Natural Language Processing

5. Recurrent neural networks

Katya Chernyak

CS HSE

October 8, 2018

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

Classification problem: $\gamma : \boldsymbol{x} \to \boldsymbol{y}$

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

Sequence modelling II

Sequence classification

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

Classification problem: $\gamma : \boldsymbol{x} \to y$

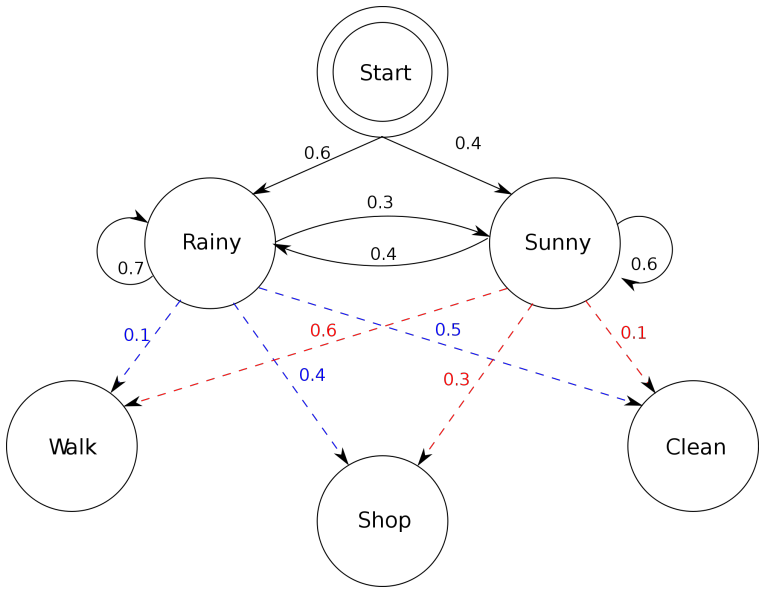
1. Activity recognition: x – pulse rate, y – activity (walking, running, peace)
2. Opinion mining: x – sentence, y – sentiment (positive, negative)
3. Trading: x – stock market, y – action (sell, buy, do nothing)

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

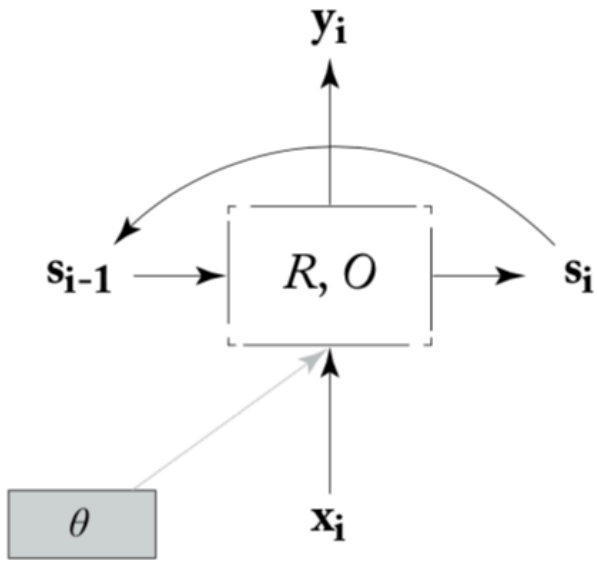
Problems:

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



Recurrent neural network

- Input: sequence of vectors
- $x_{1:n} = x_1, x_2, \ldots, x_n$, $x_i \in \mathbb{R}^{d_{in}}$
- 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

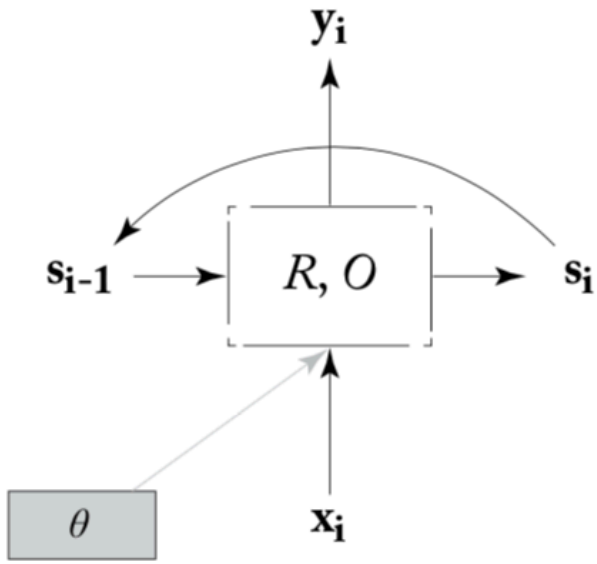
$$p(l_j|\boldsymbol{x}_j) = \texttt{softmax}(RNN(\boldsymbol{x}_{1:j}) \times W + b)_{[j]}$$

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

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

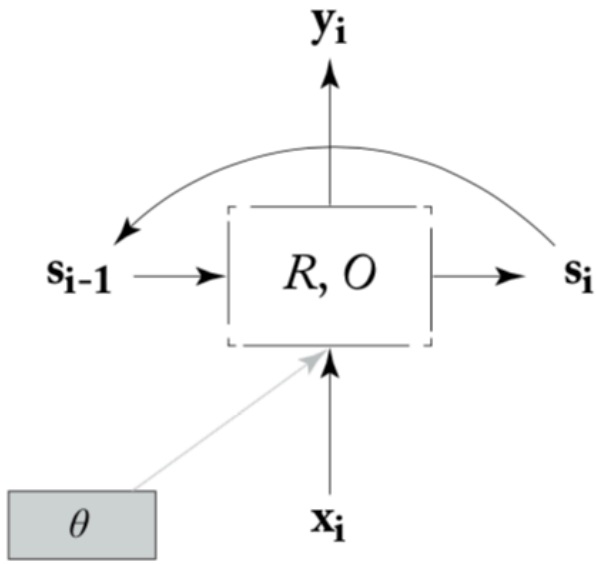
More details on RNN

- $RNN^*(x_{1:n}, s_0) = y_{1:n}$
- $y_i = O(s_i)$ – 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

