

Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

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 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)

Sequence modelling II

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 $\text{length}(\mathbf{x}) = n$, there are L^n possible solutions

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

- 1 Part of speech tagging: x - word, y - part of speech (verb, noun, etc.)
- 2 Genome annotation: x - DNA, y - genes
- 3 HEP tracking: x - a set of hits with backgrounds, y - hit classification

Sequence labelling tasks

POS tagging and Named Entity Recognition

X (words)	the	cat	sat	on	a	mat
Y (tags)	DET	NOUN	VERB	PREP	DET	NOUN

Table: POS tagging

Alex	is	going	to	Los	Angeles
B-PER	O	O	O	B-LOC	I-LOC

Table: NER (IOB2)

Alex	travels	with	Marty	A.	Rick	to	NY	city
S-PER	O	O	B-PER	I-PER	E-PER	O	B-LOC	E-LOC

Table: NER (IOBES)

Sequence modelling III

Sequence transduction / transformation

- ① $\mathbf{x} = x_1, x_2, \dots, x_n, x_i \in V_{source}$ - objects
- ② $\mathbf{y} = y_1, y_2, \dots, y_n, y_i \in V_{target}$ - objects
- ③ $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$ – training data
- ④ $\mathbf{x}^{(1)}, \mathbf{y}^{(1)}$ are of different length

Transduction problem: $\mathbf{x}_{source} \rightarrow \mathbf{y}_{target}$

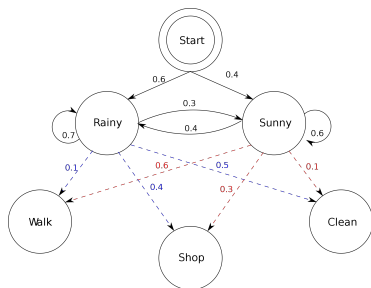
- ① Machine translation: x – sentence in German, y – sentence in English
- ② Speech recognition: x – spoken language, y – text
- ③ Chat bots: x – question, y – answer

Traditional ML approaches to sequence modeling

- 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

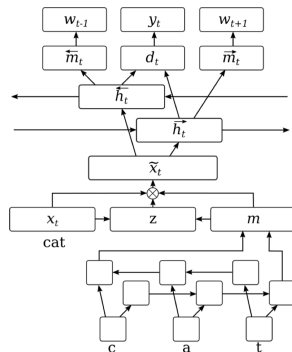


DL approaches to sequence modeling

- Neural networks
- Recurrent neural network and its modifications: LSTM, GRU, Highway
- 2D Convolutional Neural Network
- Transformer
- Pointer network

Problems:

- 1 Training time
- 2 Amount of training data



Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

Neural language model [1]

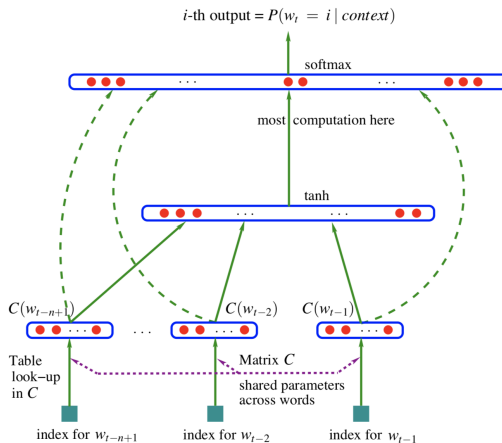
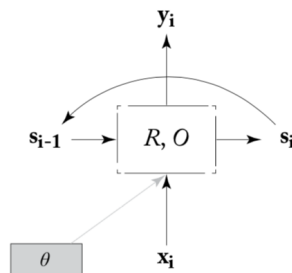


Figure: Neural language model

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 $x_{1:i}$ 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 classification

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

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

2 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

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

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
- $x_i \in \mathbb{R}^{d_{in}}$, $y_i \in \mathbb{R}^{d_{out}}$, $s_i \in \mathbb{R}^{d_{out}}$
- θ – shared weights

