Natural Language Processing

Info 159/259

Lecture 7: Language models 2 (Feb 11, 2020)

David Bamman, UC Berkeley

My favorite food is …

- … is the food I love you too
- is the best food I’ve ever had
- is the worst I can get
- is the best food places in the area
- is the best beer
- is in my coffee
- is here and there are a lot better options
- is a good friend

Language Model

- Vocabulary $\mathcal{V}$ is a finite set of discrete symbols (e.g., words, characters); $V = |\mathcal{V}|$
- $\mathcal{V}^+$ is the infinite set of sequences of symbols from $\mathcal{V}$; each sequence ends with STOP
- $x \in \mathcal{V}^+$

Language Model

Language modeling is the task of estimating $P(w)$

Language Model

arma virumque cano

- arms man and I sing
- Arms, and the man I sing [Dryden]
- I sing of arms and a man

- When we have choices to make about different ways to say something, a language models gives us a formal way of operationalizing their fluency
 (in terms of how likely they are exist in the language)

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)

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1})$$

$$\times P(\text{STOP} \mid w_{n-1}, w_n)$$

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

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 $\mathcal{Y}$

$\mathcal{Y}$ = {the, of, a, dog, iphone, …}

x = (context)
y = word

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

$$P(Y = 1 \mid X = x; \beta) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}$$

$$= \frac{\exp(x^\top \beta)}{\exp(x^\top 0) + \exp(x^\top \beta)}$$

Logistic regression

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

output space $\mathcal{Y} = \{1, \ldots, K\}$

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_1	β_2	β_3	β_4	β_5
the	1.33	-0.80	-0.54	0.87	0
and	1.21	-1.73	-1.57	-0.13	0
bravest	0.96	-0.05	0.24	0.81	0
love	1.49	0.53	1.01	0.64	0
loved	-0.52	-0.02	2.21	-2.53	0
genius	0.98	0.77	1.53	-0.95	0
not	-0.96	2.14	-0.71	0.43	0
fruit	0.59	-0.76	0.93	0.03	0
BIAS	-1.92	-0.70	0.94	-0.63	0

Language Model

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

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

Unigram LM

- A unigram language model here would have just one feature: a bias term.

Feature	β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
BIAS	-1.92	-0.70	0.94	-0.63	0

Bigram LM

Feature	Value
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=brave	0
w_{i-1}=love	0
w_{i-1}=loved	0
w_{i-1}=dog	0
w_{i-1}=not	0
w_{i-1}=fruit	0
BIAS	1

β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
-0.78	-0.80	-0.54	0.87	0
1.21	-1.73	-1.57	-0.13	0
0.96	-0.05	0.24	0.81	0
1.49	0.53	1.01	0.64	0
-0.52	-0.02	2.21	-2.53	0
0.98	0.77	1.53	-0.95	0
-0.96	2.14	-0.71	0.43	0
0.59	-0.76	0.93	0.03	0
-1.92	-0.70	0.94	-0.63	0

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

Feature	Value
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=brave	0
w_{i-1}=love	0
w_{i-1}=loved	0
w_{i-1}=dog	0
w_{i-1}=not	0
w_{i-1}=fruit	0
BIAS	1

β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
-0.78	-0.80	-0.54	0.87	0
1.21	-1.73	-1.57	-0.13	0
0.96	-0.05	0.24	0.81	0
1.49	0.53	1.01	0.64	0
-0.52	-0.02	2.21	-2.53	0
0.98	0.77	1.53	-0.95	0
-0.96	2.14	-0.71	0.43	0
0.59	-0.76	0.93	0.03	0
-1.92	-0.70	0.94	-0.63	0

Trigram LM

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

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-2}=loved ^ w_{i-1}=the	0
w_{i-2}=genius ^ w_{i-1}=the	0
w_{i-2}=not ^ w_{i-1}=the	0
w_{i-2}=fruit ^ w_{i-1}=the	0
BIAS	1

Smoothing

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

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

L2 regularization

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

- 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)

L1 regularization

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

- L1 regularization encourages coefficients to be exactly 0.

- η again controls how much of a penalty to pay for coefficients that are far from 0 (optimize on development data)

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.

“JACKSONVILLE, Fla. — Stressed and exhausted families across the Southeast were assessing the damage from Hurricane Irma on Tuesday, even as flooding from the storm continued to plague some areas, like Jacksonville, and the worst of its wallop was being revealed in others, like the Florida Keys.

Officials in Florida, Georgia and South Carolina tried to prepare residents for the hardships of recovery from the ______________”

https://www.nytimes.com/2017/09/12/us/irma-jacksonville-florida.html

“WASHINGTON — Senior Justice Department officials intervened to overrule front-line prosecutors and will recommend a more lenient sentencing for Roger J. Stone Jr., convicted last year of impeding investigators in a bid to protect his longtime friend President _________”

https://www.nytimes.com/2020/02/11/us/politics/roger-stone-sentencing.html

Kim Kardashian West

@KimKardashian

Follow

OMG I had to take the whole family to go see The Fault In Our Stars ✨ This movie _____

"WASHINGTON — Senior Justice Department officials intervened to overrule front-line prosecutors and will recommend a more lenient sentencing for Roger J. Stone Jr., convicted last year of impeding investigators in a bid to protect his longtime friend President ________"

feature classes	example
ngrams (w_{i-1}, $w_{i-2}:w_{i-1}$, $w_{i-3}:w_{i-1}$)	w_{i-2}="friend", w_i="donald"
gappy ngrams	w_1="stone" and w_2="donald"
spelling, capitalization	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

