

CS11-711 Advanced NLP

Sequence Modeling

Graham Neubig



Carnegie Mellon University

Language Technologies Institute

Site

<https://phontron.com/class/anlp2024/>

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**.

- 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**.

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>)

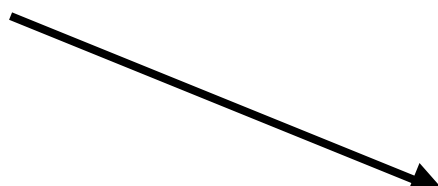
Types of Sequential Prediction Problems

Types of Prediction: Binary, Multi-class, Structured

- Two classes (**binary classification**)

I hate this movie  positive
negative

- Multiple classes (**multi-class classification**)

I hate this movie  very good
good
neutral
bad
very bad

- Exponential/infinite labels (**structured prediction**)

I hate this movie  PRP VBP DT NN

I hate this movie  *kono eiga ga kirai*

Types of Prediction: Unconditioned vs. Conditioned

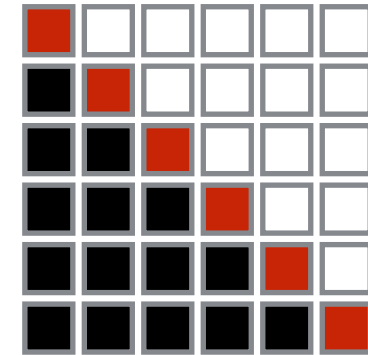
- **Unconditioned Prediction:** Predict the probability of a single variable $P(X)$
- **Conditioned Prediction:** Predict the probability of an output variable given an input $P(Y|X)$

Types of Unconditioned Prediction

Left-to-right Autoregressive Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \dots, x_{i-1})$$

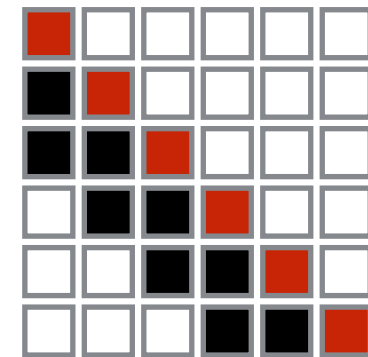
(e.g. RNN or Transformer LM)



Left-to-right Markov Chain (order n-1)

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_{i-n+1}, \dots, x_{i-1})$$

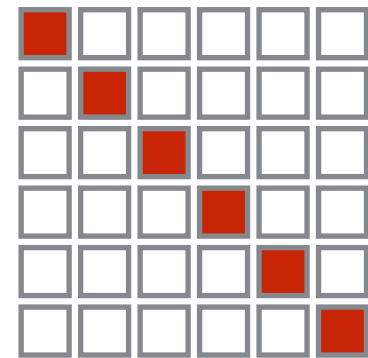
(e.g. n-gram LM, feed-forward LM)



Independent Prediction

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

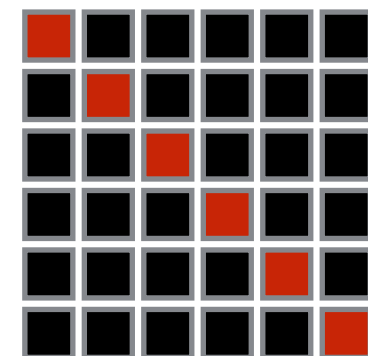
(e.g. unigram model)



Bidirectional Prediction

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i | x_{\neq i})$$

(e.g. masked language model)



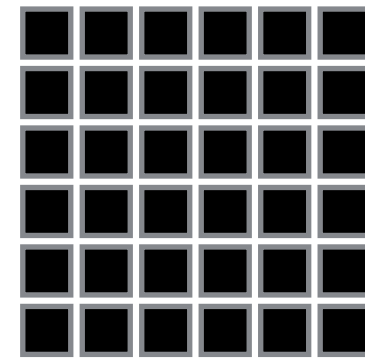
Types of Conditioned Prediction

Autoregressive Conditioned Prediction

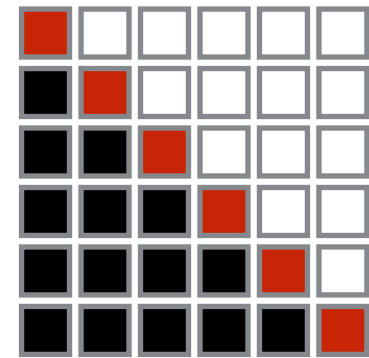
$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X, y_1, \dots, y_{i-1})$$

(e.g. seq2seq model)

Source X



Target Y

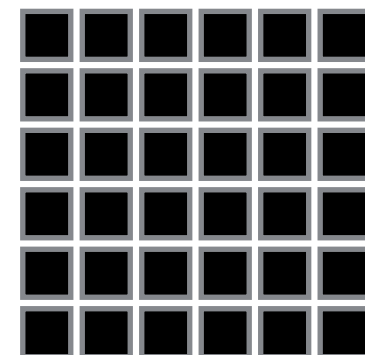


Non-autoregressive Conditioned Prediction

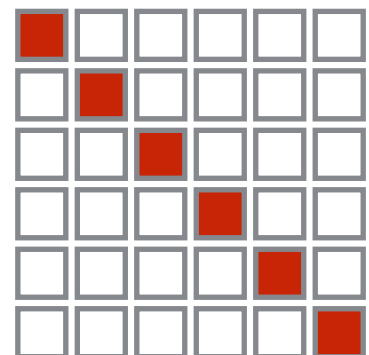
$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X)$$

(e.g. sequence labeling, non-autoregressive MT)

Source X



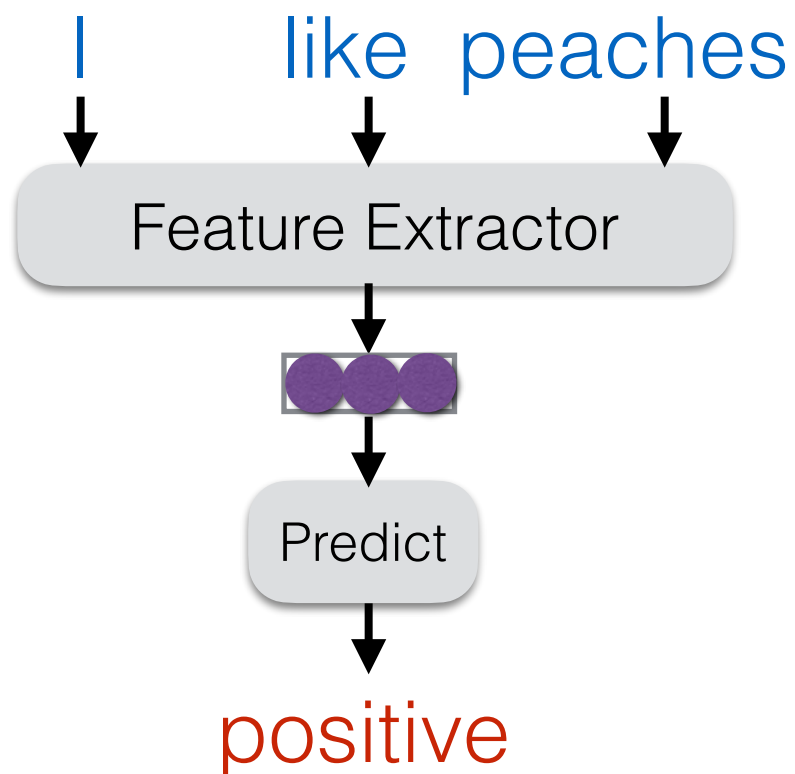
Target Y



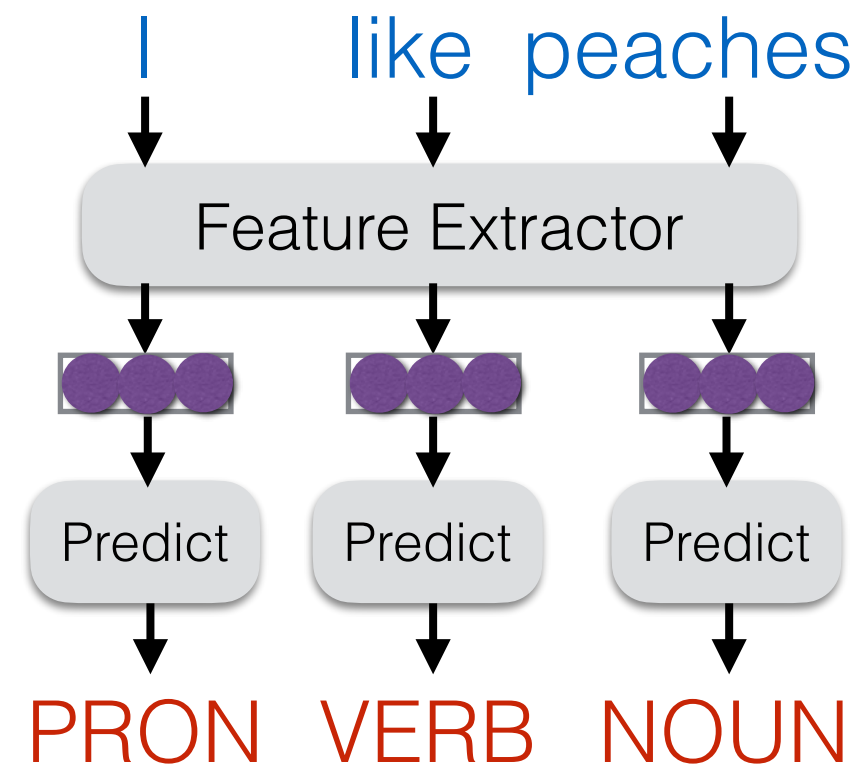
Basic Modeling Paradigm: Extract Features -> Predict

- Given an input text X
- Extract features H
- Predict labels Y

Text Classification



Sequence Labeling



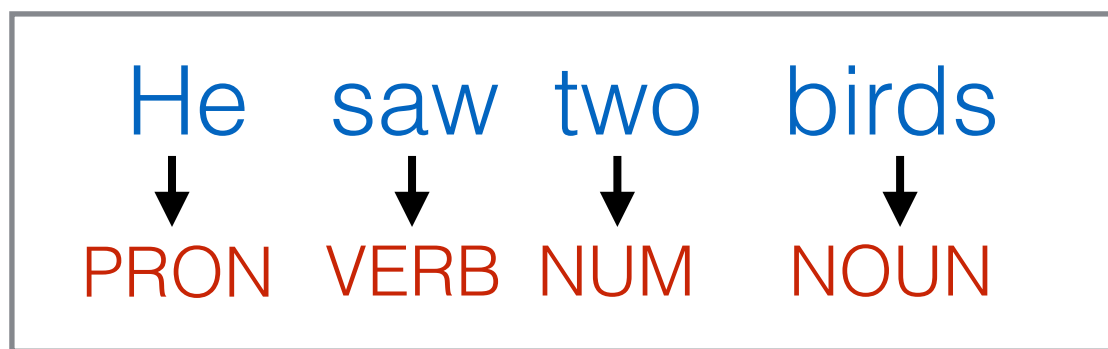
An Aside:

More on Sequence Labeling

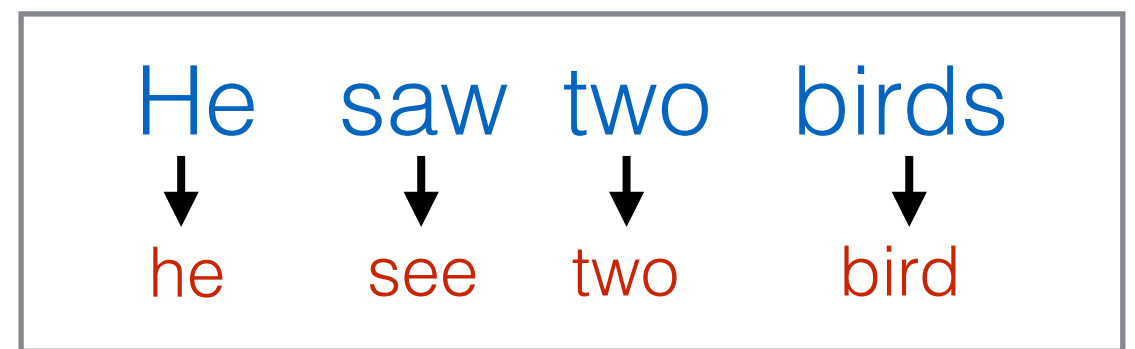
Sequence Labeling

- Given an input text X , predict an output label sequence Y of equal length!