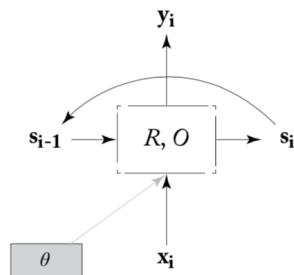


Figure: Goldberg, Yoav. Neural network methods for natural language processing

More details on RNN

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

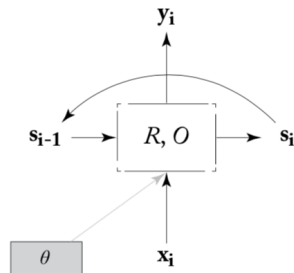
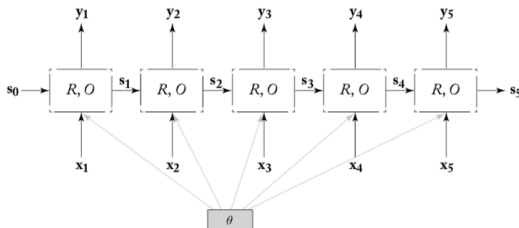


Figure: Goldberg, Yoav. Neural network methods for natural language processing

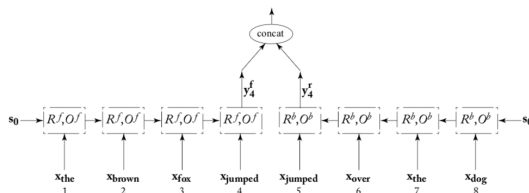
RNN unrolled



$$s_4 = R(s_3, x_4) = R(R(s_2, x_3), x_4) = R(R(R(s_1, x_2), x_3), x_4) = \\ = R(R(R(R(s_0, x_1), x_2), x_3), x_4)$$

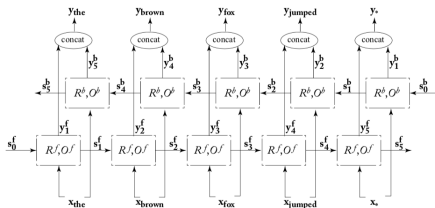
Bidirectional RNN (Bi-RNN)

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



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

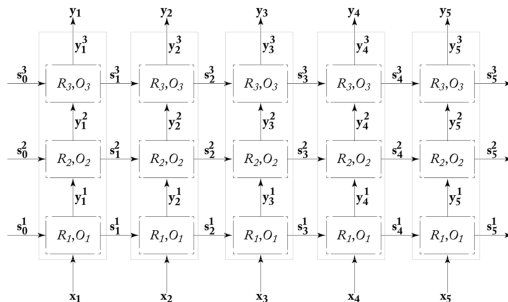
Bi-RNN



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

Figure: Goldberg, Yoav. Neural network methods for natural language processing

Multilayer RNN



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

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.

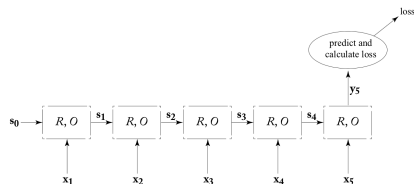


Figure: Goldberg, Yoav. Neural network methods for natural language processing

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.

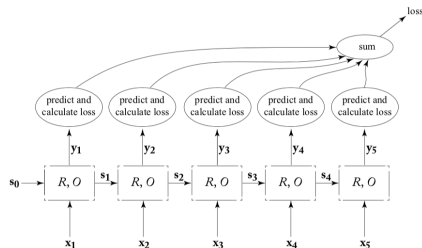
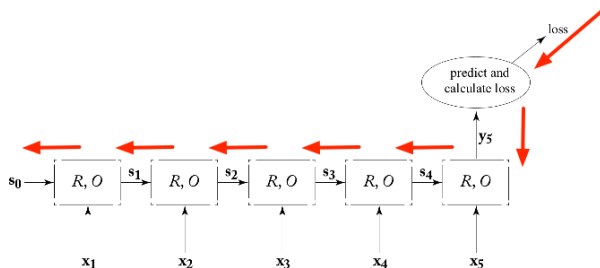


