

Natural Language Processing

5. Recurrent neural networks

Katya Chernyak

CS HSE

October 2, 2018

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

Sequential data

① Time series

- ▶ Financial data analysis: stock market, commodities, Forex
- ▶ Healthcare: pulse rate, sugar level (from medical equipment and wearables)

② Text and speech: speech understanding, text generation

③ Spatiotemporal data

- ▶ Self-driving and object tracking
- ▶ Plate tectonic activity

④ Physics: jet identification

⑤ etc.

Sequence modelling I

Sequence labelling

- ① $\mathbf{x} = x_1, x_2, \dots, x_n$, $x_i \in V$, - objects
- ② $\mathbf{y} = y_1, y_2, \dots, y_n$, $y_i \in \{1, \dots, L\}$ - labels
- ③ $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$ – training data
- ④ exponential number of possible solutions : if $\text{length}(x) = n$, there are L^n possible solutions

Classification problem: $\gamma : \mathbf{x} \rightarrow \mathbf{y}$

- ① Speech recognition: x – spoken words, y – transcription
- ② Genome annotation: x – DNA, y – genes

Sequence modelling II

Sequence classification

- ① $\mathbf{x} = x_1, x_2, \dots, x_n$, $x_i \in V$, - objects
- ② $y \in \{1, \dots, L\}$ - labels
- ③ $\{(\mathbf{x}^{(1)}, y_1), (\mathbf{x}^{(2)}, y_2), \dots, (\mathbf{x}^{(m)}, y_m)\}$ – training data

Classification problem: $\gamma : \mathbf{x} \rightarrow y$

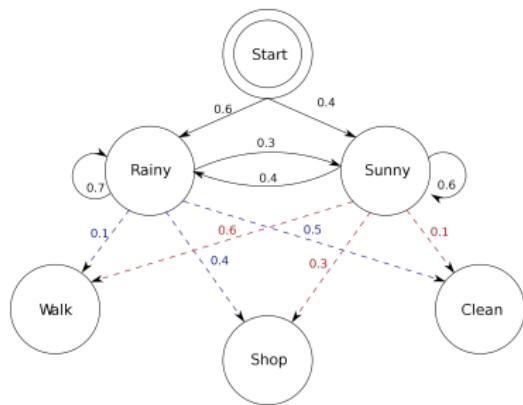
- ① Activity recognition: x – pulse rate, y – activity (walking, running, peace)
- ② Opinion mining: x – sentence, y – sentiment (positive, negative)
- ③ Trading: x – stock market, y – action (sell, buy, do nothing)

Traditional ML approaches to sequence modelling

- Hidden Markov Models (HMM)
- Conditional Random Fields (CRF)
- Local classifier: for each x define features, based on x_{-1}, x_+1 , etc, and perform classification n times

Problems:

- ① Markov assumption: fixed length history
- ② Computation complexity



1 Sequence modelling

2 Recurrent neural network

- Definition
- Training

3 Gated architectures

4 RNN generators

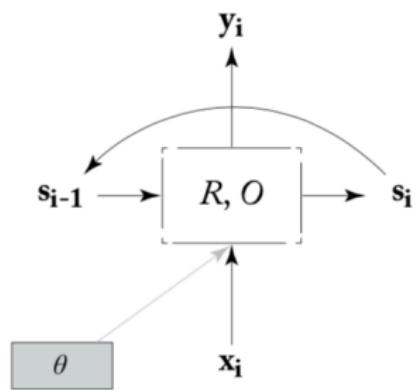
5 Bonus I: RecNN

6 Bonus II: HAN

7 Bonus III: IE

Recurrent neural network

- Input: sequence of vectors
- $x_{1:n} = x_1, x_2, \dots, x_n, x_i \in \mathbb{R}^{d_{in}}$
- Output: a single vector
 $y_n = RNN(x_{1:n}), y_n \in \mathbb{R}^{d_{out}}$
- For each prefix $x_{i:j}$ define an output vector y_i :
 $y_i = RNN(x_{1:i})$
- RNN^* is a function returning this sequence for input sequence $x_{1:n}$:
 $y_{1:n} = RNN^*(x_{1:n}), y_i \in \mathbb{R}^{d_{out}}$



Sequence modelling with RNN

① Sequence labelling

Produce an output y_i for each input RNN reads in. Put a dense layer on top of each output to predict the desired class of the input

$$p(l_j | \mathbf{x}_j) = \text{softmax}(RNN(\mathbf{x}_{1:j}) \times W + b)_{[j]}$$

