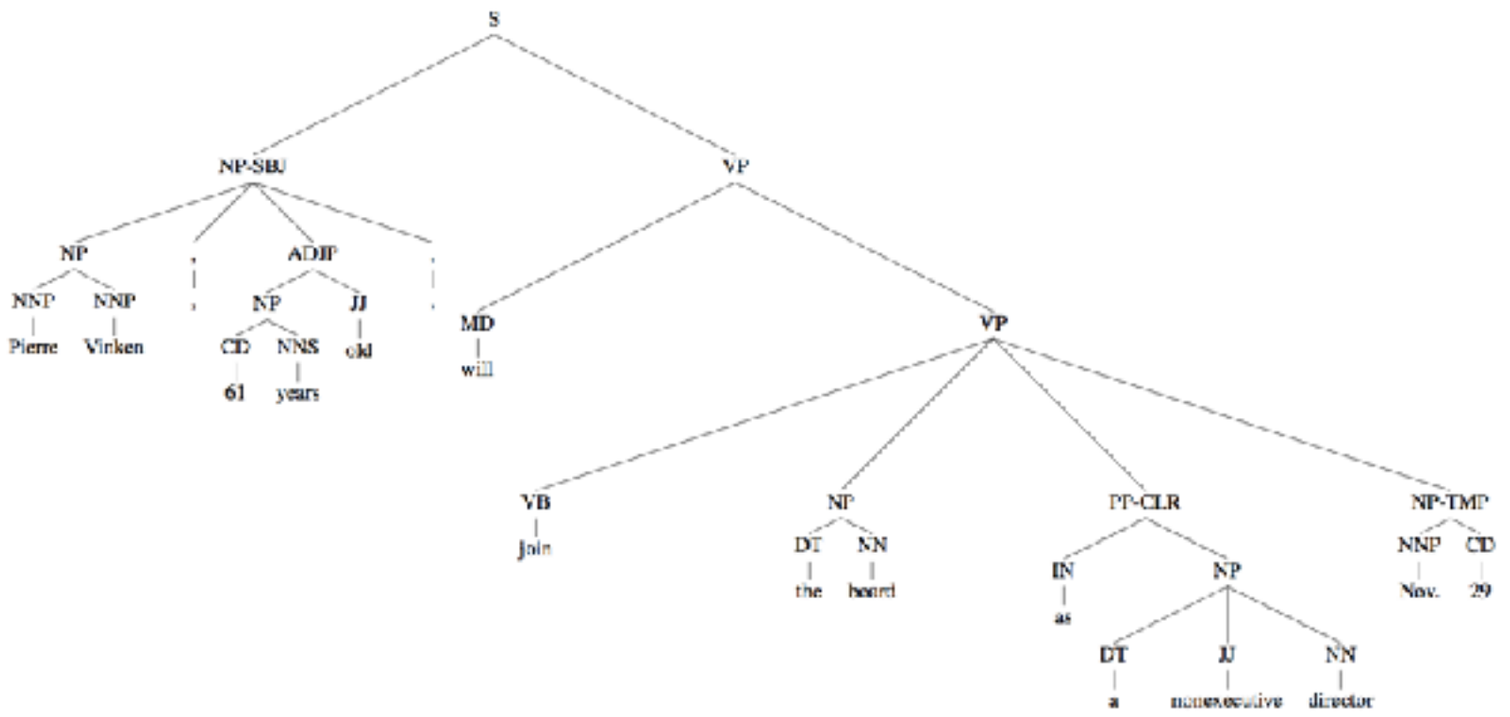
Smith 2017

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 → β):

$$P(T, S) = \prod_{i}^{n} P(\beta \mid A)$$

Estimating PCFGs

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

(equivalently)

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

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
NP, PRP [0,1]	∅					
	VBD [1,2]	∅				
		DT [2,3]				
			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]	∅				
		DT [2,3]				
			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]	∅				
		DT [2,3]				
			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]	∅				
		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]	∅				
		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]	∅	∅				
	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]	∅	∅				
	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]	∅	∅				
	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]