Figure: Goldberg, Yoav. Neural network methods for natural language processing

Backpropagation through time



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

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

Vanishing gradient problem

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

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

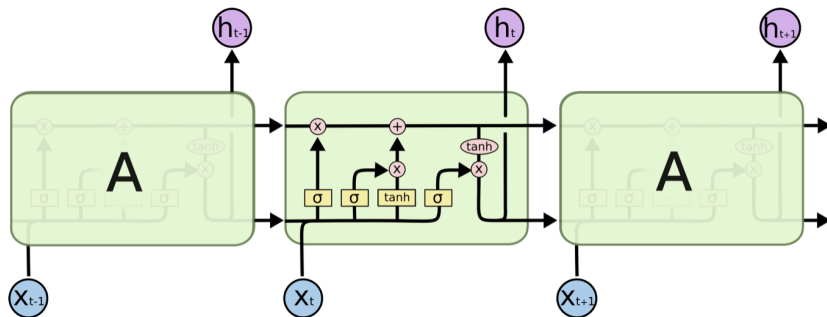
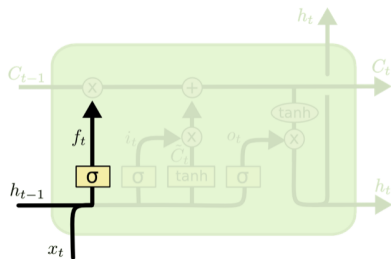


Figure: colahblog<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

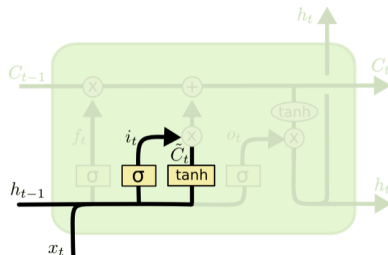
Long short term memory



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

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

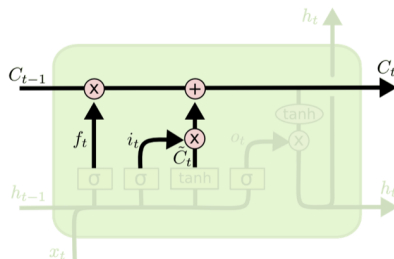
Long short term memory



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

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

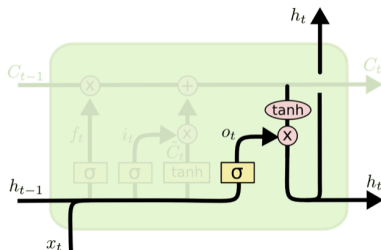
Long short term memory



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

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

Long short term memory

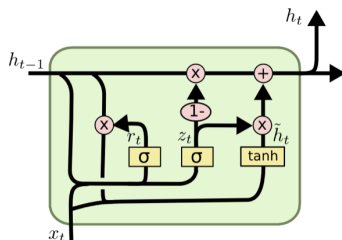


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

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

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

Gated recurrent unit



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

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

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

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

Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators**
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

Language model

- 1 Compute the probability of a sequence of words:

$$P(w_1, w_2, \dots, w_n)$$

- 2 Predict next word:

$$P(w_n | w_1, w_2, \dots, w_{n-1})$$

Perplexity

$$2^{H(p)} = 2^{\frac{1}{|V|} - \sum_x \log_2 p(x)}$$

Sequence generation

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

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

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

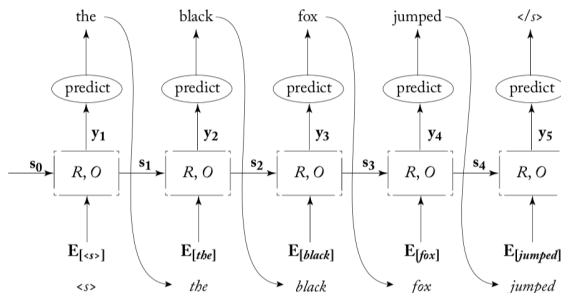


Figure: Goldberg, Yoav. Neural network methods for natural language processing

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-68786834d4c5>

Pros and cons of RNNs

① Advantages:

- ▶ RNNs are popular and successful for variable-length sequences
- ▶ The gating models such as LSTM are suited for long-range error propagation