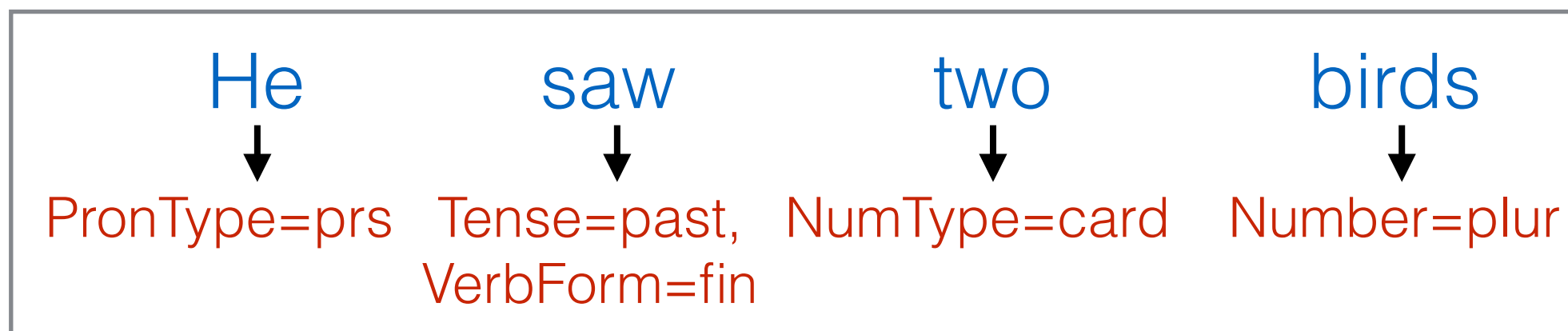
Part of Speech Tagging



Lemmatization



Morphological Tagging



... and more!

Span Labeling

- Given an input text X , predict an output spans and labels Y .

Named Entity Recognition

Graham Neubig is teaching at Carnegie Mellon University
PER ORG

Syntactic Chunking

Graham Neubig is teaching at Carnegie Mellon University
NP VP NP

Semantic Role Labeling

Graham Neubig is teaching at Carnegie Mellon University
Actor Predicate Location

... and more!

Span Labeling as Sequence Labeling

- Predict **B**eginning, **I**n, and **O**ut tags for each word in a span

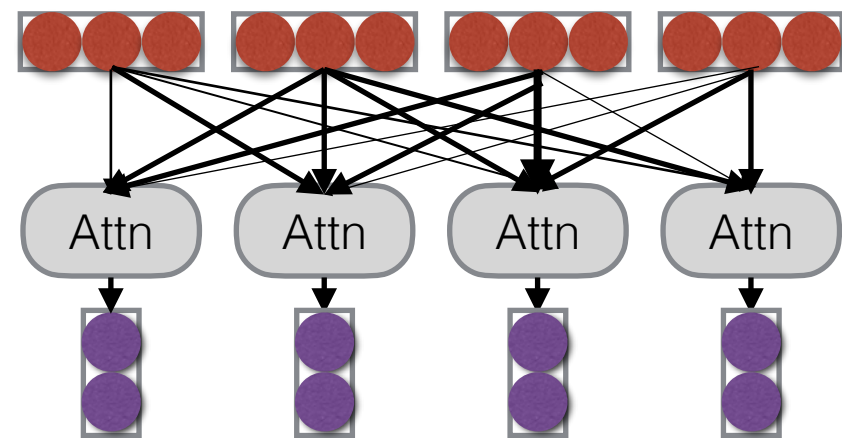
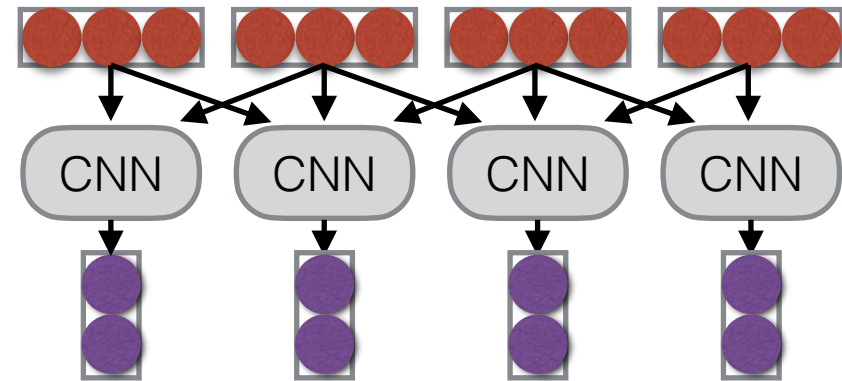
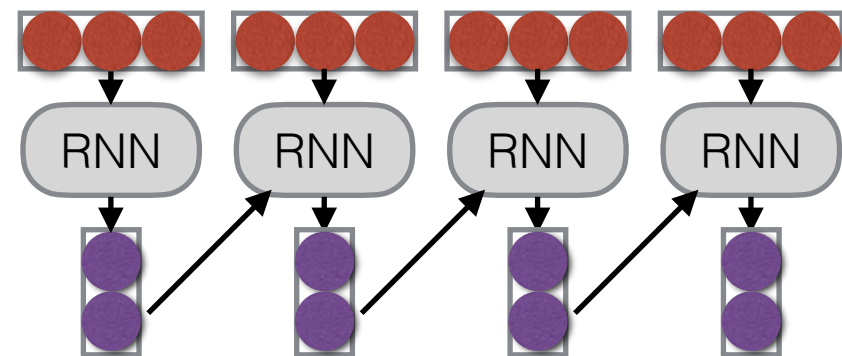
Graham Neubig is teaching at Carnegie Mellon University
PER ORG

Graham Neubig is teaching at Carnegie Mellon University
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
B-PER I-PER O O O B-ORG I-ORG I-ORG

Types of Sequence Models

Three Major Types of Sequence Models

- **Recurrence:** Condition representations on an encoding of the history
- **Convolution:** Condition representations on local context
- **Attention:** Condition representations on a weighted average of all tokens



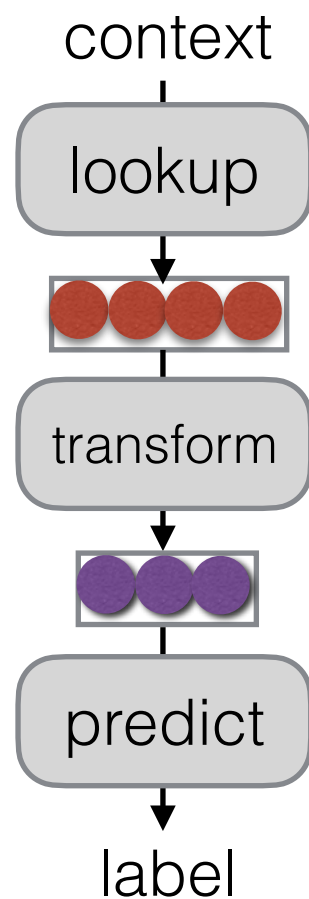
A Sequence Model: Recurrent Neural Networks

Recurrent Neural Networks

(Elman 1990)

