Deep Learning for Natural Language Processing

Xipeng Qiu
xpqiu@fudan.edu.cn
http://nlp.fudan.edu.cn
http://nlp.fudan.edu.cn

Fudan University

2016/5/29, CCF ADL, Beijing

Table of Contents

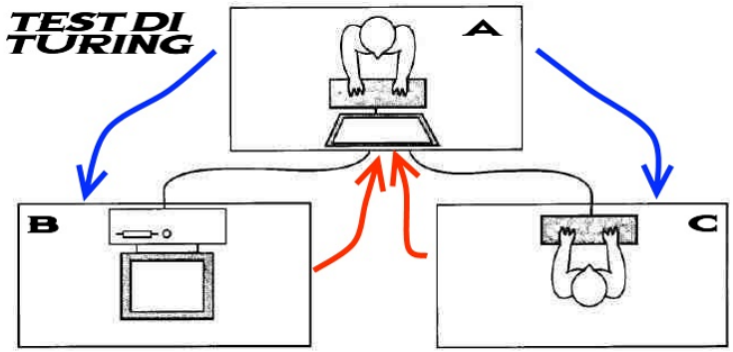
Table of Contents

Begin with "AI"

Human: Memory, Computation

Computer: Learning, Thinking, Creativity

Turing Test



Artificial Intelligence

Definition from Wikipedia

Artificial intelligence (AI) is the intelligence exhibited by machines. Colloquially, the term "artificial intelligence" is likely to be applied when a machine uses cutting-edge techniques to competently perform or mimic "cognitive" functions that we intuitively associate with human minds, such as "learning" and "problem solving".

Research Topics

- Knowledge Representation
- Machine Learning
- Natural Language Processing
- Computer Vision
- ...

Challenge: Sematic Gap

Text in Human

床前明月光，
疑是地上霜。
举头望明月，
低头思故乡。

Challenge: Sematic Gap

Text in Computer

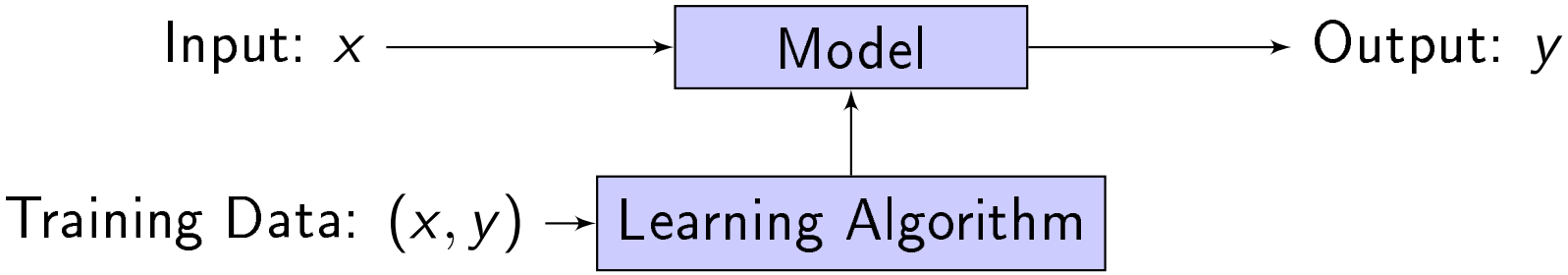
11100101101110101000101011100101100010011000110111100110100110001000
11011100110100111001000100011100101100001011000100111101111101111001000
11100111100101101001000111100110100110001010111111001011001110010110
00011100100101110001000101011101001100111001001110011100011100000001000
00100000001000000010000000001010111001001011100010111110111001011010
00101101001110011010011100100110111110011010011000100011101110011010011100
10001000111011111011110010001100000010101110010010111011000111011100
01101001001011010011100110100000001001110111100110100101011000010111100
11100110100001111000111000000010000010

Challenge: Sematic Gap

Figure: Guernica (Picasso)

Table of Contents

Machine Learning



Basic Concepts of Machine Learning

- Input Data: $(x_i, y_i), 1 \leq i \leq m$
- Model:
  - Linear Model: $y = f(x) = w^T x + b$
  - Generalized Linear Model: $y = f(x) = w^T \phi(x) + b$
  - Non-linear Model: Neural Network
- Criterion:
  - Loss Function:
    $L(y, f(x)) \rightarrow$ Optimization
  - Empirical Risk:
    $Q(\theta) = \frac{1}{m} \cdot \sum_{i=1}^{m} L(y_i, f(x_i, \theta)) \rightarrow$ Minimization
  - Regularization: $\|\theta\|^2$
- Objective Function: $Q(\theta) + \lambda \|\theta\|^2$

Loss Function

Given an test sample (x, y), the predicted label is $f(x, \theta)$

0-1 Loss

$$L(y, f(x,\theta)) = \begin{cases} 0 & \textbf{if } y = f(x,\theta) \\ 1 & \textbf{if } y \neq f(x,\theta) \end{cases} \tag{1}$$

$$= I(y \neq f(x,\theta)), \tag{2}$$

here I is indicator function.

Quadratic Loss

$$L(y, \hat{y}) = (y - f(x,\theta))^2 \tag{3}$$

Loss Function

Cross-entropy Loss We regard $f_i(x, \theta)$ as the conditional probability of class i.

$$f_i(x, \theta) \in [0, 1], \qquad \sum_{i=1}^{C} f_i(x, \theta) = 1 \tag{4}$$

$f_y(x, \theta)$ is the likelihood function of y. Negative Log Likelihood function is

$$L(y, f(x, \theta)) = -\log f_y(x, \theta). \tag{5}$$

Loss Function

We use one-hot vector $\mathbf{y}$ to represent class c in which $y_c = 1$ and other elements are 0.
Negative Log Likelihood function can be rewritten as

$$L(y, f(x, \theta)) = -\sum_{i=1}^{C} y_i \log f_i(x, \theta). \quad (6)$$

y_i is distribution of gold labels. Thus, Eq 6 is Cross Entropy Loss function.

Loss Function

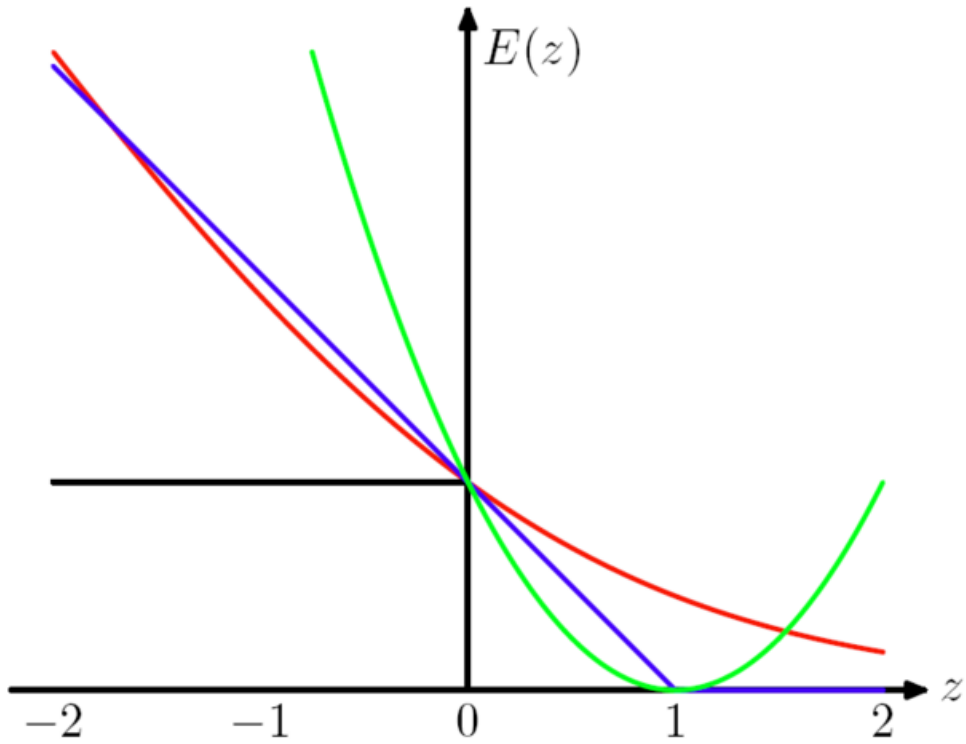
Hinge Loss For binary classification, y and $f(x, \theta)$ are in $\{-1, +1\}$. Hinge Loss is

$$L(y, f(x, \theta)) = \max(0, 1 - yf(x, \theta)) \tag{7}$$

$$= |1 - yf(x, \theta)|_+. \tag{8}$$

Loss Function

For binary classification, y and $f(x, \theta)$ are in $\{-1, +1\}$.
$z = yf(x, \theta)$.



http://www.cs.cmu.edu/ yandongl/loss.html

Parameter Learning

In ML, our objective is to learn the parameter θ to minimize the loss function.

$$\theta^* = \arg\min_{\theta} \mathcal{R}(\theta_t) \tag{9}$$

$$= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(y^{(i)}, f(x^{(i)}, \theta)\big). \tag{10}$$

Gradient Descent:

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \tag{11}$$

$$= \mathbf{a}_t - \lambda \sum_{i=1}^{N} \frac{\partial \mathcal{R}\big(\theta_t; x^{(i)}, y^{(i)}\big)}{\partial \theta}, \tag{12}$$

λ is also called **Learning Rate** in ML.

Stochastic Gradient Descent (SGD)

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}\left(\theta_t; x^{(t)}, y^{(t)}\right)}{\partial \theta}, \tag{13}$$

Two Tricks of SGD

- Early-Stop
- Shuffle

Linear Classification

For binary classification $y \in \{0, 1\}$, the classifier is

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} > 0 \\ 0 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} \leq 0 \end{cases} = I(\mathbf{w}^{\mathrm{T}}\mathbf{x} > 0), \tag{14}$$

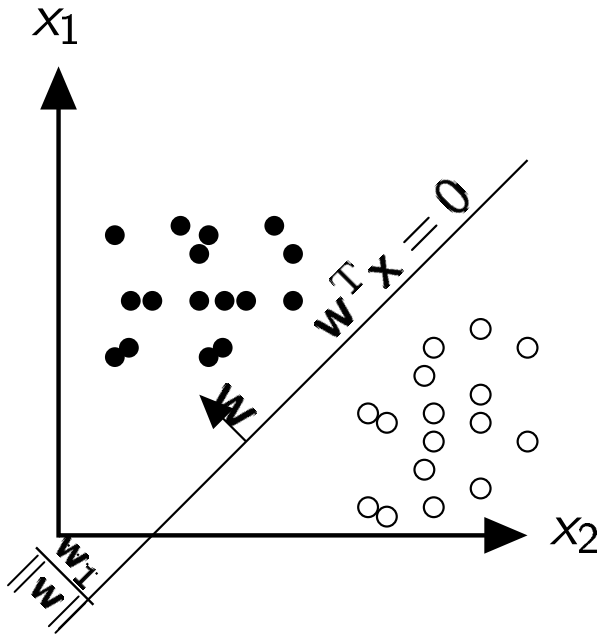


Figure: Binary Linear Classification

Logistic Regression

How to learn the parameter $\mathbf{w}$: Perceptron, Logistic Regression, etc.
The posterior probability of $y = 1$ is

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}, \tag{15}$$

where, $\sigma(\cdot)$ is logistic function.
The posterior probability of $y = 0$ is $P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x})$.

Logistic Regression

Given n samples $(x^{(i)}, y^{(i)}), 1 \leq i \leq N$, we use the cross-entropy loss function.

$$\mathcal{J}(\mathbf{w}) = -\sum_{i=1}^{N} \left(y^{(i)} \log \left(\sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)}) \right) \right) \tag{16}$$

The gradient of $\mathcal{J}(\mathbf{w})$ is

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \left(\mathbf{x}^{(i)} \cdot \left(\sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)}) - y^{(i)} \right) \right) \tag{17}$$

Initialize $\mathbf{w}_0 = 0$, and update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda \frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}}, \tag{18}$$

Multiclass Classification

Generally, $y = \{1, \cdots, C\}$, we define C discriminant functions

$$f_c(\mathbf{x}) = \mathbf{w}_c^{\mathrm{T}}\mathbf{x}, \qquad c = 1, \cdots, C, \tag{19}$$

where $\mathbf{w}_c$ is weight vector of class c.
Thus,

$$\hat{y} = \arg\max_{c=1}^{C} \mathbf{w}_c^{\mathrm{T}}\mathbf{x} \tag{20}$$

Softmax Regression

Softmax regression is a generalization of logistic regression to multi-class classification problems.
With **softmax**, the posterior probability of $y = c$ is

$$P(y = c|\mathbf{x}) = \mathbf{softmax}(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}) = \frac{\exp(\mathbf{w}_c^{\top}\mathbf{x})}{\sum_{i=1}^{C}\exp(\mathbf{w}_i^{\top}\mathbf{x})}. \tag{21}$$

To represent class c by one-hot vector

$$\mathbf{y} = [I(1 = c), I(2 = c), \cdots, I(C = c)]^{\mathrm{T}}, \tag{22}$$

where $I()$ is indictor function.

Softmax Regression

Redefine Eq 21,

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{softmax}(W^{\mathrm{T}}\mathbf{x}) \\ &= \frac{\exp(W^{\mathrm{T}}\mathbf{x})}{\underline{1}^{\mathrm{T}}\exp\left((W^{\mathrm{T}}\mathbf{x})\right)} \\ &= \frac{\exp(\mathbf{z})}{\underline{1}^{\mathrm{T}}\exp\left(\mathbf{z}\right)},\end{aligned} \tag{23}$$

where,
$W = [\mathbf{w}_1, \cdots, \mathbf{w}_C]$,
$\hat{\mathbf{y}}$ is predicted posterior probability,
$\hat{\mathbf{z}} = W^{\mathrm{T}}\mathbf{x}$ is input of **softmax** function.

Softmax Regression

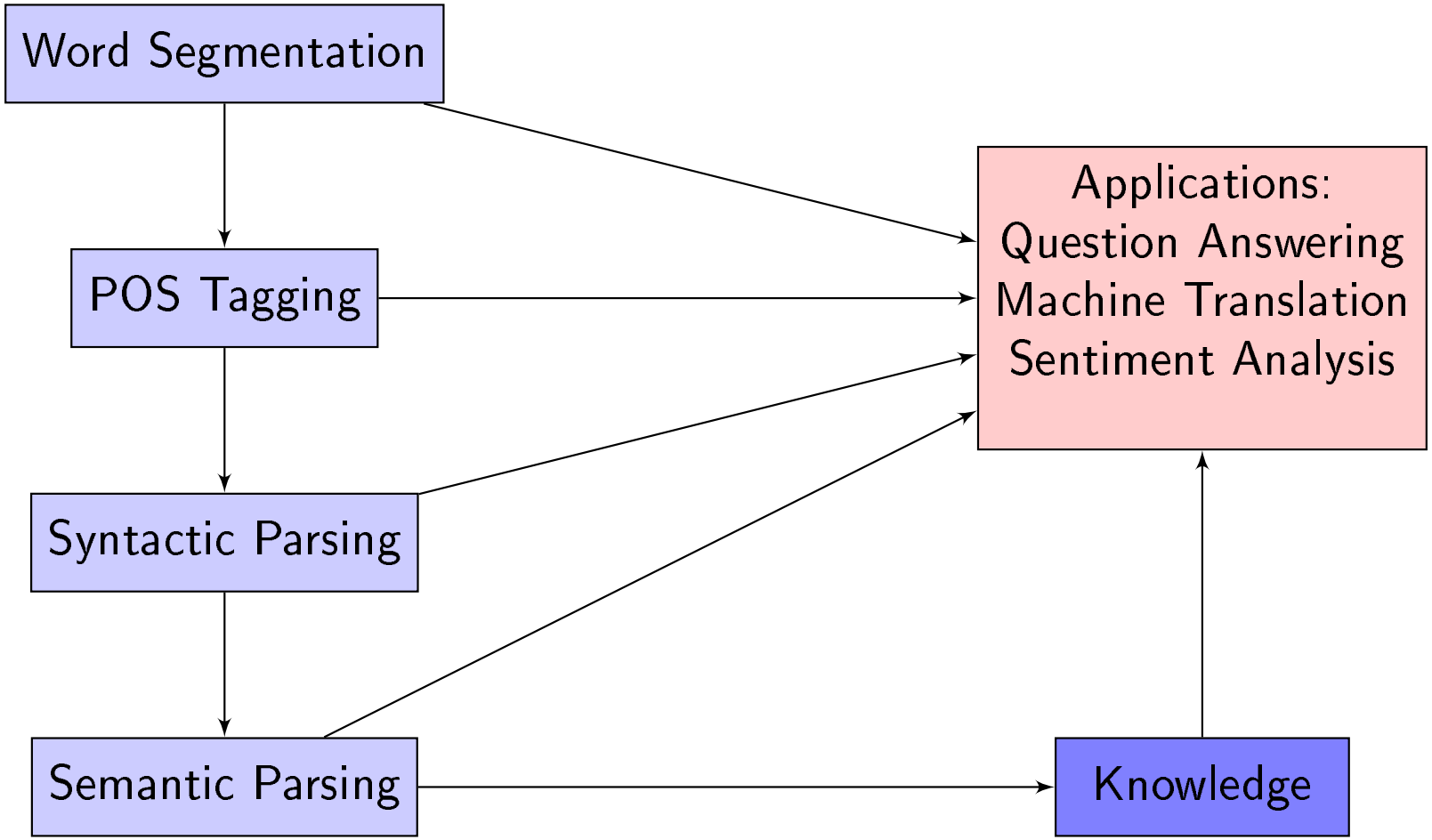
Given training set $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), 1 \leq i \leq N$, the cross-entropy loss is

$$\mathcal{J}(W) = -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbf{y}_c^{(i)} \log \hat{\mathbf{y}}_c^{(i)} = -\sum_{i=1}^{N} (\mathbf{y}^{(i)})^{\mathrm{T}} \log \hat{\mathbf{y}}^{(i)}$$

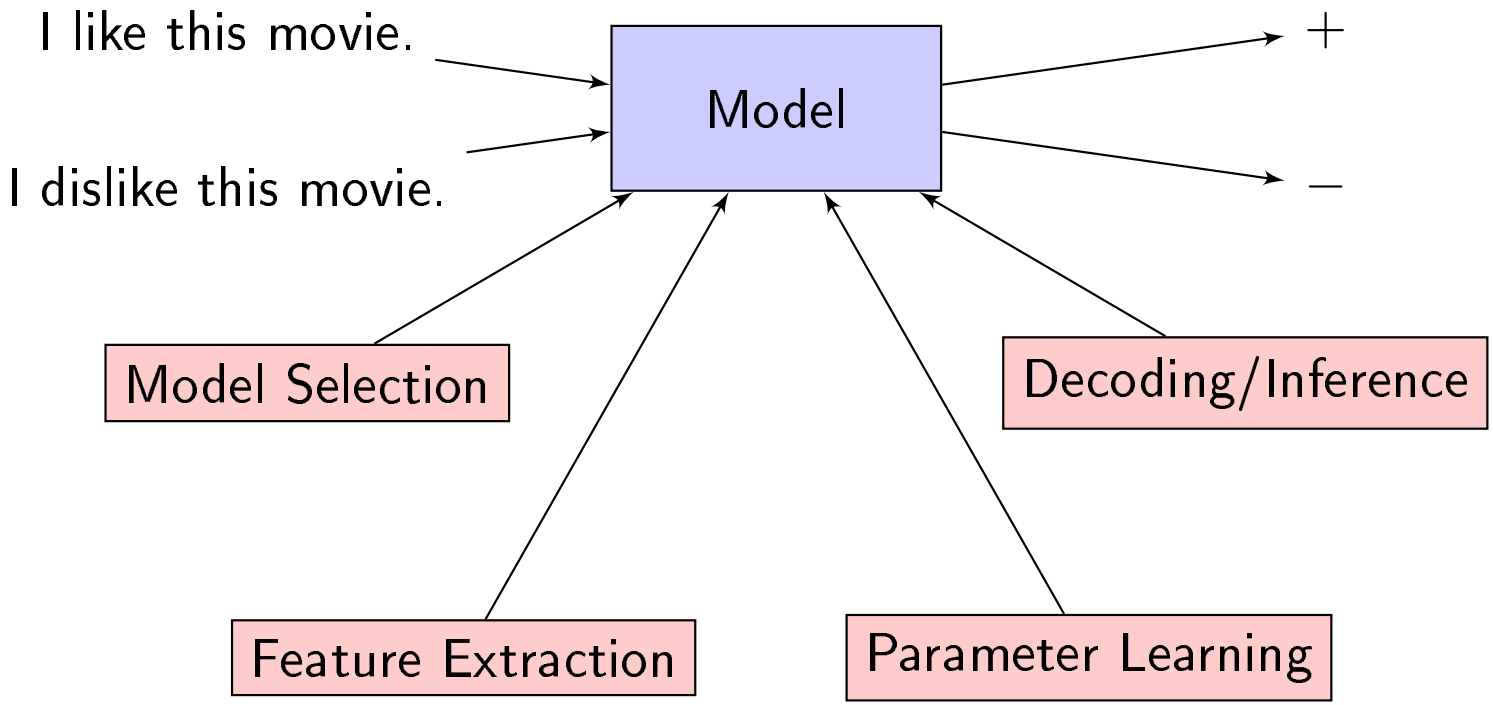
The gradient of $\mathcal{J}(W)$ is

$$\frac{\partial \mathcal{J}(W)}{\partial \mathbf{w}_c} = -\sum_{i=1}^{N} \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)_c \tag{24}$$