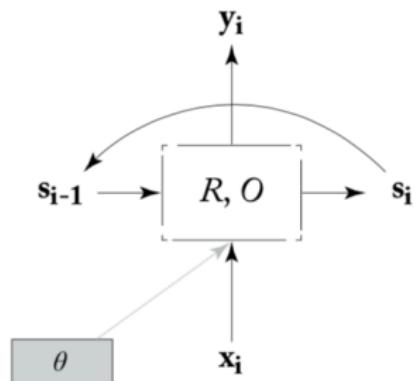
② Sequence classification

Put a dense layer on top of RNN to predict the desired class of the sequence after the whole sequence is processed

$$p(l_j | \mathbf{x}_{1:n}) = \text{softmax}(RNN(\mathbf{x}_{1:n}) \times W + b)_{[j]}$$

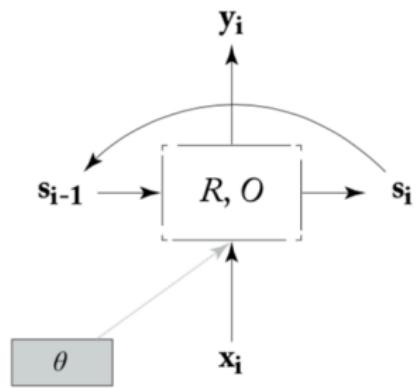
More details on RNN

- $RNN^*(x_{1:n}, s_0) = y_{1:n}$
- $y_i = O(s_i)$ – simple activation function
- $s_i = R(s_{i-1}, x_i)$, where R is a recursive function, s_i is a state vector
- s_0 is initialized randomly or is a zero vector
- $x_i \in \mathbb{R}^{d_{in}}$, $y_i \in \mathbb{R}^{d_{out}}$, $s_i \in \mathbb{R}^{f(d_{out})}$
- θ – shared weights

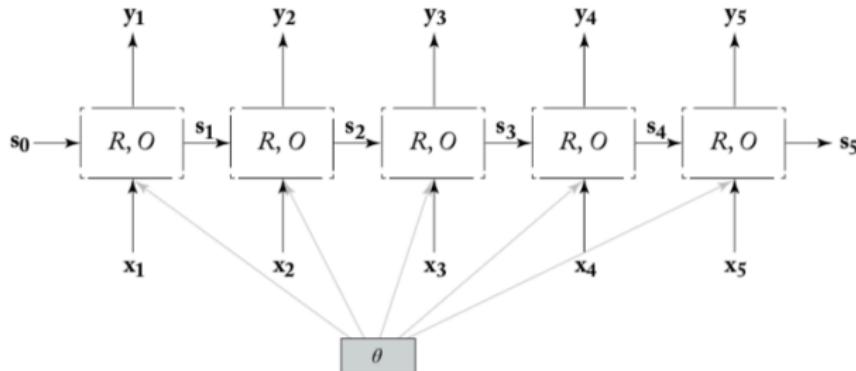


More details on RNN

- $s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$
- $y_i = O(s_i) = s_i$
- $y_i, s_i, b \in \mathbb{R}^{d_{out}}, x_i \in \mathbb{R}^{d_{in}}$
- $W^x \in \mathbb{R}^{d_{in} \times d_{out}}, W^s \in \mathbb{R}^{d_{out} \times d_{out}}$



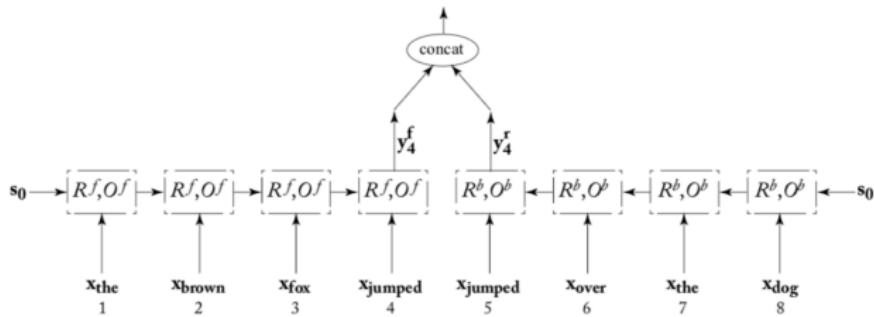
RNN unrolled



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

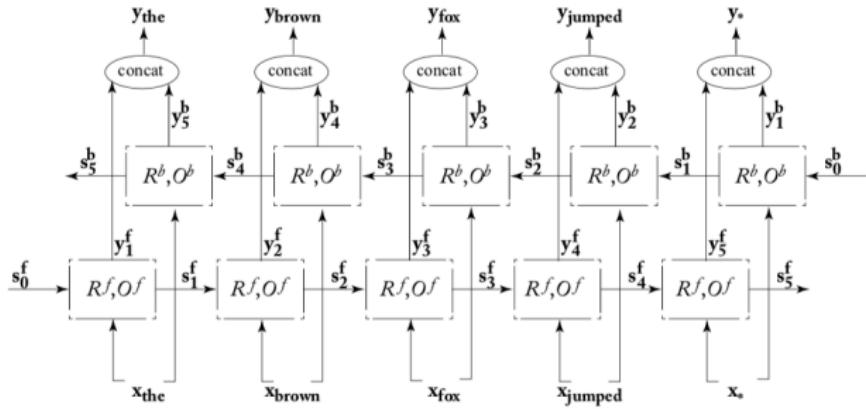
Bidirectional RNN (Bi-RNN)

The input sequence can be read from left to right and from right to left.
Which direction is better?



