

CS11-711 Advanced NLP

Sequence Modeling and Recurrent Neural Networks

Daniel Fried and Robert Frederking
with slides from Graham Neubig



Carnegie Mellon University
Language Technologies Institute

Site

<https://cmu-anlp.github.io/>

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>)

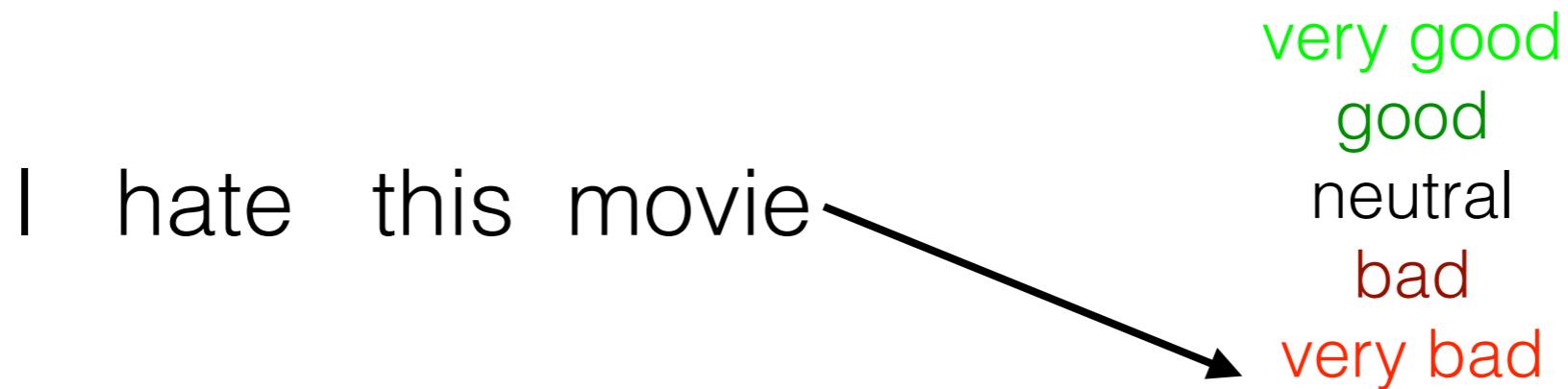
Types of Sequential Prediction Problems

Types of Prediction: Binary, Multi-class, Structured

- Two classes (**binary classification**)



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



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

I hate this movie → PRP VBP DT NN

I hate this movie → *kono eiga ga kirai*

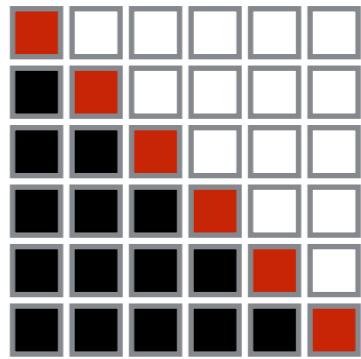
Types of Prediction: Unconditioned vs. Conditioned

- **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

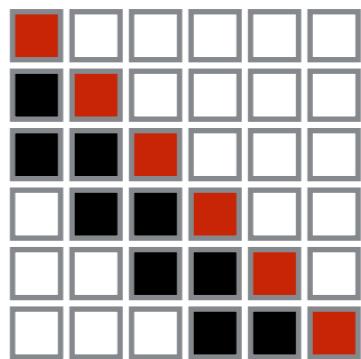
Left-to-right Autoregressive Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \dots, x_{i-1}) \quad (\text{e.g. RNN LM})$$



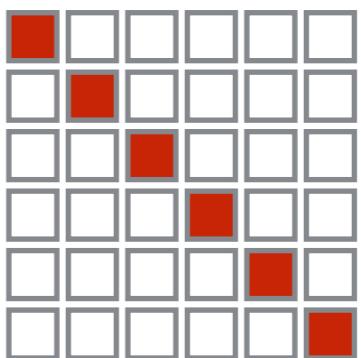
Left-to-right Markov Chain (order n-1)

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_{i-n+1}, \dots, x_{i-1}) \quad (\text{e.g. } n\text{-gram LM, feed-forward LM})$$



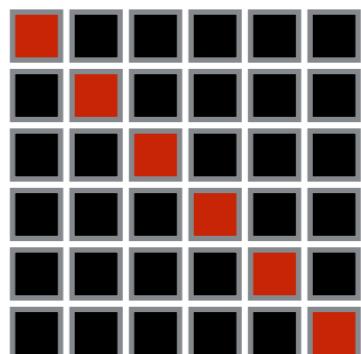
Independent Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i) \quad (\text{e.g. unigram model})$$



Bidirectional Prediction

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i | x_{\neq i}) \quad (\text{e.g. masked language model})$$



