

CSC 439/539  
Statistical Natural Language Processing  
Lecture 7: Sequences and Neural Networks

Mihai Surdeanu  
Fall 2017

1

---

---

---

---

---

---

---

### Take-away

- Convolutional neural networks
- Recurrent neural networks
- Long short-term memory networks

2

---

---

---

---

---

---

---

### Readings

- Goldberg, Y. Neural Network Methods for Natural Language Processing.
  - Available online for UA students at:  
<https://ebookcentral-proquest-com.ezproxy1.library.arizona.edu/lib/UAZ/detail.action?docID=4843762>
- Olah, C. Conv Nets: A Modular Perspective.
  - <http://colah.github.io/posts/2014-07-Conv-Nets-Modular/>
- Olah, C. Understanding LSTM Networks.
  - <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

3

---

---

---

---

---

---

---

## Take-away

- Convolutional neural networks
- Recurrent neural networks
- Long short-term memory networks

4

---



---



---



---



---



---



---



---

## Convolutional neural networks (CNN)

- How many of you obtained better performance in assignment 2 with bigrams?
- Without  $n$ -grams, these 2 sentences are similar:
  - “It was not good, it was actually quite bad.”
  - “It was not bad, it was actually quite good.”
- CNNs capture  $n$ -grams in neural networks
  - They do it cheaply, without the need for an embedding for each bigram.

5

---



---



---



---



---



---

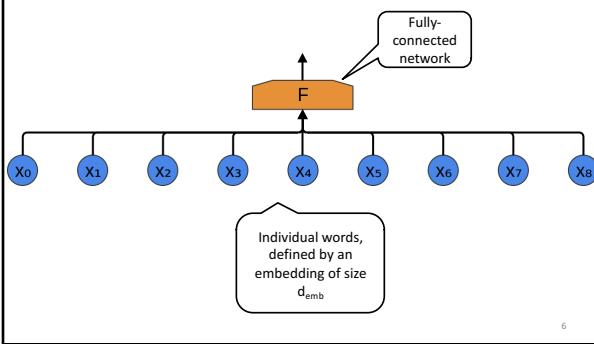


---



---

## Bag-of-word feed-forward NN



6

---



---



---



---



---



---

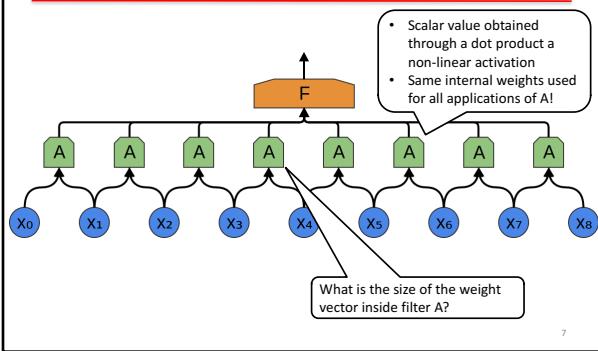


---



---

## After applying one filter A



7

---

---

---

---

---

---

---

---

## Exercise

- What values do you obtain after applying a sigmoid convolution filter of size 2 with the weights {1, 2, 3, 4} over the following sentence, where each word has an embedding of size 2?
  - W1: {1, 2}
  - W2: {2, 2}
  - W3: {1, 0}
  - W4: {-1, -2}

