Checkout this sentence. Is it correct?

The horse raced past the barn fell

CS678 Advanced Natural Language Processing

Language Modeling, Smoothing, and Recurrent Neural Networks

Antonis Anastasopoulos

https://nlp.cs.gmu.edu/course/cs678-fall22

Acknowledgement: Slides are adapted from Graham Neubig's NN4NLP class @CMU

Outline

- Language Models
- Smoothing
- Recurrent Neural Models

Are These Sentences OK?

- Jane went to the store.
- store to Jane went the.
- Jane went store.
- Jane goed to the store.
- The store went to Jane.
- The food truck went to Jane.

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Unigram

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Unigram

$$P(\text{Jane went to the store}) = P(Jane) \times P(went) \times P(to) \times P(the) \times P(store) \times P(\,.\,)\,.$$

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Unigram

$$P(\text{Jane went to the store}) = P(Jane) \times P(went) \times P(to) \times P(the) \times P(store) \times P(\,.\,)\,.$$

But word order and context matters!

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1})$$

Next Word

Context

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1})$$

Next Word — Context

$$P(\text{Jane went to the store.}) = P(Jane \mid < s >) \times P(went \mid Jane) \times P(to \mid went) \times P(the \mid to) \times P(store \mid the) \times P(. \mid store) \times P(< /s > \mid .)$$

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1})$$

Next Word

Context

The big problem: How do we predict

$$P(x_i \mid x_1, \ldots, x_{i-1})$$

?!?!

Count-based Language Models

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid \underbrace{x_{i-n+1}, \ldots, x_{i-1}}_{\text{preceding } n\text{-}1 \text{ words}}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

"N-gram LM"

(An approximation, i.e., $P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i|x_{i-n+1}, \ldots, x_{i-1})$)

A *bigram* model:

$$P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i|x_{i-1})$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$$p(chased \mid dog) = ? \quad p(cat \mid the) = ? \quad p(the \mid < s >) = ?$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$$p(chased|dog) = \frac{1}{1} = 1 \quad p(mat|the) = \frac{1}{4} = 0.25 \quad p(the|<s>) = 0.5$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

p(A cat chased the mouse .) = ?

Count-based Language Models

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$$p(\text{A cat chased the mouse .}) =$$
$$p(<s>|A) \times$$
$$p(cat|a) \times$$
$$p(chased|cat) \times$$
$$p(the|chased) \times$$
$$p(mouse|the) \times$$
$$p(.|mouse)$$

Count-based Language Models

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$p(\text{A cat chased the mouse .}) =$

$$p(<s>|A) \times$$
$$p(cat|a) \times$$
$$p(chased|cat) \times$$
$$p(the|chased) \times$$
$$p(mouse|the) \times$$
$$p(\,.\,|mouse)$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

- Add smoothing to deal with zero counts:

$$p(x_i \mid x_{i-n+1:i-1}) = \frac{c(x_{i-n+1:i}) + \alpha}{c(x_{i-n+1:i-1}) + \alpha |V|}$$

Count-based Language Models

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$$|V| = |\{the, a, cat, sat, \dots\}| = 15 \qquad \alpha = 1$$

Count-based Language Models

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$|V| = |\{the, a, cat, sat, \dots\}| = 15$ $\alpha = 1$

$p(\text{A cat chased the mouse .}) =$

$p(<s> \,\|\, A) \times$ ✓
$p(cat \,|\, a) \times$ ✓
$p(chased \,|\, cat) \times$ ✓
$p(the \,|\, chased) \times$ ✓
$p(mouse \,|\, the) \times$ ✓
$p(\,. \,|\, mouse)$ ✓

When the context is longer:
$p(\text{mouse} \,|\, \text{a cat chased the}) = ?$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

- Add smoothing to deal with zero counts:

$$P(x_i | x_{i-n+1:i-1}) = \frac{c(x_{i-n+1:i}) + \alpha}{c(x_{i-n+1:i-1}) + \alpha |V|}$$

- Another way to smooth: skip some words

$$\begin{aligned} P(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) =& \lambda P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) \\ &+ (1-\lambda) P(x_i \mid x_{1-n+2}, \ldots, x_{i-1}) \end{aligned}$$

Evaluation

- **Log-likelihood:**

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

Evaluation

- **Log-likelihood:**

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word Log Likelihood:**

$$WLL(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

Evaluation

- **Log-likelihood:**

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word Log Likelihood:**

$$WLL(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word (Cross) Entropy:**

$$H(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} -\log_2 P(E)$$

Evaluation

- **Log-likelihood:**

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word Log Likelihood:**

$$WLL(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word (Cross) Entropy:**

$$H(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} -\log_2 P(E)$$

- **Perplexity:**

$$ppl(\mathcal{E}_{test}) = 2^{H(\mathcal{E}_{test})} = e^{-WLL(\mathcal{E}_{test})}$$

Evaluation

What does “My LM achieves a perplexity of 23” mean?

https://sjmielke.com/comparing-perplexities.htm

What Can we Do w/ LMs?

- Score sentences:

Jane went to the store . → high
store to Jane went the . → low

(same as calculating loss for training)

What Can we Do w/ LMs?

- Score sentences:

Jane went to the store . → high
store to Jane went the . → low

(same as calculating loss for training)

- Generate sentences:

while didn't choose end-of-sentence symbol:
  calculate probability
  sample a new word from the probability distribution

Problems and Solutions?

- Cannot share strength among **similar words**

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ solution: class based language models

Problems and Solutions?

- Cannot share strength among **similar words**

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ solution: class based language models

- Cannot condition on context with **intervening words**

Dr. Jane Smith	Dr. Gertrude Smith

→ solution: skip-gram language models

Problems and Solutions?

- Cannot share strength among **similar words**

she bought a car she bought a bicycle
she purchased a car she purchased a bicycle

→ solution: class based language models

- Cannot condition on context with **intervening words**

Dr. Jane Smith Dr. Gertrude Smith

→ solution: skip-gram language models

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

→ solution: cache, trigger, topic, syntactic models, etc.

An Alternative:
Featurized Log-Linear Models

An Alternative: Featurized Models

- Calculate features of the context
- Based on the features, calculate probabilities
- Optimize feature weights using gradient descent, etc.

Example:

Previous words: “giving a"

Example:

Previous words: “giving a"

a
the
talk
gift
hat
…

Words we’re predicting

Example:

Previous words: “giving a"

a
the
talk
gift
hat
…

$$b=\begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \dots \end{pmatrix}$$

Words we’re predicting

How likely are they?

Example:

Previous words: “giving a"

a
the
talk
gift
hat
…

$$b=\begin{pmatrix}3.0\\2.5\\-0.2\\0.1\\1.2\\\dots\end{pmatrix} \qquad w_{1,a}=\begin{pmatrix}-6.0\\-5.1\\0.2\\0.1\\0.5\\\dots\end{pmatrix}$$

Words we’re predicting

How likely are they?

How likely are they given prev. word is “a”?

Example:

Previous words: “giving a"

a
the
talk
gift
hat
…

$$b=\begin{pmatrix}3.0\\2.5\\-0.2\\0.1\\1.2\\\dots\end{pmatrix} \quad w_{1,a}=\begin{pmatrix}-6.0\\-5.1\\0.2\\0.1\\0.5\\\dots\end{pmatrix} \quad w_{2,giving}=\begin{pmatrix}-0.2\\-0.3\\1.0\\2.0\\-1.2\\\dots\end{pmatrix}$$

Words we’re predicting

How likely are they?

How likely are they given prev. word is “a”?

How likely are they given 2nd prev. word is “giving”?

Example:

Previous words: “giving a"

a
the
talk
gift
hat
…

$b = \begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \dots \end{pmatrix}$ $w_{1,a} = \begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.5 \\ \dots \end{pmatrix}$ $w_{2,giving} = \begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \dots \end{pmatrix}$ $s = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \dots \end{pmatrix}$

Words we’re predicting

How likely are they?

How likely are they given prev. word is “a”?

How likely are they given 2nd prev. word is “giving”?

Total score

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

$$P(x_i \mid x_{i-n+1}^{i-1}) = \frac{e^{s(x_i|x_{i-n+1}^{i-1})}}{\sum_{\tilde{x}_i} e^{s(\tilde{x}_i|x_{i-n+1}^{i-1})}}$$

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

$$P(x_i \mid x_{i-n+1}^{i-1}) = \frac{e^{s(x_i|x_{i-n+1}^{i-1})}}{\sum_{\tilde{x}_i} e^{s(\tilde{x}_i|x_{i-n+1}^{i-1})}}$$

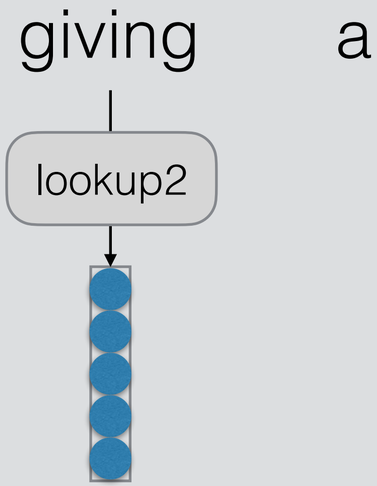
$$\text{s}=\begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \dots \end{pmatrix} \longrightarrow \text{p}=\begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ 0.444 \\ 0.090 \\ \dots \end{pmatrix}$$

A Computation Graph View

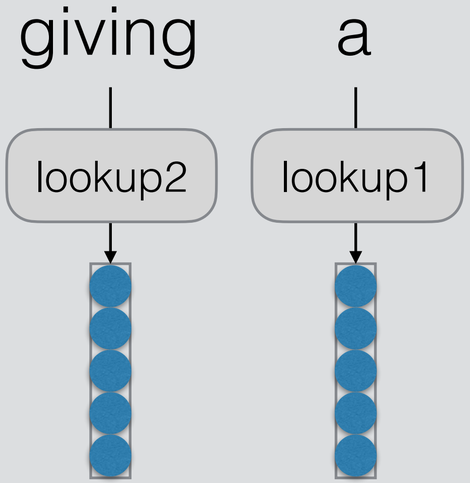
giving a

Each vector is size of output vocabulary

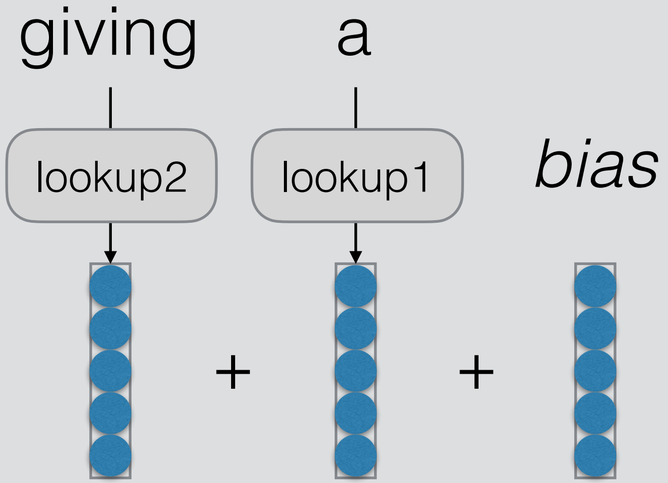
A Computation Graph View



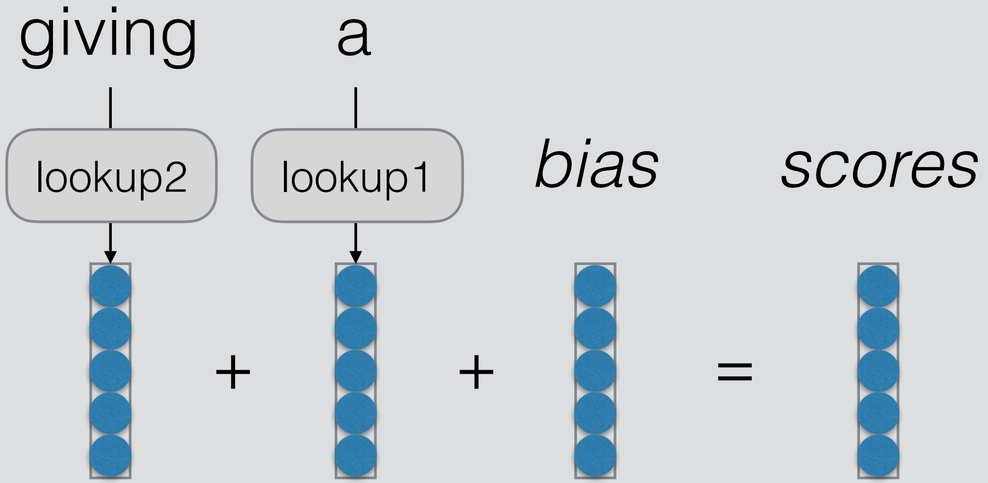
A Computation Graph View



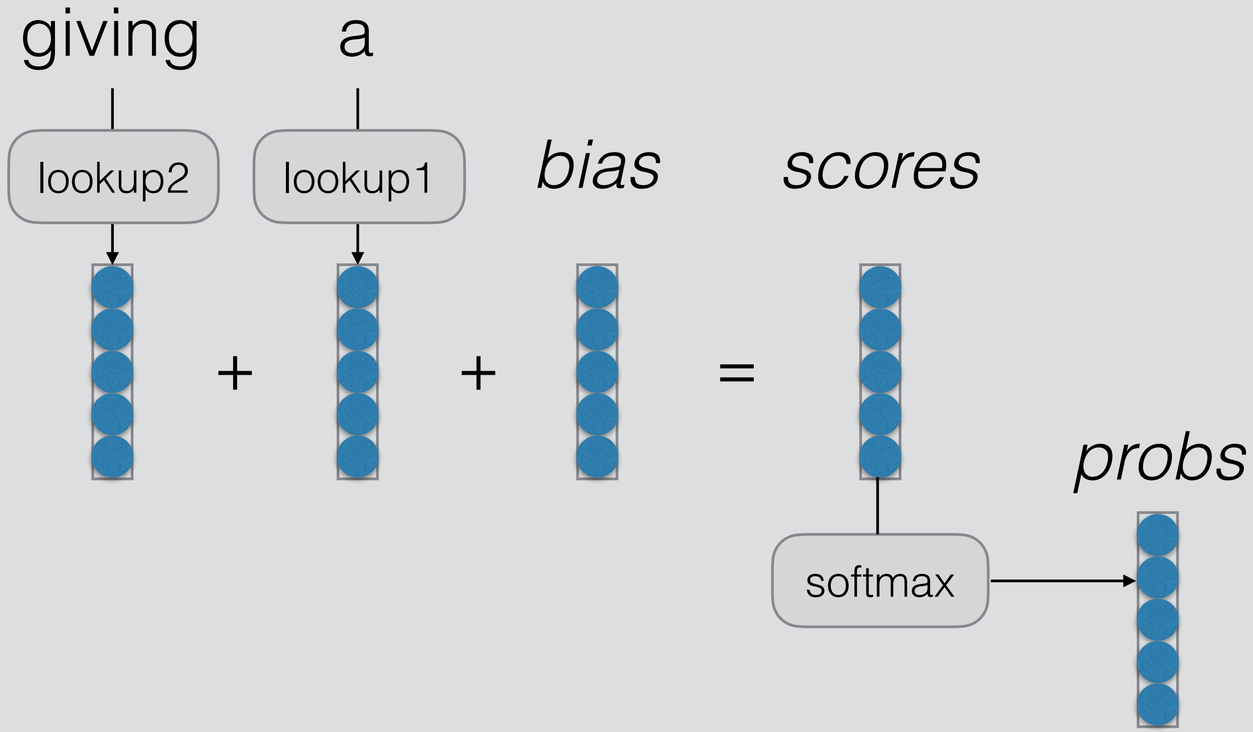
A Computation Graph View



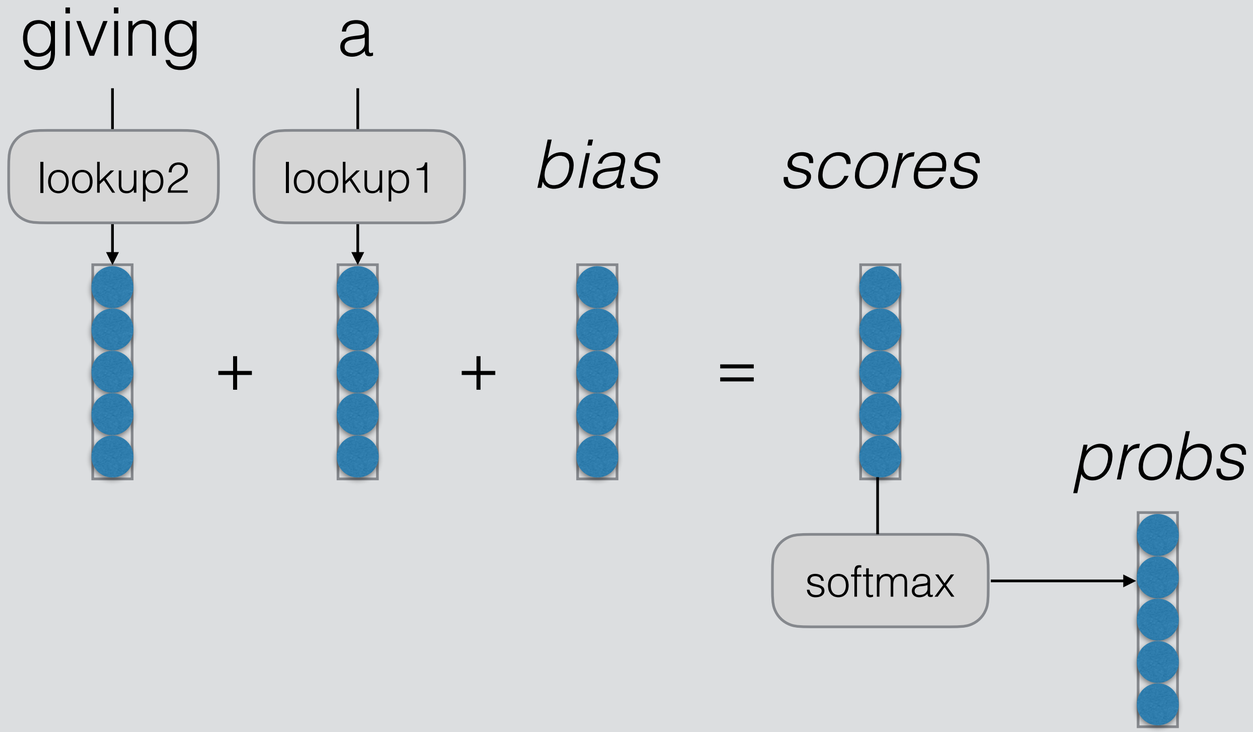
A Computation Graph View



A Computation Graph View



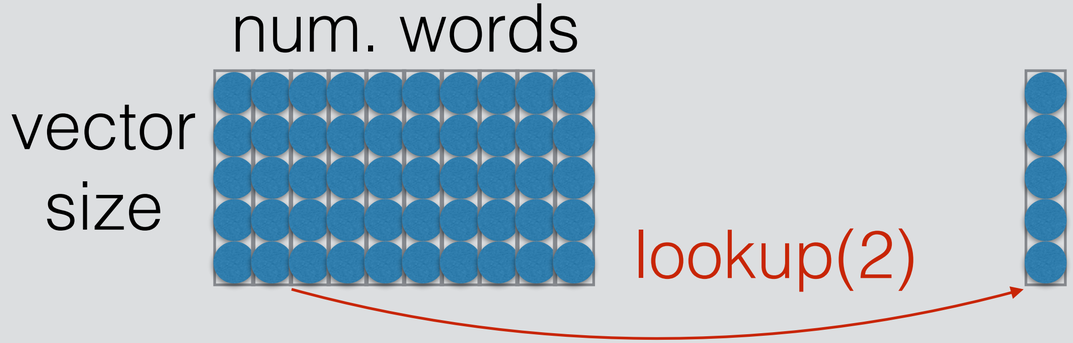
A Computation Graph View



Each vector is size of output vocabulary

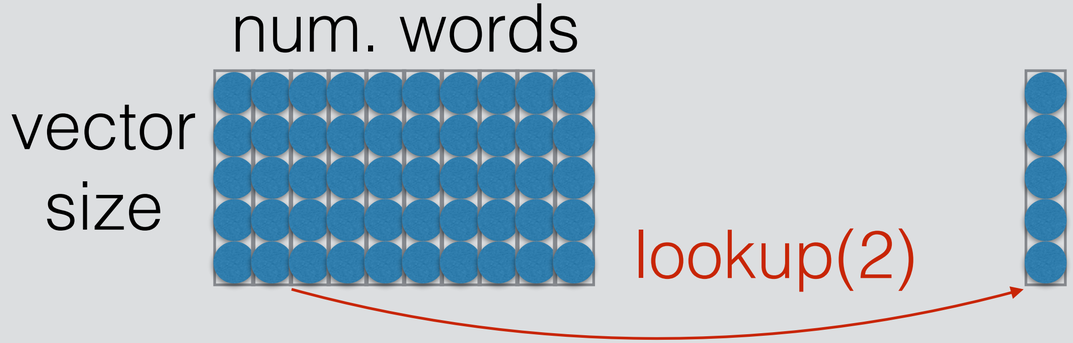
A Note: “Lookup”

- Lookup can be viewed as “grabbing” a single vector from a big matrix of word embeddings

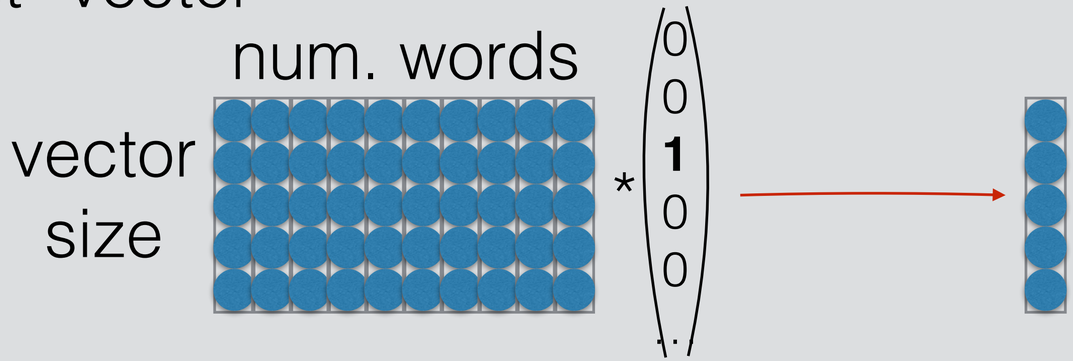


A Note: “Lookup”

- Lookup can be viewed as “grabbing” a single vector from a big matrix of word embeddings

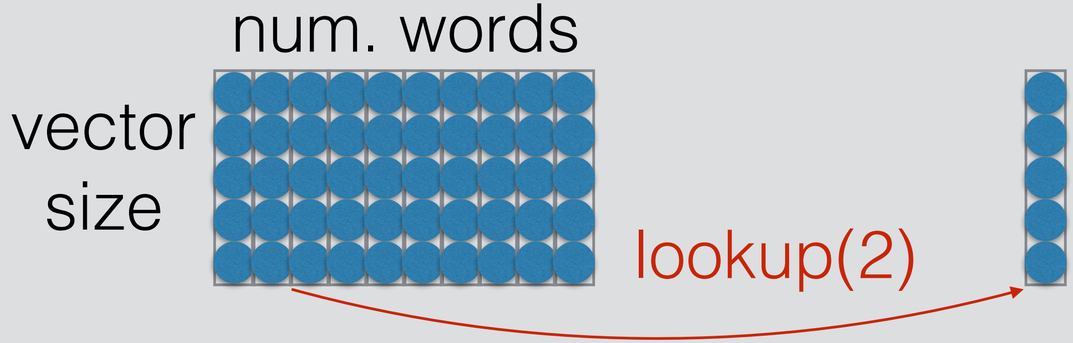


- Similarly, can be viewed as multiplying by a “one-hot” vector

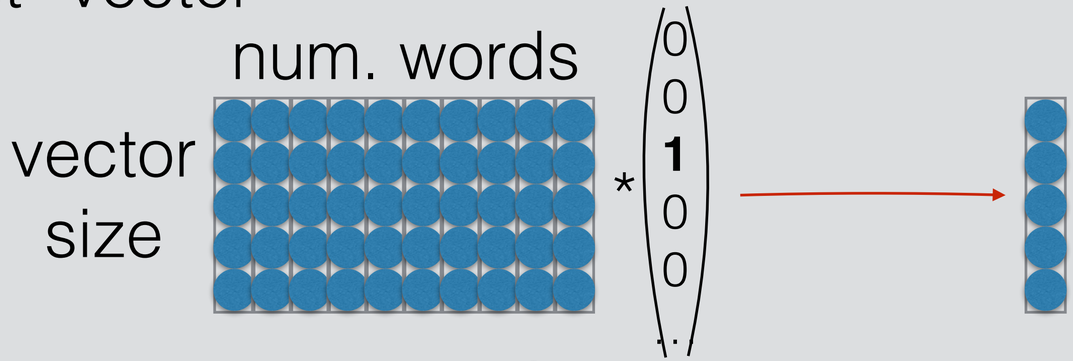


A Note: “Lookup”

- Lookup can be viewed as “grabbing” a single vector from a big matrix of word embeddings



- Similarly, can be viewed as multiplying by a “one-hot” vector



- Former tends to be faster

Training a Model

- **Reminder:** to train, we calculate a “loss function” (a measure of how bad our predictions are), and move the parameters to reduce the loss

- The most common loss function for probabilistic models is “negative log likelihood”

Training a Model

- **Reminder:** to train, we calculate a “loss function” (a measure of how bad our predictions are), and move the parameters to reduce the loss

- The most common loss function for probabilistic models is “negative log likelihood”

If element 3
(or zero-indexed, 2)
is the correct answer:

Parameter Update

- Back propagation allows us to calculate the derivative of the loss with respect to the parameters

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

Parameter Update

- Back propagation allows us to calculate the derivative of the loss with respect to the parameters

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

- Simple stochastic gradient descent optimizes parameters according to the following rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

Choosing a Vocabulary

Unknown Words

Unknown Words

- Necessity for UNK words

Unknown Words

- Necessity for UNK words
  - We won’t have all the words in the world in training data

Unknown Words

- Necessity for UNK words
  - We won’t have all the words in the world in training data
  - Larger vocabularies require more memory and computation time

Unknown Words

- Necessity for UNK words
  - We won’t have all the words in the world in training data
  - Larger vocabularies require more memory and computation time
- Common ways:
  - Frequency threshold (usually UNK <= 1)

Unknown Words

- Necessity for UNK words
  - We won’t have all the words in the world in training data
  - Larger vocabularies require more memory and computation time
- Common ways:
  - Frequency threshold (usually UNK <= 1)
  - Rank threshold

Unknown Words

A very large number of published documents contain text only. They often look boring, and they are often written in obscure language, using mile-long sentences and cryptic technical terms, using one font only, perhaps even without headings. Such style, or lack of style, might be the one you are strongly expected to follow when writing eg scientific or technical reports, legal documents, or administrative papers. It is natural to think that such documents would benefit from a few illustrative images. (However, just adding illustration might be rather useless, if the text remains obscure and unstructured.)

Unknown Words

a very large number of published documents contain text only . they often look boring , and they are often written in obscure language , using mile-long sentences and cryptic technical terms , using one font only , perhaps even without headings . such style, or lack of style, might be the one you are strongly expected to follow when writing eg scientific or technical reports , legal documents , or administrative papers . it is natural to think that such documents would benefit from a few illustrative images . (however , just adding illustration might be rather useless , if the text remains obscure and unstructured .)

truecase + tokenize

Unknown Words

a very large number of published documents contain text only . they often look boring , and they are often written in obscure language , using mile-long sentences and cryptic technical terms , using one font only , perhaps even without headings . such style, or lack of style, might be the one you are strongly expected to follow when writing eg scientific or technical reports , legal documents , or administrative papers . it is natural to think that such documents would benefit from a few illustrative images . (however , just adding illustration might be rather useless , if the text remains obscure and unstructured .)