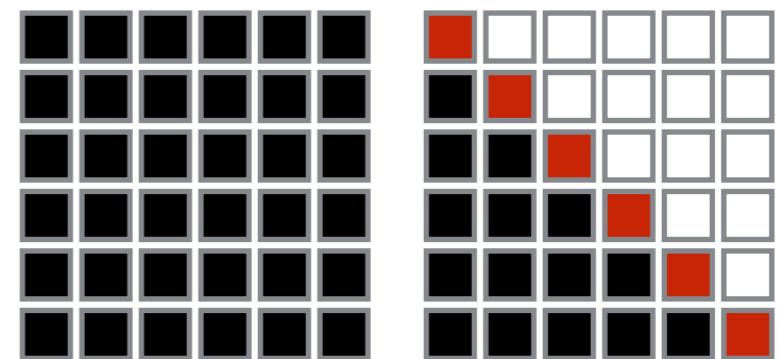
Types of Conditioned Prediction

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*

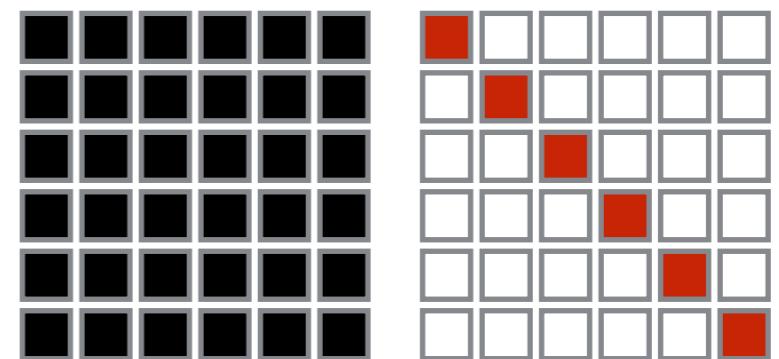


Non-autoregressive Conditioned Prediction

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

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

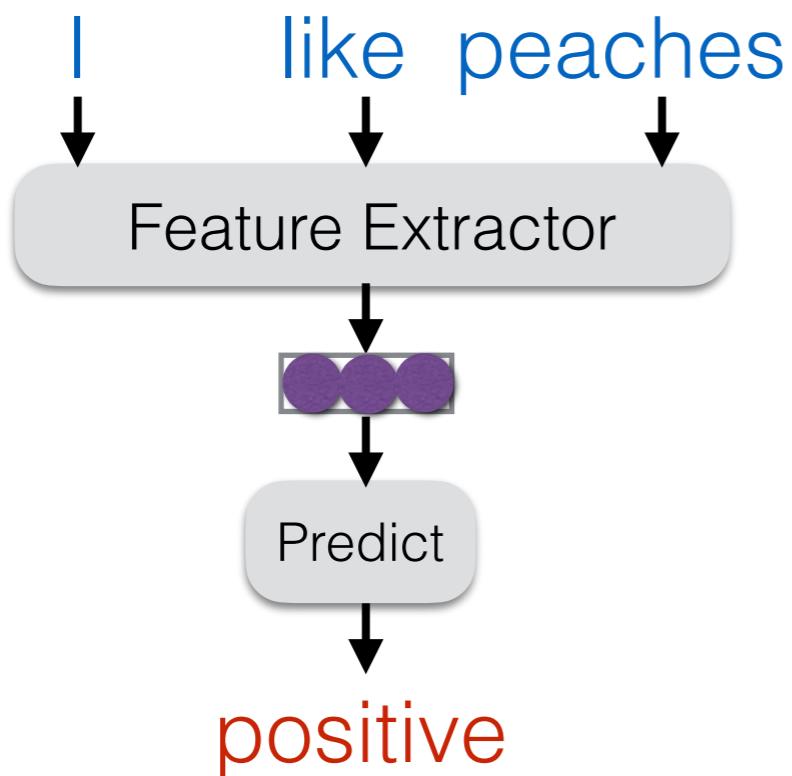
Source X *Target Y*



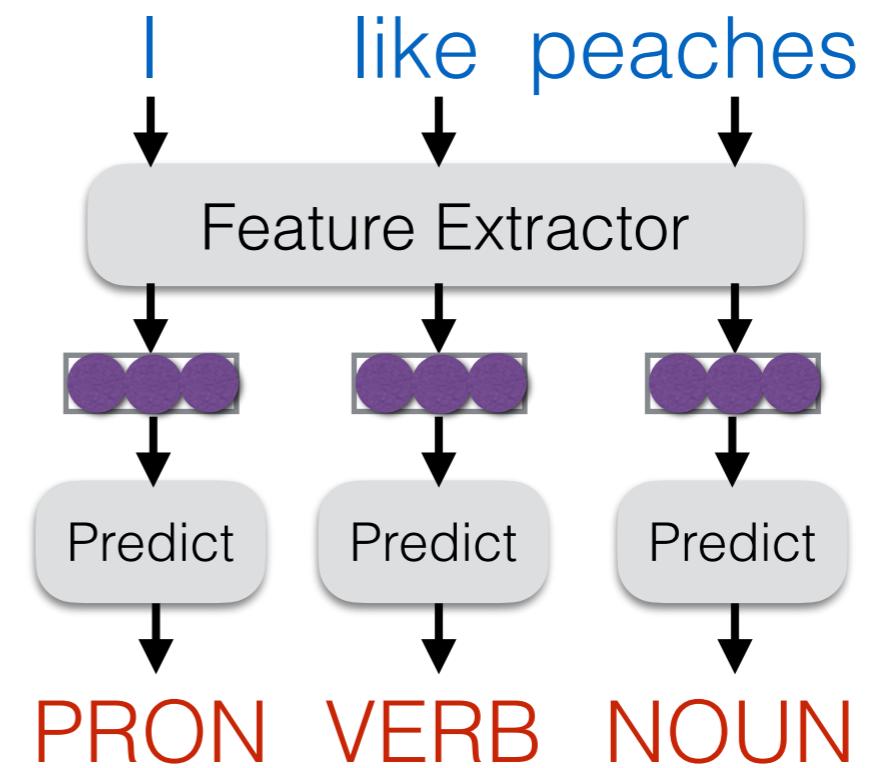
Basic Modeling Paradigm: Extract Features -> Predict

- 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

He saw two birds

↓ ↓ ↓ ↓
PRON VERB NUM NOUN

Lemmatization

He saw two birds

↓ ↓ ↓ ↓
he see two bird

Morphological Tagging

He saw two birds

↓ ↓ ↓ ↓
PronType=prs Tense=past, NumType=card Number=plur
VerbForm=fin

... and more!

Span Labeling

- Given an input text X , predict output spans (sequences of words) and labels Y .

Named Entity Recognition

Carnegie Mellon University is located in Pittsburgh.

ORG

LOC

Syntactic Chunking

Carnegie Mellon University is located in Pittsburgh.

NP

VP

NP

Semantic Role Labeling

Carnegie Mellon University is located in Pittsburgh.

Argument

Predicate

Location

... and more!

Span Labeling as Sequence Labeling

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

Carnegie Mellon University is located in Pittsburgh.

ORG

LOC



Carnegie Mellon University is located in Pittsburgh.

↓ ↓ ↓ ↓ ↓ ↓ ↓
B-ORG I-ORG I-ORG O O O B-LOC

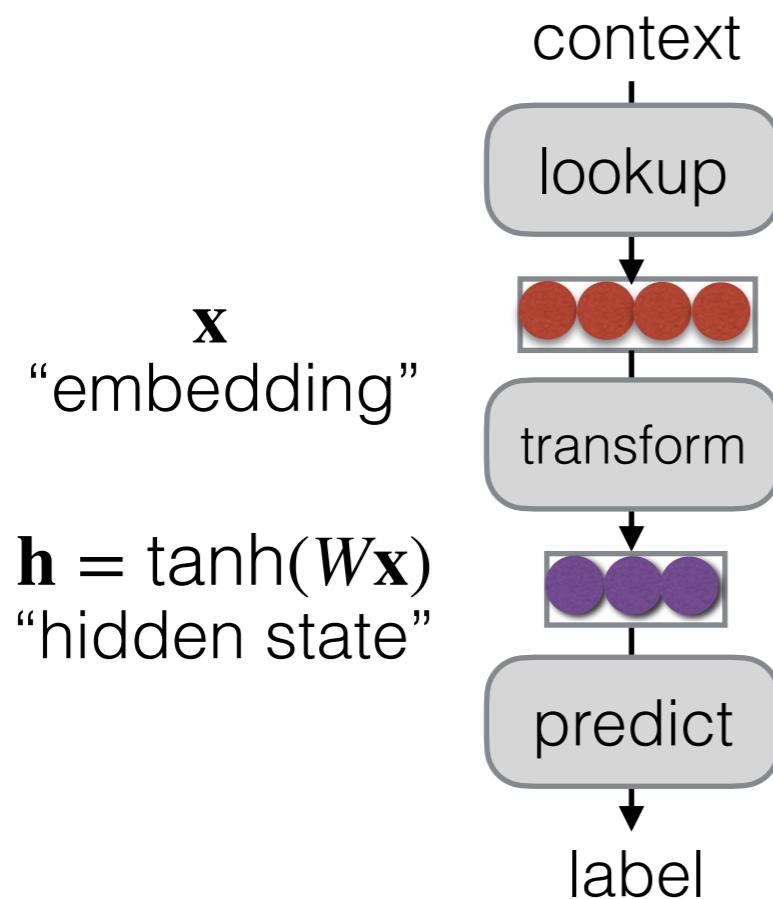
A Sequence Model: Recurrent Neural Networks