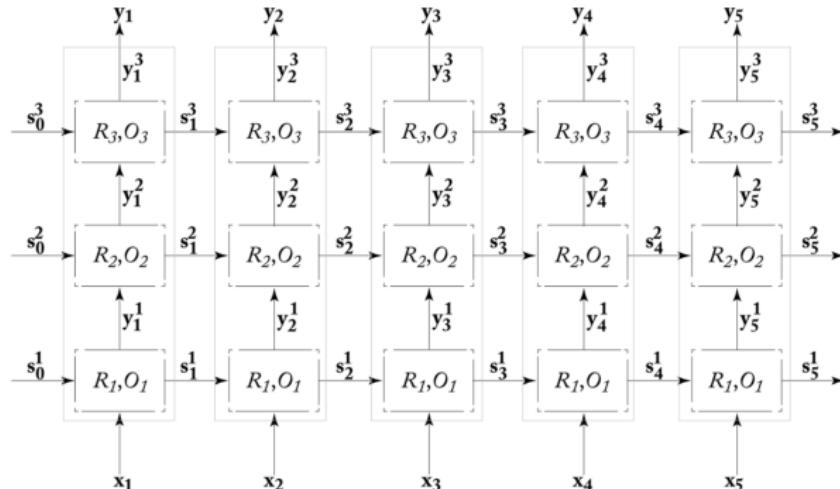
$$biRNN(x_{1:n}, i) = y_i = [RNN^f(x_{1:i}); RNN^r(x_{n:i})]$$

Bi-RNN



$$biRNN^*(x_{1:n}, i) = y_{1:n} = biRNN(x_{1:n}, 1) \dots biRNN(x_{1:n}, n)$$

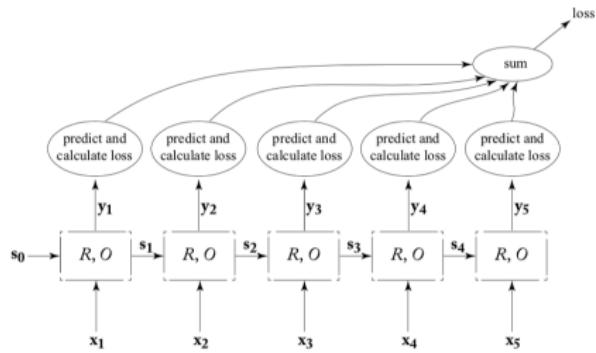
Multilayer RNN



Connections between different layers are possible too: $y_1^2 = \text{concat}(x_1, y_1^1)$

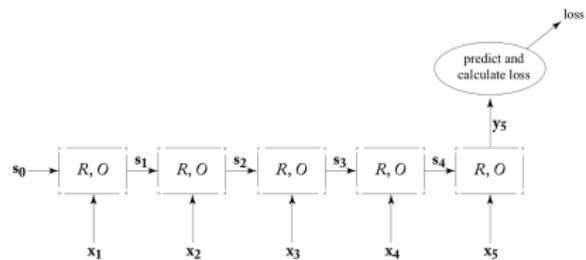
Sequence labelling

- Output \hat{t}_i for each input $x_{1,i}$
- Local loss: $L_{local}(\hat{t}_i, t_i)$
- Global loss:
$$L(\hat{t}_n, t_n) = \sum_i L_{local}(\hat{t}_i, t_i)$$
- L can take any form: cross entropy, hinge, margin, etc.

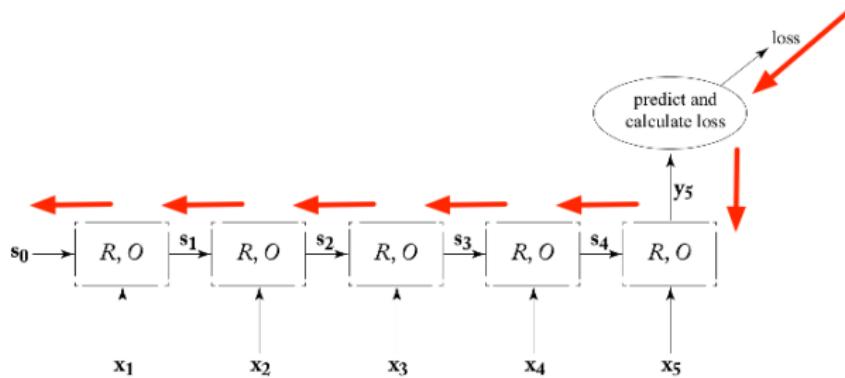


Sequence classification

- $\hat{y}_n = O(s_n)$
- prediction = $MLP(\hat{y}_n)$
- Loss: $L(\hat{y}_n, y_n)$
- L can take any form: cross entropy, hinge, margin, etc.