The idea pipeline of NLP

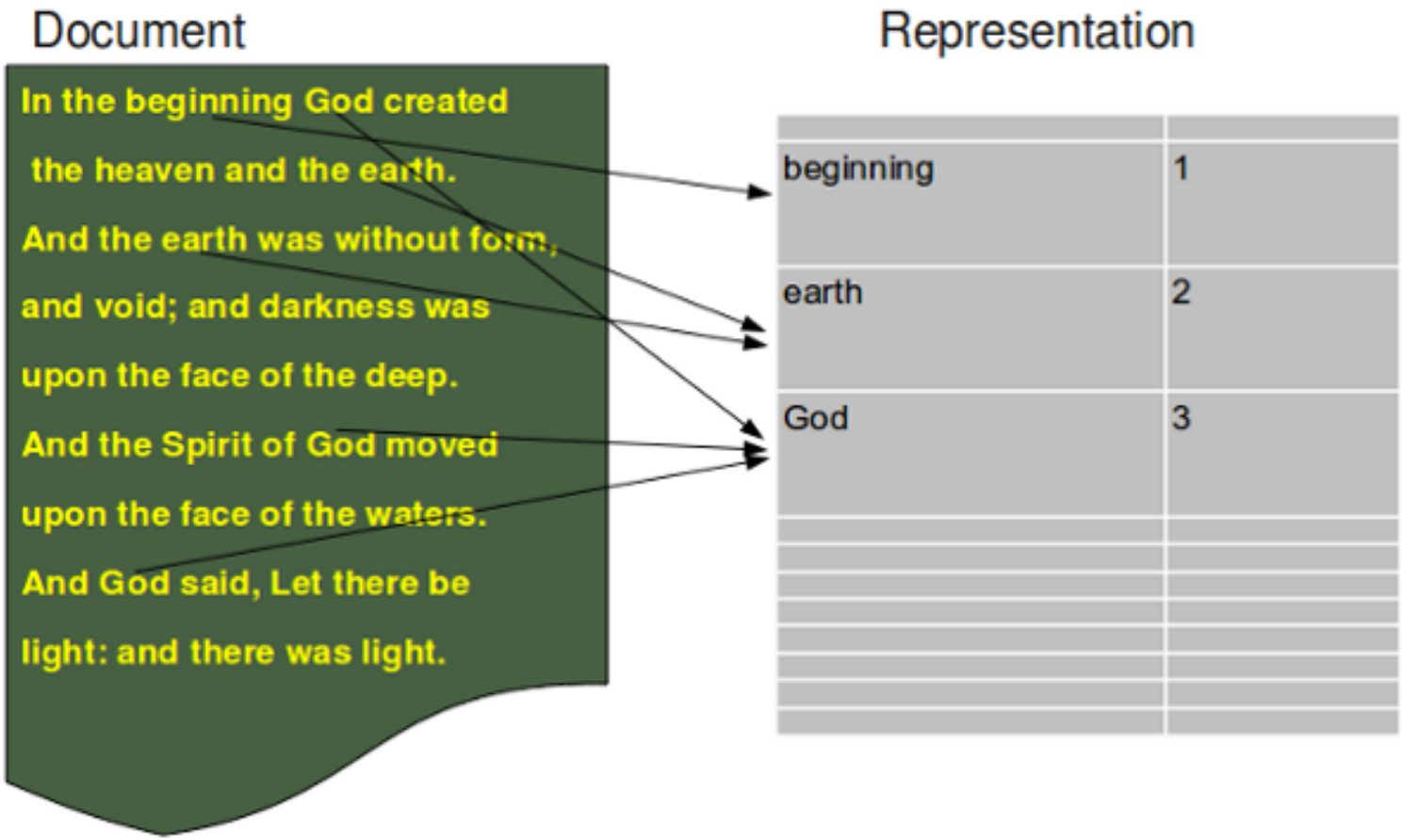


But in practice: End-to-End



Feature Extraction

Bag-of-Word



Text Classification

根据文本内容来判断文本的相应类别

D_1：“我喜欢读书”

D_2：“我讨厌读书”

	我	喜欢	讨厌	读书
D_1	1	1	0	1
D_2	1	0	1	1

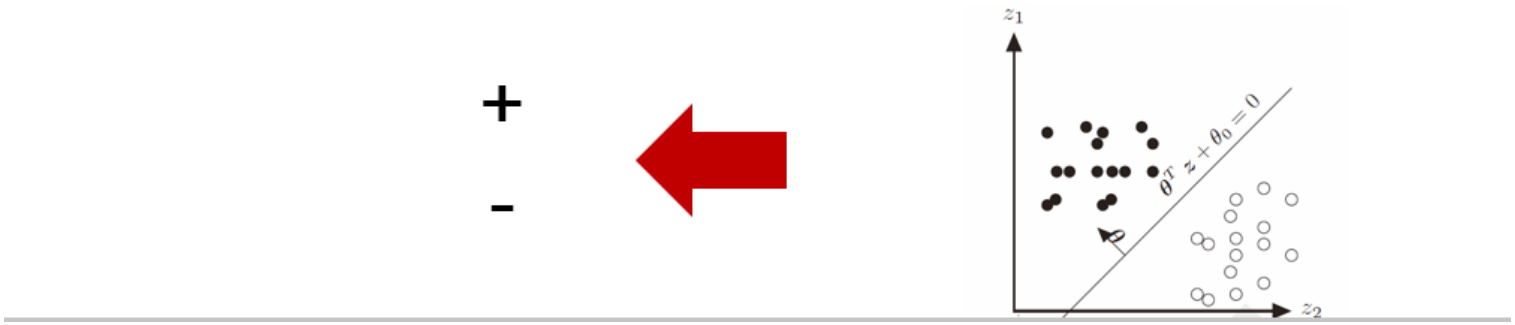
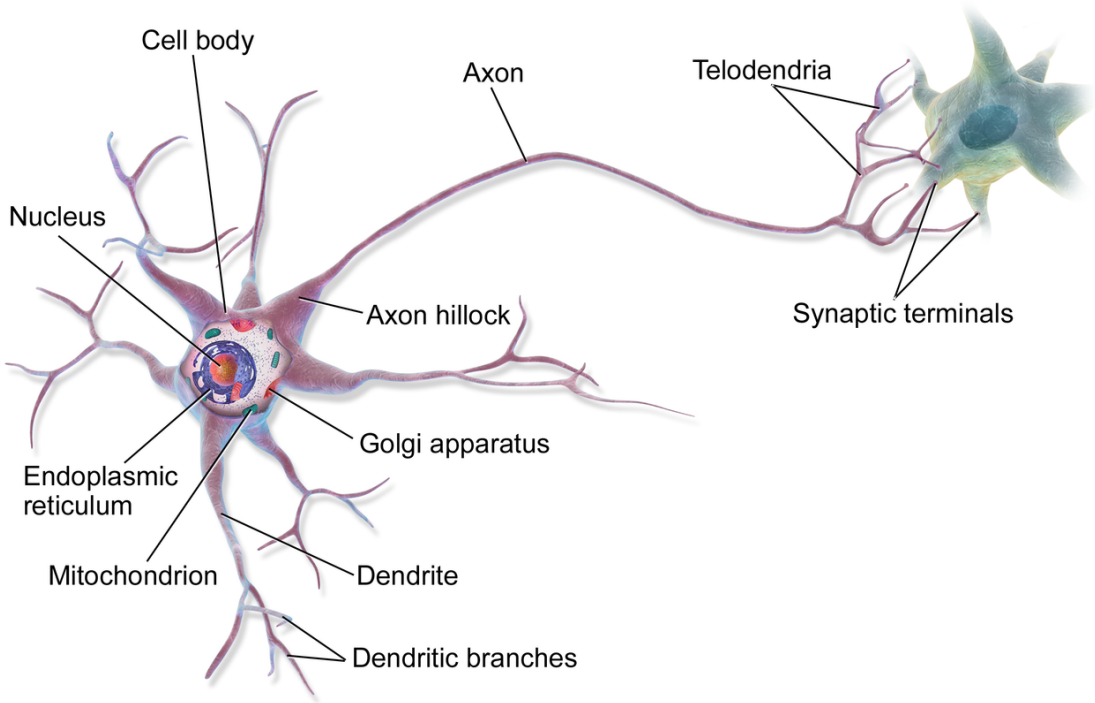


Table of Contents

Artificial Neural Network

Artificial neural networks[1] (ANNs) are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain).
Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other.



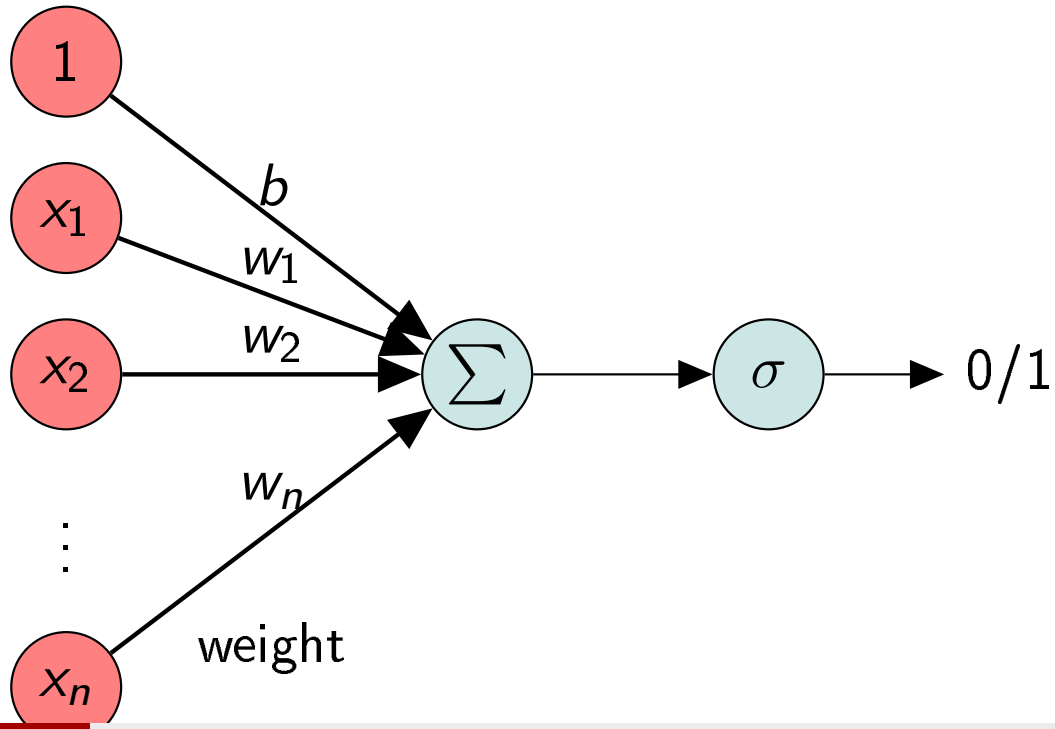
[1] https://en.wikipedia.org/wiki/Artificial_neural_network

Artificial Neuron

Input: $\mathbf{x} = (x_1, x_2, \cdots, x_n)$
State: z
Output: a

$$z = \mathbf{w}^\top \mathbf{x} + b \tag{25}$$

$$a = f(z) \tag{26}$$



Activation Function

Sigmoid Function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{27}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{28}$$

$\tanh(x) = 2\sigma(2x) - 1$

Activation Function

rectifier function[2], also called rectified linear unit (ReLU)[3].

$$\textbf{rectifier}(x) = \max(0, x) \tag{29}$$

softplus function[4]:

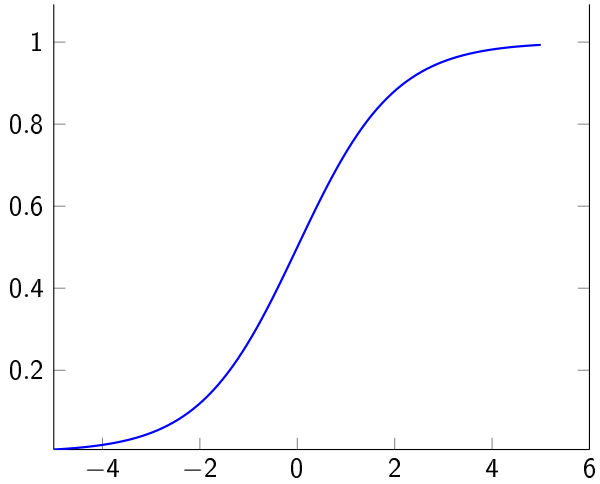
$$\textbf{softplus}(x) = \log(1 + e^x) \tag{30}$$

[2] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks". In: International Conference on Artificial Intelligence and Statistics. 2011, pp. 315–323.

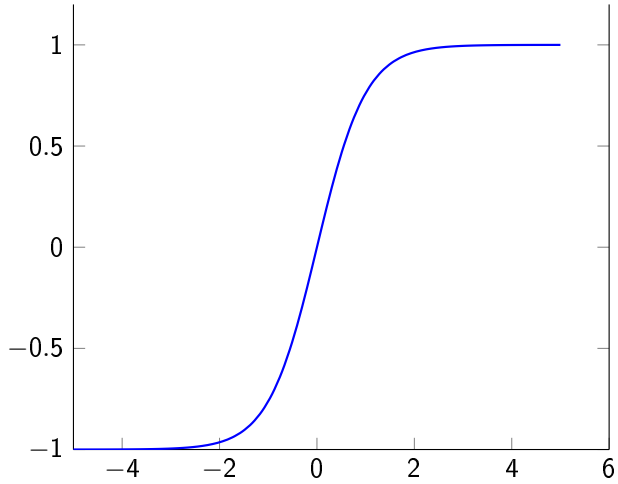
[3] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010, pp. 807–814.

[4] C. Dugas et al. "Incorporating second-order functional knowledge for better option pricing". In: Advances in Neural Information Processing Systems (2001).

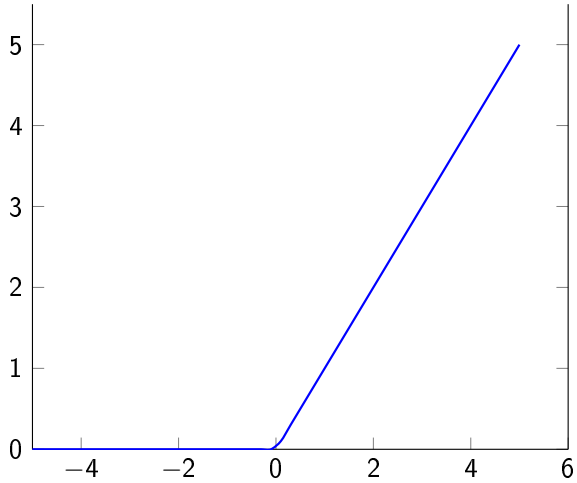
Activation Function



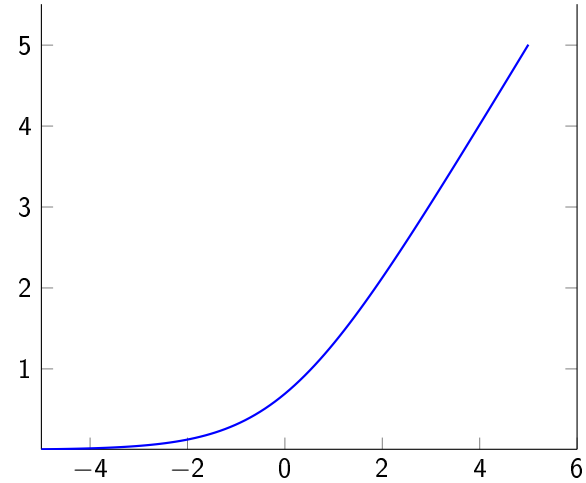
(a) logistic



(b) tanh



(c) rectifier



(d) softplus

Types of Artificial Neural Network[5]

- Feedforward neural network, also called Multilayer Perceptron (MLP).
- Recurrent neural network.

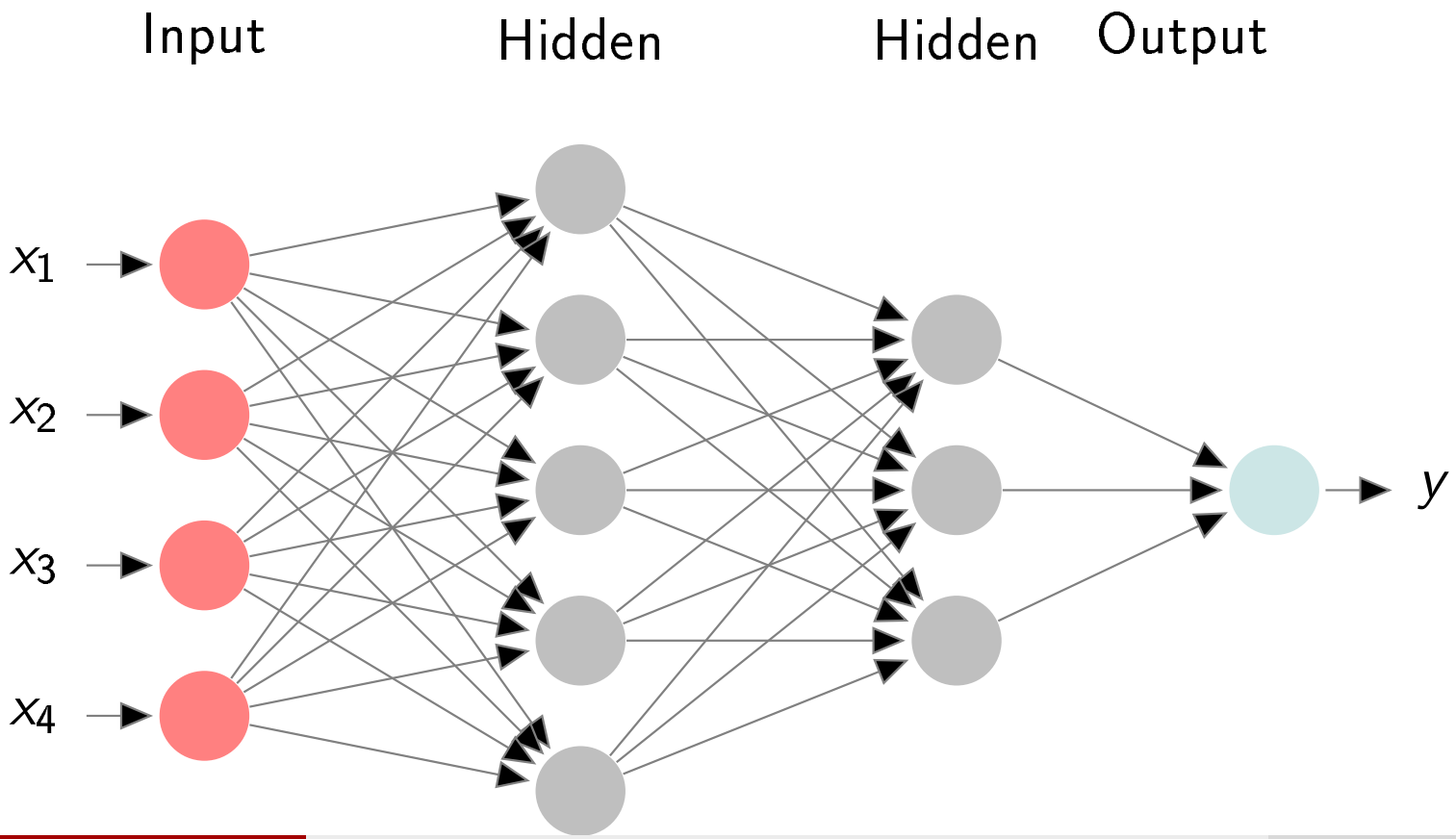
[5] https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks

Basic Concepts of Deep Learning

- Model: Artificial neural networks that consist of multiple hidden non-linear layers.
- Function: Non-linear function $y = \sigma(\sum_i w_i x_i + b)$.