Recurrent Neural Networks

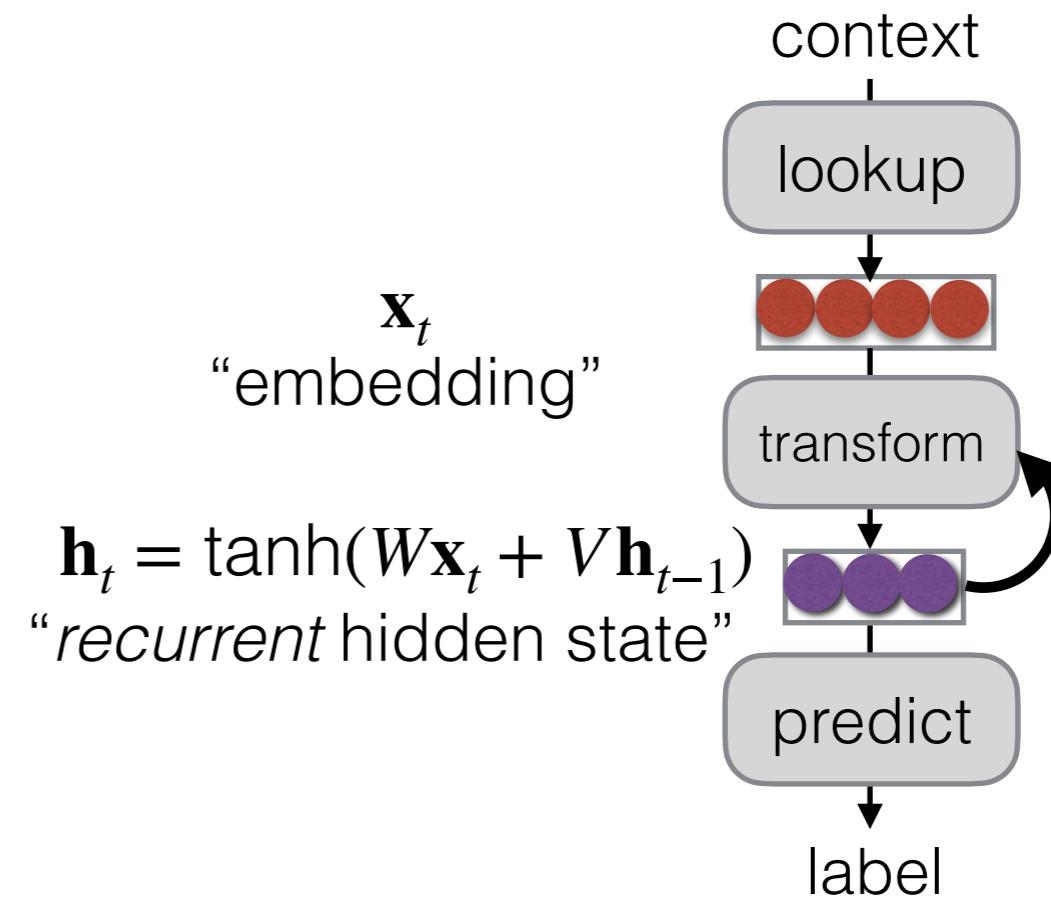
(Elman 1990)