Backpropagation through time



$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

$$\text{Chain rule: } \frac{\partial L}{\partial w} = \frac{\partial L}{\partial p(\hat{y}_5)} \frac{\partial p(\hat{y}_5)}{\partial s_4} \left(\frac{\partial s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_w} + \dots \right)$$

Vanishing gradient problem

Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial p(\hat{y}_5)} \frac{\partial p(\hat{y}_5)}{\partial s_4} \left(\frac{\partial s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_w} + \dots \right)$
 g – sigmoid

- ➊ Many sigmoids near 0 and 1
 - ▶ Gradients $\rightarrow 0$
 - ▶ Not training for long term dependencies
- ➋ Many sigmoids > 1
 - ▶ Gradients $\rightarrow +\infty$
 - ▶ Not training again

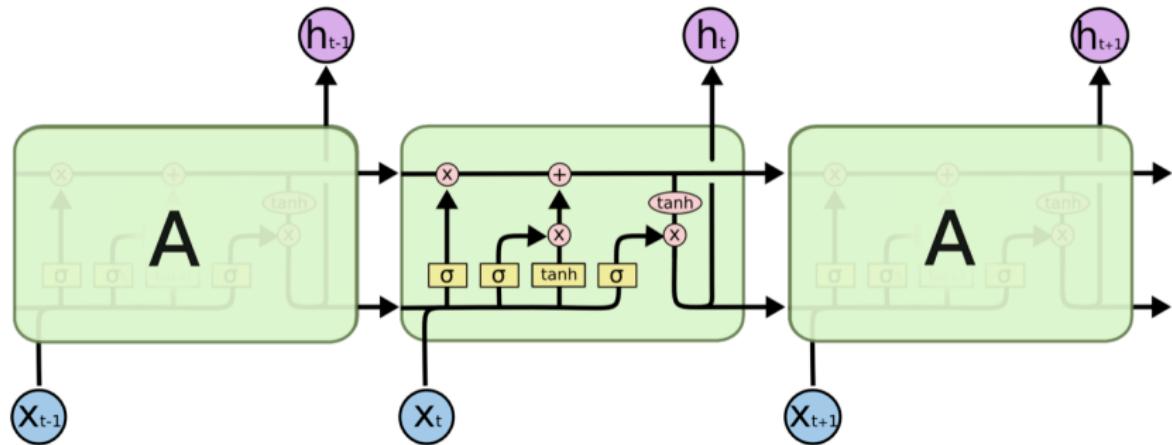
Solution: gated architectures (LSTM and GRU)

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

Controlled memory access

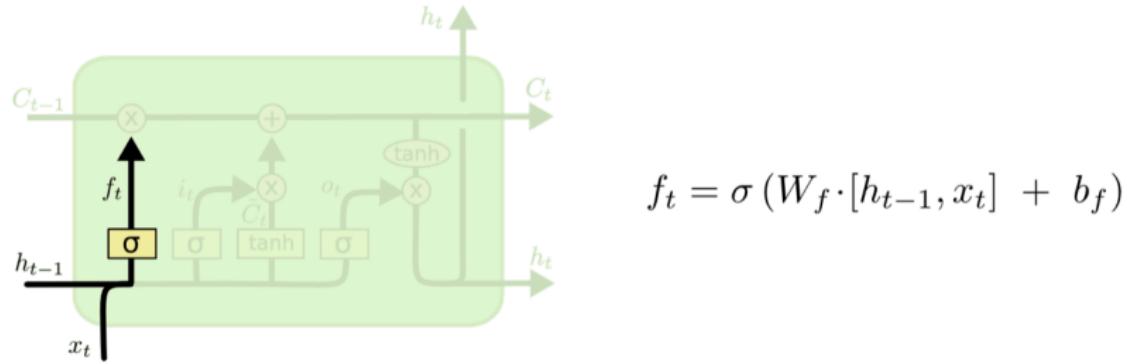
- Entire memory vector is changed: $s_{i+1} = R(x_i, s_i)$
- Controlled memory access: $s_{i+1} = g \odot R(x_i, s_i) + (1 - g)s_i$
 $g \in [0, 1]^d, s, x \in \mathbb{R}^d$
- Differential gates: $\sigma(g), g' \in \mathbb{R}^d$
- This controllable gating mechanism is the basis of the LSTM and the GRU architectures