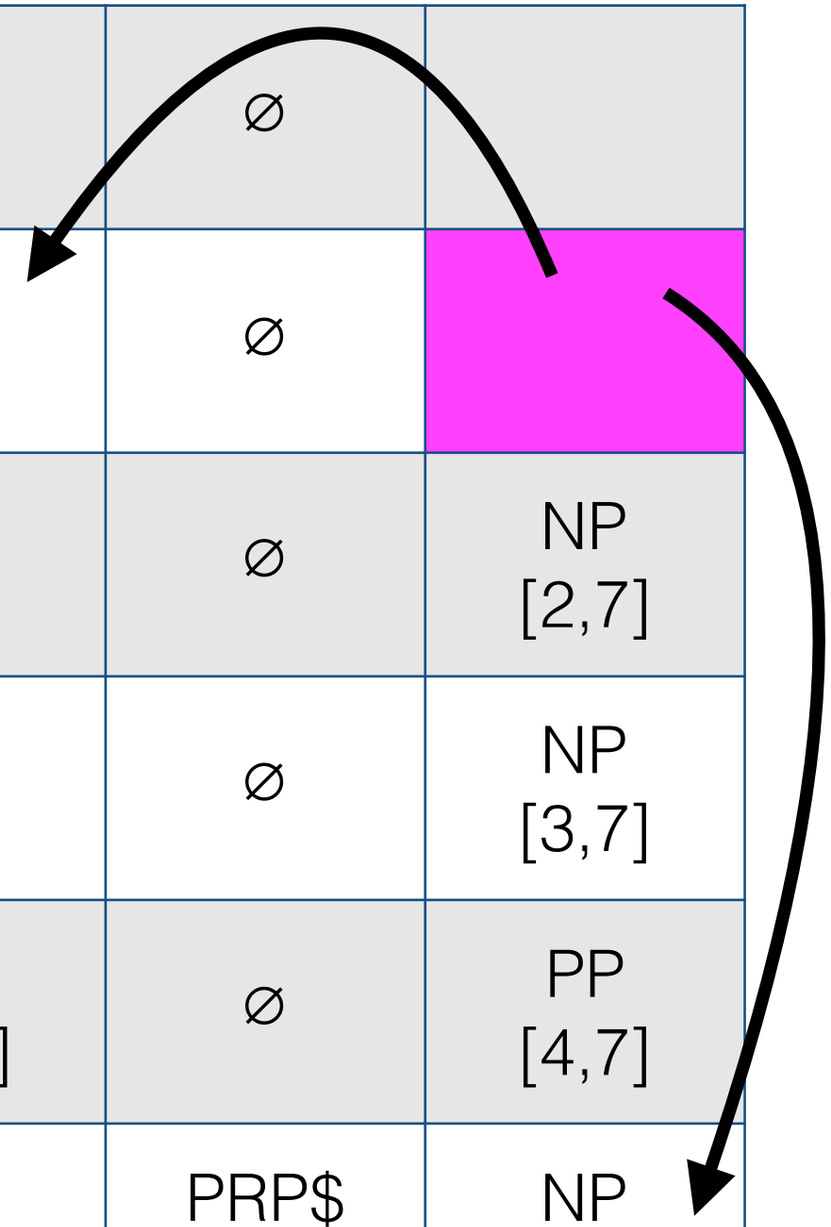
I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		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]