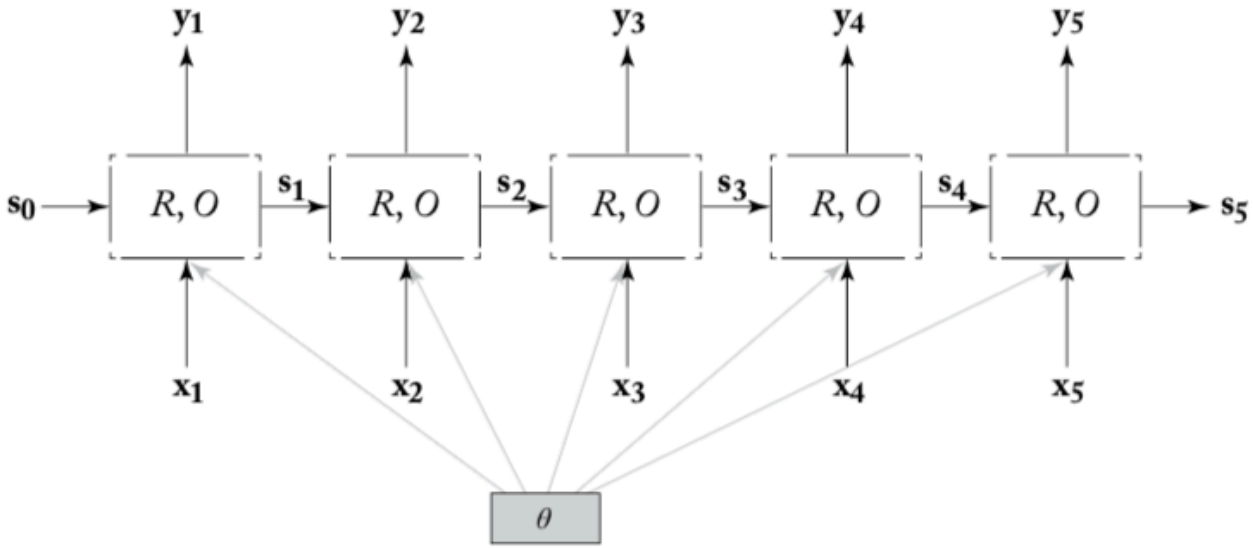


More details on RNN

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



RNN unrolled

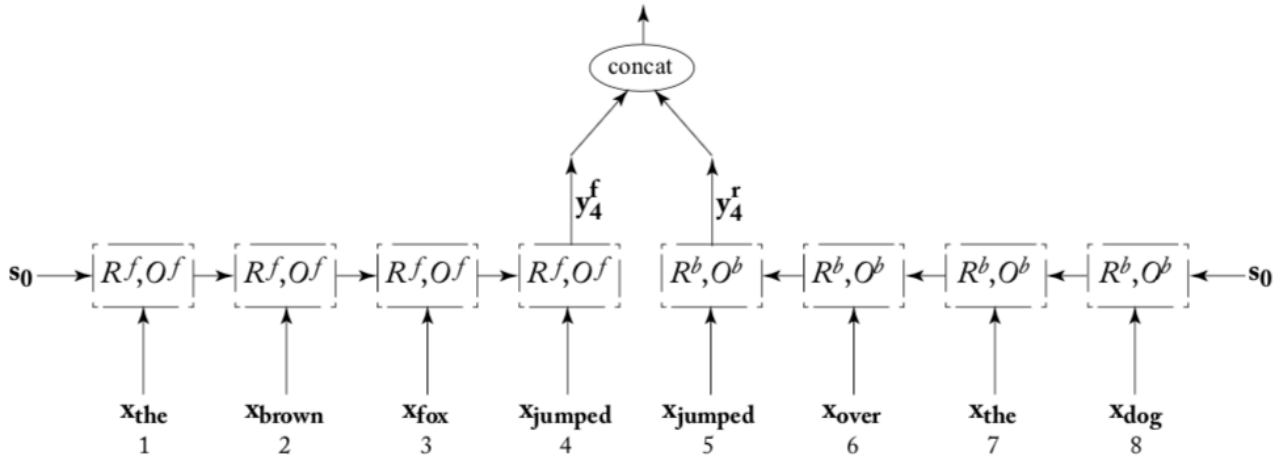


$$s_4 = R(s_3, x_4) = R(R(s_2, x_3), x_4) = R(R(R(s_1, x_2), x_3), x_4) =$$

$$= R(R(R(R(s_0, x1), x_2), x_3), x_4)$$

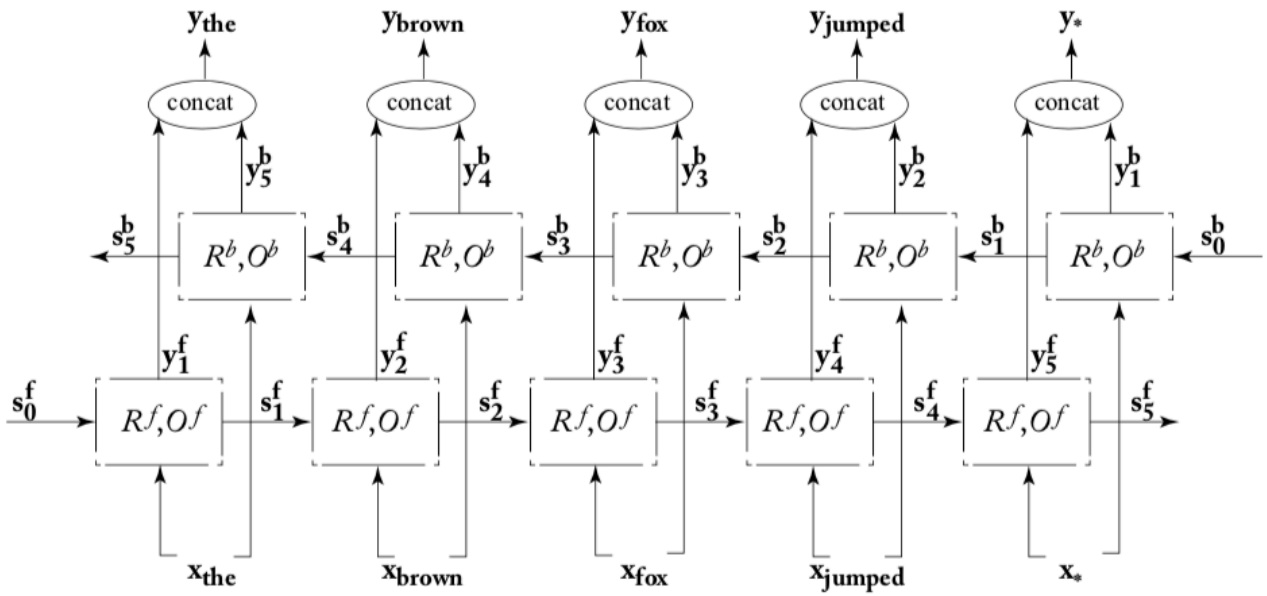
Bidirectional RNN (Bi-RNN)

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



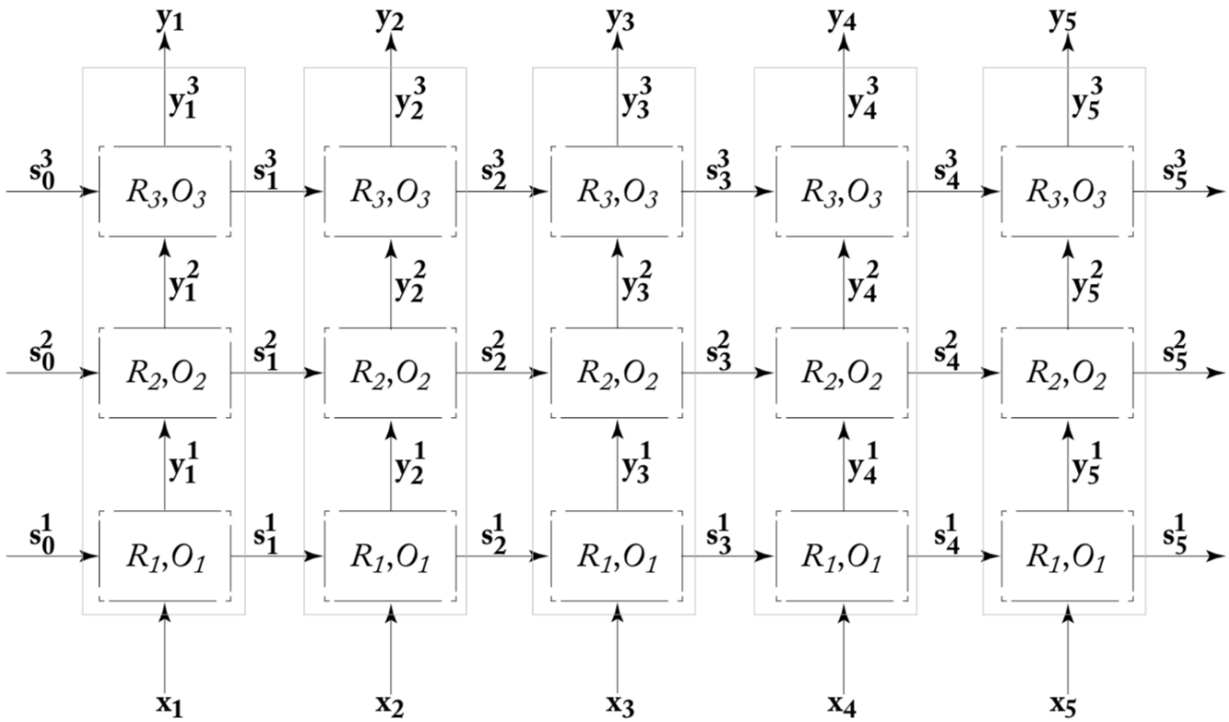
$$biRNN(x_{1:n}, i) = y_i = [RNN^f(x_{1:i}); RNN^r(x_{n:i})]$$