- Tools to “remember” information

Feed-forward NN

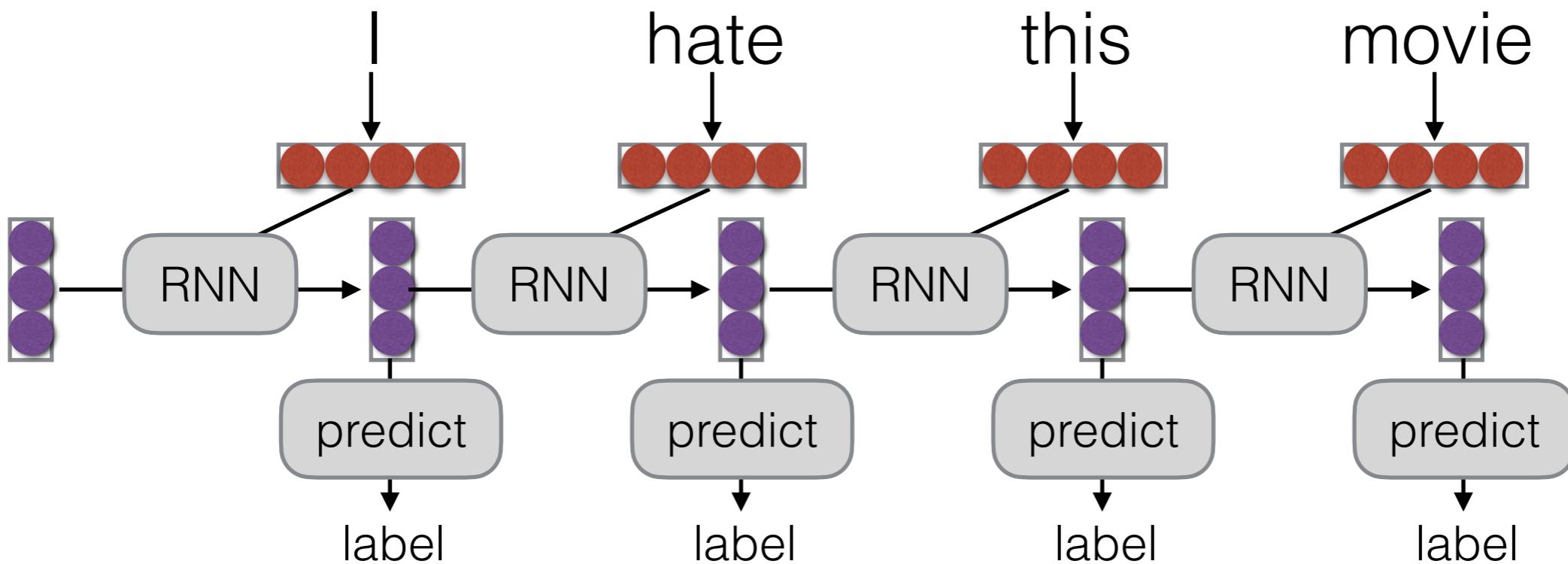


Recurrent NN

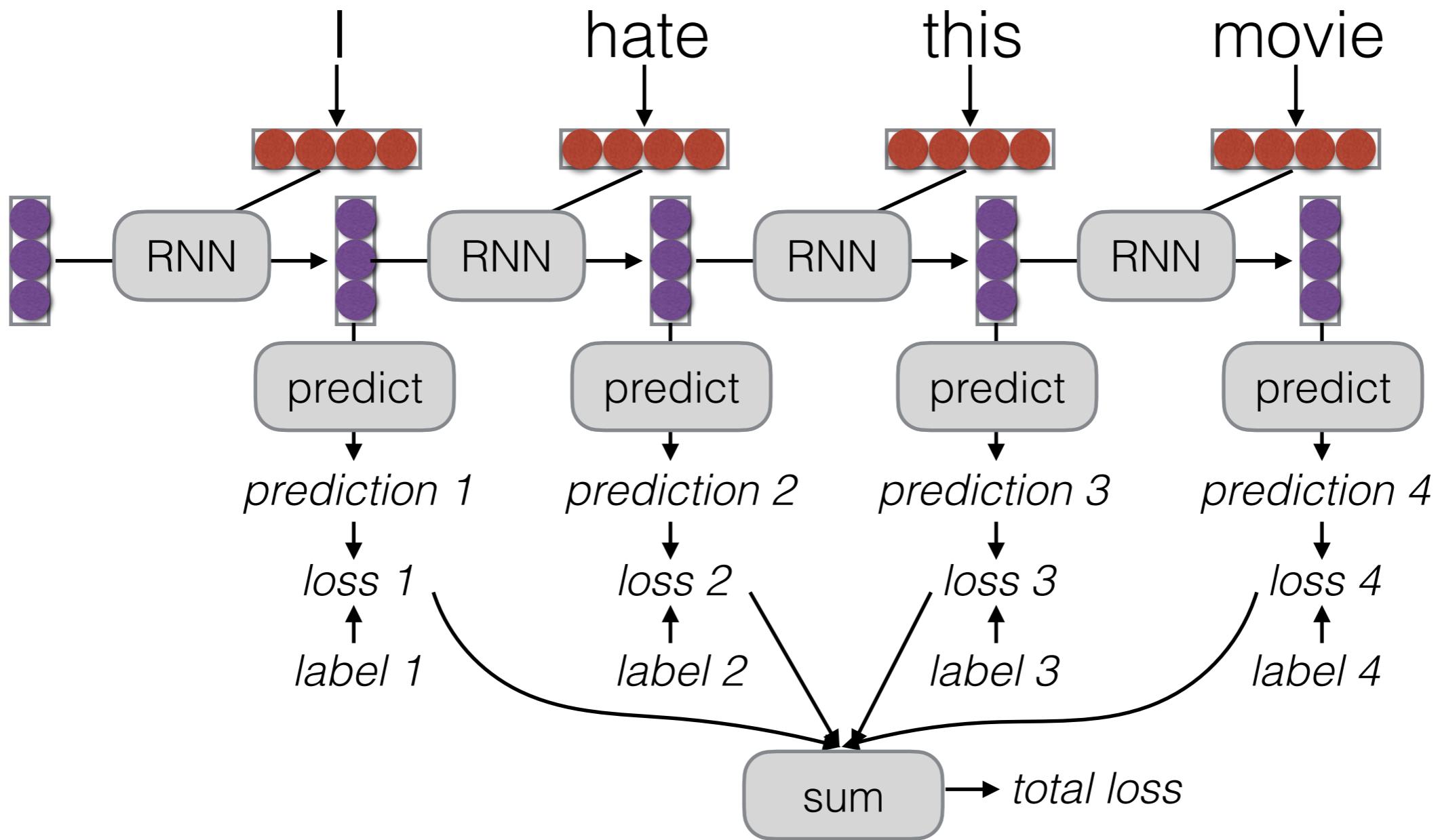


Unrolling in Time

- What does processing a sequence look like?

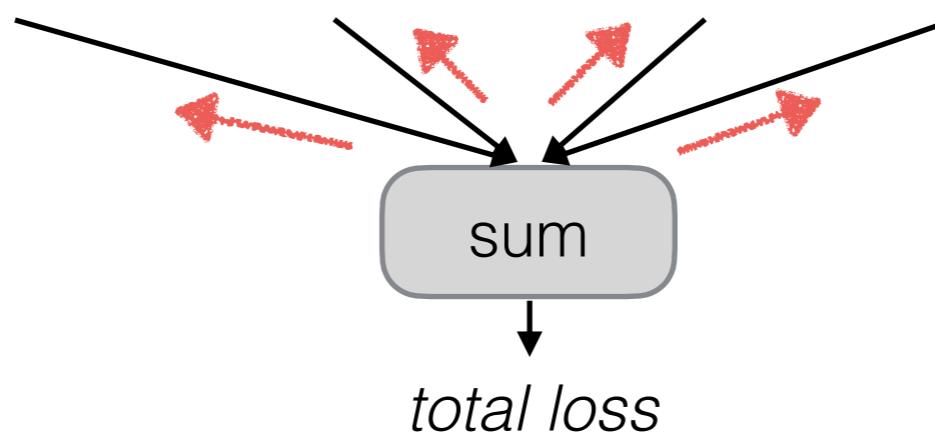


Training RNNs



RNN Training

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

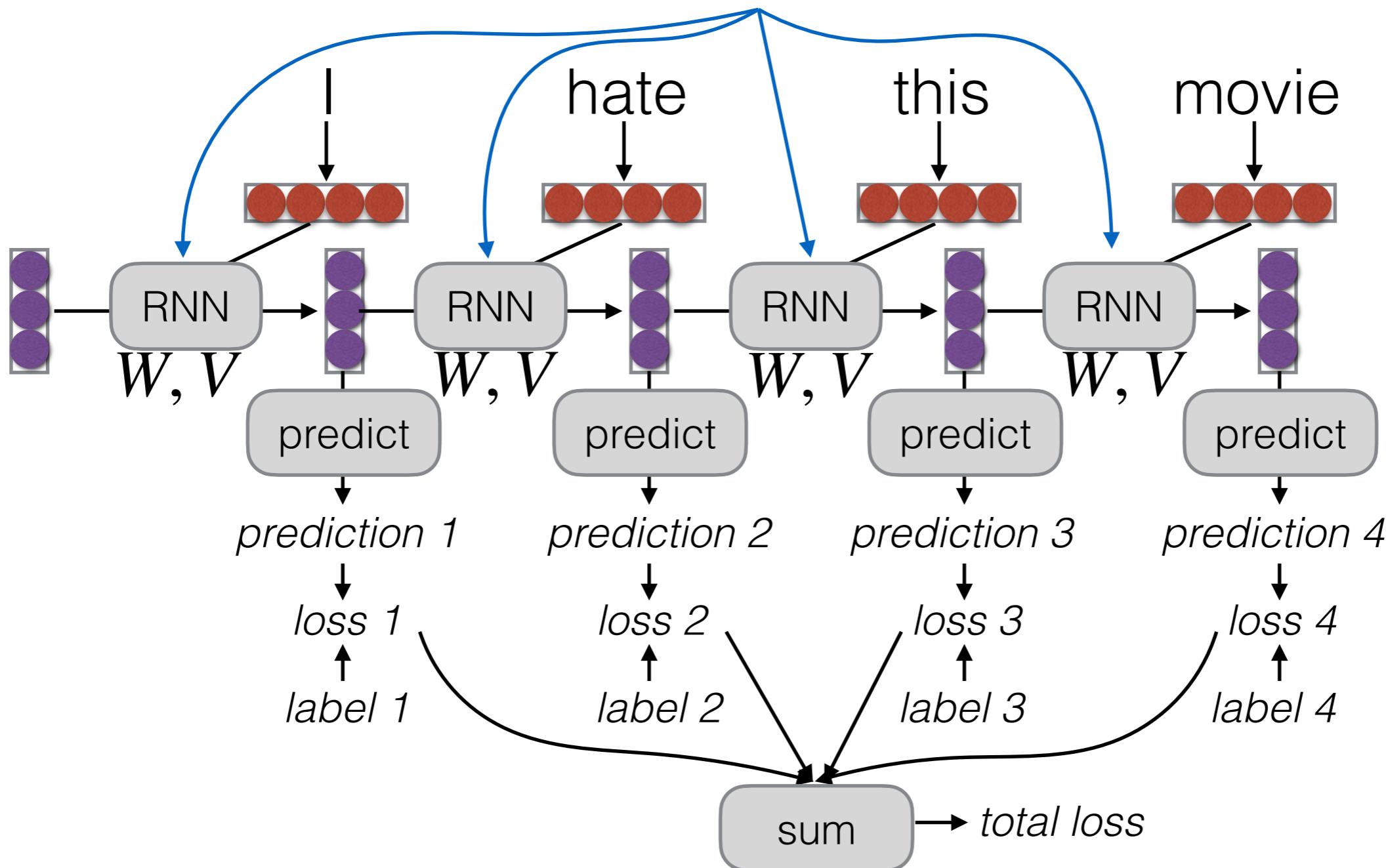


- Parameters are tied across time, derivatives are aggregated across all time steps
- This is historically called “backpropagation through time” (BPTT)

Parameter Tying

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

Parameters are shared! Derivatives are accumulated.