Find rare words (e.g. with freq<2)

Unknown Words

a very large number of published documents contain text only . they often look boring , and they are often written in obscure language , using UNK sentences and cryptic technical terms , using one font only , perhaps even without headings . such style, or lack of style, might be the one you are strongly expected to follow when writing eg scientific or technical reports , legal documents , or UNK papers . it is natural to think that such documents would benefit from a few illustrative images . (however , just adding UNK might be rather useless , if the text remains obscure and UNK .)

Substitute with UNK

Evaluation and Vocabulary

- **Important:** the vocabulary must be the same over models you compare

Evaluation and Vocabulary

- **Important:** the vocabulary must be the same over models you compare

- Or more accurately, all models must be able to generate the test set (it's OK if they can generate *more* than the test set, but not less)

  - e.g. Comparing a character-based model to a word-based model is fair, but not vice-versa

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car
she purchased a car

she bought a bicycle
she purchased a bicycle

- Cannot condition on context with **intervening words**

Dr. Jane Smith Dr. Gertrude Smith

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ not solved yet 😞

- Cannot condition on context with **intervening words**

Dr. Jane Smith	Dr. Gertrude Smith

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ not solved yet 😞

- Cannot condition on context with **intervening words**

Dr. Jane Smith	Dr. Gertrude Smith

→ solved! 😀

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ not solved yet 😞

- Cannot condition on context with **intervening words**

Dr. Jane Smith	Dr. Gertrude Smith

→ solved! 😀

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

→ not solved yet 😞

Break
Beyond Linear Models

Linear Models can’t Learn Feature Combinations

Linear Models can’t Learn Feature Combinations

farmers eat steak → **high**

Linear Models can’t Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

cows eat hay → **high**

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

cows eat hay → **high**

- These can't be expressed by linear features

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

cows eat hay → **high**

- These can't be expressed by linear features
- What can we do?
  - Remember combinations as features (individual scores for "farmers eat", "cows eat")

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

cows eat hay → **high**

- These can't be expressed by linear features
- What can we do?
  - Remember combinations as features (individual scores for "farmers eat", "cows eat")
    → Feature space explosion!

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

cows eat steak → **low**

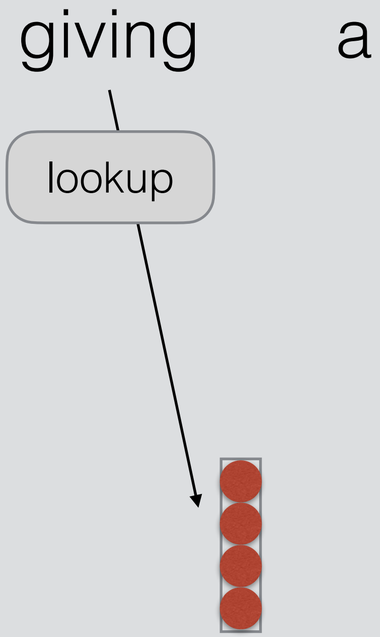
cows eat hay → **high**

- These can't be expressed by linear features
- What can we do?
  - Remember combinations as features (individual scores for "farmers eat", "cows eat")
    → Feature space explosion!
  - Neural nets

Neural Language Models

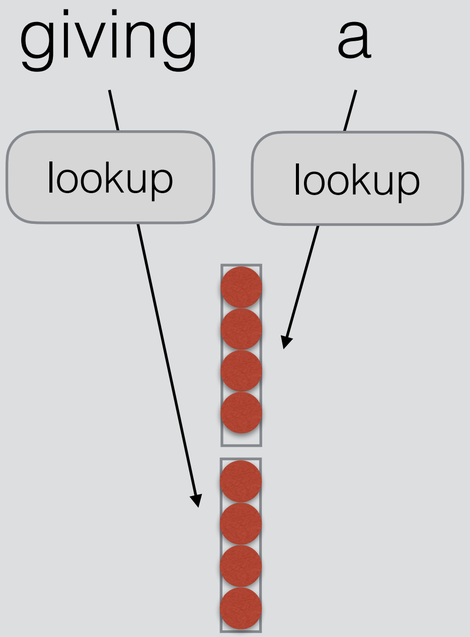
- (See Bengio et al. 2004)

Neural Language Models



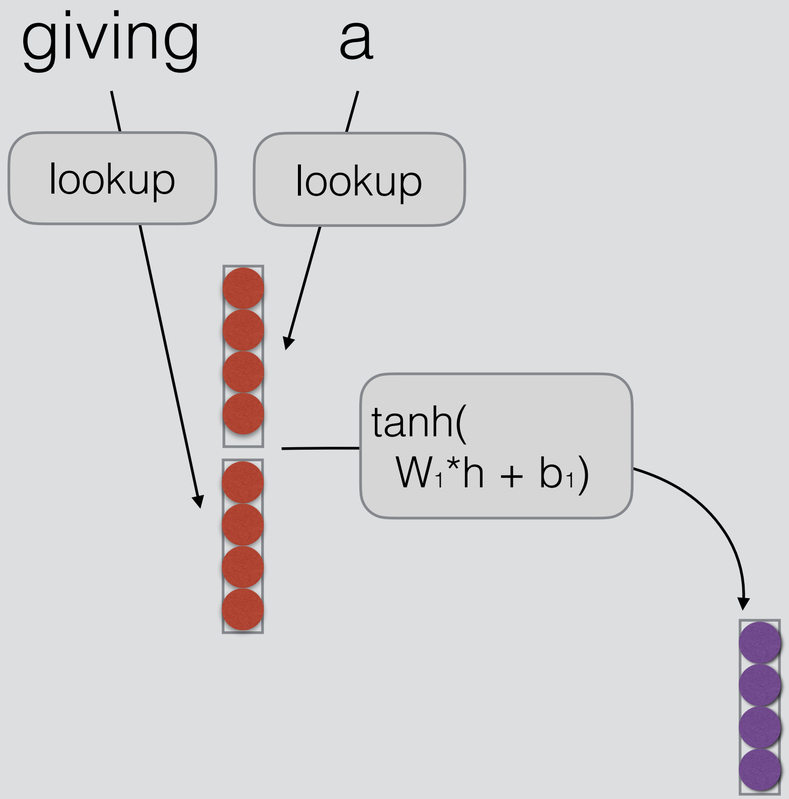
- (See Bengio et al. 2004)

Neural Language Models



- (See Bengio et al. 2004)

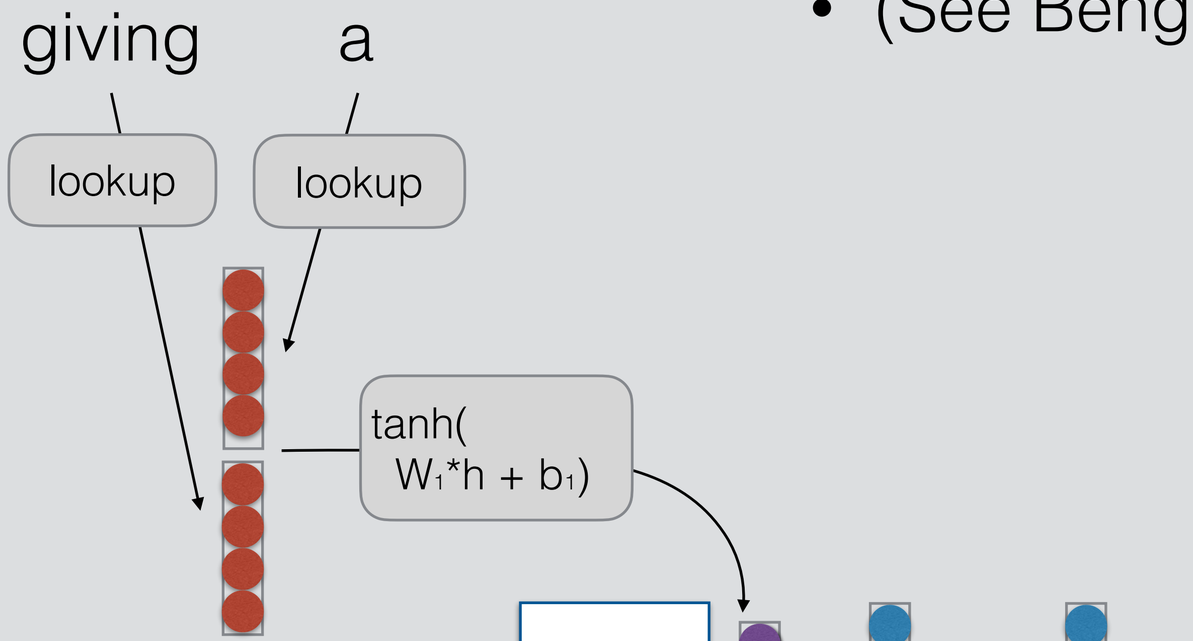
Neural Language Models



- (See Bengio et al. 2004)

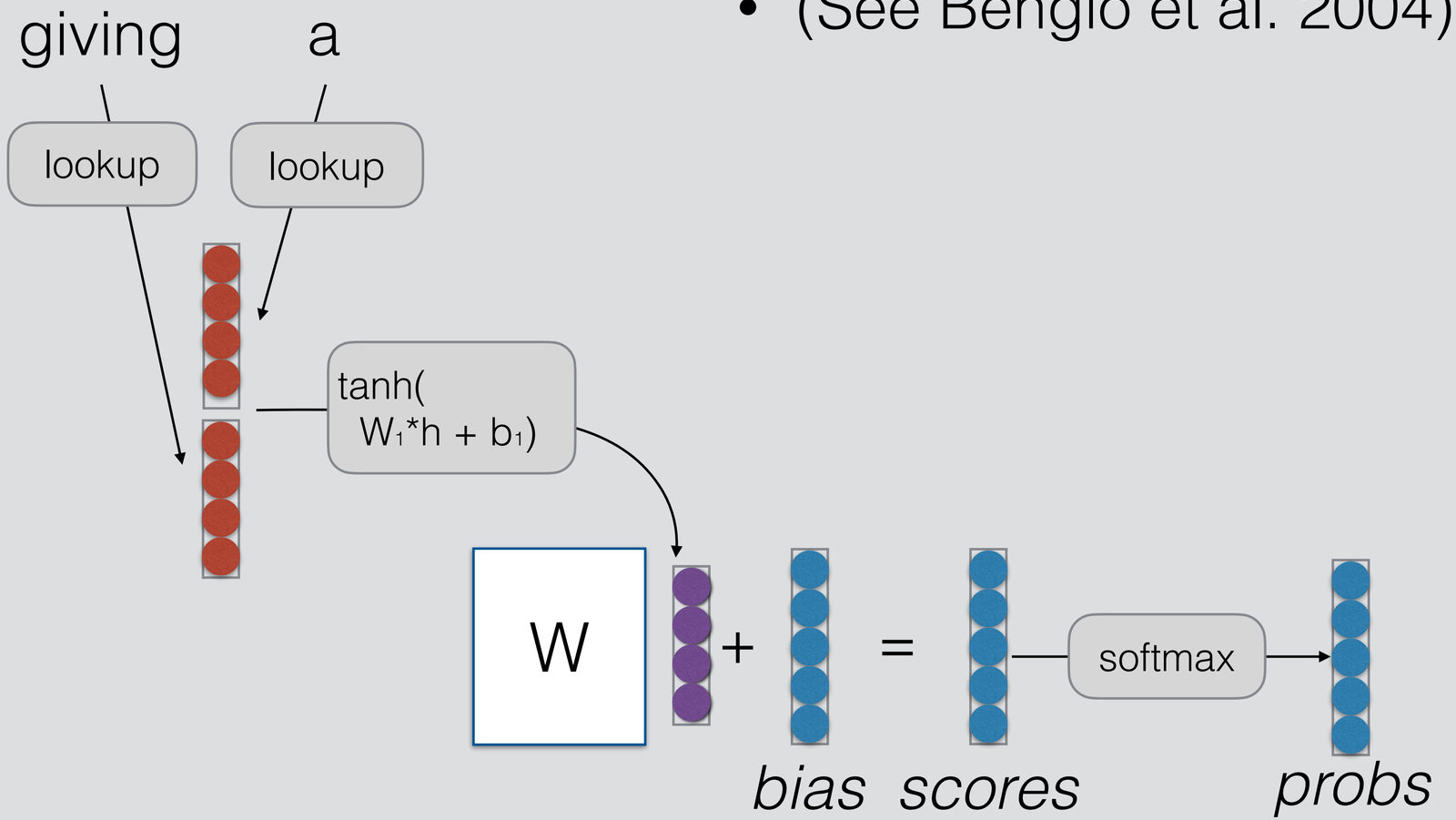
Neural Language Models

- (See Bengio et al. 2004)

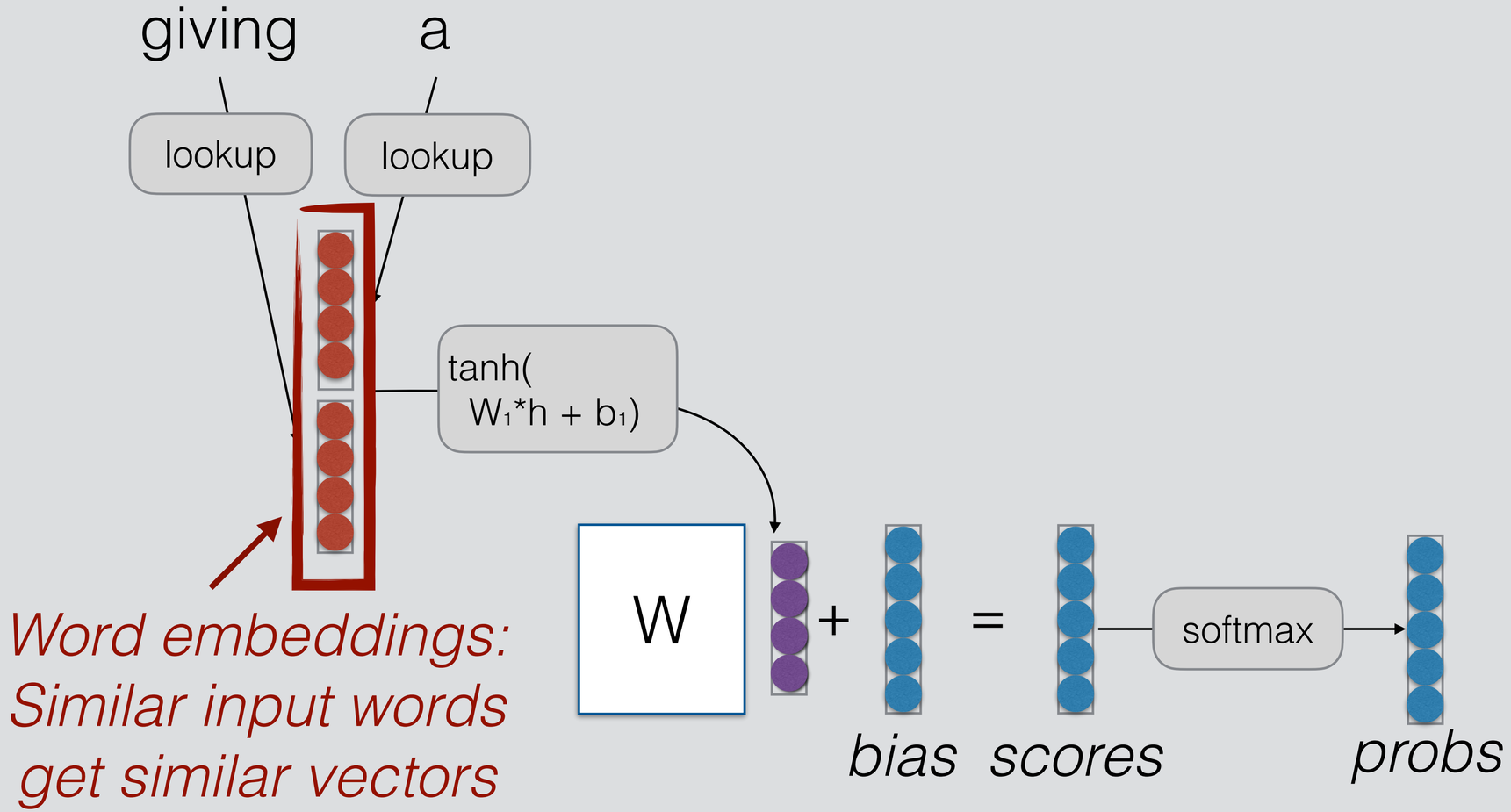


Neural Language Models

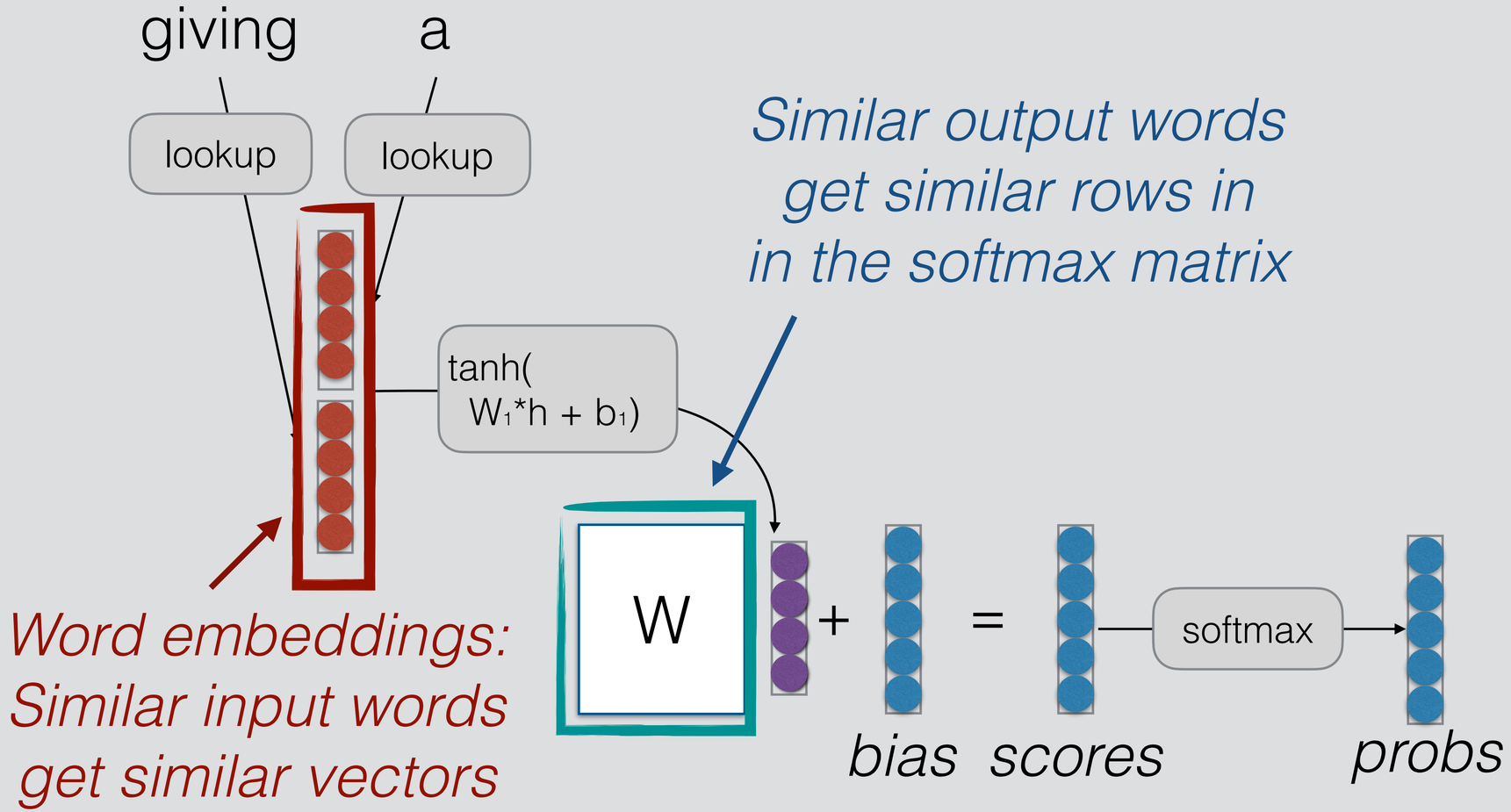
- (See Bengio et al. 2004)



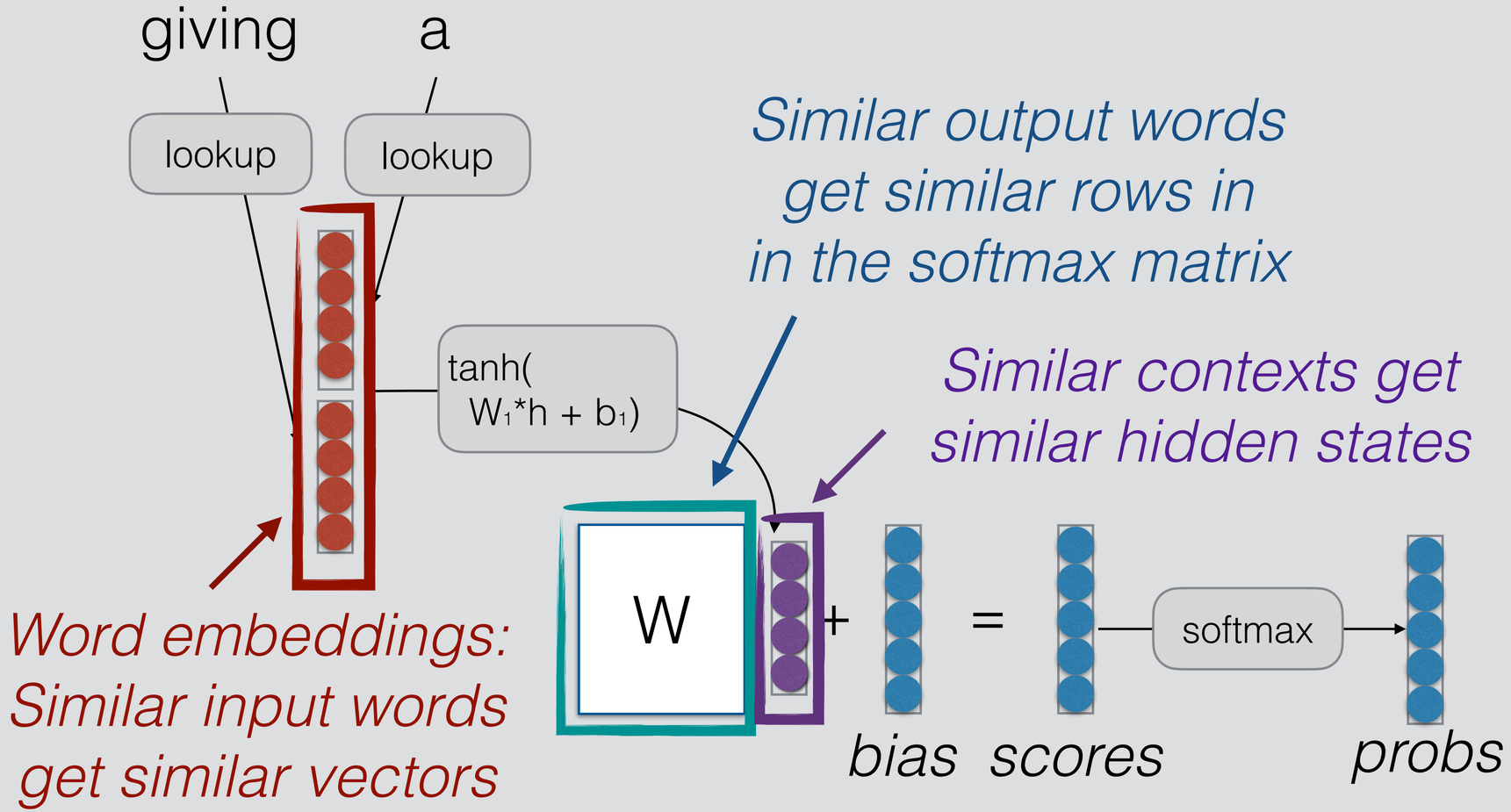
Where is Strength Shared?



Where is Strength Shared?



Where is Strength Shared?