Bi-RNN



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

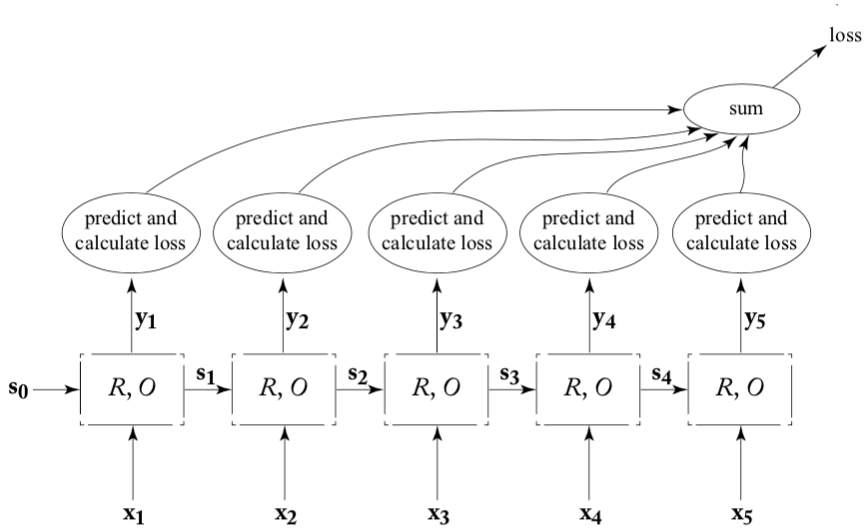
Multilayer RNN



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

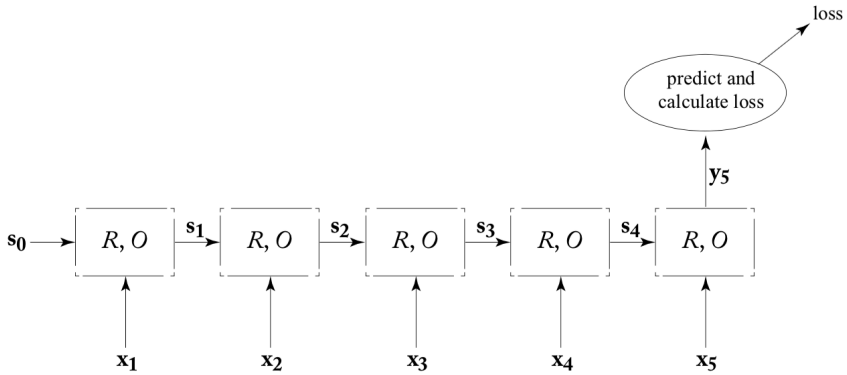
Sequence labelling

- 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.

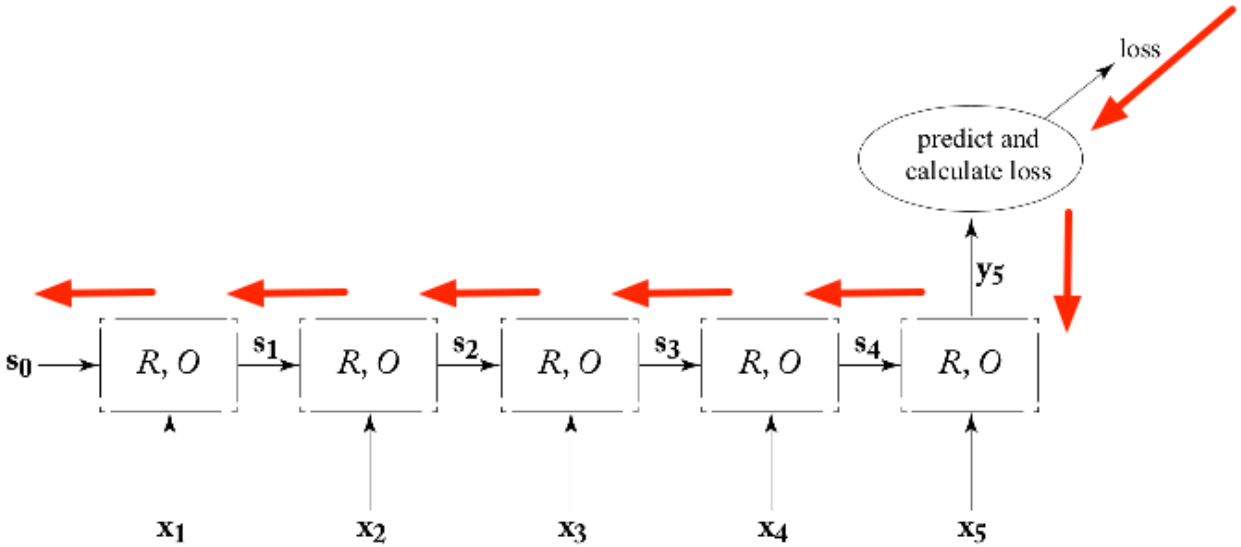


Sequence classification

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



Backpropogation through time



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

Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial p(\hat{y_5})} \frac{\partial p(\hat{y_5})}{\partial s_4} (\frac{\partial s_4}{\partial w} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial w} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_w} + \ldots)$