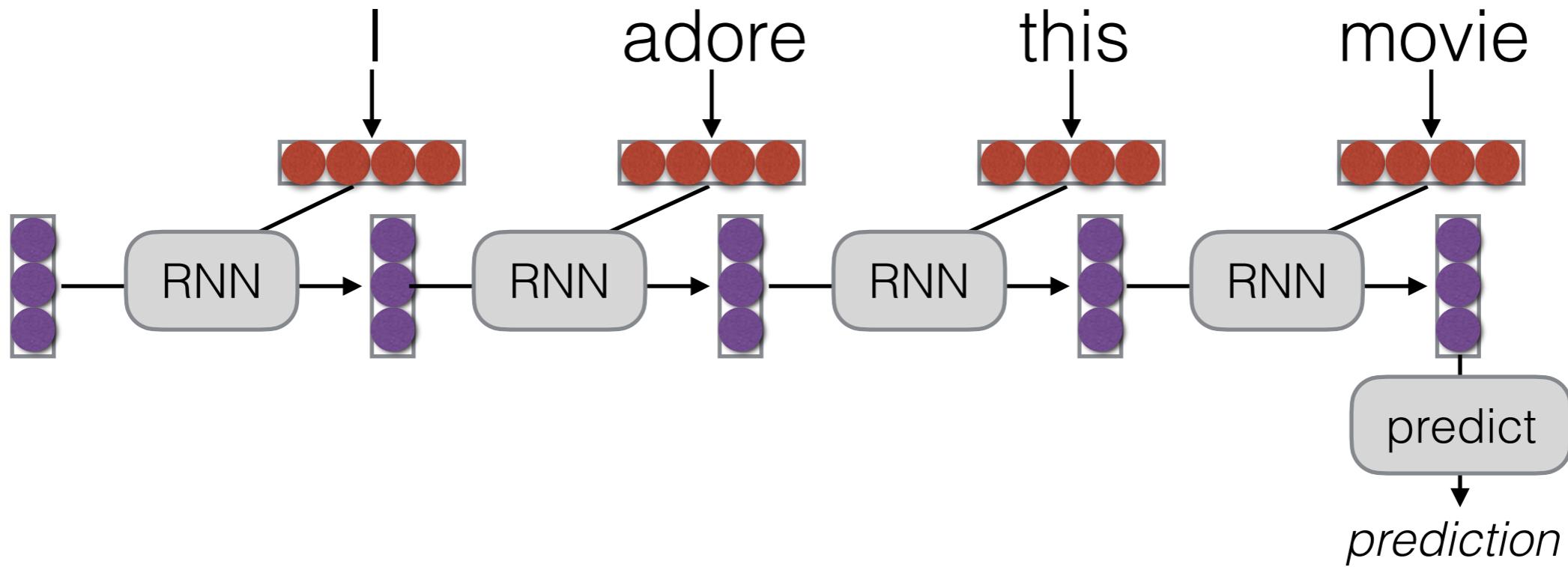
(Same for attention, convolutional networks)

Applications of RNNs

What Can RNNs Do?

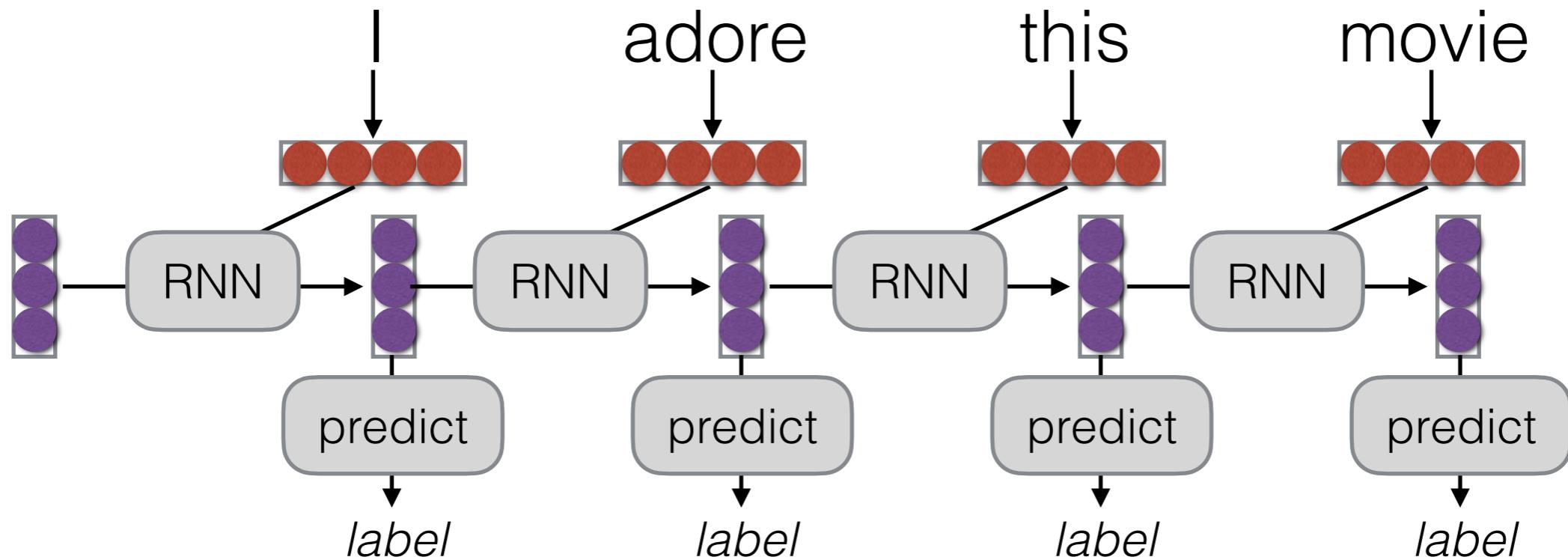
- Represent a sentence
 - Read whole sentence, make a prediction
- Represent a context within a sentence
 - Read context up until that point