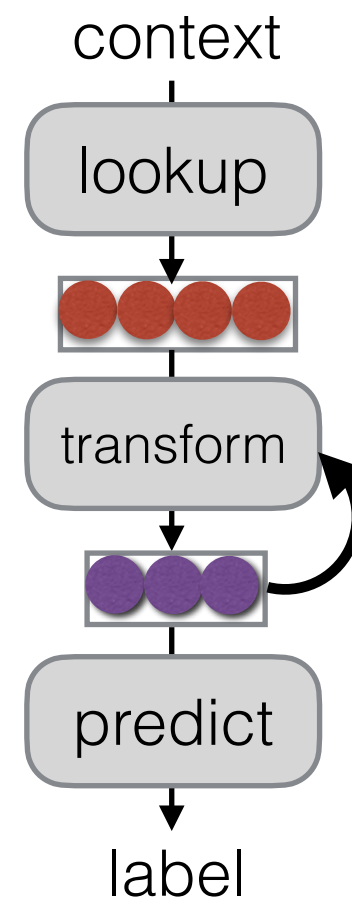
- Tools to “remember” information

Feed-forward NN



$$h_t = f(W_x x_t + b)$$

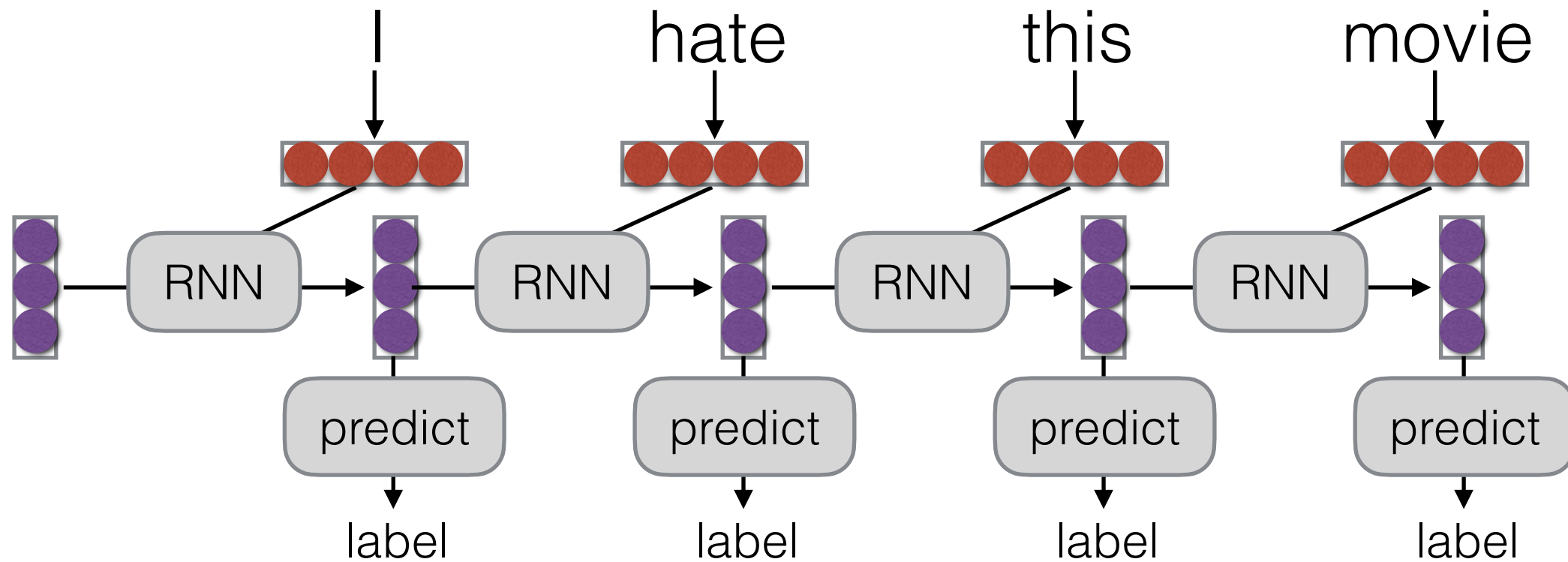
Recurrent NN



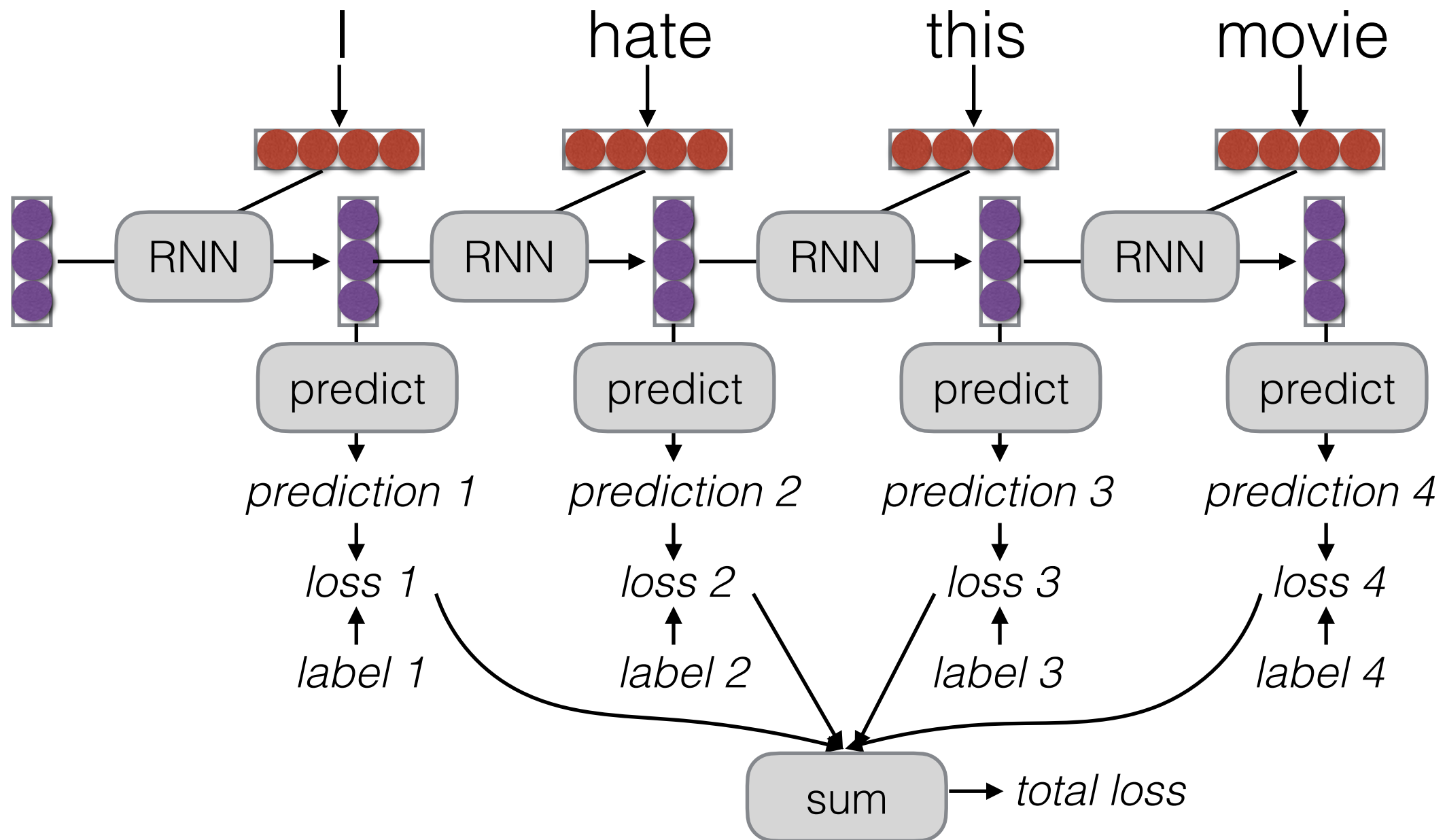
$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$

Unrolling in Time

- What does processing a sequence look like?

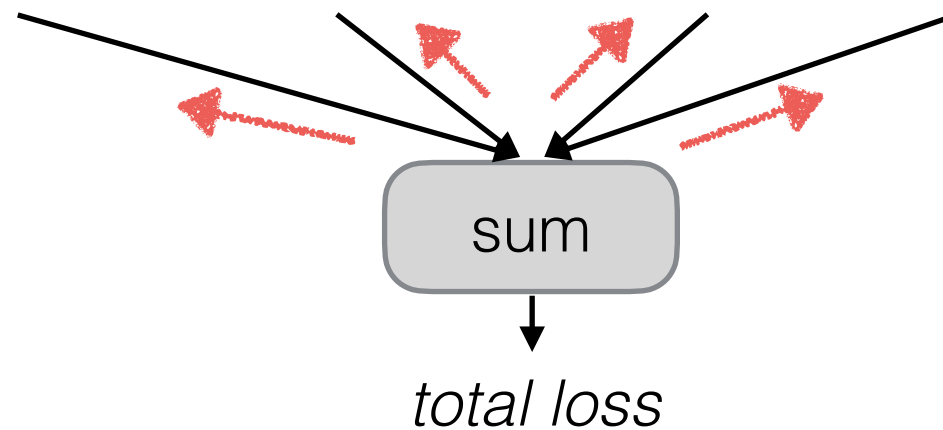


Training RNNs



RNN Training

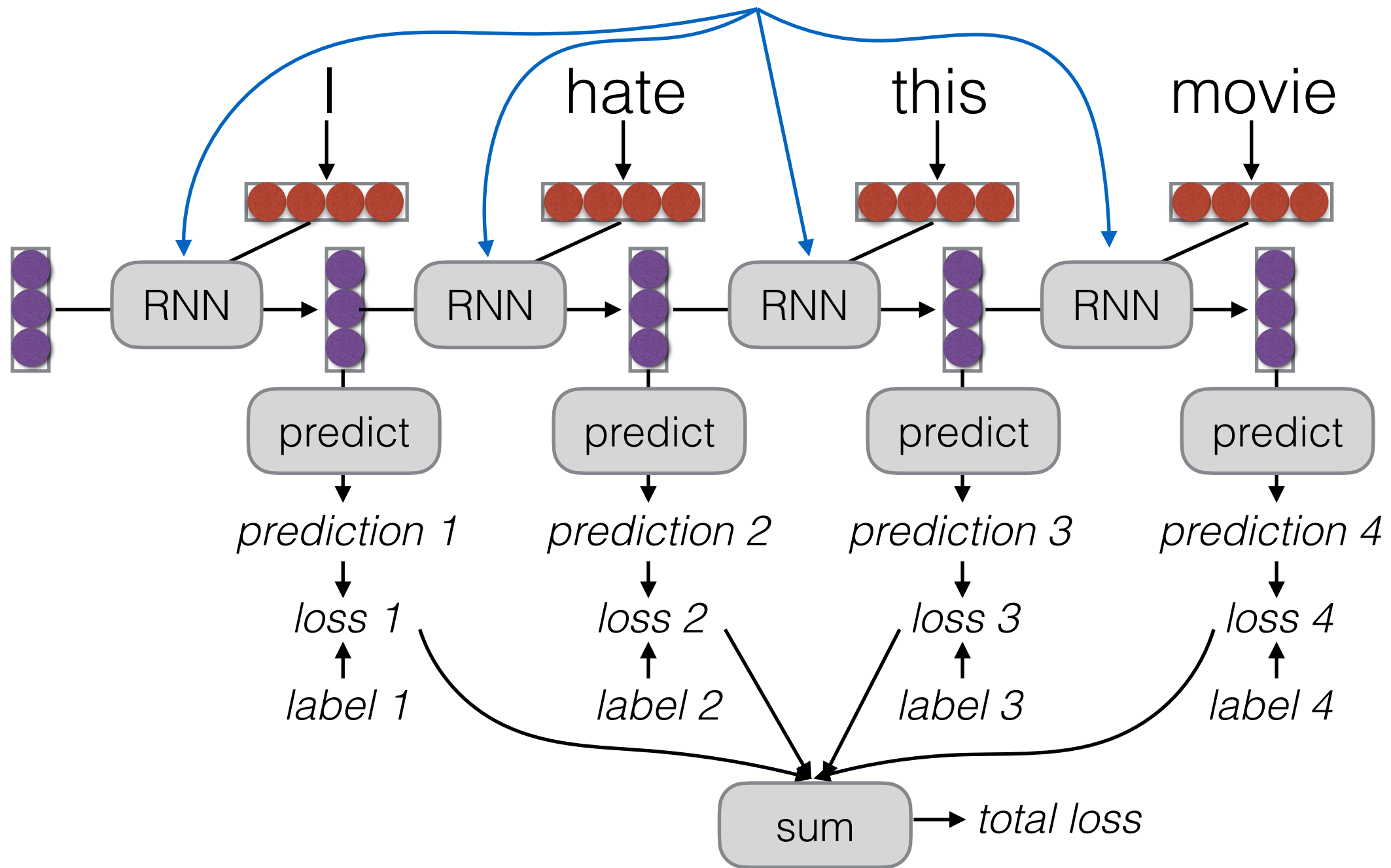
- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop



- Parameters are tied across time, derivatives are aggregated across all time steps
- This is historically called “backpropagation through time” (BPTT)

