Recurrent Neural Networks

Phạm Quang Nhật Minh
Aimesoft JSC
minhpham0902@gmail.com

Lecture outline

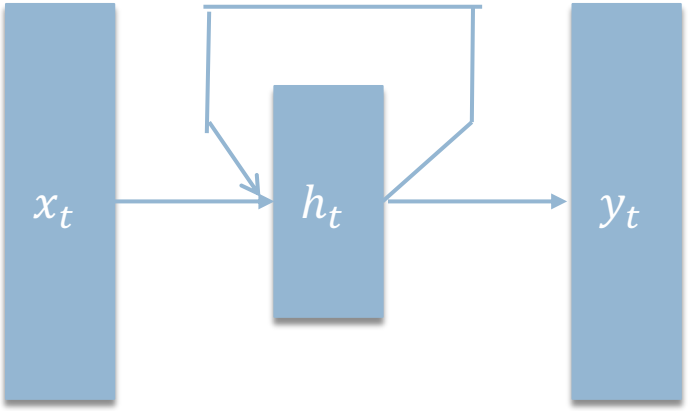
- Recurrent neural networks
- Multi-layer RNNs (Stacked RNNs)
- Bidirectional RNNs
- Two types of RNNs: LSTM and GRU

NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters in words
  - Sentences in discourse
  - ...
- Sequences have variable lengths
- Long-distance dependencies in languages
  - **He** does not have very much confidence in **himself**.
  - **She** does not have very much confidence in **herself.**

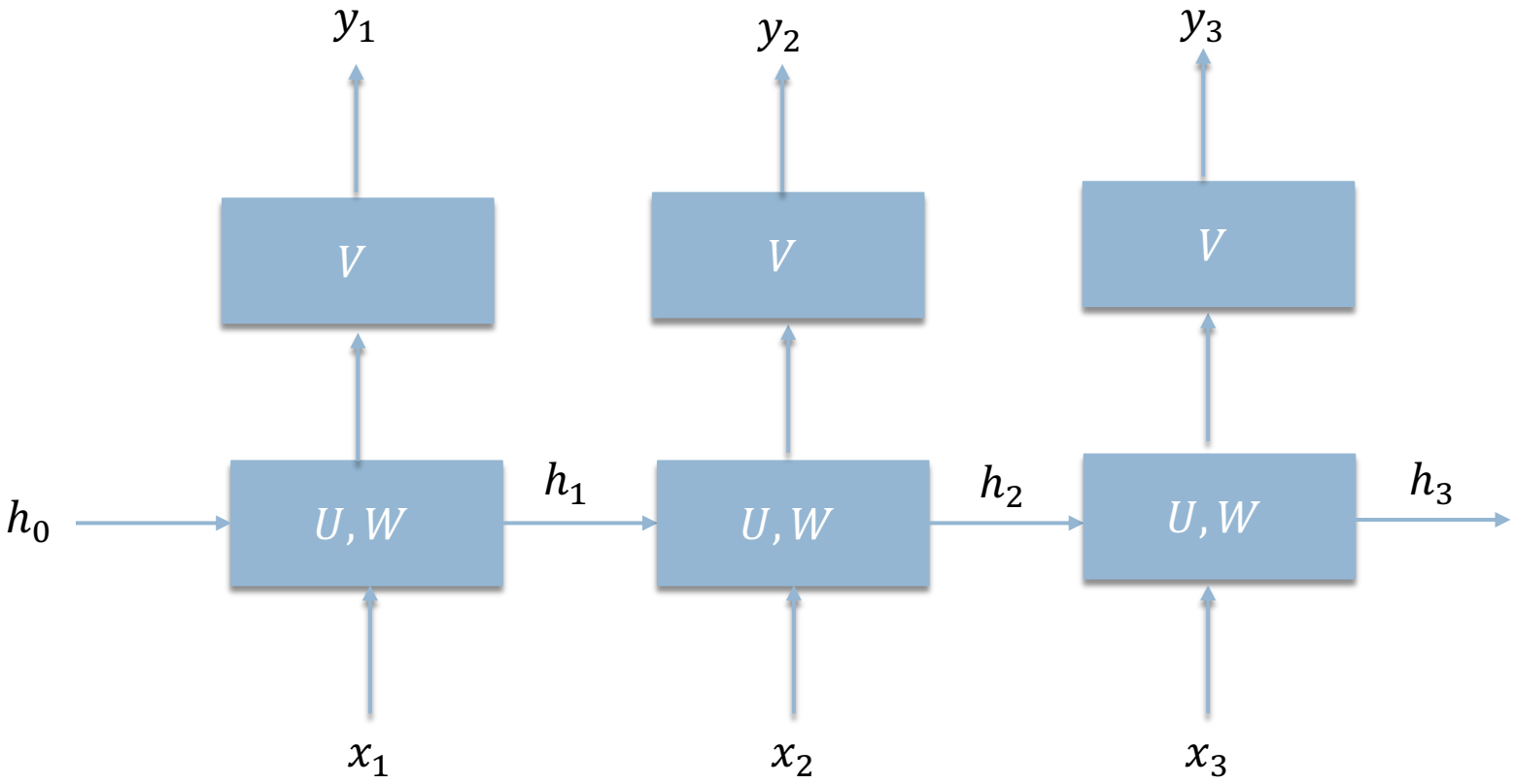
Basic ideas of Recurrent neural networks (RNN)