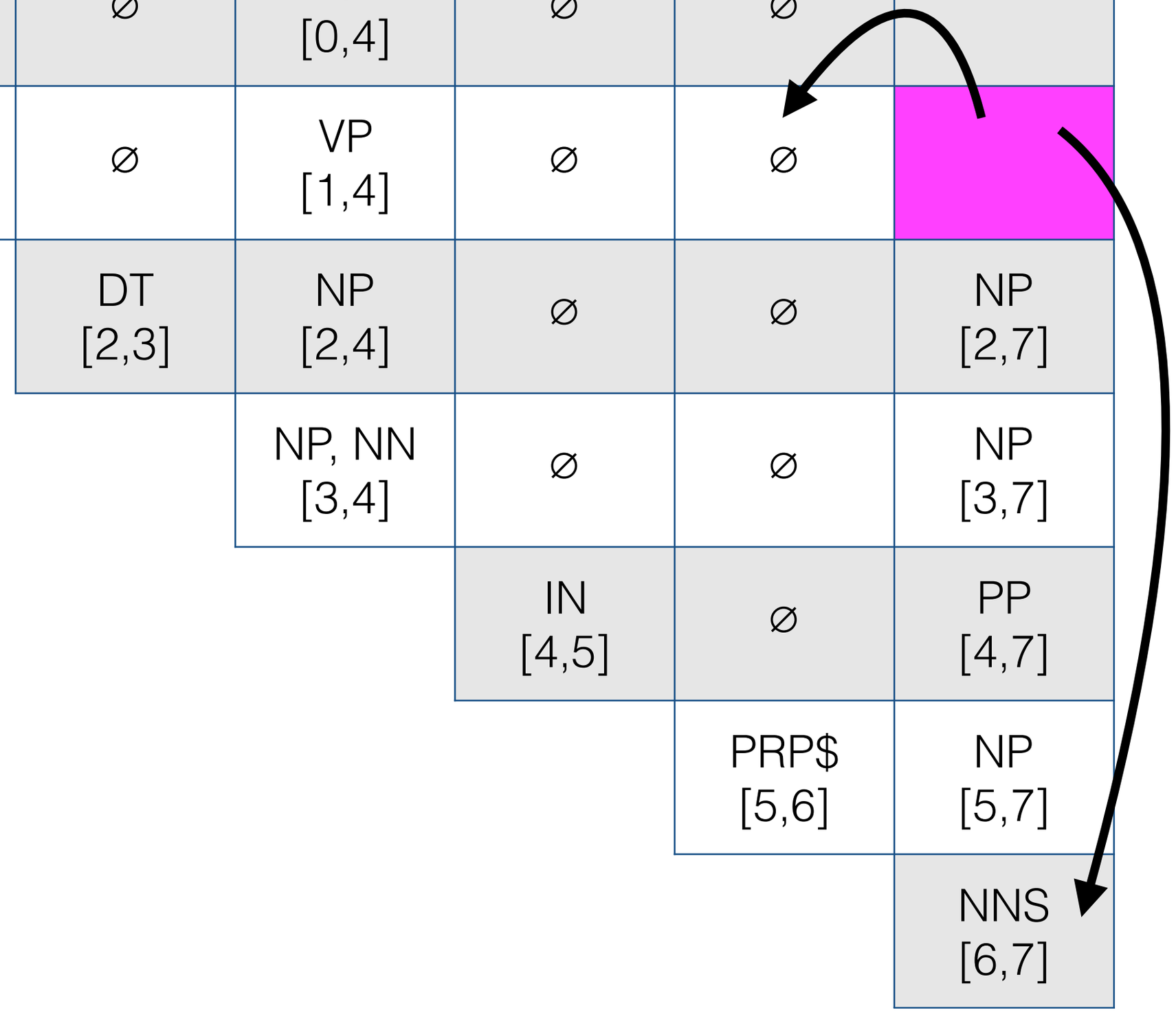
I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		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]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		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]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		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]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		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]	∅	∅	
	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]	∅	∅	
	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]

Possibilities:

$S_1 \rightarrow NP\ VP_1$
$S_2 \rightarrow NP\ VP_2$
$? \rightarrow S\ PP$
$? \rightarrow PRP\ VP_1$
$? \rightarrow PRP\ VP_2$

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 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.

I	shot	an	elephant	in	my	pajamas
PRP: -3.21 [0,1]	∅	∅	S: -19.2 [0,4]	∅	∅	S: -35.7 [0,7]
	VBD: -3.21 [1,2]	∅	VP: -14.3 [1,4]	∅	∅	VP: -30.2 [1,7]
		DT: -3.0 [2,3]	NP: -8.8 [2,4]	∅	∅	NP: -24.7 [2,7]
			NN: -3.5 [3,4]	∅	∅	NP: -19.4 [3,7]
				IN: -2.3 [4,5]	∅	PP: -13.6 [4,7]
					PRP$: -2.12 [5,6]	NP: -9.0 [5,7]
						NNS: -4.6 [6,7]

As in Viterbi, backpointers let us keep track on the path through the chart that leads to the best derivation

Midterm

- Take-home exam — you'll have 80 continuous minutes to take the midterm anytime within the 24-hour period (3:30pm 3/12—2:59pm 3/13).
- Mix of multiple choice, short answer, long answer
- Covers all material from lectures and readings