Classes

bigram	count
black car	100
blue car	37
red car	0

bigram	count
<COLOR> car	137

Tradeoffs

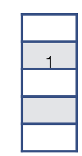
- Richer representations = more parameters, higher likelihood of overfitting
- Much slower to train than estimating the parameters of a generative model

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

Neural LM

Simple feed-forward multilayer perceptron
(e.g., one hidden layer)

input x = vector concatenation of a conditioning context of fixed size k

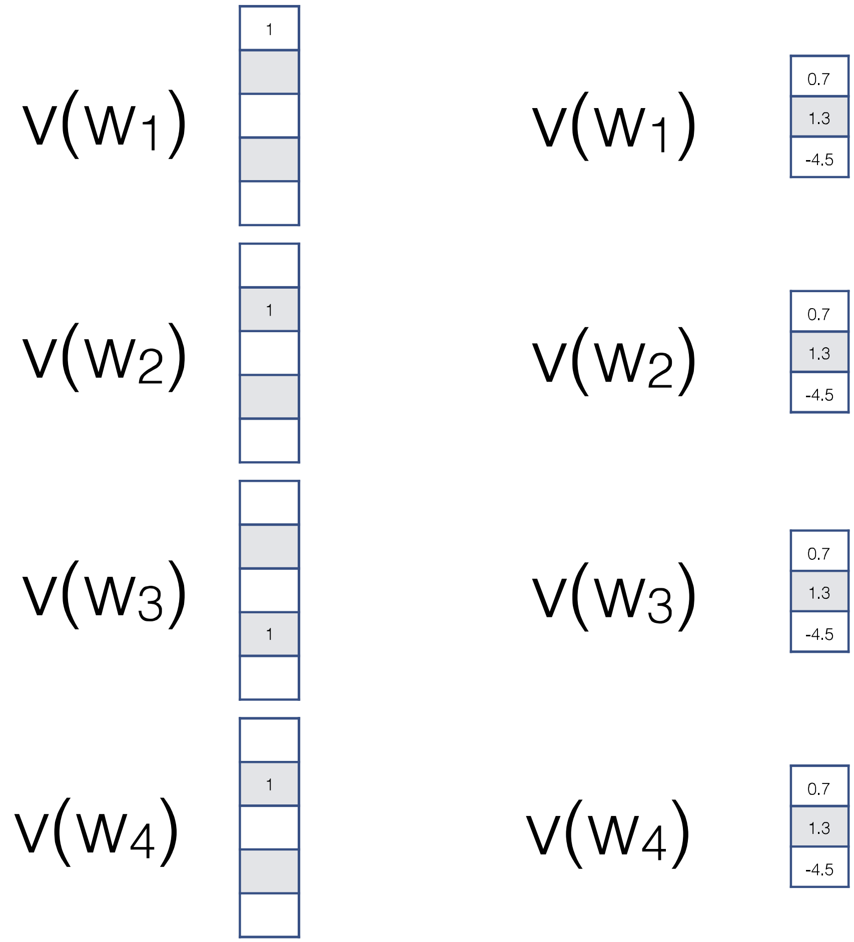


$$x = [v(w_1); \ldots ; v(w_k)]$$

Bengio et al. 2003

$$x = [v(w_1); \ldots; v(w_k)]$$

w_1 = tried
w_2 = to
w_3 = prepare
w_4 = residents