Vanishing gradient problem

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

g – sigmoid

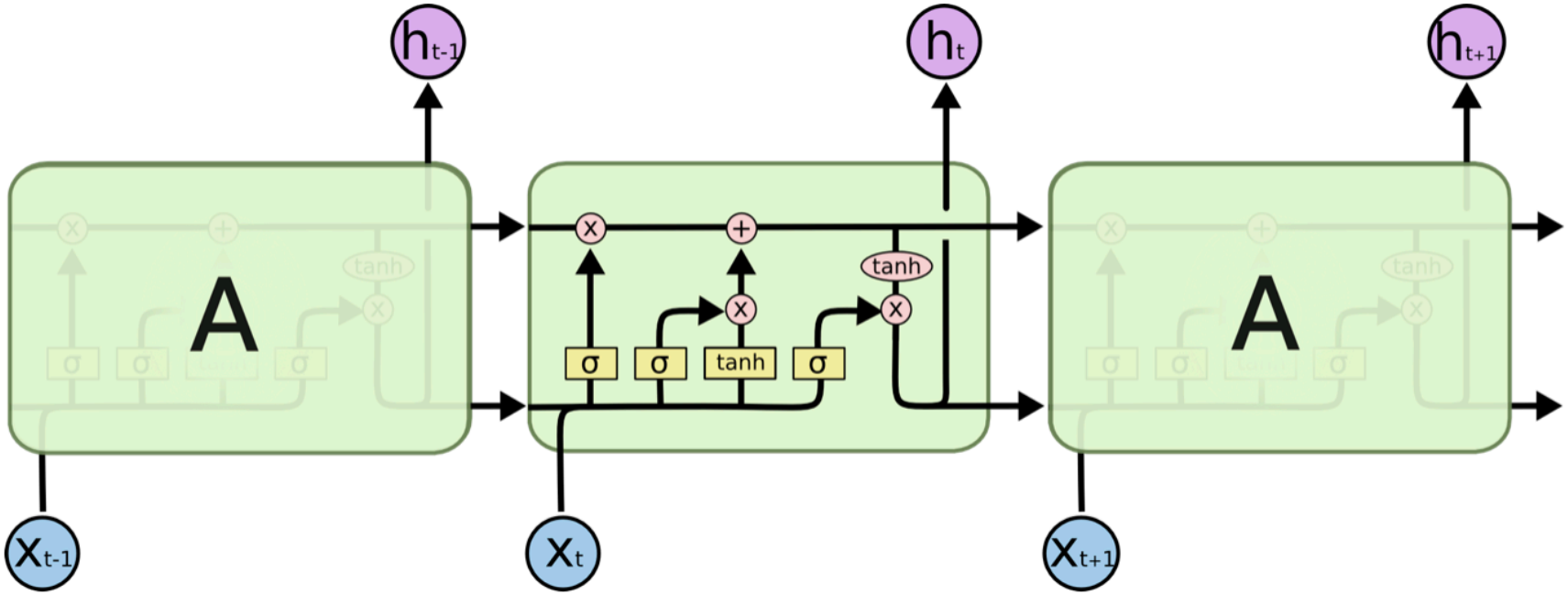
1. Many sigmoids near 0 and 1
   - Gradients $\to 0$
   - Not training for long term dependencies
2. Many sigmoids > 1
   - Gradients $\to +\text{inf}$
   - Not training again

Solution: gated architectures (LSTM and GRU)

Controlled memory access

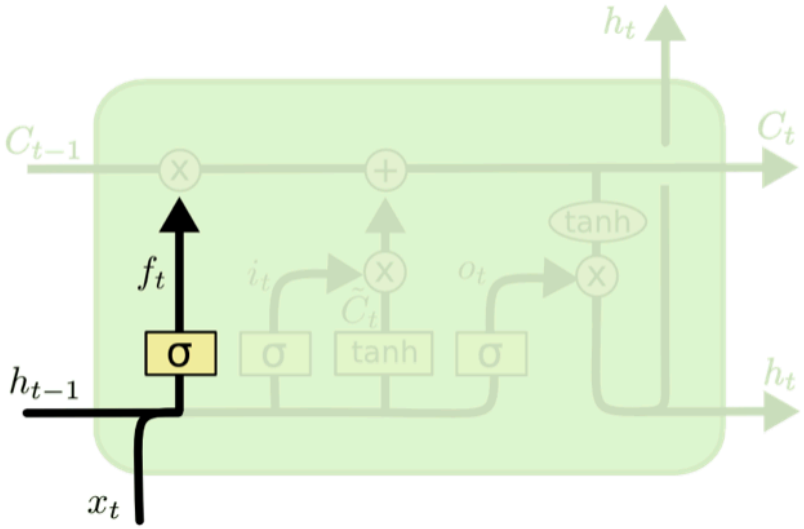
- Entire memory vector is changed: $s_{i+1} = R(x_i, s_i)$
- Controlled memory access: $s_{i+1} = g \odot R(x_i, s_i) + (1 - g)s_i$
  $g \in [0, 1]^d, s, x \in \mathbb{R}^d$
- Differential gates: $\sigma(g), g' \in \mathbb{R}^d$
- This controllable gating mechanism is the basis of the LSTM and the GRU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