- Use reccurent link
  - Output of *t-1* step is used as input for the *t* step
- Used same weights among steps



Unrolled representation of RNNs

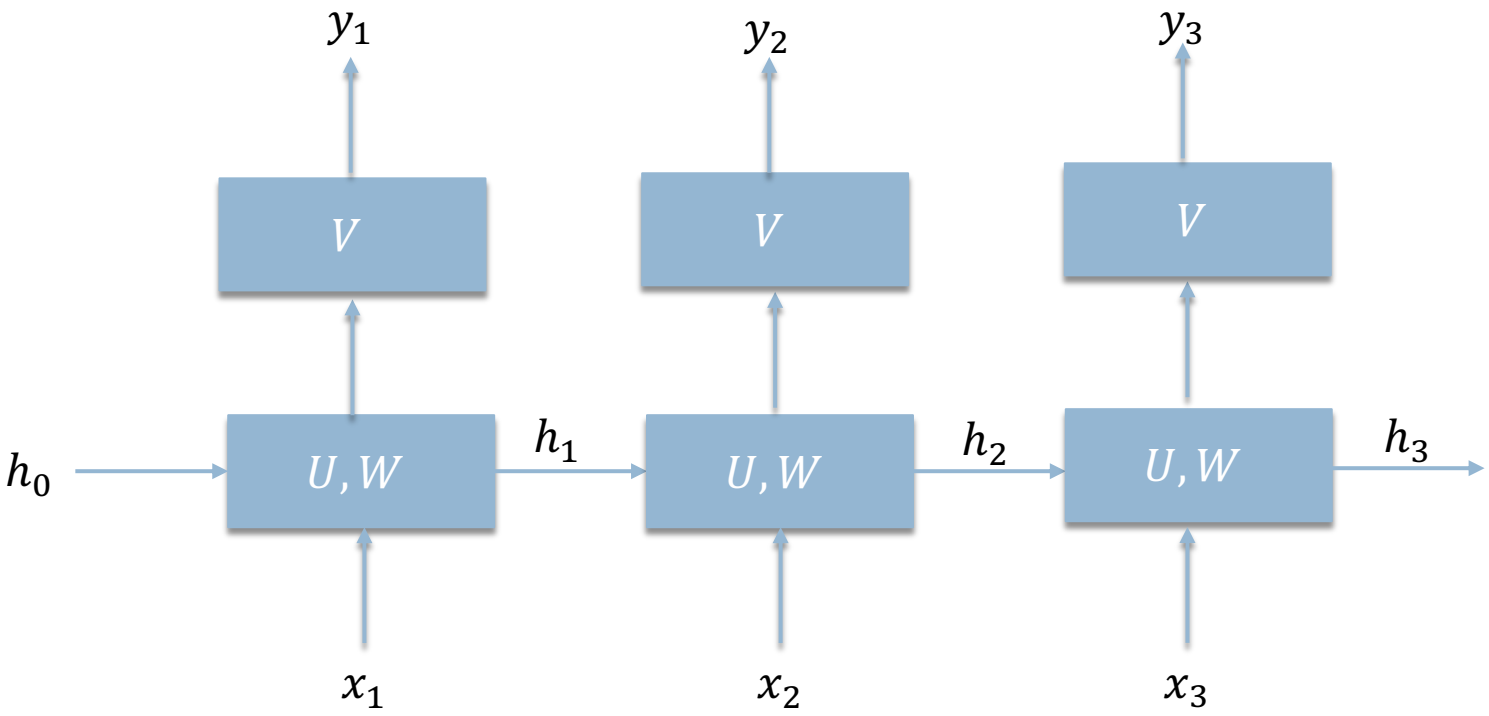
x_i are vectors



Equations in RNN

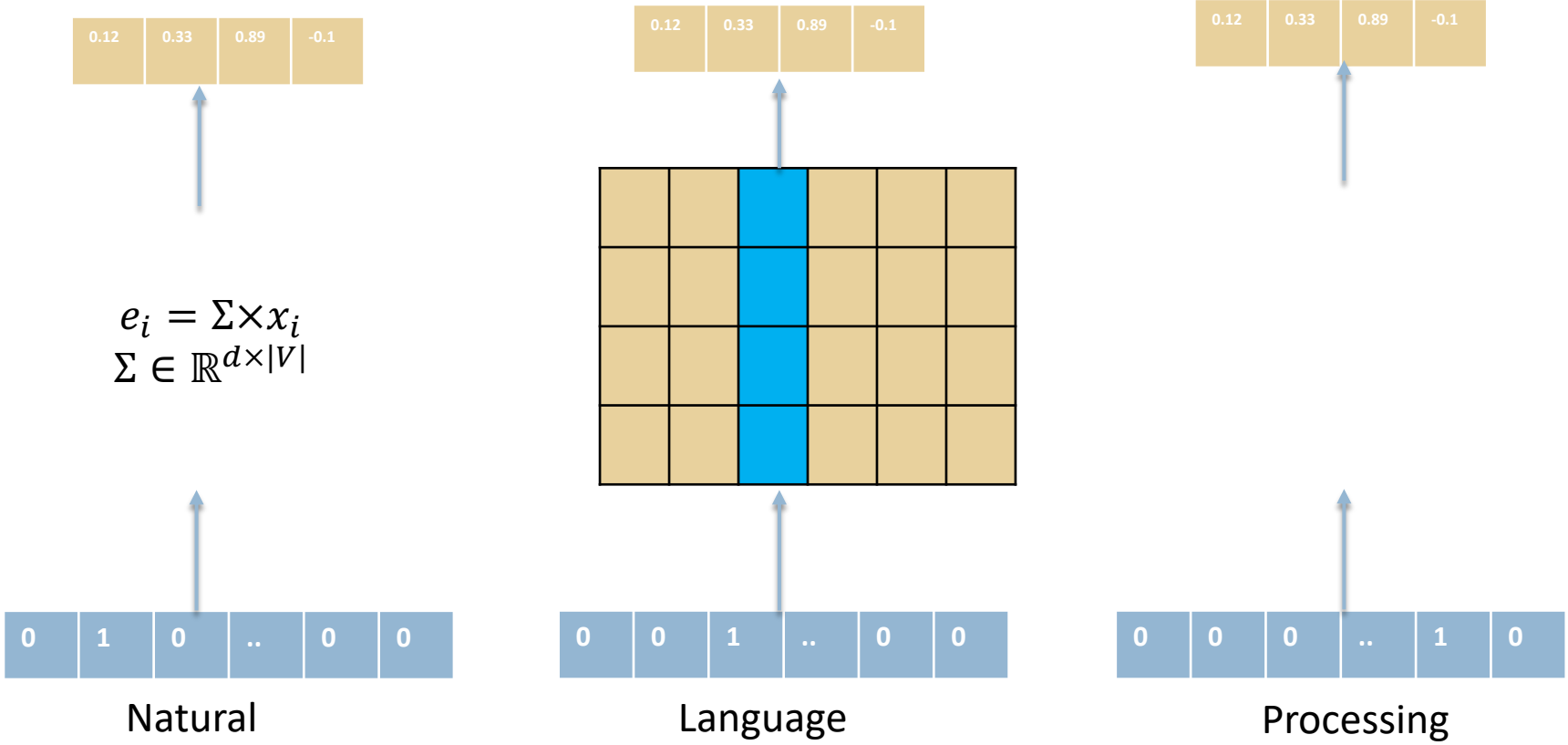
$$h_t = g(Uh_{t-1} + Wx_t)$$
$$y_t = f(Vh_t)$$
$$W \in \mathbb{R}^{d_h \times d_{in}}, U \in \mathbb{R}^{d_h \times d_h}, V \in \mathbb{R}^{d_{out} \times d_h}$$

d_{in}, d_h, d_{out} are input, hidden, and output layer dimensions



Input Layers of RNN in NLP tasks

- Each word is a vector