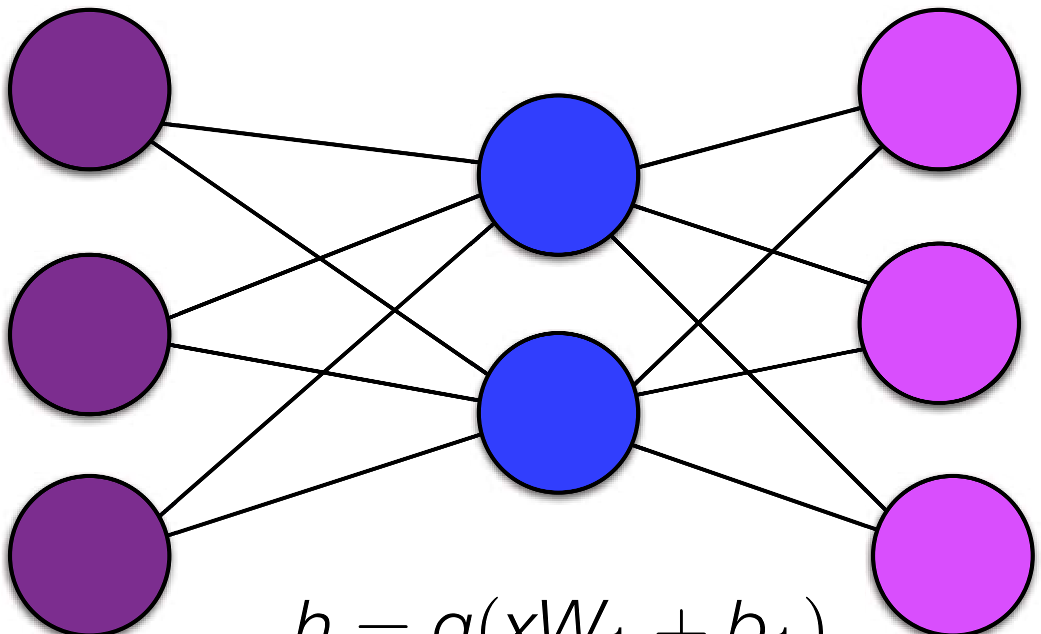
one-hot encoding

distributed representation

Bengio et al. 2003

$$W_1 \in \mathbb{R}^{kD \times H} \qquad W_2 \in \mathbb{R}^{H \times V}$$

$$b_1 \in \mathbb{R}^{H} \qquad b_2 \in \mathbb{R}^{V}$$



$$h = g(xW_1 + b_1)$$

$$x = [v(w_1); \ldots; v(w_k)] \qquad \hat{y} = \text{softmax}(hW_2 + b_2)$$

Bengio et al. 2003

Softmax

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

Neural LM

conditioning context

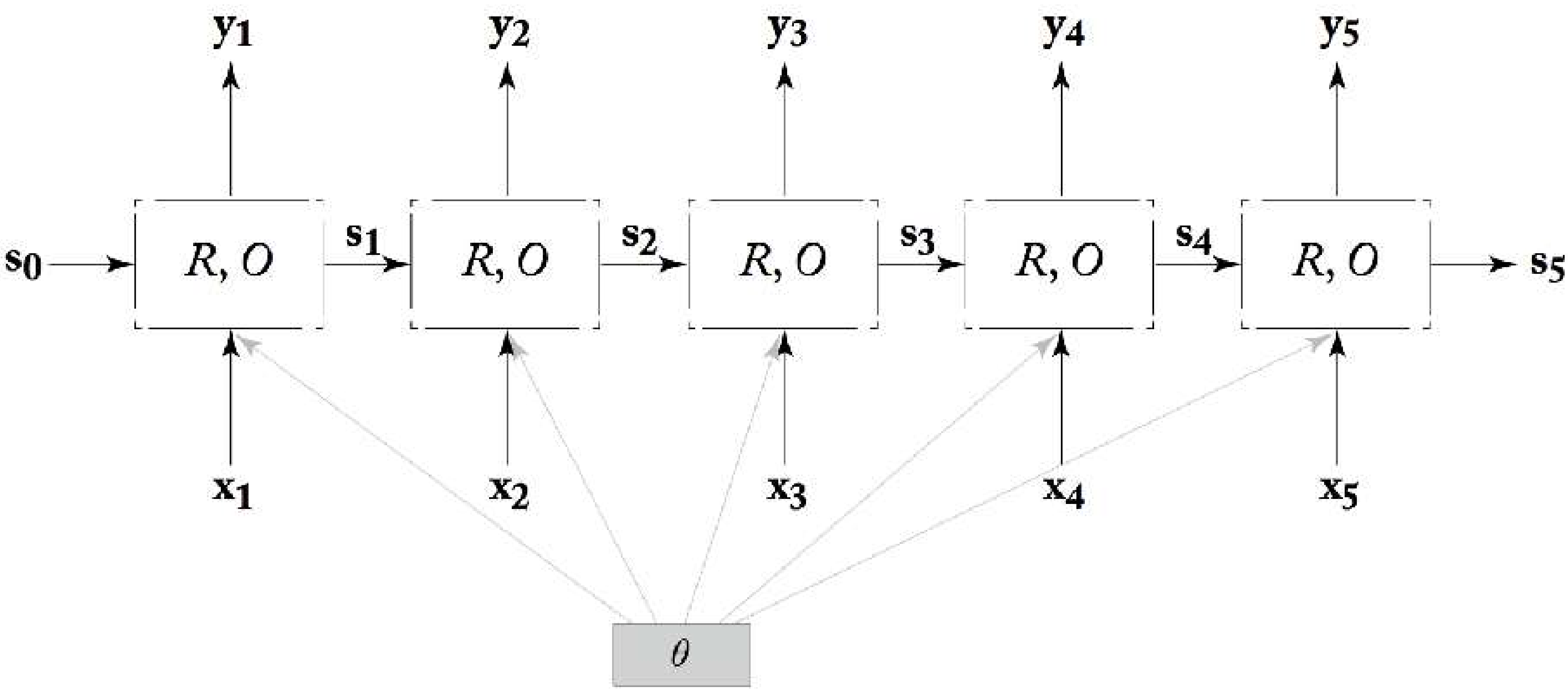
tried to prepare residents for the hardships of recovery from the

y

Recurrent neural network

- RNN allow arbitarily-sized conditioning contexts; condition on the entire sequence history.

