CCF ADL 2015
Nanchang
Oct 11, 2015

Learning to Process Natural Language in Big Data Environment

Hang Li
Noah's Ark Lab
Huawei Technologies

Part 2: Useful Deep Learning Tools

Powerful Deep Learning Tools

- (Unsupervised) Neural Word Embedding
- Recurrent Neural Networks
- Convolutional Neural Networks
- Recursive Neural Networks

Neural Word Embedding

Neural Word Embedding

- Motivation
  - Representing words with lower-dimensional (100~) real-valued vectors
  - Unsupervised setting
  - As input to neural networks
- Tool: Word2Vec
- Method: SGNS (Skip-Gram with Negative Sampling)

Skip-Gram with Negative Sampling (Mikolov et al., 2013)

- Input: occurrences between words and contexts

M	c_1	c_2	c_3	c_4	c_5
w_1	5		1	2	
w_2		2			1
w_3	3			1	

- Probability model:

$$P(D=1 \mid w,c) = \sigma(\vec{w}\cdot\vec{c}) = \frac{1}{1+e^{-\vec{w}\cdot\vec{c}}}$$

$$P(D=0 \mid w,c) = \sigma(-\vec{w}\cdot\vec{c}) = \frac{1}{1+e^{\vec{w}\cdot\vec{c}}}$$

Skip-Gram with Negative Sampling

- Negative sampling: randomly sample unobserved pair w, c_N

$$\mathbf{E}_{c_{N \sim P}}[\log \sigma(-\vec{w} \cdot \vec{c}_N)]$$

- Objective in learning

$$L = \sum_{w} \sum_{c} \#(w,c) \log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbf{E}_{C_N \sim P} \log \sigma(-\vec{w} \cdot \vec{c}_N)$$

- Algorithm: stochastic gradient decent

Interpretation as Matrix Factorization (Levy & Goldberg 2014)

- Pointwise Mutual Information Matrix

M	c_1	c_2	c_3	c_4	c_5
w_1	**3**		**-.5**	**2**	
w_2		1			-0.5
w_3	1.5			1	

$$\log \frac{P(w,c)}{P(w)P(c)}$$

Interpretation as Matrix Factorization

M	c_1	c_2	c_3	c_4	c_5
w_1	**3**		**-.5**	**2**	
w_2		1			-0.5
w_3	1.5			1	

$$M = WC^T$$

matrix factorization, equivalent to SGNS

W	t_1	t_2	t_3
w_1	**7**	**0.5**	**1**
w_2		2.2	3
w_3	1	1.5	1

Word Embedding

Word Representation: Neural Word Embedding (Mikolov et al., 2013)

M	c_1	c_2	c_3	c_4	c_5
w_1	**10**		**1**	**5**	
w_2		2			1
w_3	2.5			1	

$$\log \frac{P(w,c)}{P(w)P(c)}$$

$$M = WC^T$$

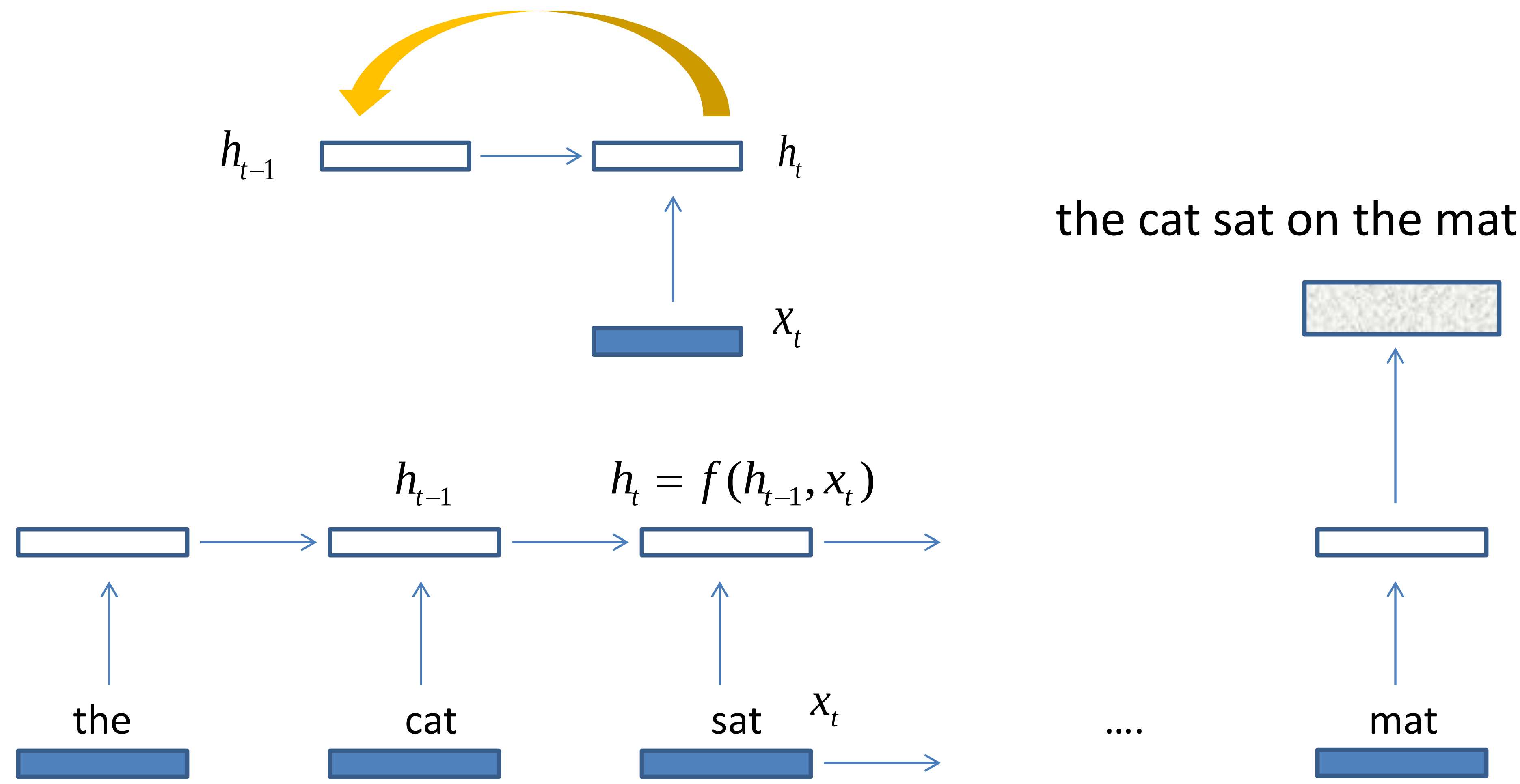
matrix factorization

W	t_1	t_2	t_3
w_1	**7**	**1**	**1**
w_2		2	3
w_3	1	1.5	2

word embedding
or word2vec

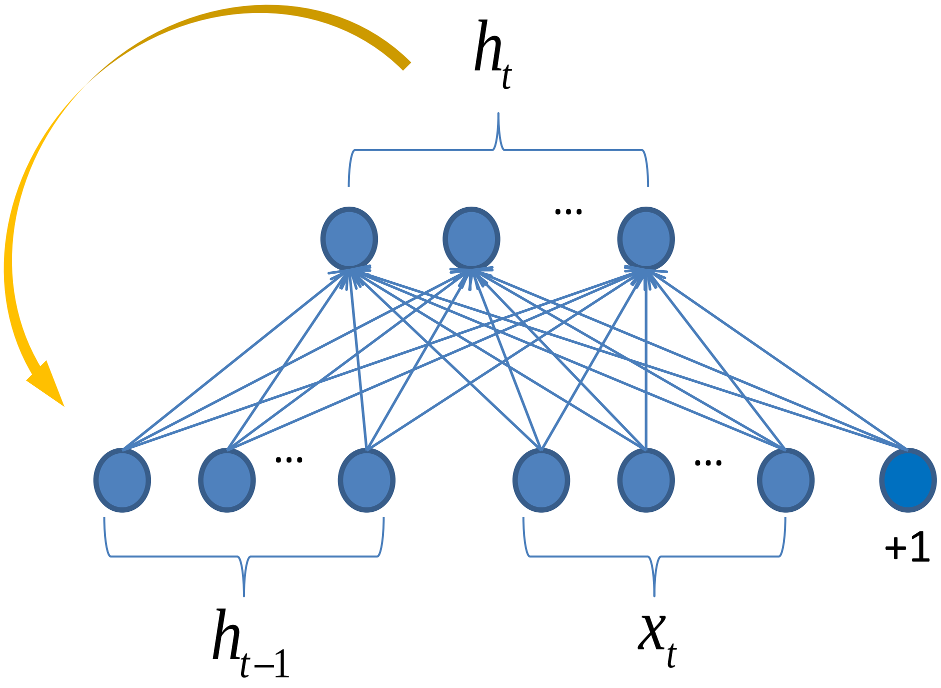
Recurrent Neural Network

Recurrent Neural Network (RNN) (Mikolov et al. 2010)