8

---

---

---

---

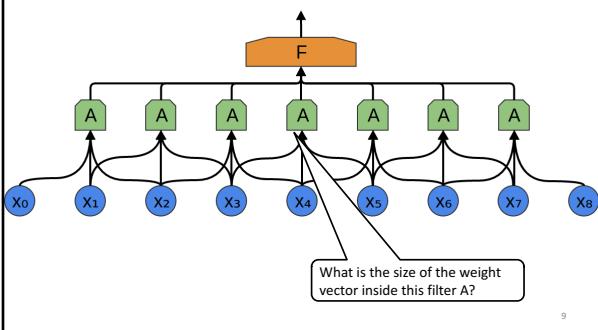
---

---

---

---

## Convolution filter of size k = 3



9

---

---

---

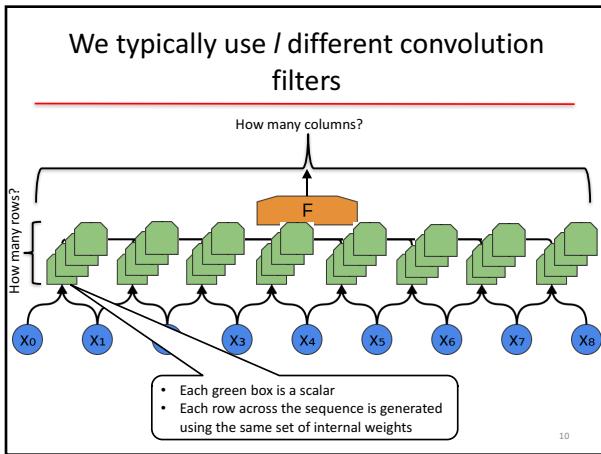
---

---

---

---

---



**Let's formalize things**

Consider a sequence of words  $w_{1:n} = w_1, \dots, w_n$ , each with their corresponding  $d_{\text{emb}}$ -dimensional word embedding  $E(w_i) = \mathbf{w}_i$ . A 1D convolution of width- $k$  works by moving a sliding-window of size  $k$  over the sentence, and applying the same "filter" to each window in the sequence, where a filter is a dot-product with a weight vector  $\mathbf{u}$ , which is often followed by a nonlinear activation function. Define the operator  $\oplus(\mathbf{w}_{i:i+k-1})$  to be the concatenation of the vectors  $\mathbf{w}_i, \dots, \mathbf{w}_{i+k-1}$ . The concatenated vector of the  $i$ th window is then  $x_i = \oplus(\mathbf{w}_{i:i+k-1}) = [\mathbf{w}_i; \mathbf{w}_{i+1}; \dots; \mathbf{w}_{i+k-1}], x_i \in \mathbb{R}^{k \cdot d_{\text{emb}}}$ .

We then apply the filter to each window-vector, resulting scalar values  $p_i$ :

$$p_i = g(x_i \cdot \mathbf{u}) \quad (13.1)$$

$$x_i = \oplus(\mathbf{w}_{i:i+k-1}) \quad (13.2)$$

$p_i \in \mathbb{R} \quad x_i \in \mathbb{R}^{k \cdot d_{\text{emb}}} \quad \mathbf{u} \in \mathbb{R}^{k \cdot d_{\text{emb}}}$ ,

where  $g$  is a nonlinear activation.

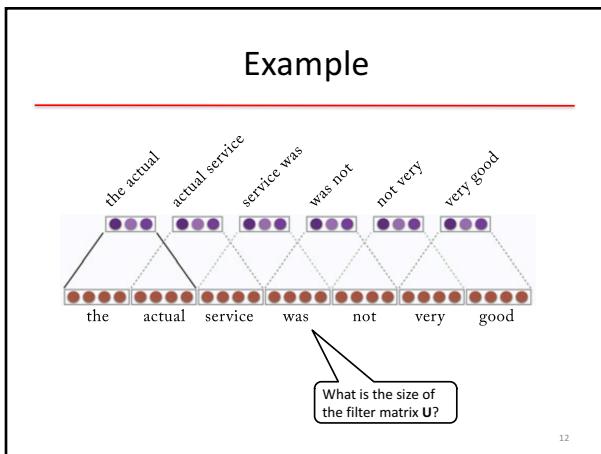
It is customary to use  $\ell$  different filters,  $\mathbf{u}_1, \dots, \mathbf{u}_\ell$ , which can be arranged into a matrix  $\mathbf{U}$ , and a bias vector  $\mathbf{b}$  is often added:

$$p_i = g(x_i \cdot \mathbf{U} + \mathbf{b}) \quad (13.3)$$

$$p_i \in \mathbb{R}^\ell \quad x_i \in \mathbb{R}^{k \cdot d_{\text{emb}}} \quad \mathbf{U} \in \mathbb{R}^{k \cdot d_{\text{emb}} \times \ell} \quad \mathbf{b} \in \mathbb{R}^\ell$$

Each vector  $p_i$  is a collection of  $\ell$  values that represent (or summarize) the  $i$ th window. Ideally, each dimension captures a different kind of indicative information.