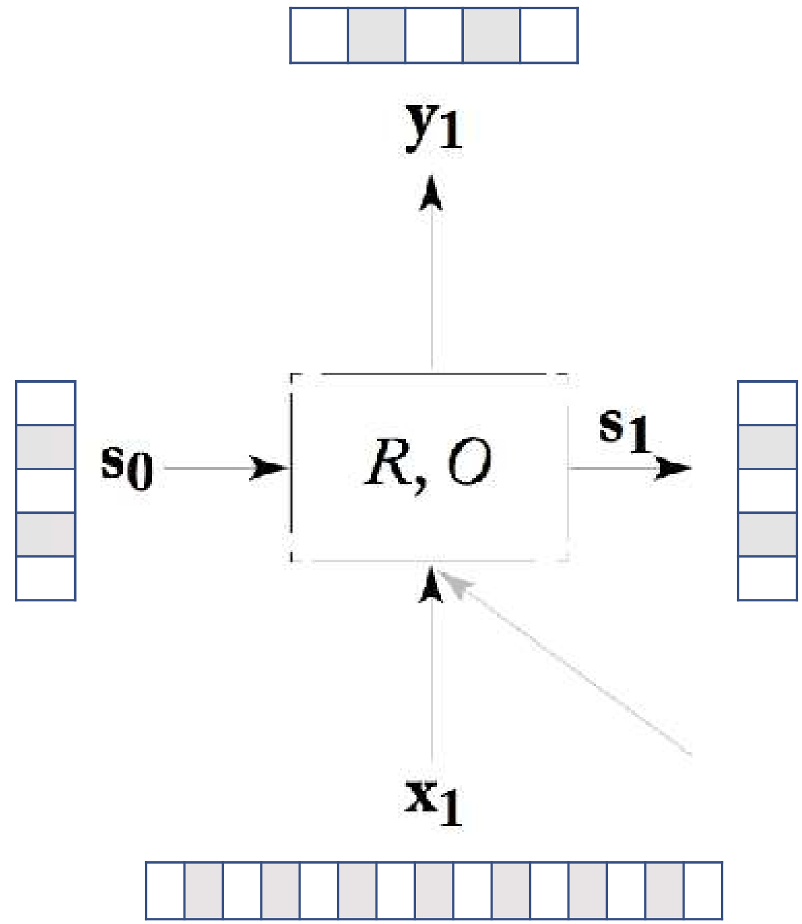
Recurrent neural network



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



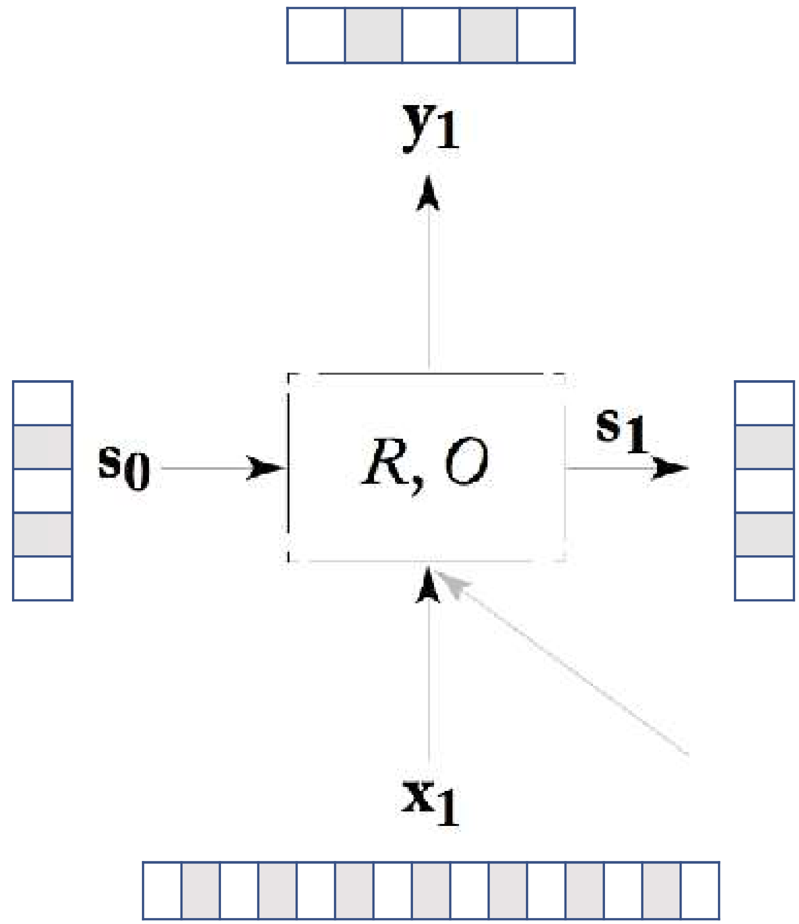
Recurrent neural network

$$s_i = R(x_i, s_{i-1})$$

R computes the output state as a function of the current input and previous state