Encoding Sentences



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

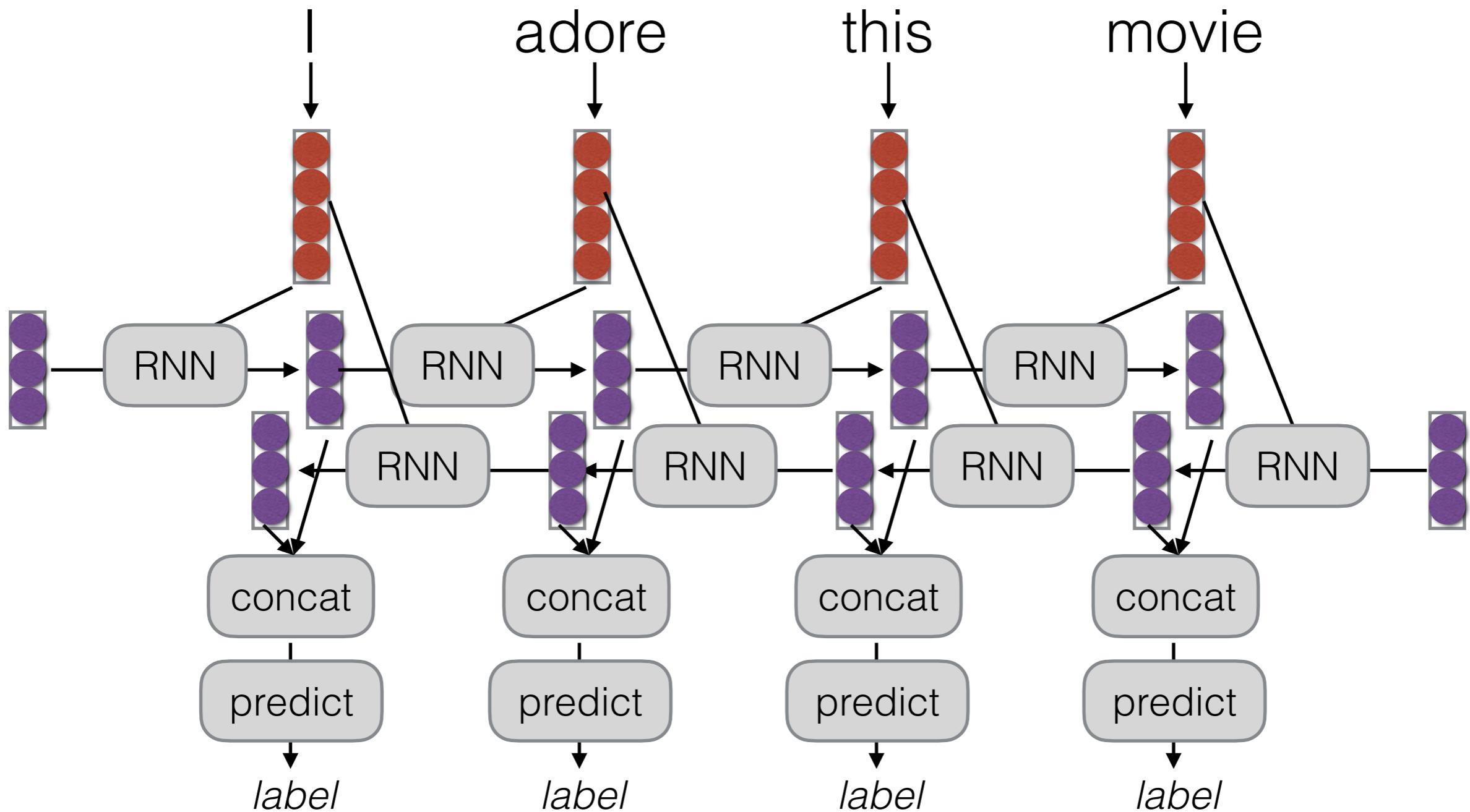
Representing Contexts



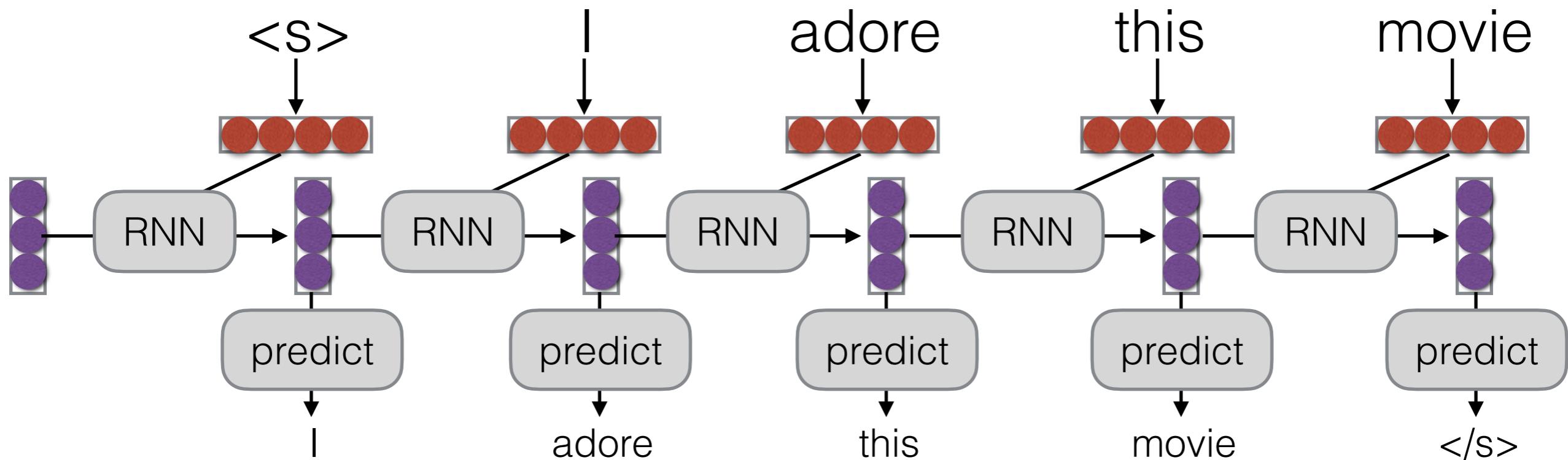
- Sequence labeling
- Calculating representations for parsing, etc.

Bi-RNNs

- A simple extension, run the RNN in both directions



e.g. Language Modeling



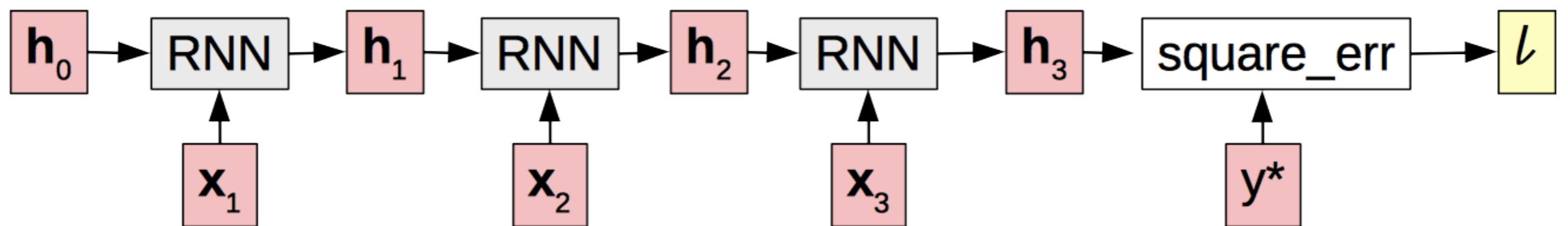
- Language modeling is like a tagging task, where each tag is the next word!
- Note: this is an autoregressive model

Vanishing Gradients

Vanishing Gradient

- Gradients typically decrease as they get pushed back

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

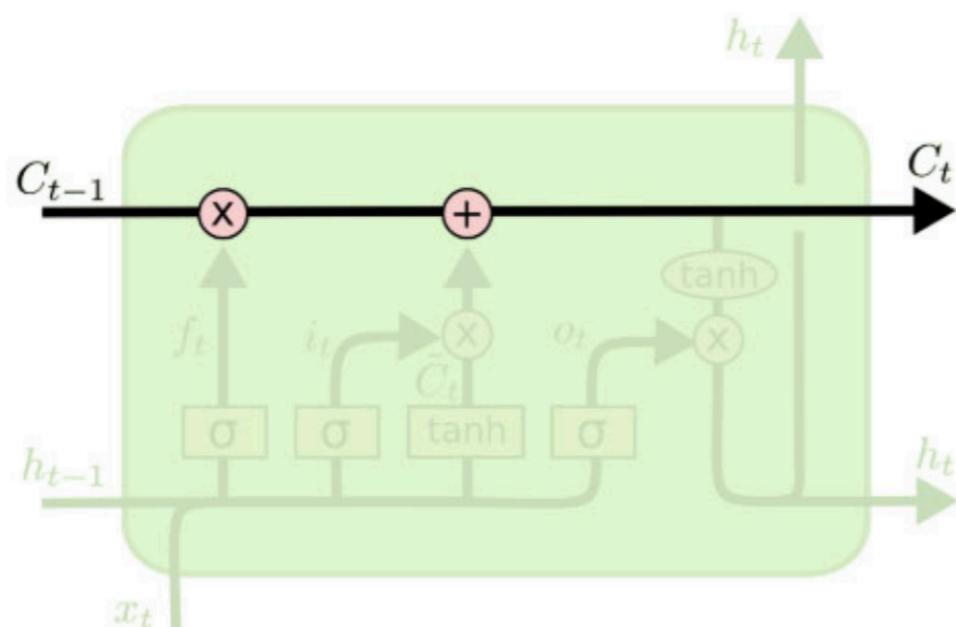


- Why? “Squashed” by non-linearities or small weights in matrices.