Long short term memory



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

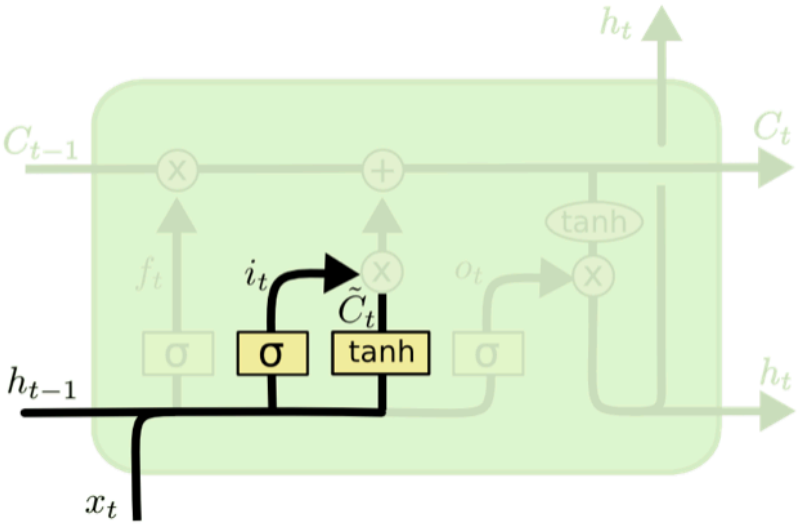
Long short term memory



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

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

Long short term memory

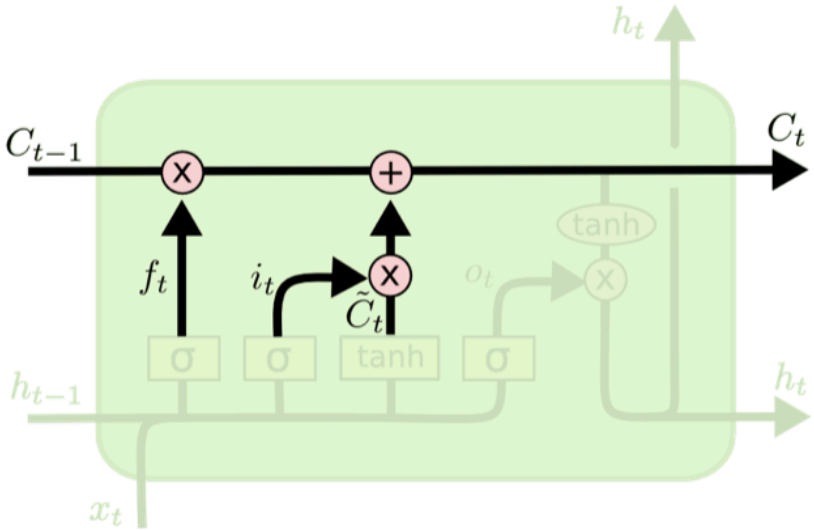


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

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

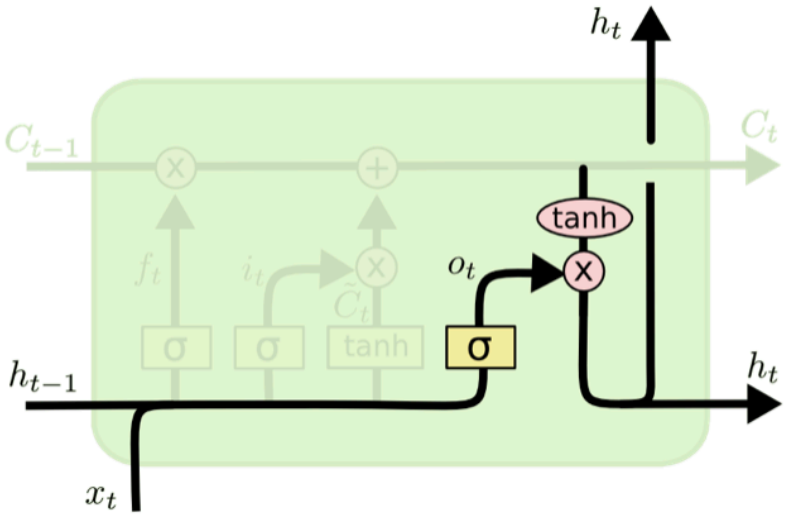
Long short term memory



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

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

Long short term memory

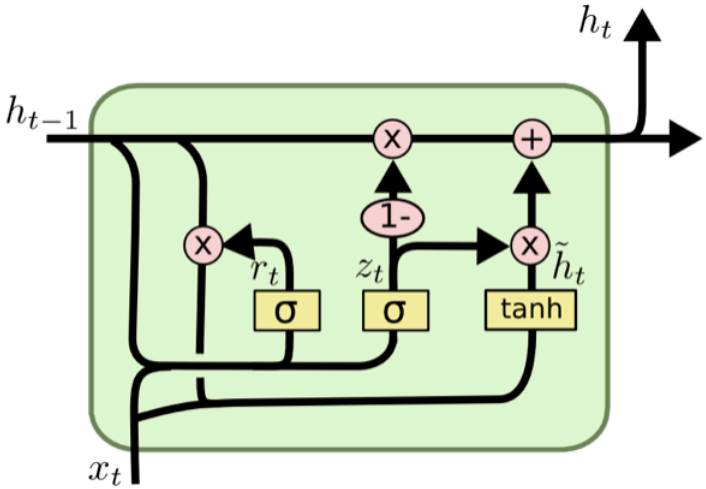


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

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

Gated recurrent unit



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

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

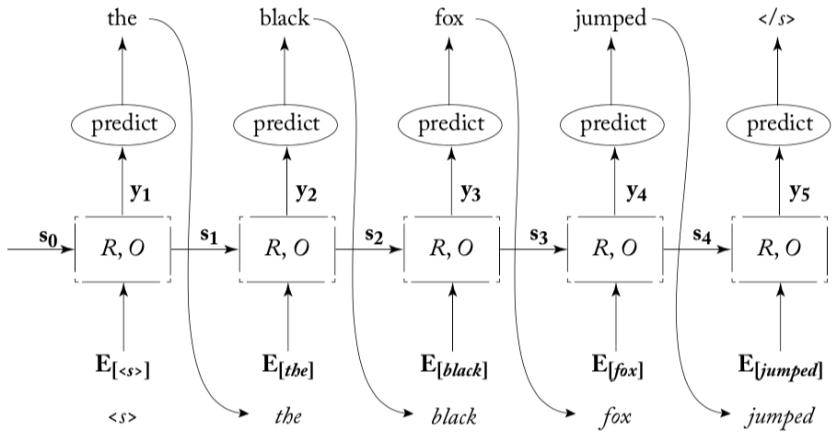
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

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

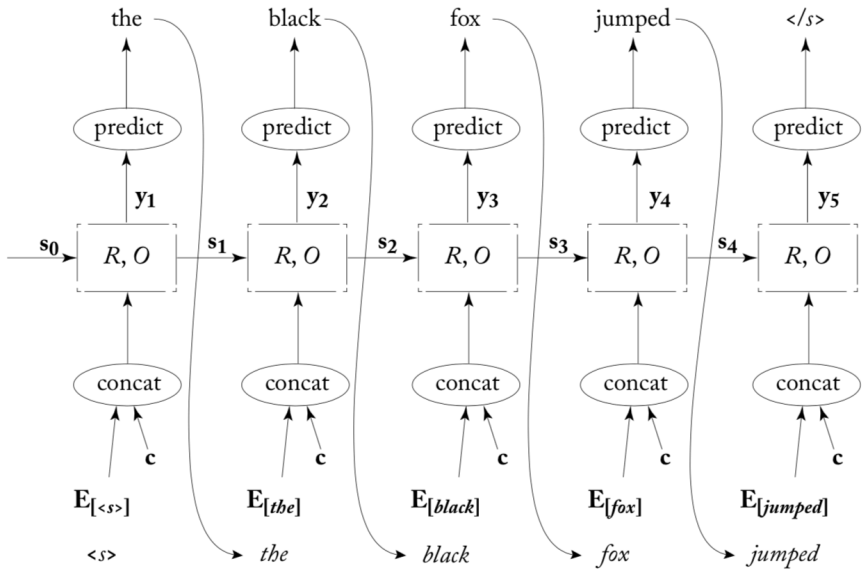
Sequence generation

- Teacher forcing:
  $x :=< s > x, y := x < /s >$
  $x :< s > x_1 x_2 \dots x_n$
  $y : x_1 x_2 \dots x_n < /s >$
- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j})$
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{t}_{1:j}))$
  $f(x) = \texttt{softmax}(MLP(x))$



Conditioned sequence generation

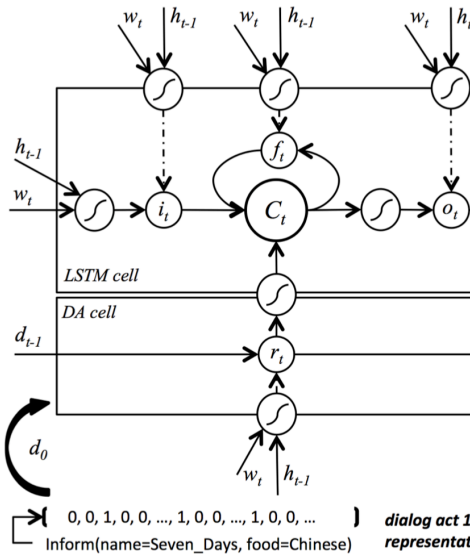
- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j}, c)$
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{v}_{1:j}))$
  $v_i = [\hat{t}_i, c]$



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`

Conditioned sequence generation [WGM+15]



#	Example Dialogue Acts and Realizations from SF Restaurant Domain
1	inform(name="red door cafe", goodformeal="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 .*
2	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 .*
#	**Example Dialogue Acts and Realizations from SF Hotel Domain**
3	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 .*
4	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 .*

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 $\rightarrow$ bidirectional RNN
4. RNN layers can be stacked $\rightarrow$ deep RNN
5. RNN suffers from gradient vanishing problem $\rightarrow$ LSTM, GRU

Topics not covered:

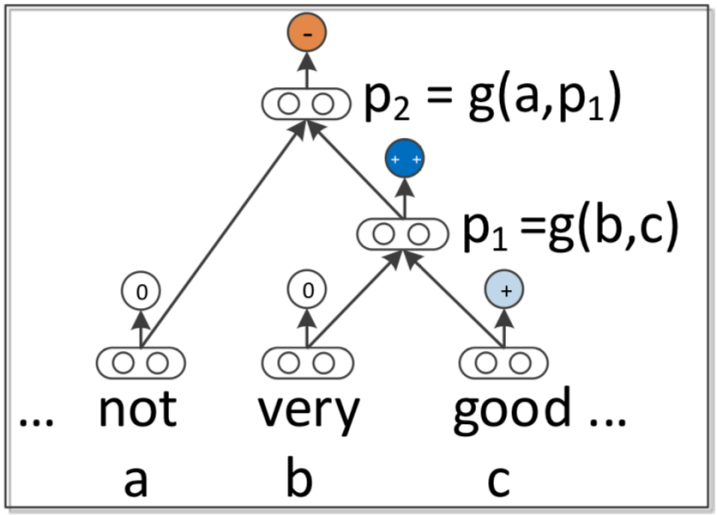
1. seq2seq models
2. Attention mechanism in RNN
3. Recursive neural networks