Parameter Tying

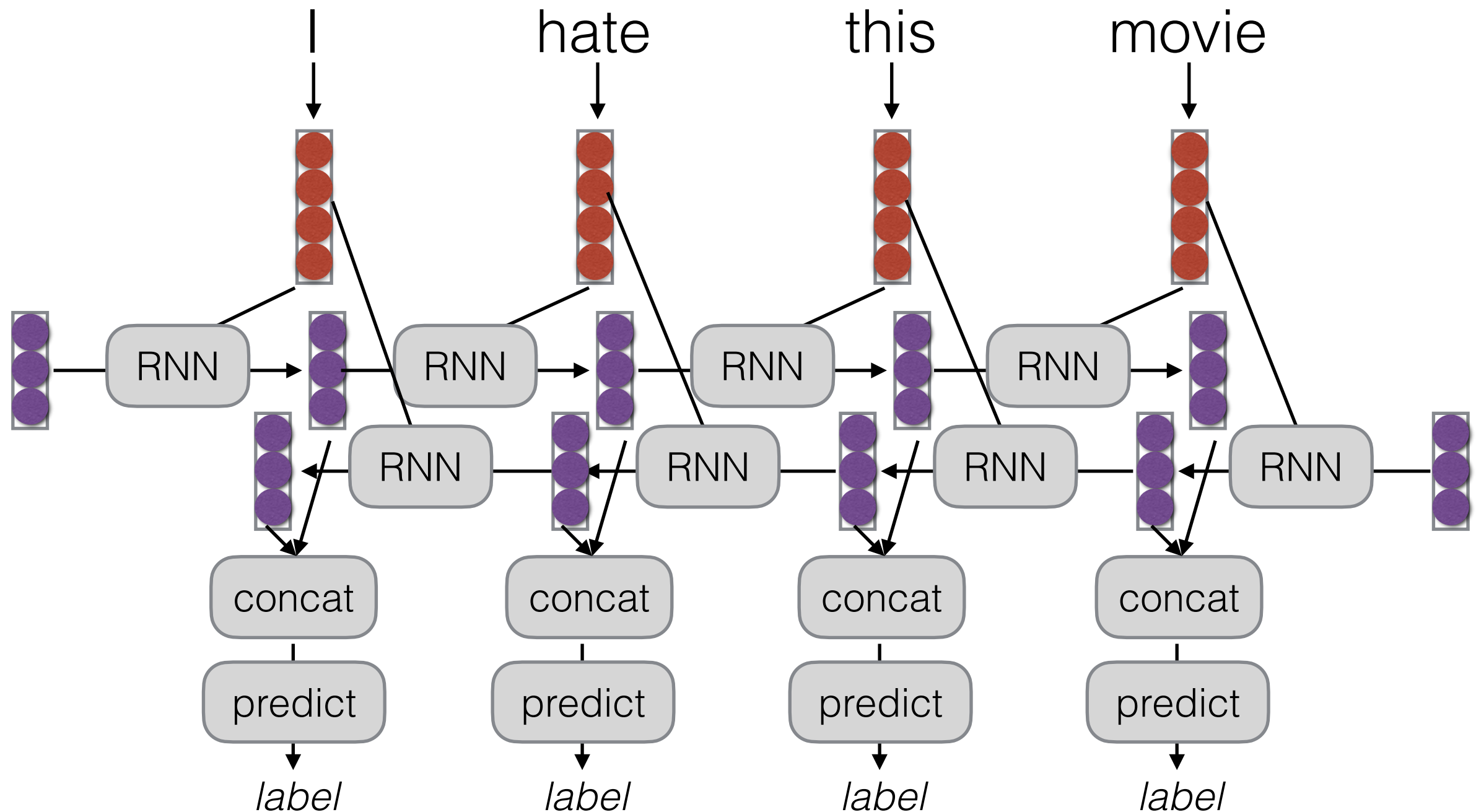
Parameters are shared! Derivatives are accumulated.



(Same for attention, convolutional networks)

Bi-RNNs

- A simple extension, run the RNN in both directions

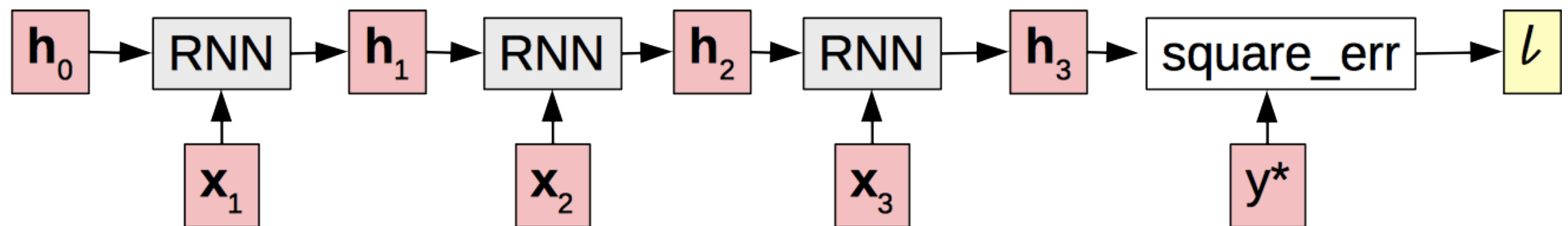


Vanishing Gradients

Vanishing Gradient

- Gradients decrease as they get pushed back

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



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

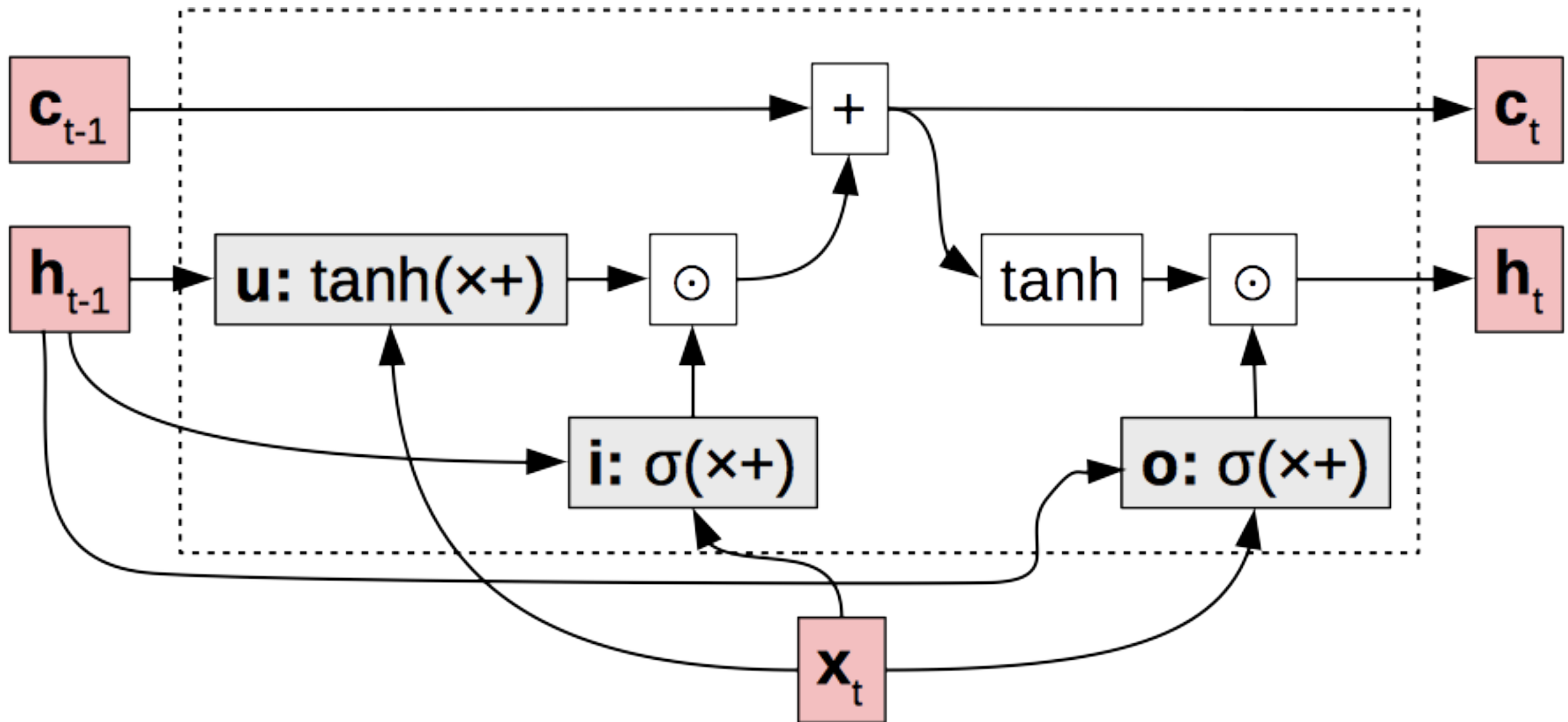
A Solution:

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

LSTM Structure

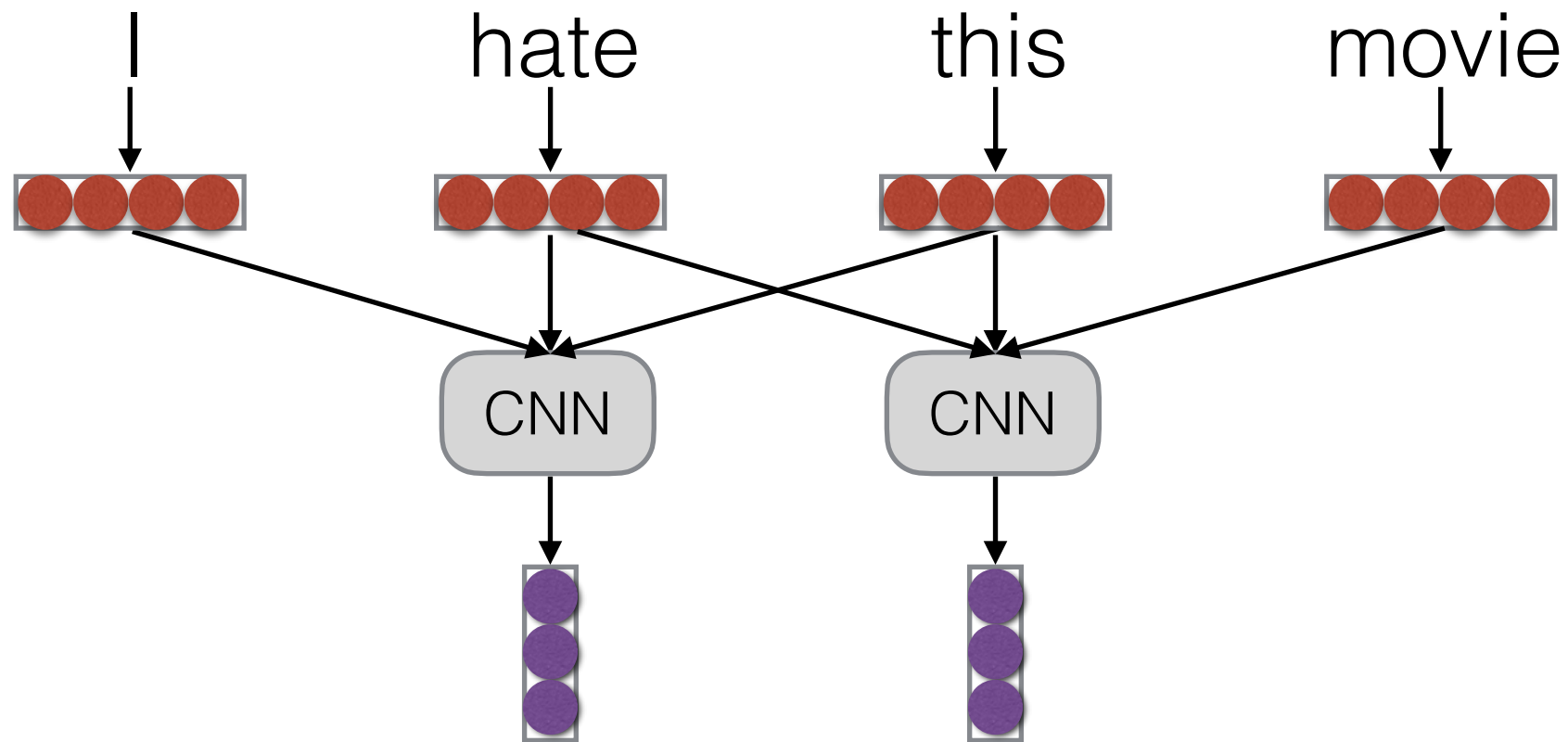


update u : what value do we try to add to the memory cell?
input i : how much of the update do we allow to go through?
output o : how much of the cell do we reflect in the next state?