Long short term memory



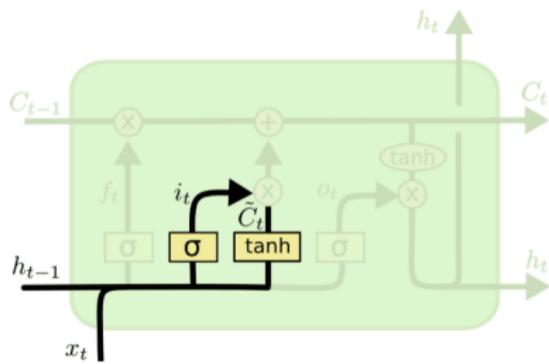
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Long short term memory



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

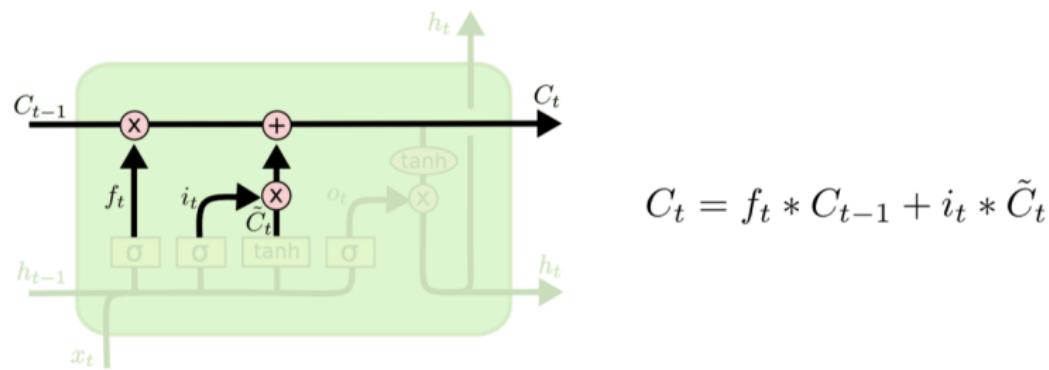
Long short term memory



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

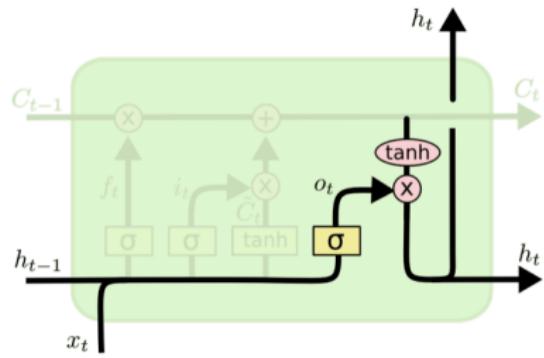
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Long short term memory



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Long short term memory

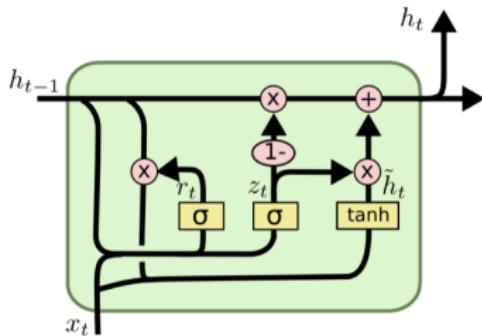


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

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

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Gated recurrent unit



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

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

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

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

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

Sequence generation

- Teacher forcing:

$$x := \langle s \rangle x, y := x \langle /s \rangle$$

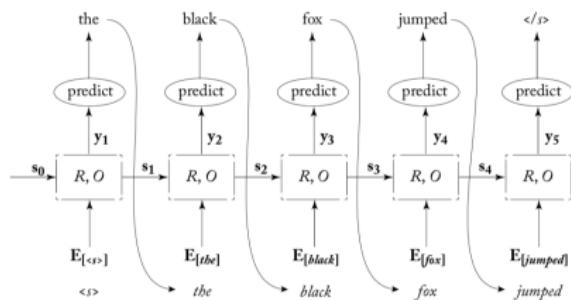
$$x := \langle s \rangle x_1 x_2 \dots x_n$$

$$y := x_1 x_2 \dots x_n \langle /s \rangle$$

- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j})$

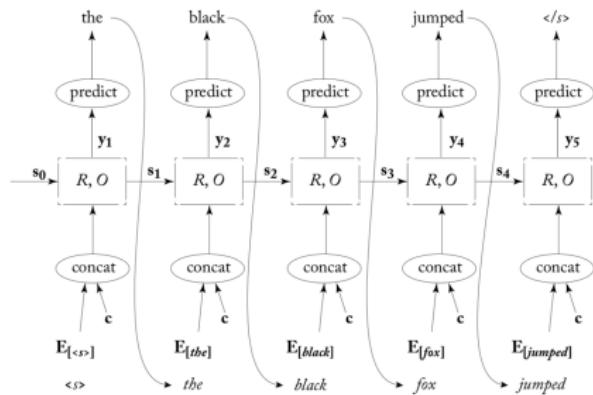
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{t}_{1:j}))$

$$f(x) = \text{softmax}(MLP(x))$$



Conditioned sequence generation

- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j}, c)$
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{v}_{1:j}))$
 $v_i = [\hat{t}_i, c]$