$$y_i = O(s_i)$$

O computes the output as a function of the current output state



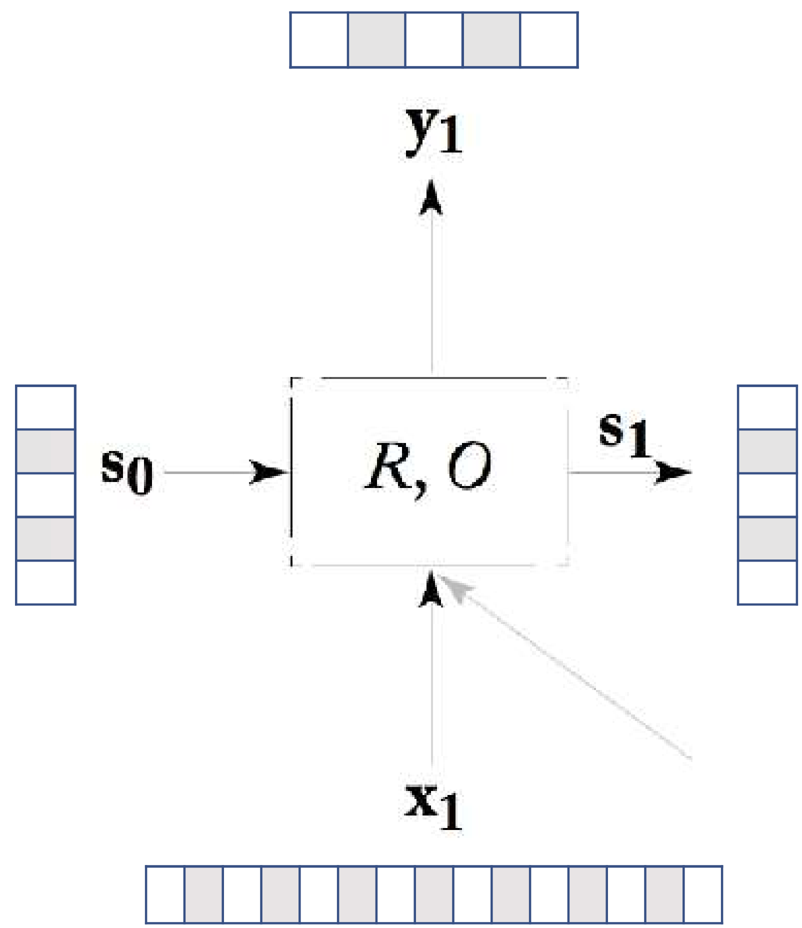
Continuous bag of words

$$s_i = R(x_i, s_{i-1}) = s_{i-1} + x_i$$

Each state is the sum of all previous inputs

$$y_i = O(s_i) = s_i$$

Output is the identity



“Simple” RNN

g = tanh or relu

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

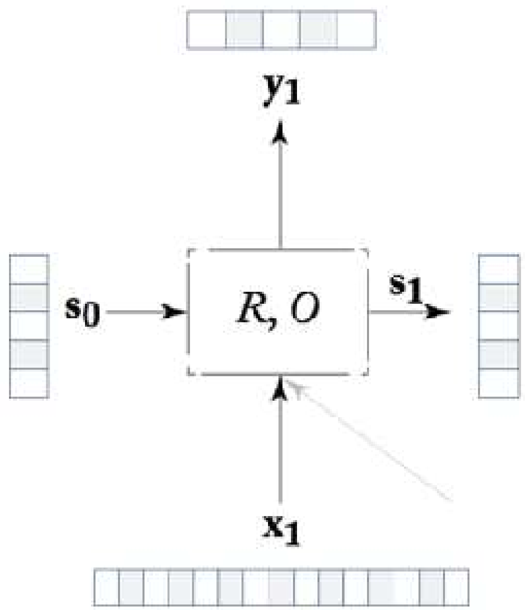
Different weight vectors W transform the previous state and current input before combining

$$W^s \in \mathbb{R}^{H \times H}$$

$$W^x \in \mathbb{R}^{D \times H}$$

$$b \in \mathbb{R}^{H}$$

$$y_i = O(s_i) = s_i$$

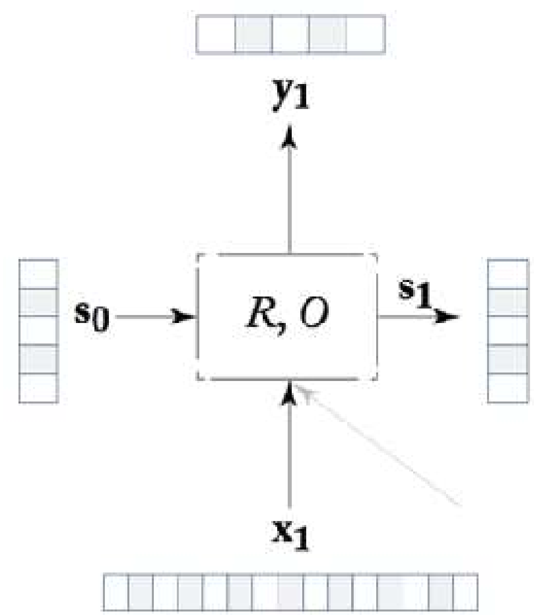


Elman 1990, Mikolov 2012

RNN LM

- The output state s_i is an H-dimensional real vector; we can transfer that into a probability by passing it through an additional linear transformation followed by a softmax

$$y_i = O(s_i) = \mathrm{softmax}(s_i W^o + b^o)$$



Elman 1990, Mikolov 2012

Training RNNs

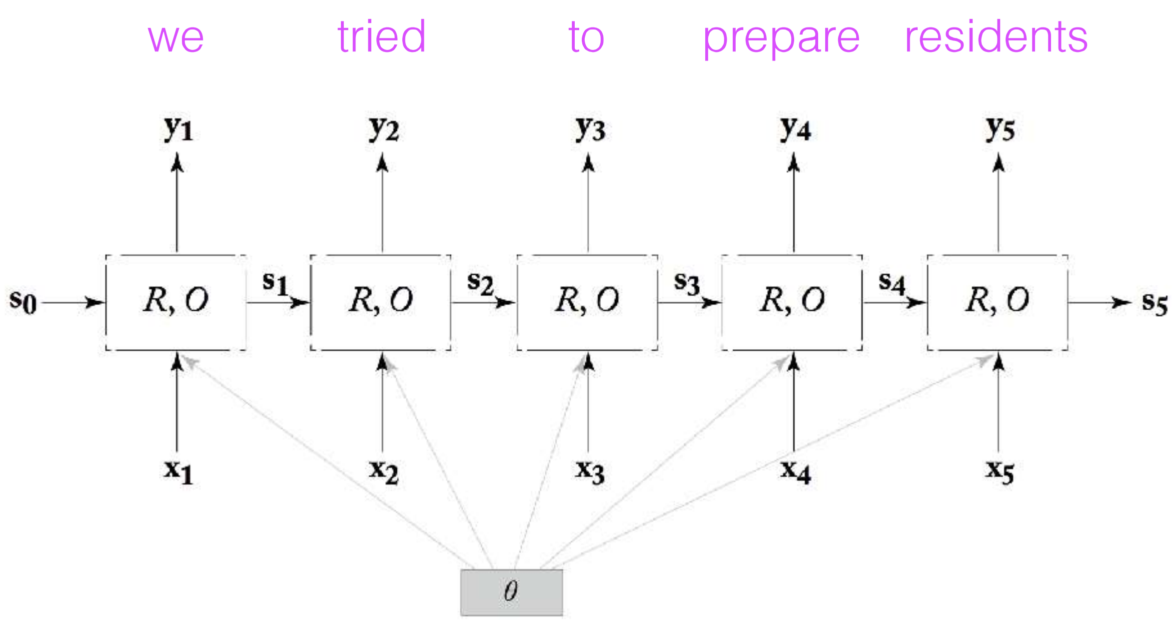
- Given this definition of an RNN:

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$
$$y_i = O(s_i) = \text{softmax}(s_iW^o + b^o)$$

- We have five sets of parameters to learn:

$$W^s, W^x, W^o, b, b^o$$

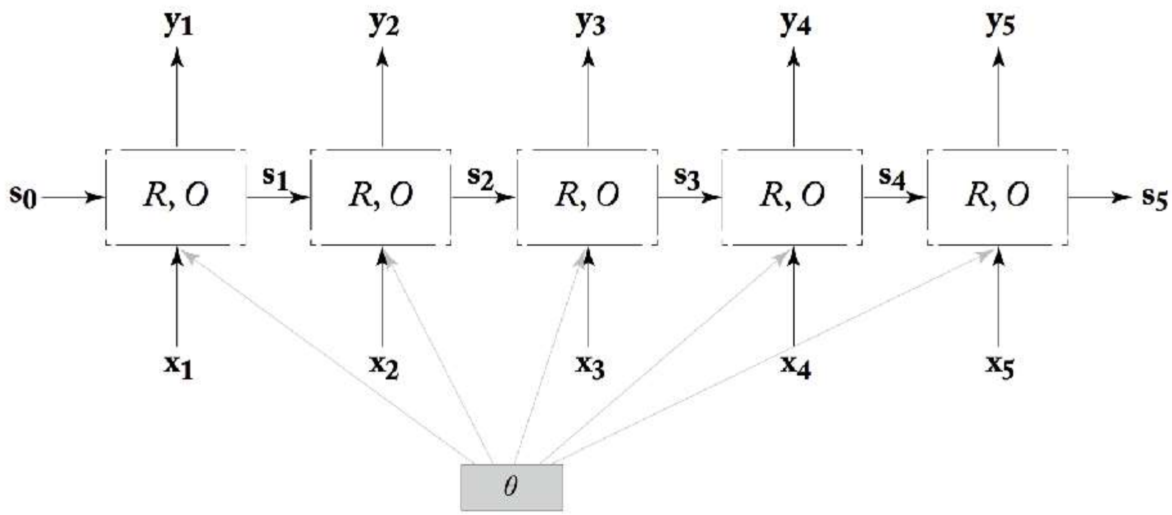
Training RNNs



- At each time step, we make a prediction and incur a loss; we know the true y (the word we see in that position)

$$\frac{\partial L(\theta)_{y_1}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_2}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_3}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_4}}{\partial W^s} \quad \frac{\partial L(\theta)_{y_5}}{\partial W^s}$$

we tried to prepare residents



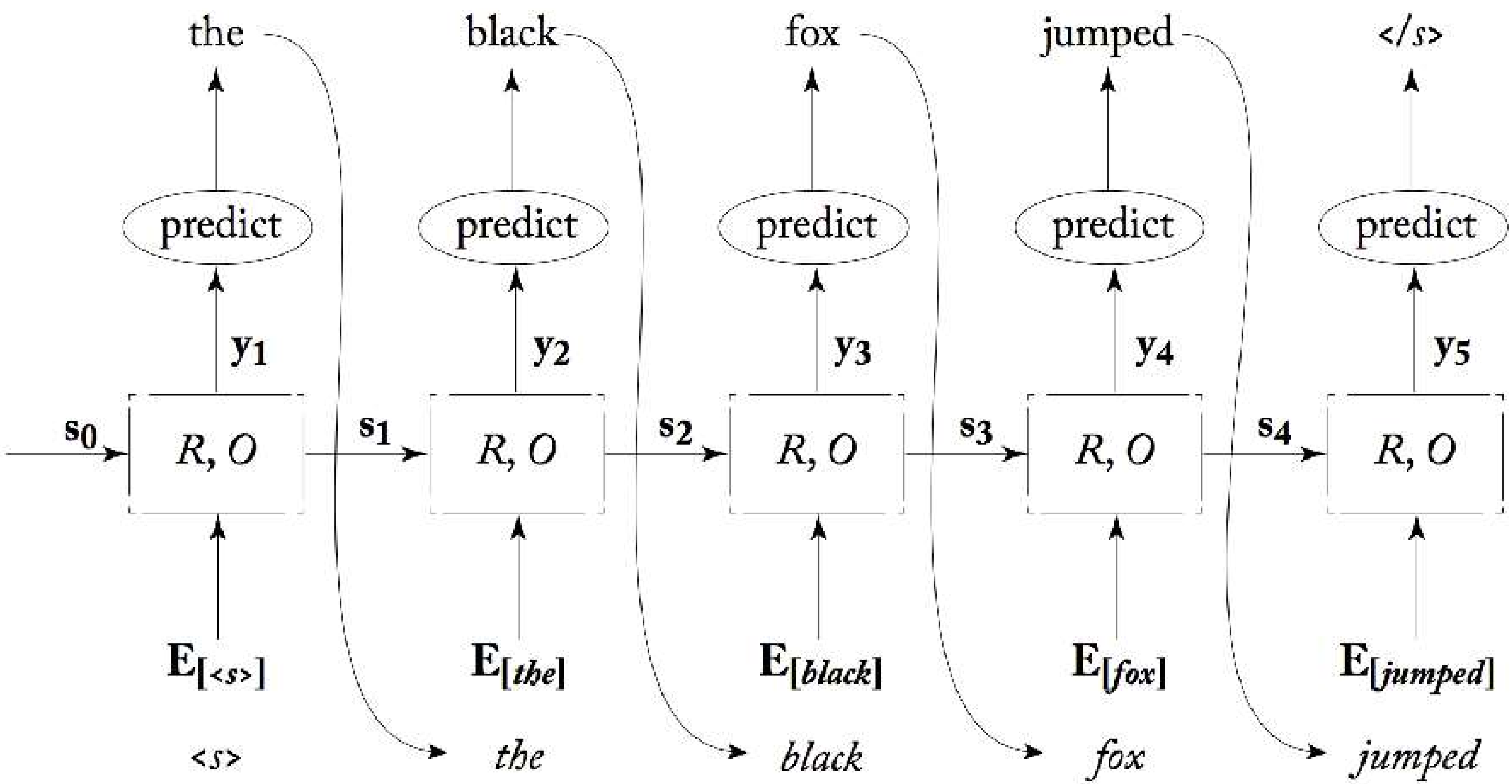
- Training here is standard backpropagation, taking the derivative of the loss we incur at step *t* with respect to the parameters we want to update