What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car
she purchased a car

she bought a bicycle
she purchased a bicycle

→ solved, and similar contexts as well! 😀

- Cannot condition on context with **intervening words**

Dr. Jane Smith Dr. Gertrude Smith

→ solved! 😀

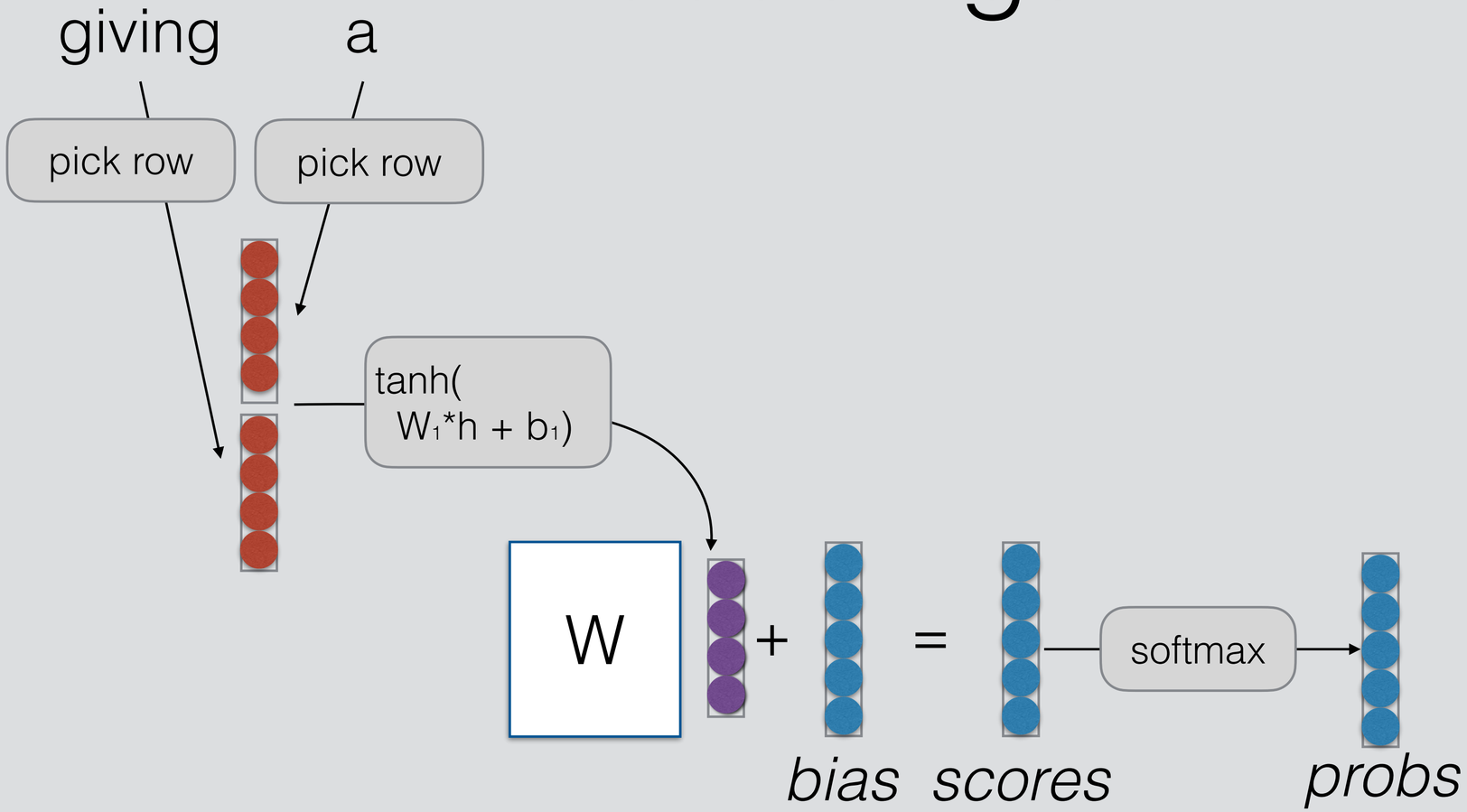
- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

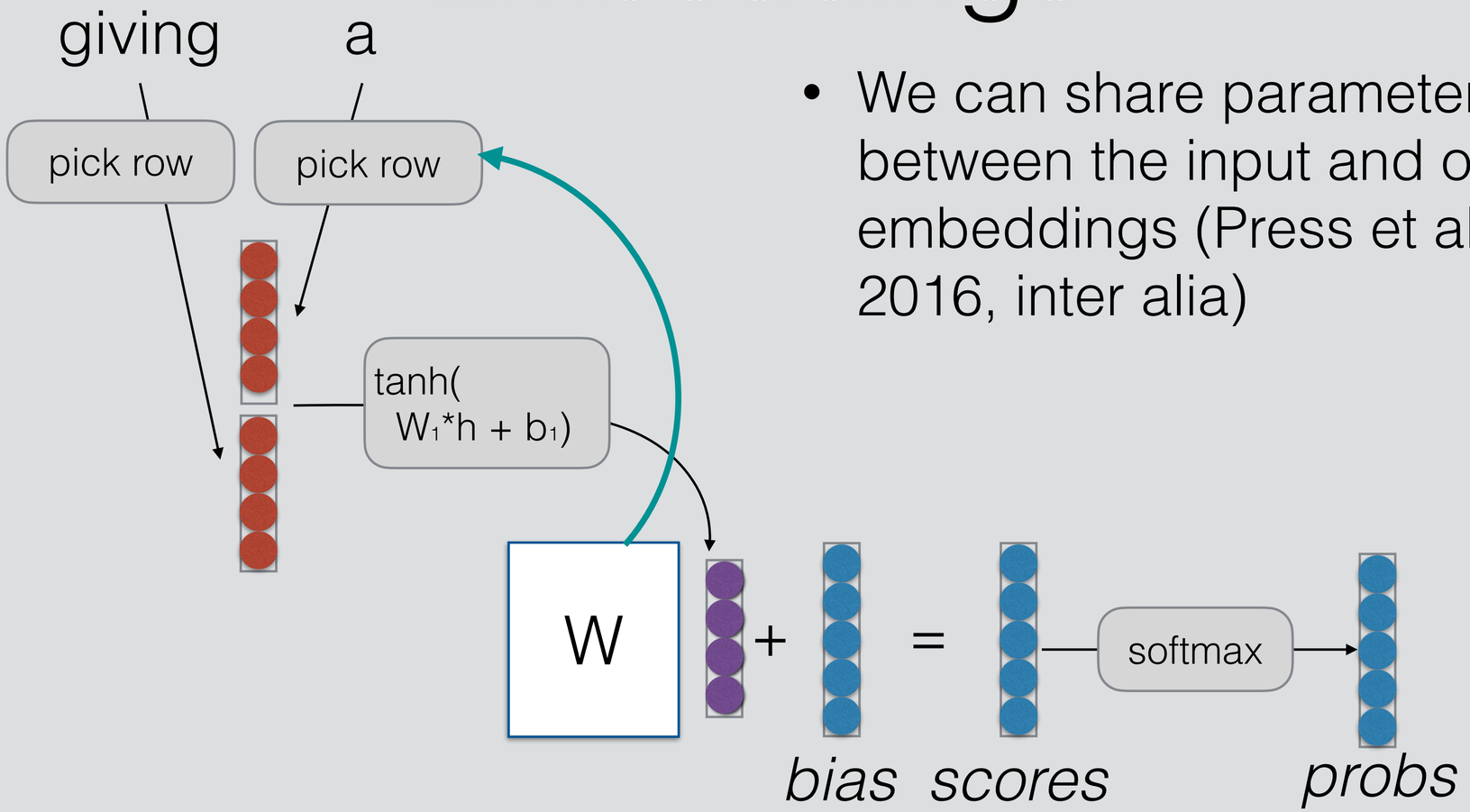
for programming class he wanted to buy his own computer

→ not solved yet 😞

Tying Input/Output Embeddings

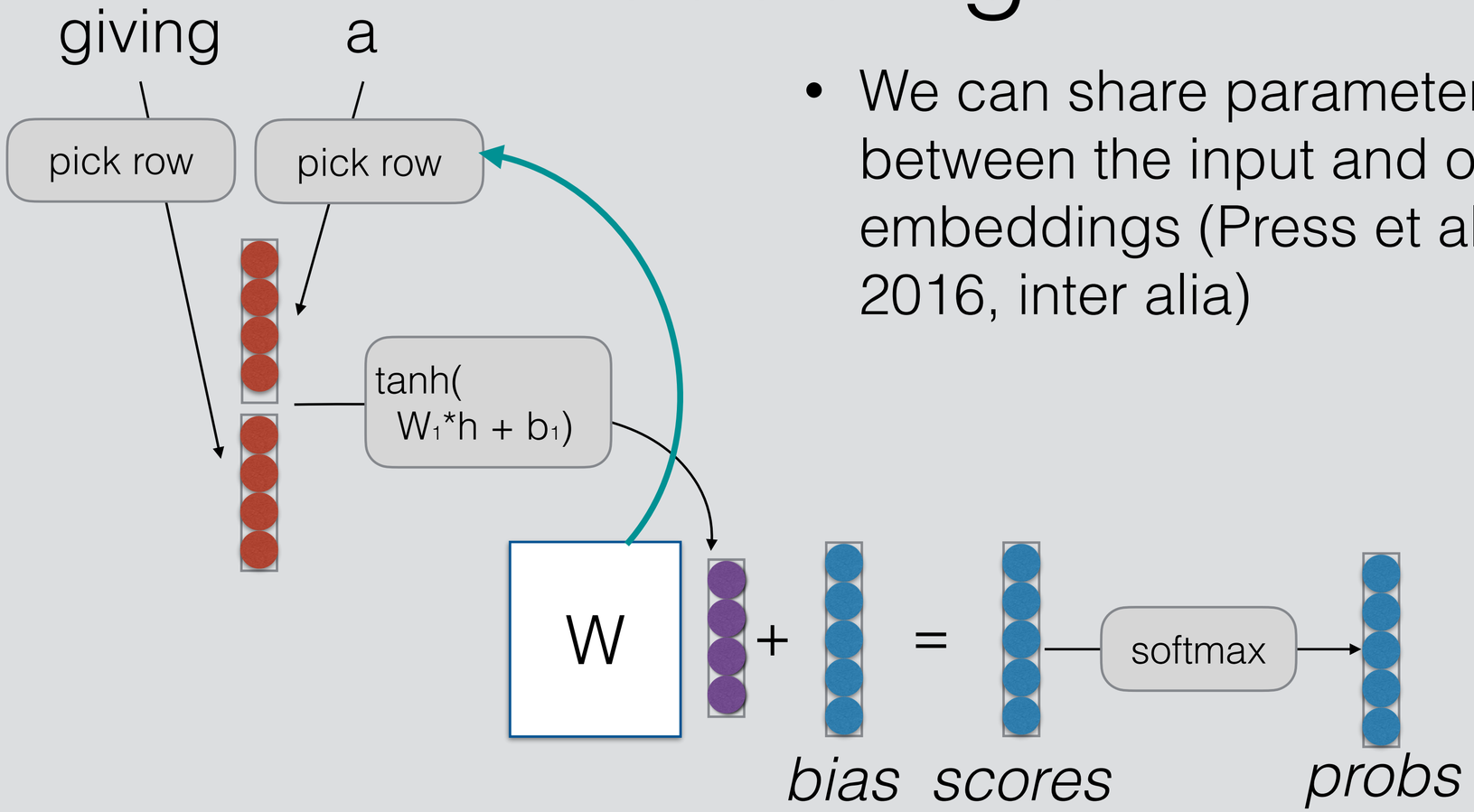


Tying Input/Output Embeddings



- We can share parameters between the input and output embeddings (Press et al. 2016, inter alia)

Tying Input/Output Embeddings



- We can share parameters between the input and output embeddings (Press et al. 2016, inter alia)

Want to try? Delete the input embeddings, and instead pick a row from the softmax matrix.

Next: Recurrent Neural Networks

NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters in words
  - Sentences in discourse
  - …

Long-distance Dependencies in Language

- Agreement in number, gender, etc.

He does not have very much confidence in **himself**.
She does not have very much confidence in **herself**.

Long-distance Dependencies in Language

- Agreement in number, gender, etc.

He does not have very much confidence in **himself**.
She does not have very much confidence in **herself**.

- Selectional preference

The **reign** has lasted as long as the life of the **queen**.
The **rain** has lasted as long as the life of the **clouds**.

Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

The trophy would not fit in the brown suitcase because it was too **small**.

Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

The trophy would not fit in the brown suitcase because it was too **small**.

Suitcase

(from Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)

Recurrent Neural Networks

(Elman 1990)

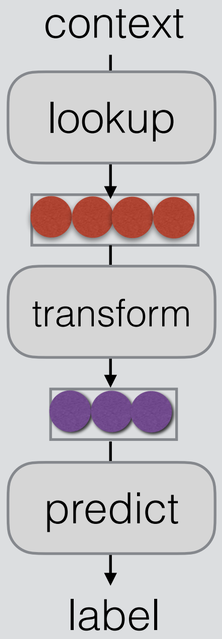
- Tools to “remember” information

Recurrent Neural Networks

(Elman 1990)

- Tools to “remember” information

Feed-forward NN

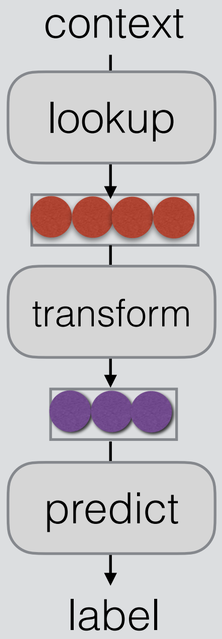


Recurrent Neural Networks

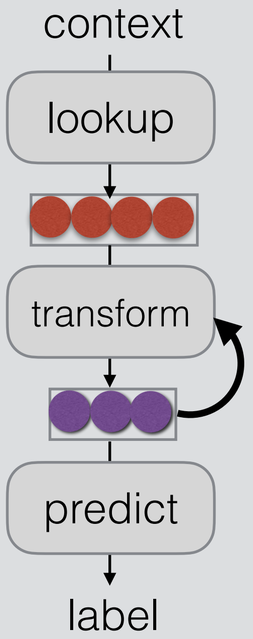
(Elman 1990)

- Tools to “remember” information

Feed-forward NN



Recurrent NN

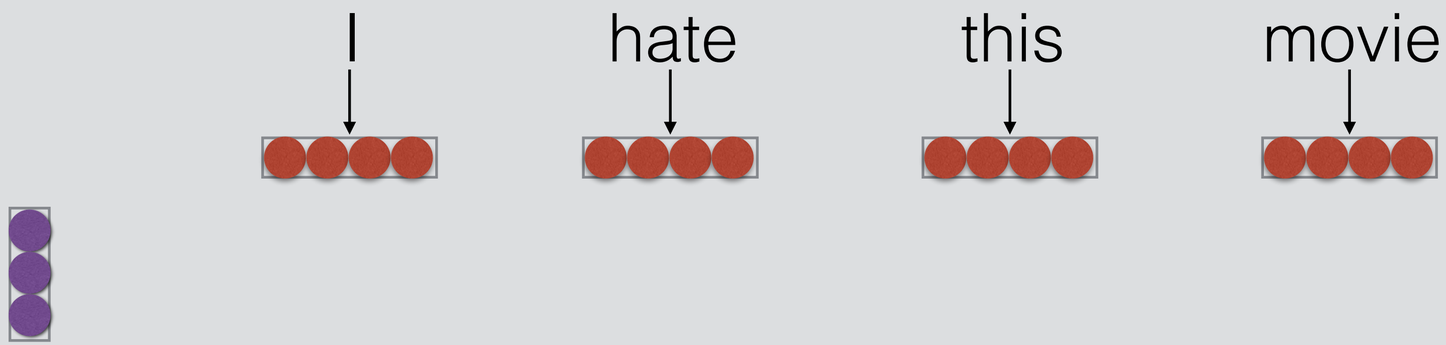


Unrolling in Time

- What does processing a sequence look like?

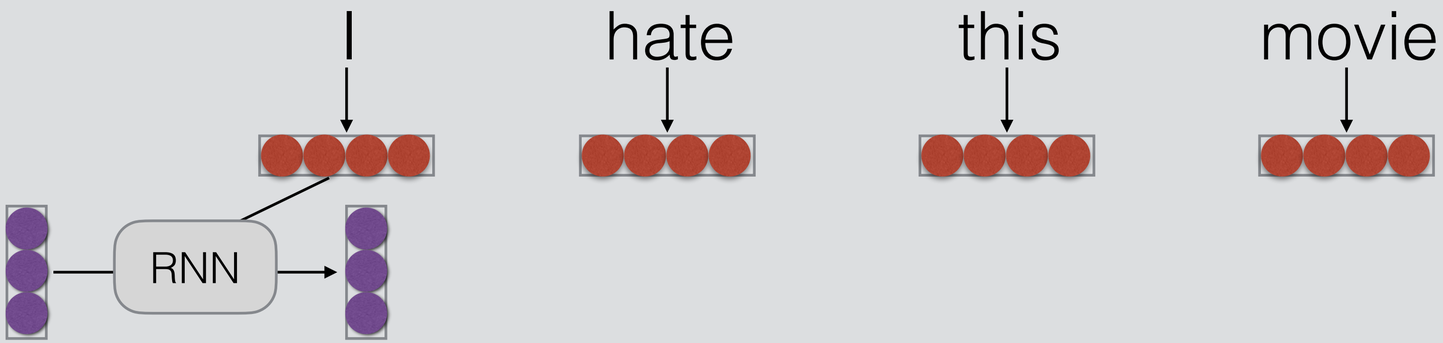
Unrolling in Time

- What does processing a sequence look like?



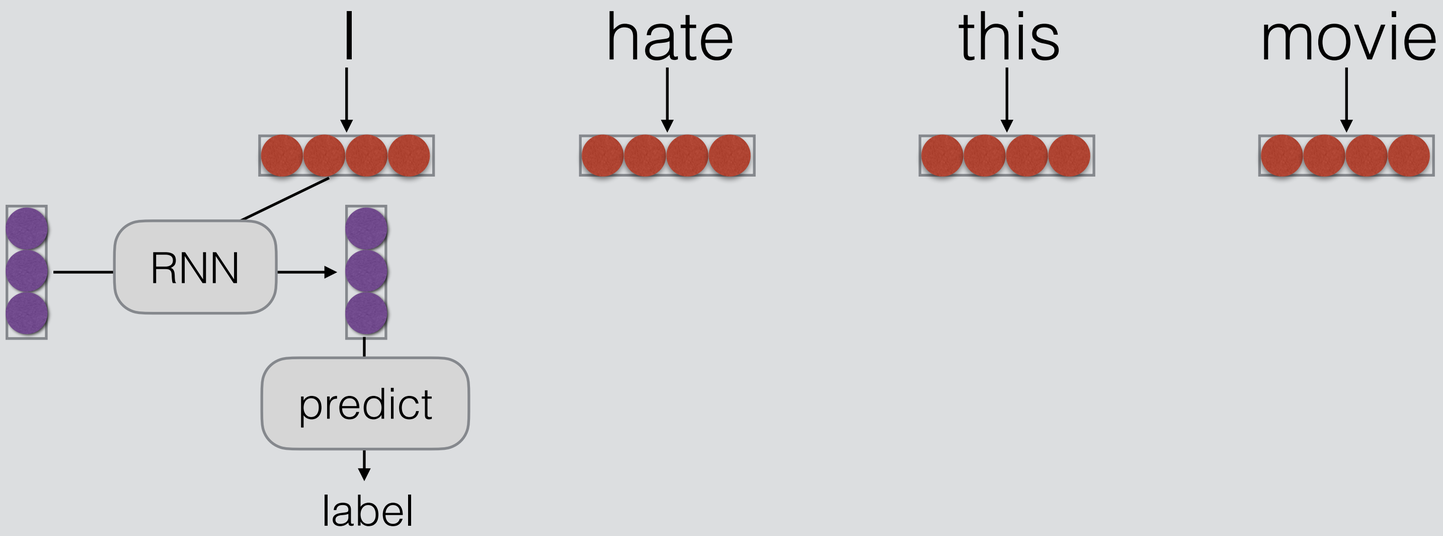
Unrolling in Time

- What does processing a sequence look like?



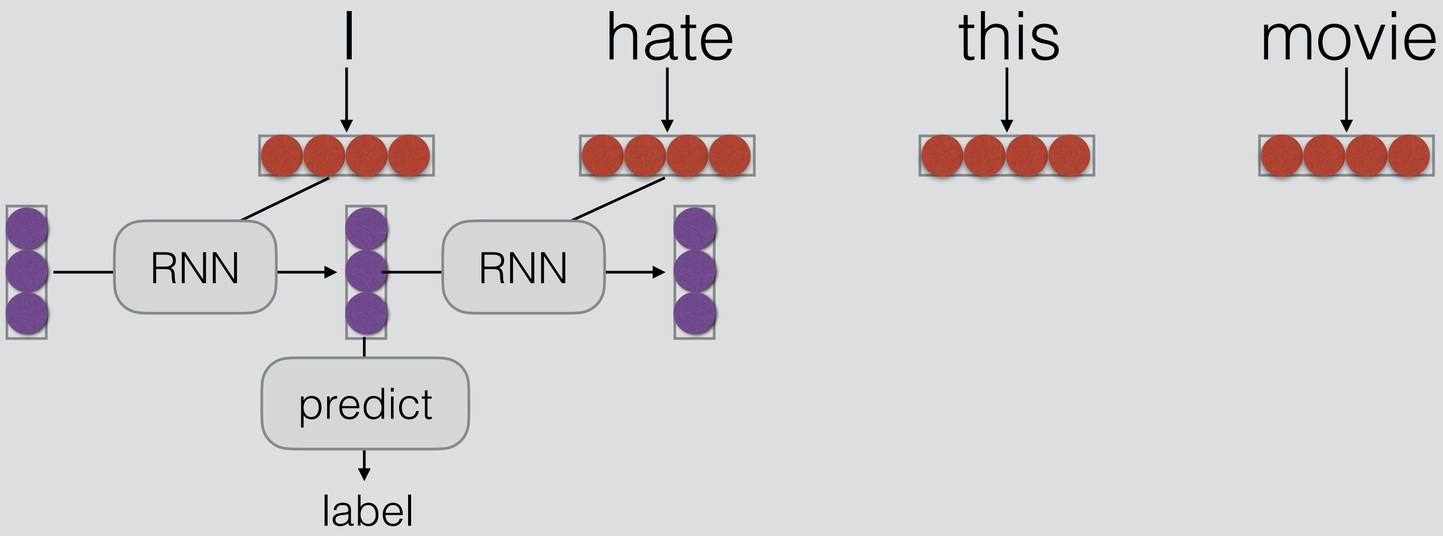
Unrolling in Time

- What does processing a sequence look like?



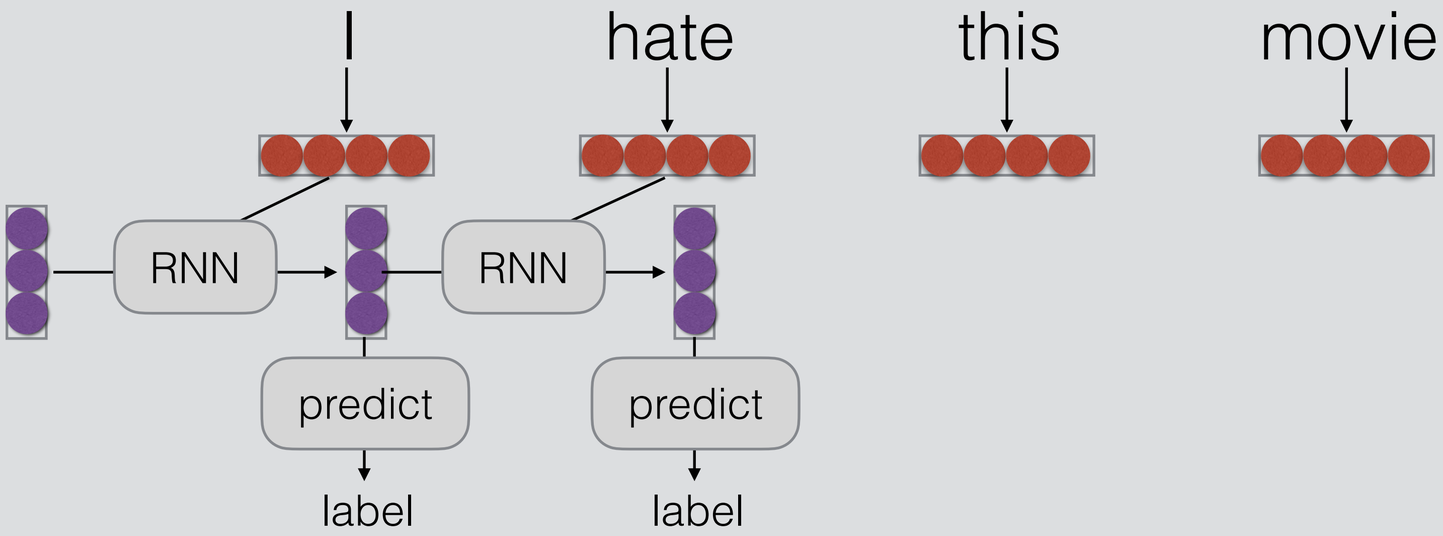
Unrolling in Time

- What does processing a sequence look like?



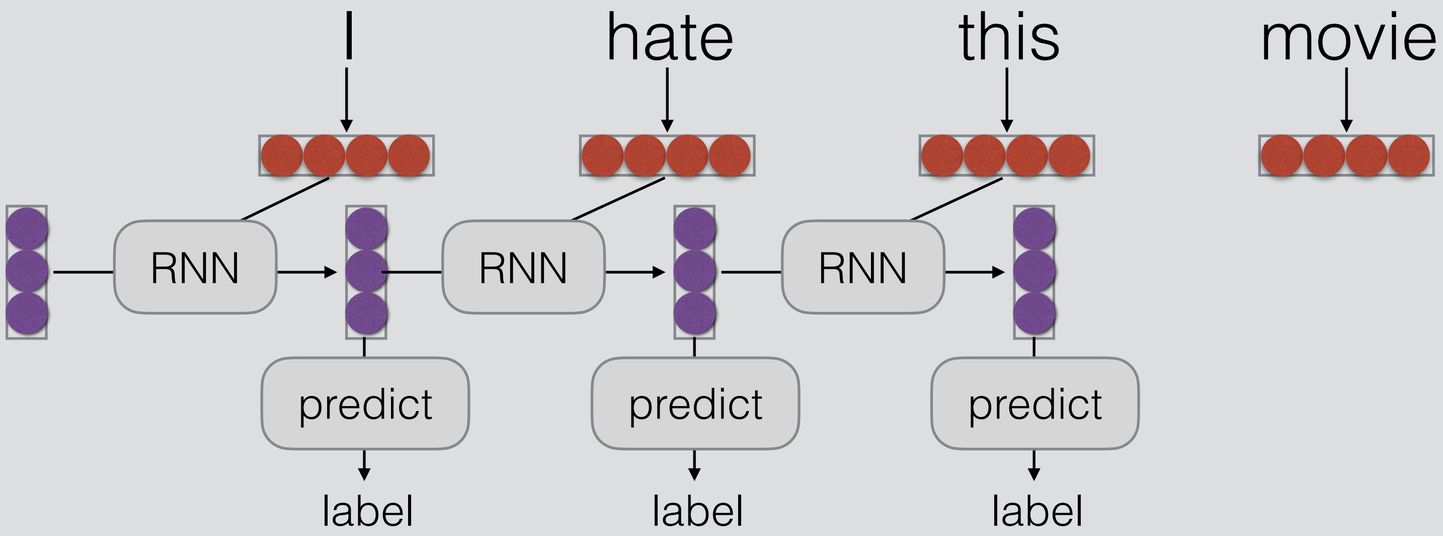
Unrolling in Time

- What does processing a sequence look like?



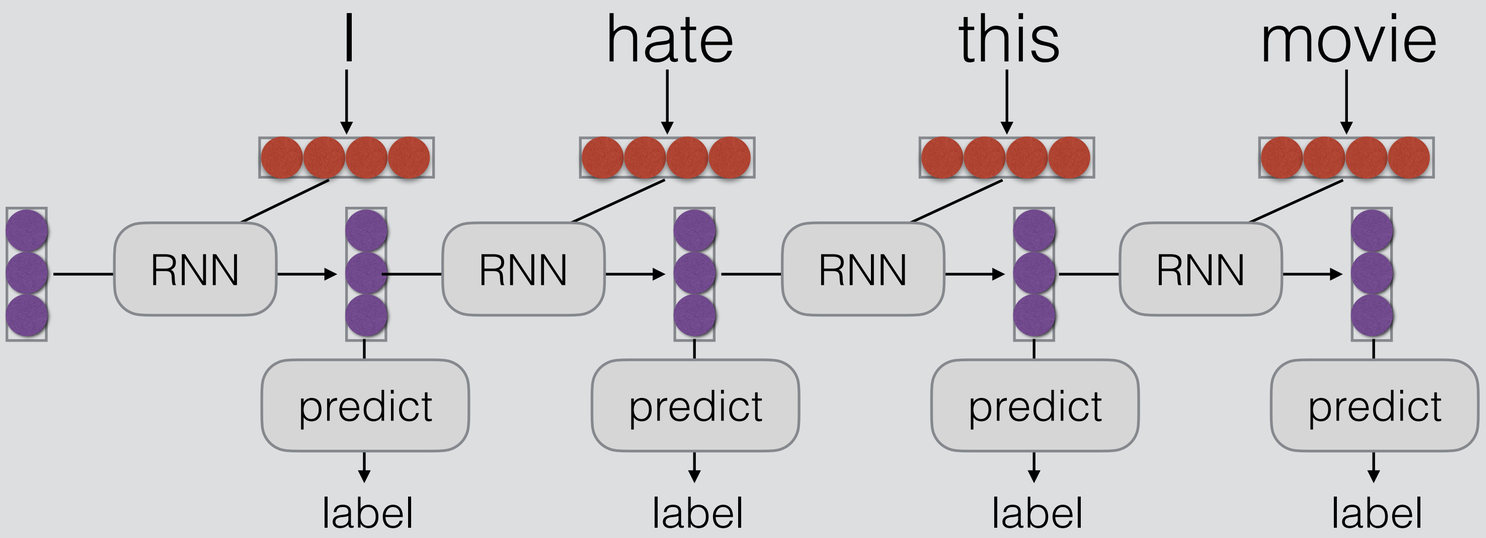
Unrolling in Time

- What does processing a sequence look like?



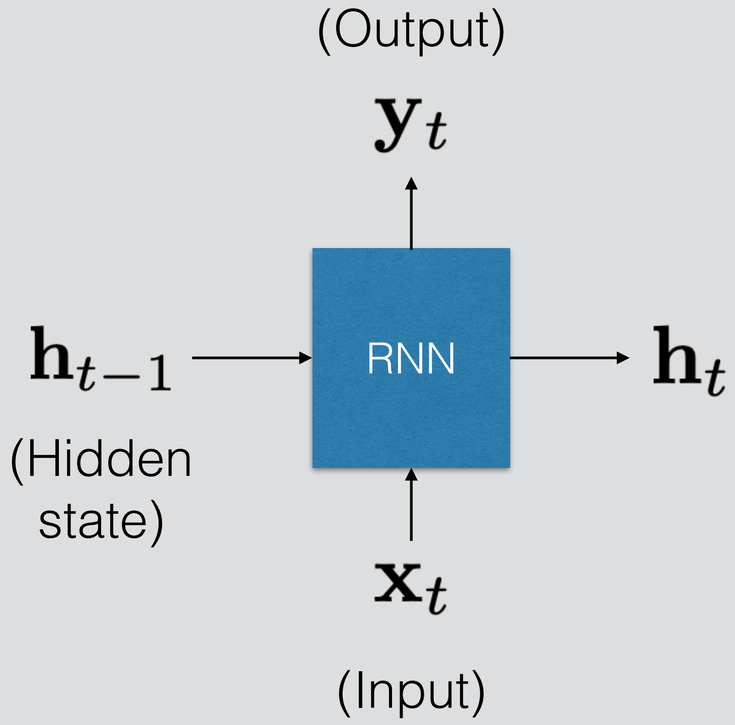
Unrolling in Time

- What does processing a sequence look like?