Feedforward Neural Network

In feedforward neural network, the information moves in only one direction forward: From the input nodes data goes through the hidden nodes (if any) and to the output nodes.
There are no cycles or loops in the network.



Feedforward Computing

Definitions:

- L Number of Layers;
- n^l Number of neurons in l-th layer;
- $f_l(\cdot)$ Activation function in l-th layer;
- $W^{(l)} \in \mathbb{R}^{n^l \times n^{l-1}}$ weight matrix between $l-1$-th layer and l-th layer;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n^l}$ bias vector between $l-1$-th layer and l-th layer;
- $\mathbf{z}^{(l)} \in \mathbb{R}^{n^l}$ state vector of neurons in l-th layer;
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n^l}$ activation vector of neurons in l-th layer.

$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{31}$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) \tag{32}$$

Feedforward Computing

$$\mathbf{z}^{(l)} = W^{(l)} \cdot f_l(\mathbf{z}^{(l-1)}) + \mathbf{b}^{(l)} \tag{33}$$

Thus,

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \cdots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} = y \tag{34}$$

Combining feedforward network and Machine Learning

Given training samples $(\mathbf{x}^{(i)}, y^{(i)}), 1 \leq i \leq N$, and feedforward network $f(\mathbf{x}|\mathbf{w}, \mathbf{b})$, the objective function is

$$J(W, \mathbf{b}) = \sum_{i=1}^{N} L(y^{(i)}, f(\mathbf{x}^{(i)}|W, \mathbf{b})) + \frac{1}{2}\lambda \|W\|_F^2, \tag{35}$$

$$= \sum_{i=1}^{N} J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)}) + \frac{1}{2}\lambda \|W\|_F^2, \tag{36}$$

where $\|W\|_F^2 = \sum_{l=1}^{L} \sum_{j=1}^{n^{l+1}} \sum_{j=}^{n^l} W_{ij}^{(l)}$.

Learning by GD

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial J(W, \mathbf{b})}{\partial W^{(l)}}, \tag{37}$$

$$= W^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}} \right) - \lambda W, \tag{38}$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}}, \tag{39}$$

$$= \mathbf{b}^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}} \right), \tag{40}$$

$$\tag{41}$$

Backpropogation

How to compute $\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial W^{(l)}}$?

$\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial W_{ij}^{(l)}}$

$$\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial W_{ij}^{(l)}} = \left(\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}}\right)^{\top} \frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}}. \tag{42}$$

We define $\delta^{(l)} = \frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{n^{(l)}}$.

Backpropogation

Because $\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$,

$$\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} = \frac{\partial (W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} = \begin{bmatrix} 0 \\ \vdots \\ a_j^{(l-1)} \\ \vdots \\ 0 \end{bmatrix} \leftarrow i\text{-th row} \tag{43}$$

Therefore,

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)} \tag{44}$$

(45)

Backpropogation

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^{\top}. \tag{46}$$

In the same way,

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}. \tag{47}$$

How to compute $\delta^{(l)}$?

$$\delta^{(l)} \triangleq \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \tag{48}$$

$$= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l+1)}} \tag{49}$$

$$= \mathbf{diag}(f_l'(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^\top \cdot \delta^{(l+1)} \tag{50}$$

$$= f_l'(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}), \tag{51}$$

Backpropogation Algorithm

Input : Training Set: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \cdots, N$, Iteration: T
Output: $W, \mathbf{b}$

Initialize $W, \mathbf{b}$;
for $t = 1 \cdots T$ **do**
  for $i = 1 \cdots N$ **do**
    (1) Feedforward Computing;
    (2) Compute $\delta^{(l)}$ by 51;
    (3) Compute gradient of parameters by 46 47;

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^{\top};$$

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)};$$

    (4) Update Parameter;

$$W^{(l)} = W^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}} \right) - \lambda W;$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}} \right);$$

  end
end

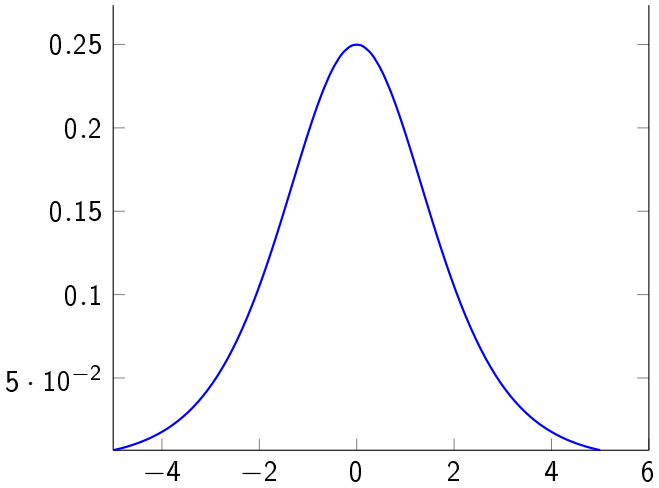
Gradient Vanishing

$$\delta^{(l)} = f_l'(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}, \tag{52}$$

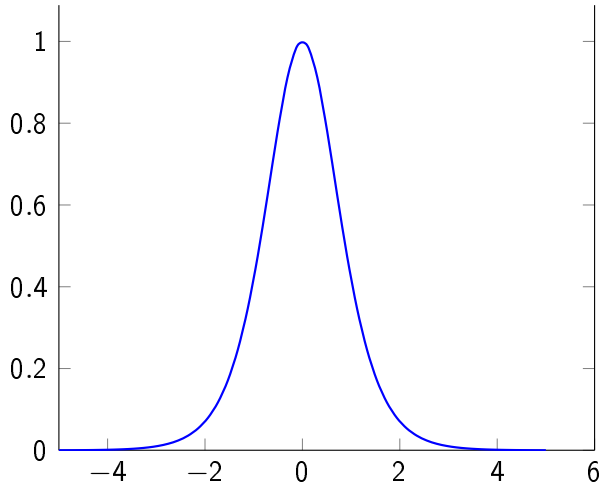
When we use sigmoid function, such as logistic $\sigma(x)$ and tanh,

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \in [0, 0.25] \tag{53}$$

$$\tanh'(x) = 1 - (\tanh(x))^2 \in [0, 1]. \tag{54}$$



(e) logistic



(f) tanh

Figure: Gradient

Difficulties

- Huge Parameters
- Non-convex Optimization
- Gradient Vanishing
- Poor Interpretability

Requirments

- High Computation
- Big Data
- Good Algorithms

Tricks and Skills[6] [7]

- Use ReLU non-linearities
- Use cross-entropy loss for classification
- SDG+mini-batch
  - Shuffle the training samples ($\longleftarrow$ very important)
  - Early-Stop
- Normalize the input variables (zero mean, unit variance)
- Schedule to decrease the learning rate
- Use a bit of L1 or L2 regularization on the weights (or a combination)
- Use "dropout" for regularization
- Data Argument

[6] G. B. Orr and K.-R. M ller. Neural networks: tricks of the trade. Springer, 2003.

[7] Geoff Hinton, Yoshua Bengio & Yann LeCun, Deep Learning, NIPS 2015 Tutorial.

Table of Contents

General Neural Architectures for NLP

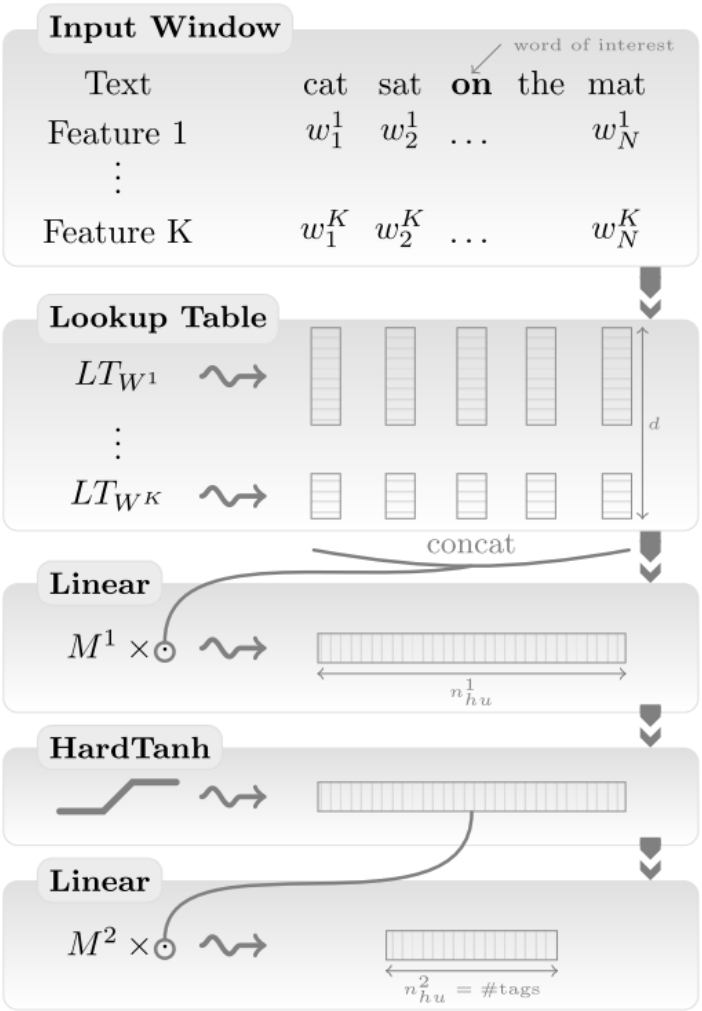
How to use neural network for the NLP tasks?

General Neural Architectures for NLP

How to use neural network for the NLP tasks?

Distributed Representation

General Neural Architectures for NLP[8]



1. represent the words/features with dense vectors (embeddings) by lookup table;
2. concatenate the vectors;
3. multi-layer neural networks.
   - classification
   - matching
   - ranking

[8] R. Collobert et al. "Natural language processing (almost) from scratch". In: The Journal of Machine Learning Research (2011).

Difference with the traditional methods

	Traditional methods	Neural methods
Features	Discrete Vector (One-hot Representation)	Dense Vector (Distributed Representation)
	High-dimension	Low-dimension
Classifier	Linear	Non-Linear

The key point is

how to encode the word, phrase, sentence, paragraph, or even document into the distributed representation?

Representation Learning

Representation Learning for NLP

- **Word Level**
  - NNLM
  - C&W
  - CBOW & Skip-Gram
- **Sentence Level**
  - **NBOW**
  - **Sequence Models**: Recurrent NN (LSTM/GRU), Paragraph Vector
  - **Topoligical Models**: Recursive NN,
  - **Convolutional Models**: Convolutional NN
- **Document Level**
  - **NBOW**
  - **Hierachical Models** two-level CNN
  - **Sequence Models** LSTM, Paragraph Vector

Let's start with Language Model

A **statistical language model** is a probability distribution over sequences of words.
A sequence W consists of L words.

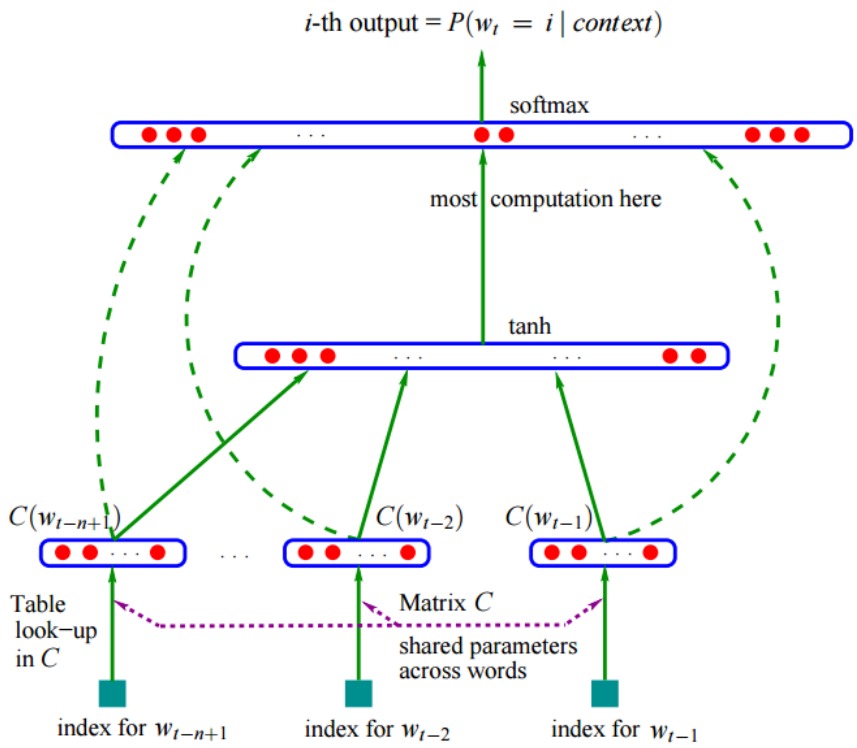
$$\begin{aligned} P(W) &= P(w_{1:L}) = P(w_1, \cdots, w_L) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\cdots P(w_L|w_{1:(L-1)}) \\ &= \prod_{i=1}^{L} P(w_i|w_{1:(i-1)}). \end{aligned} \tag{55}$$

n-gram model:

$$P(W) = \prod_{i=1}^{L} P(w_i|w_{(i-n+1):(i-1)}). \tag{56}$$

Neural Probabilistic Language Model[9]

- turn unsupervised learning into supervised learning;
- avoid the data sparsity of n-gram model;
- project each word into a low dimensional space.



[9] Y. Bengio, R. Ducharme, and P. Vincent. "A Neural probabilistic language model". In: Journal of Machine Learning Research (2003).

Problem of Very Large Vocabulary

Softmax output:

$$P_\theta^h = \frac{\exp(s_\theta(w,h))}{\sum_{w'} \exp(s_\theta(w',h))}, \tag{57}$$

Unfortunately both evaluating P_θ^h and computing the corresponding likelihood gradient requires normalizing over the entire vocabulary

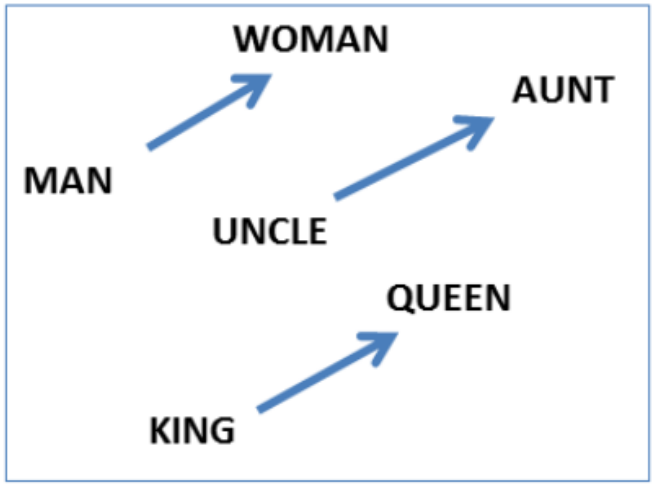
- Hierarchical Softmax: a tree-structured vocabulary[10]
- Negative Sampling[11], noise-contrastive estimation (NCE)[12]

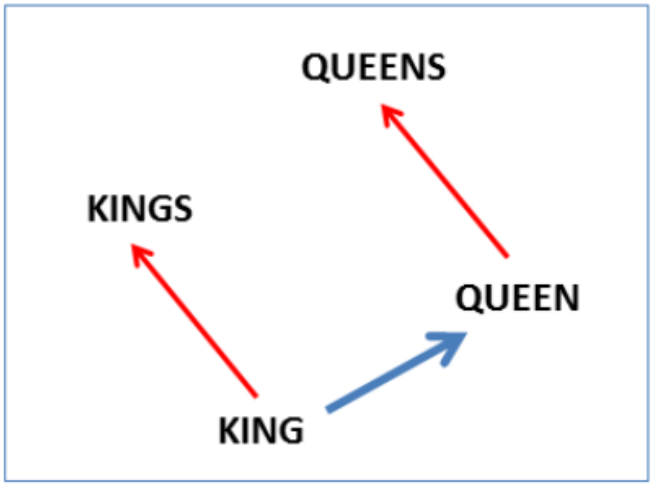
[10] A. Mnih and G. Hinton. "A scalable hierarchical distributed language model". In: Advances in neural information processing systems (2009); F. Morin and Y. Bengio. "Hierarchical Probabilistic Neural Network Language Model." In: Aistats. Vol. 5. Citeseer. 2005, pp. 246–252.

[11] T. Mikolov et al. "Efficient estimation of word representations in vector space". In: arXiv preprint arXiv:1301.3781 (2013).

[12] A. Mnih and K. Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation". In: Advances in Neural Information Processing Systems. 2013, pp. 2265–2273.