Modeling trees with Recursive NN

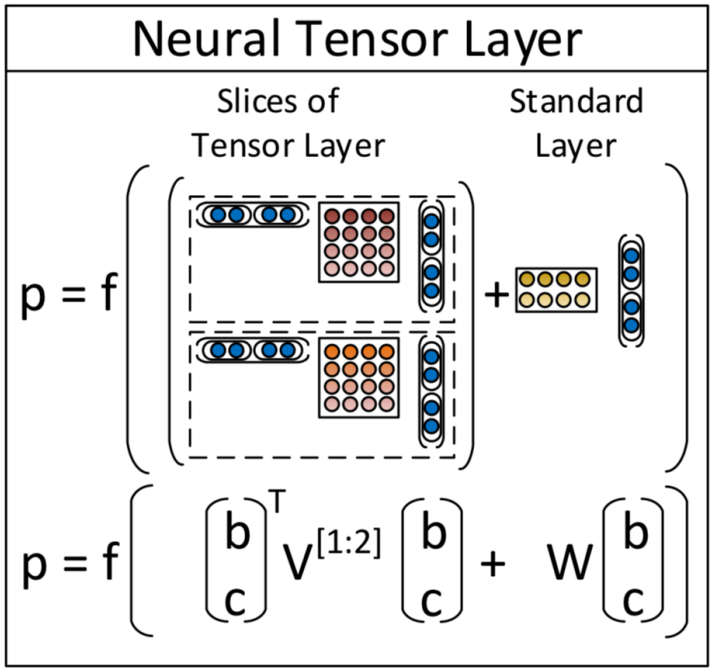
- Input: $x_1, x_2, \ldots, x_n$
- A binary tree T can be represented as a unique set of triplets (i, k, j), s.t. $i < k < j$, $x_{i:j}$ is parent of $x_{i:k}, i_{k+1,j}$
- RecNN takes as an input a binary tree and returns as output a corresponding set of inside state vectors $\boldsymbol{s}_{\boldsymbol{i:j}}^{\boldsymbol{A}}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

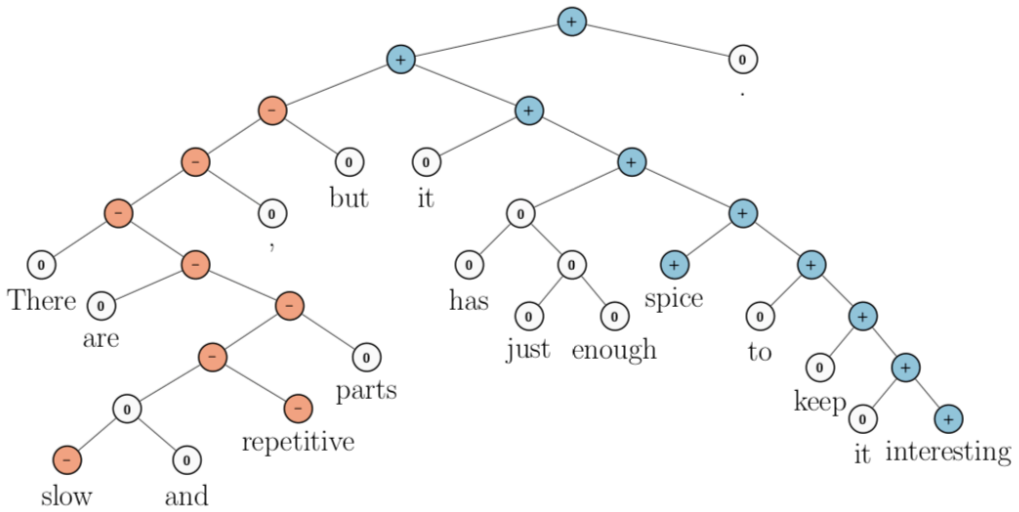
RecNN

- Input: $x_1, x_2, \ldots, x_n$ and a binary tree T
- $RecNN(x_1, x_2, \ldots, x_n, T) =$
  $= \{\boldsymbol{s}^{\boldsymbol{A}}_{\boldsymbol{i:j}} \in \mathbb{R}^d | q^A_{i:j} \in T\}$
- $\boldsymbol{s}^{\boldsymbol{A}}_{\boldsymbol{i:i}} = v(x_i)$
- $\boldsymbol{s}^{\boldsymbol{A}}_{\boldsymbol{i:j}} = R(A, B, C, \boldsymbol{s}^{\boldsymbol{B}}_{\boldsymbol{i:k}}, \boldsymbol{s}^{\boldsymbol{C}}_{\boldsymbol{k+1:j}})$,
  $q^B_{i:k} \in T, q^C_{k+1:j} \in T$
- $R(A, B, C, \boldsymbol{s}^{\boldsymbol{B}}_{\boldsymbol{i:k}}, \boldsymbol{s}^{\boldsymbol{C}}_{\boldsymbol{k+1:j}}) =$
  $g([\boldsymbol{s}^{\boldsymbol{B}}_{\boldsymbol{i:k}}, \boldsymbol{s}^{\boldsymbol{C}}_{\boldsymbol{k+1:j}}]W)$



RecNN [SPW+13]


Neural Tensor Layer
Slices of Tensor Layer
Standard Layer
p = f
+
p = f
b
c
T
V[1:2]
b
c
+
W
b
c


There
are
slow
and
repetitive
parts
but
,
it
has
just
enough
spice
to
keep
it
interesting
.