  - Dense vector obtained by an embedding layer (lookup matrix)



Forward inference in RNN

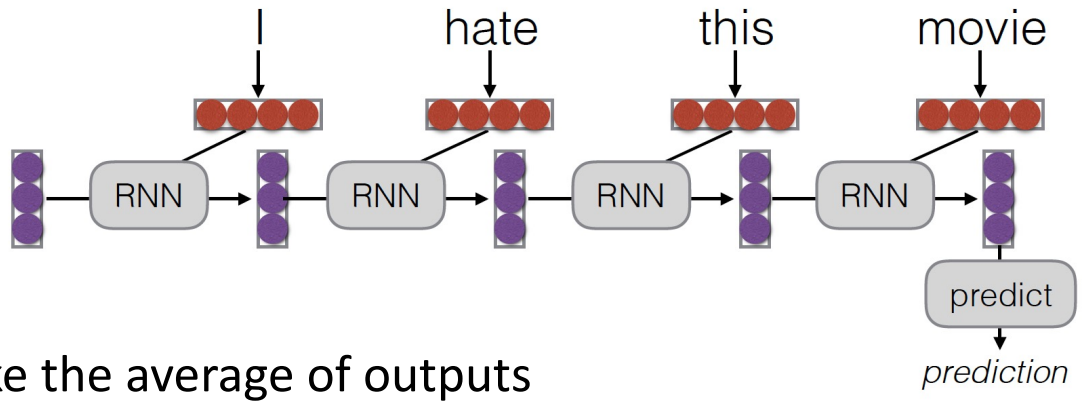
function FORWARDRNN($\mathbf{x}$, *network*) **returns** output sequence $\mathbf{y}$

$\mathbf{h}_0 \leftarrow 0$
for $i \leftarrow 1$ **to** LENGTH($\mathbf{x}$) **do**
  $\mathbf{h}_i \leftarrow g(\mathbf{U}\mathbf{h}_{i-1} + \mathbf{W}\mathbf{x}_i)$
  $\mathbf{y}_i \leftarrow f(\mathbf{V}\mathbf{h}_i)$
return y

The matrices **U, V** and **W** are shared across time, while new values for h and y are calculated with each time step