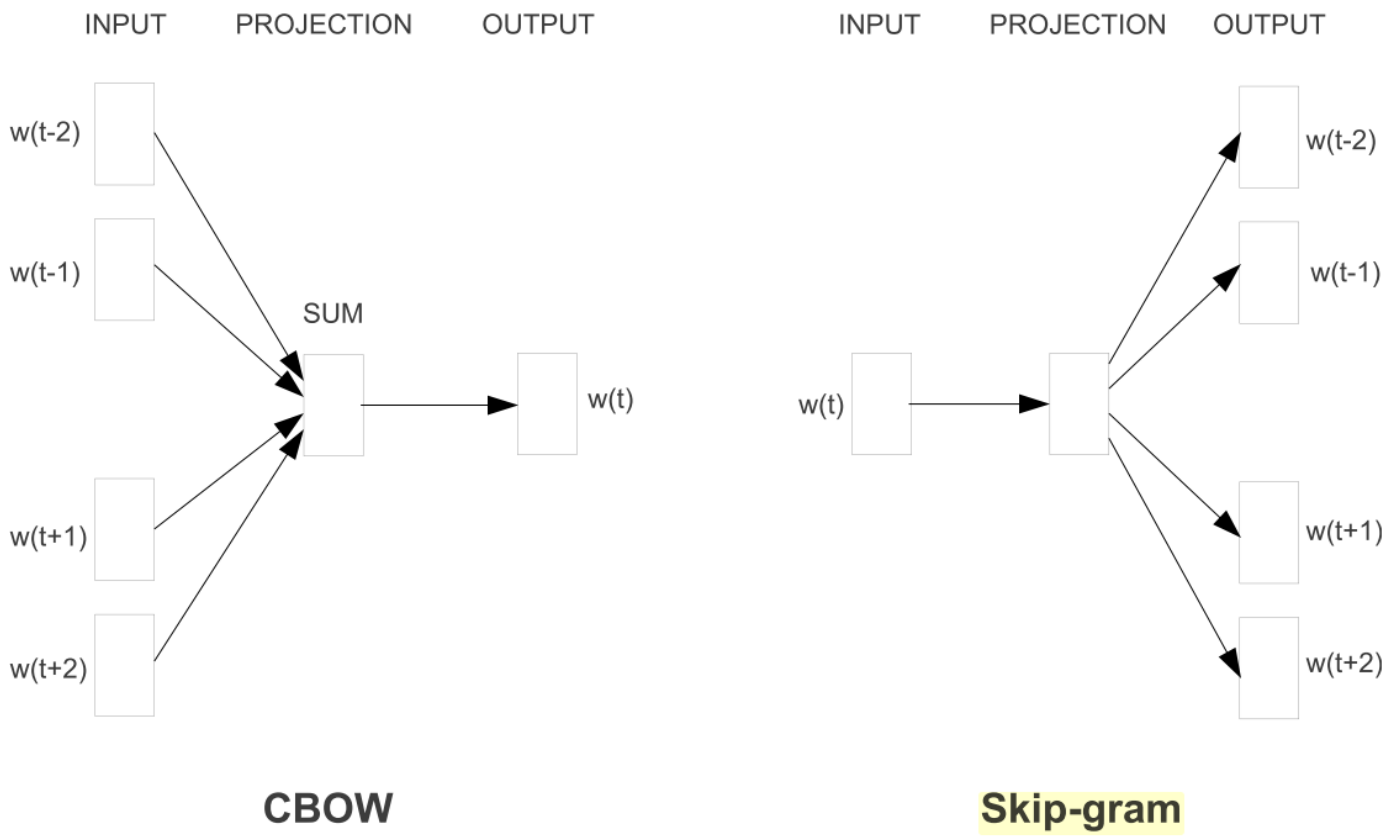
Linguistic Regularities of Word Embeddings[13]





[13] T. Mikolov, W.-t. Yih, and G. Zweig. "Linguistic Regularities in Continuous Space Word Representations." In: HLT-NAACL. 2013, pp. 746–751.

Skip-Gram Model[14]



[14] Mikolov et al., "Efficient estimation of word representations in vector space".

Skip-Gram Model

Given a pair of words (w, c), the probability that the word c is observed in the context of the target word w is given by

$$Pr(D = 1|w, c) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{c})},$$

where $\mathbf{w}$ and $\mathbf{c}$ are embedding vectors of w and c respectively.
The probability of not observing word c in the context of w is given by,

$$Pr(D = 0|w, c) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{c})}.$$

Skip-Gram Model with Negative Sampling

Given a training set $\mathcal{D}$, the word embeddings are learned by maximizing the following objective function:

$$J(\theta) = \sum_{w,c \in \mathcal{D}} Pr(D = 1|w, c) + \sum_{w,c \in \mathcal{D}'} Pr(D = 0|w, c),$$

where the set $\mathcal{D}'$ is randomly sampled negative examples, assuming they are all incorrect.

Table of Contents

Convolutional Neural Network

Convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network.
Distinguishing features[15]:

1. Local connectivity
2. Shared weights
3. Subsampling

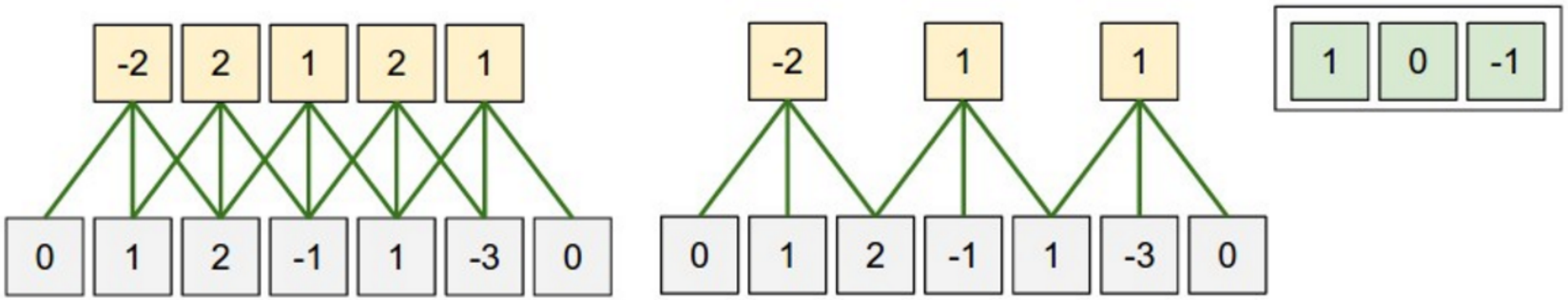
[15] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: Proceedings of the IEEE 11 (1998).

Convolution

Convolution is an integral that expresses the amount of overlap of one function as it is shifted over another function.
Given an input sequence $x_t,\ t = 1, \cdots, n$ and a **filter** $f_t,\ t = 1, \cdots, m$, the convolution is

$$y_t = \sum_{k=1}^{n} f_k \cdot x_{t-k+1}. \tag{58}$$

One-dimensional convolution



[15] Figure from: http://cs231n.github.io/convolutional-networks/

Two-dimensional convolution

Given an image x_{ij}, $1 \leq i \leq M$, $1 \leq j \leq N$ and filter f_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, the convolution is

$$y_{ij} = \sum_{u=1}^{m} \sum_{v=1}^{n} f_{uv} \cdot x_{i-u+1,j-v+1}. \tag{59}$$

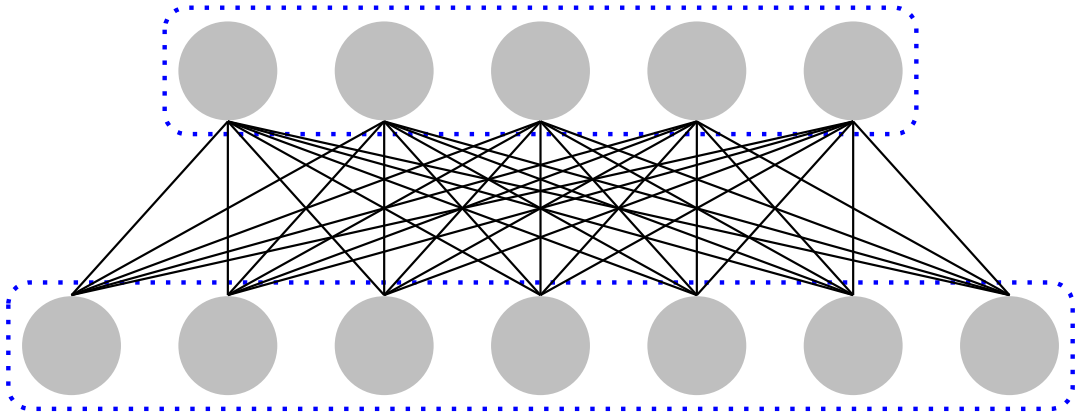
Mean filter: $f_{uv} = \frac{1}{mn}$

Convolutional Layer

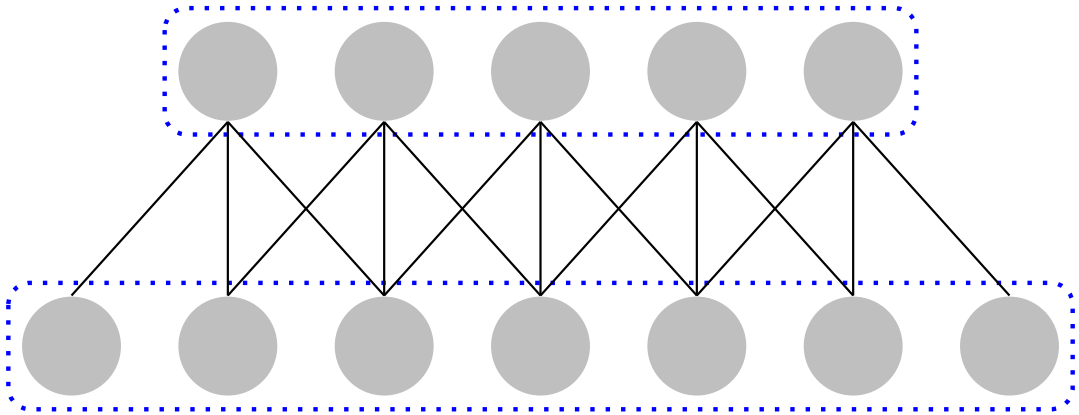
$$\mathbf{a}^{(l)} = f(\mathbf{w}^{(l)} \otimes \mathbf{a}^{(l-1)} + b^{(l)}), \tag{60}$$

$\otimes$ is convolutional operation.
$\mathbf{w}^{(l)}$ is shared by all the neurons of l-th layer.
Just need $m+1$ parameters, and $n^{(l+1)} = n^{(l)} - m + 1$.

Fully Connected Layer V.S. Convolutional Layer



(a) Fully Connected Layer



(b) Convolutional Layer

Pooling Layer

It is common to periodically insert a pooling layer in-between successive convolutional layers.

- progressively reduce the spatial size of the representation
- reduce the amount of parameters and computation in the network
- avoid overfitting

For a feature map $X^{(l)}$, we divide it into several (non-)overlapped regions $R_k, k = 1, \cdots, K$. A pooling function $\mathbf{down}(\cdots)$ is

$$X_k^{(l+1)} = f(Z_k^{(l+1)}), \tag{61}$$

$$= f\left(w^{(l+1)} \cdot \mathbf{down}(R_k) + b^{(l+1)}\right). \tag{62}$$

Pooling Layer

$$X^{(l+1)} = f(Z^{(l+1)}), \tag{63}$$

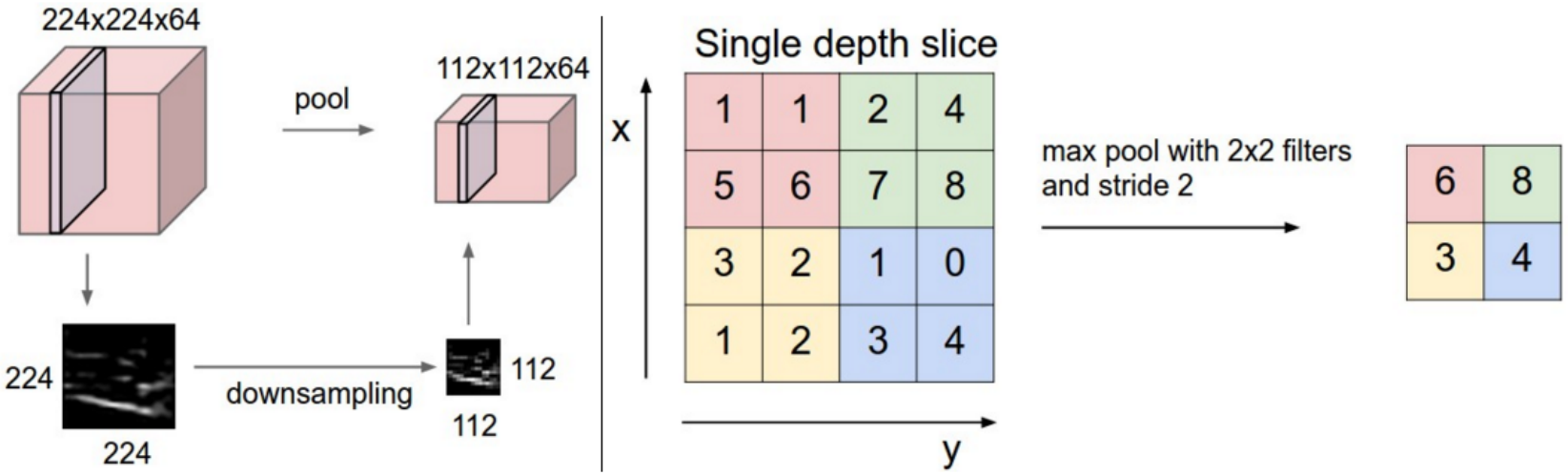
$$= f\left(w^{(l+1)} \cdot \mathbf{down}(X^l) + b^{(l+1)}\right), \tag{64}$$

Two choices of $\mathbf{down}(\cdot)$: Maximum Pooling and Average Pooling.

$$pool_{max}(R_k) = \max_{i \in R_k} a_i \tag{65}$$

$$pool_{avg}(R_k) = \frac{1}{|R_k|} \sum_{i \in R_k} a_i. \tag{66}$$

Pooling Layer



[15] Figure from: http://cs231n.github.io/convolutional-networks/

Large Scale Visual Recognition Challenge

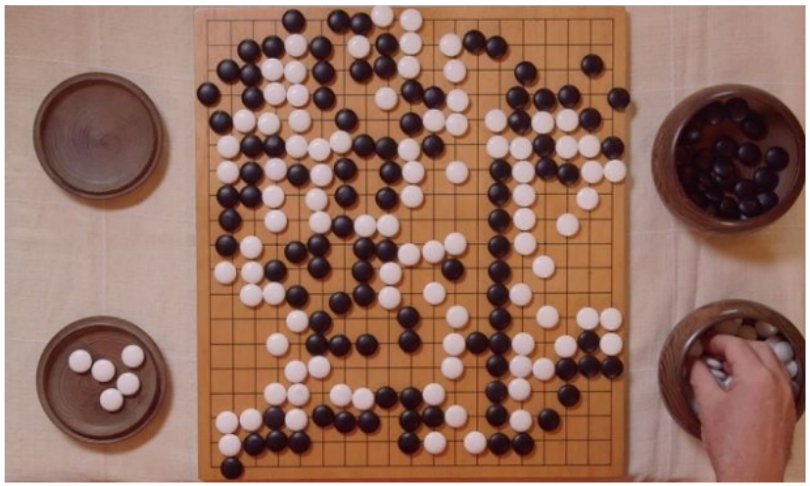
2010-2015

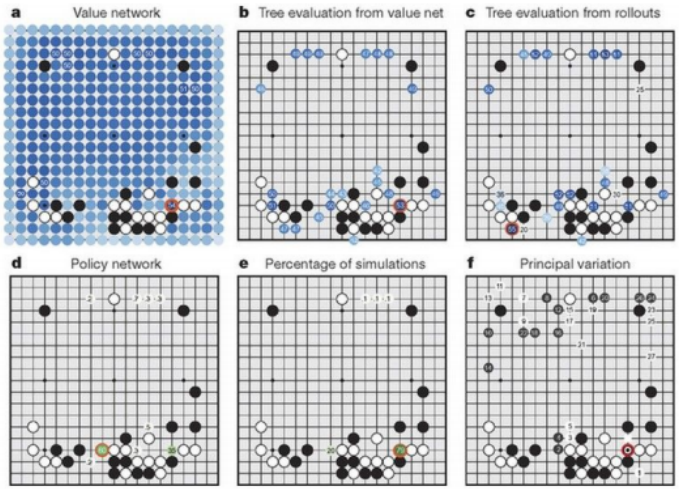
2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

DeepMind's AlphaGo





[15] http://cs231n.stanford.edu/

DeepMind's AlphaGo

The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

policy network:
[19x19x48] Input
CONV1: 192 5x5 filters , stride 1, pad 2 => [19x19x192]
CONV2..12: 192 3x3 filters, stride 1, pad 1 => [19x19x192]
CONV: 1 1x1 filter, stride 1, pad 0 => [19x19] *(probability map of promising moves)*

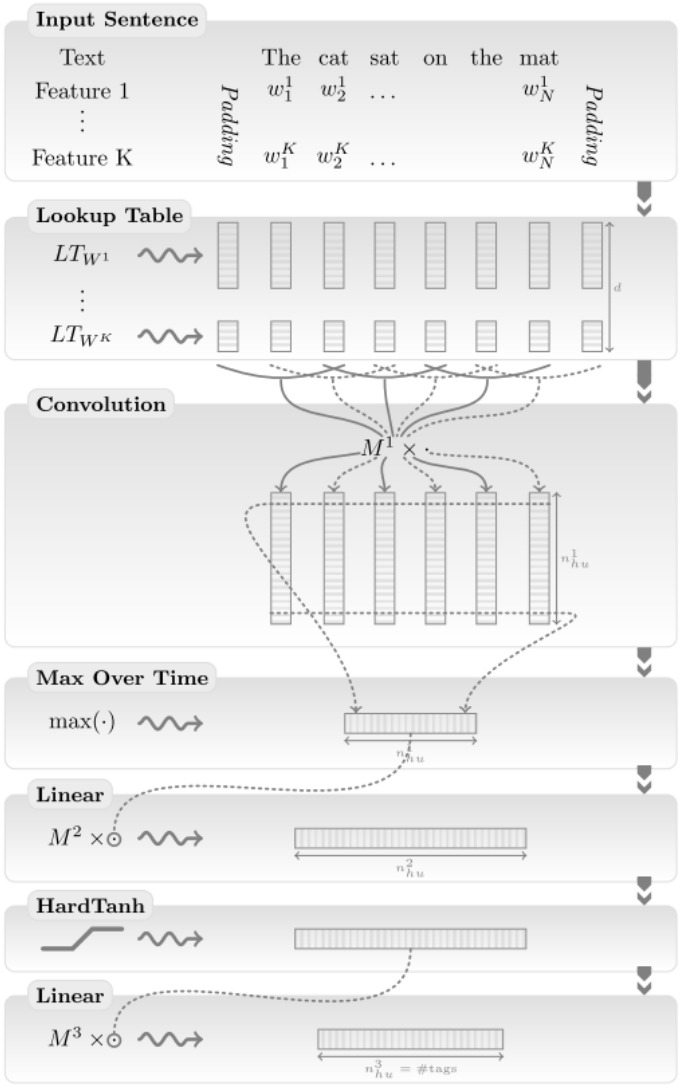
[15] http://cs231n.stanford.edu/

CNN for Sentence Modeling

Input: A sentence of length n,
After Lookup layer, $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathcal{R}^{d \times n}$

- variable-length input
- convolution
- pooling

CNN for Sentence Modeling[16]



Key steps

- convolution

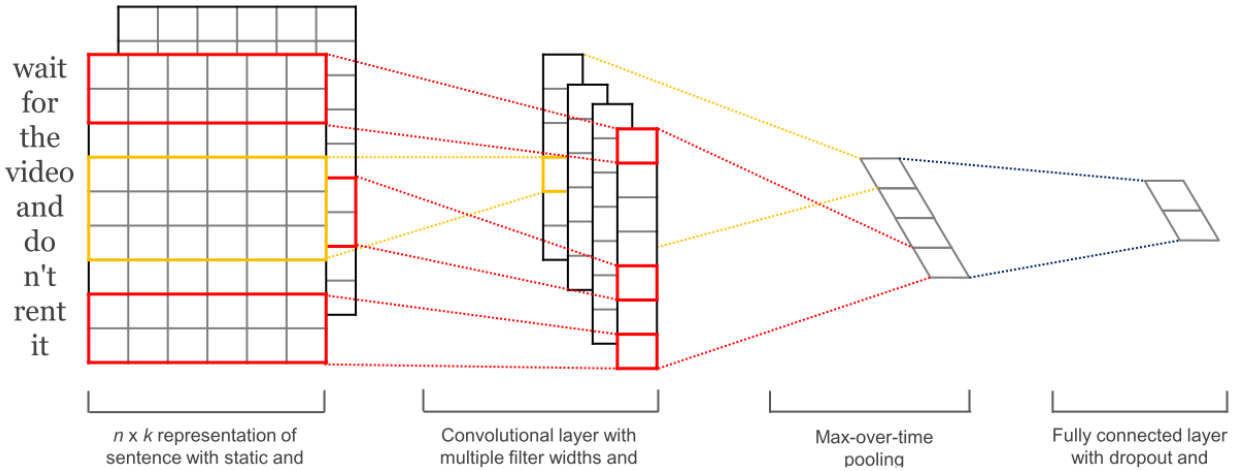
  $\mathbf{z}_{t:t+m-1} = \mathbf{x}_t \oplus \mathbf{x}_{t+1} \oplus \mathbf{x}_{t+m-1} \in \mathcal{R}^{dm}$
- matrix-vector operation

  $\mathbf{x}_t^l = f(W^l \mathbf{z}_{t:t+m-1} + \mathbf{b}^l)$
- Pooling (max over time)

  $\mathbf{x}_i^l = \max_t \mathbf{x}_{i,t}^{l-1}$

[16] Collobert et al., "Natural language processing (almost) from scratch".