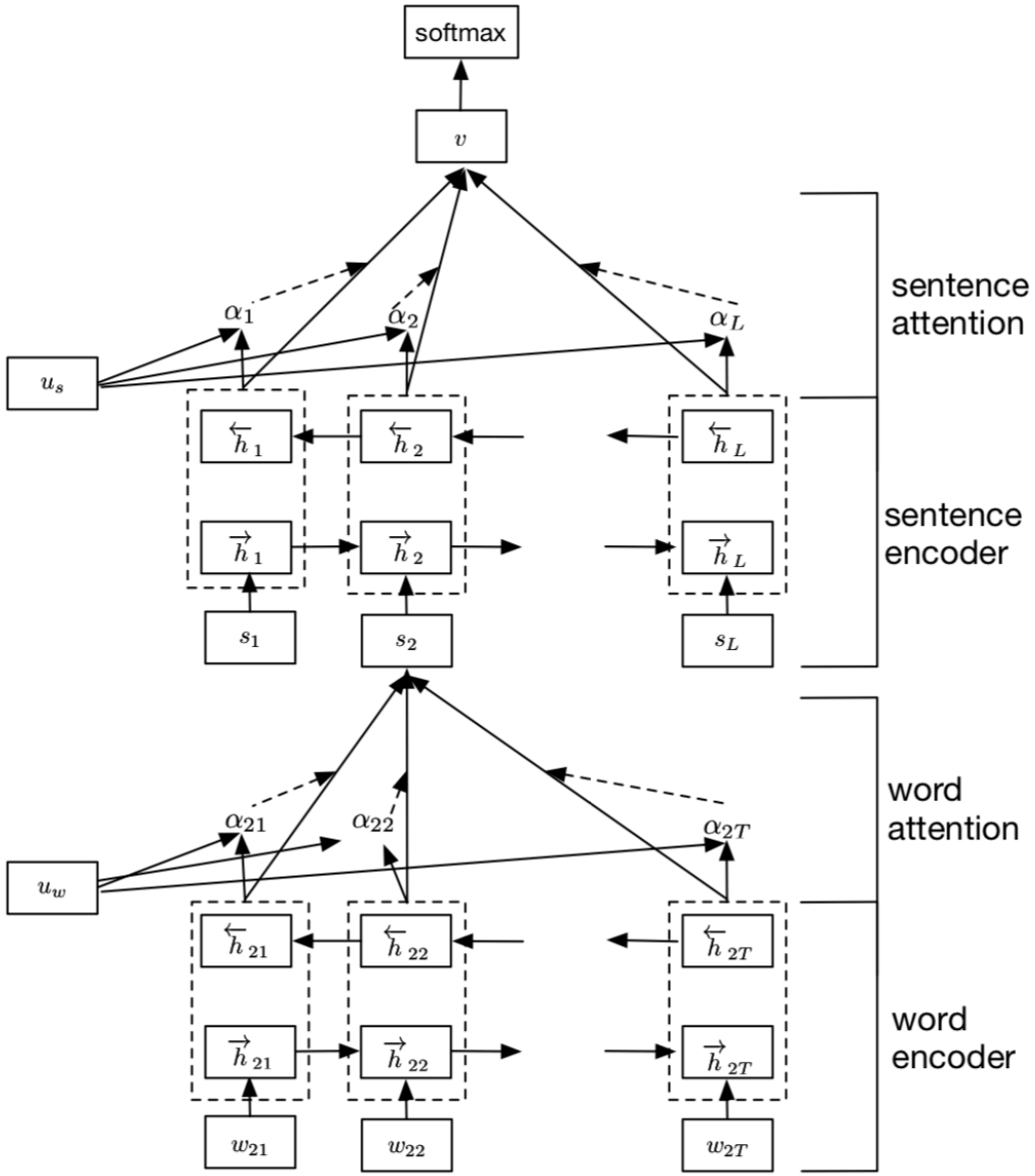
Hierarchical Attention Networks [YYD+16]

$$
\begin{aligned}
x_{it} &= W_e w_{it}, t \in [1, T], \\
\overrightarrow{h}_{it} &= \overrightarrow{\mathrm{GRU}}(x_{it}), t \in [1, T], \\
\overleftarrow{h}_{it} &= \overleftarrow{\mathrm{GRU}}(x_{it}), t \in [T, 1].
\end{aligned}
$$

