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

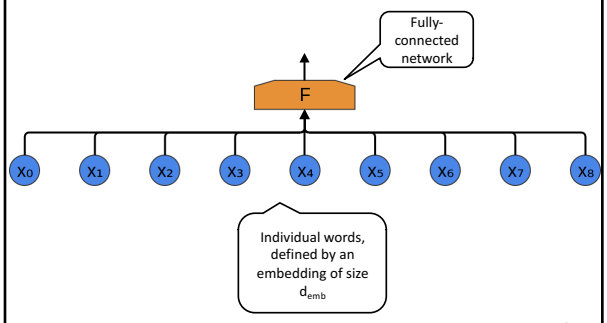
4

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.

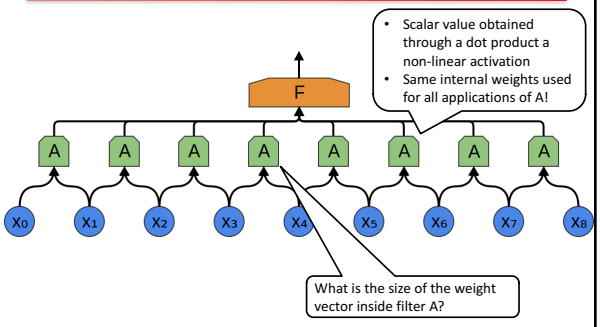
5

Bag-of-word feed-forward NN



6

After applying one filter A



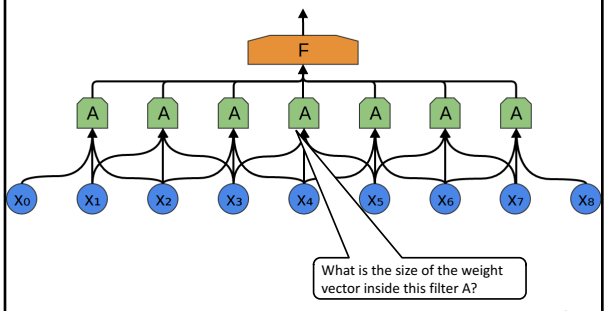
7

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}

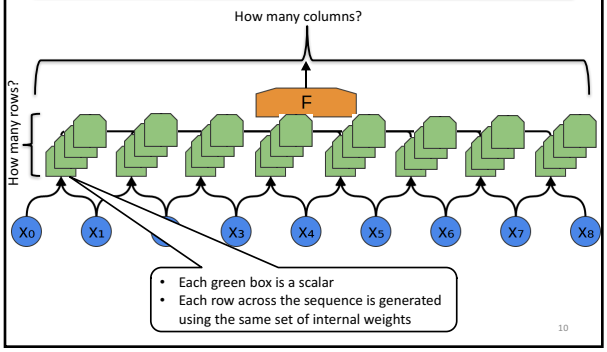
8

Convolution filter of size k = 3



9

We typically use / different convolution filters



10

Let's formalize things

Consider a sequence of words $w_{1:n} = w_1, \ldots, w_n$, each with their corresponding d_{emb} dimensional word embedding $\boldsymbol{E}_{[w_i]} = \boldsymbol{w_i}$. 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 $\boldsymbol{u}$, which is often followed by a nonlinear activation function. Define the operator $\oplus(\boldsymbol{w_{i:i+k-1}})$ to be the concatenation of the vectors $\boldsymbol{w_i}, \ldots, \boldsymbol{w_{i+k-1}}$. The concatenated vector of the ith window is then $\boldsymbol{x_i} = \oplus(\boldsymbol{w_{i:i+k-1}}) = [\boldsymbol{w_i}; \boldsymbol{w_{i+1}}; \ldots; \boldsymbol{w_{i+k-1}}], \boldsymbol{x_i} \in \mathbb{R}^{k \cdot d_{emb}}$.

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

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

$$\boldsymbol{x_i} = \oplus(\boldsymbol{w_{i:i+k-1}}) \quad (13.2)$$

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

where g is a nonlinear activation.

It is customary to use ℓ different filters, $\boldsymbol{u_1}, \ldots, \boldsymbol{u_\ell}$, which can be arranged into a matrix $\boldsymbol{U}$, and a bias vector $\boldsymbol{b}$ is often added:

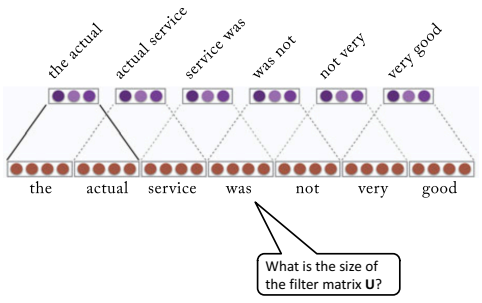
$$\boldsymbol{p_i} = g(\boldsymbol{x_i} \cdot \boldsymbol{U} + \boldsymbol{b}) \quad (13.3)$$

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

Each vector $\boldsymbol{p_i}$ is a collection of ℓ values that represent (or summarize) the ith window. Ideally, each dimension captures a different kind of indicative information.

11

Example



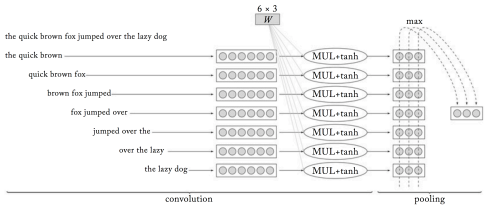
12

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 **c** of size *l*, representing the entire sequence.
  - Why?

13

Max pooling



14

Max and average pooling

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

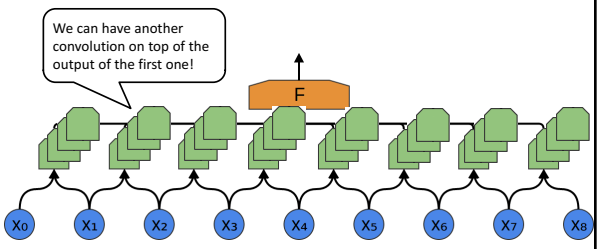
15

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}

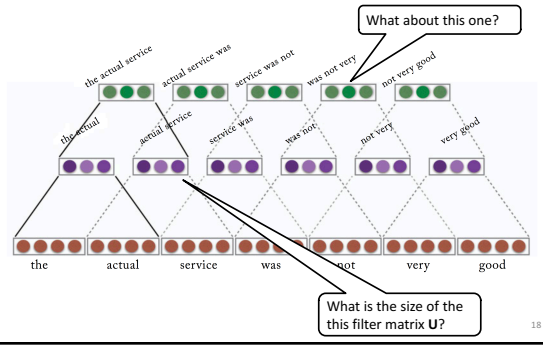
16

Hierarchical convolutions



17

Example



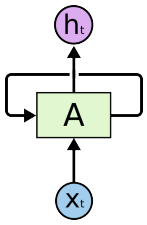
18

Take-away

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

19

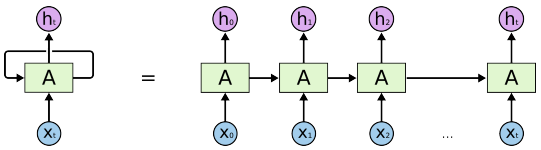
RNNs have loops



"Rolled" representation

20

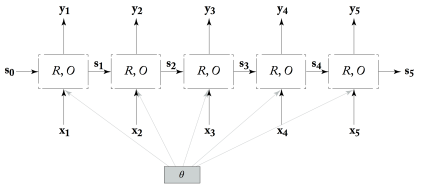
Unrolled RNNs



What does this look like?

21

Let's formalize things

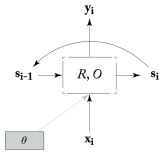


22

Let's formalize things

$$\text{RNN}^*(\boldsymbol{x_{1:n}}; \boldsymbol{s_0}) = \boldsymbol{y_{1:n}}$$
$$\boldsymbol{y_i} = O(\boldsymbol{s_i})$$
$$\boldsymbol{s_i} = R(\boldsymbol{s_{i-1}}, \boldsymbol{x_i})$$

$\boldsymbol{x_i} \in \mathbb{R}^{d_{in}}$, $\boldsymbol{y_i} \in \mathbb{R}^{d_{out}}$, $\boldsymbol{s_i} \in \mathbb{R}^{f(d_{out})}$.



23

Simple RNN

$$\boldsymbol{s_i} = R_{\text{SRNN}}(\boldsymbol{x_i}, \boldsymbol{s_{i-1}}) = g(\boldsymbol{s_{i-1}}\boldsymbol{W^s} + \boldsymbol{x_i}\boldsymbol{W^x} + \boldsymbol{b})$$
$$\boldsymbol{y_i} = O_{\text{SRNN}}(\boldsymbol{s_i}) = \boldsymbol{s_i}$$

$\boldsymbol{s_i}, \boldsymbol{y_i} \in \mathbb{R}^{d_s}$, $\boldsymbol{x_i} \in \mathbb{R}^{d_x}$, $\boldsymbol{W^x} \in \mathbb{R}^{d_x \times d_s}$, $\boldsymbol{W^s} \in \mathbb{R}^{d_s \times d_s}$, $\boldsymbol{b} \in \mathbb{R}^{d_s}$.

- Alternative but equivalent formulation:

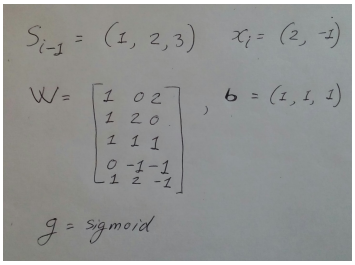
$$\boldsymbol{s_i} = R_{\text{SRNN}}(\boldsymbol{x_i}, \boldsymbol{s_{i-1}}) = g([\boldsymbol{s_{i-1}}; \boldsymbol{x_i}]\boldsymbol{W} + \boldsymbol{b})$$
$$\boldsymbol{y_i} = O_{\text{SRNN}}(\boldsymbol{s_i}) = \boldsymbol{s_i}$$

$\boldsymbol{s_i}, \boldsymbol{y_i} \in \mathbb{R}^{d_s}$, $\boldsymbol{x_i} \in \mathbb{R}^{d_x}$, $\boldsymbol{W} \in \mathbb{R}^{(d_x + d_s) \times d_s}$, $\boldsymbol{b} \in \mathbb{R}^{d_s}$.

24

Exercise

- Given the simple RNN below, in the alternative notation, and the given state (s_{i-1}) and input (x_i), what is the current output y_i?



25

Training

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

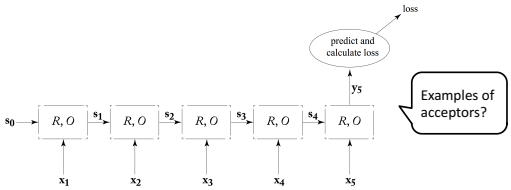
26

Unlike MEMMs, RNNs *do* encode the whole sequence

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

27

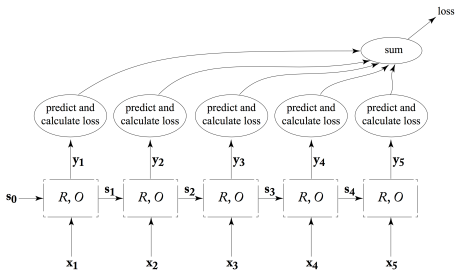
Common RNNs: Acceptor, Encoder



- 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

28

Common RNNs: Transducer



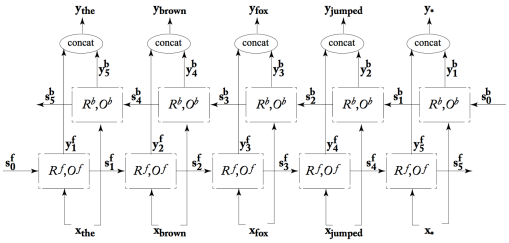
29

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

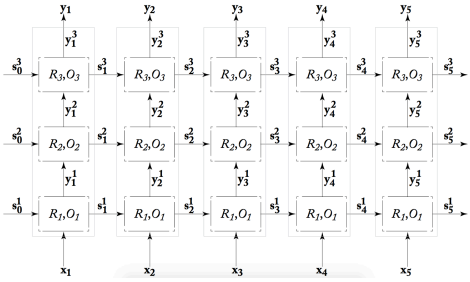
30

Bidirectional RNNs



31

Multi-layer (stacked) RNNs



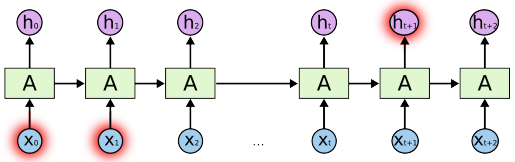
32

Take-away

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

33

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

34

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

35

Binary gates

$$\begin{bmatrix}8\\11\\3\\7\\5\\15\end{bmatrix} \leftarrow \begin{bmatrix}0\\1\\0\\0\\0\\1\end{bmatrix} \odot \begin{bmatrix}10\\11\\12\\13\\14\\15\end{bmatrix} + \begin{bmatrix}1\\0\\1\\1\\1\\0\end{bmatrix} \odot \begin{bmatrix}8\\9\\3\\7\\5\\8\end{bmatrix}$$

$\mathbf{s}'$ — Current state

$\mathbf{g}$ — Binary gate which decides what to keep from input and what from state

$\mathbf{x}$ — Current input

$(\mathbf{1}-\mathbf{g})$

$\mathbf{s}$ — Previous state

36

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

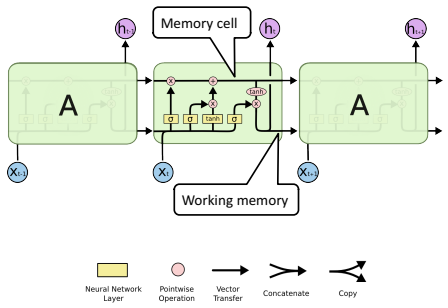
37

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

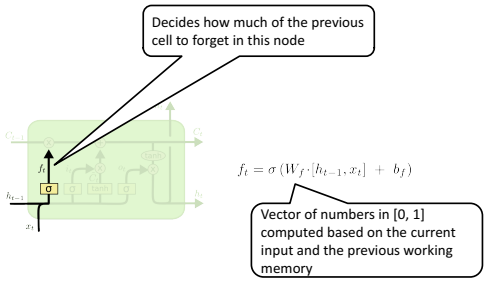
38

LSTM overview



39

LSTM walkthrough: forget gate



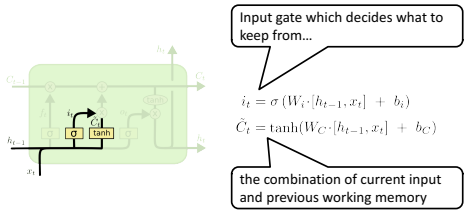
40

Practice exercise

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

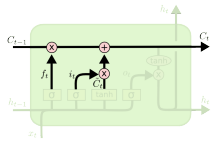
41

LSTM walkthrough: input gate



42

LSTM walkthrough: cell state



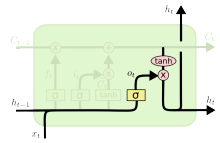
Forgets some of the previous cell

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

And adds some of the new input (and previous working memory)

43

LSTM walkthrough: output gate



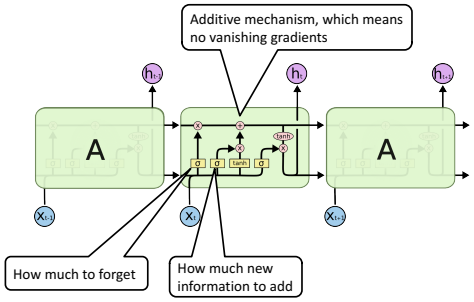
Decides how much of the cell is relevant for the current working memory == output

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

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

44

LSTM summary



45

Alternative formulation for LSTMs

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

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

$s_j \in \mathbb{R}^{2 \cdot d_h}$, $x_i \in \mathbb{R}^{d_x}$, $c_j, h_j, i, f, o, z \in \mathbb{R}^{d_h}$, $W^{xo} \in \mathbb{R}^{d_x \times d_h}$, $W^{ho} \in \mathbb{R}^{d_h \times d_h}$.

46

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

47

Alternative: Gated Recurrent Units (GRU)

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

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

$s_j, \tilde{s_j} \in \mathbb{R}^{d_s}$, $x_i \in \mathbb{R}^{d_x}$, $z, r \in \mathbb{R}^{d_s}$, $W^{xo} \in \mathbb{R}^{d_x \times d_s}$, $W^{so} \in \mathbb{R}^{d_s \times d_s}$.

48

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

49

Take-away

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

50