Sequence generation

- Examples of generated texts:

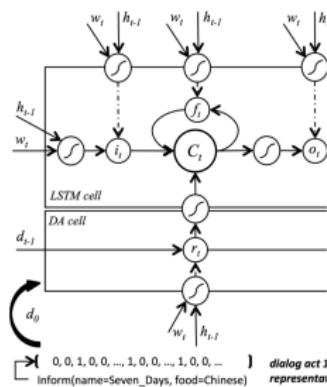
<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

- Examples of generated MIDI music:

<https://towardsdatascience.com/>

[how-to-generate-music-using-a-lstm-neural-network-in-keras](https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras)

Conditioned sequence generation [WGM⁺15]



| # | Example Dialogue Acts and Realizations from SF Restaurant Domain |
|---|---|
| 1 | <pre>inform(name="red door cafe", goodformmeal="breakfast", area="cathedral hill", kidsallowed="no") red door cafe is a good restaurant for breakfast in the area of cathedral hill and does not allow children . red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow children . red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow kids . red door cafe is good for breakfast and is in the area of cathedral hill and does not allow children . red door cafe does not allow kids and is in the cathedral hill area and is good for breakfast .</pre> |
| 2 | <pre>informonly(name="dosa on fillmore and kiss seafood", pricerange="expensive", near="lower pacific heights") there is no place other than dosa on fillmore and kiss seafood that are expensive near to lower pacific heights . dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights . the only listed restaurant near lower pacific heights in the expensive price range is dosa on fillmore and kiss seafood . i apologize , dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights . i apologize , dosa on fillmore and kiss seafood are the only expensive restaurants near lower pacific heights .</pre> |
| # | Example Dialogue Acts and Realizations from SF Hotel Domain |
| 3 | <pre>inform(type="hotel", count="182", dogsallowed="dontcare") there are 182 hotels if you do not care whether dogs are allowed . there are 182 hotels if you do not care whether they allow dogs . 182 hotels are available if dogs allowed or not is not an issue . there are 182 hotels if allowing dogs or not is not an issue . there are 182 hotels if whether dogs are allowed does not matter .</pre> |
| 4 | <pre>informonly(name="red victorian bed breakfast", acceptscreditcards="yes", near="haight", hasinternet="yes") red victorian bed breakfast is the only hotel near haight and accepts credit cards and has internet . red victorian bed breakfast is the only hotel near haight and has internet and accepts credit cards . red victorian bed breakfast is the only hotel near haight that accept credit cards and offers internet . the red victorian bed breakfast has internet and near haight , it does accept credit cards . the red victorian bed breakfast is the only hotel near haight that accepts credit cards , and offers internet .</pre> |

Conclusion

Topics covered:

- ① RNN is a powerful tool for sequence modeling
- ② RNN usage scenarios: sequence labelling, sequence classification, sequence generation
- ③ RNN layers can be reversed → bidirectional RNN
- ④ RNN layers can be stacked → deep RNN
- ⑤ RNN suffers from gradient vanishing problem → LSTM, GRU

Topics not covered:

- ① seq2seq models
- ② Attention mechanism in RNN
- ③ Recursive neural networks

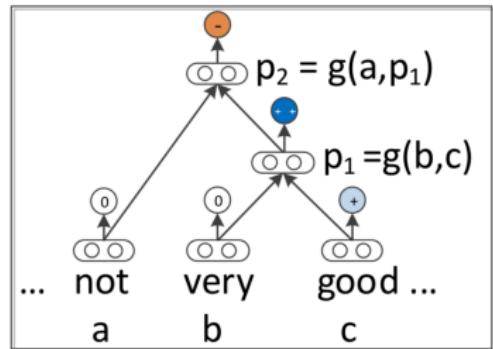
- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