Generation

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked
dog	walked	in

Generation



the
black
fox
jumped
</s>
predict
predict
predict
predict
predict
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_3$
$\mathbf{y}_4$
$\mathbf{y}_5$
$\mathbf{s}_0$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$\mathbf{s}_4$
R, O
R, O
R, O
R, O
R, O
$\mathbf{E}_{[<s>]}$
$\mathbf{E}_{[the]}$
$\mathbf{E}_{[black]}$
$\mathbf{E}_{[fox]}$
$\mathbf{E}_{[jumped]}$
<s>
the
black
fox
jumped

Conditioned generation

- In a basic RNN, the input at each timestep is a representation of the word at that position

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

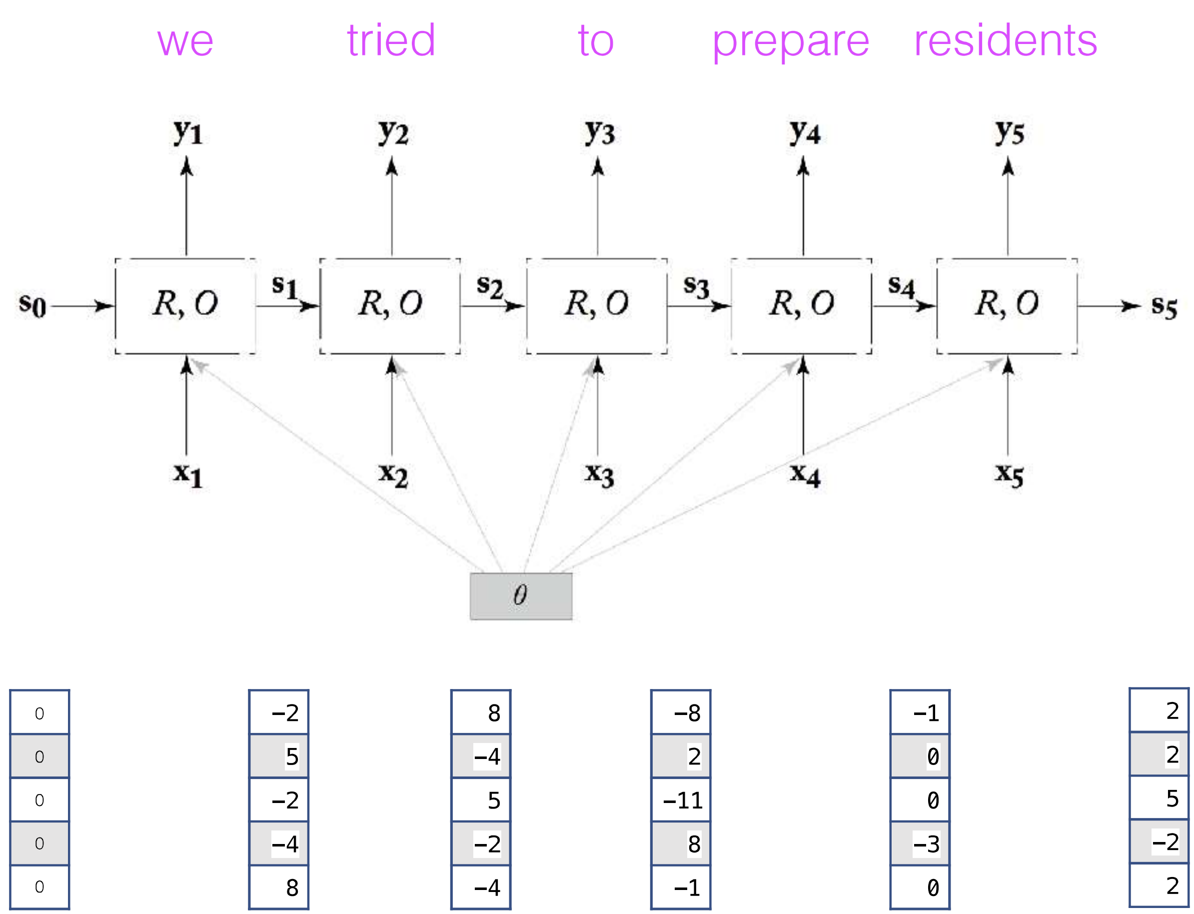
- But we can also condition on any arbitrary context (topic, author, date, metadata, dialect, etc.)

