Neural Networks in NLP: The Curse of Indifferentiability

Lili Mou
doublepower.mou@gmail.com
http://sei.pku.edu.cn/~moull12

Outline

- **Preliminary**
  - **Word embeddings**
  - Sequence-to-sequence generation
- Indifferentiability, solutions, and applications
- A case study in semantic parsing

Language Modeling

- One of the most fundamental problems in NLP.
- Given a corpus $\mathbf{w}=w_1w_2\ldots w_t$, the goal is to maximize $p(\mathbf{w})$

Language Modeling

- One of the most fundamental problems in NLP.
- Given a corpus $\mathbf{w}=w_1w_2\ldots w_t$, the goal is to maximize $p(\mathbf{w})$
- Philosophical discussion: Does "probability of a corpus/sentence" make sense?
  - Recognize speech
  - Wreck a nice beach
- All in all, NLP (especially publishing in NLP) is pragmatic.

Decomposition of the Joint probability

- $p(\mathbf{w}) = p(w_1)p(w_1|w_2)p(w_3|w_1w_2) \ldots p(w_t|w_1w_2 \ldots w_{t-1})$

Decomposition of the Joint probability

- $p(\mathbf{w}) = p(w_1)p(w_1|w_2)p(w_3|w_1w_2) \ldots p(w_t|w_1w_2 \ldots w_{t-1})$

Minor question:

- Can we decompose any probabilistic distribution into this form? Yes.
- Is it necessary to decompose a probabilistic distribution into this form? No.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, Zhi Jin. "Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation." In COLING, 2016.

Markov Assumption

- $p(\mathbf{w}) = p(w_1)p(w_1|w_2)p(w_3|w_1w_2) \ldots p(w_t|w_1w_2...w_{t-1})$

  $\approx p(w_1)p(w_1|w_2)p(w_3|w_2) \ldots p(w_t|w_{t-1})$

- A word is dependent only on its previous n-1 words and independent of its position,

  I.e., provided with the previous n-1 words, the current word is independent of other random variables.

$$p(\boldsymbol{w}) \approx \prod_{t=1}^{m} p\left(w_t \middle| \boldsymbol{w}_{t-n+1}^{t-1}\right)$$

Multinomial Estimate

- Maximum likelihood estimation for a multinomial distribution is merely counting.

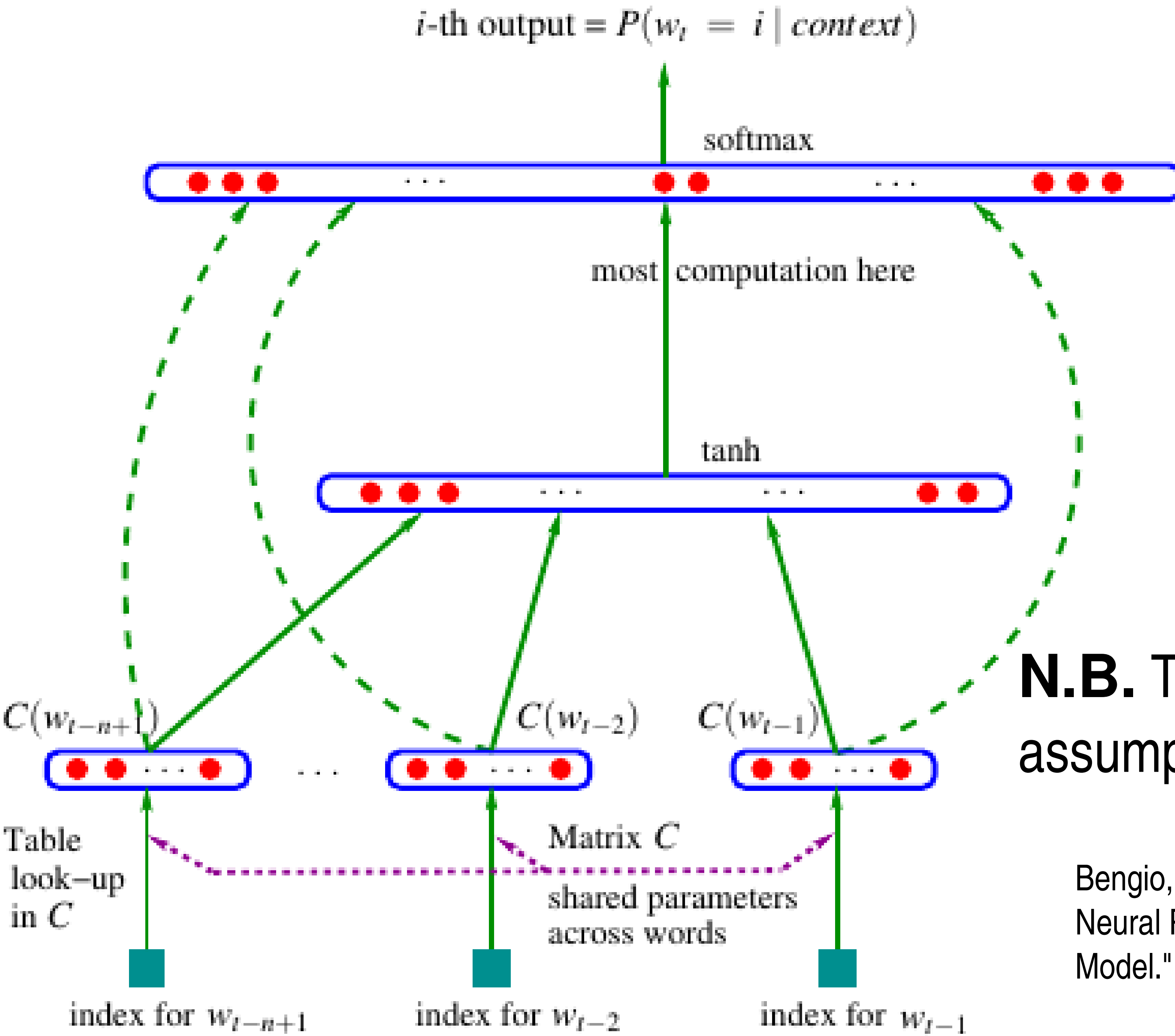
$$p(w_n|\boldsymbol{w}_1^{n-1}) = \frac{\#\boldsymbol{w}_1^n}{\#\boldsymbol{w}_1^{n-1}}$$