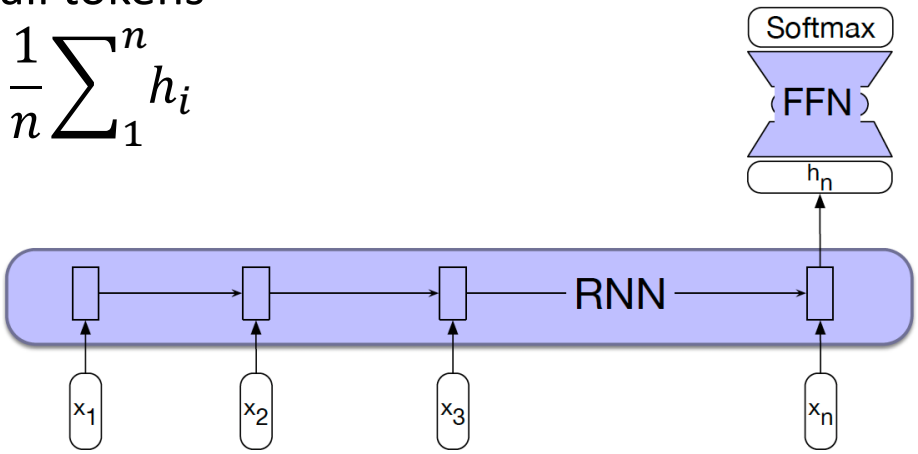
RNN for Sequence Classification

- Use the output of the hidden layer at of the last token as the representation of the entire sequence