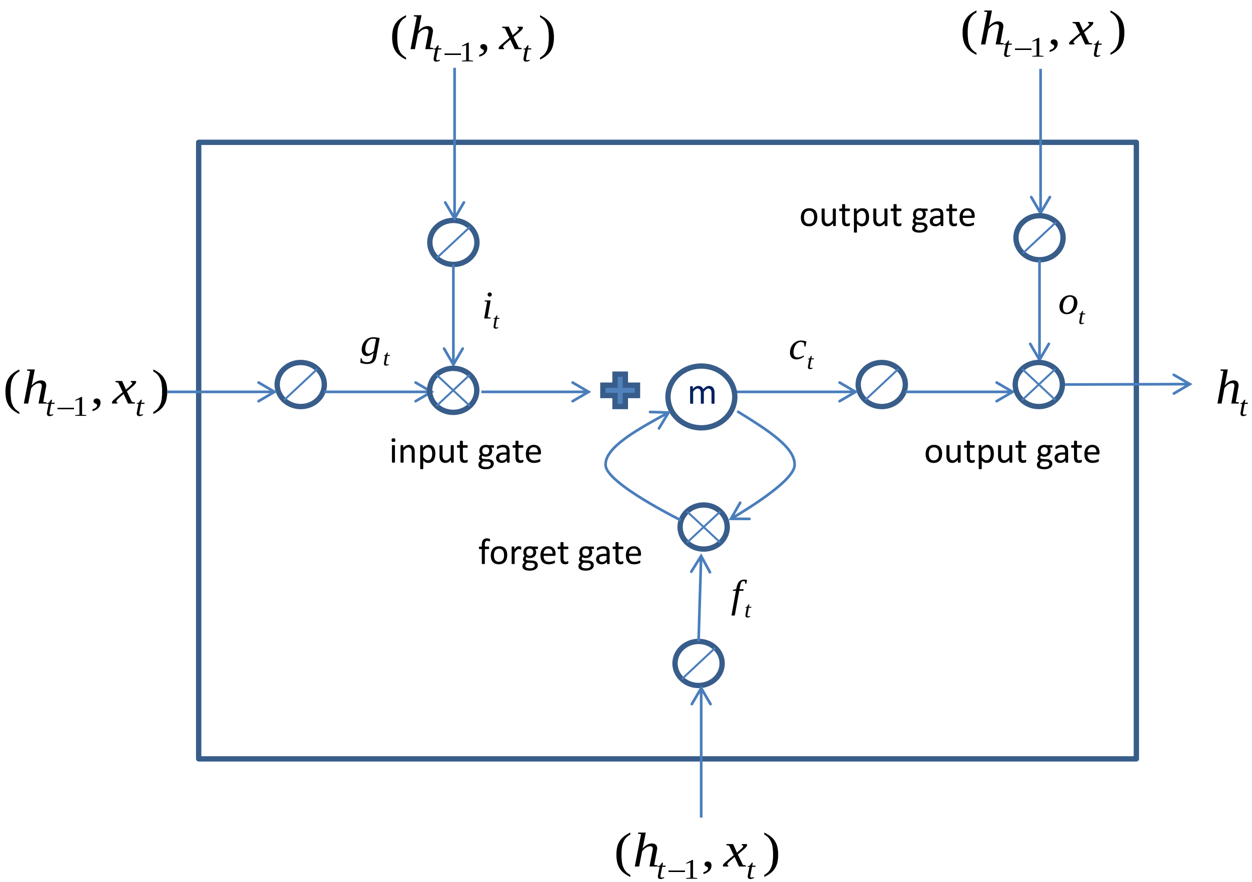


Simple Recurrent Neural Network

$$h_t = f(h_{t-1}, x_t) = \sigma(W_h h_{t-1} + W_x x_t + b)$$



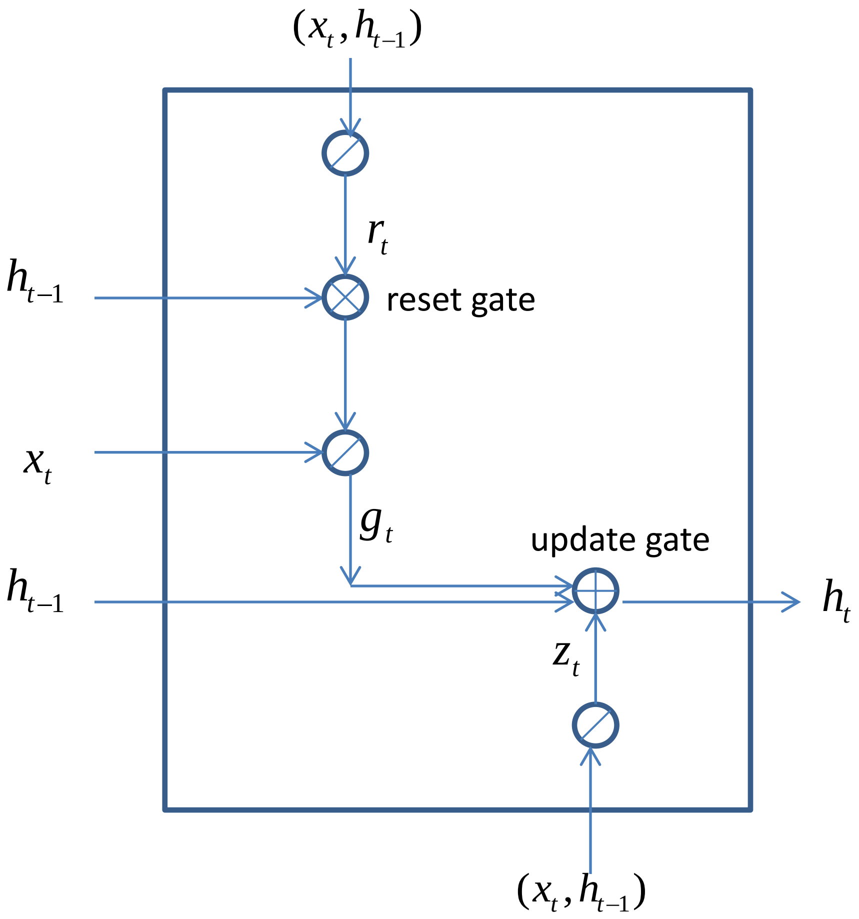
Long Term Short Memory (LSTM) (Hochreiter & Schmidhuber, 1997)



- Have a memory (vector) to memorize previous values
- Use input gate, output gate, forget gate
- Gate: element-wise product with vector with values in [0,1]

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i)$$
$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$
$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$
$$g_t = \tanh(W_{gh}h_{t-1} + W_{gx}x_t + b_g)$$
$$c_t = i_t \otimes g_t + f_t \otimes c_{t-1}$$
$$h_t = o_t \otimes \tanh(c_t)$$