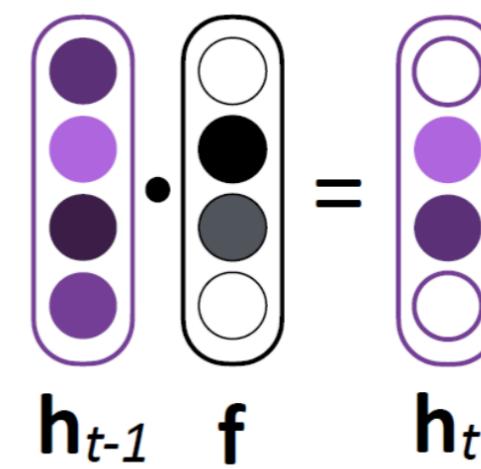
A Solution: Long Short-term Memory

(Hochreiter and Schmidhuber 1997)

- **Basic idea:** make additive connections between time steps (a “conveyor belt”)
- Addition does not modify the gradient, no vanishing
- Gates to control the information flow

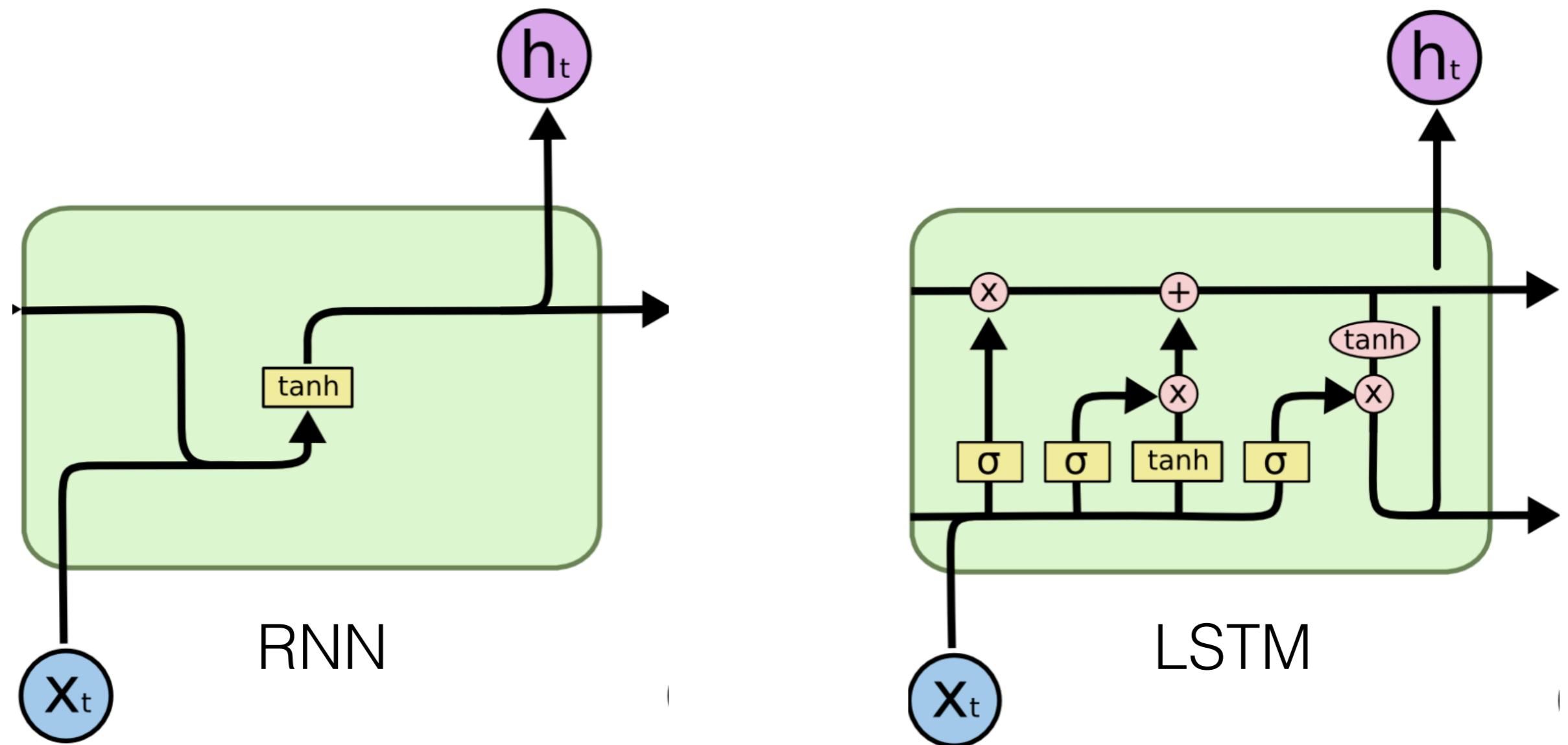


Cell State

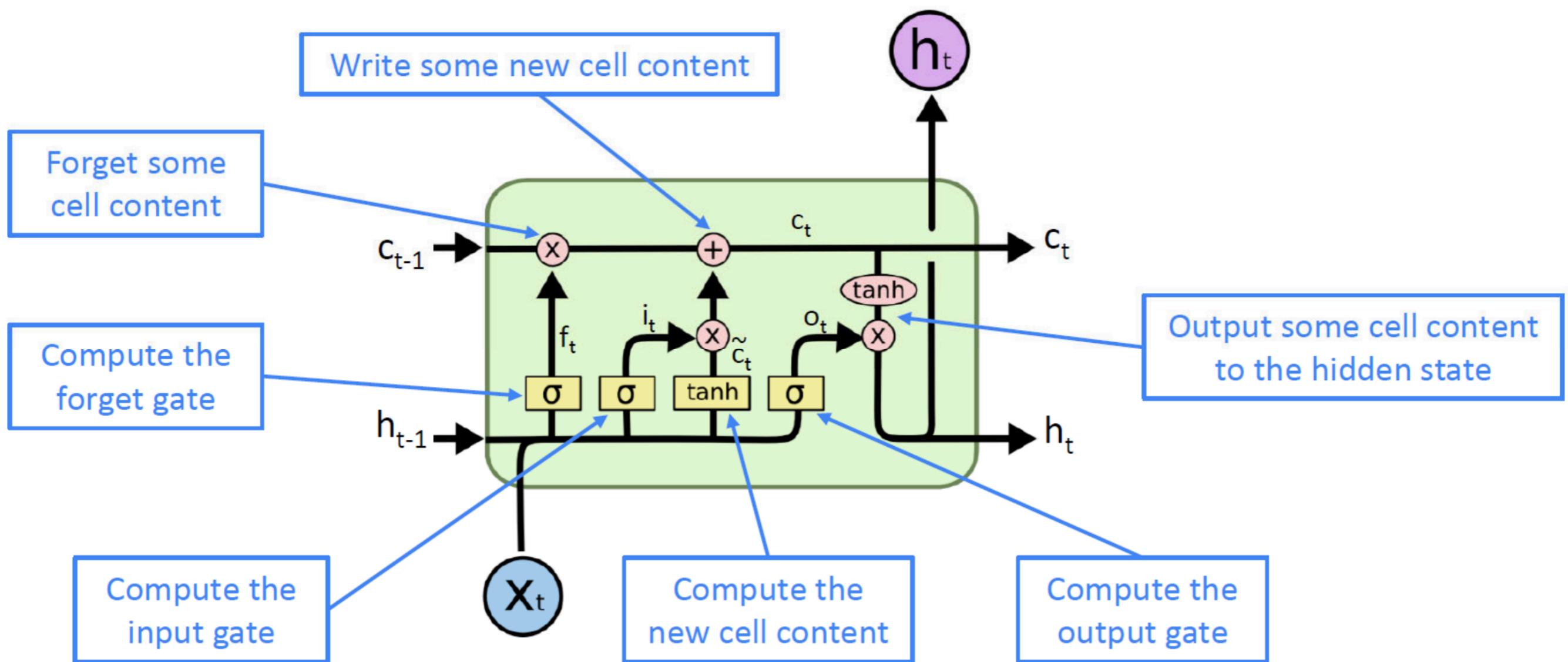


Gating

RNN vs LSTM



LSTMs



Understanding RNNs

LSTMs Can Model Length

- Train *character* LSTM language model (predict next character based on history) over two datasets: War and Peace and Linux kernel source code
- Visualize activations of specific cells (components of **c**) to understand them
- Counter: know when to generate \n

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.