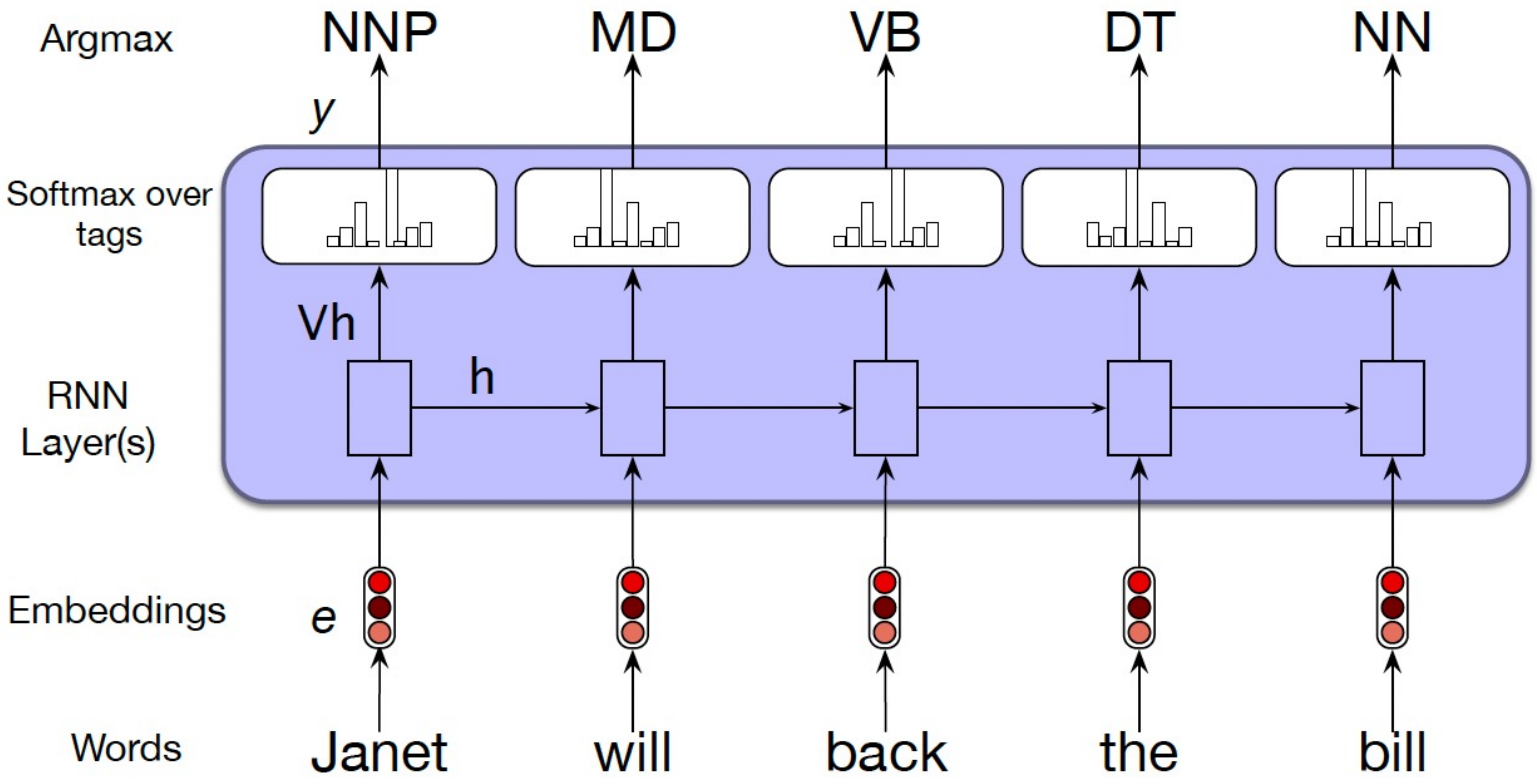


We can take the average of outputs of hidden layer for all tokens

$$h_{mean} = \frac{1}{n}\sum_{1}^{n} h_i$$



RNN for Sequence Labeling

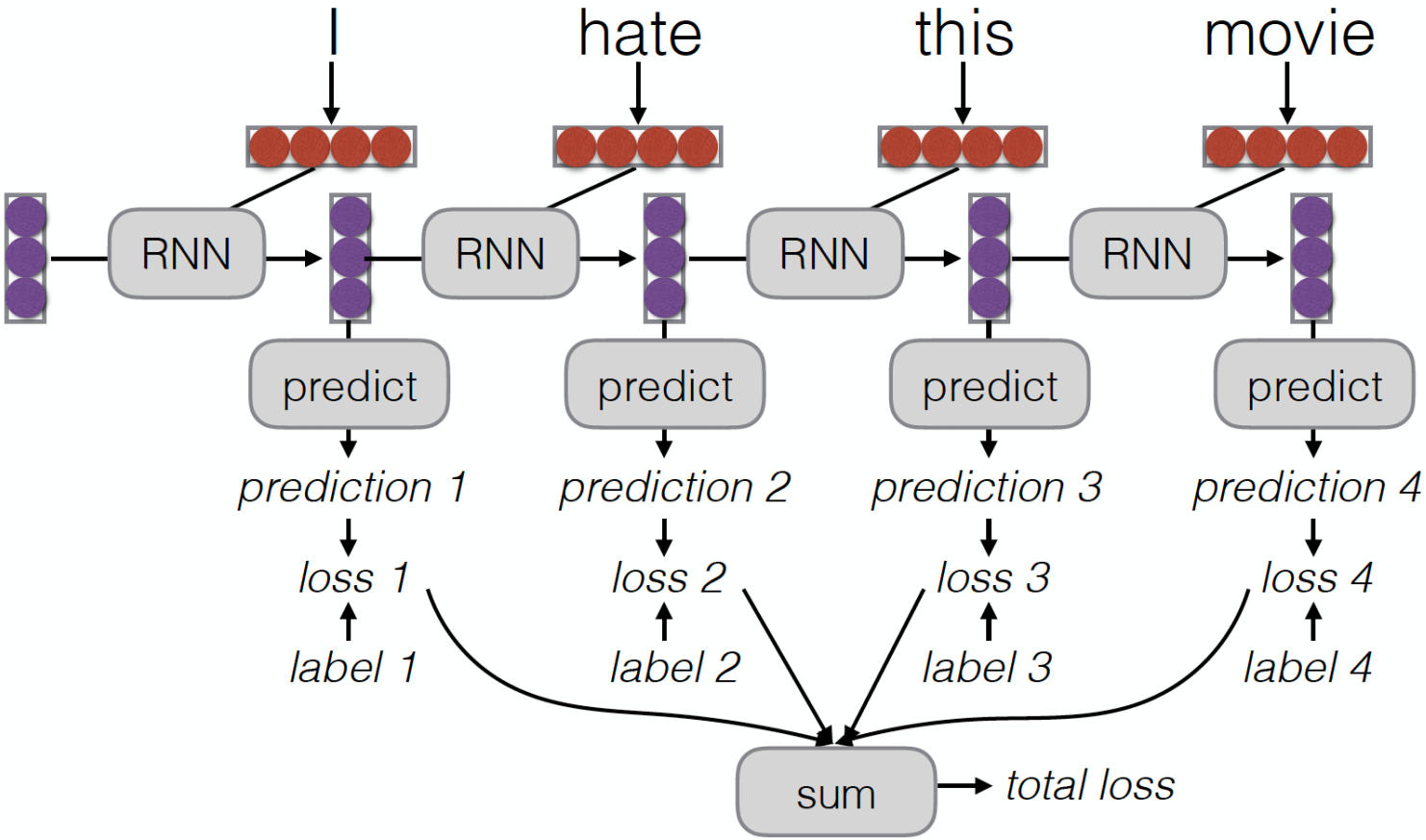


Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

Training RNNs

- Training RNN steps:
  - Unroll recurrent neural networks
  - Apply backpropagation to calculate gradients
- Algorithm of training RNN is called backpropagation through time (BPTT) (Werbos, 1990)