Gated Recurrent Unit (GRU) (Cho et al., 2014)



- Have a memory (vector) to memorize previous values
- Use reset gate, update gate

$$r_t = \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r)$$
$$z_t = \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z)$$
$$g_t = \tanh(W_{gh}(r \otimes h_{t-1}) + W_{gx}x_t + b_g)$$
$$h_t = z_t h_{t-1} + (1 - z_t) g_t$$

Recurrent Neural Network Language Model

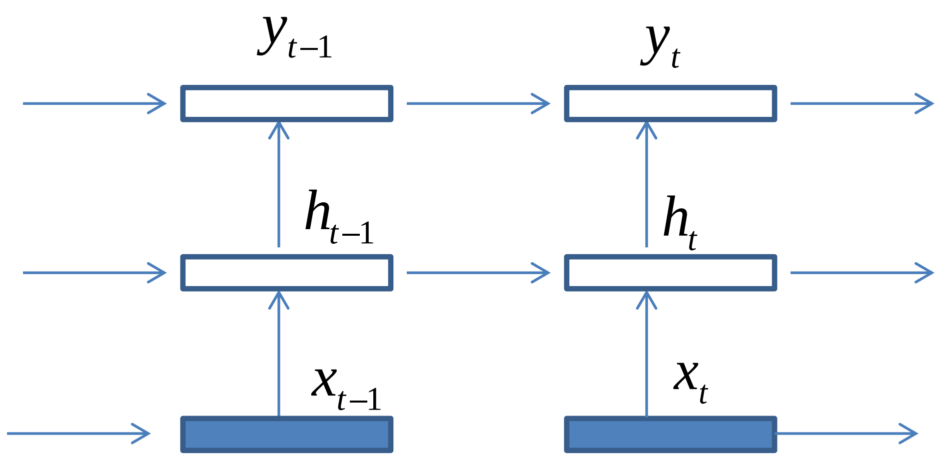
Model

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_{hx})$$

$$y_t = P(y_t = x_t \mid x_1 \cdots x_{t-1}) = \mathrm{soft\,max}(W h_t + b)$$

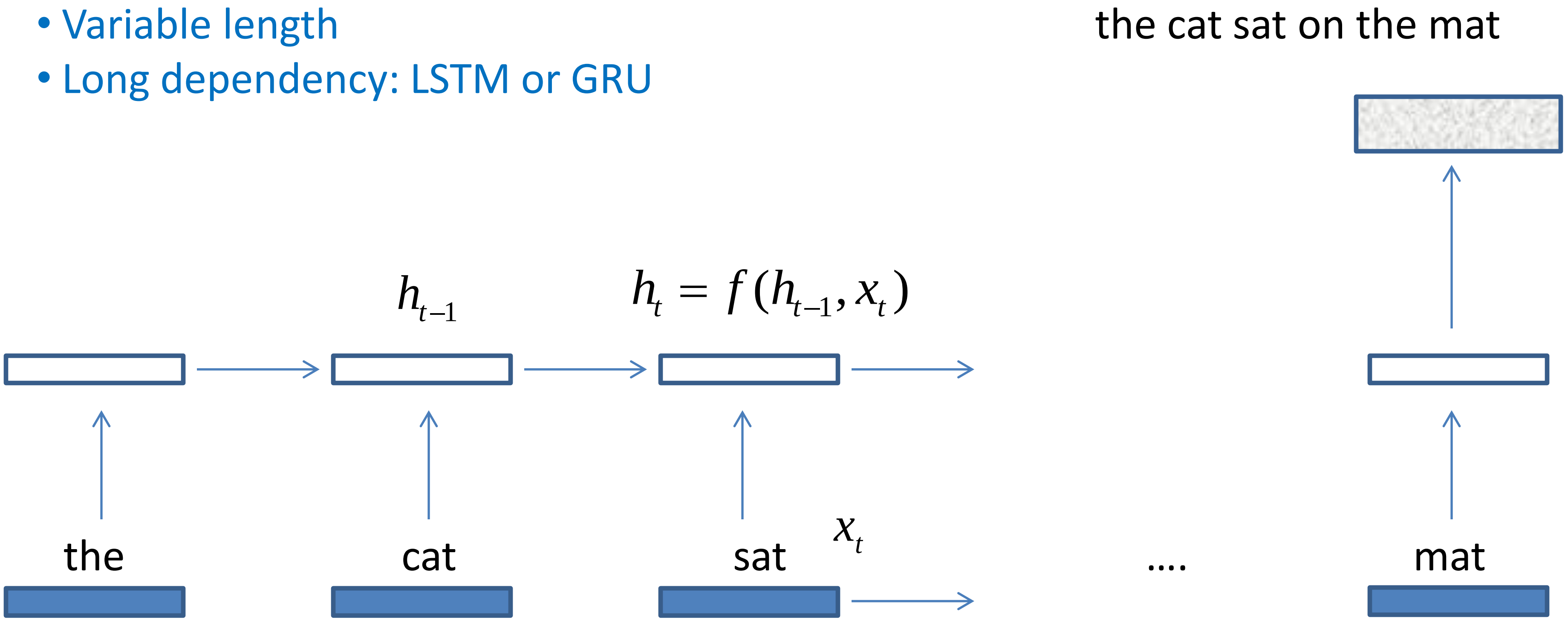
Objective of Learning

$$\frac{1}{T} \sum_{t=1}^{T} -\log \hat{y}_t$$



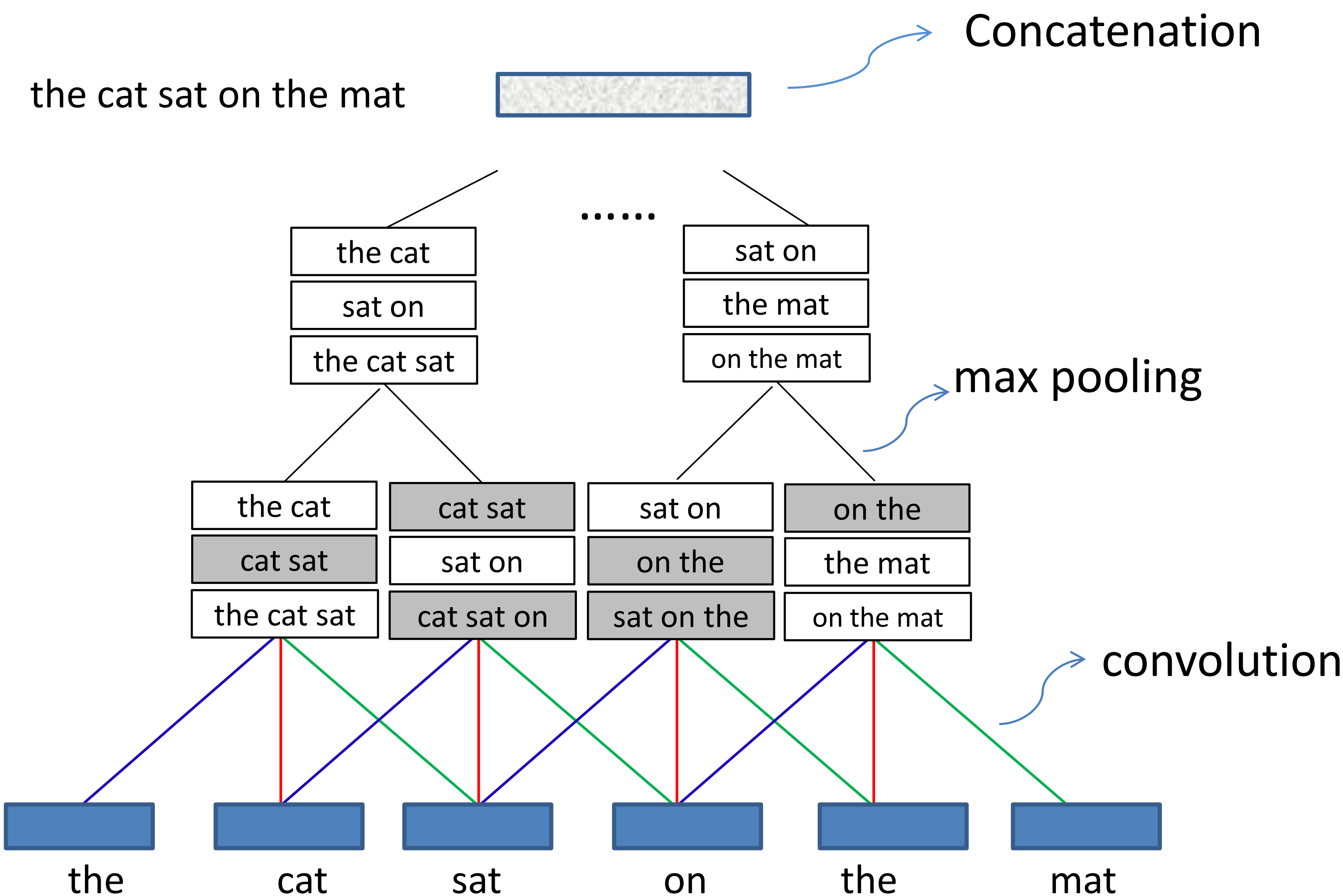
Recurrent Neural Network (RNN) (Mikolov et al. 2010)