CNN for Sentence Modeling[17]



Key steps

- convolution

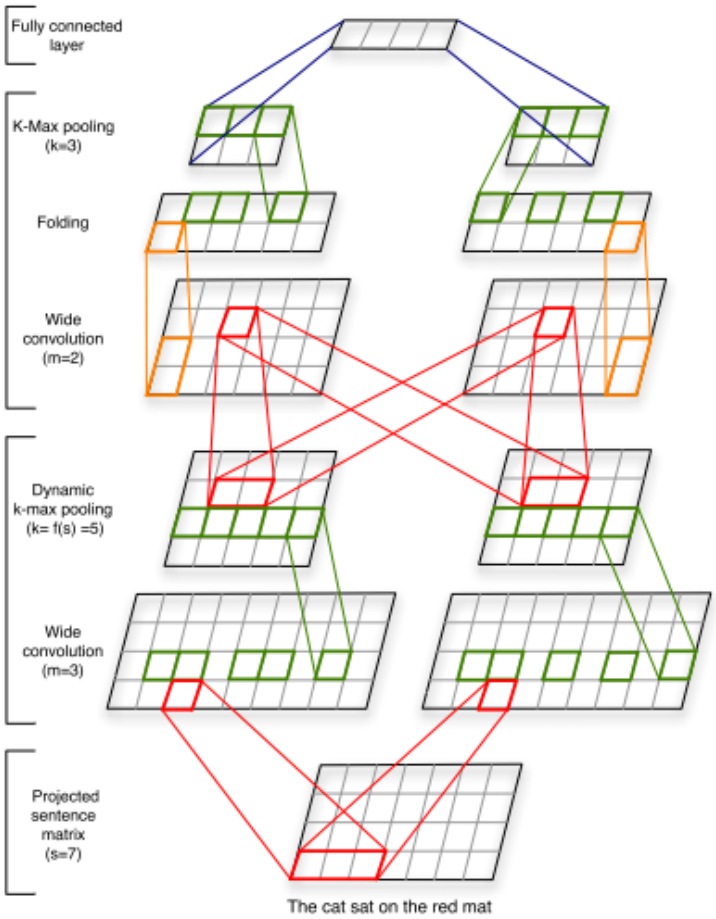
  $\mathbf{z}_{t:t+m-1} = \mathbf{x}_t \oplus \mathbf{x}_{t+1} \oplus \mathbf{x}_{t+m-1} \in \mathcal{R}^{dm}$
- vector-vector operation

  $x_t^l = f(\mathbf{w}^l \mathbf{z}_{t:t+m-1} + b^l)$
- multiple filters / multiple channels
- pooling (max over time)

  $\mathbf{x}_i^l = \max_t \mathbf{x}_{i,t}^{l-1}$

[17] Y. Kim. "Convolutional neural networks for sentence classification". In: arXiv preprint arXiv:1408.5882 (2014).

Dynamic CNN for Sentence Modeling[18]

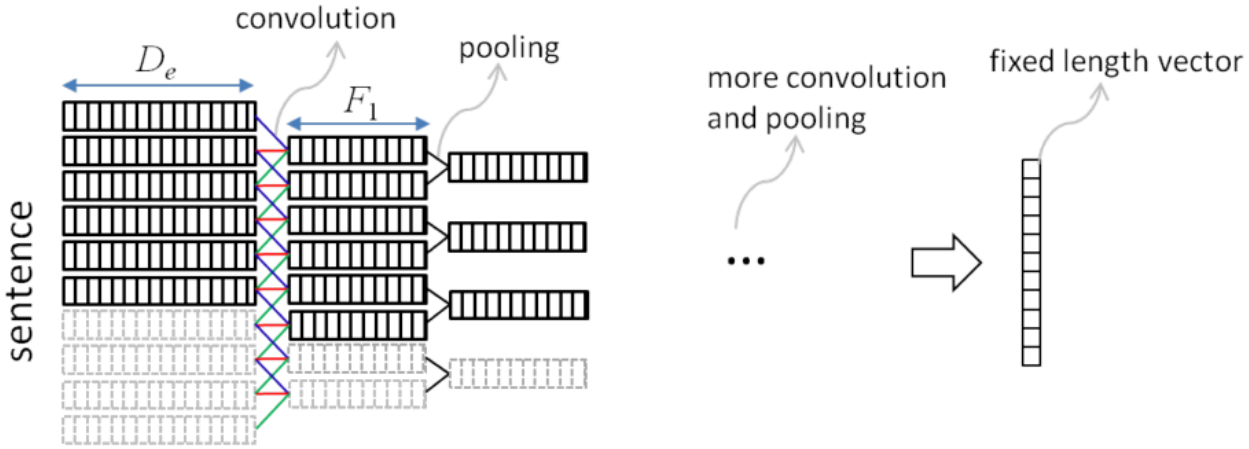


Key steps

- one-dimensional convolution
  $\mathbf{x}^l_{i,t} = f(\mathbf{w}^l_i \mathbf{x}_{i,t:t+m-1} + b^l_i)$
- k-max pooling (max over time)
  $k^l = \max(k^{top}, \frac{|L-l|}{L} n)$
- (optional) folding: sums every two rows in a feature map component-wise.
- multiple filters / multiple channels

[18] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A Convolutional Neural Network for Modelling Sentences". In: Proceedings of ACL. 2014.

CNN for Sentence Modeling[19]



Key steps

- convolution

  $\mathbf{z}_{t:t+m-1} = \mathbf{x}_t \oplus \mathbf{x}_{t+1} \oplus \mathbf{x}_{t+m-1} \in \mathcal{R}^{dm}$
- matrix-vector operation

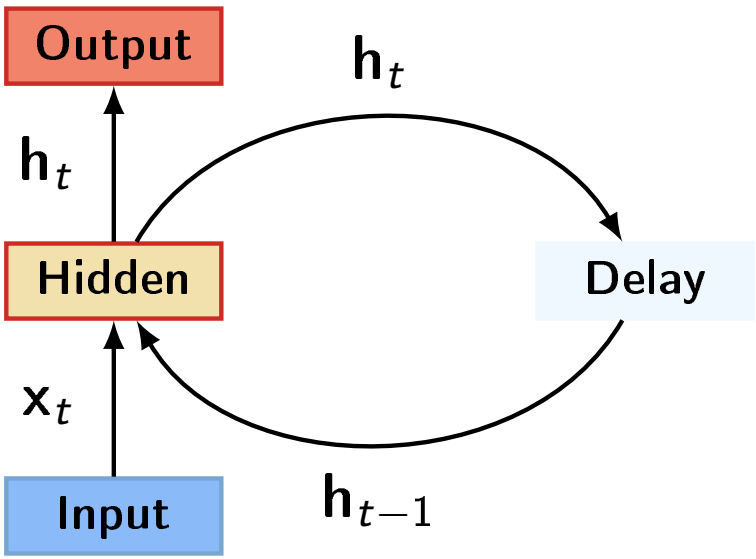
  $\mathbf{x}_t^l = f(\mathbf{W}^l \mathbf{z}_{t:t+m-1}^{l-1} + b^l)$
- binary max pooling (over time)

  $\mathbf{x}_{i,t}^l = \max(\mathbf{x}_{i,2t-1}^{l-1}, \mathbf{x}_{i,2t}^{l-1})$

[19] B. Hu et al. "Convolutional neural network architectures for matching natural language sentences". In: Advances in Neural Information Processing Systems. 2014.

Table of Contents

Recurrent Neural Network (RNN)



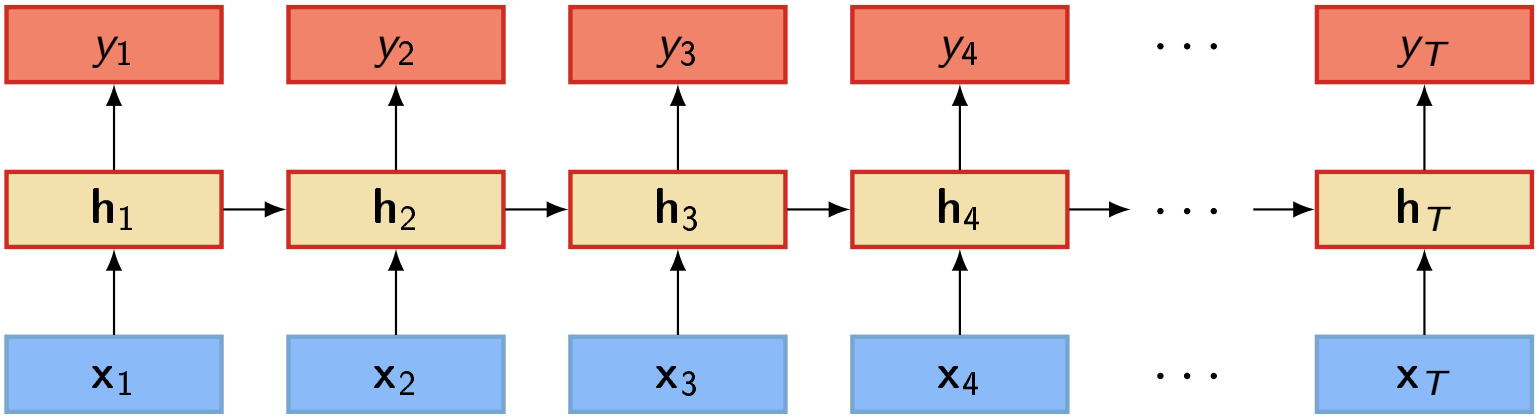
The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time. More formally, given a sequence $\mathbf{x}^{(1:n)} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}, \ldots, \mathbf{x}^{(n)})$, the RNN updates its recurrent hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases} \quad (67)$$

Simple Recurrent Network[20]

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}), \tag{68}$$

where f is non-linear function.



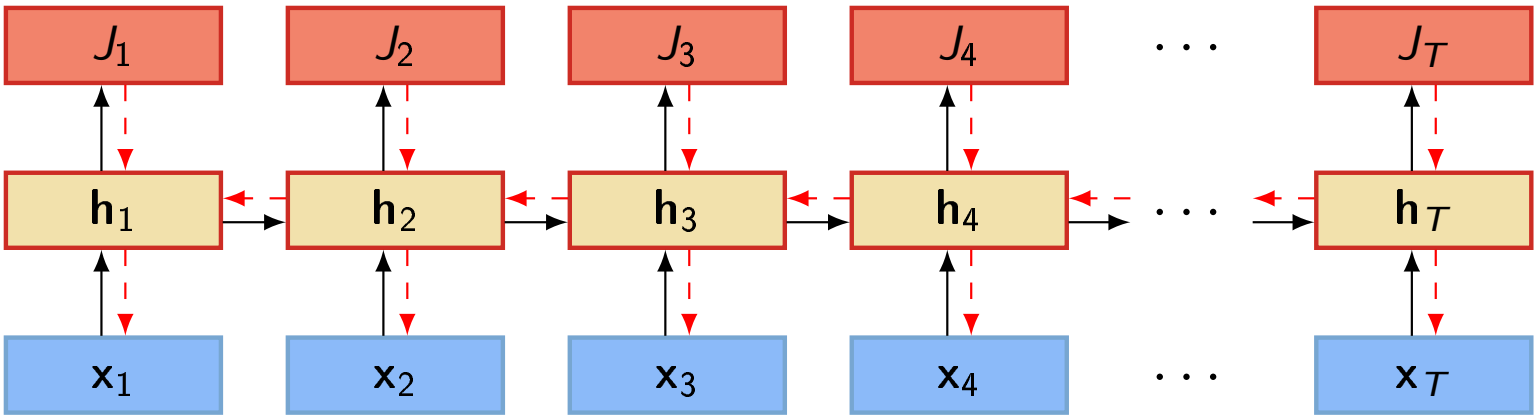
[20] J. L. Elman. "Finding structure in time". In: Cognitive science 2 (1990).

Backpropagation Through Time, BPTT

Loss at time t is $\mathcal{J}_t$, the whole loss is $\mathcal{J} = \sum_{t=1}^{T} \mathcal{J}_t$.
The gradient of $\mathcal{J}$ is

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^{T} \frac{\partial \mathcal{J}_t}{\partial U} \tag{69}$$

$$= \sum_{t=1}^{T} \frac{\partial \mathbf{h}_t}{\partial U} \frac{\partial \mathcal{J}_t}{\partial \mathbf{h}_t}, \tag{70}$$



Gradient of RNN

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial \mathbf{h}_k}{\partial U} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathcal{J}_t}{\partial \mathbf{y}_t}, \tag{71}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \tag{72}$$

$$= \prod_{i=k+1}^{t} U^T \, \mathbf{diag}[f'(h_{i-1})]. \tag{73}$$

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial \mathbf{h}_k}{\partial U} \left(\prod_{i=k+1}^{t} U^T \, \mathbf{diag}(f'(h_{i-1})) \right) \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathcal{J}_t}{\partial \mathbf{y}_t}. \tag{74}$$

Long-Term Dependencies

Define $\gamma = \|U^T \mathbf{diag}(f'(h_{i-1}))\|$,

- Exploding Gradient Problem: When $\gamma > 1$ and $t - k \to \infty$, $\gamma^{t-k} \to \infty$.
- Vanishing Gradient Problem: When $\gamma < 1$ and $t - k \to \infty$, $\gamma^{t-k} \to 0$.

There are various ways to solve **Long-Term Dependency** problem.

Long Short Term Memory Neural Network (LSTM)[21]

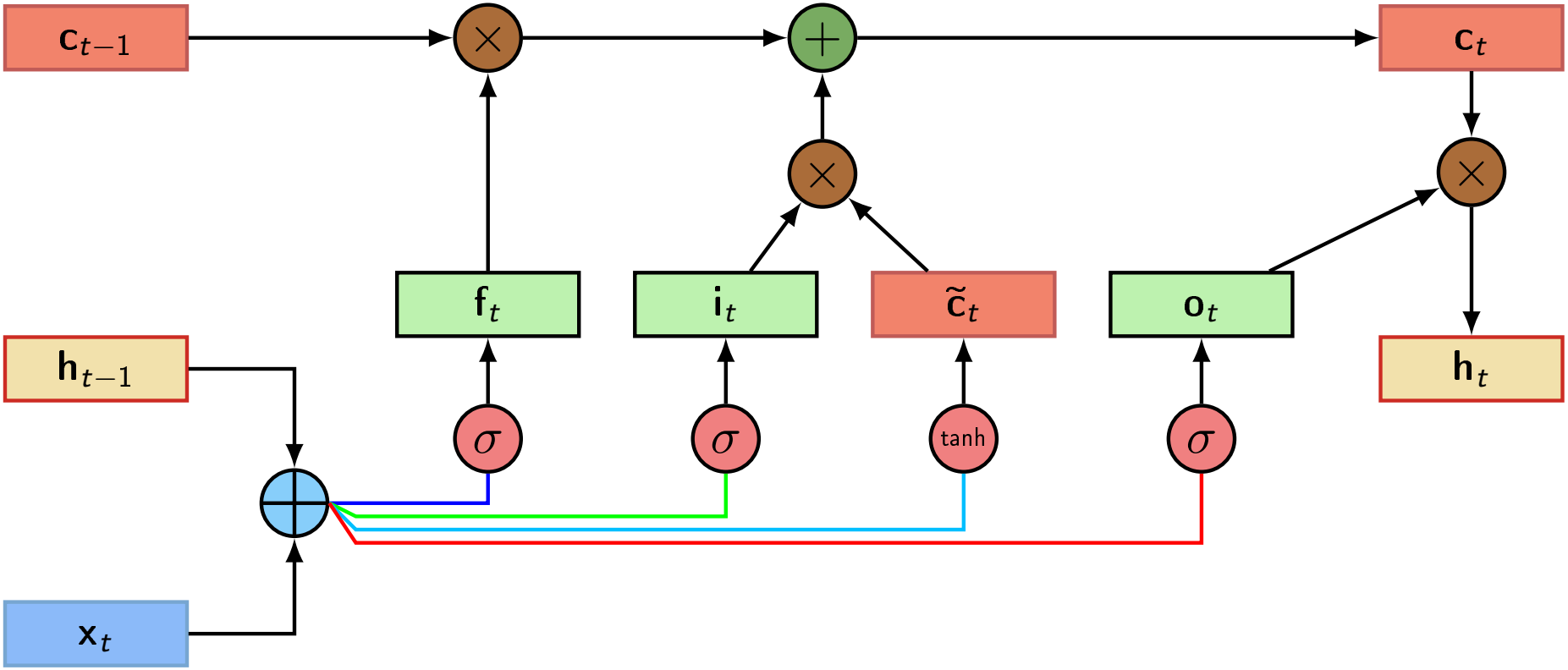
The core of the LSTM model is a memory cell $\mathbf{c}$ encoding memory at every time step of what inputs have been observed up to this step.
The behavior of the cell is controlled by three "gates":

- input gate $\mathbf{i}$
- output gate $\mathbf{o}$
- forget gate $\mathbf{f}$

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}), \tag{75}$$
$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1}), \tag{76}$$
$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t), \tag{77}$$
$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1}), \tag{78}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{79}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{80}$$

[21] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: Neural computation 8 (1997).

LSTM Architecture



Gated Recurrent Unit, GRU[22]

Two gates: update gate $\mathbf{z}$ and reset gate $\mathbf{r}$.