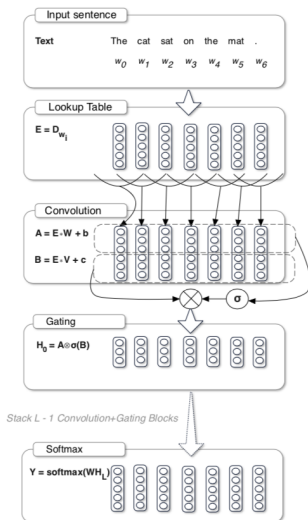
② Problems:

- ▶ The sequentiality prohibits parallelization within instances
- ▶ Long-range dependencies still tricky, despite gating

Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM**
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

Language Modeling with Gated Convolutional Networks



- **Embeddings** $\in D^{|V| \times e}$
- **Input:** $w_0, \dots, w_n \rightarrow E = [D_{w_0}, \dots, D_{w_n}]$
- **Hidden layers:** h_0, \dots, h_n :

$$h_l(X) = (X \times W + b) \circ \sigma(X \times V + c)$$
- **Gated linear unit:** $X \circ \sigma(X)$
- **Output:** $Y = \text{softmax}(WH_L)$

Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer**
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

The Transformer

An alternative architecture to RNN which allows of parallel and faster training

- Several layers of identical modules
- Each module consists of Multi-Head Attention and Feed Forward layers
- Input: embeddings. To get embeddings for numerical input, apply any dense layer
- Positional embeddings to make use of the order of the sequence

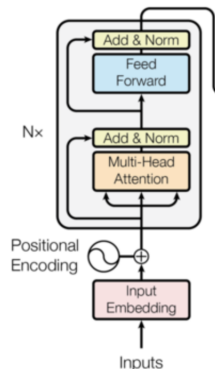


Figure: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 2017.

Scaled Dot-Product Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where the input consists of queries Q and keys K of dimension d_k and values V of dimension d_v

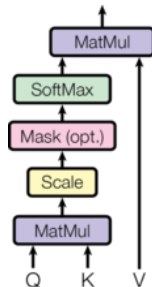


Figure: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 2017.

Multi-head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions

$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1, \dots, \text{head}_h) W^O,$$

where

$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and W are projection matrices.

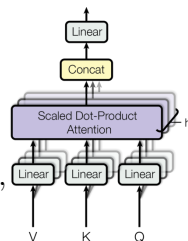


Figure: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 2017.

The Transformer

Bringing it all together:

- LayerNorm: $\frac{x - \mu}{\sigma}$
- Residual connection:
 $\text{LayerNorm}(x + \text{Sublayer}(x))$
- Position-wise Feed-Forward Networks:
 $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$

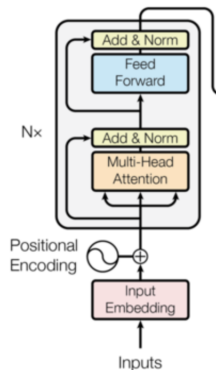


Figure: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 2017.

Positional Encoding

We need to inject some information about the relative or absolute position of x_{pos} in the sequence:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Positional encoding: $x = x + PE(x)$

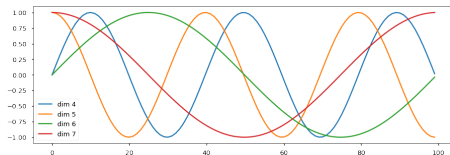


Figure: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 2017.

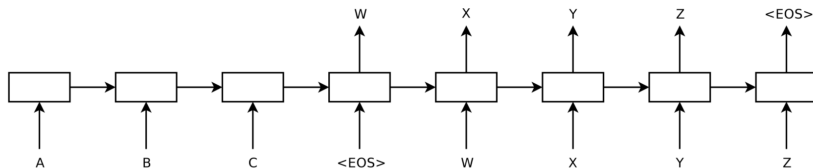
Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq**
- 7 Take-home message
- 8 Bonus: Recursive NN

Sequence 2 sequence learning

Encoder-decoder model for:

- 1 Machine translation
- 2 Chit-chat bots



Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message**
- 8 Bonus: Recursive NN