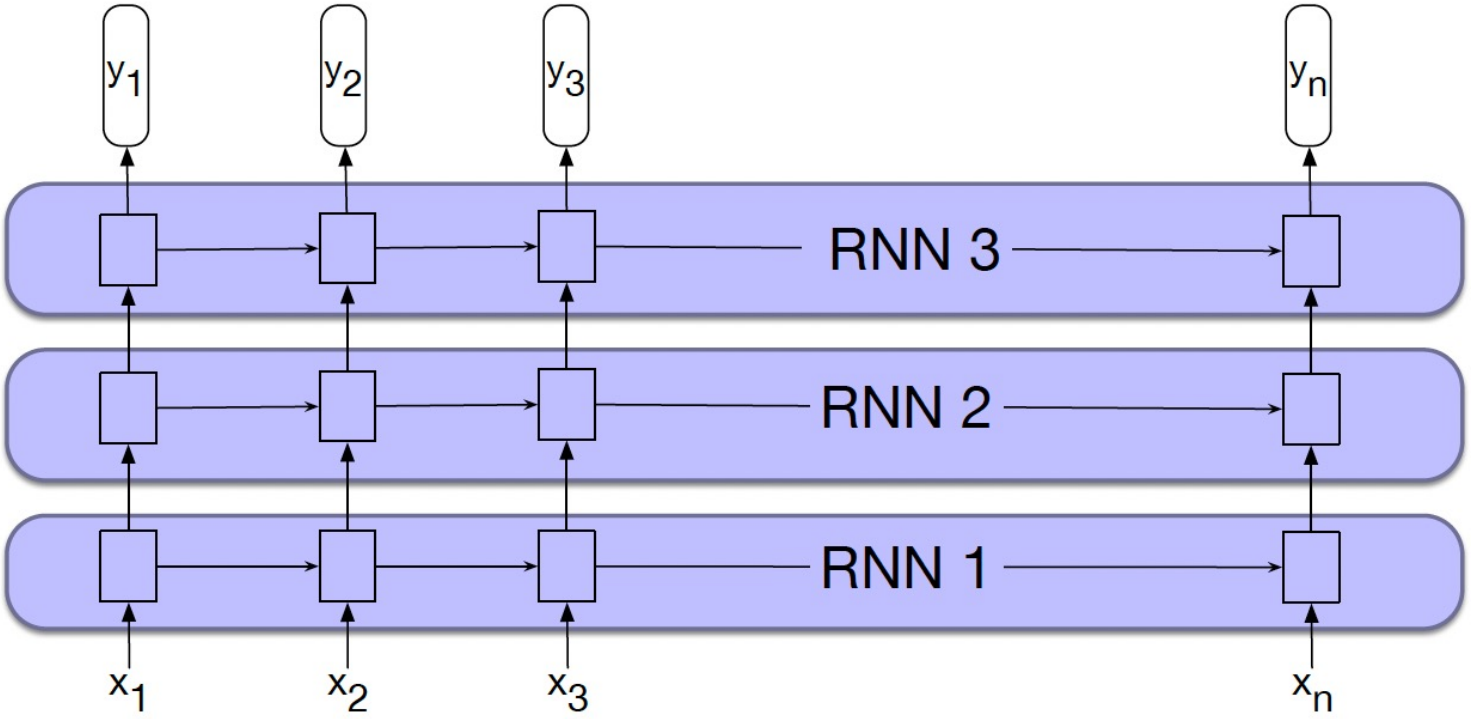
Training RNNs in tagging problems



Lecture outline

- Recurrent neural networks
- Multi-layer RNNs (Stacked RNNs)
- Bidirectional RNNs
- Two types of RNNs: LSTM and GRU

Stacked RNNs



The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

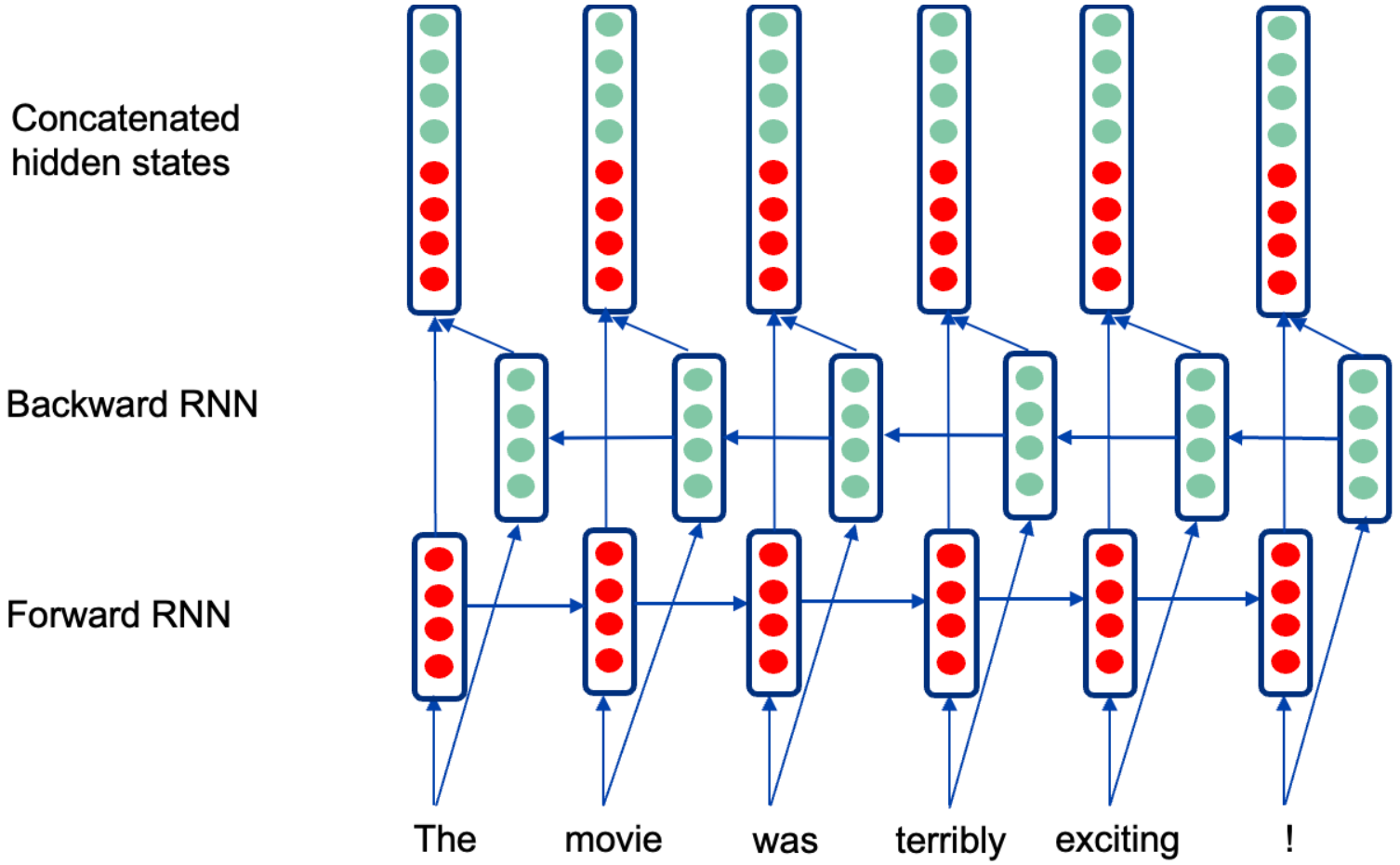
Stacked RNNs generally outperform single-layer networks but the training cost is higher than single-layer RNN networks