Convolution

Convolution

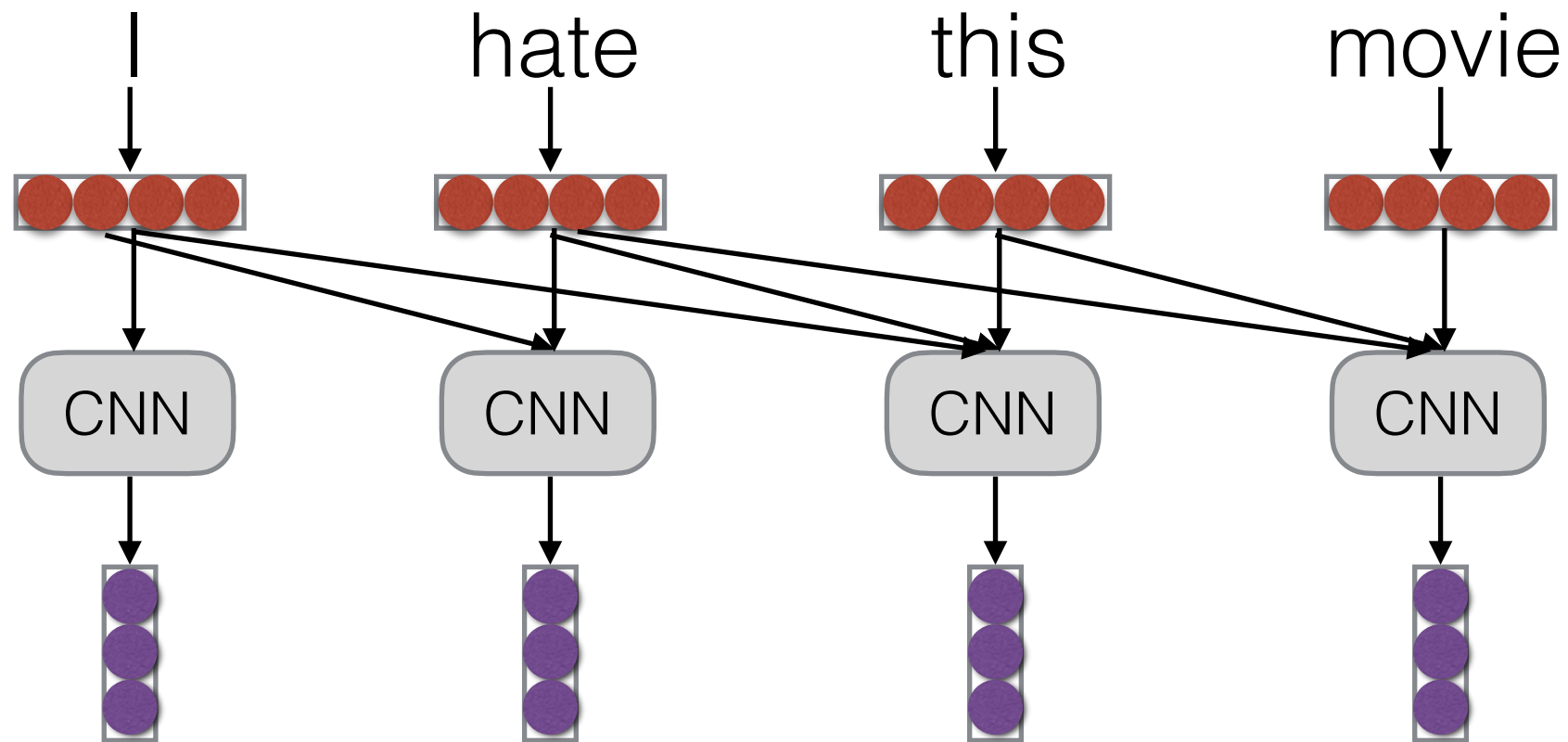
- Calculate based on local context



$$h_t = f(W[x_{t-1}; x_t; x_{t+1}])$$

Convolution for Auto-regressive Models

- Functionally identical, just consider previous context



Attention

Basic Idea

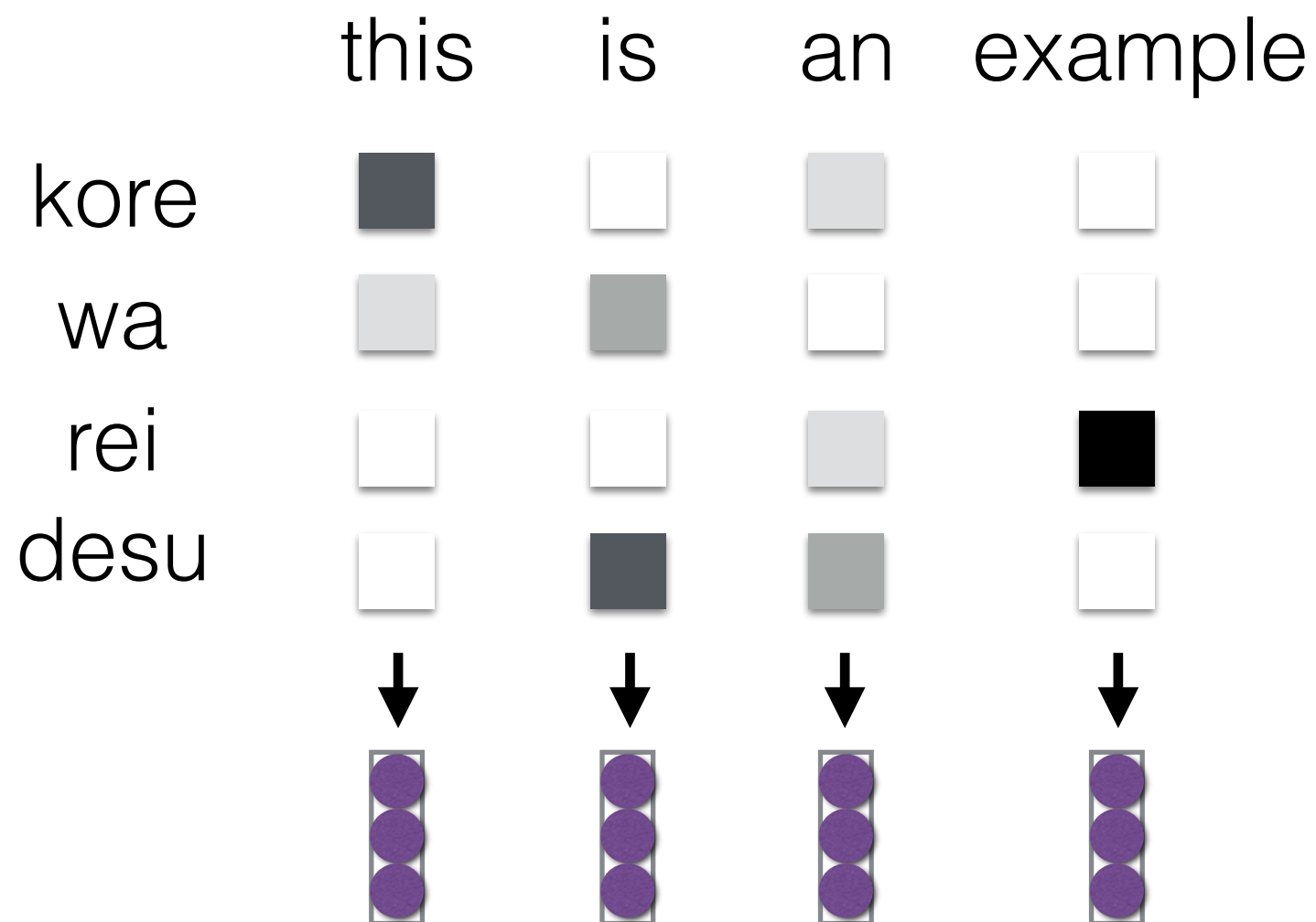
(Bahdanau et al. 2015)

- Encode each token in the sequence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”

Cross Attention

(Bahdanau et al. 2015)