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \tag{81}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \tag{82}$$

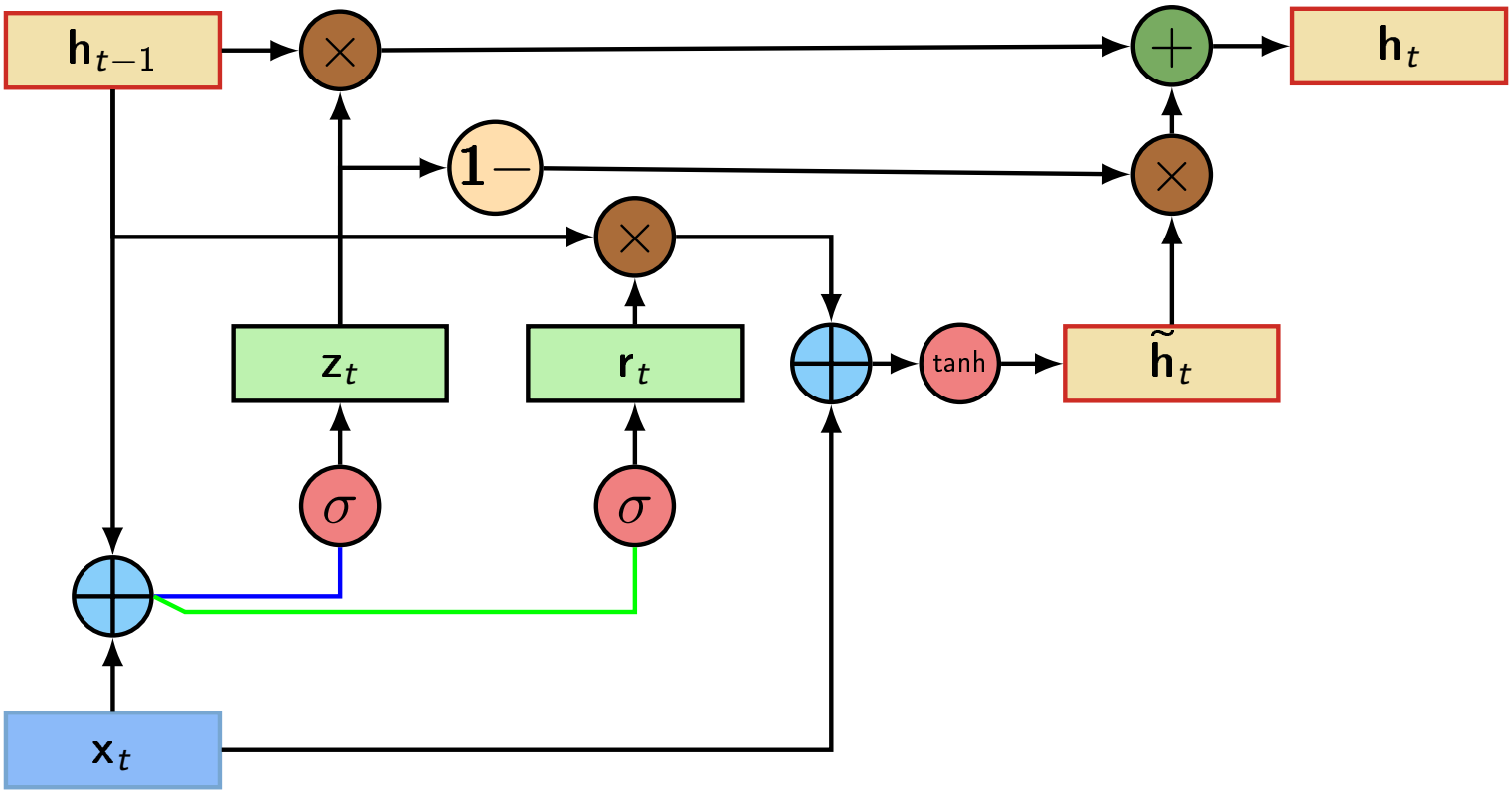
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})), \tag{83}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t, \tag{84}$$

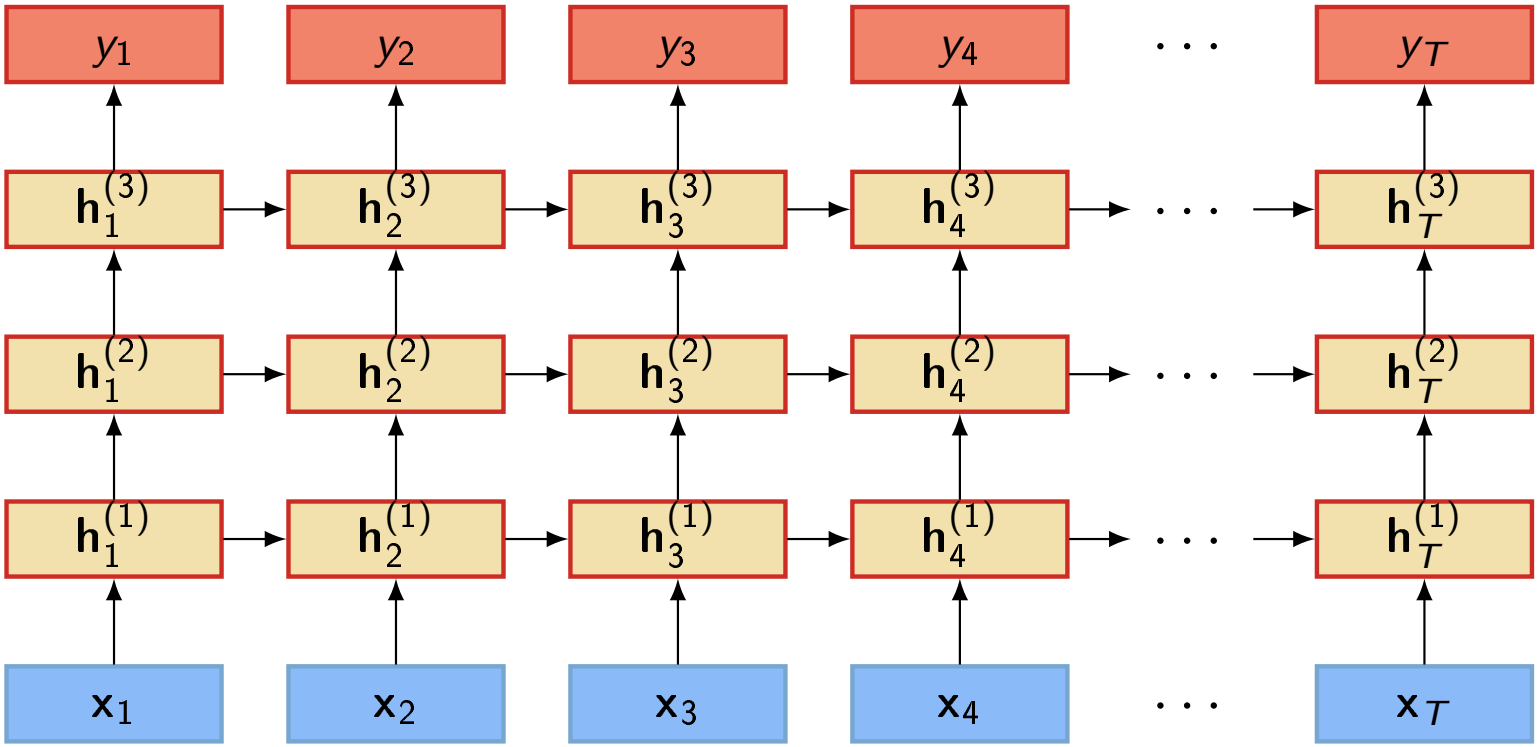
$$\tag{85}$$

[22] K. Cho et al. "Learning phrase representations using rnn encoder-decoder for statistical machine translation". In: arXiv preprint arXiv:1406.1078 (2014); J. Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: arXiv preprint arXiv:1412.3555 (2014).

GRU Architecture



Stacked RNN

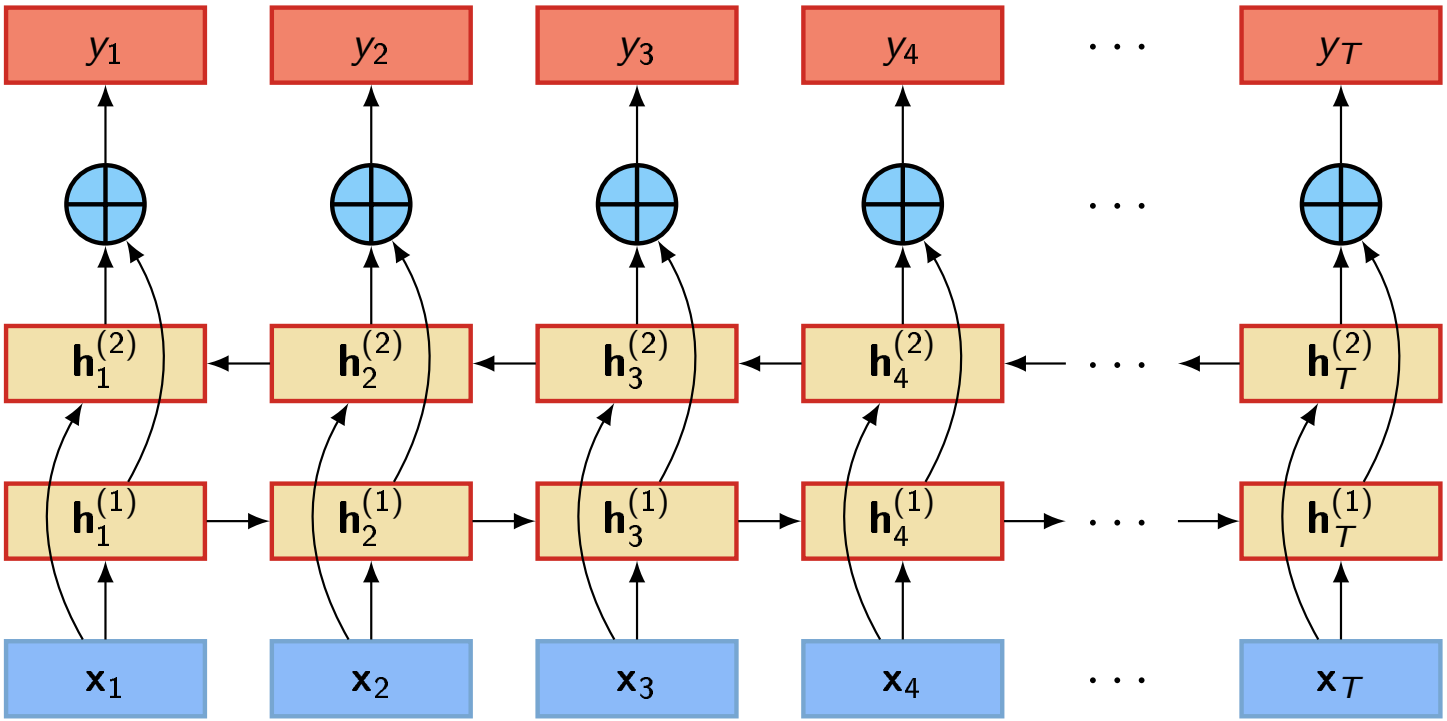

y_1 y_2 y_3 y_4 $\cdots$ y_T
$\mathbf{h}_1^{(3)}$ $\mathbf{h}_2^{(3)}$ $\mathbf{h}_3^{(3)}$ $\mathbf{h}_4^{(3)}$ $\cdots$ $\mathbf{h}_T^{(3)}$
$\mathbf{h}_1^{(2)}$ $\mathbf{h}_2^{(2)}$ $\mathbf{h}_3^{(2)}$ $\mathbf{h}_4^{(2)}$ $\cdots$ $\mathbf{h}_T^{(2)}$
$\mathbf{h}_1^{(1)}$ $\mathbf{h}_2^{(1)}$ $\mathbf{h}_3^{(1)}$ $\mathbf{h}_4^{(1)}$ $\cdots$ $\mathbf{h}_T^{(1)}$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\cdots$ $\mathbf{x}_T$

Bidirectional RNN

$$\mathbf{h}_t^{(1)} = f(\mathbf{U}^{(1)}\mathbf{h}_{t-1}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}_t + \mathbf{b}^{(1)}), \tag{86}$$

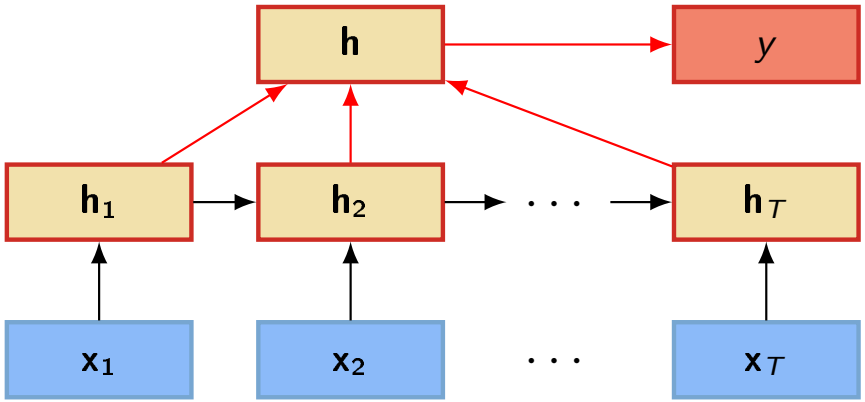
$$\mathbf{h}_t^{(2)} = f(\mathbf{U}^{(2)}\mathbf{h}_{t+1}^{(2)} + \mathbf{W}^{(2)}\mathbf{x}_t + \mathbf{b}^{(2)}), \tag{87}$$

$$\mathbf{h}_t = \mathbf{h}_t^{(1)} \oplus \mathbf{h}_t^{(2)}. \tag{88}$$

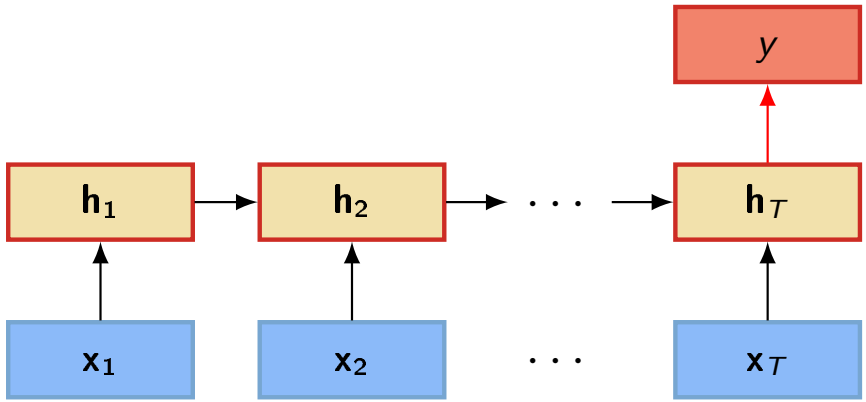


Application of RNN: Sequence to Category

- Text Classification
- Sentiment Classification



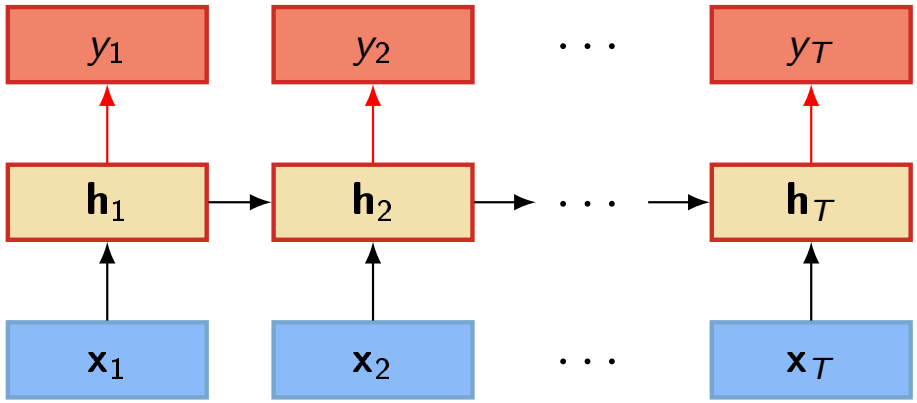
(c) Mean



(d) Last

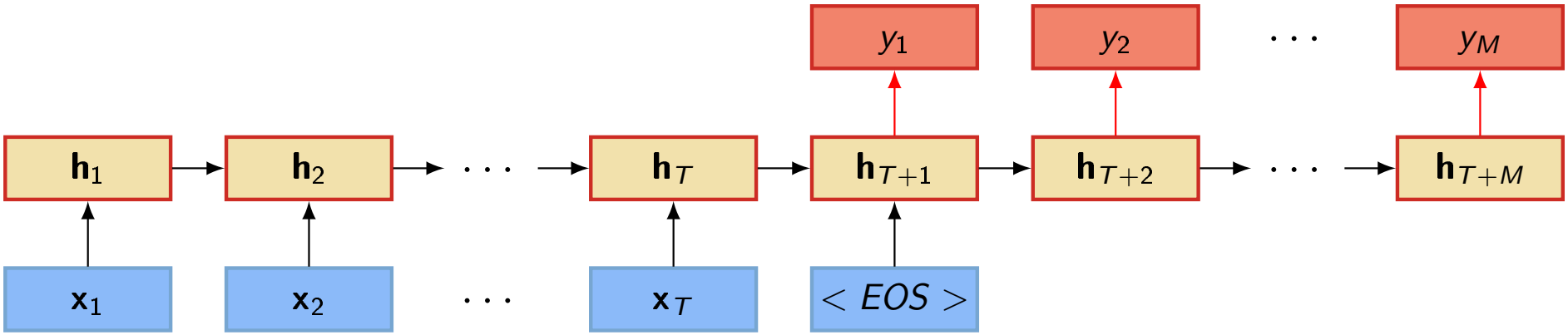
Application of RNN: Synchronous Sequence to Sequence

- Sequence Labeling, such as Chinese word segmentation, POS tagging

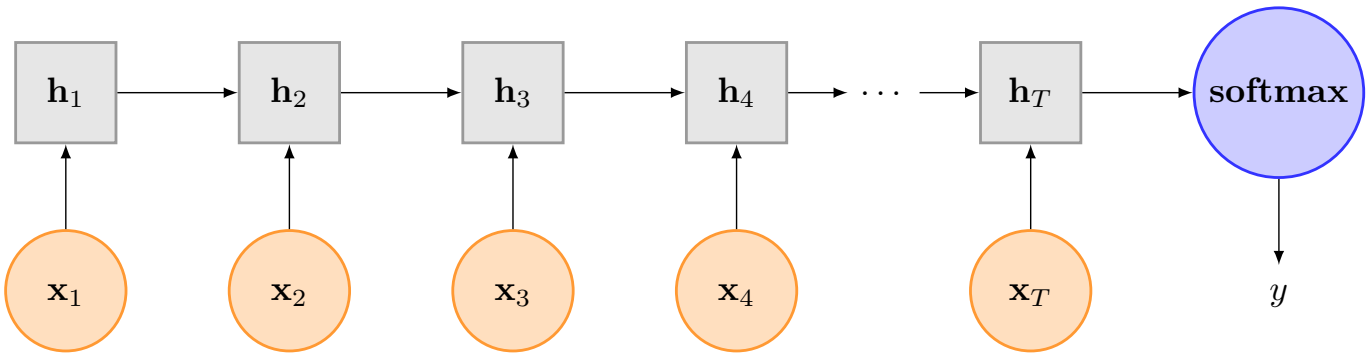


Application of RNN: Asynchronous Sequence to Sequence

- Machine Translation

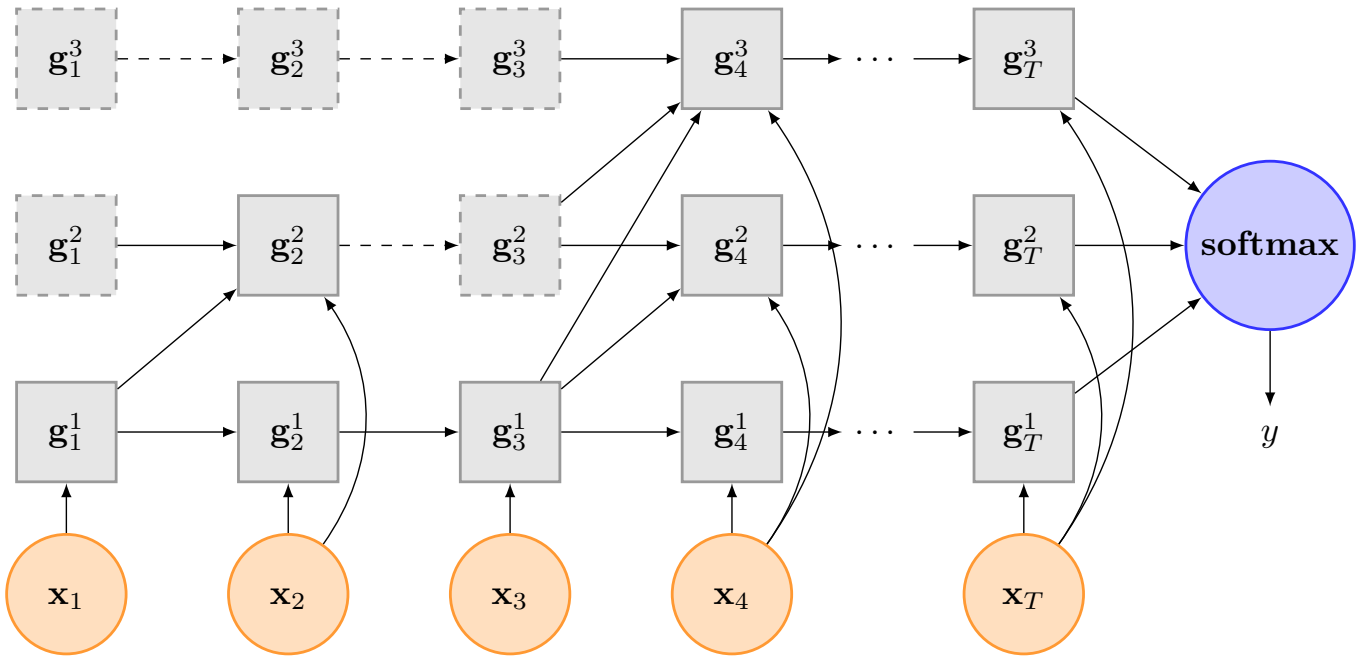


Unfolded LSTM for Text Classification



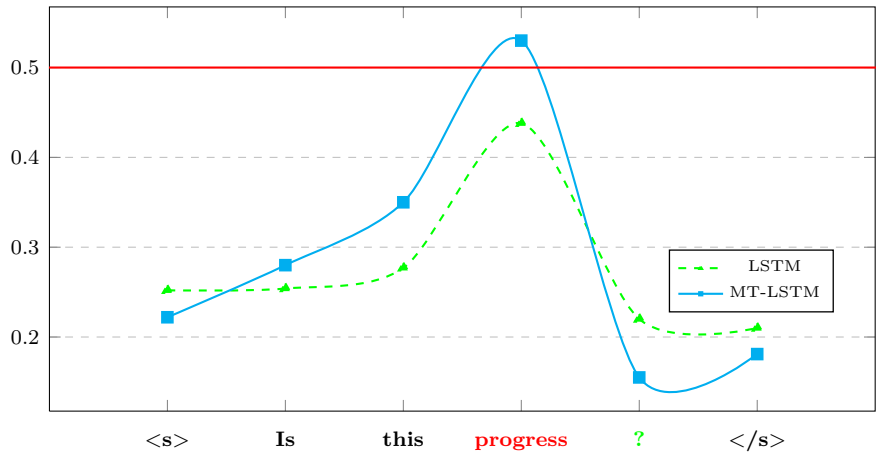
Drawback: long-term dependencies need to be transmitted one-by-one along the sequence.