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + [x_i; c]W^x + b)$$

Conditioned generation

- What information could you condition on in order to predict the next word?

feature classes
location
addressee



Each state *i* encodes information seen until time *i*
and its structure is optimized to predict the next word

Character LM

- Vocabulary $\mathcal{V}$ is a finite set of discrete characters
- When the output space is small, you're putting a lot of the burden on the structure of the model
- Encode long-range dependencies (suffixes depend on prefixes, word boundaries etc.)

Character LM

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character LM

```
\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$.
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$
is an injective and let $\mathfrak q$ be an abelian sheaf on $X$.
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let $\mathcal{F}$ be an abelian quasi-coherent sheaf on $\mathcal{C}$.
Let $\mathcal{F}$ be a coherent $\mathcal{O}_X$-module. Then
$\mathcal{F}$ is an abelian catenary over $\mathcal{C}$.
\item The following are equivalent
\begin{enumerate}
\item $\mathcal{F}$ is an $\mathcal{O}_X$-module.
\end{lemma}
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character LM

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Drawbacks

- Very expensive to train (especially for large vocabulary, though tricks exists — cf. hierarchical softmax)
- Backpropagation through long histories leads to vanishing gradients (cf. LSTMs in a few weeks).
- But they consistently have some of the strongest performances in perplexity evaluations.

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM [50]	140.2	116.7	106.6	3.8%	3.4%
Feedforward neural network LM [40]	141.8	114.8	105.2	4.2%	3.7%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	**89.4**	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	**92.9**	90.0	8.5%	6.9%

Mikolov 2010

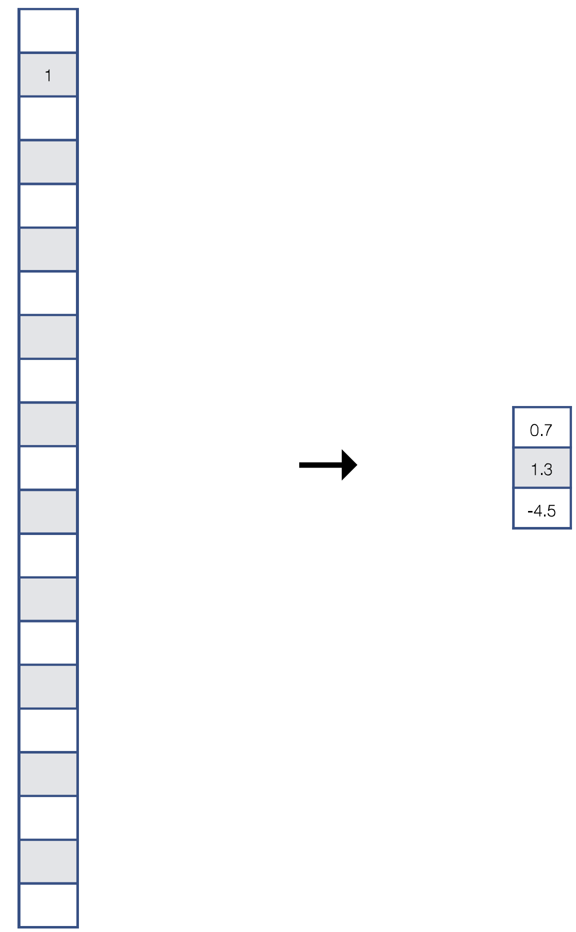
Model	Size	D	Valid	Test
Medium LSTM, Zaremba (2014)	10M	2	86.2	82.7
Large LSTM, Zaremba (2014)	24M	2	82.2	78.4
VD LSTM, Press (2016)	51M	2	75.8	73.2
VD LSTM, Inan (2016)	9M	2	77.1	73.9
VD LSTM, Inan (2016)	28M	2	72.5	69.0
VD RHN, Zilly (2016)	24M	10	67.9	65.4
NAS, Zoph (2016)	25M	-	-	64.0
NAS, Zoph (2016)	54M	-	-	62.4
LSTM	10M	1	61.8	59.6
LSTM		2	63.0	60.8
LSTM		4	62.4	60.1
RHN		5	66.0	63.5
LSTM	24M	1	61.4	59.5
LSTM		2	62.1	59.6
LSTM		4	60.9	58.3
RHN		5	64.8	62.2

Melis, Dyer and Blunsom 2017

Distributed representations

- Some of the greatest power in neural network approaches is in the representation of words (and contexts) as low-dimensional vectors

- We'll talk much more about that on Thursday.



You should feel comfortable:

- Calculate the probability of a sentence given a trained model
- Estimating (e.g., trigram) language model
- Evaluating perplexity on held-out data
- Sampling a sentence from a trained model

Tools

- SRILM
  http://www.speech.sri.com/projects/srilm/

- KenLM
  https://kheafield.com/code/kenlm/

- Berkeley LM
  https://code.google.com/archive/p/berkeleylm/