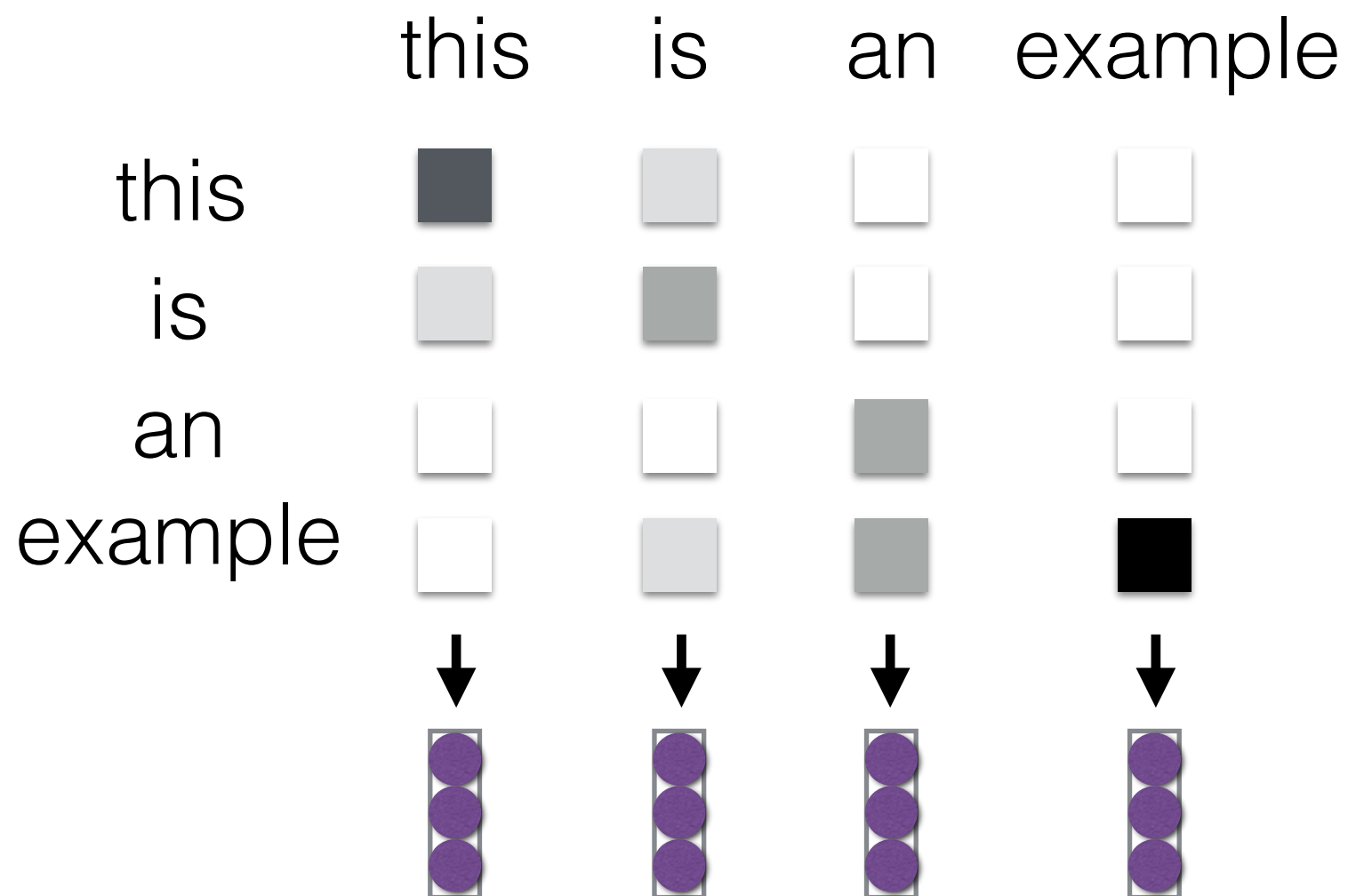
- Each element in a sequence attends to elements of another sequence



Self Attention

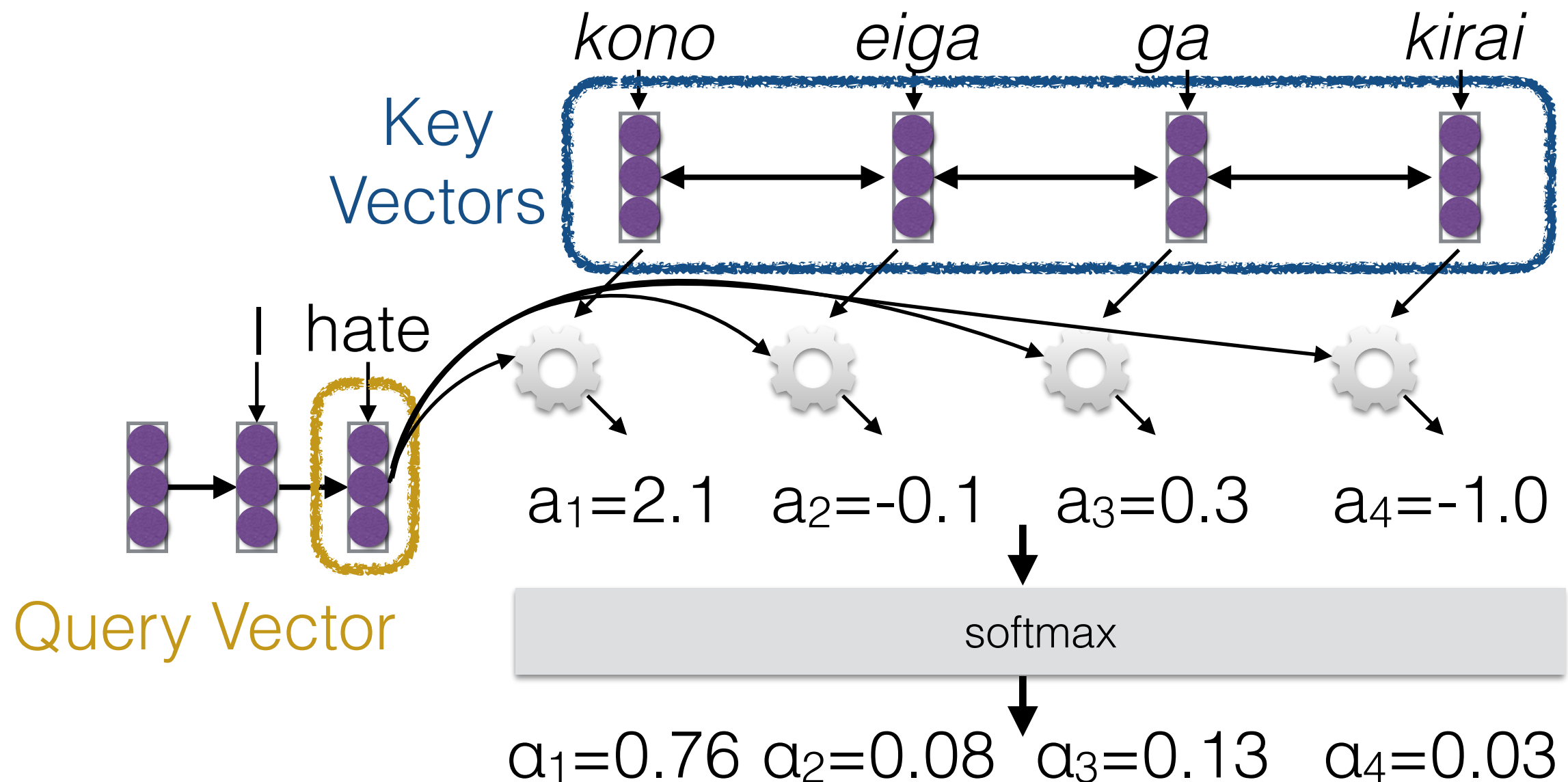
(Cheng et al. 2016, Vaswani et al. 2017)

- Each element in the sequence attends to elements of that sequence → context sensitive encodings!



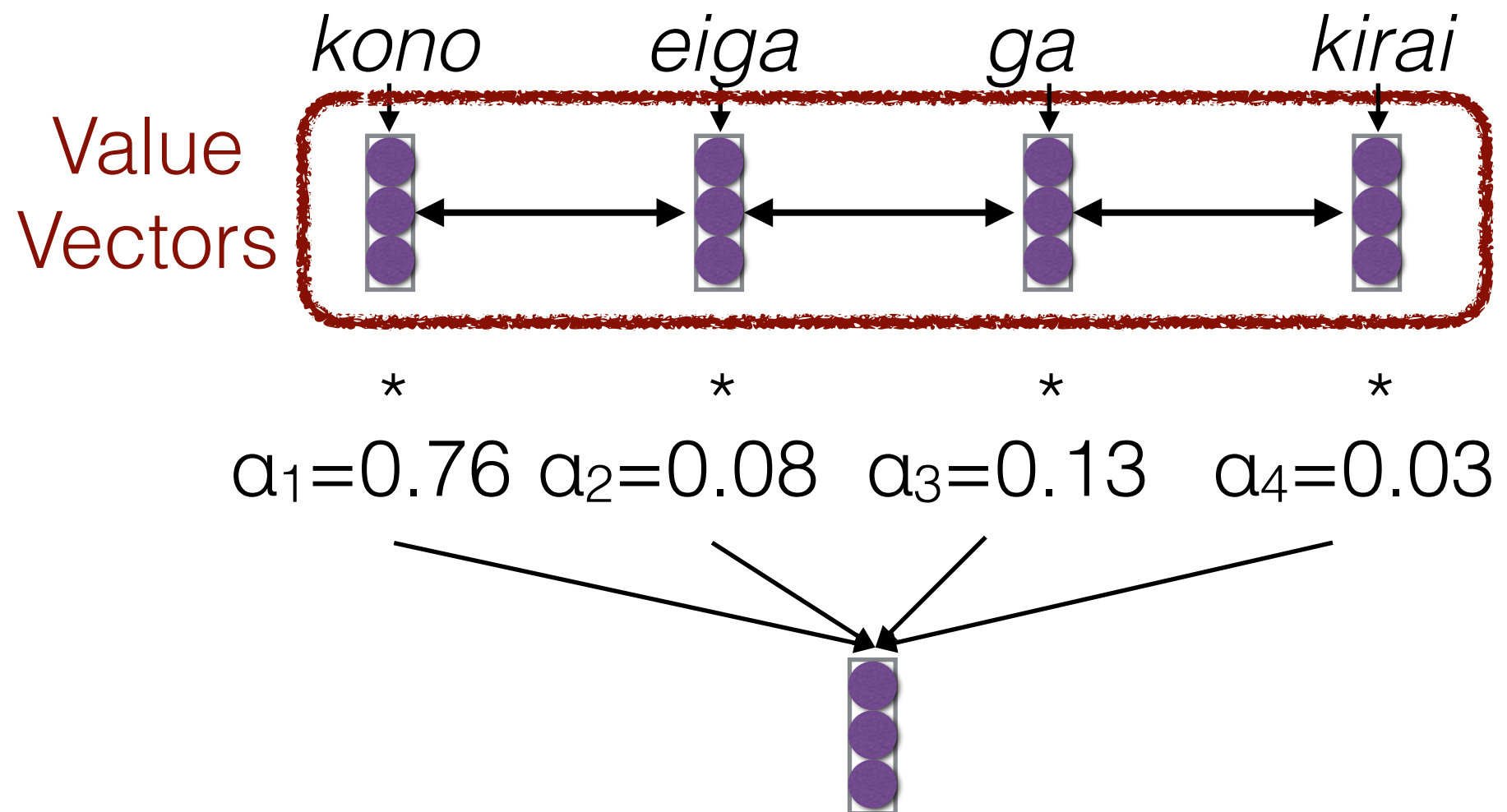
Calculating Attention (1)

- Use “query” vector (decoder state) and “key” vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



Calculating Attention (2)

- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

A Graphical Example

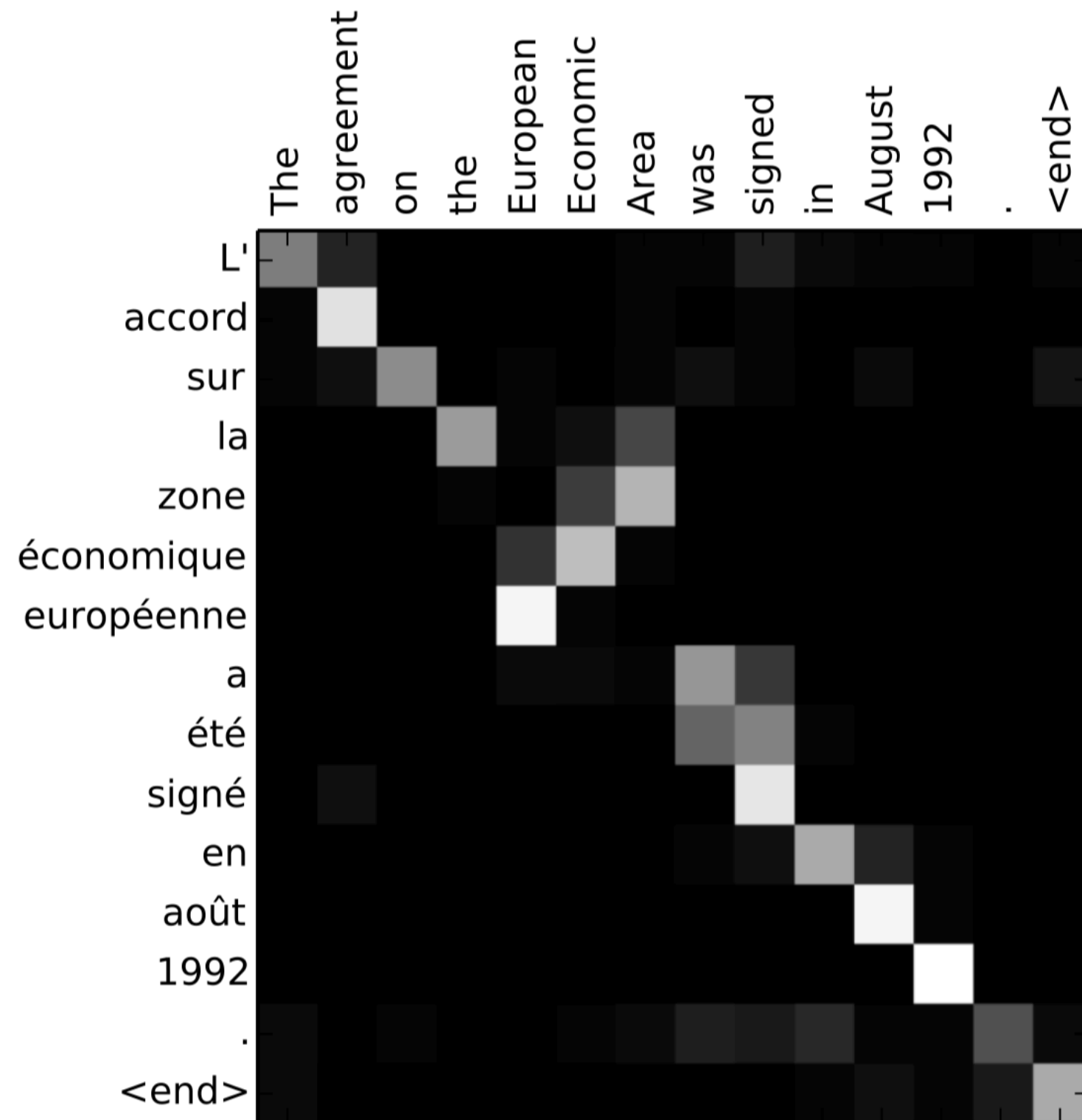


Image from Bahdanau et al. (2015)

Attention Score Functions (1)

- \mathbf{q} is the query and \mathbf{k} is the key
- **Multi-layer Perceptron** (Bahdanau et al. 2015)

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_2^\top \tanh(W_1[\mathbf{q}; \mathbf{k}])$$

- Flexible, often very good with large data
- **Bilinear** (Luong et al. 2015)

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top W \mathbf{k}$$

Attention Score Functions (2)

- **Dot Product** (Luong et al. 2015)

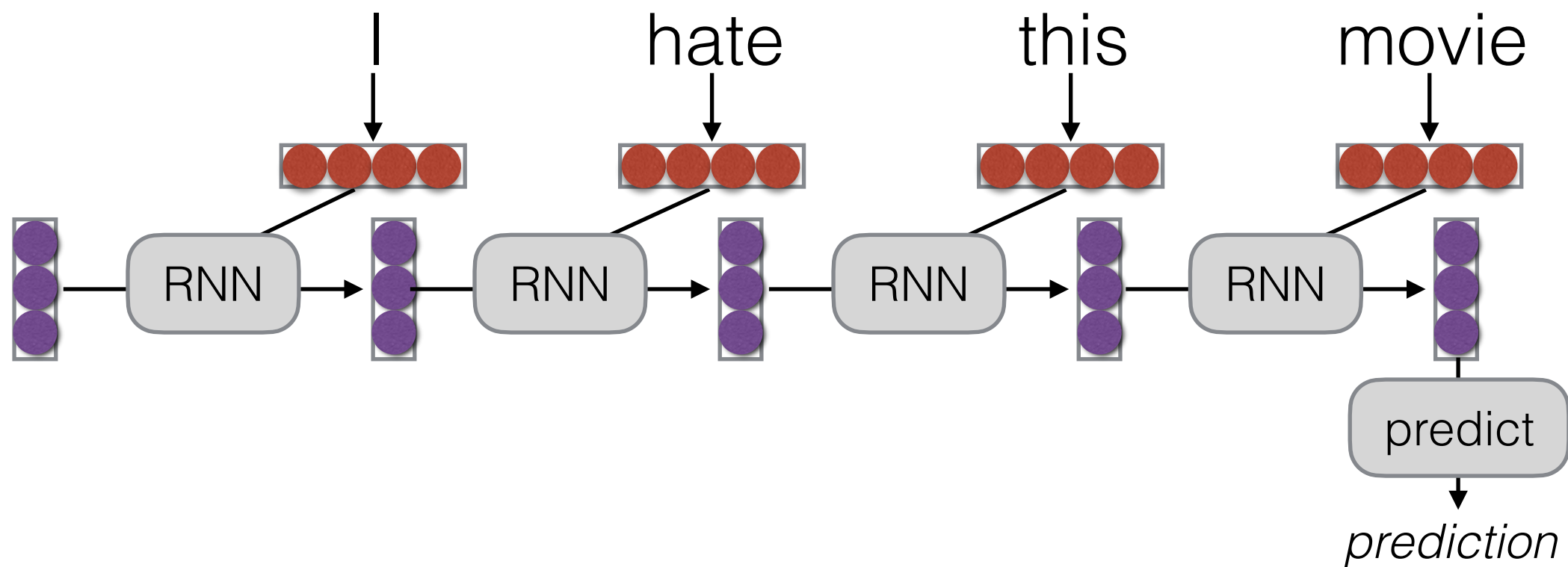
$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k}$$

- No parameters! But requires sizes to be the same.
- **Scaled Dot Product** (Vaswani et al. 2017)
 - *Problem:* scale of dot product increases as dimensions get larger
 - *Fix:* scale by size of the vector

$$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^\top \mathbf{k}}{\sqrt{|\mathbf{k}|}}$$

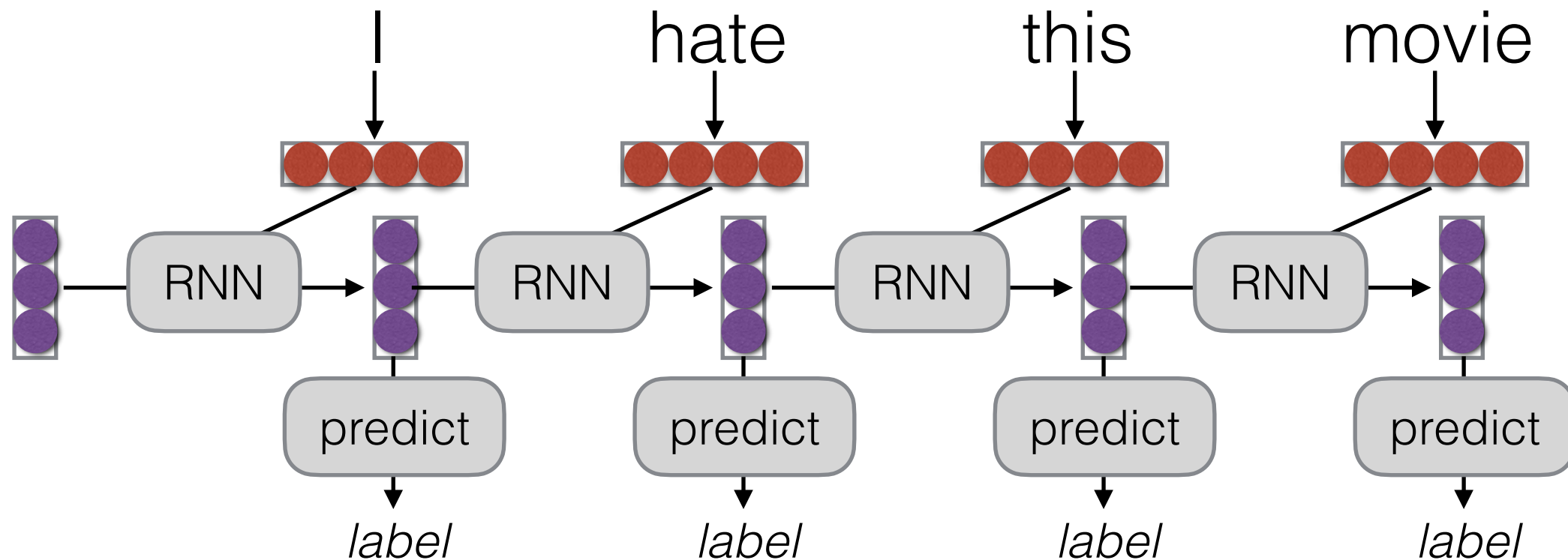
Applications of Sequence Models

Encoding Sequences



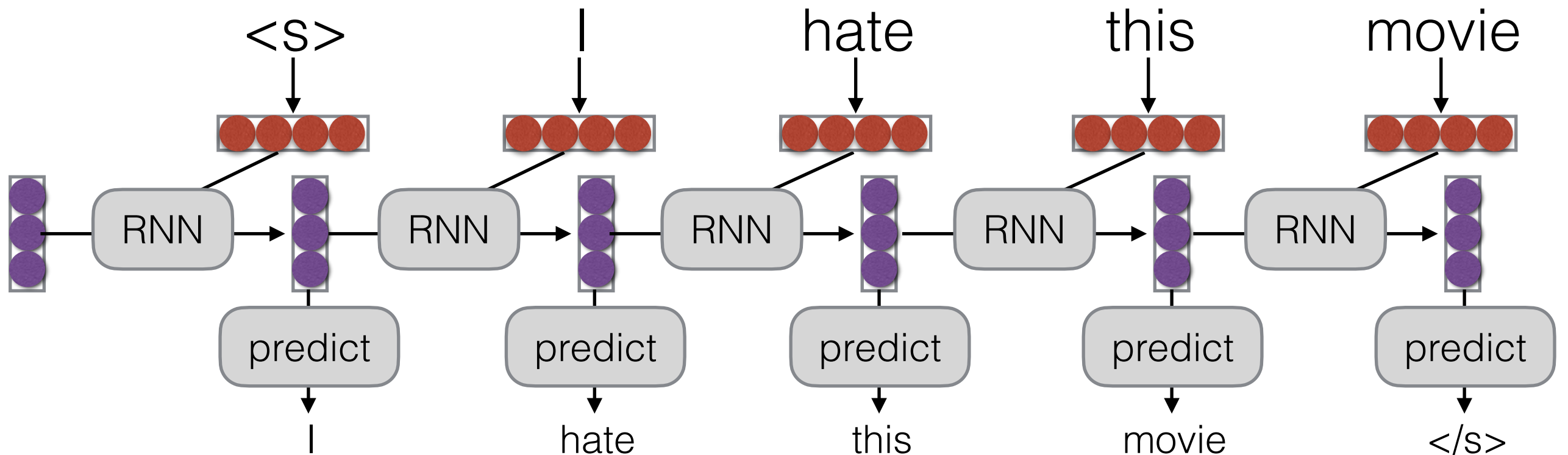
- Binary or multi-class prediction
- Sentence representation for retrieval, sentence comparison, etc.