- On sequence of words
- Variable length
- Long dependency: LSTM or GRU



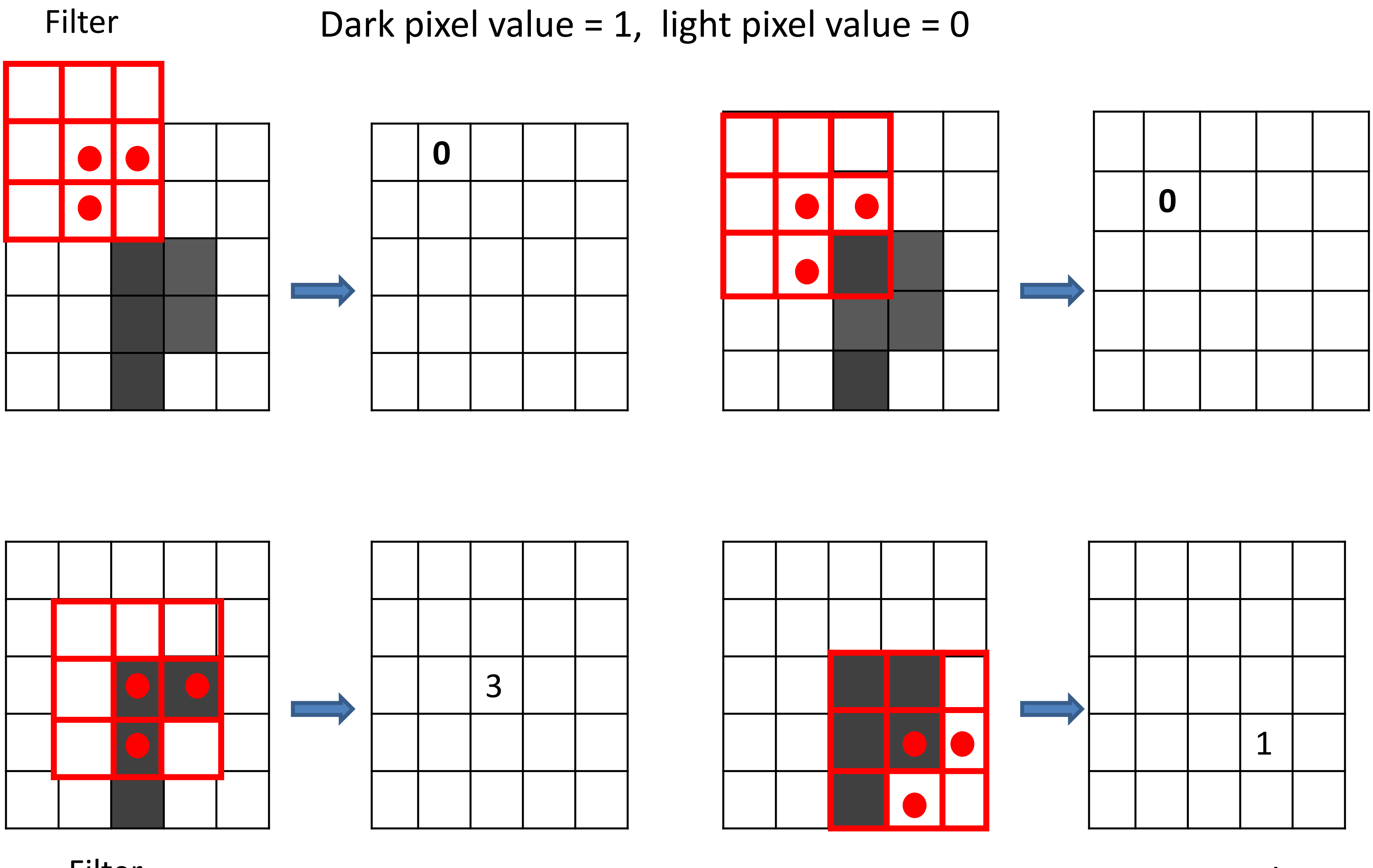
Convolutional Neural Network

Convolutional Neural Network (CNN)
(Hu et al., 2014)



Example: Image Convolution

Dark pixel value = 1, light pixel value = 0

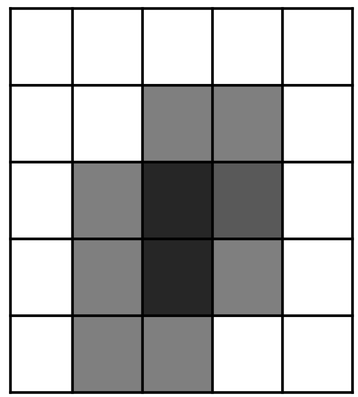


Leow Wee Kheng

Example: Image Convolution

0	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**0**
0	**1**	**3**	**2**	**0**
0	**1**	**3**	**1**	**0**
0	**1**	**1**	**0**	**0**

Feature Map



Convolution

$$z_i^{(l,f)} = \sigma(w^{(l,f)} \cdot z_i^{(l-1)} + b^{(l,f)}) \quad f = 1,2,\cdots,F_l$$

$z_i^{(l,f)}$ is output of neuron of type f for location i in layer l

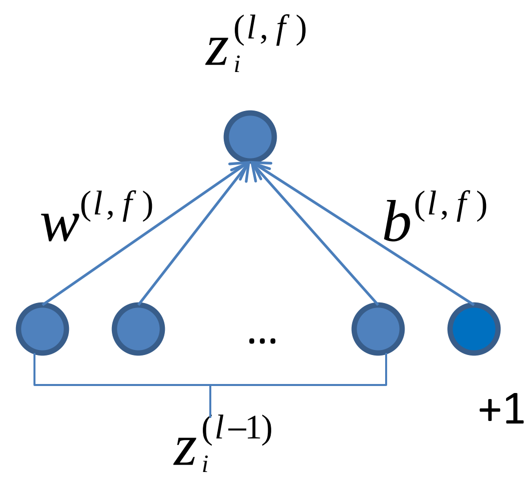
$w^{(l,f)}, b^{(l,f)}$ are parameters of neuron of type f in layer l