Recurrent Neural Networks

(Elman 1990)



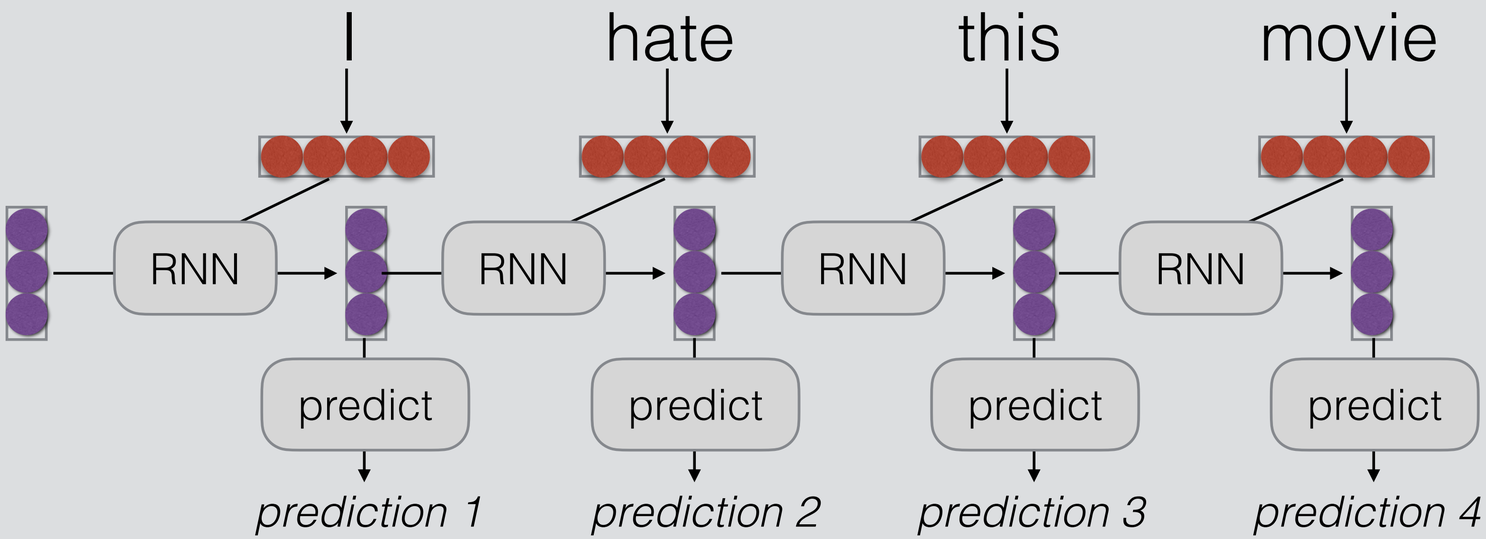
- Update hidden state

$$\mathbf{h}_t = \tanh(W\mathbf{x}_t + V\mathbf{h}_{t-1} + \mathbf{b}_h)$$

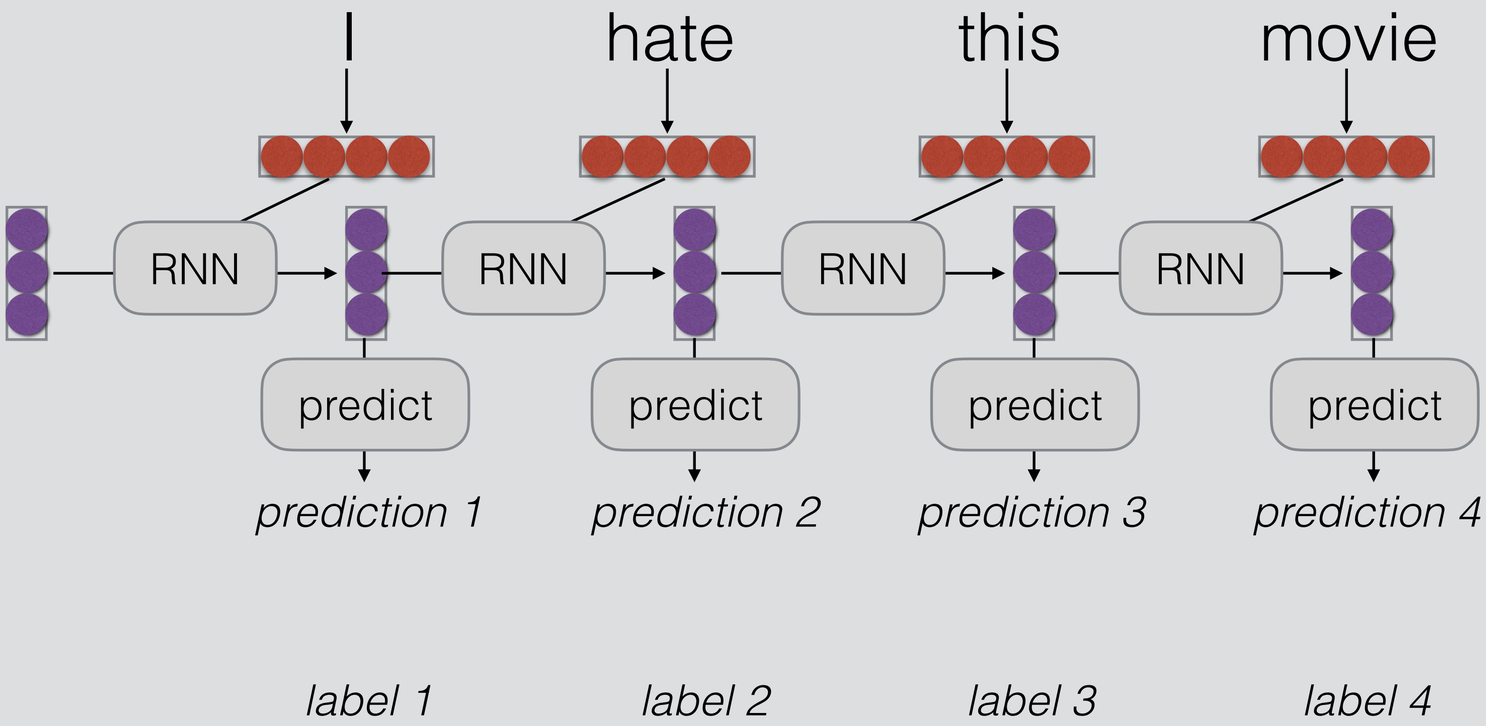
- Compute output

$$\mathbf{y}_t = \tanh(U\mathbf{h_t} + \mathbf{b}_y)$$

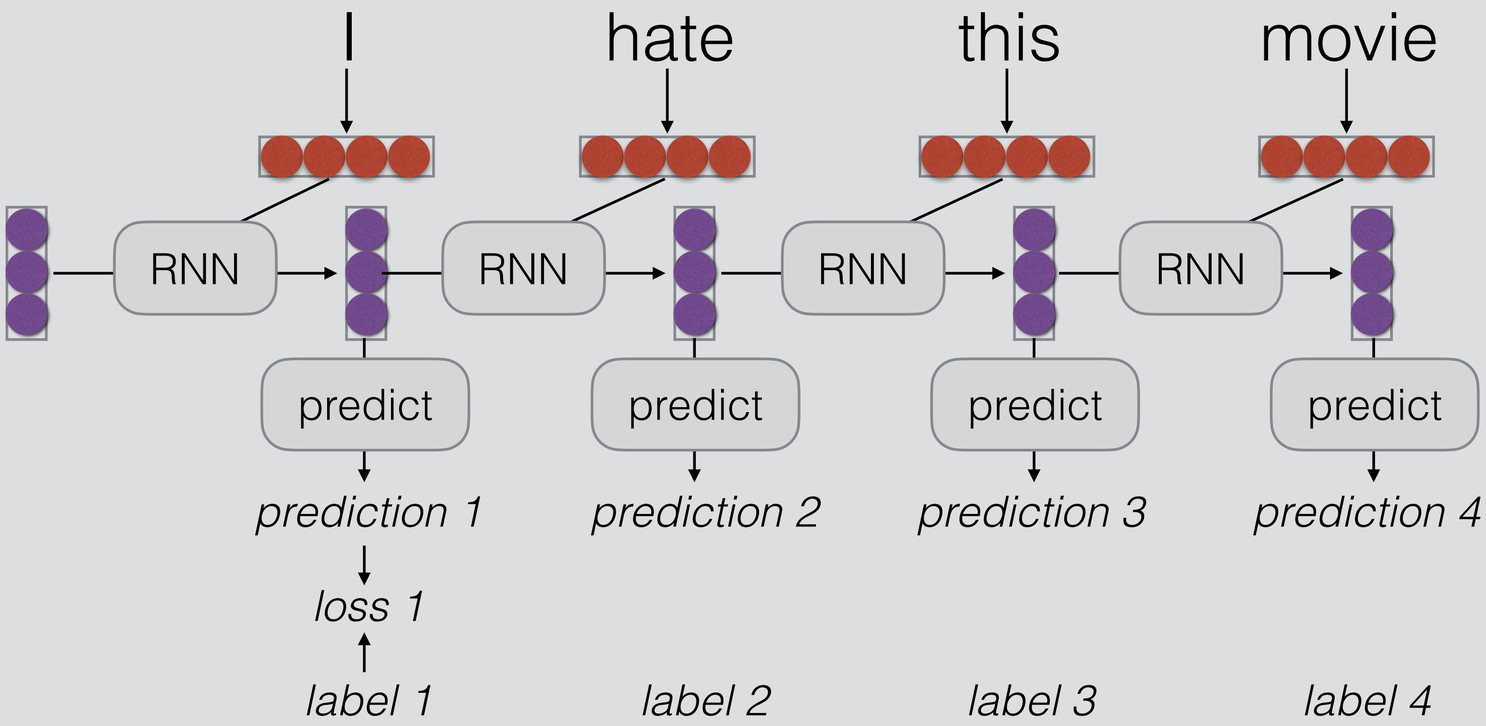
Training RNNs



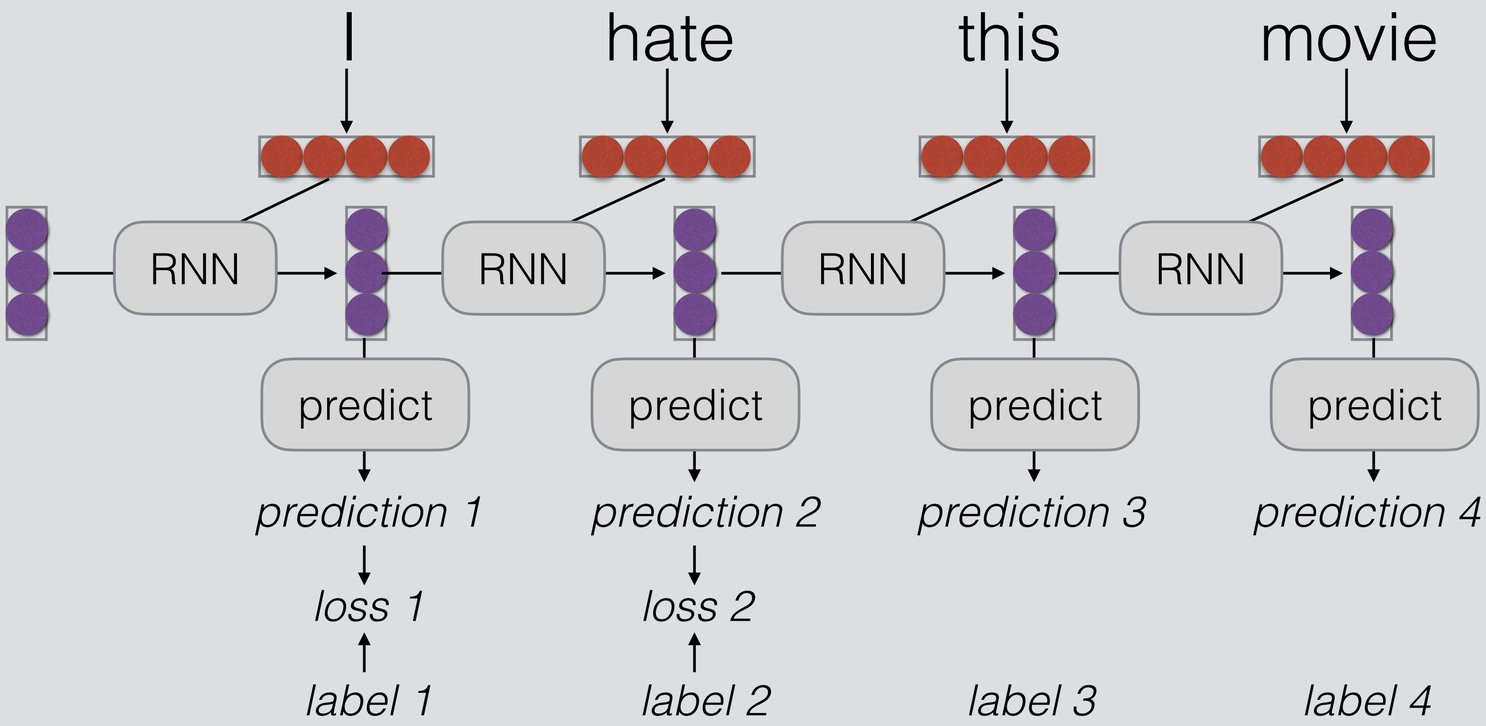
Training RNNs



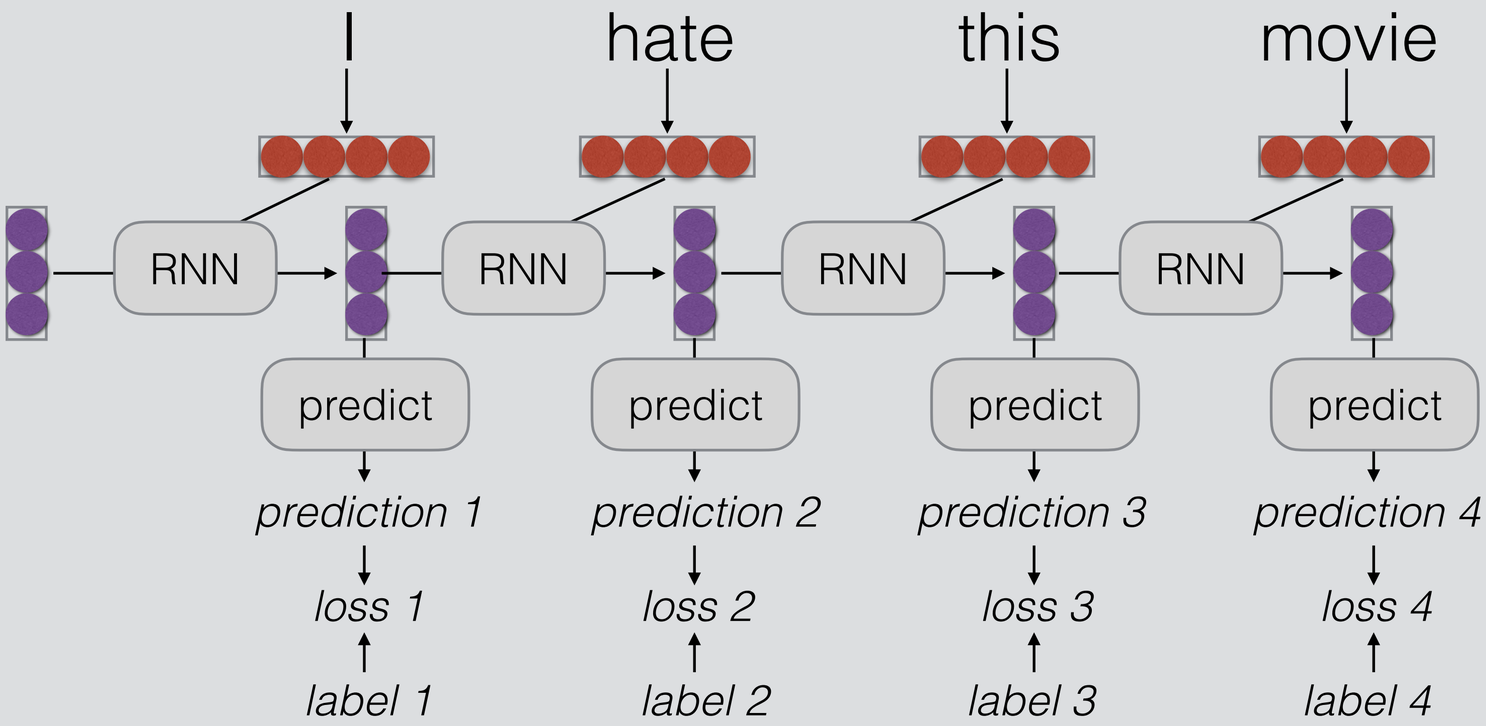
Training RNNs



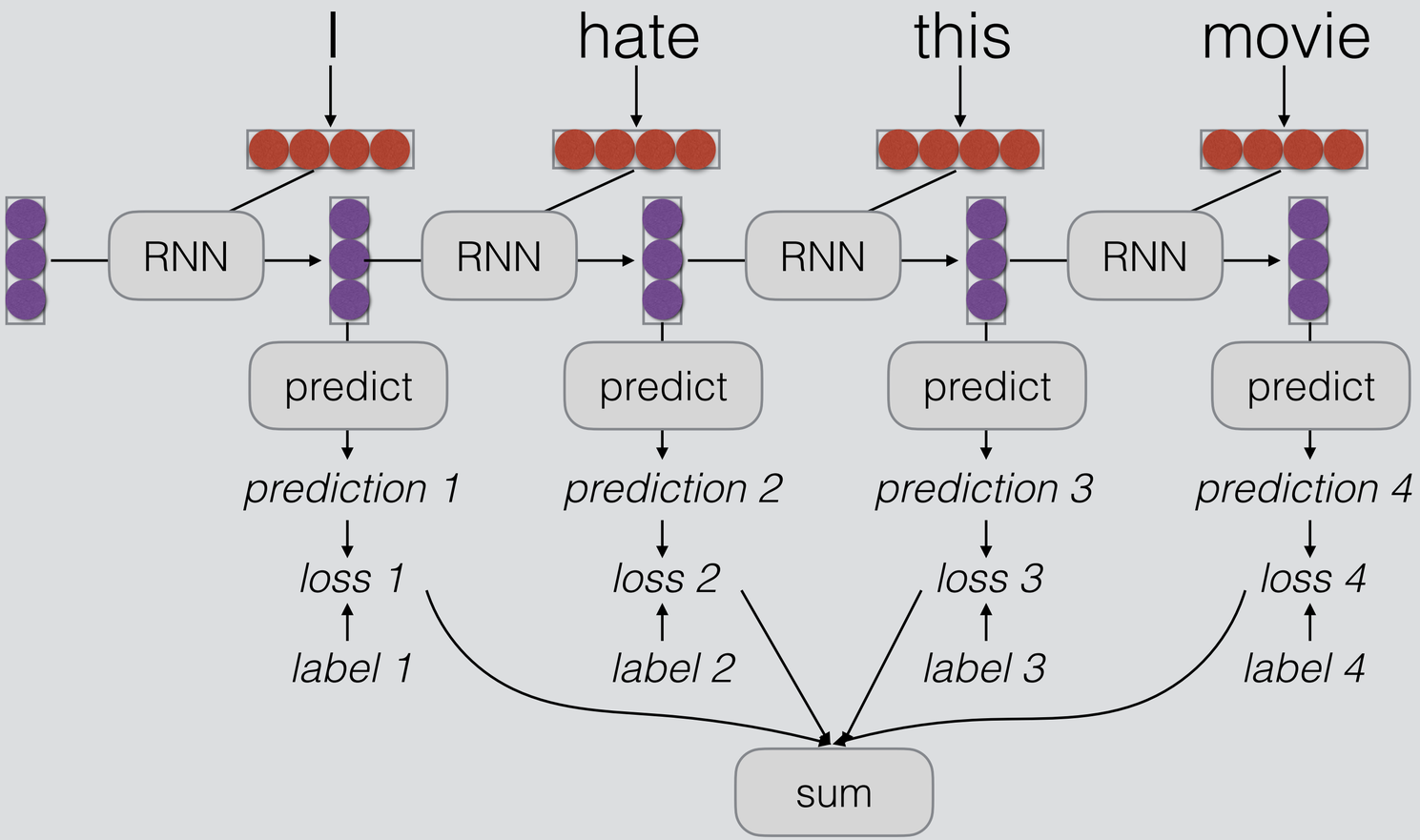
Training RNNs



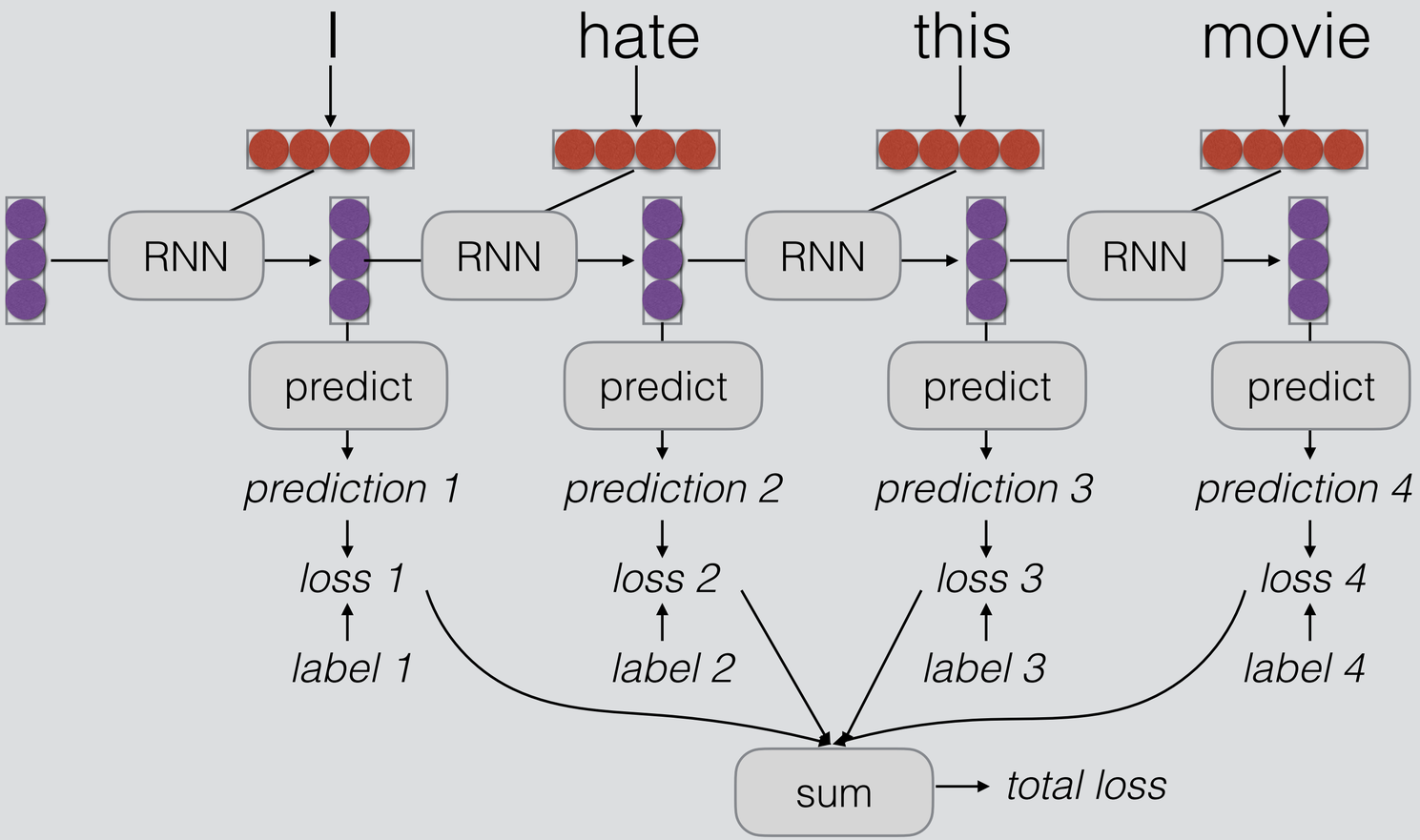
Training RNNs



Training RNNs



Training RNNs

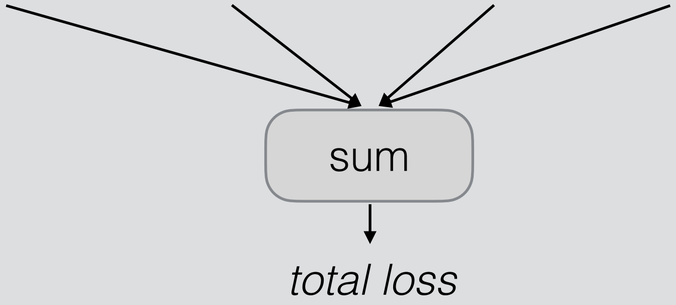


RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop

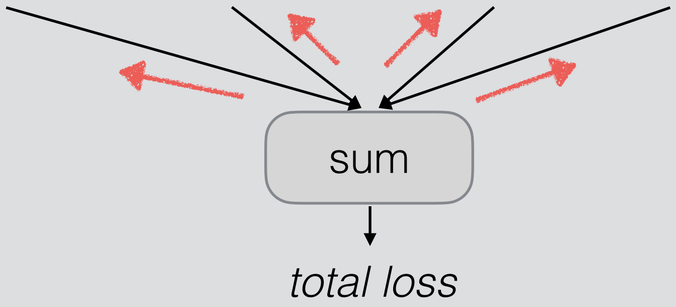
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop



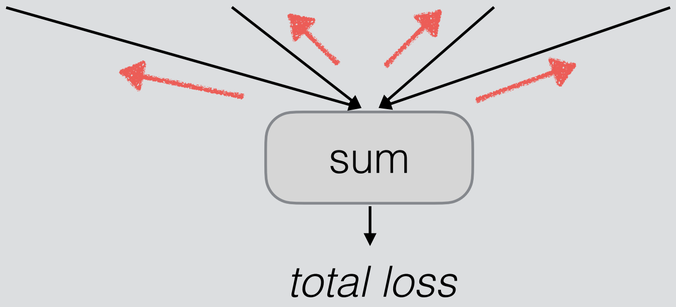
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop



RNN Training

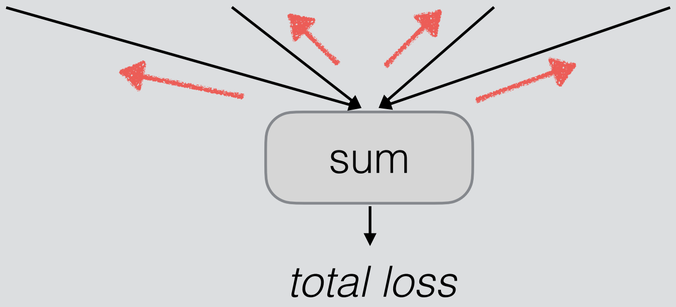
- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop



  - Parameters are tied across time, derivatives are aggregated across all time steps