Lecture outline

- Recurrent neural networks
- Multi-layer RNNs (Stacked RNNs)
- Bidirectional RNNs
- Two types of RNNs: LSTM and GRU

Bidirectional RNNs

Context in both directions (left and right) is useful in some task (such as POS Tagging)



Bidirectional RNN: Formulas

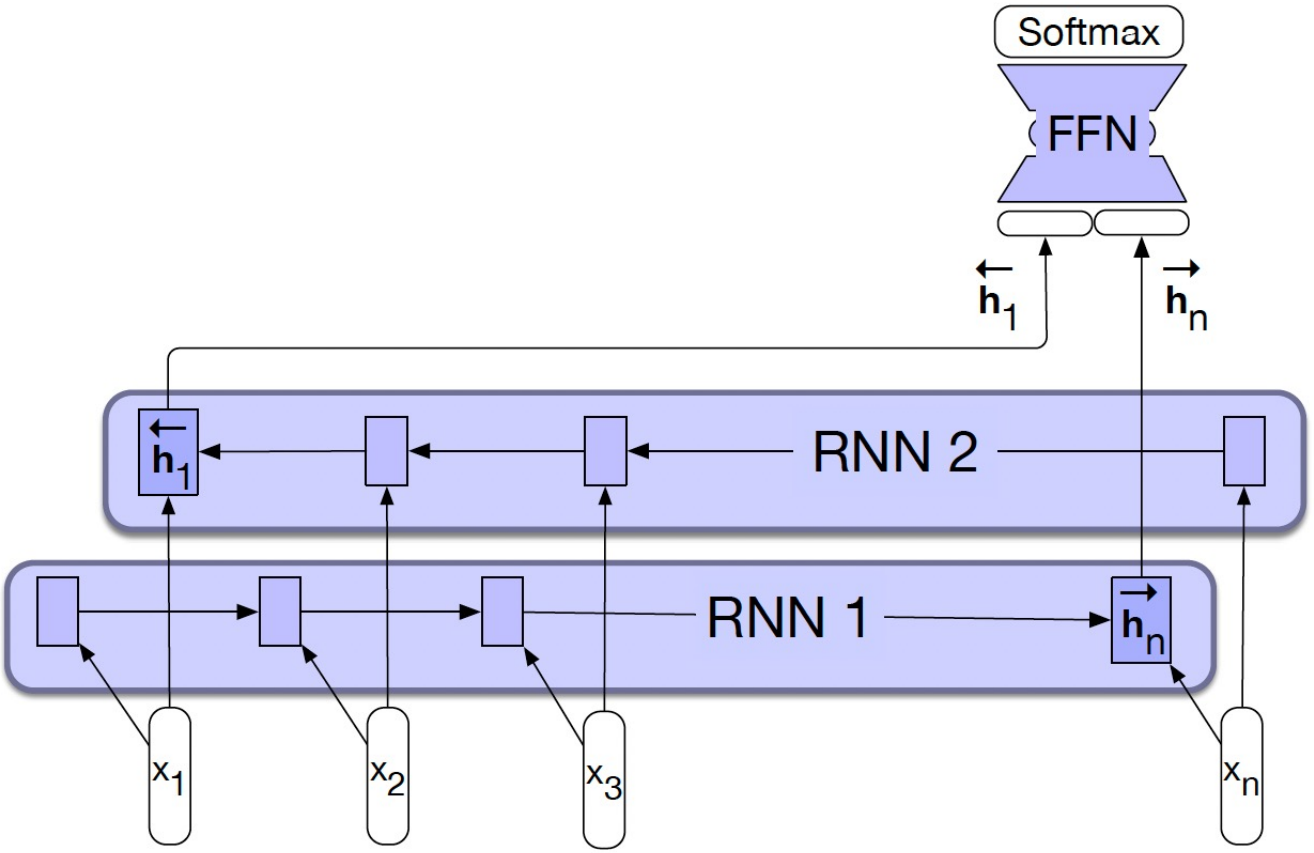
On time step t:

Forward RNN $\quad \vec{h}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(x_1, \ldots, x_t)$

Backward RNN $\quad \overleftarrow{h}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(x_t, \ldots, x_n)$

Concatenated hidden states $\quad h^{(t)} = [\vec{h}^{(t)}; \overleftarrow{h}^{(t)}]$

Bidirectional RNN for Sequence Classification



The final hidden units from the forward and backward passes are combined to represent the entire sequence. This combined representation serves as input to the subsequent classifier.

Lecture outline

- Recurrent neural networks
- Multi-layer RNNs (Stacked RNNs)
- Bidirectional RNNs
- Two types of RNNs: LSTM and GRU

Why Long Short-term Memory (LSTM)?