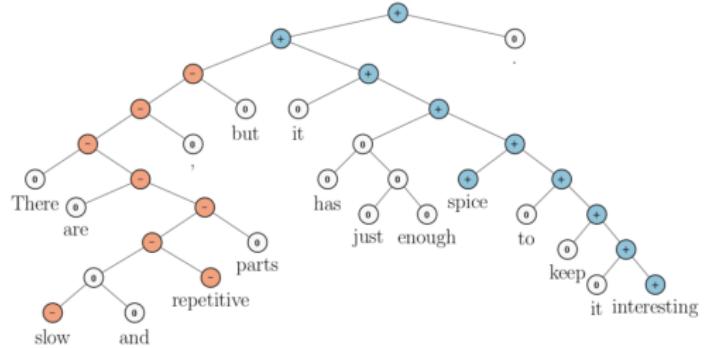
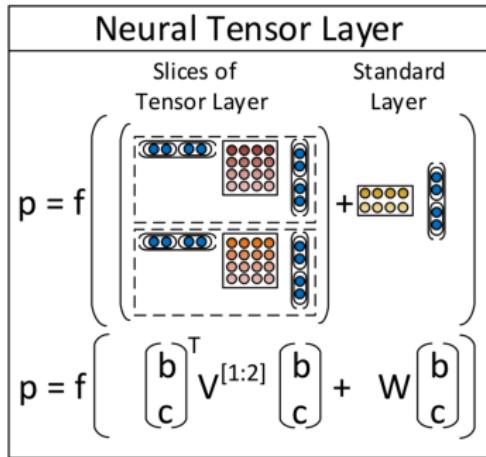
Modeling trees with Recursive NN

- Input: x_1, x_2, \dots, x_n
- A binary tree T can be represented as a unique set of triplets (i, k, j) , s.t. $i < k < j$, $x_{i:j}$ is parent of $x_{i:k}, x_{k+1:j}$
- RecNN takes as an input a binary tree and returns as output a corresponding set of inside state vectors $s_{i:j}^A \in \mathbb{R}^d$
- Each state vector $s_{i:j}^A$ represents the corresponding tree node $q_{i:j}^A$ and encodes the entire structure rooted at that node

RecNN

- Input: x_1, x_2, \dots, x_n and a binary tree T
- $RecNN(x_1, x_2, \dots, x_n, T) = \{s_{i:j}^A \in \mathbb{R}^d | q_{i:j}^A \in T\}$
- $s_{i:i}^A = v(x_i)$
- $s_{i:j}^A = R(A, B, C, s_{i:k}^B, s_{k+1:j}^C), q_{i:k}^B \in T, q_{k+1:j}^C \in T$
- $R(A, B, C, s_{i:k}^B, s_{k+1:j}^C) = g([s_{i:k}^B, s_{k+1:j}^C]W)$





- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

Hierarchical Attention Networks [YYD⁺16]

$$x_{it} = W_e w_{it}, t \in [1, T],$$

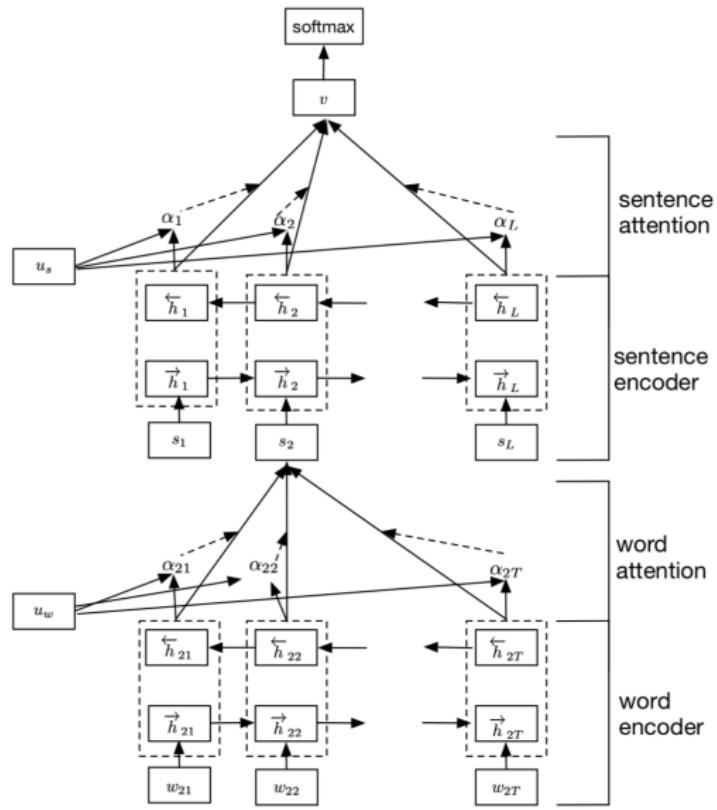
$$\vec{h}_{it} = \overrightarrow{\text{GRU}}(x_{it}), t \in [1, T],$$

$$\overleftarrow{h}_{it} = \overleftarrow{\text{GRU}}(x_{it}), t \in [T, 1].$$

$$u_{it} = \tanh(W_w h_{it} + b_w)$$

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}$$

$$s_i = \sum_t \alpha_{it} h_{it}.$$



- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
- 3 Gated architectures
- 4 RNN generators
- 5 Bonus I: RecNN
- 6 Bonus II: HAN
- 7 Bonus III: IE