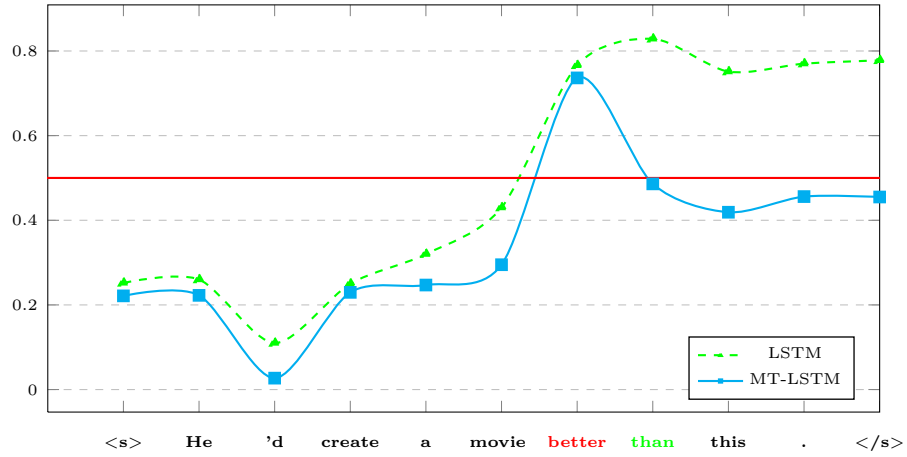
Unfolded Multi-Timescale LSTM[23]

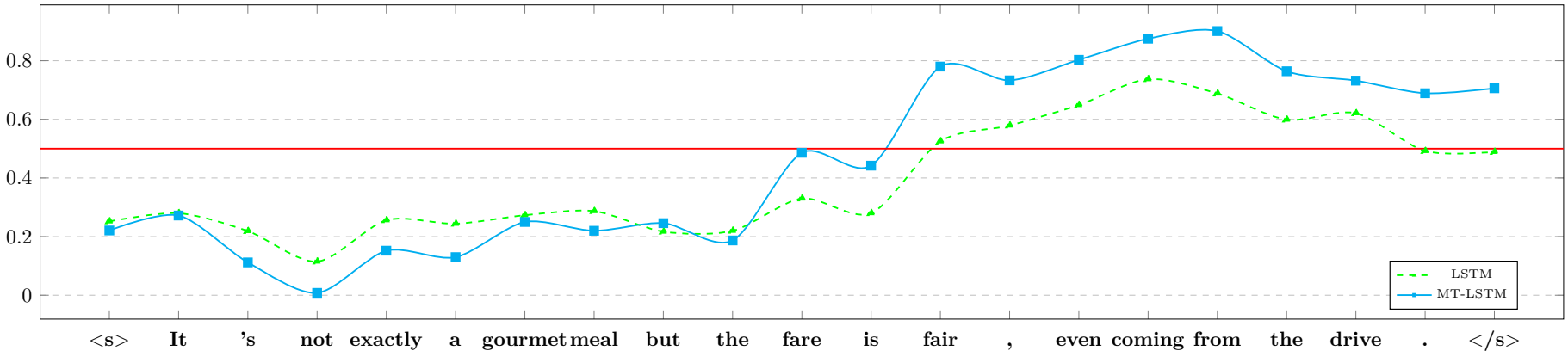


[23] P. Liu et al. "Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents". In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2015.

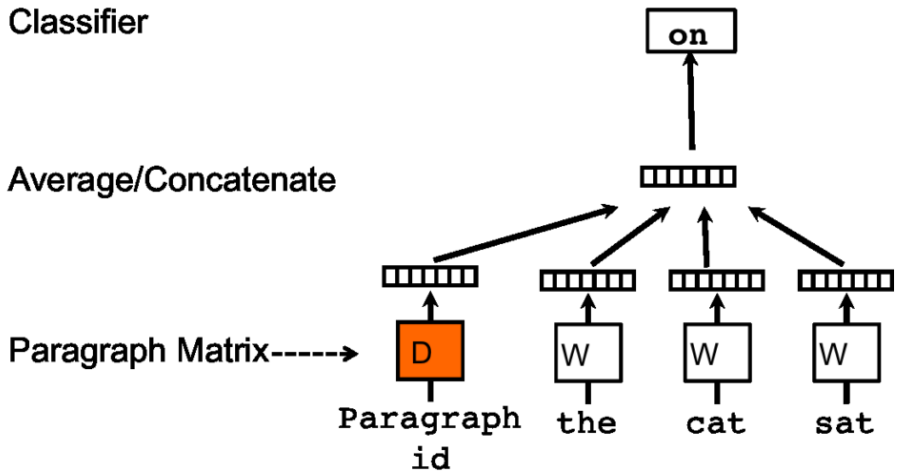
LSTM for Sentiment Analysis







Paragraph Vector[24]



[24] Q. V. Le and T. Mikolov. "Distributed representations of sentences and documents". In: arXiv preprint arXiv:1405.4053 (2014).

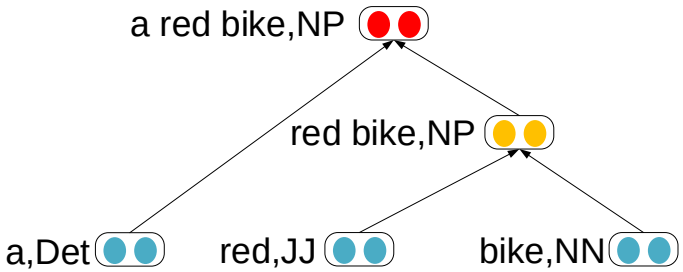
Memory Mechanism

What differences among the various models from memory view?

	Short-term	Long short-term	Global	External
SRN	Yes	No	No	No
LSTM/GRU	Yes	Yes	Maybe	No
PV	Yes	Yes	Yes	No
NTM/DMN	Yes	Yes	Maybe	Yes

Table of Contents

Recursive Neural Network (RecNN)[25]



Topological models compose the sentence representation following a given topological structure over the words.

Given a labeled binary parse tree, $((p_2 \rightarrow ap_1), (p_1 \rightarrow bc))$, the node representations are computed by

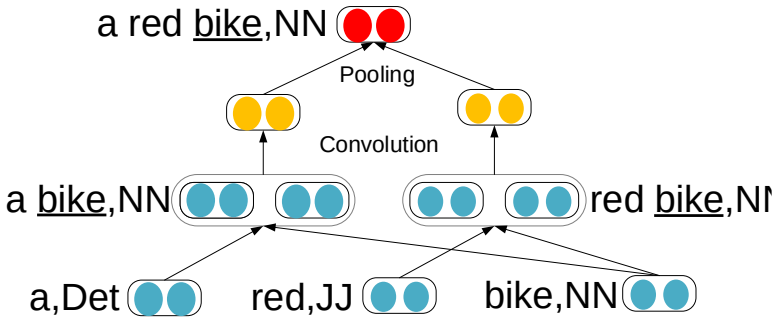
$$\mathbf{p}_1 = f(\mathbf{W} \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}),$$

$$\mathbf{p}_2 = f(\mathbf{W} \begin{bmatrix} \mathbf{a} \\ \mathbf{p}_1 \end{bmatrix}).$$

[25] R. Socher et al. "Parsing with compositional vector grammars". In: Proceedings of ACL. 2013.

Recursive Convolutional Neural Network[26]

Recursive neural network can only process the binary combination and is not suitable for dependency parsing.

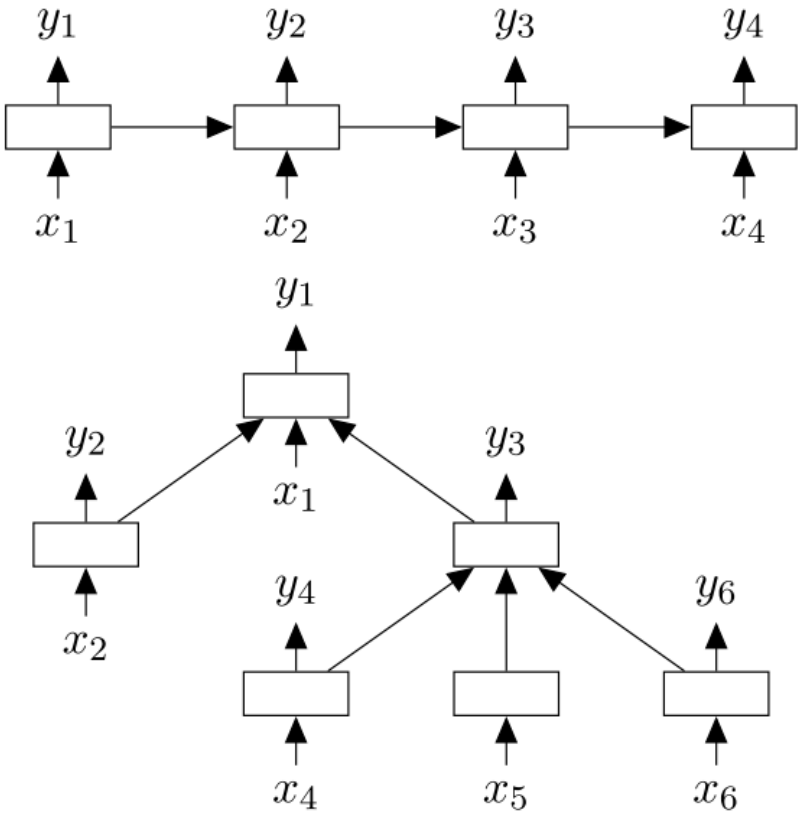


- introducing the convolution and pooling layers;
- modeling the complicated interactions of the head word and its children.

[26] C. Zhu et al. "A Re-Ranking Model For Dependency Parser With Recursive Convolutional Neural Network". In: Proceedings of Annual Meeting of the Association for Computational Linguistics. 2015.

Tree-Structured LSTMs[27]

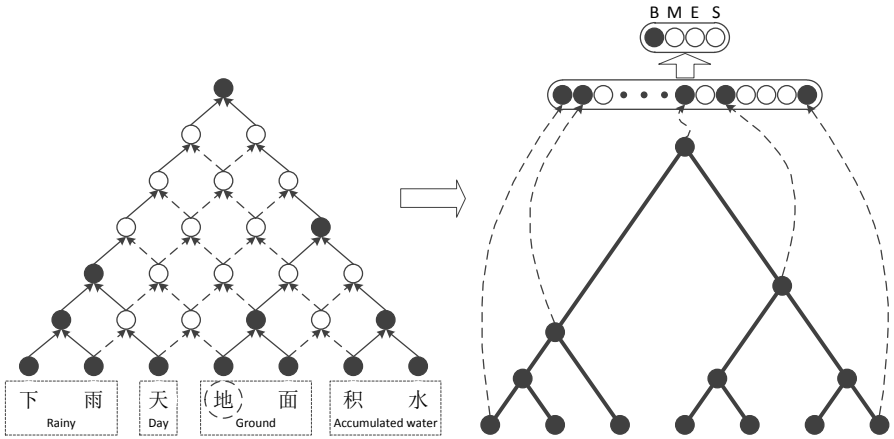
Natural language exhibits syntactic properties that would naturally combine words to phrases.



- Child-Sum Tree-LSTMs
- *N*-ary Tree-LSTMs

[27] K. S. Tai, R. Socher, and C. D. Manning. "Improved semantic representations from tree-structured long short-term memory networks". In: arXiv preprint arXiv:1503.00075 (2015).

Gated Recursive Neural Network[28]



- DAG based Recursive Neural Network
- Gating mechanism

An relative complicated solution

GRNN models the complicated combinations of the features, which selects and preserves the useful combinations via reset and update gates.

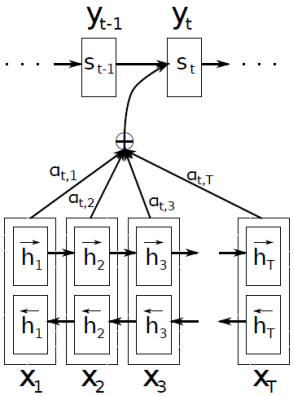
[28] X. Chen et al. "Gated Recursive Neural Network For Chinese Word Segmentation". In: Proceedings of Annual Meeting of the Association for Computational Linguistics. 2015; X. Chen et al. "Sentence Modeling with Gated Recursive Neural Network". In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2015.

Table of Contents

Attention


HERE COMES THE HURRICANE SEASON
JERRY BROWN
TOURIST SURGERY
TIME
Radical Chicks

Attention Model[29]



1. The context vector c_i is computed as a weighted sum of h_i:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

2. The weight α_{ij} is computed by

$$\alpha_{ij} = \text{softmax}(\mathbf{v}^T \tanh(\mathbf{W} s_{i-1} + \mathbf{U} h_j))$$

[29] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: ArXiv e-prints (Sept. 2014). arXiv: 1409.0473 [cs.CL].

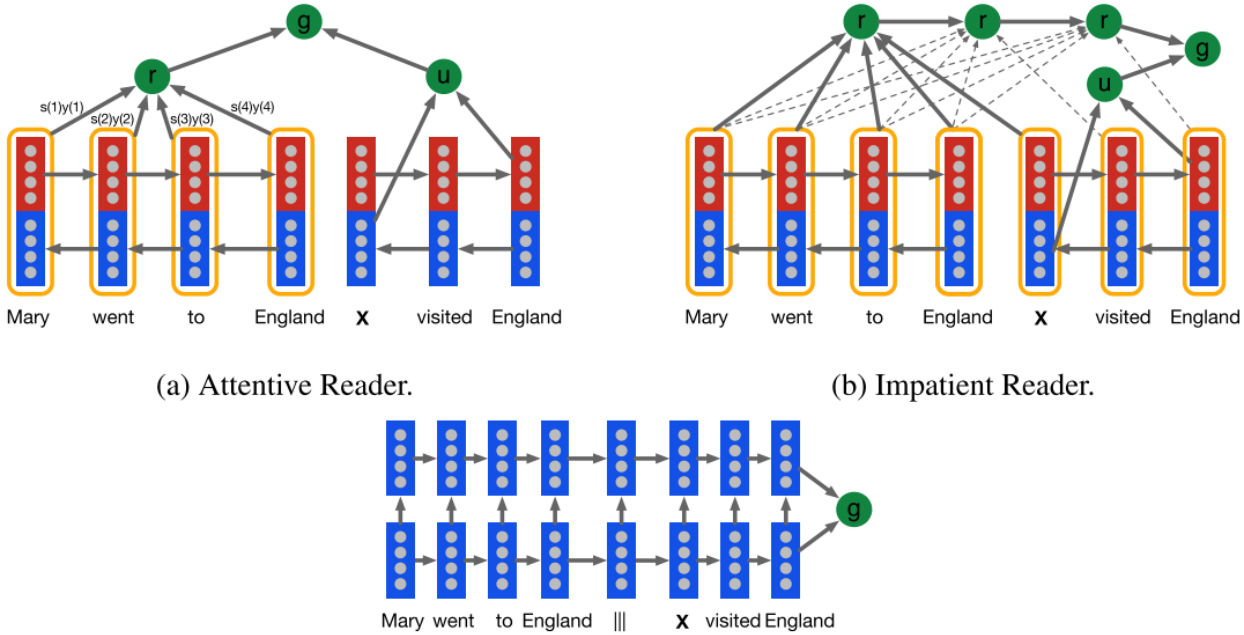
Table of Contents

Question Answering

I: Jane went to the hallway.
I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden
I: Everybody is happy.
Q: What's the sentiment?
A: positive
Q: What are the POS tags?
A: NN VBZ JJ .

I: Peter's sister is called Isabelle.
Q: What are the mentions?
A: [[Peter] 's sister] is called [Isabelle] .
I: Peter's sister is called [Isabelle].
Q: Is it coreferent with: [Peter's sister] is called Isabelle?
A: yes
Q: Is it coreferent with: [Peter] 's sister is called Isabelle?
A: no
I: The answer is far from obvious.
Q: In French?
A: La réponse est loin d'être évidente.