Take-home message

- There is a lot of sequential data around us
- Before DL: HMM, MEMM
- Mid 2010 DL: RNN, LSTM, etc
- Late 2010 DL: the Transformer
- 2020: stack more transformer blocks (Trasformer XL)

Today

- 1 Sequence modelling
- 2 Recurrent neural network
 - Definition
 - Training
 - Gated architectures
- 3 RNN generators
- 4 Convolutional LM
- 5 The Transformer
- 6 seq2seq
- 7 Take-home message
- 8 Bonus: Recursive NN

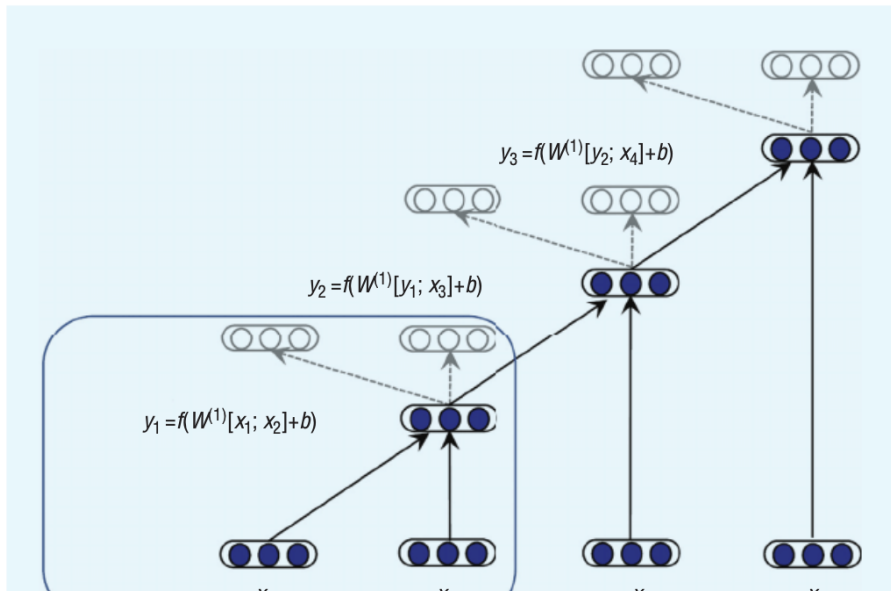
Modeling trees with Recursive NN

- 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

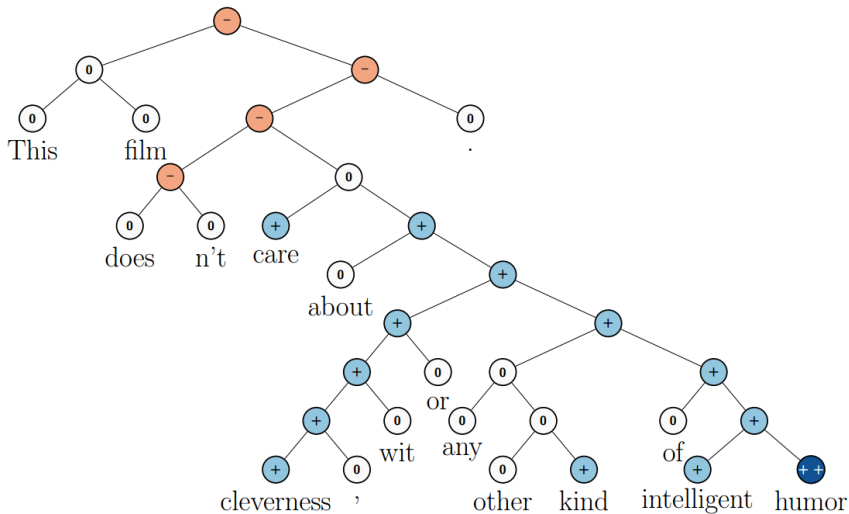
RecNN

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

RecNN



RecNN [2]



Reference I



Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.



R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

Reading

- ① Neural Networks for NLP. Joav Goldberg, Ch. 14-16
- ② Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach, Ch. 15
- ③ Dan Jurafsky, James H. Martin. Speech and Language Processing, Ch. 3, Ch. 8