σ is sigmoid function

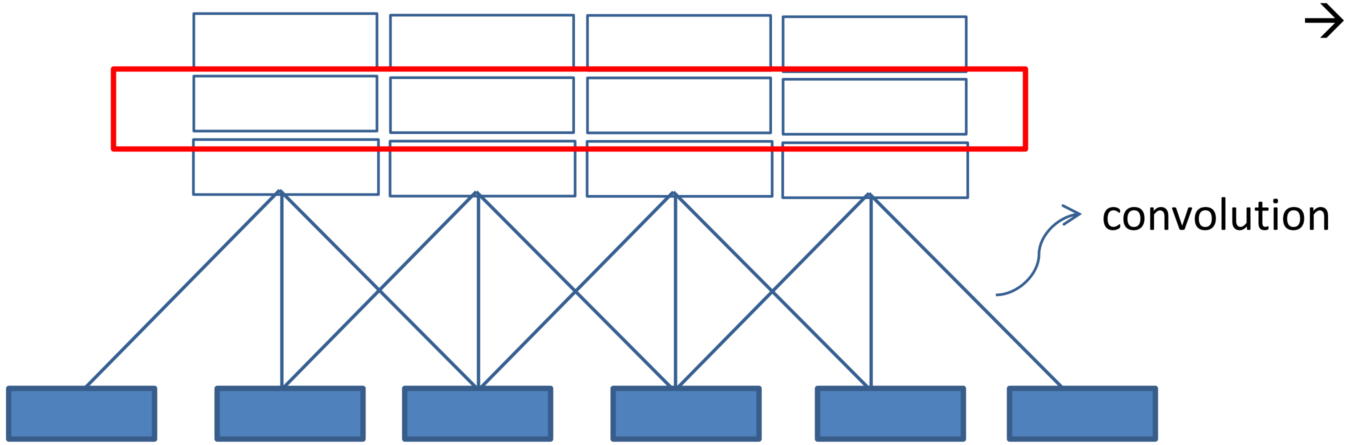
$z_i^{(l-1)}$ is input of neuron for location i from layer $l-1$

$z_i^{(0)}$ is input from cancatenated word vectors for location i

$$z_i^{(0)} = [x_i^T, x_{i+1}^T, \cdots x_{i+h-1}^T]^T$$



Filter → feature map
→ neuron

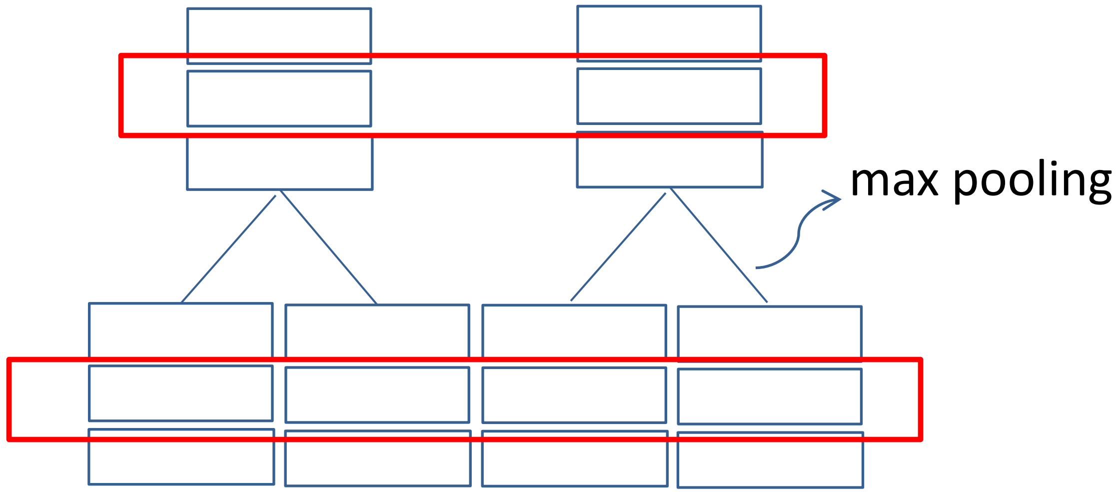


Max Pooling

$$z_i^{(l,f)} = \max(z_{2i-1}^{(l-1,f)}, z_{2i}^{(l-1,f)})$$

$z_i^{(l,f)}$ is output of pooling of type f for location i in layer l

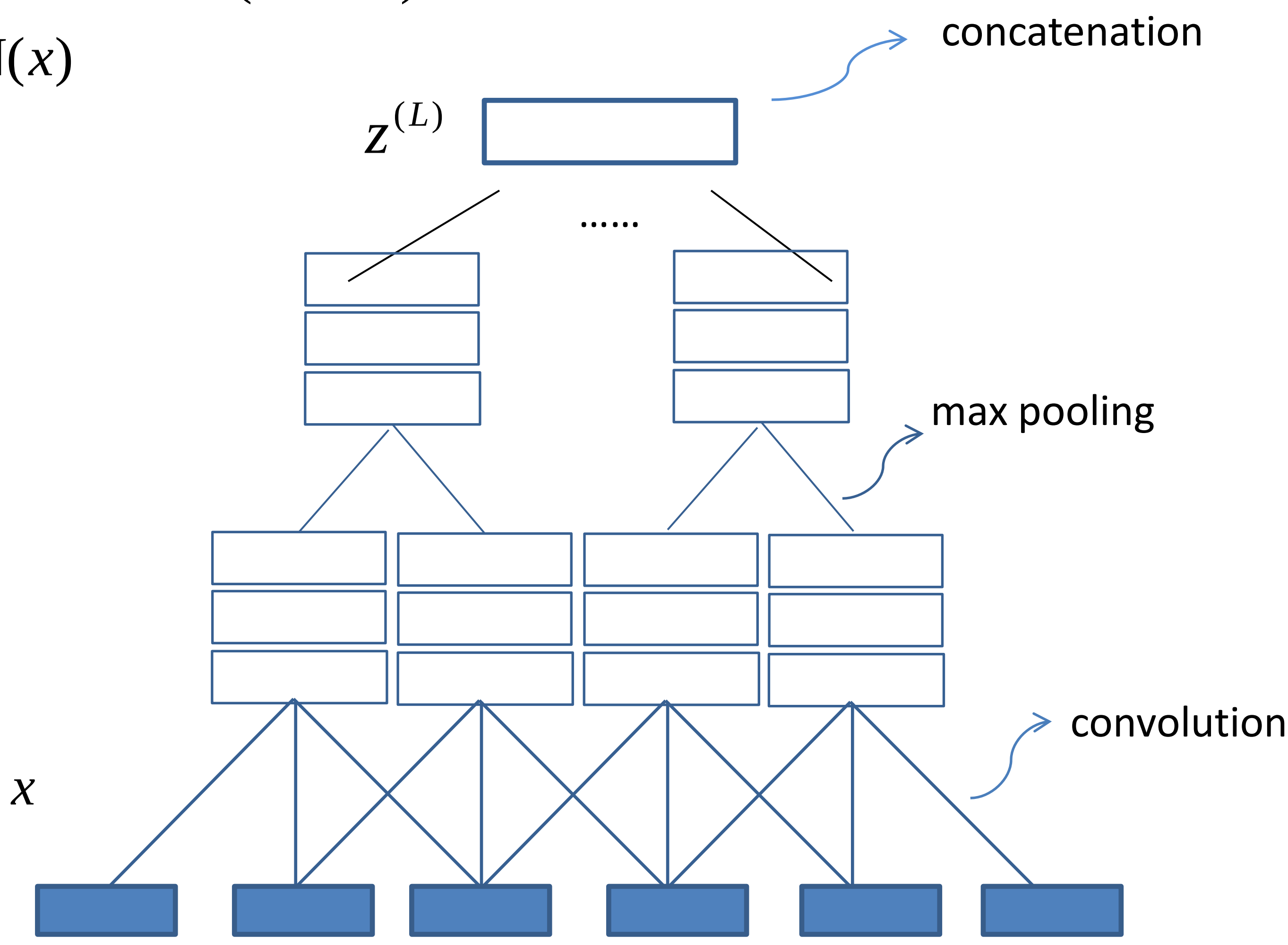
$z_{2i-1}^{(l-1,f)}, z_{2i}^{(l-1,f)}$ are input of pooling of type f for location i in layer l