RNN Training

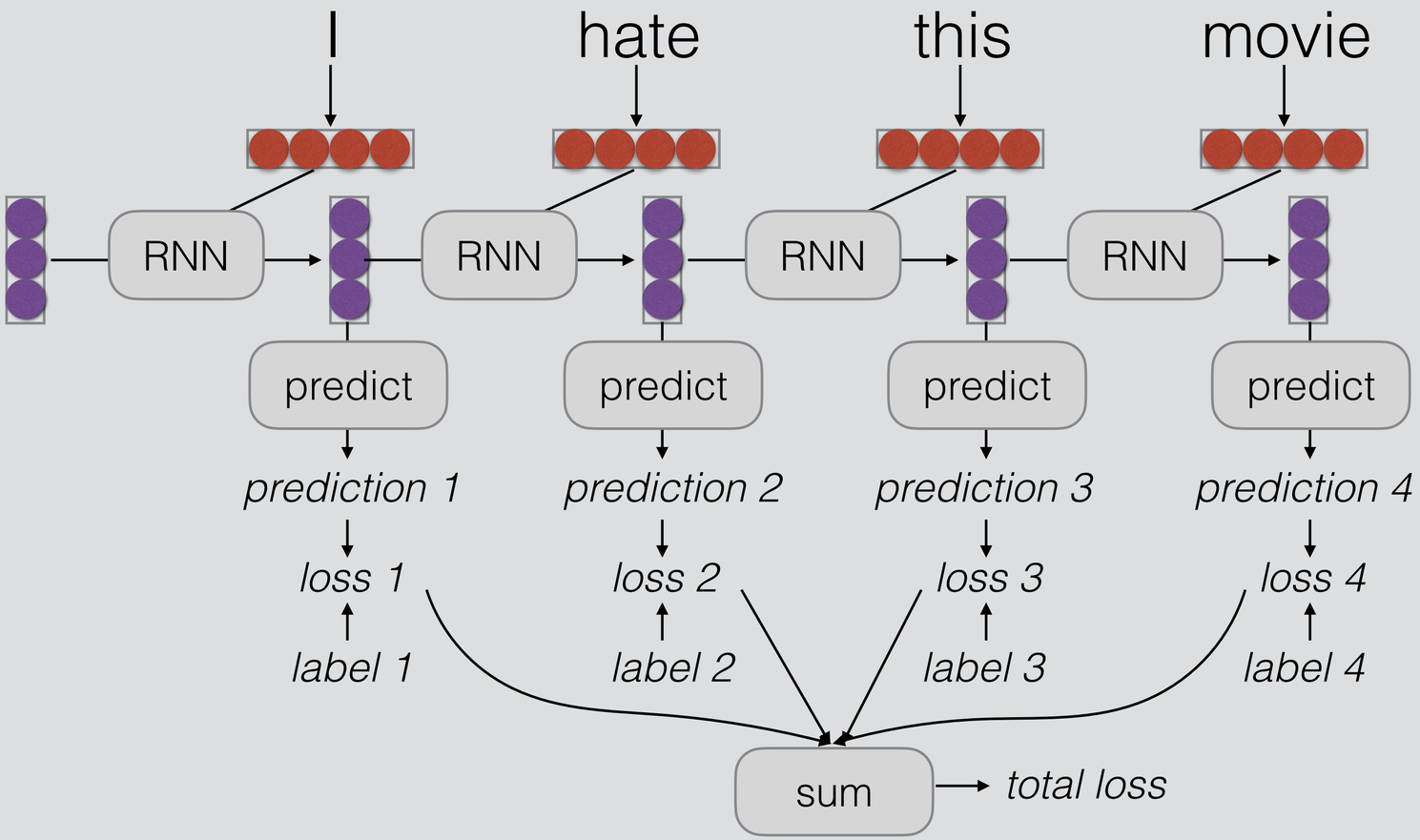
- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop



  - Parameters are tied across time, derivatives are aggregated across all time steps

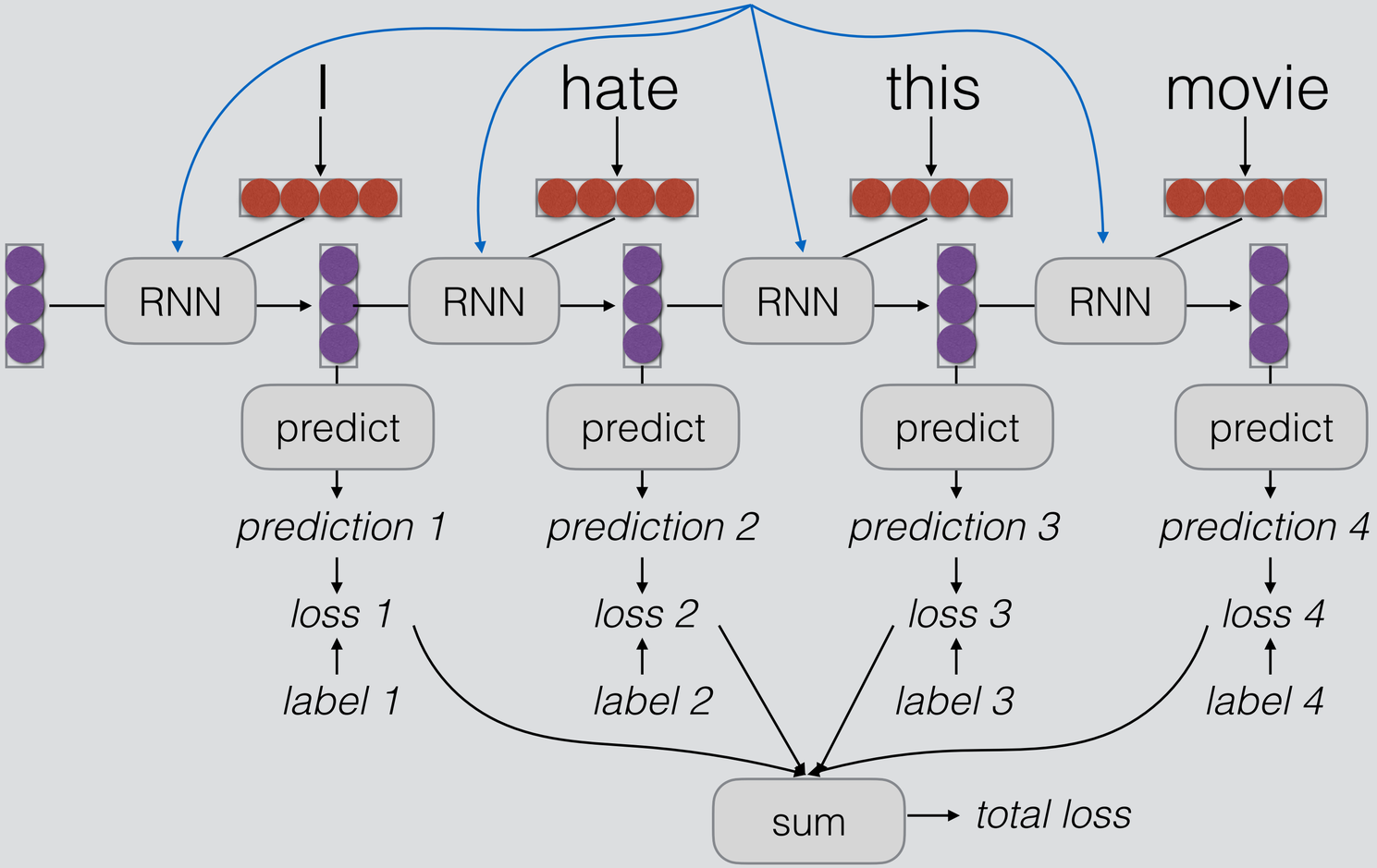
  - This is historically called “backpropagation through time” (BPTT)

Parameter Tying



Parameter Tying

Parameters are shared! Derivatives are accumulated.

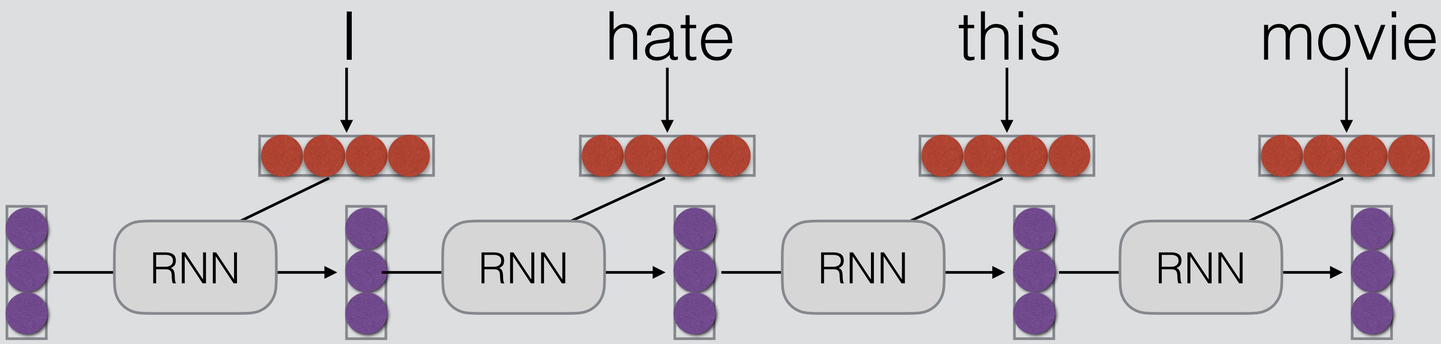


Applications of RNNs

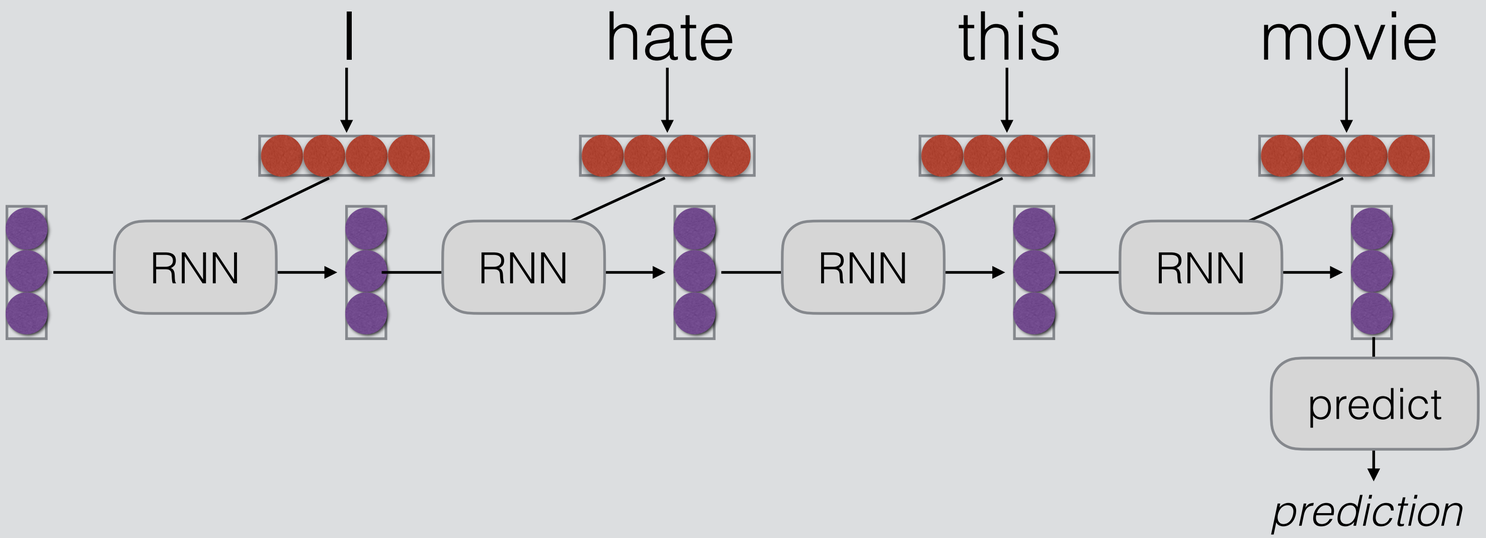
What Can RNNs Do?

- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point

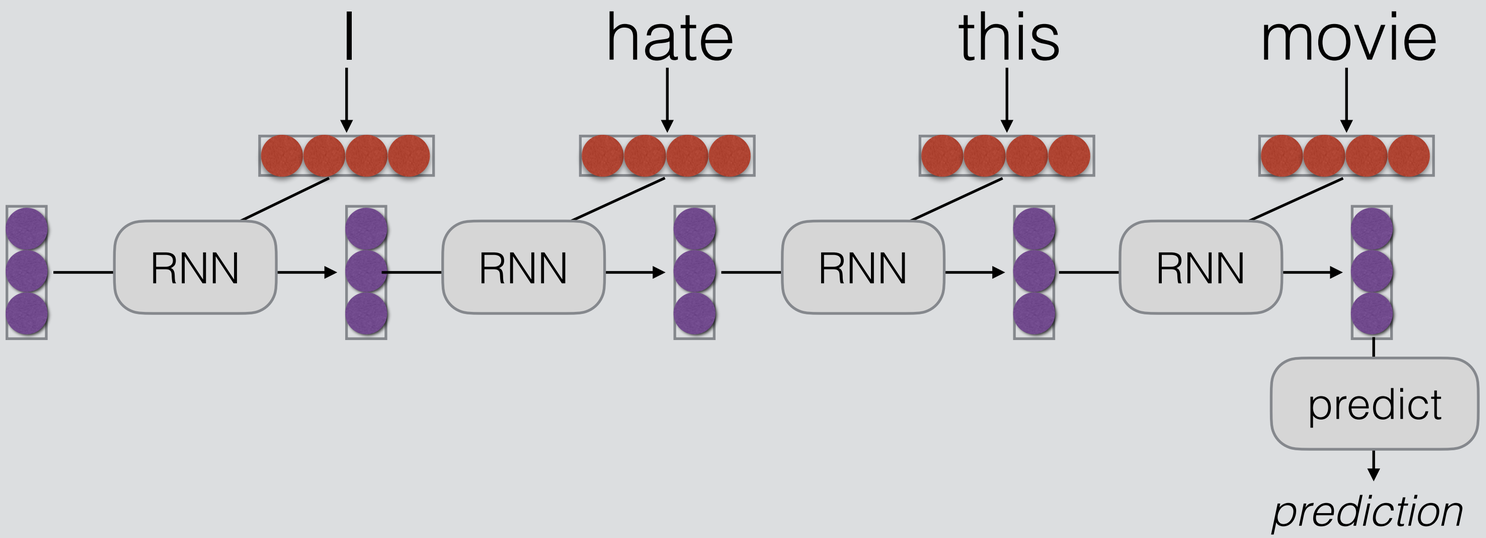
Representing Sentences



Representing Sentences

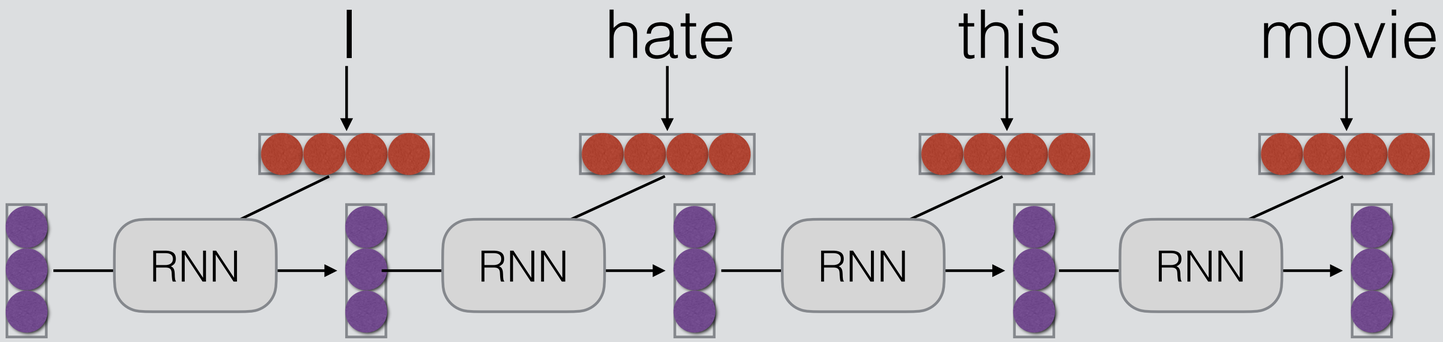


Representing Sentences

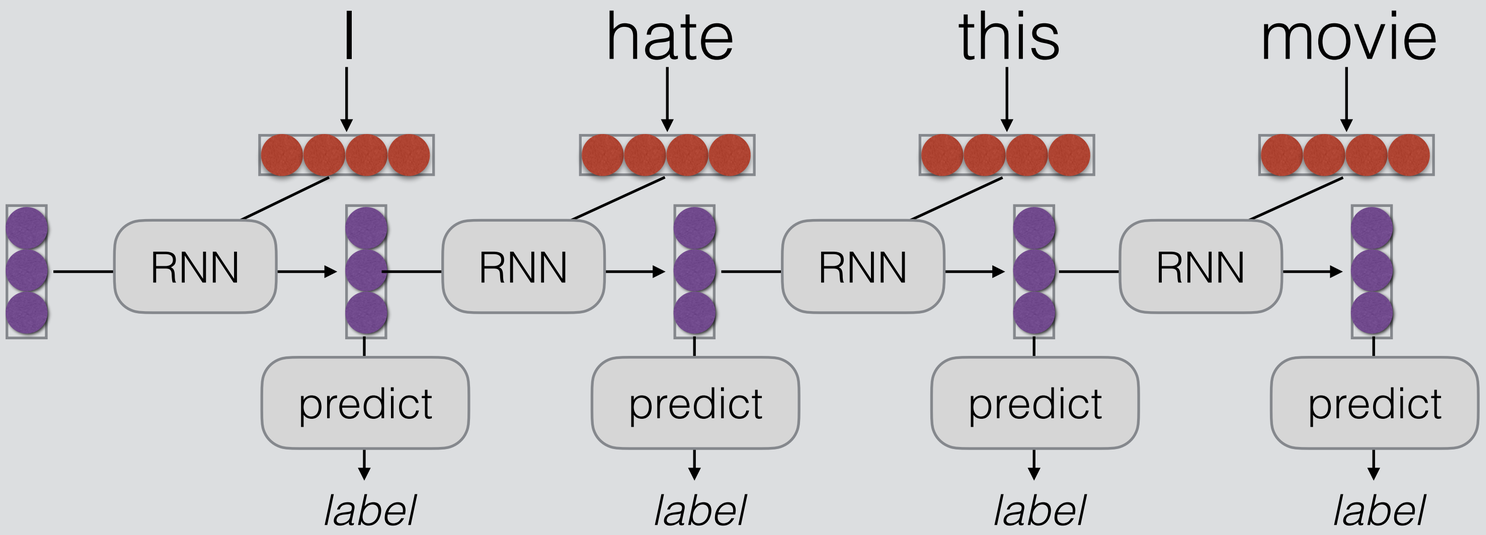


- Sentence classification
- Conditioned generation
- Retrieval

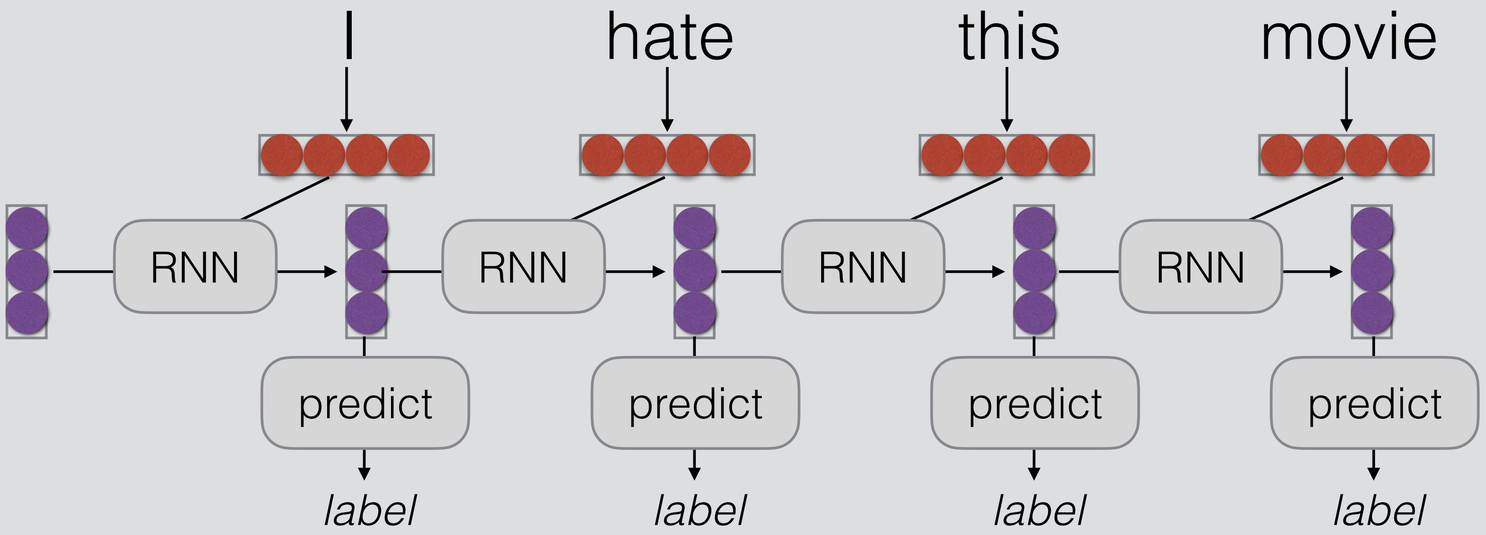
Representing Contexts



Representing Contexts



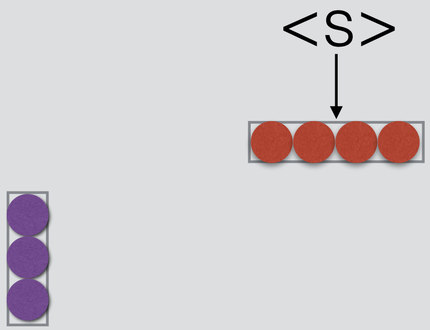
Representing Contexts



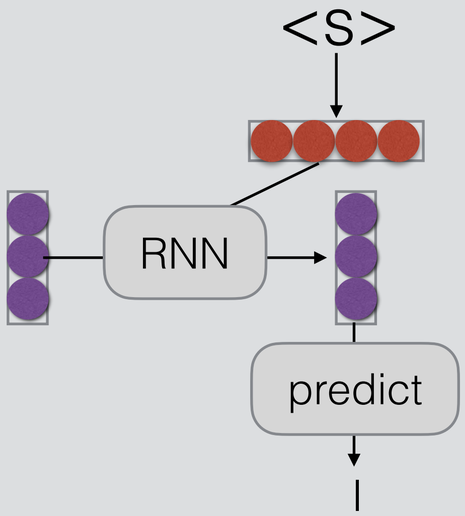
- Tagging
- Language Modeling
- Calculating Representations for Parsing, etc.