11



## Vector pooling

- Applying the convolution over the text results in  $m$  vectors  $\mathbf{p}_{1:m}$ , each  $\mathbf{p}_i$  a vector of size  $l$ .
- These vectors are then combined (pooled) into a single vector  $\mathbf{c}$  of size  $l$ , representing the entire sequence.
- Why?

13

---



---



---



---



---



---



---

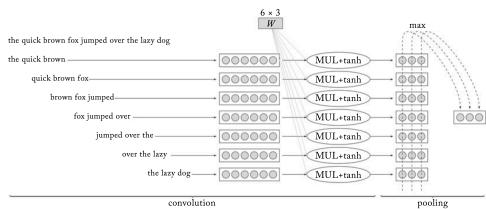


---



---

## Max pooling



14

---



---



---



---



---



---



---



---



---

## Max and average pooling

- Max pooling
  - $c_{[j]} = \max_{1 < i \leq m} p_i_{[j]} \quad \forall j \in [1, \ell]$
- Average pooling
  - $c = \frac{1}{m} \sum_{i=1}^m p_i$

15

---



---



---



---



---



---



---



---



---

## Practice Exercise

- What vector do you obtain after applying a sigmoid convolution filter of size  $k = 2$ , and  $l = 2$  with the weights  $\{ \{1, 2, 3, 4\}, \{2, 1, 1, 2\} \}$ , together with max pooling, over the following sentence, where each word has an embedding of size 2?
  - W1: {1, 2}
  - W2: {2, 2}
  - W3: {1, 0}
  - W4: {-1, -2}

16

---



---



---



---



---



---

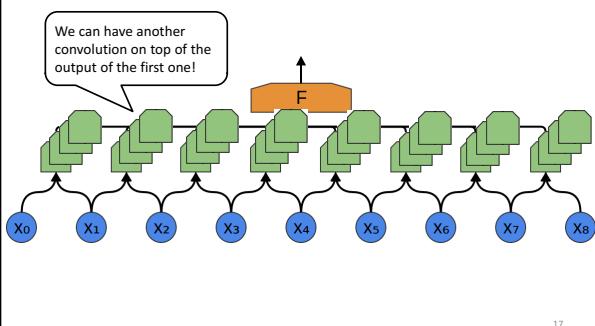


---



---

## Hierarchical convolutions



17

---



---



---



---



---



---

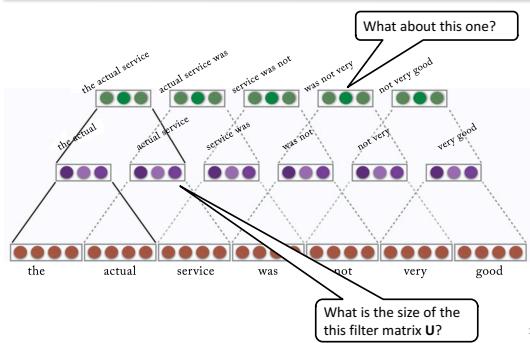


---



---

## Example



18

---



---



---



---



---



---



---



---

## Take-away

---

- Convolutional neural networks
- Recurrent neural networks (RNN)
- Long short-term memory networks

19

---

---

---

---

---

---

---

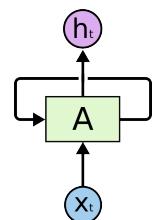
---

---

---

## RNNs have loops

---



"Rolled" representation

20

---

---

---

---

---

---

---

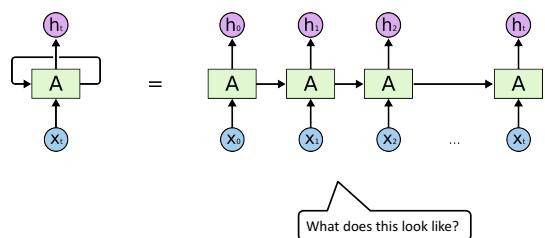
---

---

---

## Unrolled RNNs

---



21

---

---

---

---

---

---

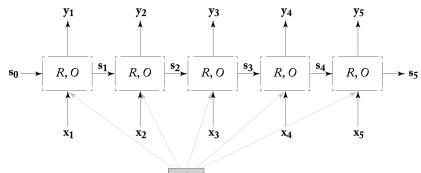
---

---

---

---

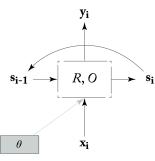
## Let's formalize things



22

## Let's formalize things

$$\begin{aligned} \text{RNN}^*(x_{1:n}; s_0) &= y_{1:n} \\ y_i &= O(s_i) \\ s_i &= R(s_{i-1}, x_i) \\ x_i &\in \mathbb{R}^{d_{in}}, \quad y_i \in \mathbb{R}^{d_{out}}, \quad s_i \in \mathbb{R}^{f(d_{out})}. \end{aligned}$$