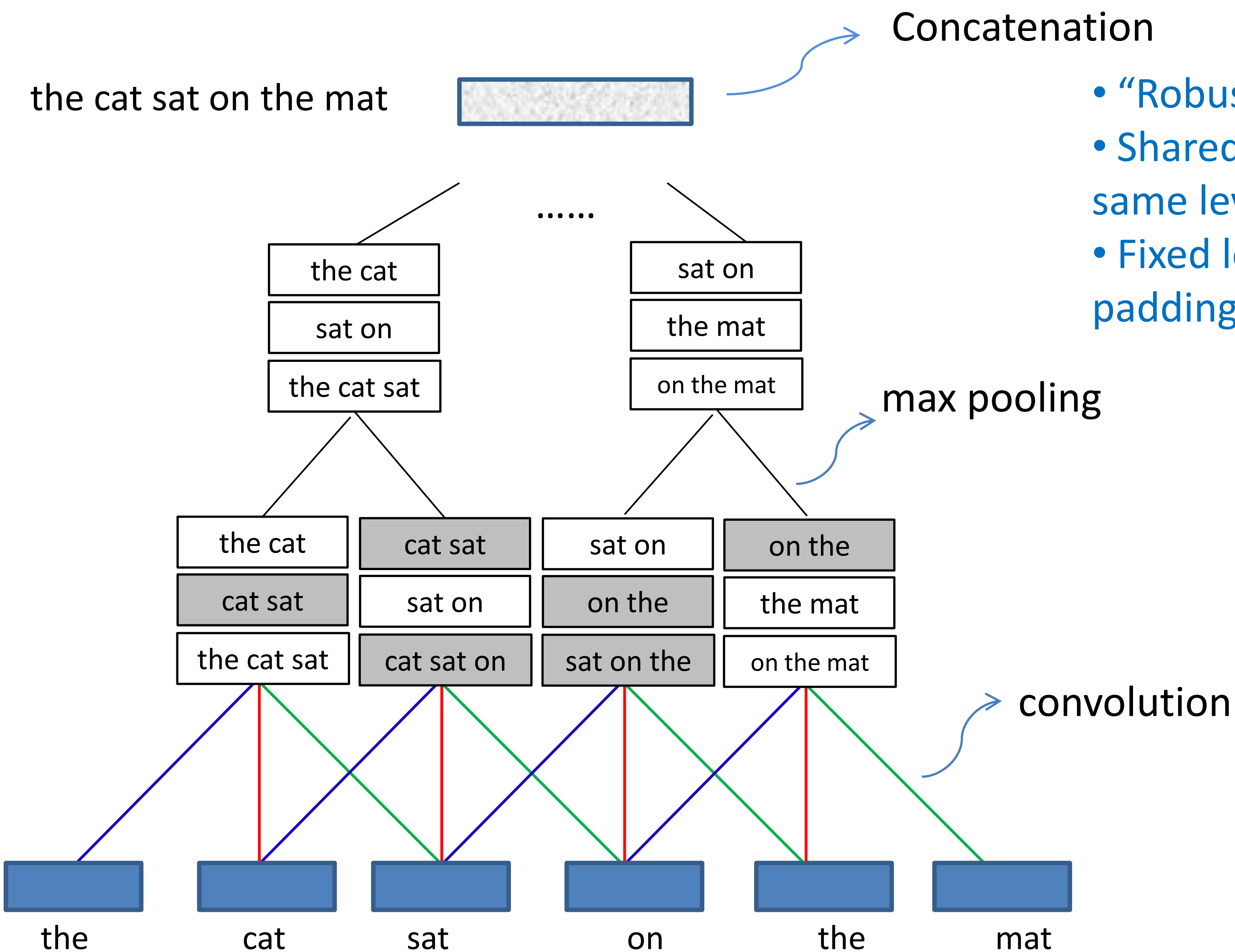
Sentence Classification Using Convolutional Neural Network

$$y = f(x) = \mathrm{softmax}(Wz + b)$$

$$z = \mathrm{CNN}(x)$$



Convolutional Neural Network (CNN)
(Hu et al. 2014, Blunsom et al. 2014)

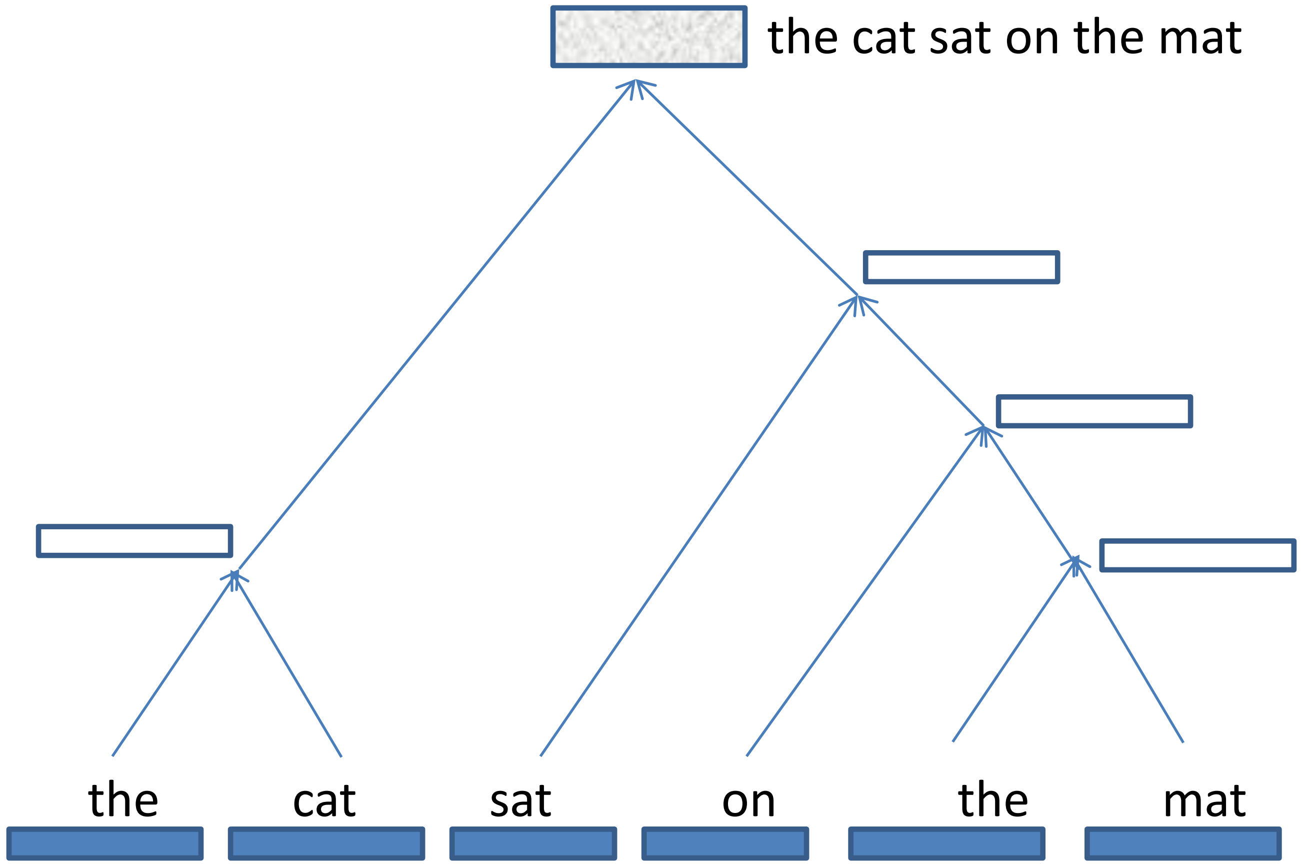


- “Robust parsing”
- Shared parameter on same level
- Fixed length, zero padding

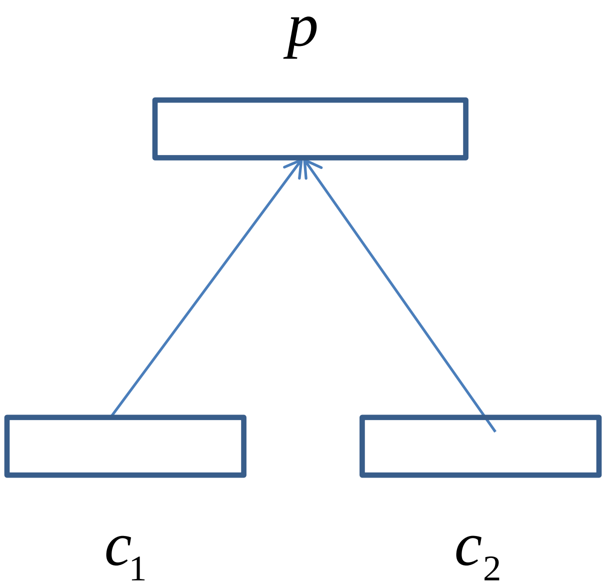
Recursive Neural Network

Recursive Neural Network

(Socher et al., 2013)