Text annotation

Stanford CoreNLP 3.9.1 (updated 2018/04/05)

- Text to annotate -

The music was so loud that it couldn't be enjoyed

- Annotations -

parts-of-speech X coreference X

- Language -

English

Submit

Part-of-Speech:

DT NN VBD RB JJ IN PRP MD RB VB VBN

1 The music was so loud that it could n't be enjoyed

Coreference:

Mention -----coref----- Mention

1 The music was so loud that it could n't be enjoyed

Text annotation

Stanford CoreNLP 3.9.1 (updated 2018/04/05)

— Text to annotate —

Charles Aznavour : 'French Frank Sinatra' dies aged 94

— Annotations —

parts-of-speech named entities dependency parse openie

— Language —

English

Submit

Part-of-Speech:



Named Entity Recognition:



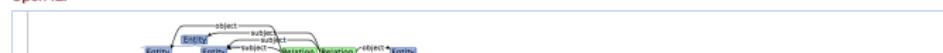
Basic Dependencies:



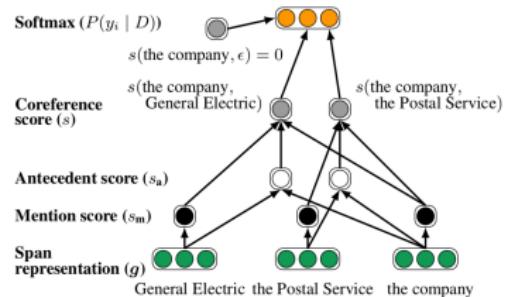
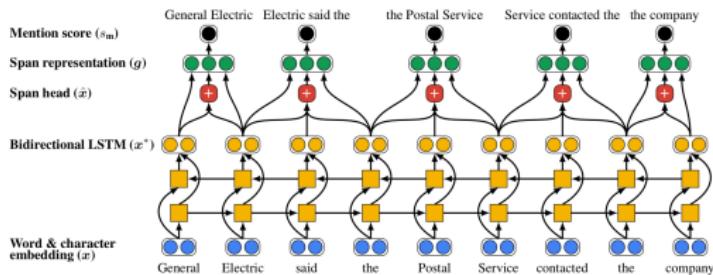
Enhanced++ Dependencies:



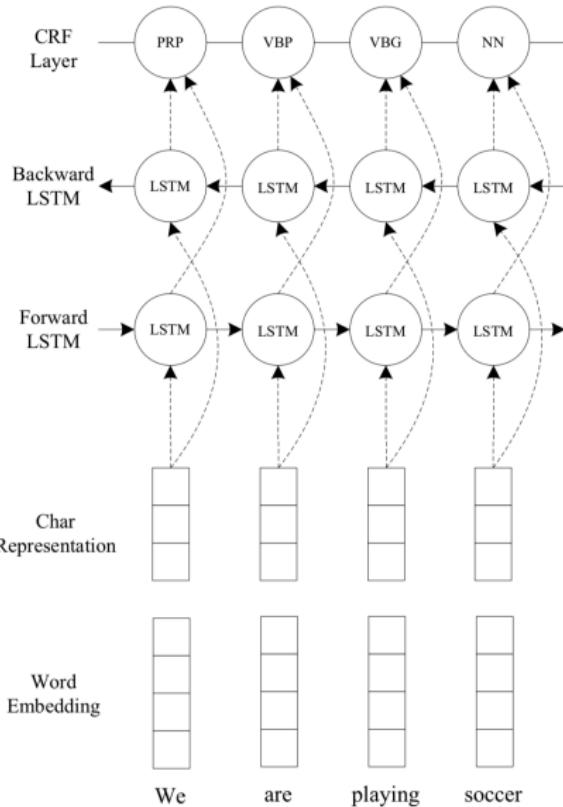
Open IE:



Coreference resolution [LHLZ17]



Named entity recognition [MH16]



References I

-  Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer, *End-to-end neural coreference resolution*, arXiv preprint arXiv:1707.07045 (2017).
-  Xuezhe Ma and Eduard Hovy, *End-to-end sequence labeling via bi-directional lstm-cnns-crf*, arXiv preprint arXiv:1603.01354 (2016).
-  Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.
-  Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young, *Semantically conditioned lstm-based natural language generation for spoken dialogue systems*, arXiv preprint arXiv:1508.01745 (2015).

References II

-  Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, *Hierarchical attention networks for document classification*, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.