23

## Simple RNN

$$\begin{aligned} s_i &= R_{\text{SRNN}}(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b) \\ y_i &= O_{\text{SRNN}}(s_i) = s_i \\ s_i, y_i &\in \mathbb{R}^{d_s}, \quad x_i \in \mathbb{R}^{d_x}, \quad W^x \in \mathbb{R}^{d_x \times d_s}, \quad W^s \in \mathbb{R}^{d_s \times d_s}, \quad b \in \mathbb{R}^{d_s}. \end{aligned}$$

- Alternative but equivalent formulation:

$$\begin{aligned} s_i &= R_{\text{SRNN}}(x_i, s_{i-1}) = g([s_{i-1}; x_i]W + b) \\ y_i &= O_{\text{SRNN}}(s_i) = s_i \\ s_i, y_i &\in \mathbb{R}^{d_s}, \quad x_i \in \mathbb{R}^{d_x}, \quad W \in \mathbb{R}^{(d_x+d_s) \times d_s}, \quad b \in \mathbb{R}^{d_s}. \end{aligned}$$

24

## Training

- RNNs are just very deep neural networks
    - What is the problem in this scenario?
  - Training is the same: backpropagation *through time* (BPTT)

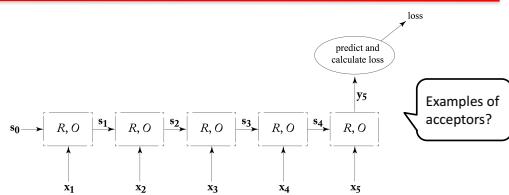
25

Unlike MEMMs, RNNs *do* encode the whole sequence

$$\begin{aligned}
 s_4 &= R(s_3, x_4) \\
 &= R(\overbrace{R(s_2, x_3)}^{s_3}, x_4) \\
 &= R(R(\overbrace{R(s_1, x_2)}^{s_2}, x_3), x_4) \\
 &= R(R(R(\overbrace{R(s_0, x_1)}^{s_1}, x_2), x_3), x_4).
 \end{aligned}$$

26

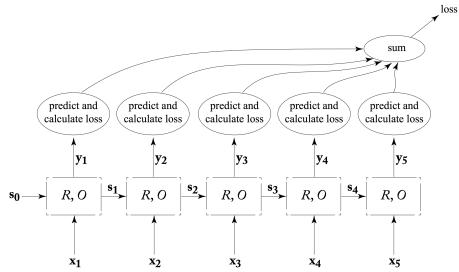
## Common RNNs: Acceptor, Encoder



- Encoder: acceptor + follow up model
  - Encoder example: document summarizer
    - RNN to generate an encoding of the whole sequence
    - Separate model to select which sentences/words to include in the summary

27

## Common RNNs: Transducer



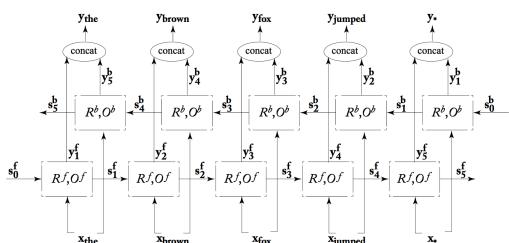
28

## Practice Exercise

- Find two examples each of acceptors, encoders, and transducers that are *not* in NLP
  - Explain how you would implement them in a RNN

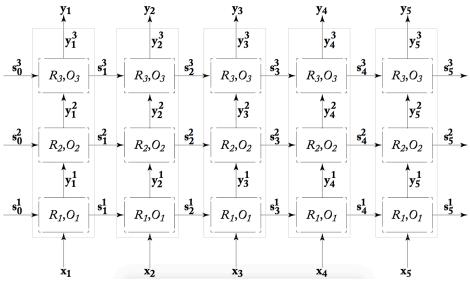
29

## Bidirectional RNNs



30

## Multi-layer (stacked) RNNs



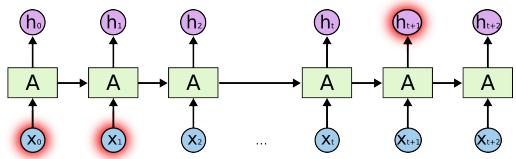
31

## Take-away

- Convolutional neural networks
- Recurrent neural networks
- Long short-term memory networks

32

## The problem with (vanilla) RNNs



- Vanilla RNNs do encode context, but poorly...
  - The “make something up” example
  - Or, a language model example: “I grew up in France... I speak fluent French”
- Plus, the vanishing gradient problem

33

## LSTMs solve these problems with two advantages

1. Additive (rather than multiplicative) architectures
  - Mitigate vanishing gradient problem
2. They learn what to remember and what to forget from the previous elements
  - Can remember longer contexts better
- Both advantages are implemented using gates
  - Hence this family of architectures is called gated architectures

34

---



---



---



---



---



---

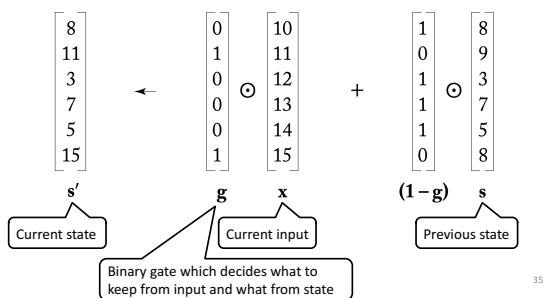


---



---

## Binary gates



35

---



---



---



---



---



---



---



---

## In practice: differentiable gates

- Arbitrary real numbers, instead of 0s and 1s
  - Coupled with a sigmoid, so the values to be used are between 0 and 1
- These are differentiable, and are trained with the rest of the network!
  - They are just a component in our new custom architecture

36

---



---



---



---



---



---



---



---

## LSTM intuition

- LSTMs split the state  $s$  into two components:
  - Memory cell, or cell state ( $c$ )
    - A “conveyor belt” that lets information flow along the entire sequence
    - Designed to remember/forget across the sequence
    - Controlled through differentiable gates
  - Working memory, or hidden state ( $h$ )
    - Combines the current input with the current memory cell and the previous working memory

37

---



---



---



---



---



---

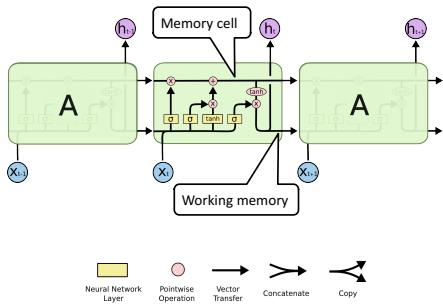


---



---

## LSTM overview



38

---



---



---



---



---



---

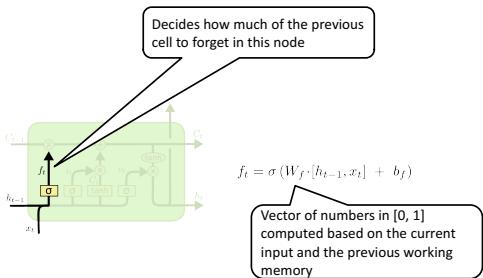


---



---

## LSTM walkthrough: forget gate



39

---



---



---



---



---



---



---



---

## Practice exercise

- Given:
  - $h_0 = \{1, 2, 1\}$
  - $x_1 = \{2, 1, 3\}$
  - $W_f$  has dimension  $6 \times 3$ , with values all 1
  - $b_f = 2$
- What is the dimension and the values in the  $f_t$  vector?

40

---



---



---



---



---



---

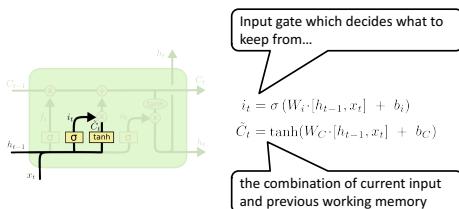


---



---

## LSTM walkthrough: input gate



41

---



---



---



---



---



---

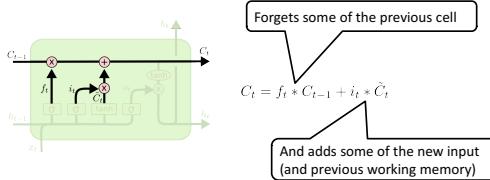


---



---

## LSTM walkthrough: cell state



42

---



---



---



---



---



---

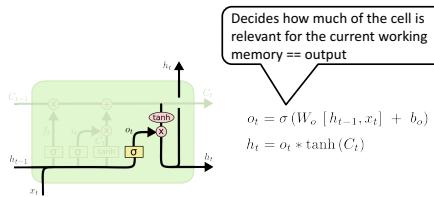


---



---

## LSTM walkthrough: output gate



43

---



---



---



---



---



---

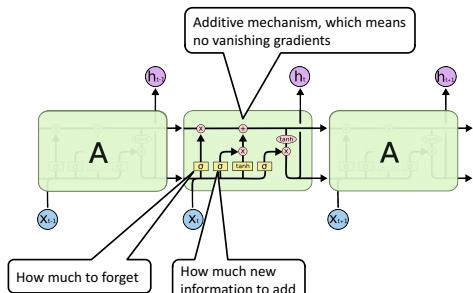


---



---

## LSTM summary



44

---



---



---



---



---



---



---



---

## Alternative formulation for LSTMs

$$\begin{aligned}
 s_j &= R_{\text{LSTM}}(s_{j-1}, x_j) = [c_j; h_j] \\
 c_j &= f \odot c_{j-1} + i \odot z \\
 h_j &= o \odot \tanh(c_j) \\
 i &= \sigma(x_j W^{xi} + h_{j-1} W^{hi}) \\
 f &= \sigma(x_j W^{xf} + h_{j-1} W^{hf}) \\
 o &= \sigma(x_j W^{xo} + h_{j-1} W^{ho}) \\
 z &= \tanh(x_j W^{xz} + h_{j-1} W^{hz})
 \end{aligned}$$

$$y_j = O_{\text{LSTM}}(s_j) = h_j$$

$$s_j \in \mathbb{R}^{2 \cdot d_h}, \quad x_j \in \mathbb{R}^{d_x}, \quad c_j, h_j, i, f, o, z \in \mathbb{R}^{d_h}, \quad W^{x \circ} \in \mathbb{R}^{d_x \times d_h}, \quad W^{h \circ} \in \mathbb{R}^{d_h \times d_h}$$

45

---



---



---



---



---



---



---



---

## LSTMs advantages and disadvantages

- They work really really well on many sequence problems in NLP and beyond
  - But they are kinda complicated...
  - Plus, they take a long time to train

46

Alternative: Gated Recurrent Units (GRU)

$$\begin{aligned}
s_j &= R_{\text{GRU}}(s_{j-1}, x_j) = (1 - z) \odot s_{j-1} + z \odot \tilde{s}_j \\
z &= \sigma(x_j W^{xz} + s_{j-1} W^{zs}) \\
r &= \sigma(x_j W^{xr} + s_{j-1} W^{zs}) \\
\tilde{s}_j &= \tanh(x_j W^{xs} + (r \odot s_{j-1}) W^{sg})
\end{aligned}$$

$$y_j = O_{\text{GRU}}(s_j) = s_j$$

$$s_i, \tilde{s}_i \in \mathbb{R}^{d_s}, \quad x_i \in \mathbb{R}^{d_x}, \quad z, r \in \mathbb{R}^{d_s}, \quad W^{x\circ} \in \mathbb{R}^{d_x \times d_s}, \quad W^{s\circ} \in \mathbb{R}^{d_s \times d_s}.$$

47

## Practice exercise

- Design an even simpler gated architecture, with a single gate, which keeps the core advantages of LSTMs and GRUs
    - Additive memory
    - Forget mechanism
  - Hint: start from the GRU and remove things

48

## Take-away

- 
- Convolutional neural networks
  - Recurrent neural networks
  - Long short-term memory networks

49

---

---

---

---

---

---

---

---

---

---