Recursive Neural Network



$$score = U \cdot p$$

$$p = \tanh\left[W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b \right]$$

Learning of Recursive Neural Network

- The score of a tree is the sum of the scores of its nodes.

$$s(x, y) = \sum_{n \in nodes(y)} s\ (n)$$

- Max margin parsing

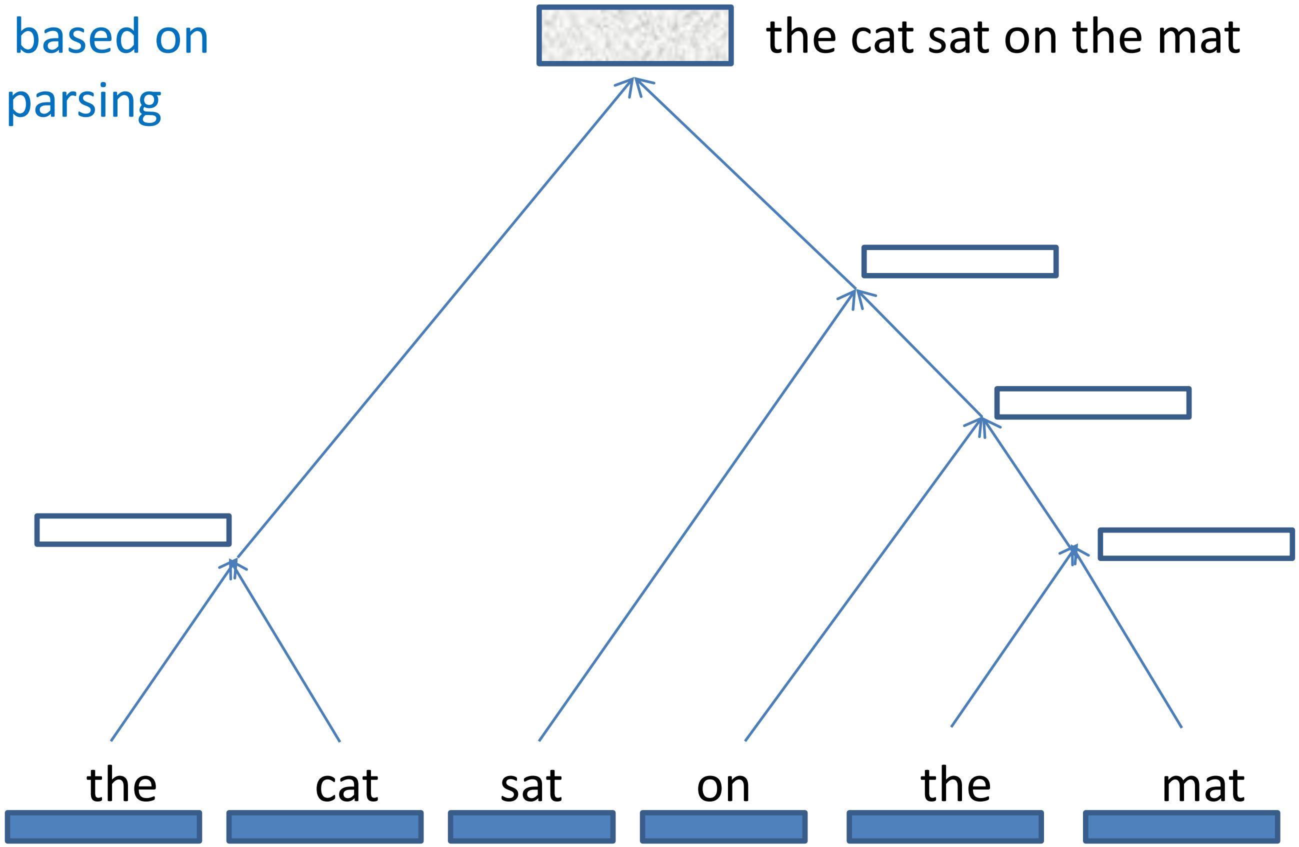
$$L = s(x, y) - \max_{z \in Z(x)} \left(s(x, z) + \Delta(y, z) \right)$$

$$\Delta(y, z) : \text{penalty on incorrect tree}$$

$$Z(x) : \text{greedily searched trees}$$

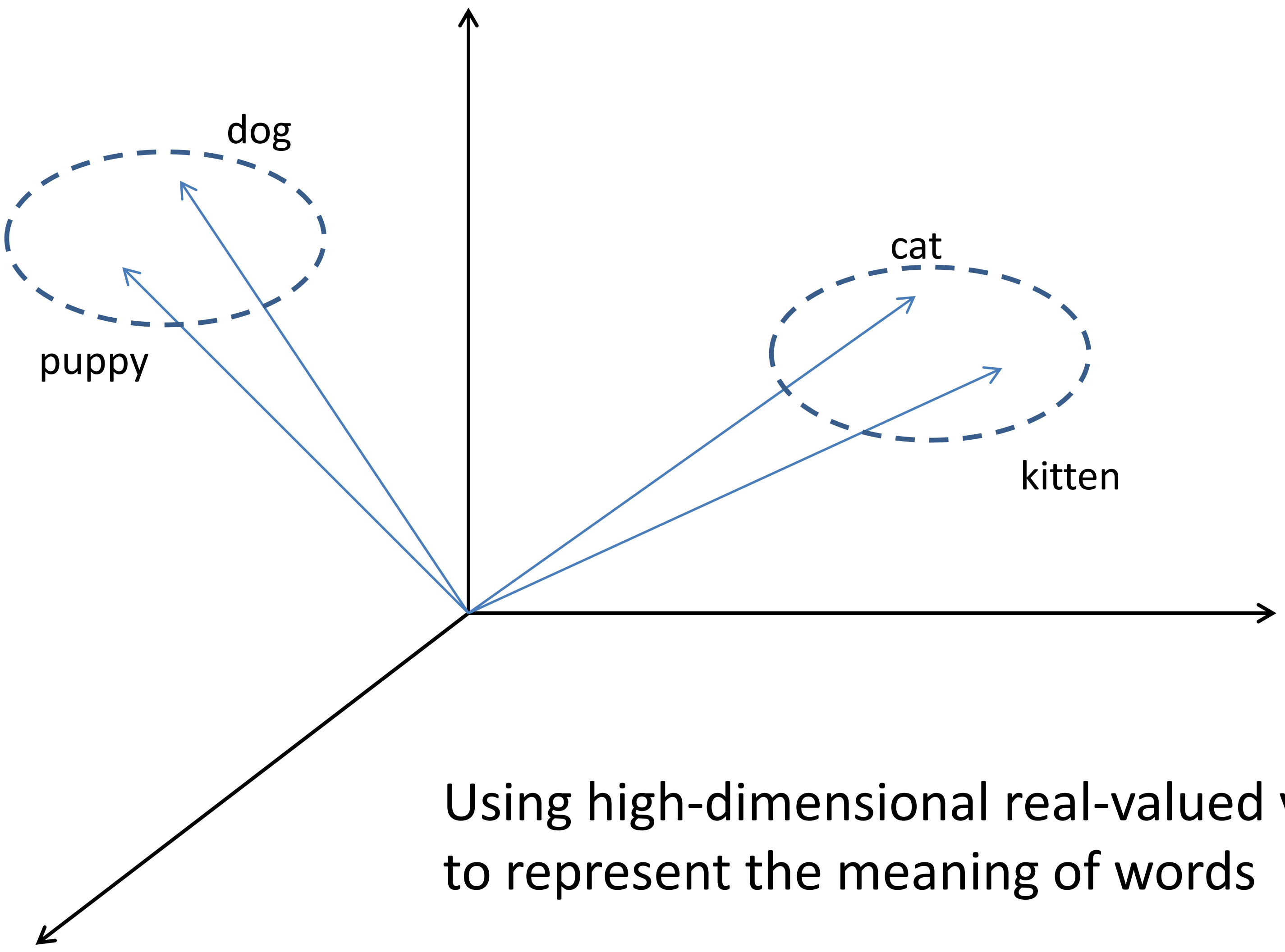
Recursive Neural Network (RNN) (Socher et al. 2013)