LSTMs Can Model Long-Term Bits

- Train *character* LSTM language model (predict next character based on history) over two datasets: War and Peace and Linux kernel source code
- Visualize activations of specific cells (components of **c**) to understand them
- Binary switch: tells us if we're in a quote or not

"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."

LSTMs Can Be Uninterpretable

- Train *character* LSTM language model (predict next character based on history) over two datasets: War and Peace and Linux kernel source code
- Visualize activations of specific cells (components of **c**) to understand them
- Some cells are uninterpretable: probably doing double-duty, or only make sense in the context of another cell's activations

```
/* 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.
    */
```

Handling Long Sequences

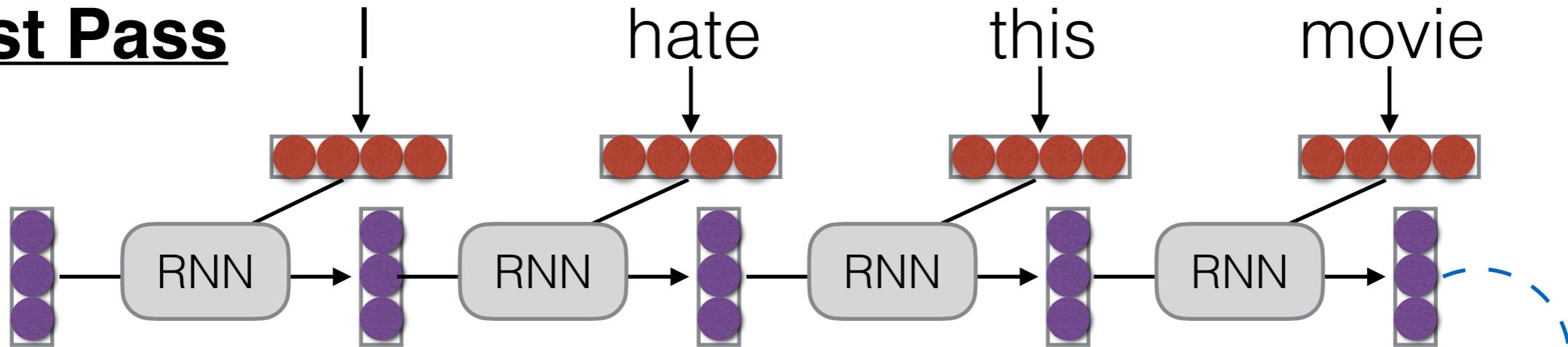
Handling Long Sequences

- 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

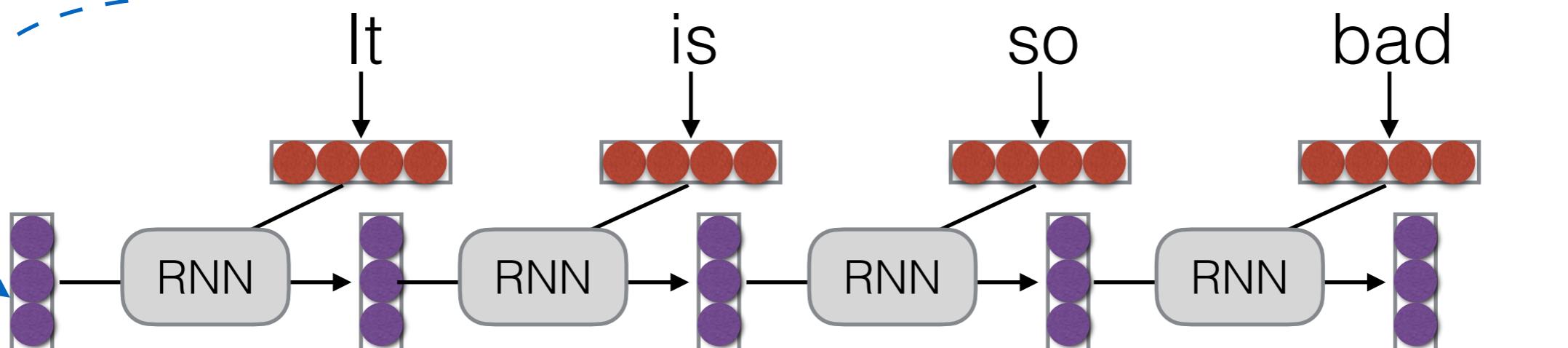
- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass



2nd Pass

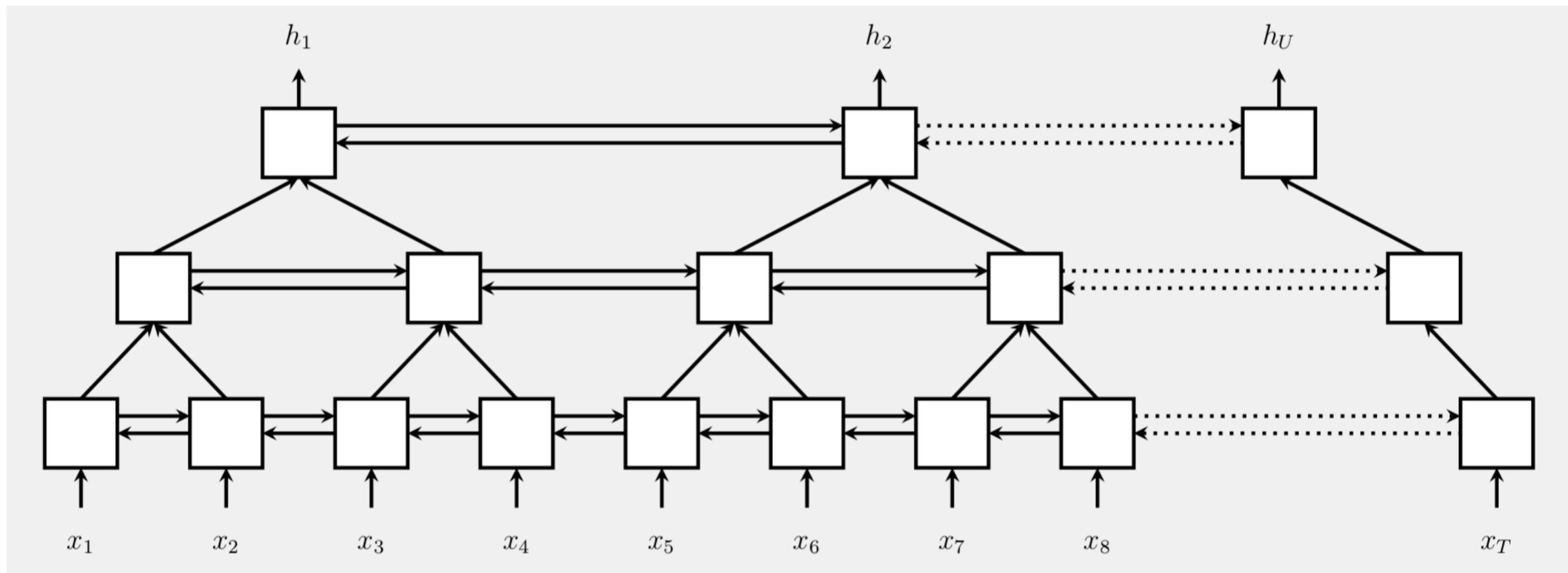
state only, no backprop



Multi-scale/Pyramidal Architectures

(e.g. Chan et al. 2015)

- Downscale between layers



Types of Prediction
Recurrent Neural Nets
RNN Applications
Vanishing Gradients

Understanding RNNs
Efficiency Tricks
Handling Long Sequences
RNN Variants

Questions?