Encoding Tokens



- Sequence labeling
- Language Modeling

e.g. Language Modeling



- Language modeling is like a tagging task, where each tag is the next word!
- Note: this is an autoregressive model

Efficiency Tricks for Sequence Modeling

Handling Mini-batching

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

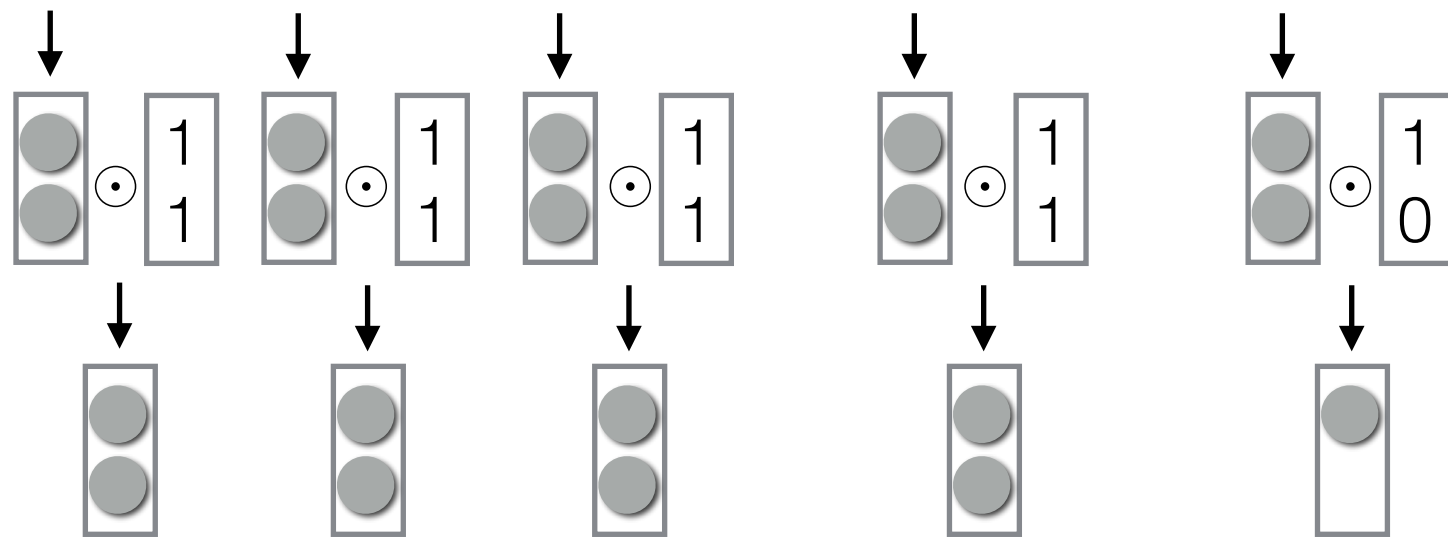
Mini-batching Method

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

Padding

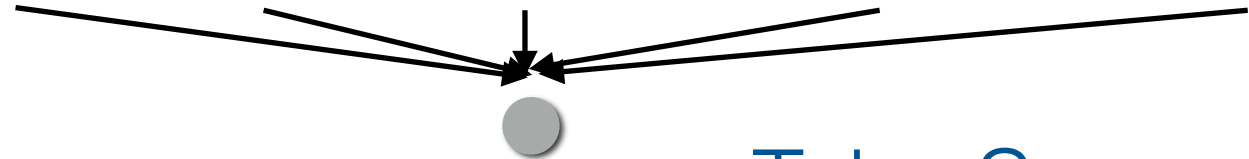
Loss

Calculation



Mask

Take Sum



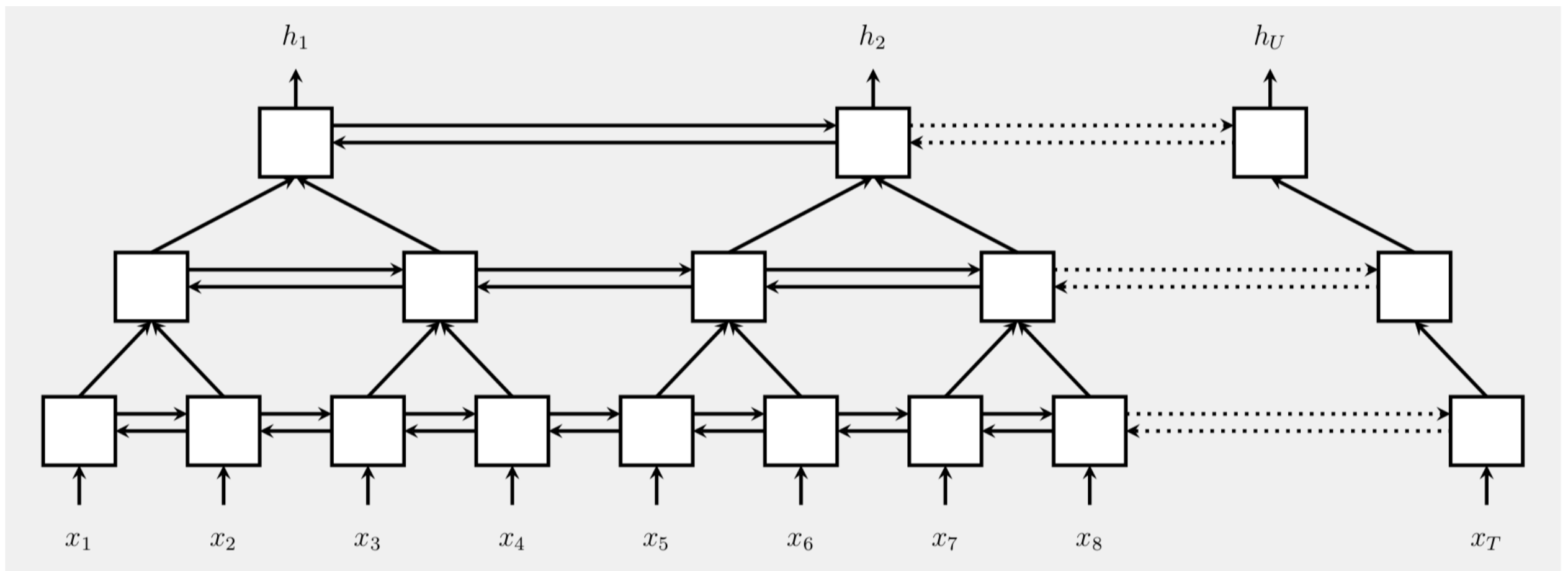
Bucketing/Sorting

- 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

Strided Architectures

(e.g. Chan et al. 2015)

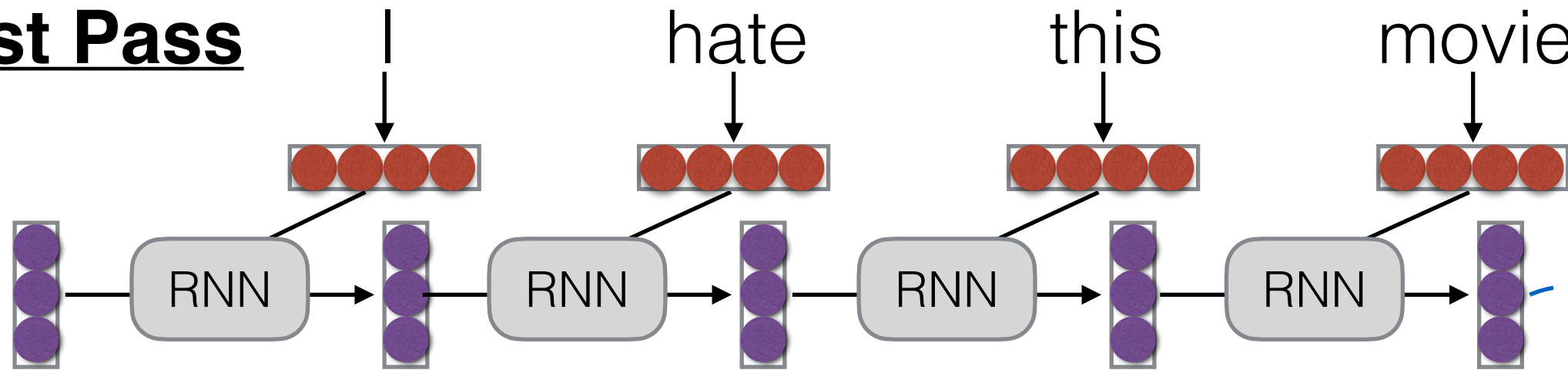
- Downscale between layers



Truncated BPTT

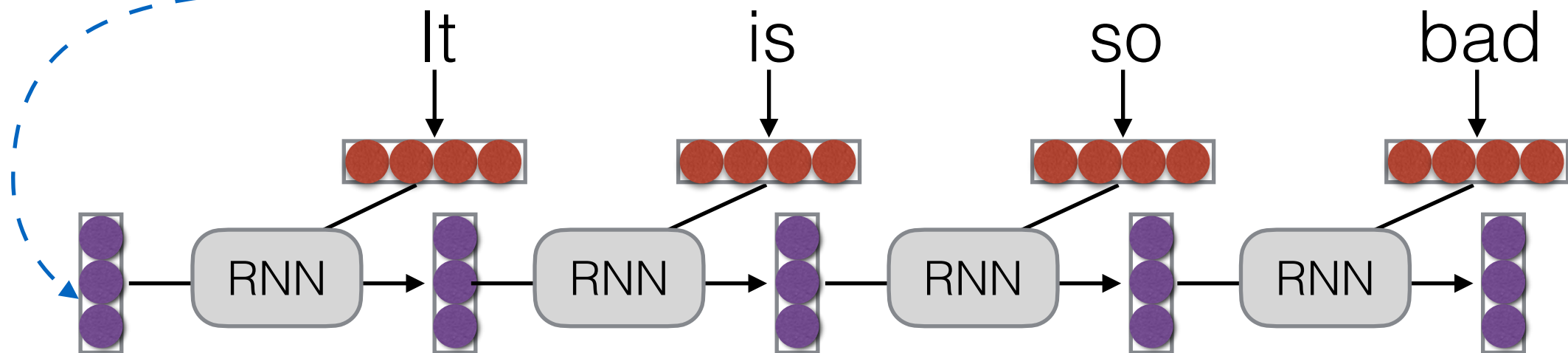
- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass



2nd Pass

state only, no backprop



Questions?