- On parse tree of sentence
- Learning is based on
max margin parsing



Learning of Sentence Representation

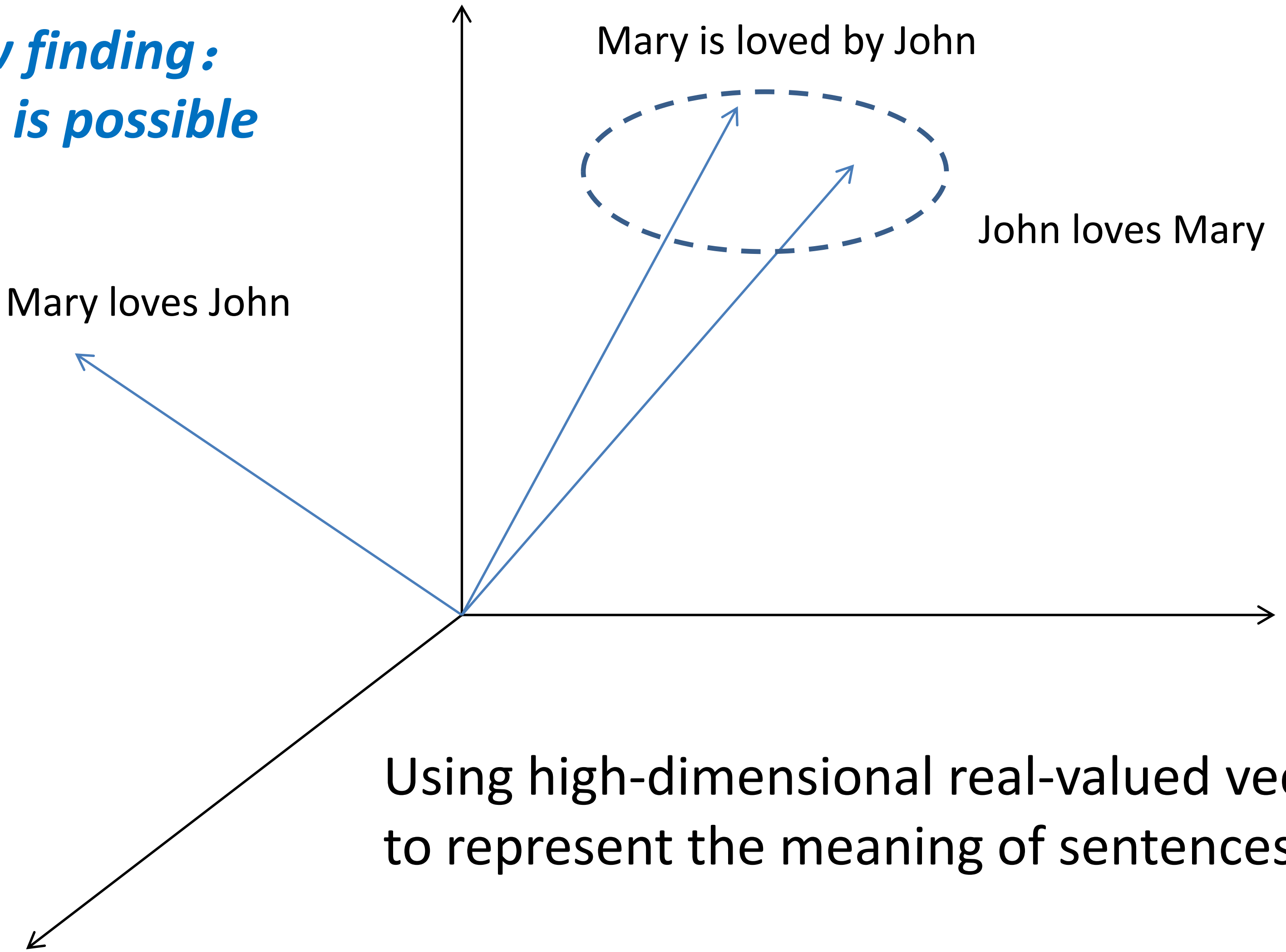
Representation of Word Meaning



Using high-dimensional real-valued vectors to represent the meaning of words

Representation of Sentence Meaning

New finding：
This is possible



Using high-dimensional real-valued vectors
to represent the meaning of sentences

Recent Breakthrough in Distributional Linguistics

- From words to sentences
- Compositional
- Representing syntax, semantics, even pragmatics

How Is Learning of Sentence Meaning Possible?

- Deep neural networks (complicated non-linear models)
- Big Data
- Task-oriented
- Error-driven and gradient-based

Natural Language Tasks

- Classification: assigning a label to a string

$$s \rightarrow c$$

- Generation: creating a string

$$\rightarrow s$$

- Matching: matching two strings

$$s, t \rightarrow \mathbf{R}^{+}$$

- Translation: transforming one string to another

$$s \rightarrow t$$

- Structured prediction: mapping string to structure

$$s \rightarrow s'$$

Natural Language Applications Can Be Formalized as Tasks

- Classification
  - Sentiment analysis
- Generation
  - Language modeling
- Matching
  - Search
  - Question answering
- Translation
  - Machine translation
  - Natural language dialogue (single turn)
  - Text summarization
  - Paraphrasing
- Structured Prediction
  - Information Extraction
  - Parsing

Learning of Representations in Tasks

- Classification

$$s \to r \to c$$

- Generation

$$\to s(r)$$

- Matching

$$s, t \to r \to \mathbf{R}^+$$

- Translation

$$s \to r \to t$$

- Structured Prediction

$$s \to s' + r$$

Our Observation

- Unsupervised word-embedding (e.g., Word2Vec) is needed, only when there is not enough data for supervised word-embedding
- Convolutional Neural Network is suitable for matching tasks
- Recurrent Neural Network is suitable for generation tasks
- Not observed so far that Recursive Neural Network works better than the other two models

References

- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent Neural Network based Language Model. *InterSpeech* 2010.
- Omer Levy, Yoav Goldberg, and Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *TACL* 2015 pp. 211-225.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality. *NIPS* 2013, pp. 3111-3119.
- Hochreiter, S., & Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780, 1997.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*.
- Hu, B., Lu, Z., Li, H., & Chen, Q. Convolutional Neural Network Architectures for Matching Natural Language Sentences. *NIPS* 2014 (pp. 2042-2050).
- Blunsom, P., Grefenstette, E., & Kalchbrenner, N. (2014). A Convolutional neural network for modeling sentences. *ACL* 2014.
- Socher, Richard, John Bauer, Christopher D. Manning, and Andrew Y. Ng. "Parsing with compositional vector grammars." *ACL* 2013.

Thank you!

hangli.hl@huawei.com