LSTM[30]



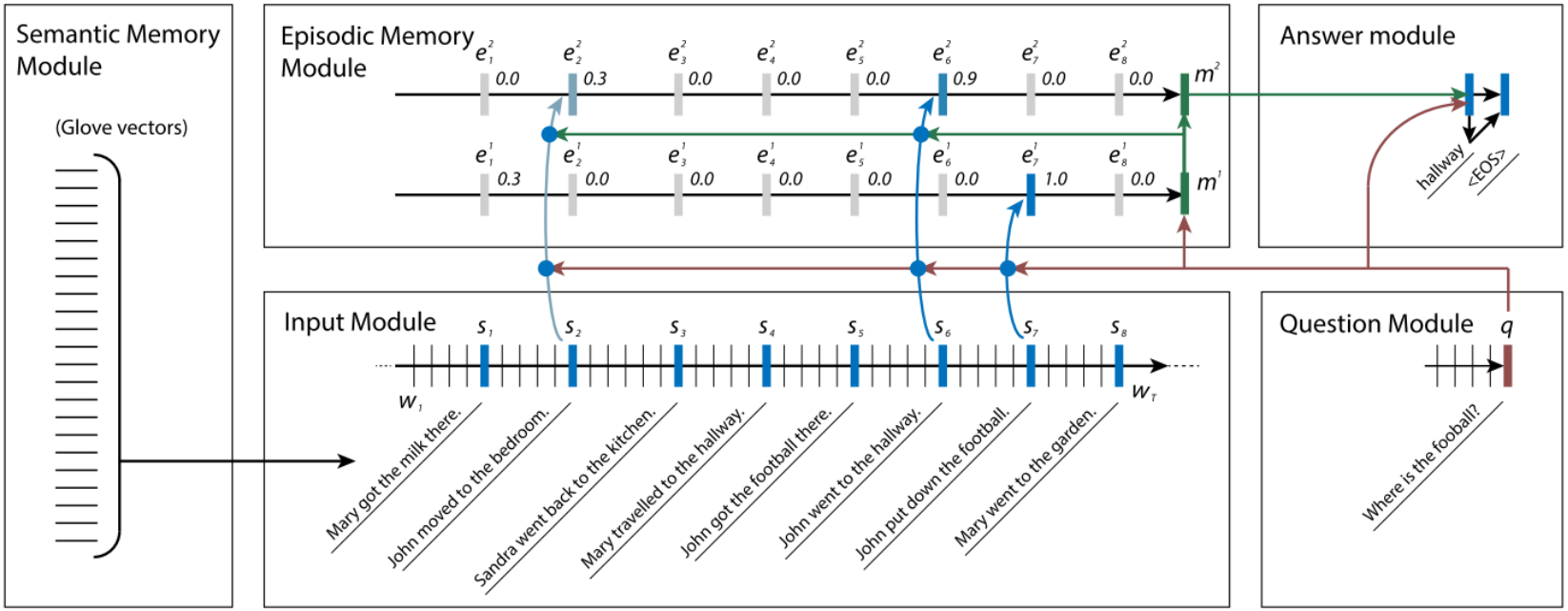
(a) Attentive Reader.

(b) Impatient Reader.

(c) A two layer Deep LSTM Reader with the question encoded following the document.

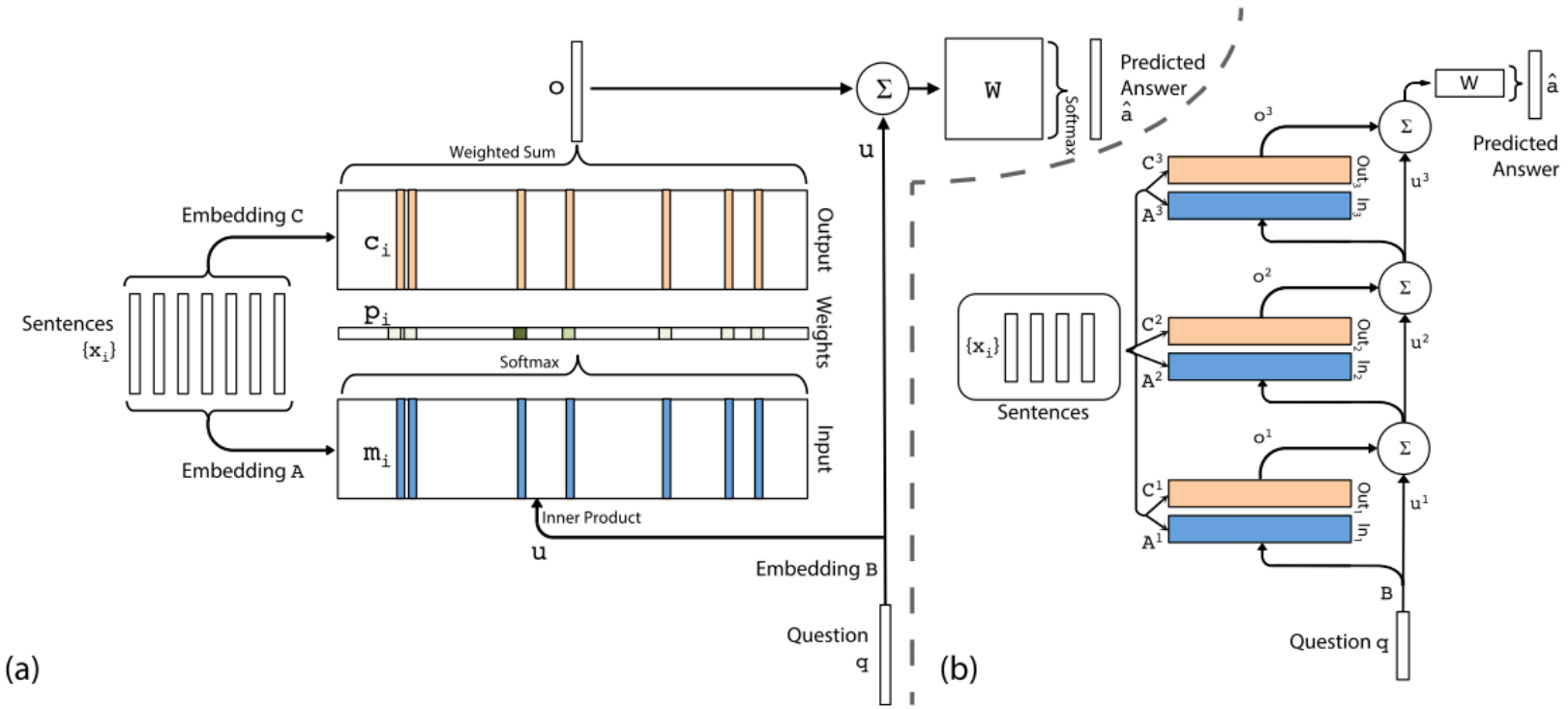
[30] K. M. Hermann et al. "Teaching machines to read and comprehend". In: Advances in Neural Information Processing Systems. 2015, pp. 1684–1692.

Dynamic Memory Networks[31]



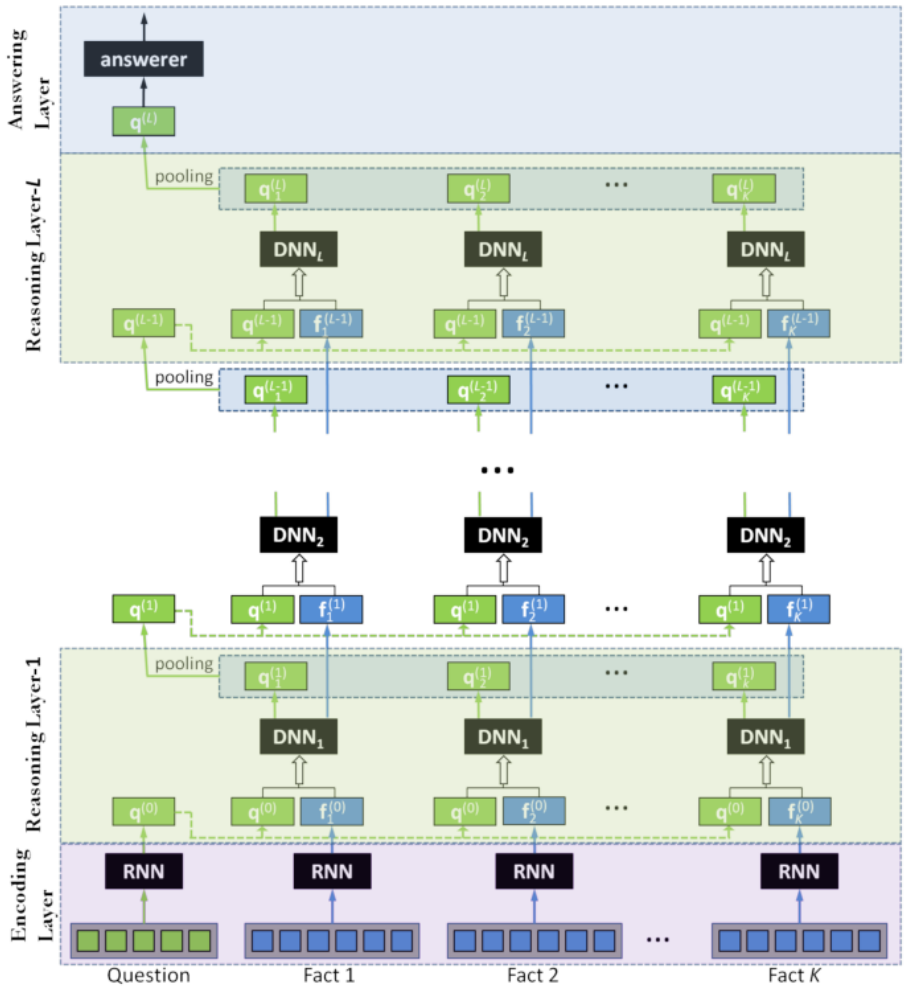
[31] A. Kumar et al. "Ask me anything: Dynamic memory networks for natural language processing". In: arXiv preprint arXiv:1506.07285 (2015).

Memory Networks[32]



[32] S. Sukhbaatar, J. Weston, R. Fergus, et al. "End-to-end memory networks". In: Advances in Neural Information Processing Systems. 2015, pp. 2431–2439.

Neural Reasoner[33]

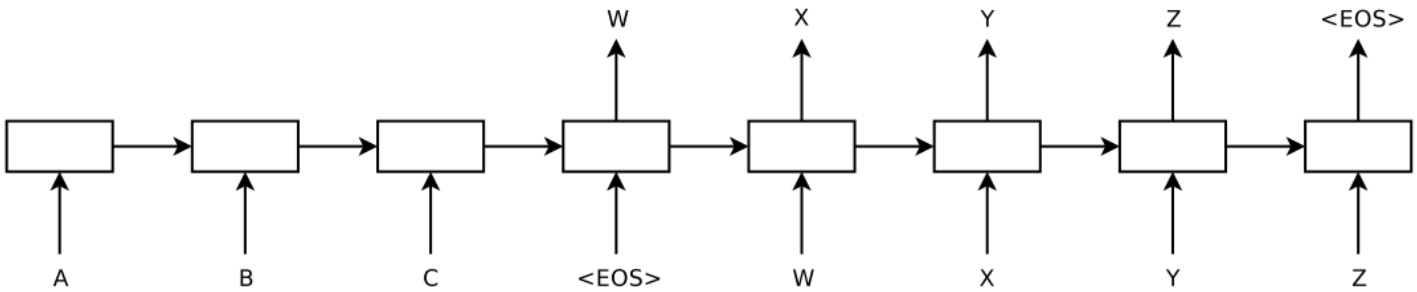


[33] B. Peng et al. "Towards Neural Network-based Reasoning". In: arXiv preprint arXiv:1508.05508 (2015).

Table of Contents

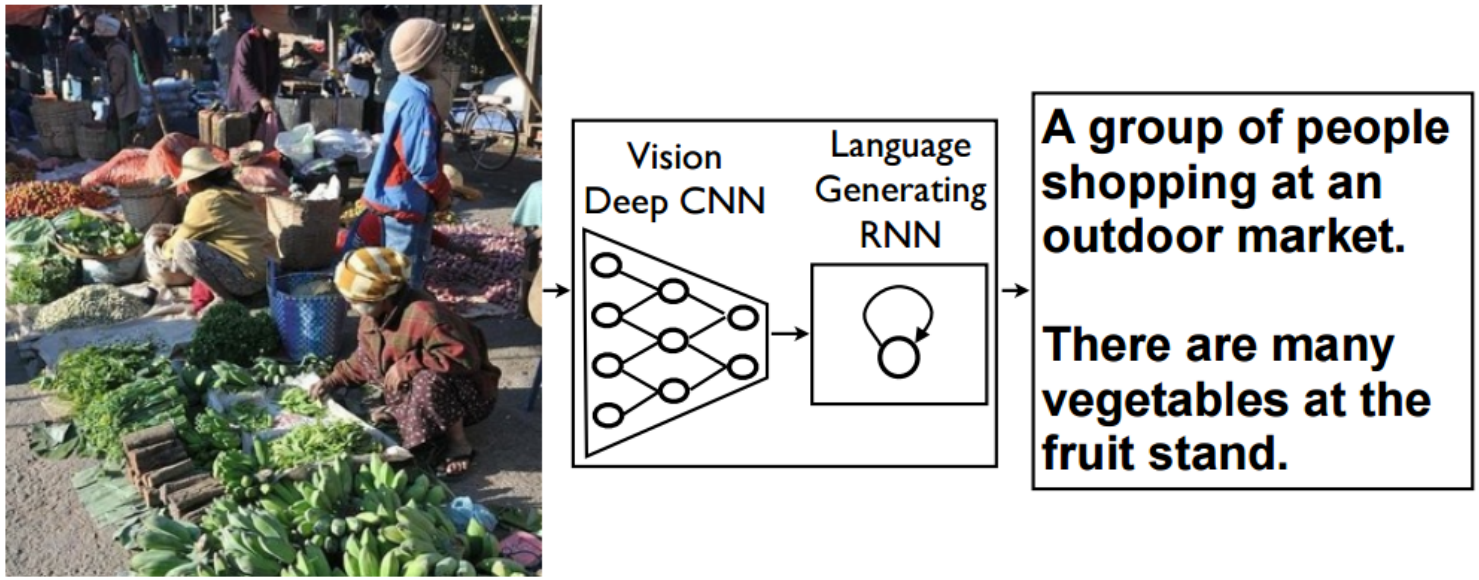
Sequence to Sequence Machine Translation[34]

Neural machine translation is a recently proposed framework for machine translation based purely on neural networks.



[34] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". In: Advances in Neural Information Processing Systems. 2014, pp. 3104–3112.

Image Caption[35][36][37]



[35] A. Karpathy and L. Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, pp. 3128–3137.

[36] O. Vinyals et al. "Show and tell: A neural image caption generator". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, pp. 3156–3164.

[37] K. Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: arXiv preprint arXiv:1502.03044 (2015).

Image Caption

Figure 5. A selection of evaluation results, grouped by human rating.

Table of Contents

Text Matching

Among many natural language processing (NLP) tasks, such as text classification, question answering and machine translation, a common problem is modelling the relevance/similarity of a pair of texts, which is also called **text semantic matching**.

Three types of interaction models:

- Weak interaction Models
- Semi-interaction Models
- Strong Interaction Models

Weak interaction Models

Some early works focus on sentence level interactions, such as ARC-I[38], CNTN[39] and so on. These models first encode two sequences into continuous dense vectors by separated neural models, and then compute the matching score based on sentence encoding.

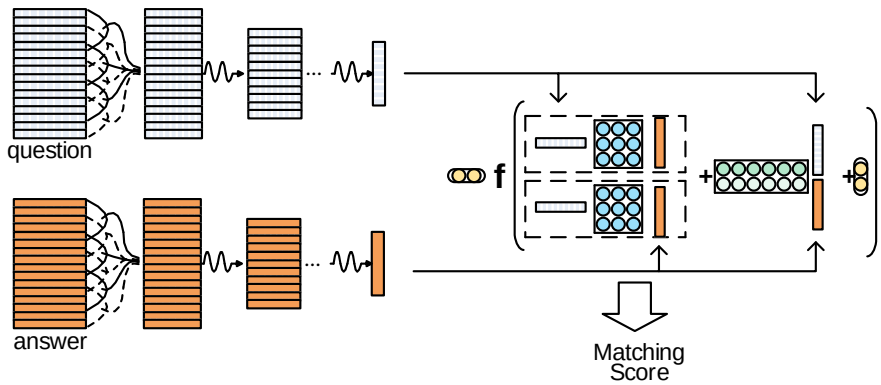


Figure: Convolutional Neural Tensor Network

[38] Hu et al., "Convolutional neural network architectures for matching natural language sentences".

[39] X. Qiu and X. Huang. "Convolutional Neural Tensor Network Architecture for Community-based Question Answering". In: Proceedings of International Joint Conference on Artificial Intelligence. 2015.

Semi-interaction Models

Another kind of models use soft attention mechanism to obtain the representation of one sentence by depending on representation of another sentence, such as ABCNN[40], Attention LSTM[41]. These models can alleviate the weak interaction problem to some extent.

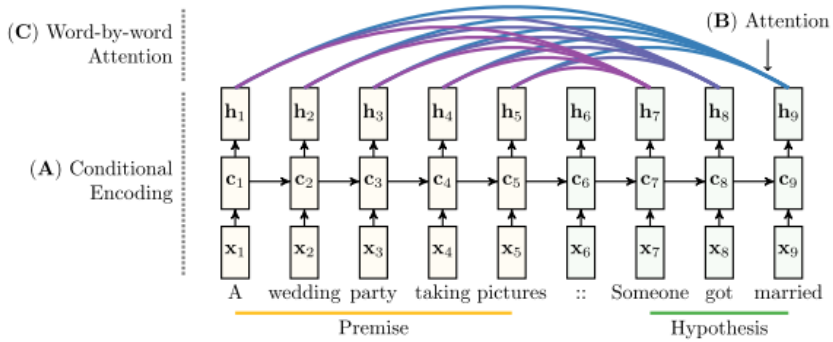


Figure: Attention LSTM[42]

[40] W. Yin et al. "ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs". In: arXiv preprint arXiv:1512.05193 (2015).

[41] Hermann et al., "Teaching machines to read and comprehend".

[42] T. Rockt schel et al. "Reasoning about Entailment with Neural Attention". In: arXiv preprint arXiv:1509.06664 (2015).

Strong Interaction Models

The models build the interaction at different granularity (word, phrase and sentence level), such as ARC-II[43], MV-LSTM[44], coupled-LSTMs[45]. The final matching score depends on these different levels of interactions.

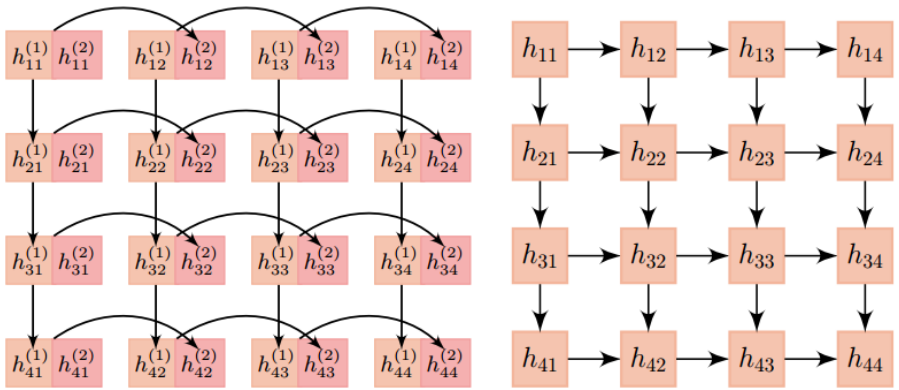


Figure: coupled-LSTMs

[43] Hu et al., "Convolutional neural network architectures for matching natural language sentences".

[44] S. Wan et al. "A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations". In: AAAI. 2016.

[45] P. Liu, X. Qiu, and X. Huang. "Modelling Interaction of Sentence Pair with coupled-LSTMs". In: arXiv preprint arXiv:1605.05573 (2016).

Conclusion of DL4NLP (just kidding)

- Long long ago: you must know intrinsic rules of data.
- Past ten years: you just know effective features of data.
- Nowadays: you just need to have big data.

Challenges & Open Problems

- Depth of network
- Rare words
- Fundamental data structure
- Long-term dependencies
- Natural language understanding & reasoning
- Biology inspired model
- Unsupervised learning

DL4NLP from scratch

- Select a real problem
- Translate your problem to (supervised) learning problem
- Prepare your data and hardware (GPU)
- Select a library: Theano, Keras, TensorFlow
- Find an open source implementation
- Incrementally writing your own code
- Run it.

More Information

《神经网络与深度学习》最新讲义：http://nlp.fudan.edu.cn/dl-book/

Recommended Courses

CS224d: Deep Learning for Natural Language Processing

- http://cs224d.stanford.edu/
- 斯坦福大学Richard Socher
- 主要讲解自然语言处理领域的各种深度学习模型

CS231n:Convolutional Neural Networks for Visual Recognition

- http://cs231n.stanford.edu/
- 斯坦福大学Fei-Fei Li Andrej Karpathy
- 主要讲解CNN、RNN在图像领域的应用