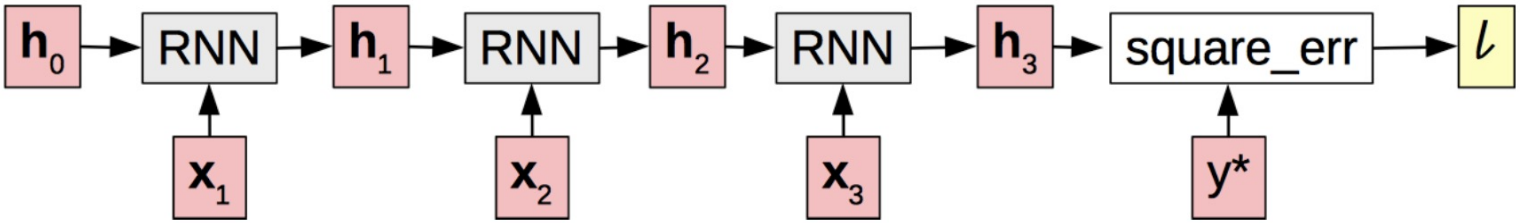
- Standard RNNs could not handle long-term dependencies well
  - “I grew up in France… I speak fluent *French*.”

Why Long Short-term Memory (LSTM)?

- Vanishing Gradient Descent problem
  - Gradients decrease as they get pushed back

$$\frac{dl}{dh_0} = \frac{dh_1}{dh_0} \times \frac{dh_2}{dh_1} \times \frac{dh_3}{dh_2} \times \frac{dl}{dh_3}$$

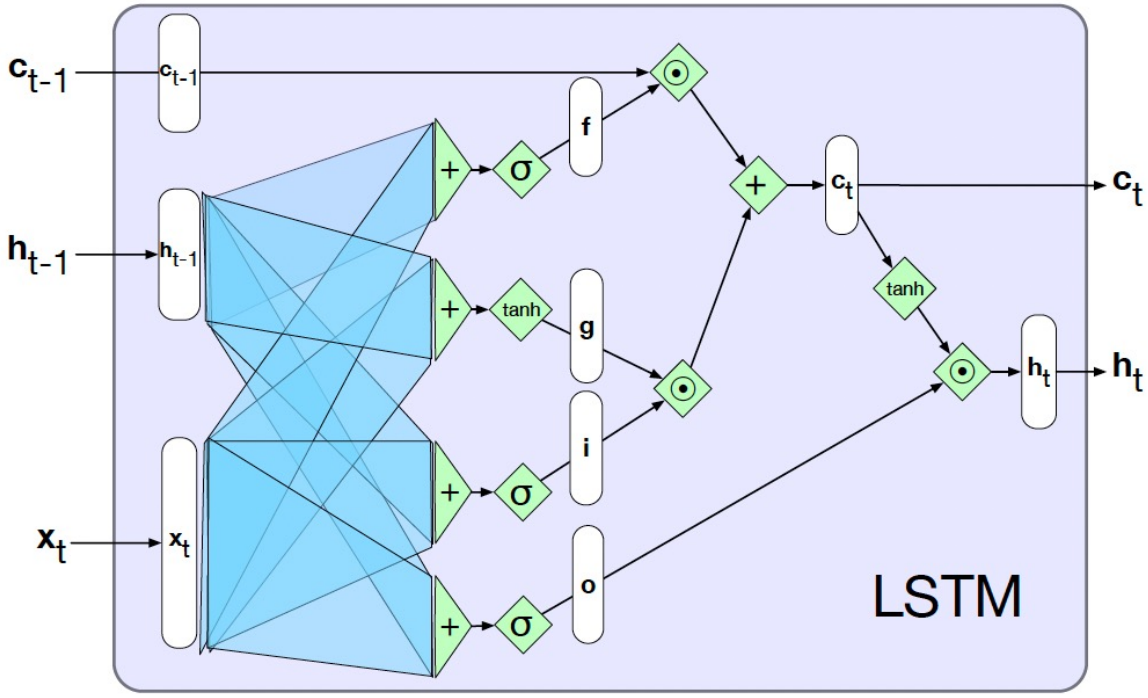
  - $\frac{dl}{dh_0} = \text{tiny}, \frac{dl}{dh_1} = \text{small}, \frac{dl}{dh_3} = \text{med}, \frac{dl}{dh_4} = \text{large}$



Basic ideas of LSTM networks

- On step *t*, there is a hidden state *and* a cell state
  - The cell stores long-term information
  - The LSTM can erase, write and read information from the cell
- The selection of which information is erased/written/read is controlled by gates

LSTM Network architecture



A single LSTM unit displayed as a computation graph. The inputs to each unit consists of the current input, x, the previous hidden state h_{t-1}, , and the previous context c_{t-1}, . The outputs are a new hidden state, h_t and an updated context, c_t.

LSTM Network Equations

- **Forget gate**: to delete information from the context that is no needed

$$\mathbf{f}_t = \sigma(\mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{W}_f \mathbf{x}_t)$$
$$\mathbf{k}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t$$

- **Input gate**: to select information to add to the current context

$$\mathbf{g}_t = \tanh(\mathbf{U}_g \mathbf{h}_{t-1} + \mathbf{W}_g \mathbf{x}_t)$$
$$\mathbf{i}_t = \sigma(\mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{W}_i \mathbf{x}_t)$$
$$\mathbf{j}_t = \mathbf{g}_t \odot \mathbf{i}_t$$
$$\mathbf{c}_t = \mathbf{j}_t + \mathbf{k}_t$$

- **Output gate:** to decide what information is required for the current hidden state

$$\mathbf{o}_t = \sigma(\mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{W}_o \mathbf{x}_t)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

How does LSTM solve vanishing gradients?

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
- LSTM doesn't *guarantee* that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies