Part-of-Speech Tagging: Neural Methods

Katharina Kann — CSCI/LING5832

Word Classes: Parts of Speech

- Traditional parts of speech:
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.

Word Classes: Parts of Speech

- Traditional parts of speech:
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
  - Lots of names for this notion: Part of speech, lexical category, word class, morphological class, lexical tag...

Word Classes: Parts of Speech

- Traditional parts of speech:
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
  - Lots of names for this notion: Part of speech, lexical category, word class, morphological class, lexical tag...
  - Lots of debate within linguistics about the number, nature, and universality of these categories

Word Classes: Parts of Speech

- Sources of evidence
  - Morphological evidence
    - walk, walking, walked, walks
    - probably a verb

Word Classes: Parts of Speech

- Sources of evidence
  - Morphological evidence
    - walk, walking, walked, walks
    - probably a verb
  - Distributional evidence
    - The crash, A crash, Two crashes, The big crash…
    - probably a noun

Penn Treebank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	“to”	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential ‘there’	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

POS Tagging

- The process of assigning a part of speech or lexical class marker to each word in a text.
- Often a useful first step in an NLP pipeline.

POS Tagging

- The process of assigning a part of speech or lexical class marker to each word in a text.
- Often a useful first step in an NLP pipeline.
- Fast and accurate taggers are widely available for many languages.
  - (Now we will learn how to make one!)

POS Tagging

- The process of assigning a part of speech or lexical class marker to each word in a text.
- The is our first example of a **sequence labeling** task:
  - Assigning a category label to each element of a sequence.

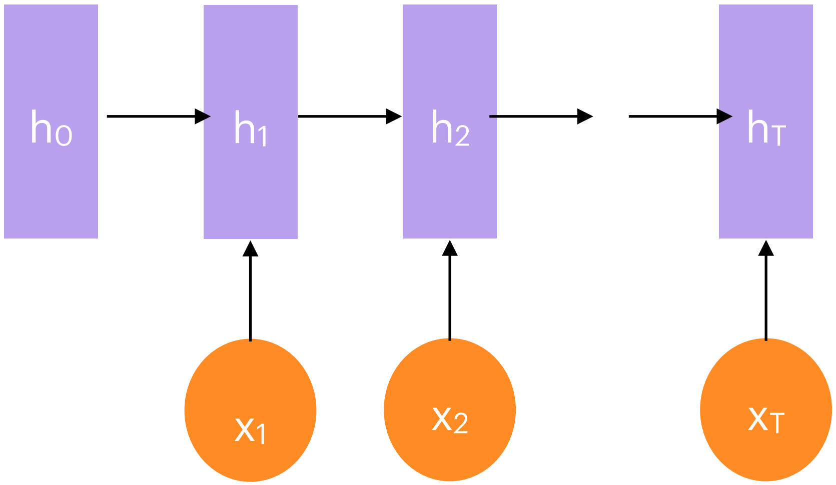
Recap from Lecture 07: More on RNNs

Recurrent Neural Networks

Input: $x_1, x_2, \ldots, x_T$

$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$

$y = argmax(\text{softmax}(W_y h_T))$

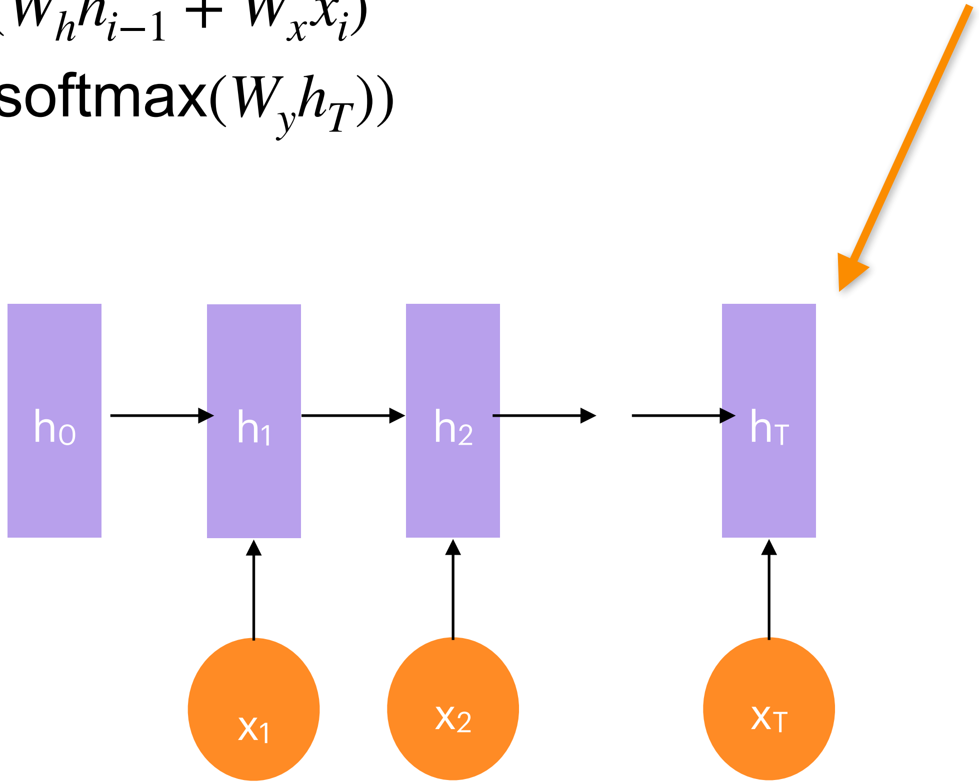


Recurrent Neural Networks

Input: $x_1, x_2, \ldots, x_T$

$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$

$y = argmax(\text{softmax}(W_y h_T))$



Problems with Recurrent Neural Networks

- During backpropagation, since RNNs multiply the same matrix over and over, gradients vanish or explode.

Problems with Recurrent Neural Networks

- During backpropagation, since RNNs multiply the same matrix over and over, gradients vanish or explode.
- Cannot capture long-term dependencies (or learn much) with uninformative gradients!

Problems with Recurrent Neural Networks

- During backpropagation, since RNNs multiply the same matrix over and over, gradients vanish or explode.
- Cannot capture long-term dependencies (or learn much) with uninformative gradients!
- History gets forgotten over time.

Problems with Recurrent Neural Networks

The flights the airline was cancelling were full.

Problems with Recurrent Neural Networks

The flights the airline was cancelling were full.

Problems with Recurrent Neural Networks

The flights the airline was cancelling were full.

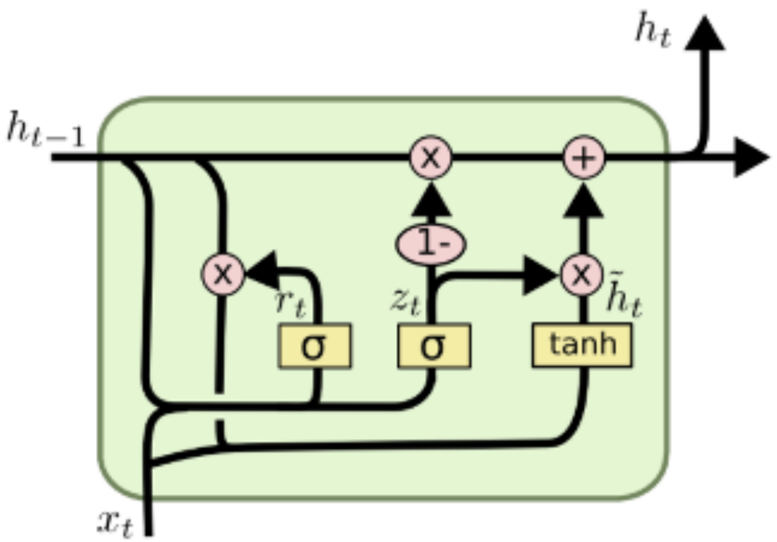
Solutions

- **Gradient Clipping**: Normalize the gradient of a parameter vector when its L2 norm reaches a certain value.

Solutions

- **Gradient Clipping**: Normalize the gradient of a parameter vector when its L2 norm reaches a certain value.
- Explicit **memory cell** with which a model can keep important long-term information.

Gated Recurrent Units (GRUs)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

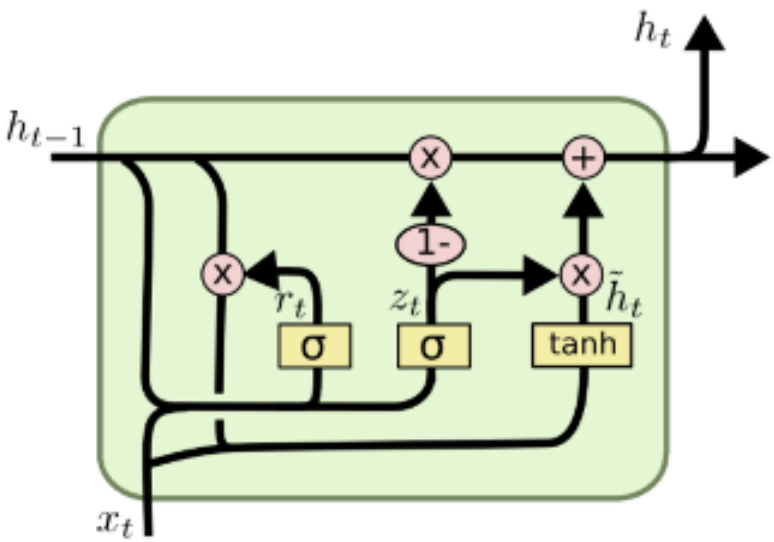
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Units (GRUs)



$$\boxed{z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

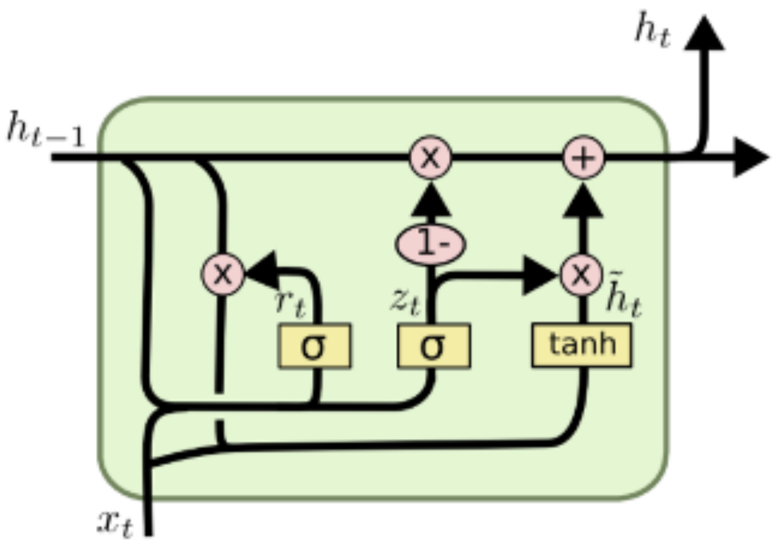
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Update gate: controls how much of the previous hidden state to update

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Units (GRUs)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

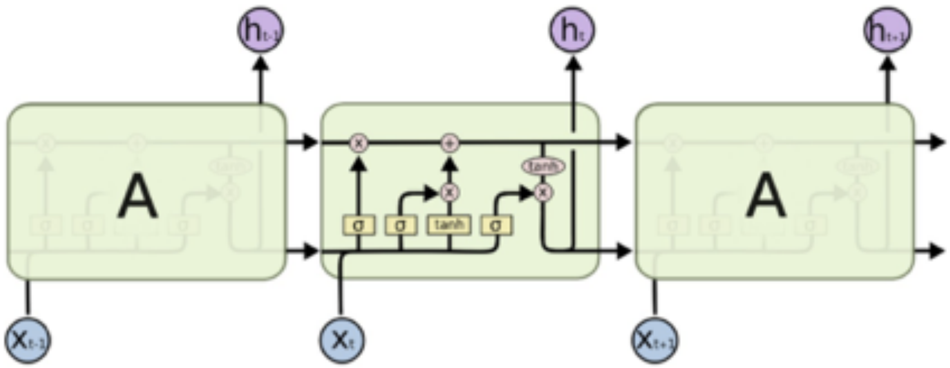
Reset gate: controls how much of the previous hidden state to keep

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory Networks (LSTMs)

- First proposed by Hochreiter and Schmidhuber (1997).
- Similar to the GRU, but some more parameters.
- Performance is usually extremely similar.

Long Short-Term Memory Networks (LSTMs)

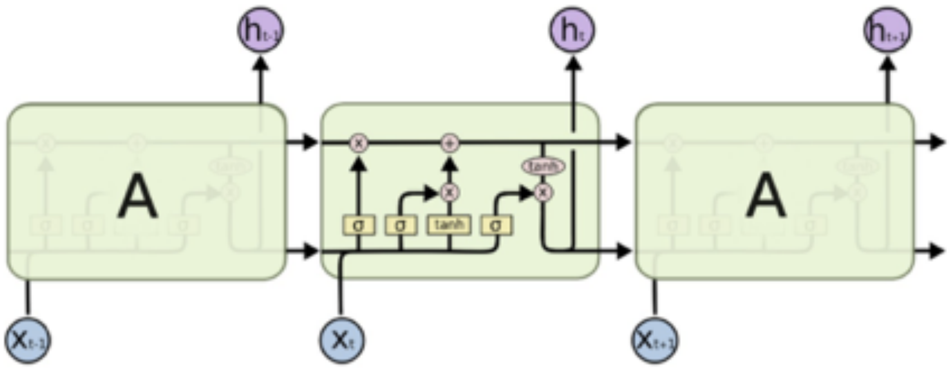


The repeating module in an LSTM contains four interacting layers.

$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/, J&M, Ch.9

Long Short-Term Memory Networks (LSTMs)



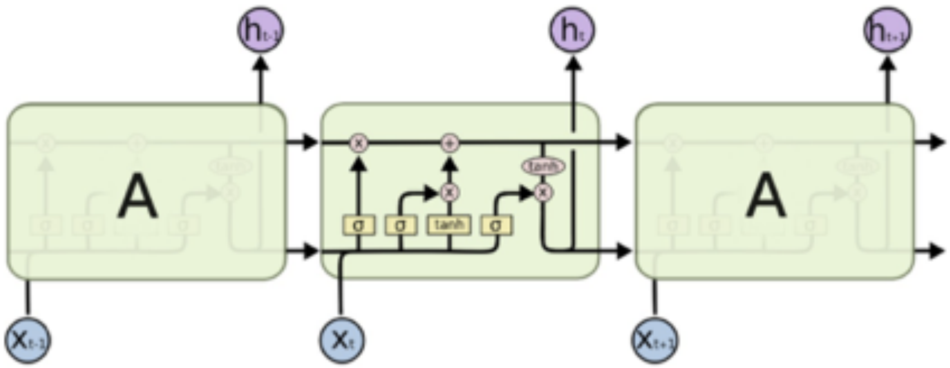
The repeating module in an LSTM contains four interacting layers.

$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Add gate

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/, [J&M, Ch.9]()

Long Short-Term Memory Networks (LSTMs)



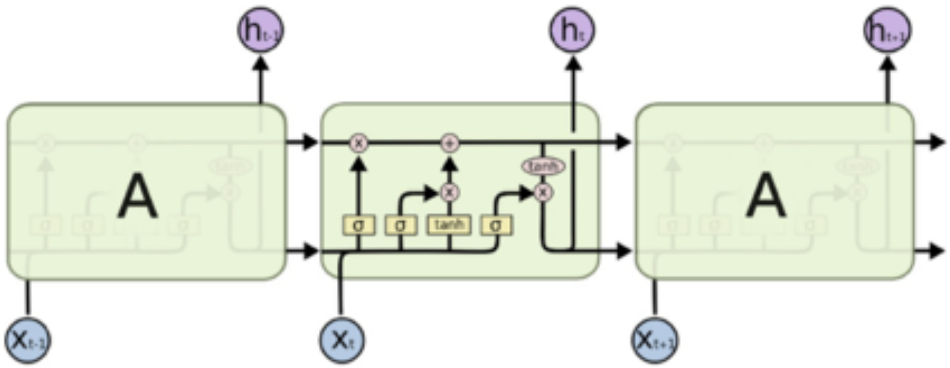
The repeating module in an LSTM contains four interacting layers.

$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Forget gate

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/, [J&M, Ch.9]()

Long Short-Term Memory Networks (LSTMs)



The repeating module in an LSTM contains four interacting layers.

$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Output gate

Visual Obtained from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/, [J&M, Ch.9]()

How Do We Get a Sentence Representation?

- Apply the RNN to all words in a sequence.
- Use the last hidden vector as the representation of your sentence.
- The sentence representation is often called *h*.

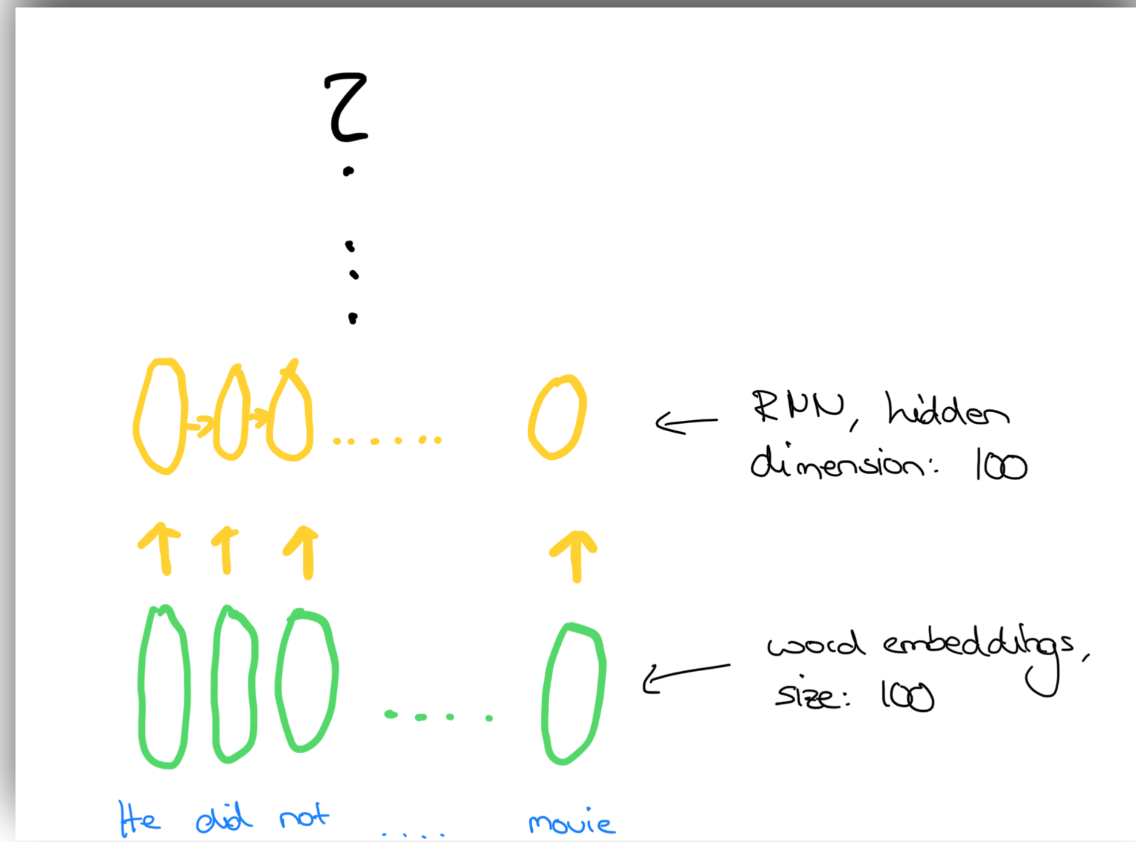
Bidirectional Recurrent Neural Networks

- Most prominent one: BiLSTM
- Idea: Read the input from both sides.
  - Beginning isn’t forgotten!
- For the final representation:
  - Concatenate the last hidden vectors of the two individual RNNs.
  - (What is the size of the final sentence representation computed by a bidirectional RNN?)

In-Class Exercise (Exercise 3 in Lecture 07)

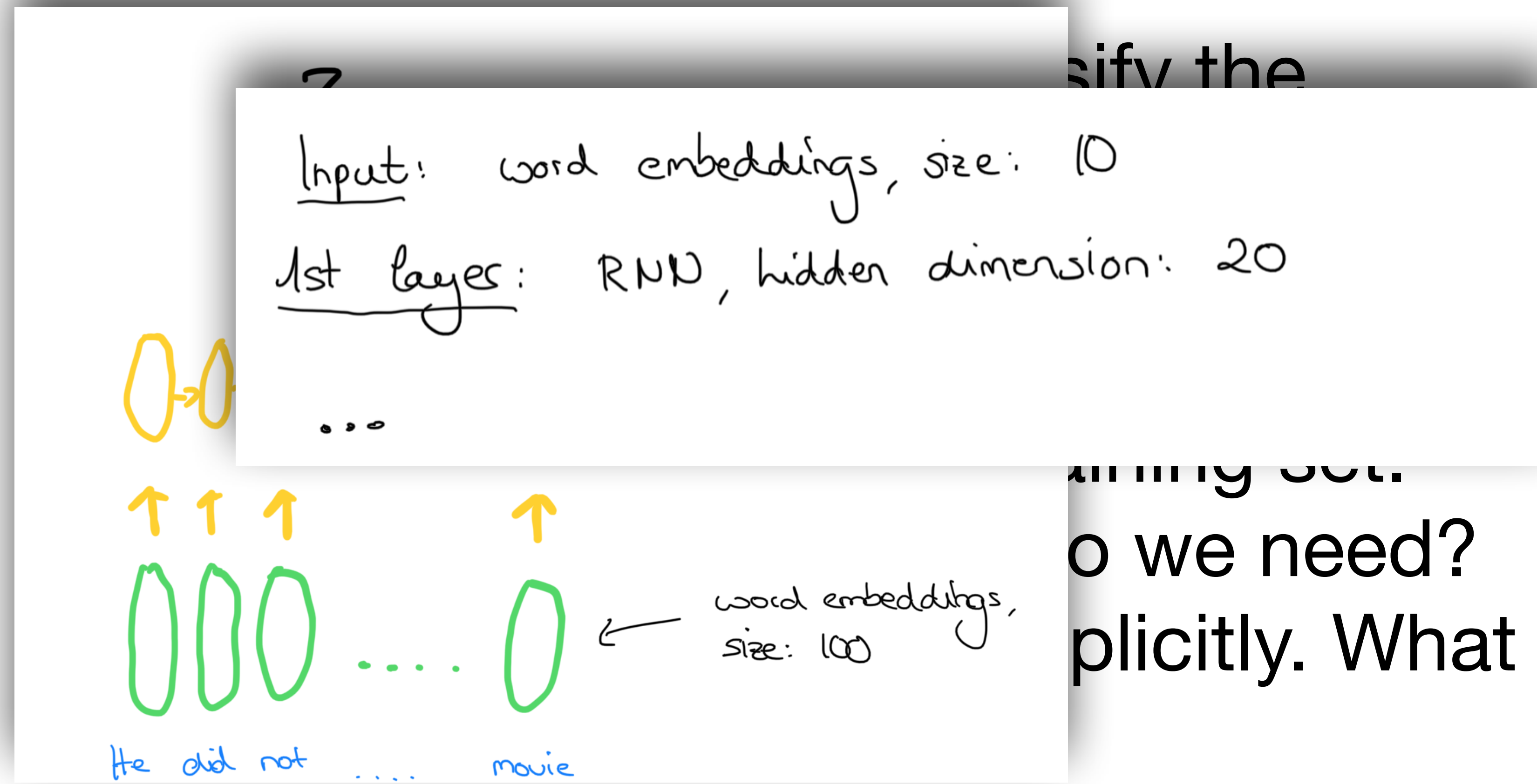
- Assume that we want to classify the sentiment of tweets into *positive*, *neutral*, and *negative*.
- Design a neural network that can learn this task, given a large-enough training set.
- Which preprocessing steps do we need?
- Write down all dimensions explicitly. What is your output dimension?
- (Keep your architecture safe for later!)

In-Class Exercise (Exercise 3 in Lecture 07)



- sify the *tive*, *neutral*,
- can learn this aining set.
- o we need?
- plicitly. What
- (Keep your architecture safe for later!)

In-Class Exercise (Exercise 3 in Lecture 07)



- sify the
- aining set.
- o we need?
- plicitly. What
- (Keep your architecture safe for later!)

RNNs for Sequence Labeling

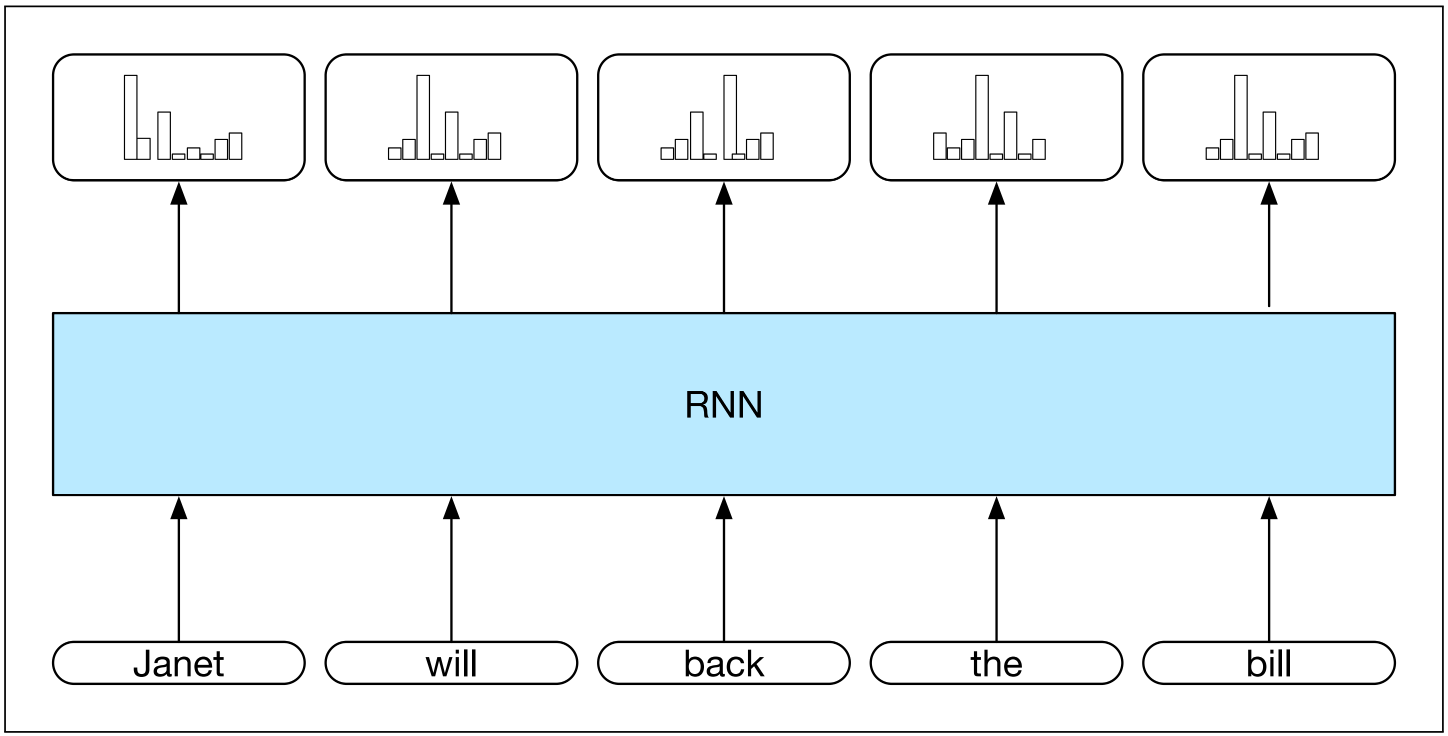
Sequence Labeling with RNNs

- For sequence labeling, we don’t exclusively use the last hidden state of the RNN
    - (Why?)

Sequence Labeling with RNNs

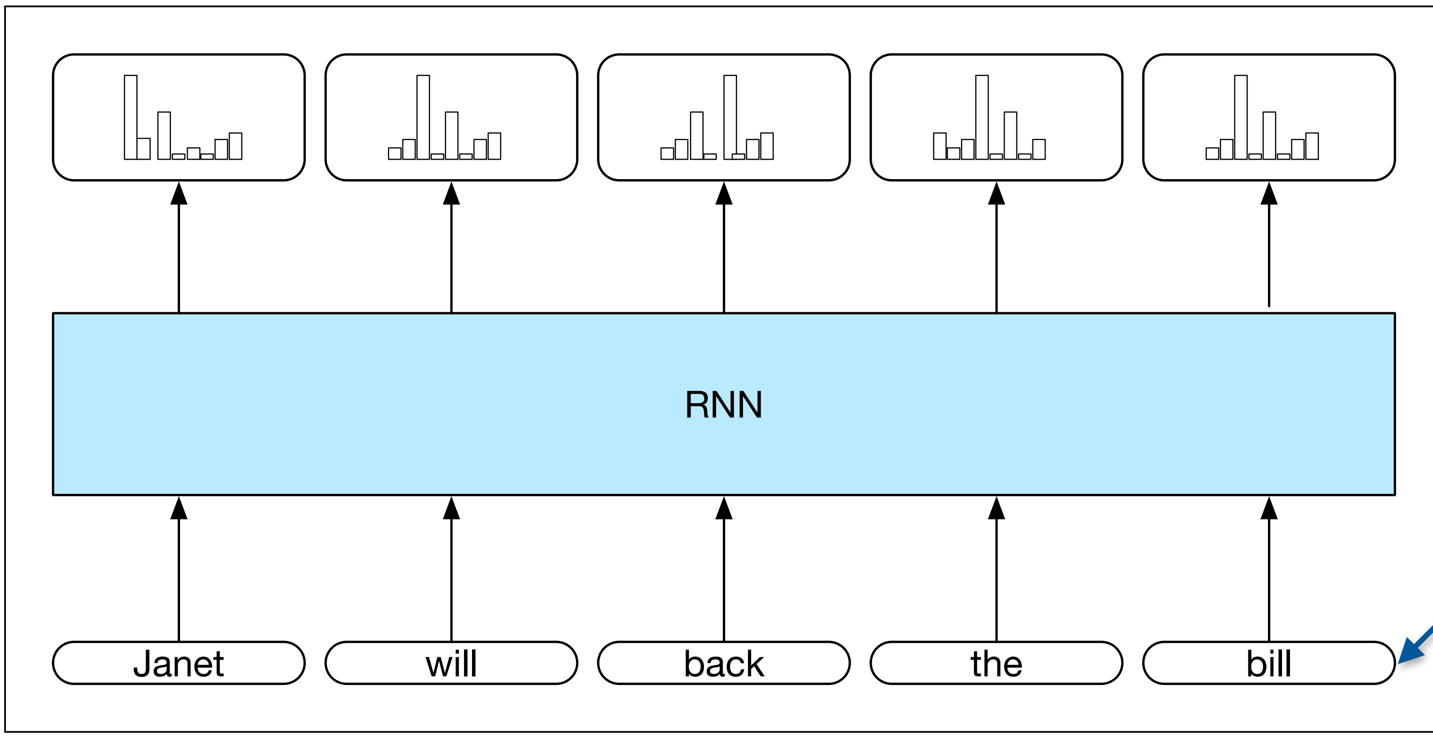
- For sequence labeling, we don't exclusively use the last hidden state of the RNN
  - (Why?)
- Instead, we assign a class to each hidden state
  - Use a feedforward layer and a sigmoid/softmax function

Sequence Labeling with RNNs



J&M, Ch.9

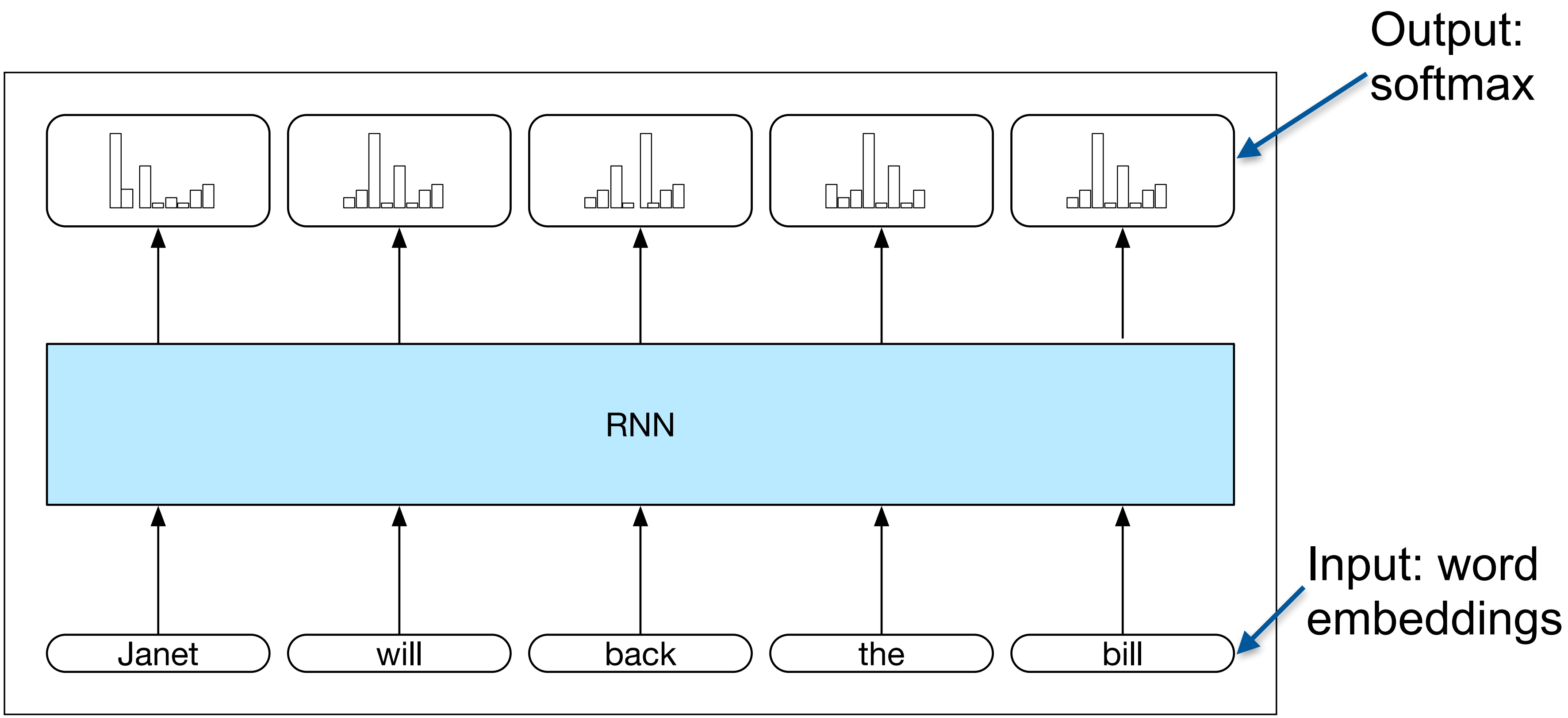
Sequence Labeling with RNNs



Input: word embeddings

J&M, Ch.9

Sequence Labeling with RNNs



J&M, Ch.9

Example: Sequence Labeling with RNNs

Input: $x_1, x_2, \ldots, x_T$

$$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$$

$$y_i = argmax(\text{softmax}(W_y h_i))$$

Example: Sequence Labeling with RNNs

Input: $x_1, x_2, \ldots, x_T$

$$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$$

$$y_i = argmax(\text{softmax}(W_y h_i))$$

- (What would be the formula to predict the label for sequence labeling?)

Example: Sequence Labeling with RNNs

Input: $x_1, x_2, \ldots, x_T$

$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$

$y_i = argmax(\text{softmax}(W_y h_i))$

- **we like books**

- we = [1, 2], like = [0.5, 0.1], books = [0.2, 1]
- W_h = [[0.4]], W_x = [[0.1, 0.14]], W_y = [[0.3], [0.2], [0.01]], h_0 = [0]
- Compute the class of “we”!

Example: Sequence Labeling with RNNs

Input: $x_1, x_2, \ldots, x_T$

$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$

$y_i = argmax(\text{softmax}(W_y h_i))$

- **we like books**

- we = [1, 2], like = [0.5, 0.1], books = [0.2, 1]
- W_h = [[0.4]], W_x = [[0.1, 0.14]], W_y = [[0.3], [0.2], [0.01]], h_0 = [0]
- Compute the class of "we"!

- h_1 = sigmoid([0] +[0.1 + 0.28]) = sigmoid([0.38]) = 0.59
- y_1 = argmax(softmax([0.3*0.59, 0.2*0.59, 0.01 * 0.59])) = argmax(softmax([1.77, 1.18, 0.0059])) = argmax([0.58, 0.32, 0.1])

Example: Sequence Labeling with RNNs

Input: $x_1, x_2, \ldots, x_T$

$h_i =$ sigmoid$(W_h h_{i-1} + W_x x_i)$

$y_i = argmax($softmax$(W_y h_i))$

- **we like books**

- we = [1, 2], like = [0.5, 0.1], books = [0.2, 1]
- W_h = [[0.4]], W_x = [[0.1, 0.14]], W_y = [[0.3], [0.2], [0.01]], h_0 = [0]
- Compute the class of “we”!

- h_1 = sigmoid([0] +[0.1 + 0.28]) = sigmoid([0.38]) = 0.59
- y_1 = argmax(softmax([0.3*0.59, 0.2*0.59, 0.01 * 0.59])) = argmax(softmax([1.77, 1.18, 0.0059])) = argmax([$\boxed{0.58}$, 0.32, 0.1])

In-Class Exercise 2

Input: $x_1, x_2, \ldots, x_T$

$h_i = \text{sigmoid}(W_h h_{i-1} + W_x x_i)$

$y_i = argmax(\text{softmax}(W_y h_i))$

- **we like books**

- we = [1, 2], like = [0.5, 0.1], books = [0.2, 1]
- W_h = [[0.4]], W_x = [[0.1, 0.14]], W_y = [[0.3], [0.2], [0.01]], h_0 = [0]

- h_1 = sigmoid([0] +[0.1 + 0.28]) = sigmoid([0.38]) = 0.59
- y_1 = argmax(softmax([0.3*0.59, 0.2*0.59, 0.01 * 0.59])) = argmax(softmax([1.77, 1.18, 0.0059])) = argmax([0.58, 0.32, 0.1])

- Compute the classes of “like” and “books”!

Hierarchical LSTMs

Characters and POS Tagging

- Until now, we considered only word-level information for (neural) POS tagging
- Character information can be beneficial!

Characters and POS Tagging

- Until now, we considered only word-level information for (neural) POS tagging
- Character information can be beneficial!
    - employ**er**, runn**er**, work**er**, …

Characters and POS Tagging

- Until now, we considered only word-level information for (neural) POS tagging
- Character information can be beneficial!
  - employ**er**, runn**er**, work**er**, …
  - happi**ness**, sad**ness**, rich**ness**…

Characters and POS Tagging

- Until now, we considered only word-level information for (neural) POS tagging
- Character information can be beneficial!
  - employ**er**, runn**er**, work**er**, …
  - happi**ness**, sad**ness**, rich**ness**…
  - help**ful**, joy**ful**, cheer**ful**, …

Characters and POS Tagging

- We want to use this information for POS tagging
- How can we do that?

Characters and POS Tagging

- We want to use this information for POS tagging
- How can we do that?
  - RNN over characters

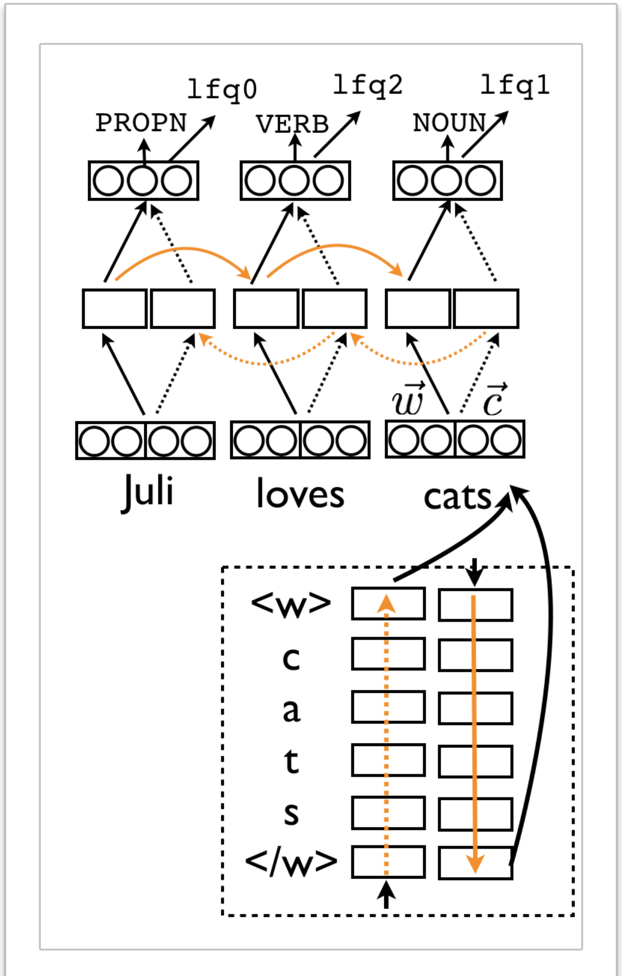
Characters and POS Tagging

- We want to use this information for POS tagging
- How can we do that?
  - RNN over characters
  - CNN over characters

Characters and POS Tagging

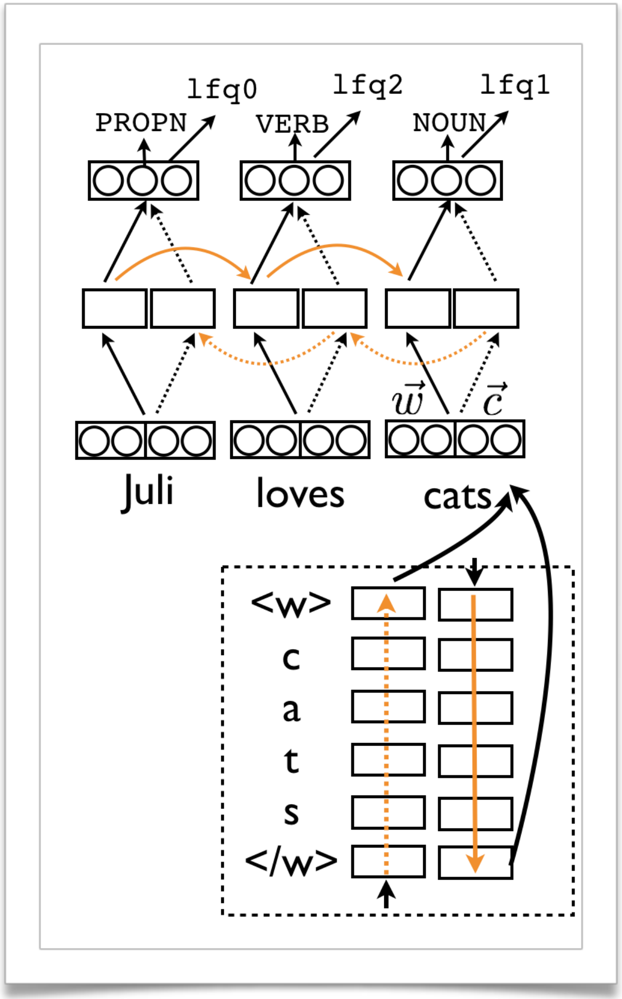
- We want to use this information for POS tagging
- How can we do that?
  - RNN over characters
  - CNN over characters
  - Two RNNs: one over characters and one over words!

Hierarchical RNN for POS Tagging



From [Plank et al., 2016]()

Hierarchical RNN for POS Tagging



From [Plank et al., 2016]

- Concatenate last hidden states of the character-level RNN with the word embeddings!

Other Sequence Labeling Tasks in NLP

Named Entity Recognition

- Named entities are spans in a text that correspond to names of people, places or organizations:
  - Barack Obama
  - CNN
  - Donald Trump Jr.
- Task: Find named entities in a given text
  - This required finding **spans** which correspond to named entities

IOB Encoding

- Label token with
  - B: beginning of span of interest
  - I: inside of span of interest
  - O: “outside”, not part of the span of interest

Named Entity Recognition

(9.13) *United cancelled the flight from Denver to San Francisco.*

B O O O O B O B I

J&M, Ch.9

Named Entity Recognition

(9.13)	*United*	*cancelled*	*the*	*flight*	*from*	*Denver*	*to*	*San*	*Francisco.*
	B	O	O	O	O	B	O	B	I

J&M, Ch.9

(9.14)	*United*	*cancelled*	*the*	*flight*	*from*	*Denver*	*to*	*San*	*Francisco.*
	B-ORG	O	O	O	O	B-LOC	O	B-LOC	I-LOC

J&M, Ch.9

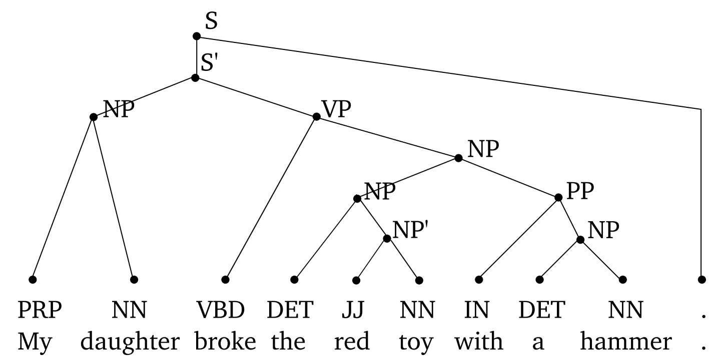
- We can encode more detail!

More Sequence Labeling Tasks

- Morphological tagging
- Language identification for code-switched text
- Constituency parsing
- …

More Sequence Labeling Tasks

- Morphological tagging
- Language identification for code-switched text
- Constituency parsing
- ...



Linearized tree (absolute scale):

3NP 2S' 3VP 5NP 6NP' 4NP 5PP 6NP 1S

Linearized tree (relative scale):

3NP -1S' 1VP 2NP 1NP' -2NP 1PP 1NP -5S

Linearized tree (simplified relative scale):

3NP -S' 1VP 2NP 1NP' -NP 1PP 1NP -S

Figure 2: An example of a binarized constituency tree, linearized both applying absolute and relative scales.

From [Gomez-Rodrıguez and Vilares (2018)](#)

Wrapping up

- Discussed today:
  - Recap: RNNs
  - RNNs for sequence labeling tasks
  - Hierarchical RNNs

- On Wednesday: Training neural networks