e.g. Language Modeling

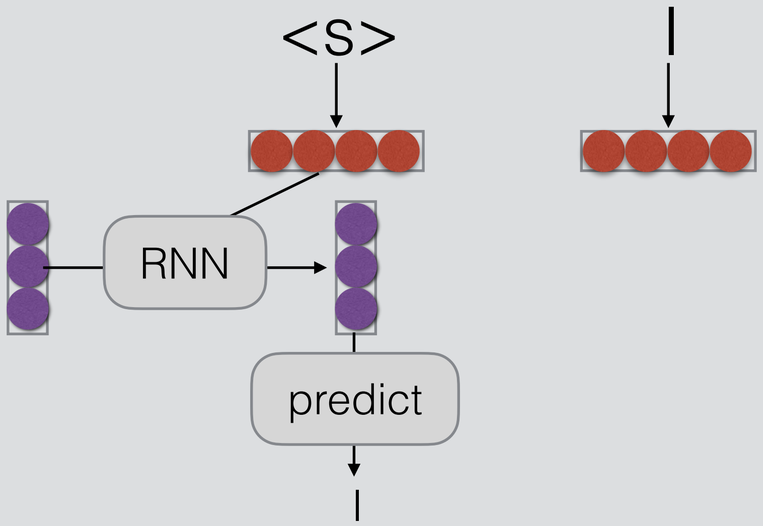
e.g. Language Modeling



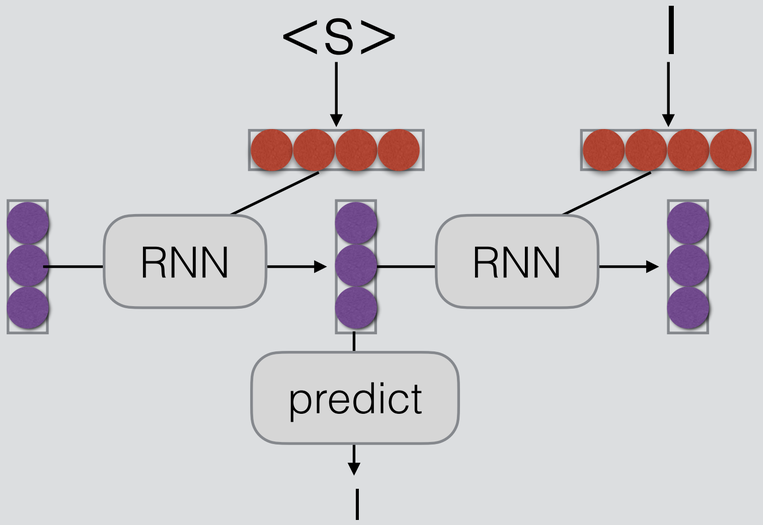
e.g. Language Modeling



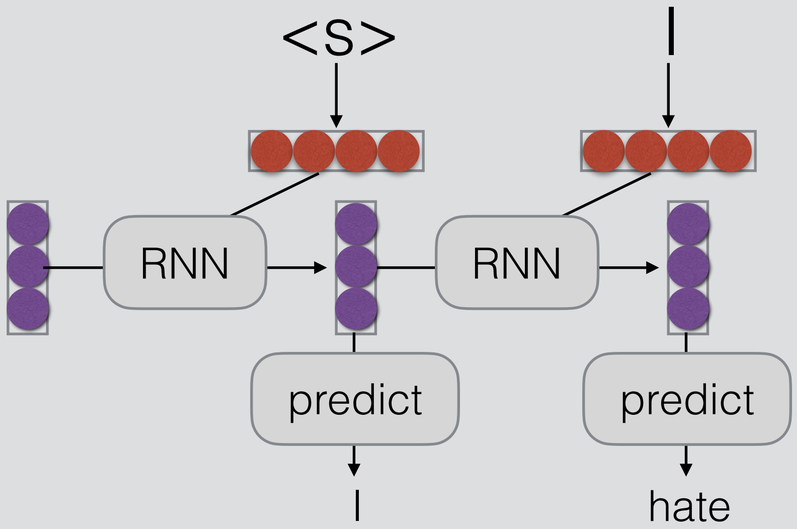
e.g. Language Modeling



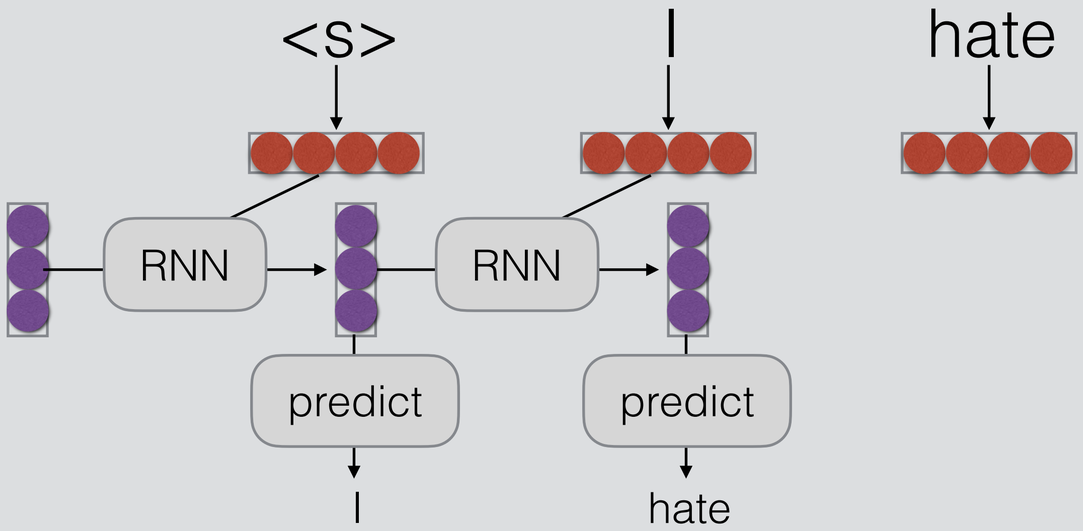
e.g. Language Modeling



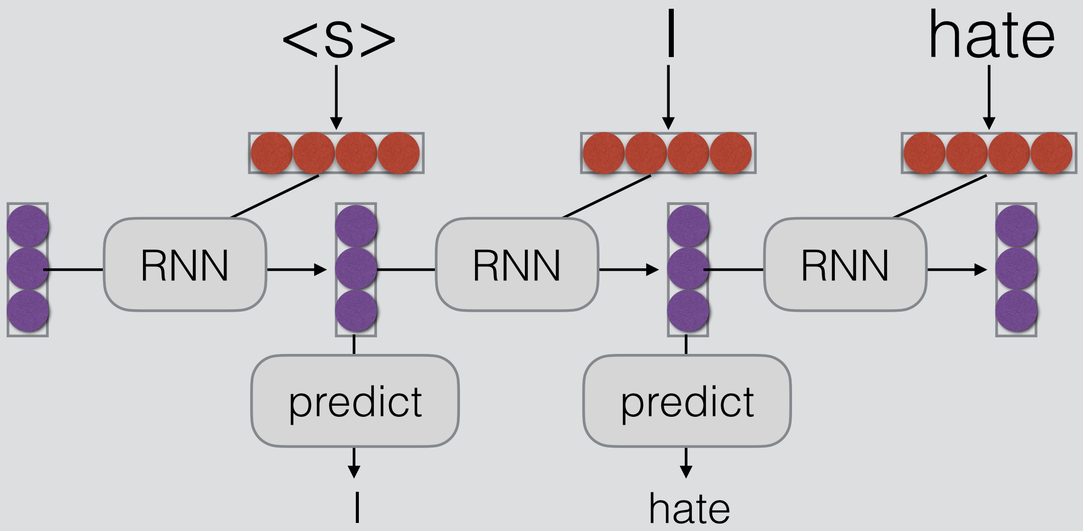
e.g. Language Modeling



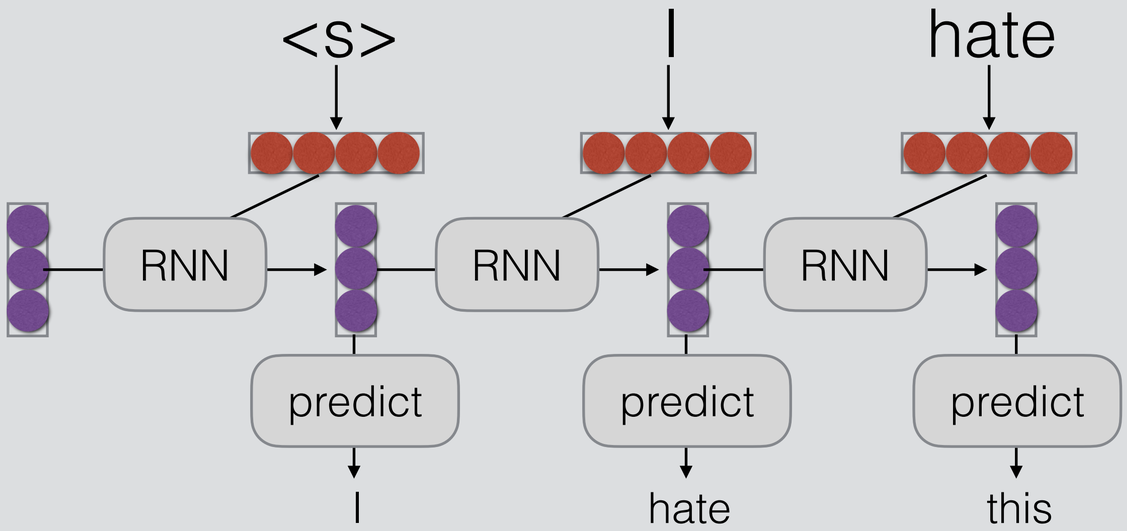
e.g. Language Modeling



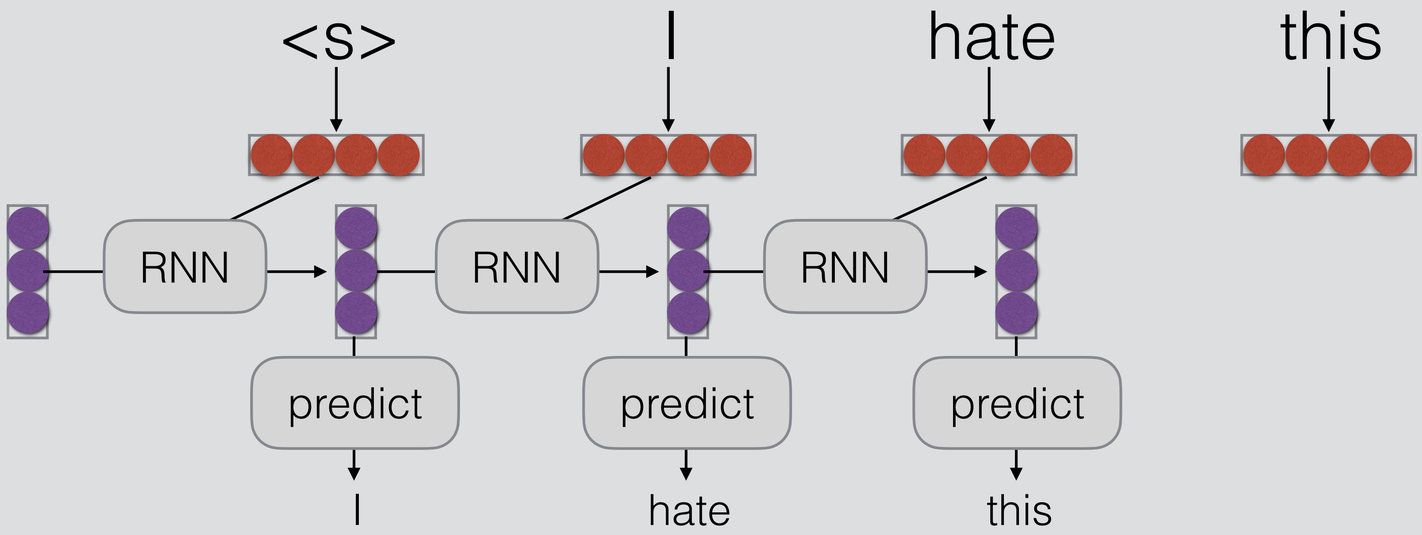
e.g. Language Modeling



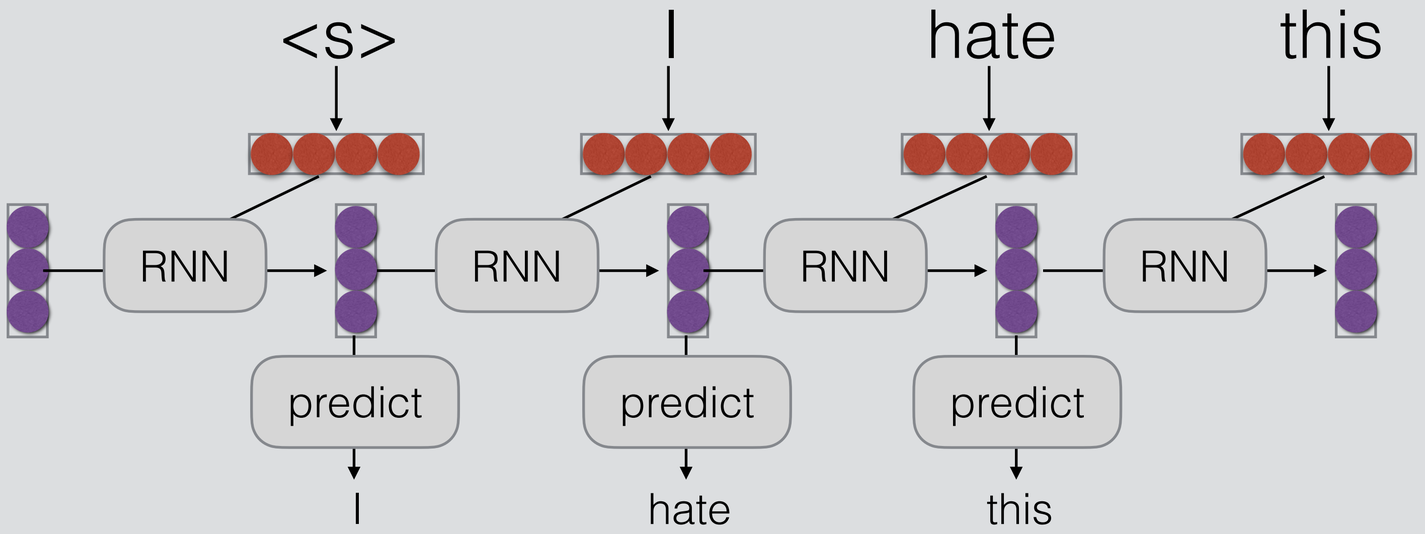
e.g. Language Modeling



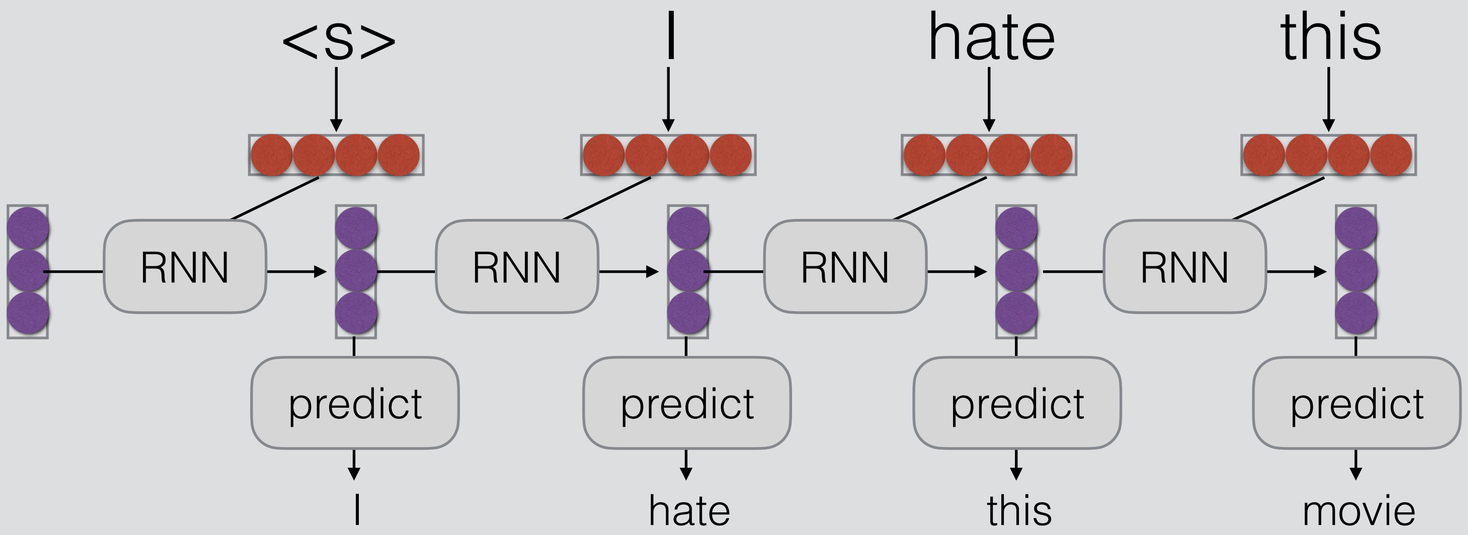
e.g. Language Modeling



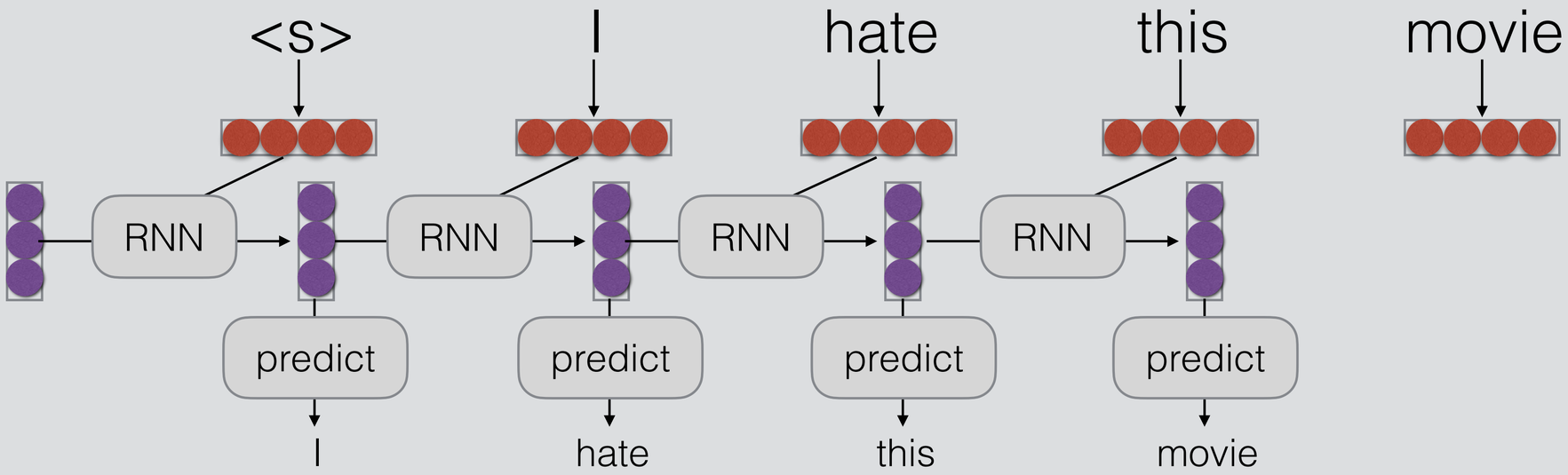
e.g. Language Modeling



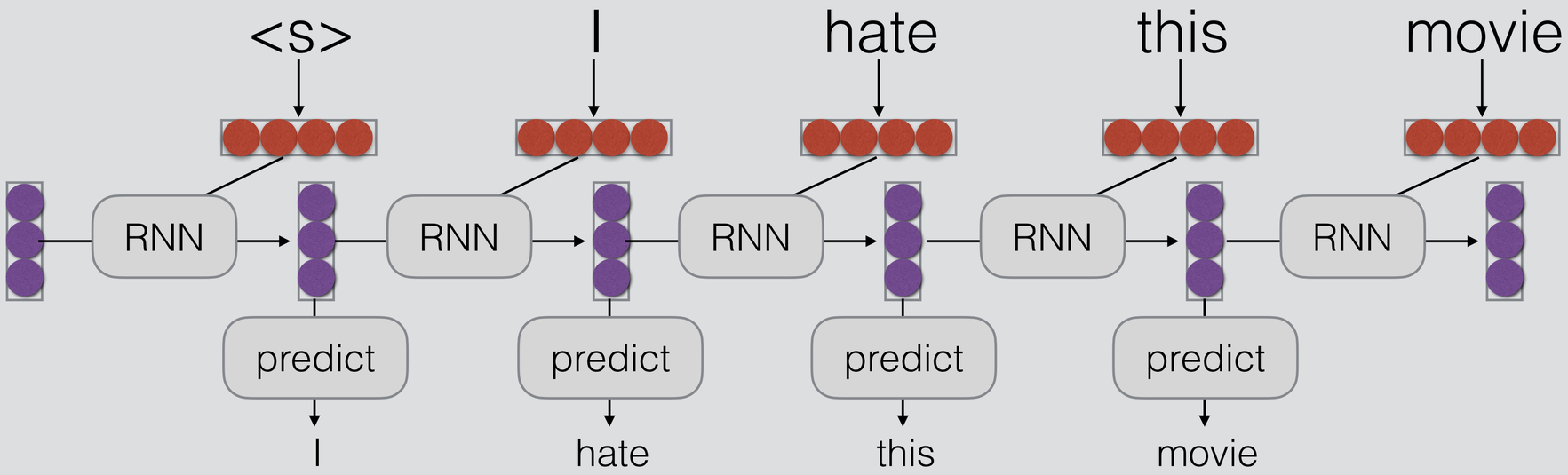
e.g. Language Modeling



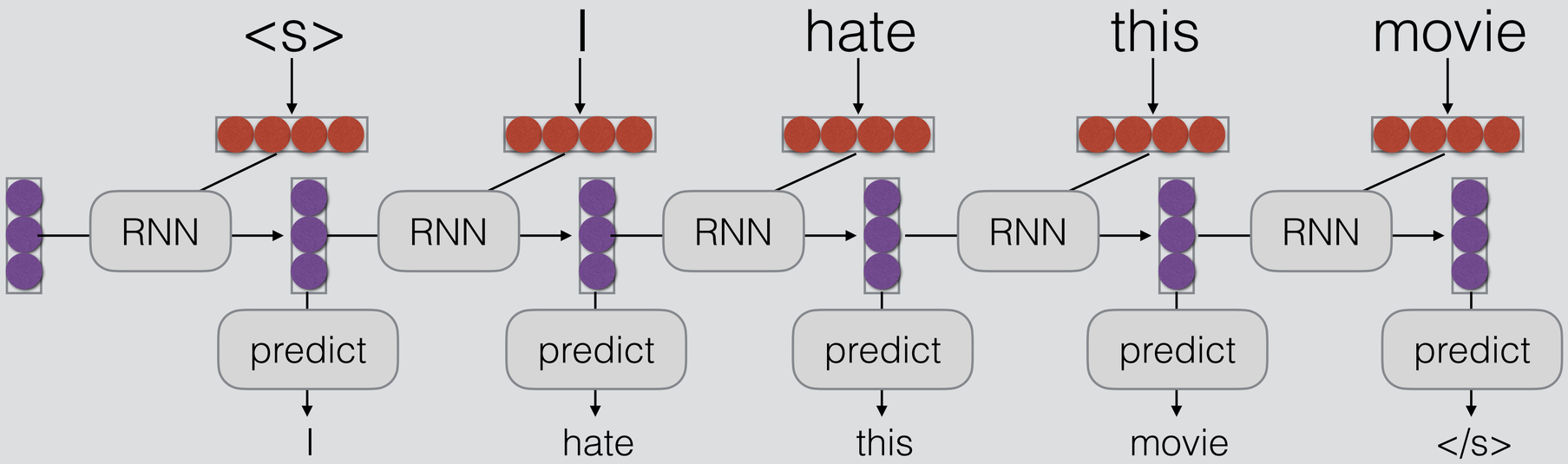
e.g. Language Modeling



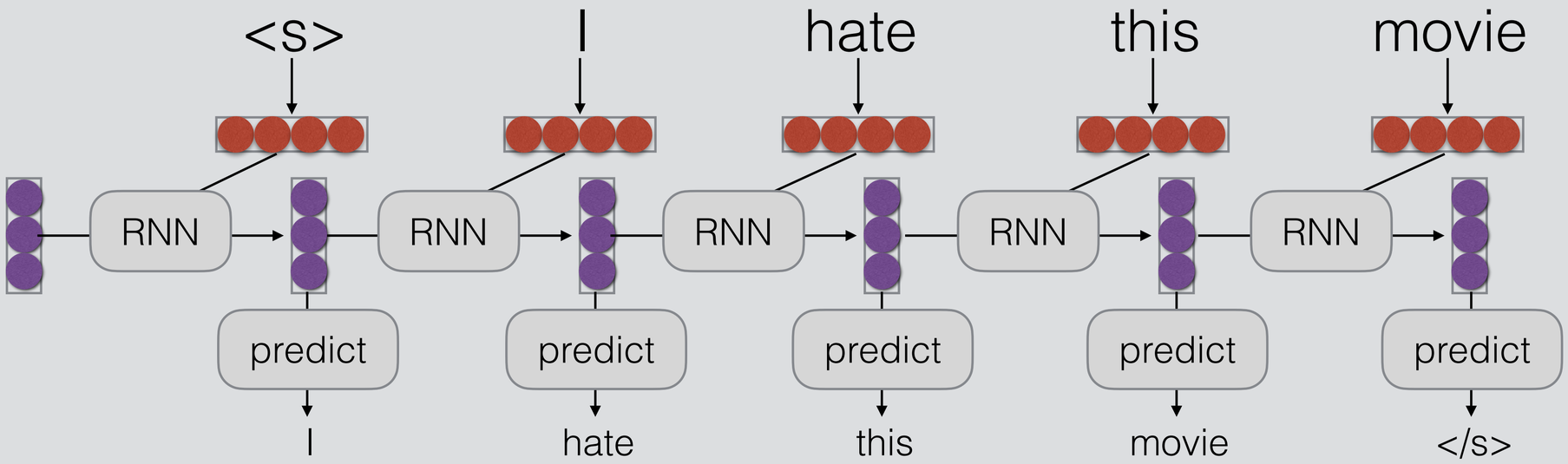
e.g. Language Modeling



e.g. Language Modeling



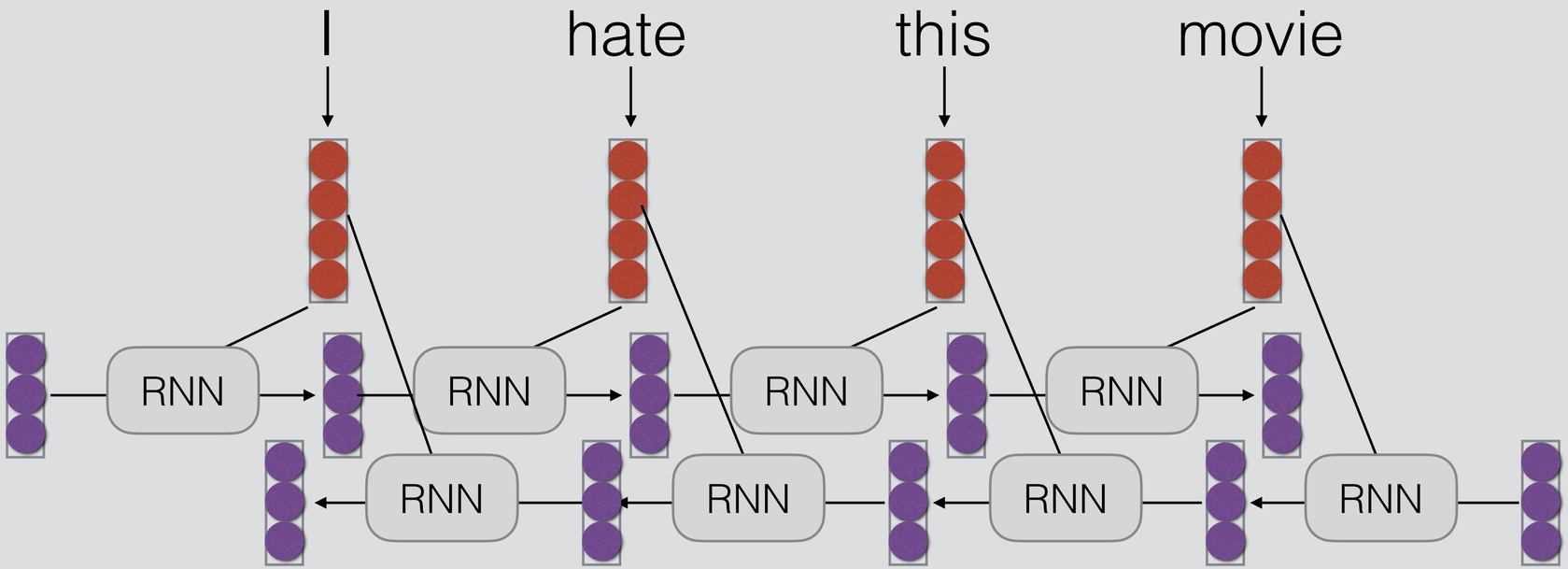
e.g. Language Modeling



- Language modeling is like a tagging task, where each tag is the next word!