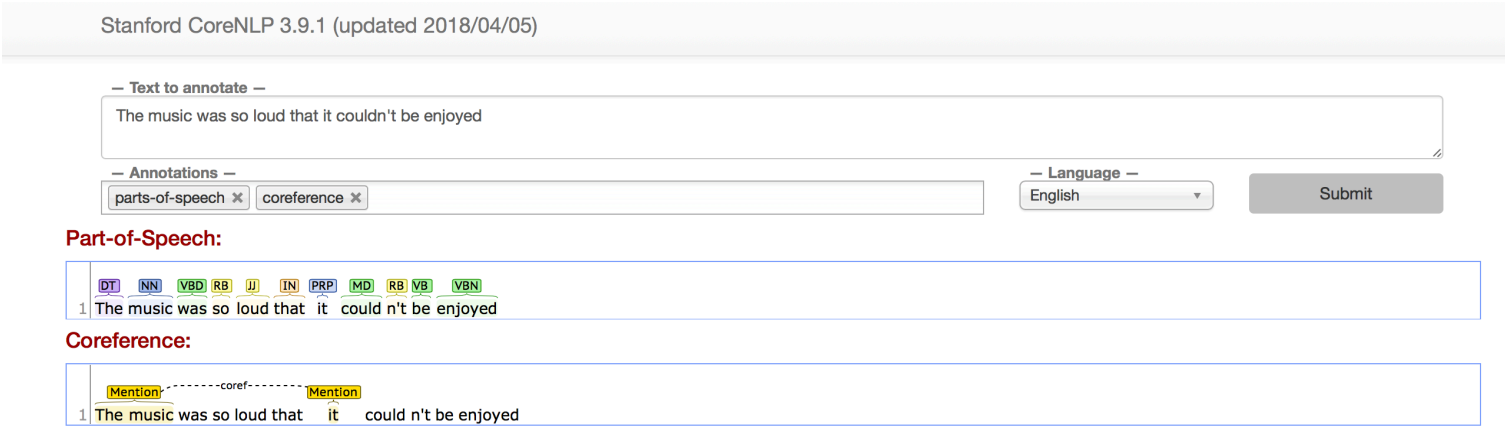
$$
u_{it} = \tanh(W_w h_{it} + b_w)
$$

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

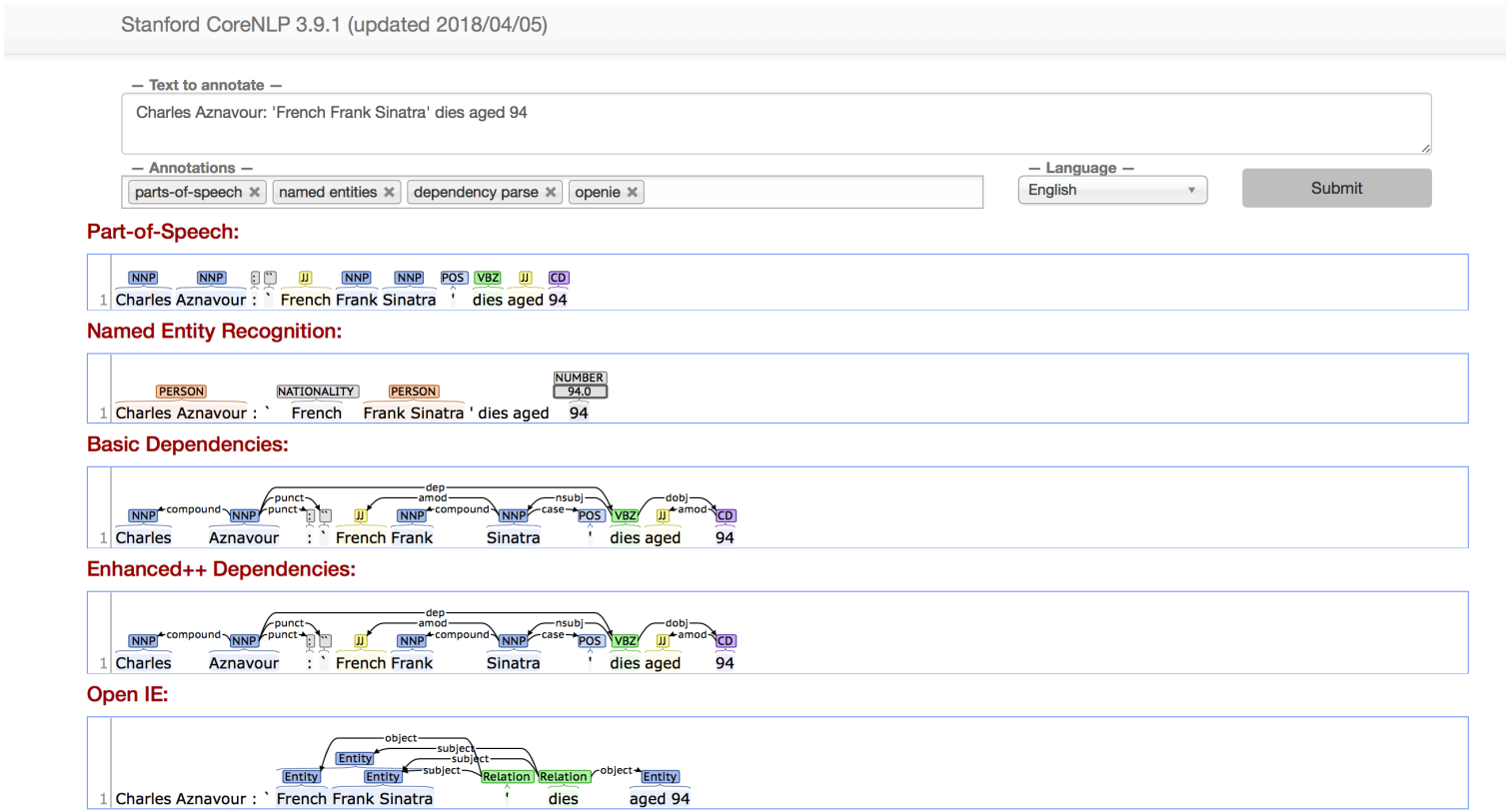
$$
s_i = \sum_t \alpha_{it} h_{it}.
$$



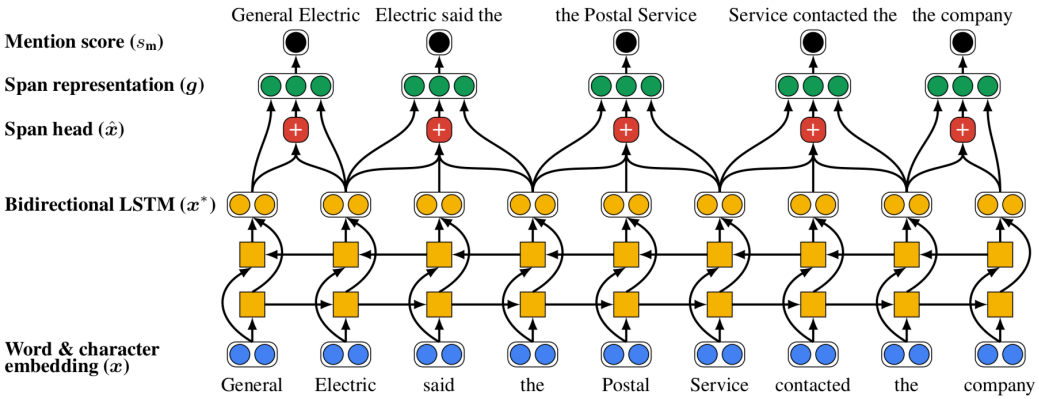
Text annotation

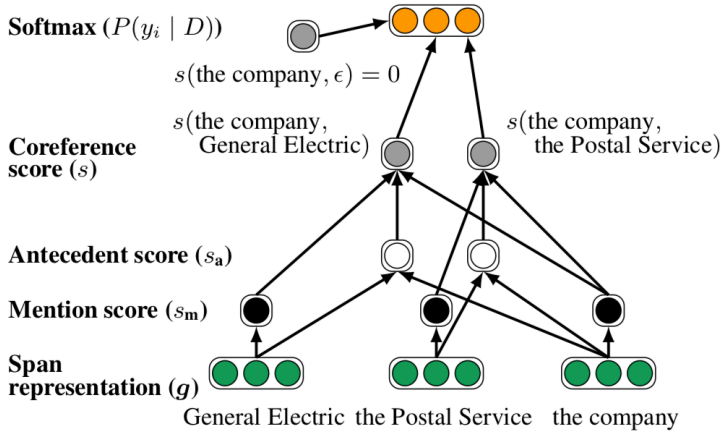


Text annotation

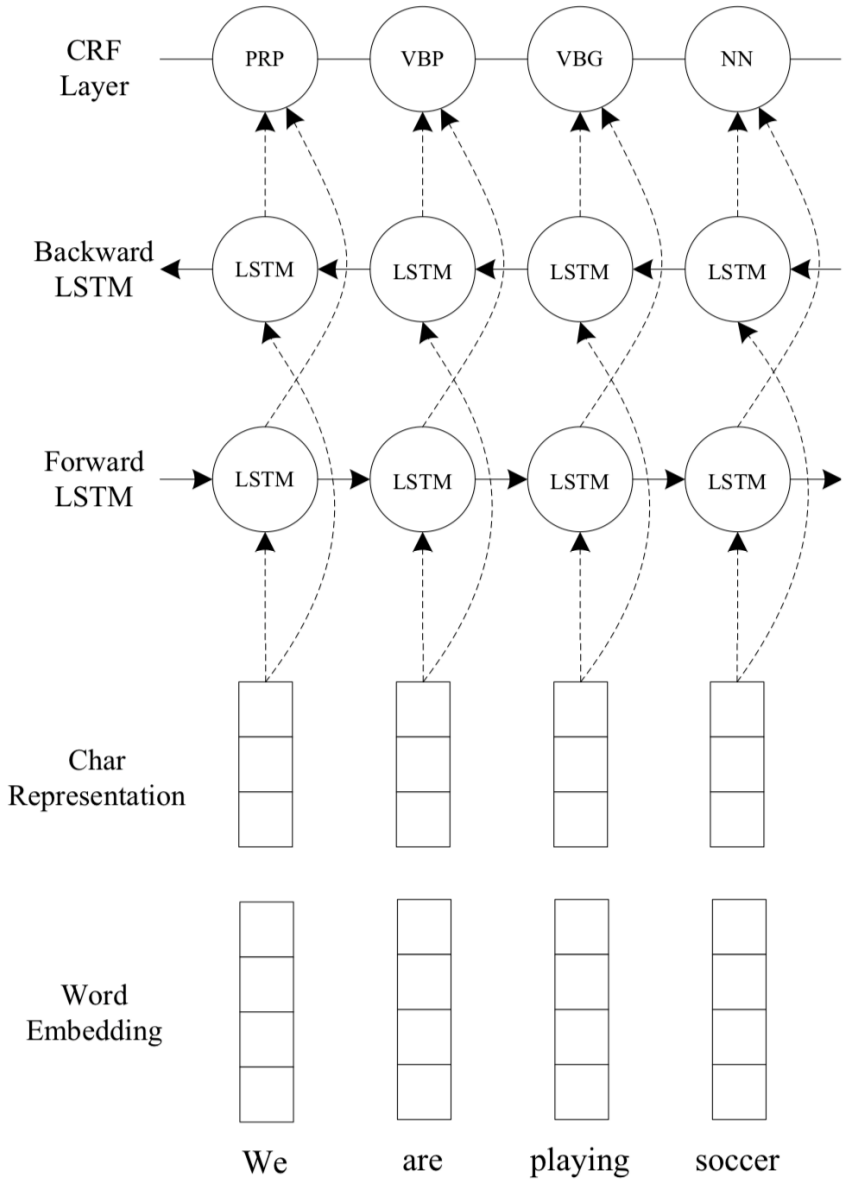


Coreference resolution [LHLZ17]





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.