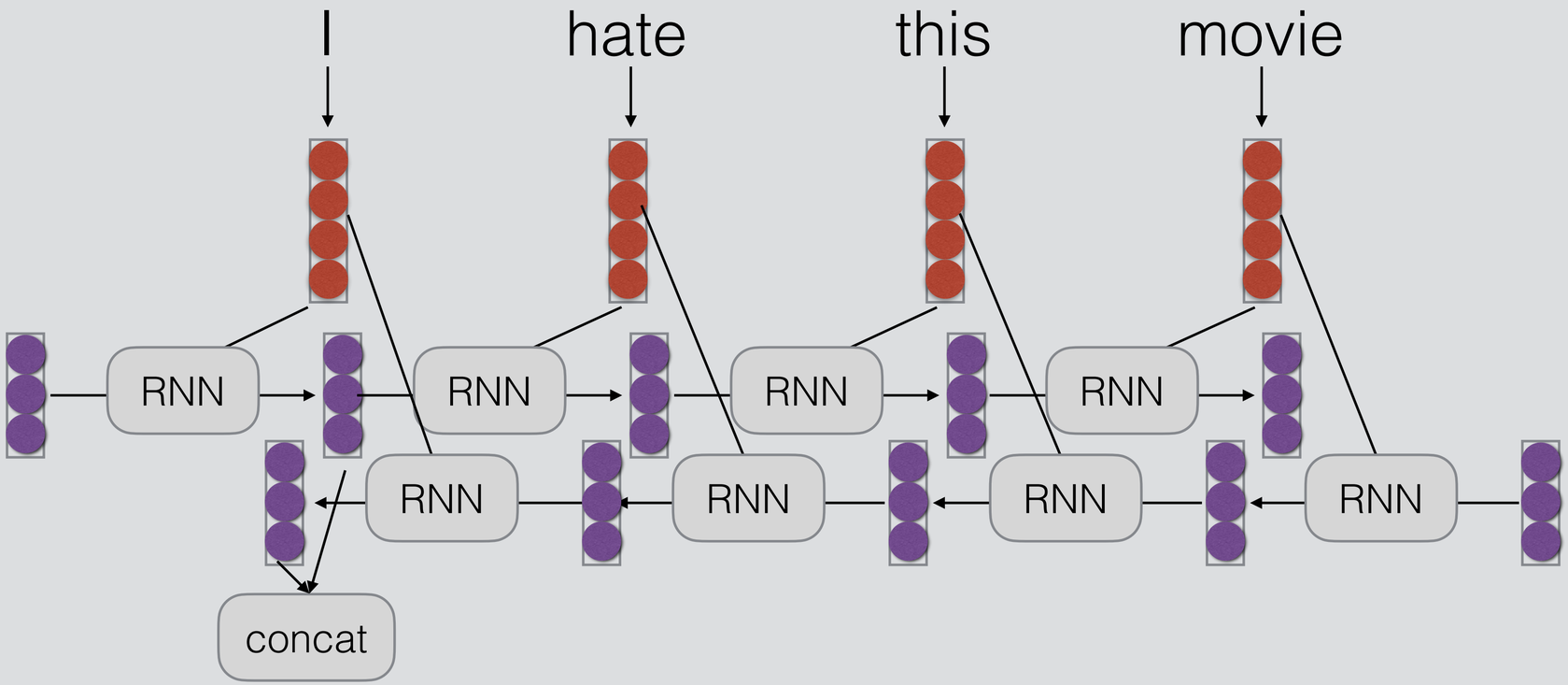
Bi-RNNs

- A simple extension, run the RNN in both directions



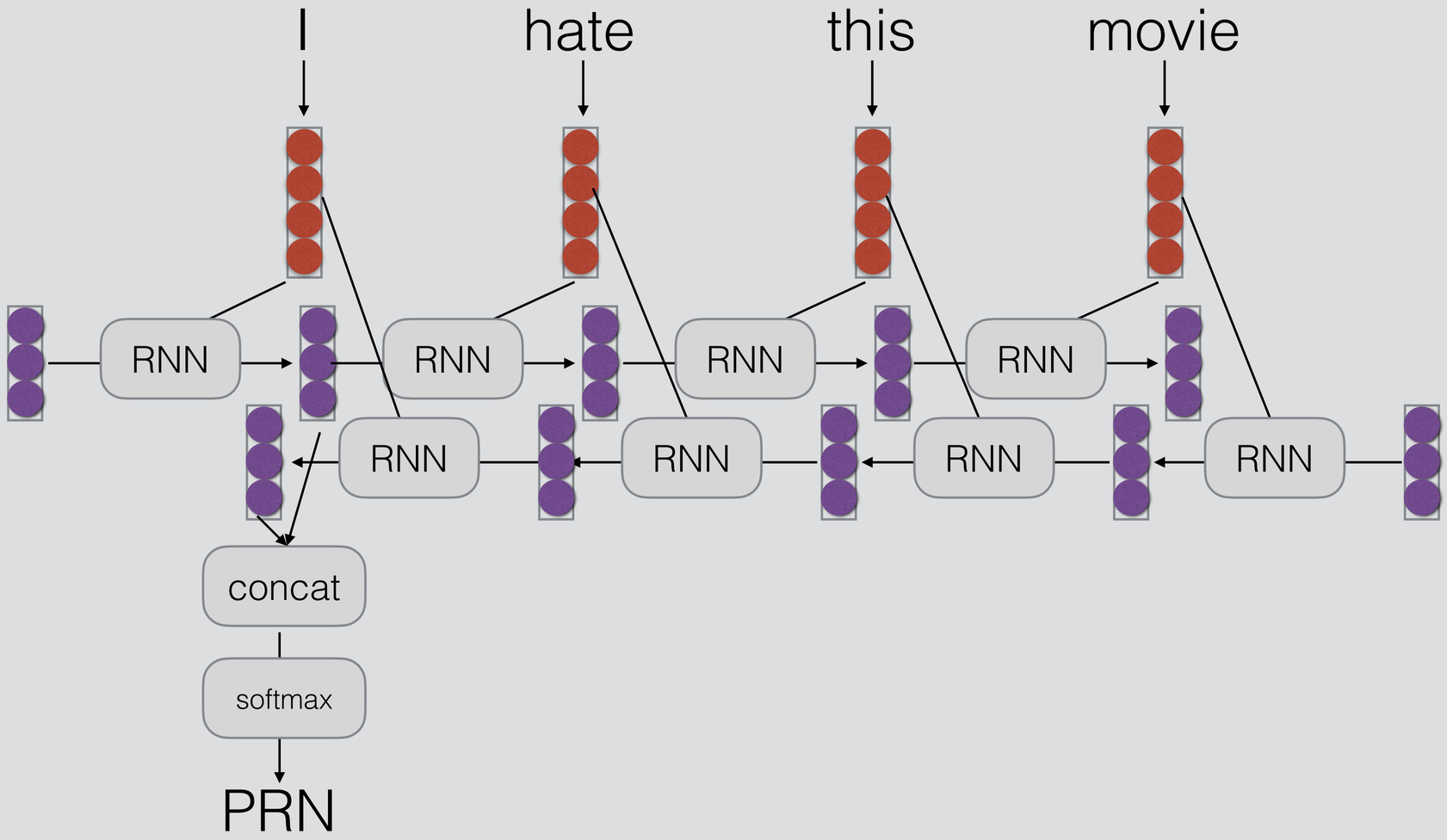
Bi-RNNs

- A simple extension, run the RNN in both directions



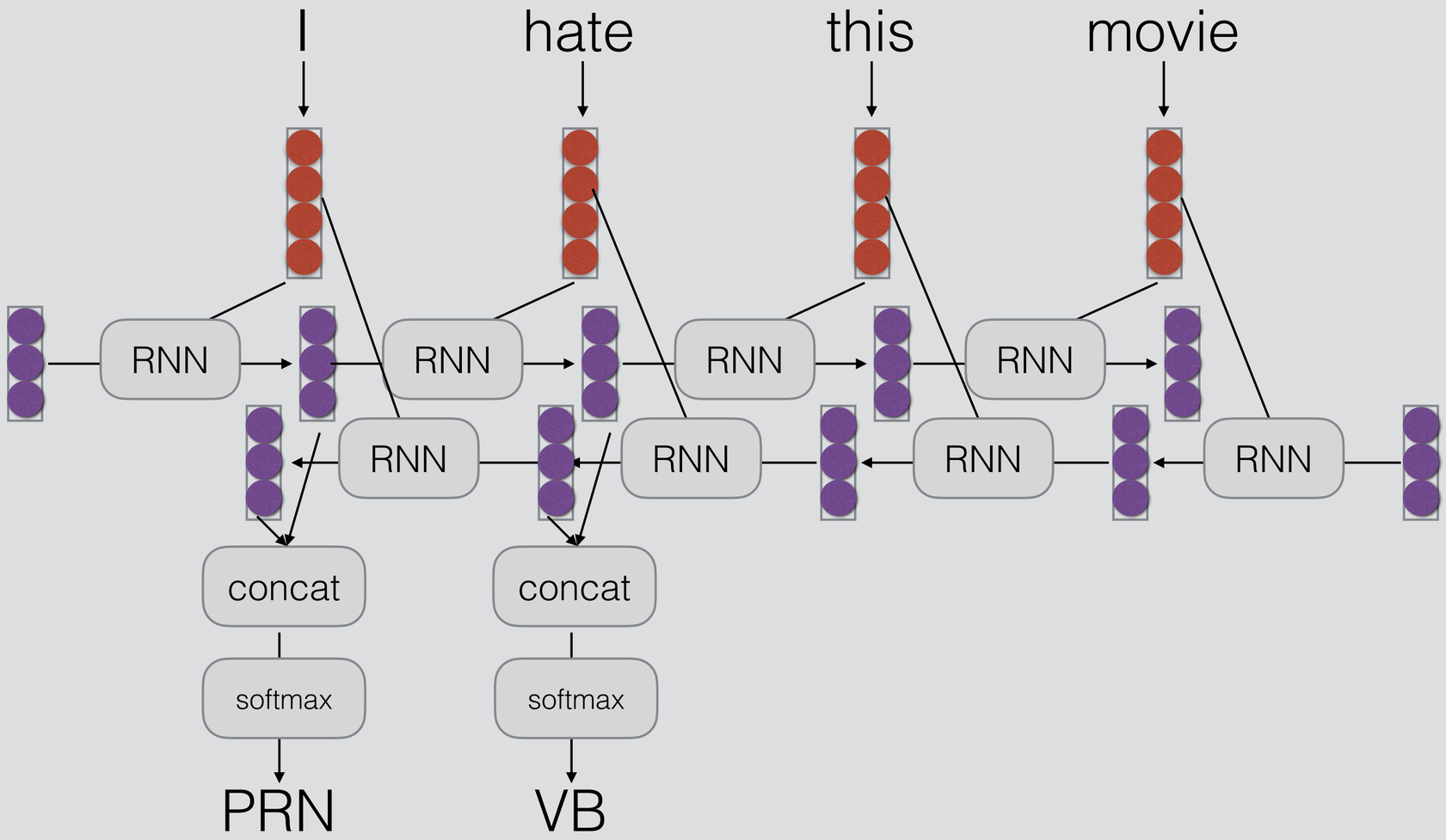
Bi-RNNs

- A simple extension, run the RNN in both directions



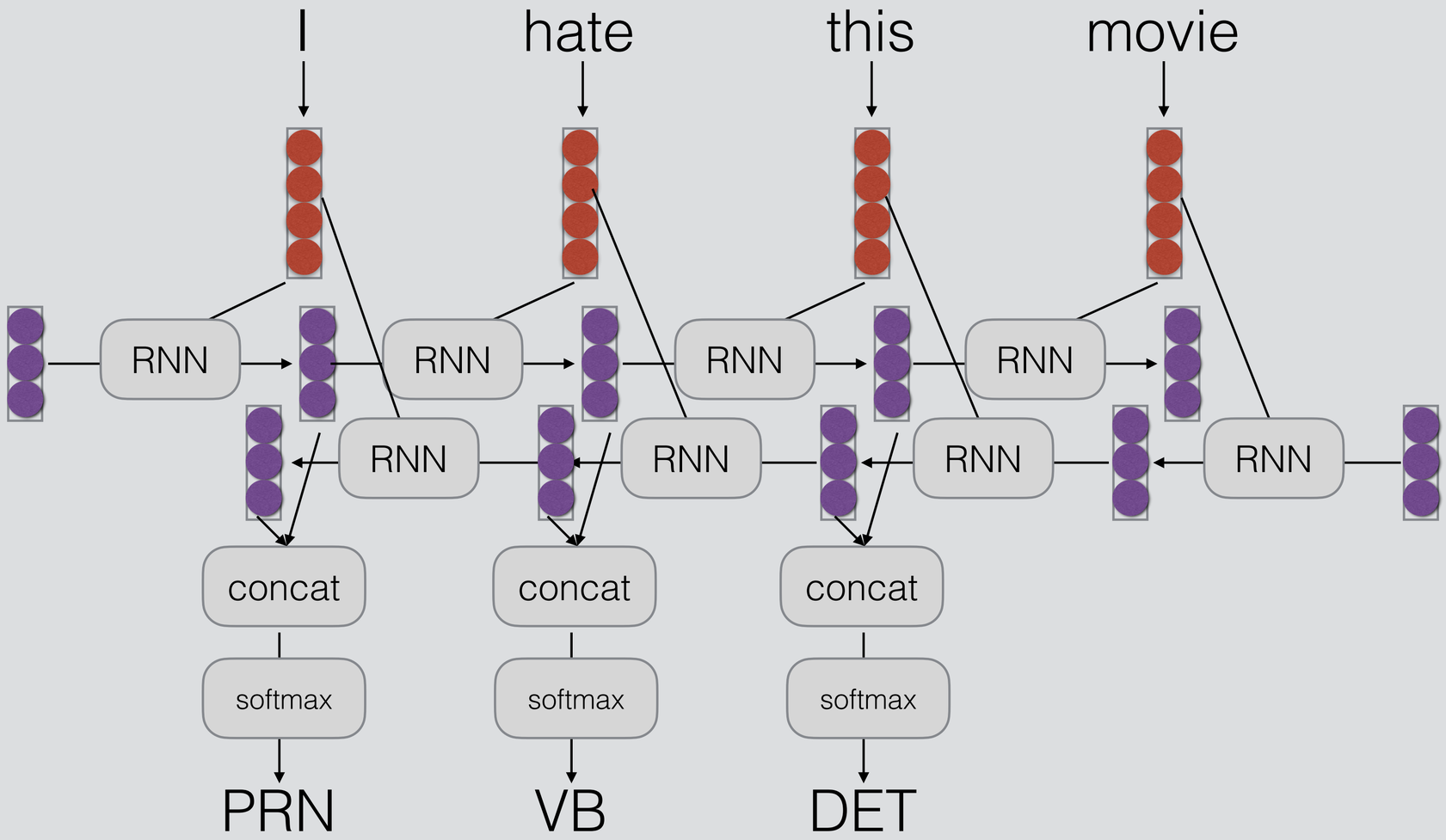
Bi-RNNs

- A simple extension, run the RNN in both directions



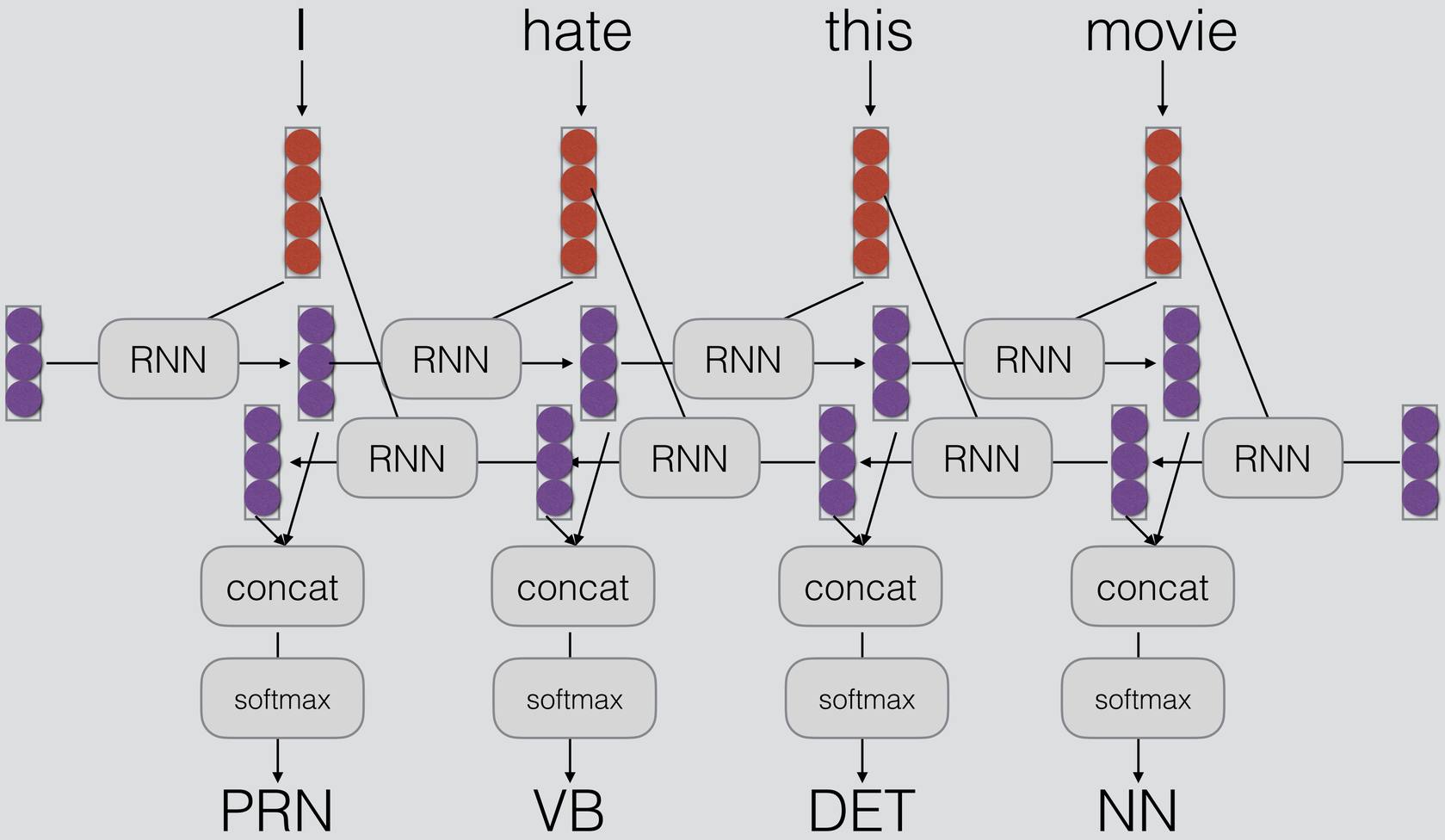
Bi-RNNs

- A simple extension, run the RNN in both directions



Bi-RNNs

- A simple extension, run the RNN in both directions



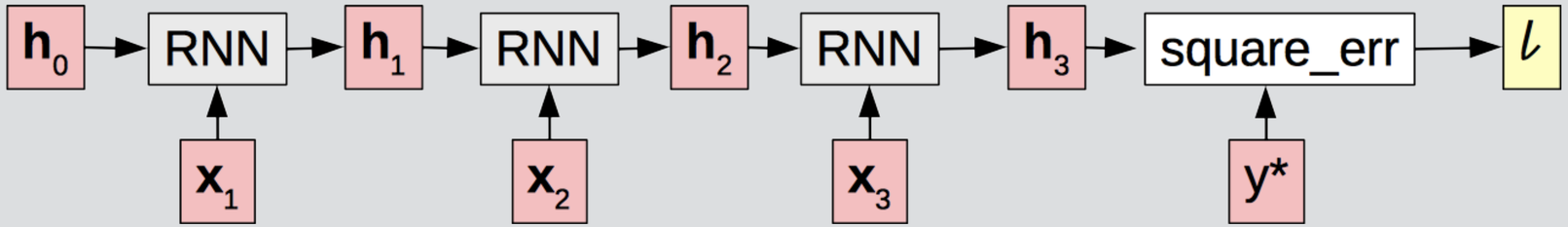
Vanishing Gradients

$$\mathbf{h}_t = \tanh(W\mathbf{x}_t + V\mathbf{h}_{t-1} + \mathbf{b}_h)$$

Vanishing Gradient

- Gradients decrease as they get pushed back

$$\frac{dl}{d_{h_0}} = \text{tiny} \qquad \frac{dl}{d_{h_1}} = \text{small} \qquad \frac{dl}{d_{h_2}} = \text{med.} \qquad \frac{dl}{d_{h_3}} = \text{large}$$



Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." *International conference on machine learning*. PMLR, 2013.

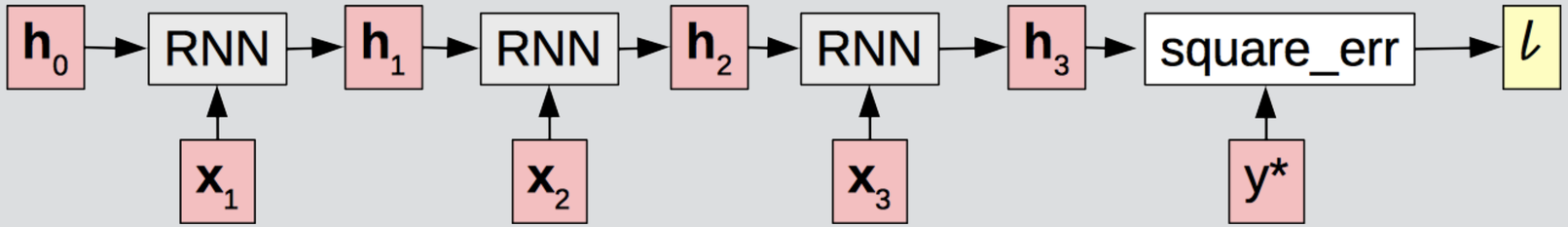
http://neuralnetworksanddeeplearning.com/chap5.html#the_vanishing_gradient_problem

$$\mathbf{h}_t = \tanh(W\mathbf{x}_t + V\mathbf{h}_{t-1} + \mathbf{b}_h)$$

Vanishing Gradient

- Gradients decrease as they get pushed back

$$\frac{dl}{d_{h_0}} = \text{tiny} \qquad \frac{dl}{d_{h_1}} = \text{small} \qquad \frac{dl}{d_{h_2}} = \text{med.} \qquad \frac{dl}{d_{h_3}} = \text{large}$$



- Why? “Squashed” by non-linearities or small weights in matrices.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." *International conference on machine learning*. PMLR, 2013.

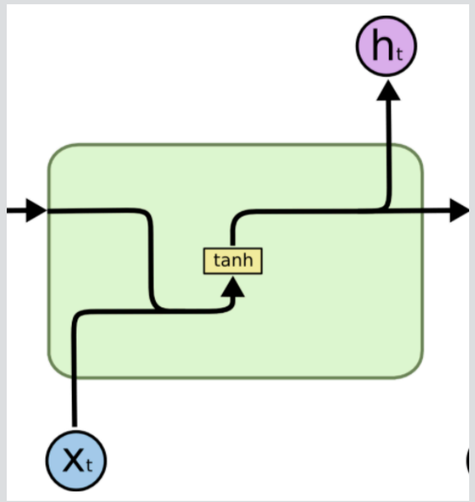
http://neuralnetworksanddeeplearning.com/chap5.html#the_vanishing_gradient_problem

A Solution:
Long Short-term Memory
(Hochreiter and Schmidhuber 1997)

- **Basic idea:** make **additive** connections between time steps
- Addition does not modify the gradient, no vanishing
- Gates to control the information flow

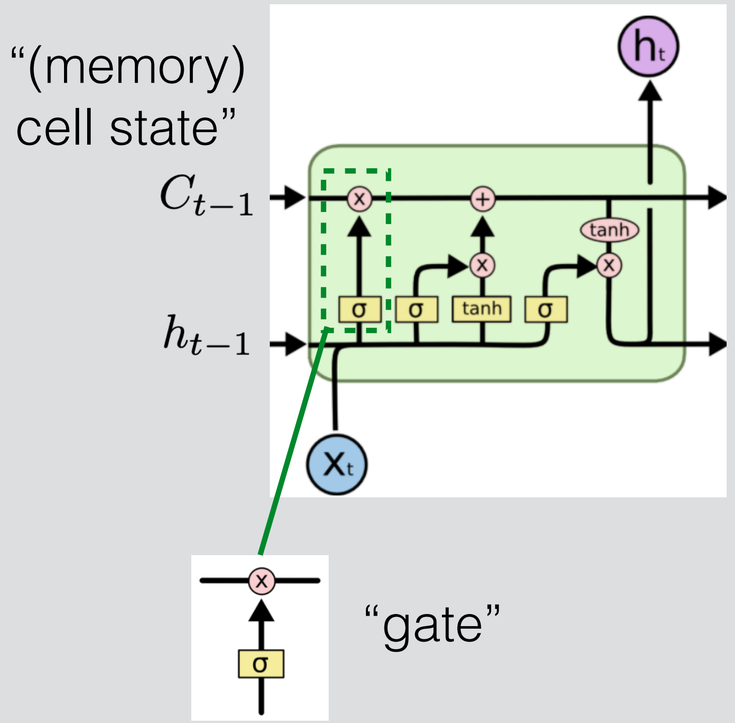
Long Short-term Memory

Standard RNN Cell



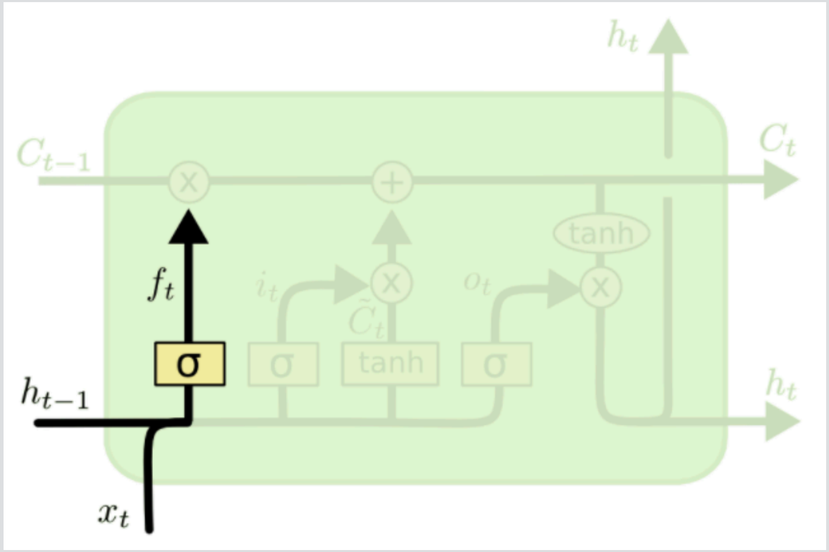
$$\mathbf{h}_t = tanh(W\mathbf{x}_t + V\mathbf{h}_{t-1} + \mathbf{b}_h)$$
$$= tanh(W'[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_h)$$

LSTM Cell



Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory



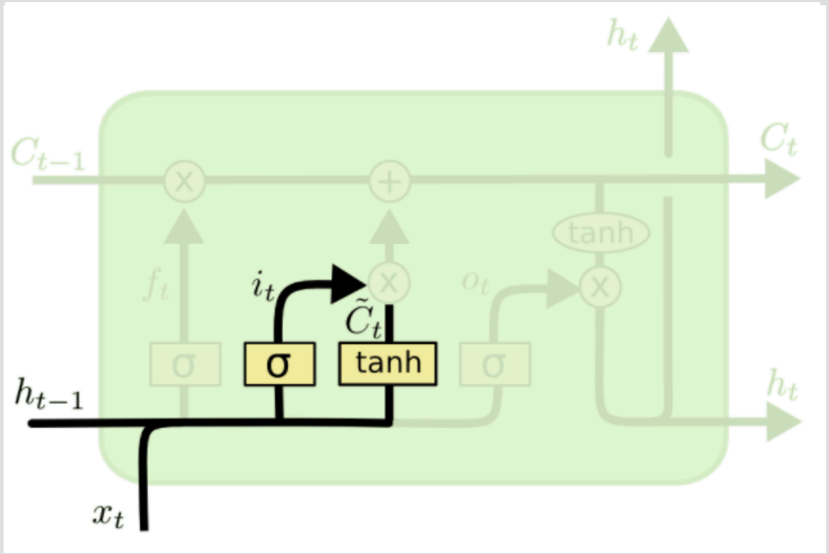
- What to forget from memory?

$$f_t = \sigma(W_f[h_{t-1}, x_t]) + b_f)$$

("forget gate")

Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory



- What to forget from memory?

$$f_t = \sigma(W_f[h_{t-1}, x_t]) + b_f)$$

("forget gate")

- What new info to remember?

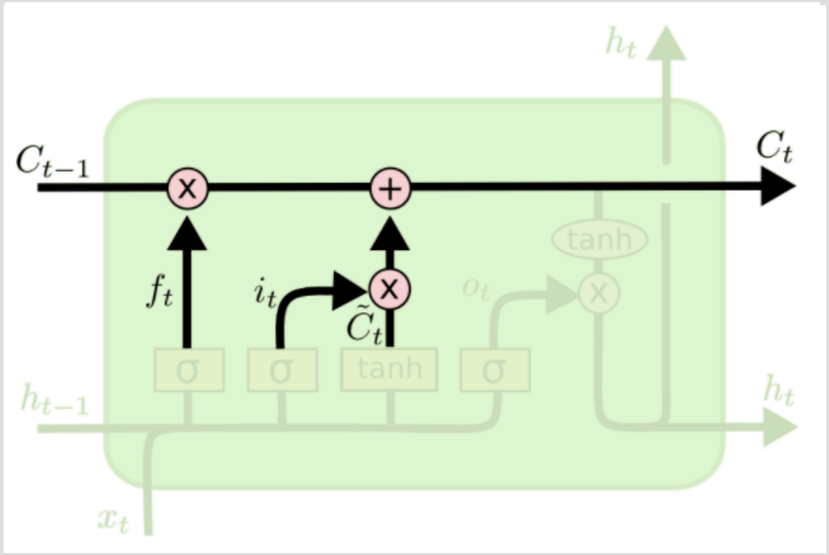
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

("input gate")

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory



- What to forget from memory?

$$f_t = \sigma(W_f[h_{t-1}, x_t]) + b_f)$$

("forget gate")

- What new info to remember?

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

("input gate")

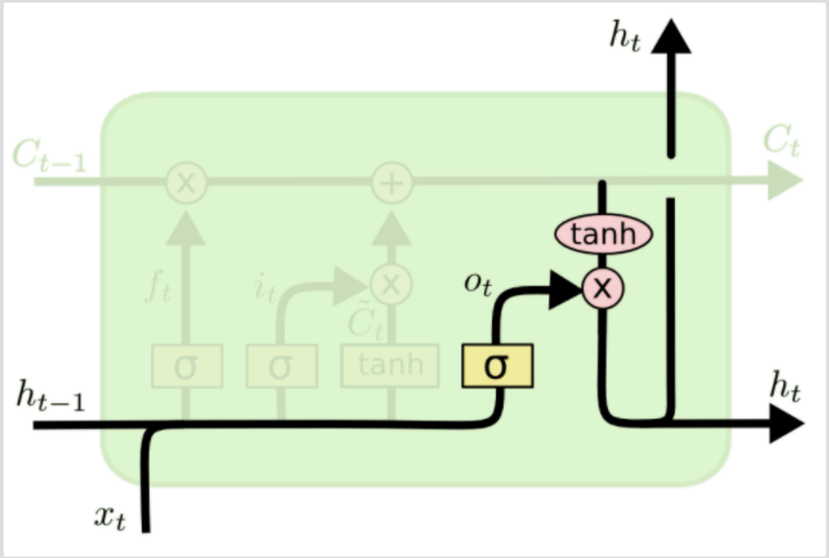
$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- Put together: update the memory

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory



- What to forget from memory?
- What new info to remember?
- Put together: update the memory
- Calculate the output

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

("output gate")

$$h_t = o_t * \tanh(C_t)$$

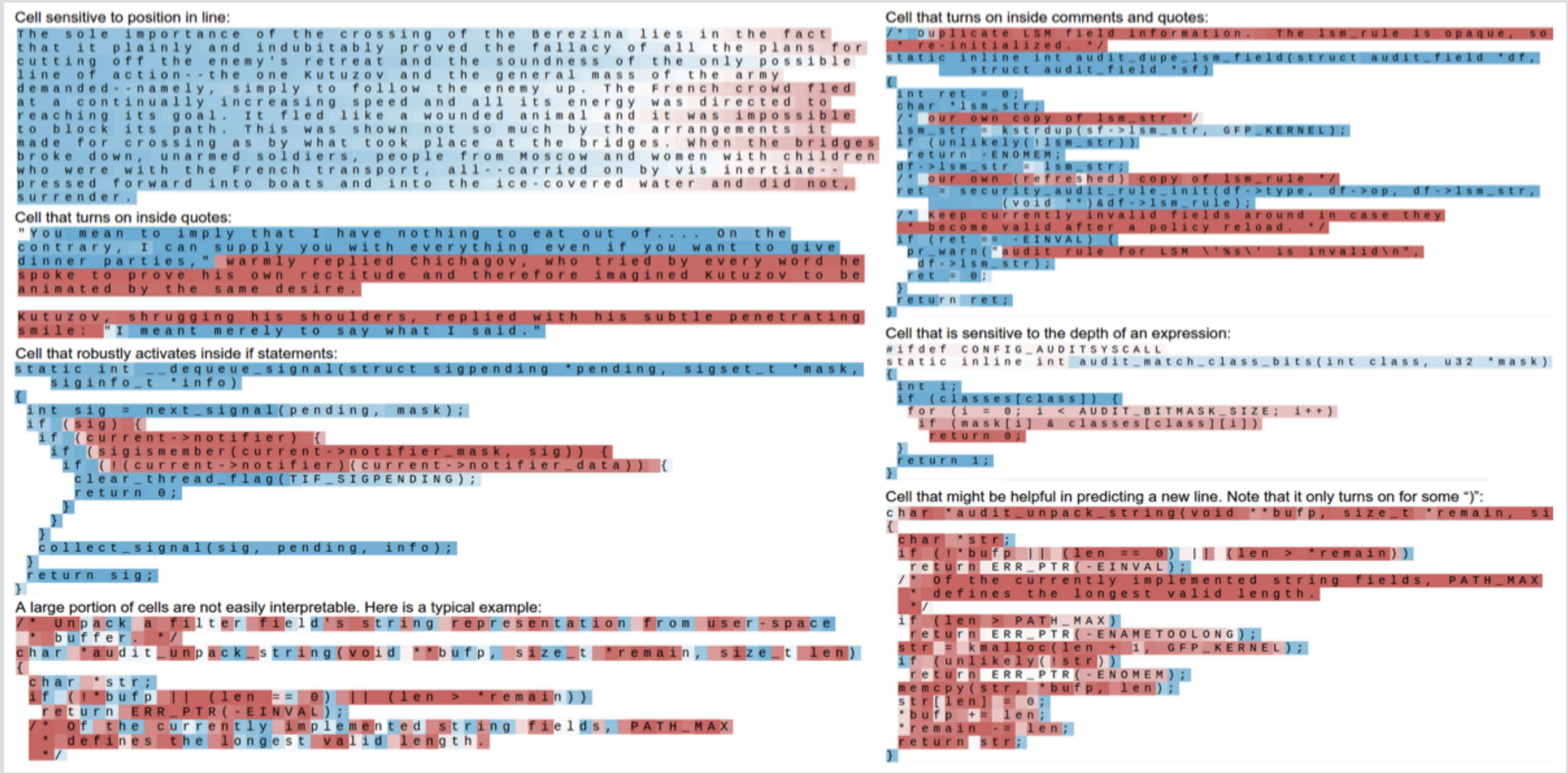
Variant: GRU — Gated Recurrent Units (Cho et al., 2014)

Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

What can LSTMs Learn? (1)

(Karpathy et al. 2015)

- Additive connections make single nodes surprisingly interpretable



Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
           struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
      df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

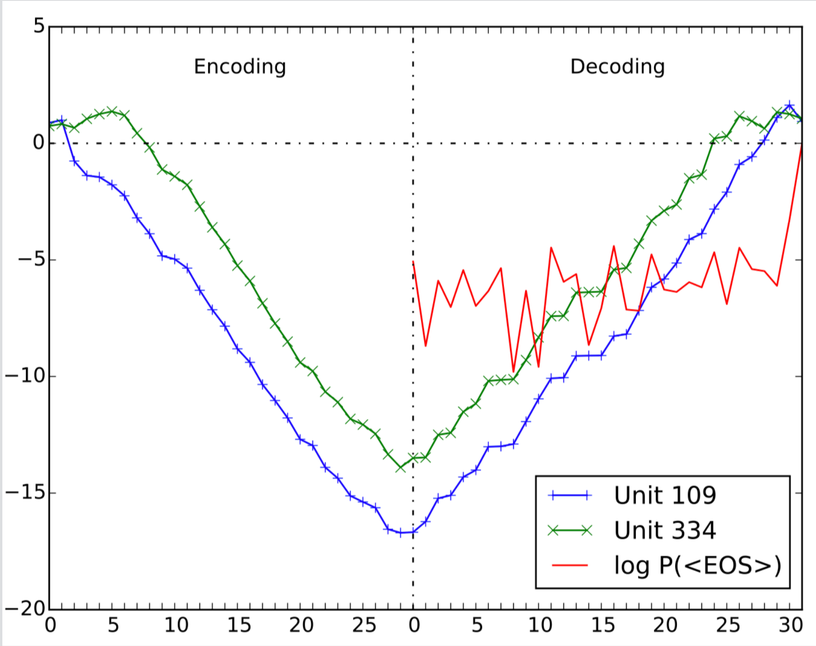
Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

What can LSTMs Learn? (2)

(Shi et al. 2016, Radford et al. 2017)

Count length of sentence



Sentiment

25 August 2003 League of Extraordinary Gentlemen: Sean Connery is one of the all time greats and I have been a fan of his since the 1950's. I went to this movie because Sean Connery was the main actor. I had not read reviews or had any prior knowledge of the movie. The movie surprised me quite a bit. The scenery and sights were spectacular, but the plot was unreal to the point of being ridiculous. In my mind this was not one of his better movies it could be the worst. Why he chose to be in this movie is a mystery. For me, going to this movie was a waste of my time. I will continue to go to his movies and add his movies to my video collection. But I can't see wasting money to put this movie in my collection

I found this to be a charming adaptation, very lively and full of fun. With the exception of a couple of major errors, the cast is wonderful. I have to echo some of the earlier comments -- Chynna Phillips is horribly miscast as a teenager. At 27, she's just too old (and, yes, it DOES show), and lacks the singing "chops" for Broadway-style music. Vanessa Williams is a decent-enough singer and, for a non-dancer, she's adequate. However, she is NOT Latina, and her character definitely is. She's also very STRIDENT throughout, which gets tiresome. The girls of Sweet Apple's Conrad Birdie fan club really sparkle -- with special kudos to Brigitta Dau and Chiara Zanni. I also enjoyed Tyne Daly's performance, though I'm not generally a fan of her work. Finally, the dancing Shriners are a riot, especially the dorky three in the bar. The movie is suitable for the whole family, and I highly recommend it.

Efficiency Tricks

Handling Mini-batching

- Mini-batching makes things much faster!

Handling Mini-batching

- Mini-batching makes things much faster!
- But mini-batching in RNNs is harder than in feed-forward networks

Handling Mini-batching

- Mini-batching makes things much faster!
- But mini-batching in RNNs is harder than in feed-forward networks
  - Each word depends on the previous word
  - Sequences are of various length

Mini-batching Method

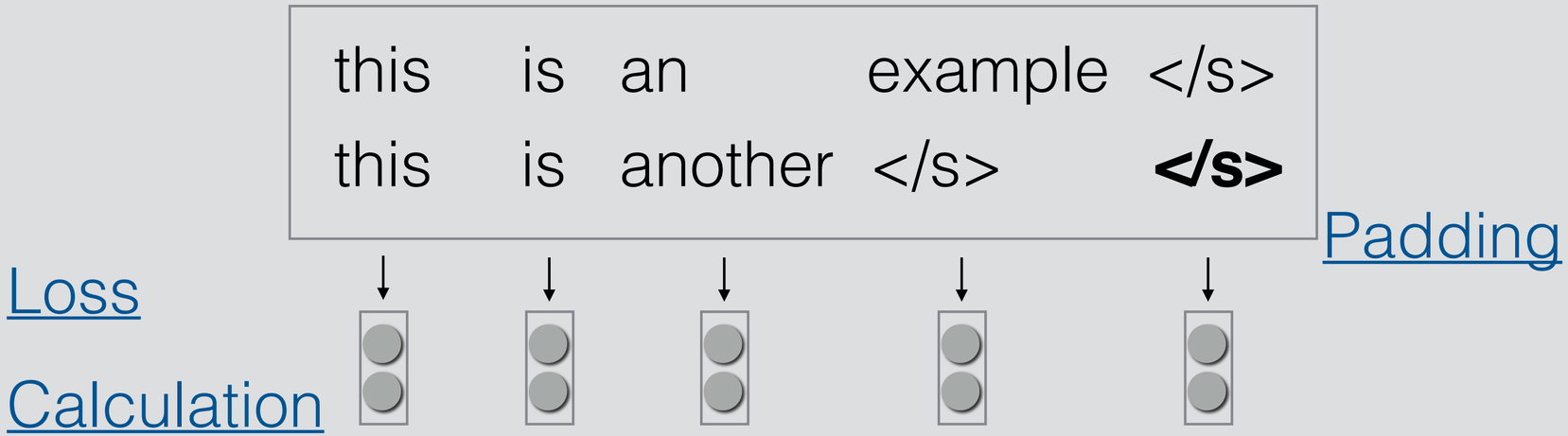
this	is	an	example	</s>
this	is	another	</s>	

Mini-batching Method

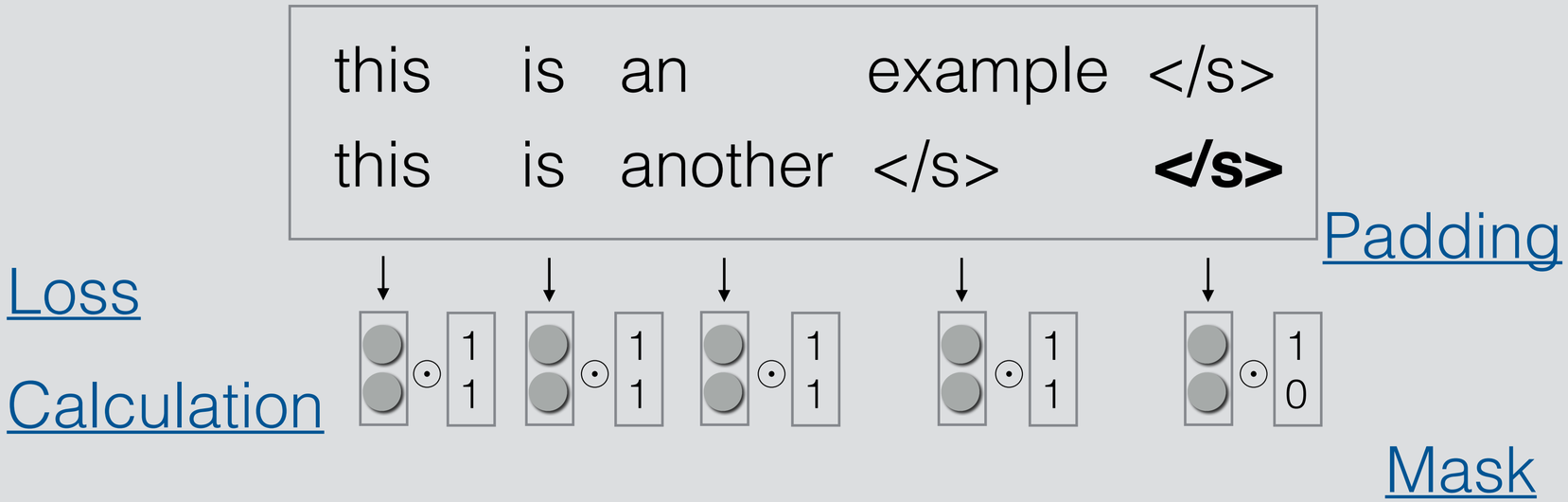
this	is	an	example	</s>
this	is	another	</s>	**</s>**

Padding

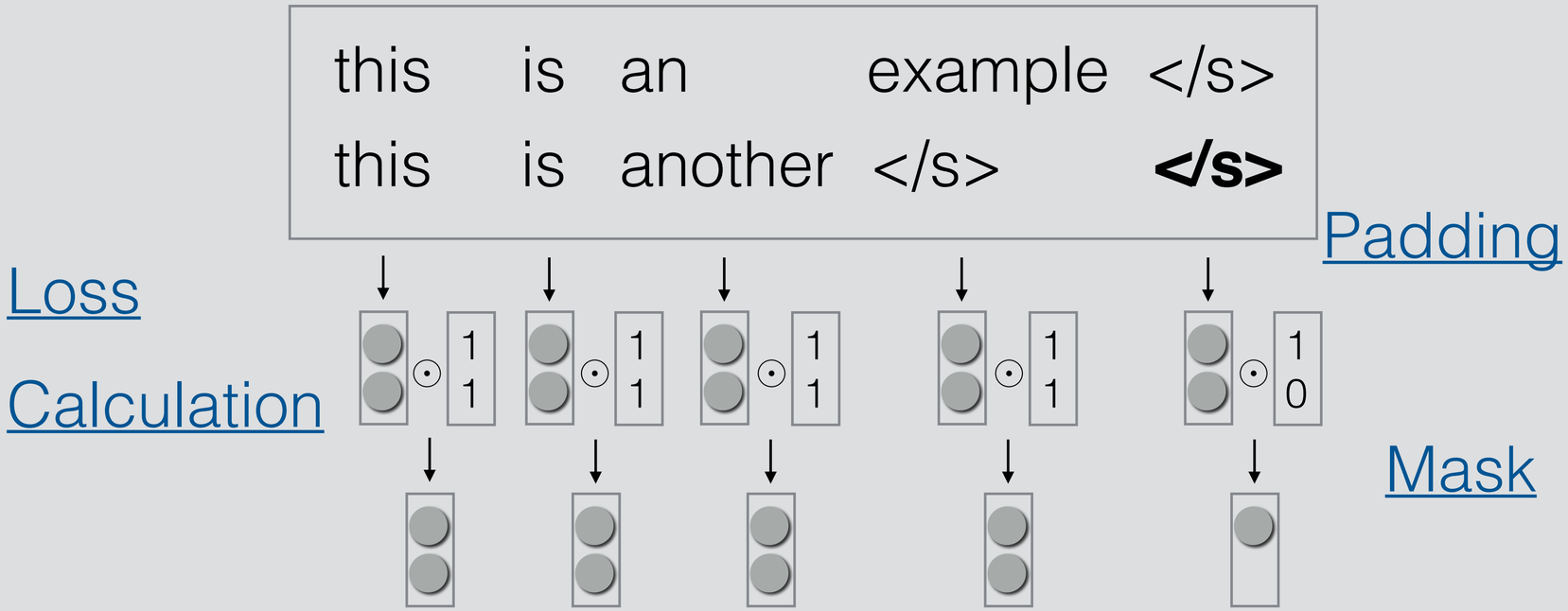
Mini-batching Method



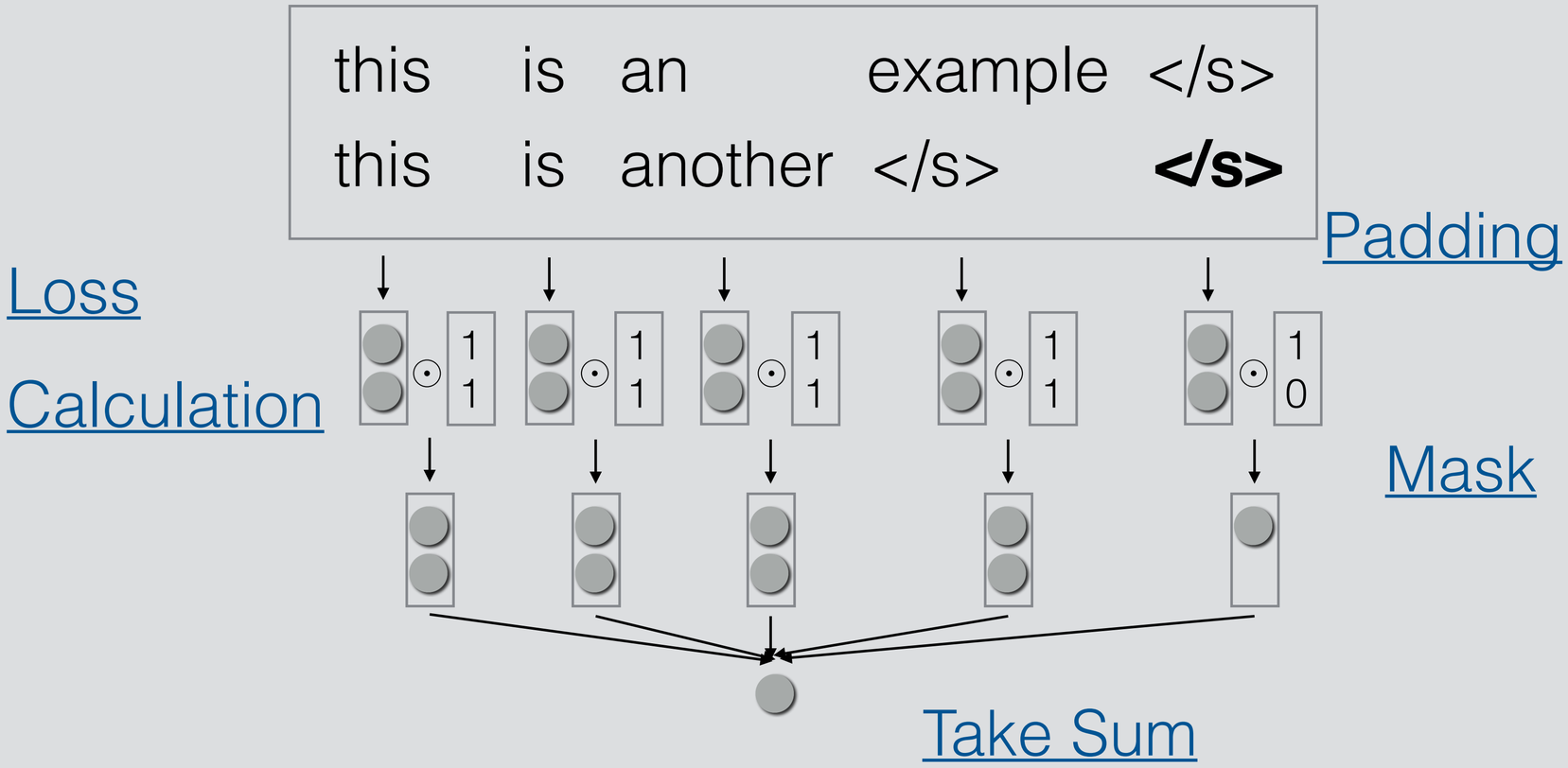
Mini-batching Method



Mini-batching Method



Mini-batching Method



Bucketing/Sorting

- If we use sentences of different lengths, too much padding and sorting can **result in decreased performance**

Bucketing/Sorting

- If we use sentences of different lengths, too much padding and sorting can **result in decreased performance**

- To remedy this: **sort sentences** so similarly-lengthed sentences are in the same batch

RNN Variants

RNN Variants

(Greffen et al. 2015)

- Gated Recurrent Units
  (GRU; Cho et al 2014)

RNN Variants

(Greffen et al. 2015)

- Gated Recurrent Units
  (GRU; Cho et al 2014)

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)
\end{aligned}
$$

Additive or Non-linear

RNN Variants

(Greffen et al. 2015)

- Gated Recurrent Units
  (GRU; Cho et al 2014)

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)
\end{aligned}
$$

Additive or Non-linear

- **Note:** GRUs cannot do things like simply count

RNN Variants

(Greffen et al. 2015)

- Gated Recurrent Units (GRU; Cho et al 2014)
- Many different types of architectures tested for LSTMs (Greffen et al. 2015)

NIG: No Input Gate: $\mathbf{i}^t = \mathbf{1}$

NFG: No Forget Gate: $\mathbf{f}^t = \mathbf{1}$

NOG: No Output Gate: $\mathbf{o}^t = \mathbf{1}$

NIAF: No Input Activation Function: $g(\mathbf{x}) = \mathbf{x}$

NOAF: No Output Activation Function: $h(\mathbf{x}) = \mathbf{x}$

CIFG: Coupled Input and Forget Gate: $\mathbf{f}^t = \mathbf{1} - \mathbf{i}^t$

NP: No Peepholes:

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{b}_i$$
$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{b}_f$$
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{b}_o$$

FGR: Full Gate Recurrence:

$$\begin{aligned}\bar{\mathbf{i}}^t = {} & \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i \\ & + \mathbf{R}_{ii} \mathbf{i}^{t-1} + \mathbf{R}_{fi} \mathbf{f}^{t-1} + \mathbf{R}_{oi} \mathbf{o}^{t-1} \\ \bar{\mathbf{f}}^t = {} & \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f \\ & + \mathbf{R}_{if} \mathbf{i}^{t-1} + \mathbf{R}_{ff} \mathbf{f}^{t-1} + \mathbf{R}_{of} \mathbf{o}^{t-1} \\ \bar{\mathbf{o}}^t = {} & \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t-1} + \mathbf{b}_o \\ & + \mathbf{R}_{io} \mathbf{i}^{t-1} + \mathbf{R}_{fo} \mathbf{f}^{t-1} + \mathbf{R}_{oo} \mathbf{o}^{t-1}\end{aligned}$$

RNN Variants

(Greffen et al. 2015)

- Gated Recurrent Units (GRU; Cho et al 2014)
- Many different types of architectures tested for LSTMs (Greffen et al. 2015)
- **Conclusion:** basic LSTM quite good, other variants (e.g. coupled input/forget gates) reasonable

NIG: No Input Gate: $\mathbf{i}^t = \mathbf{1}$
NFG: No Forget Gate: $\mathbf{f}^t = \mathbf{1}$
NOG: No Output Gate: $\mathbf{o}^t = \mathbf{1}$
NIAF: No Input Activation Function: $g(\mathbf{x}) = \mathbf{x}$
NOAF: No Output Activation Function: $h(\mathbf{x}) = \mathbf{x}$
CIFG: Coupled Input and Forget Gate: $\mathbf{f}^t = \mathbf{1} - \mathbf{i}^t$
NP: No Peepholes:

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{b}_i$$
$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{b}_f$$
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{b}_o$$

FGR: Full Gate Recurrence:

$$\begin{aligned}\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i \\ + \mathbf{R}_{ii} \mathbf{i}^{t-1} + \mathbf{R}_{fi} \mathbf{f}^{t-1} + \mathbf{R}_{oi} \mathbf{o}^{t-1} \\ \bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f \\ + \mathbf{R}_{if} \mathbf{i}^{t-1} + \mathbf{R}_{ff} \mathbf{f}^{t-1} + \mathbf{R}_{of} \mathbf{o}^{t-1} \\ \bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t-1} + \mathbf{b}_o \\ + \mathbf{R}_{io} \mathbf{i}^{t-1} + \mathbf{R}_{fo} \mathbf{f}^{t-1} + \mathbf{R}_{oo} \mathbf{o}^{t-1}\end{aligned}$$

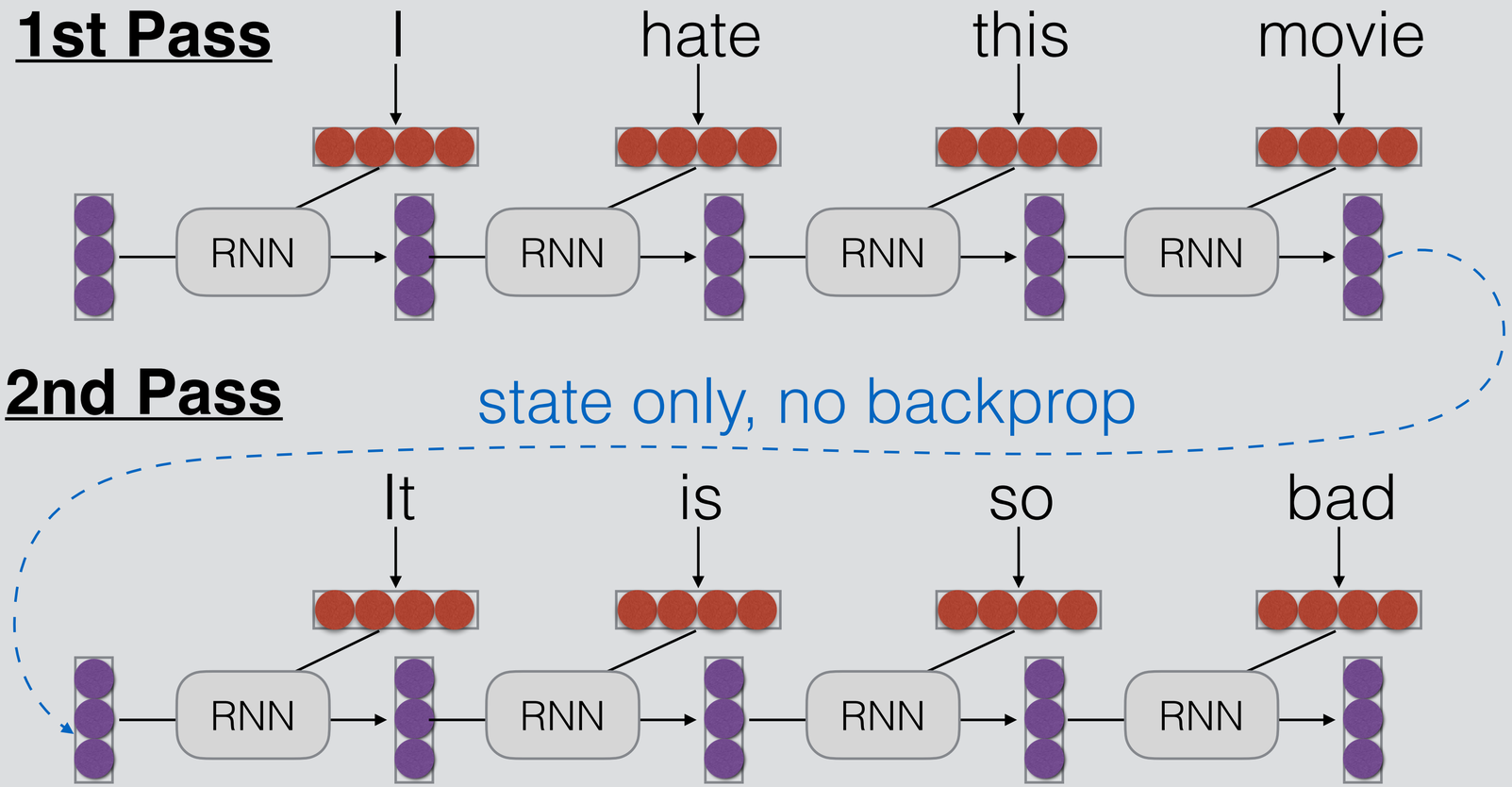
Handling Long Sequences

Handling Long Sequences

- Sometimes we would like to capture long-term dependencies over long sequences
- e.g. words in full documents
- However, this may not fit on (GPU) memory

Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment



Outline

- Language Models
- Smoothing
- Recurrent Neural Models

Questions?
(see extra slides)