- Problems
  - #para grows exp. w.r.t. *n*
  - Even for very small *n* (e.g., 2 or 3), we come across severe data sparsity because of the Zipf distribution

Parameterizing LMs with Neural Networks

- Each word is mapped to a real-valued vector, called *embeddings*.
- Neural layers capture context information (typically previous words).
- The probability $p(w|\cdot)$ is predicted by a softmax layer.

Feed-Forward Language Model

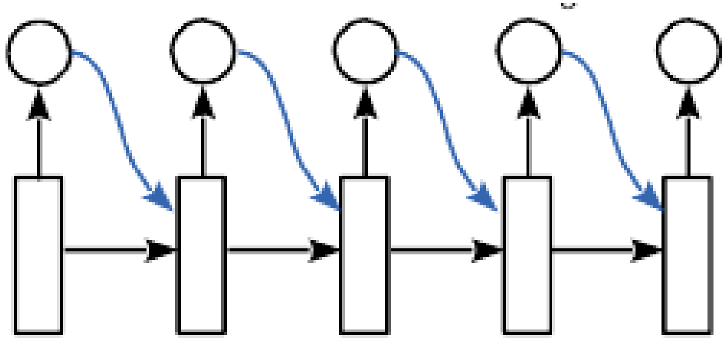


N.B. The Markov assumption also holds.

Bengio, Yoshua, et al. "A Neural Probabilistic Language Model." JMLR. 2003.

Recurrent Neural Language Model

- RNN keeps one or a few hidden states
- The hidden states change at each time step according to the input



$$\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$
$$= f\left(W_{\mathrm{in}}\boldsymbol{x}_t + W_{\mathrm{hid}}\boldsymbol{h}_{t-1}\right)$$
$$p(w_t|\boldsymbol{w}_0^{t-1}) \approx \mathrm{softmax}\left(W_{\mathrm{out}}\boldsymbol{h}_t\right)$$

- RNN directly parametrizes $p(\boldsymbol{w}) = \prod_{t=1}^{m} p(w_t|\boldsymbol{w}_1^{t-1})$

  rather than $p(\boldsymbol{w}) \approx \prod_{t=1}^{m} p\left(w_t|\boldsymbol{w}_{t-n+1}^{t-1}\right)$

Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S. Recurrent neural network based language model. In INTERSPEECH, 2010.

Complexity Concerns

- Time complexity
  - Hierarchical softmax [1]
  - Negative sampling: Hinge loss [2], Noisy contrastive estimation [3]
- Memory complexity
  - Compressing LM [4]
- Model complexity
  - Shallow neural networks are still too “deep.”
  - CBOW, SkipGram [3]

[1] Mnih A, Hinton GE. A scalable hierarchical distributed language model. NIPS, 2009.

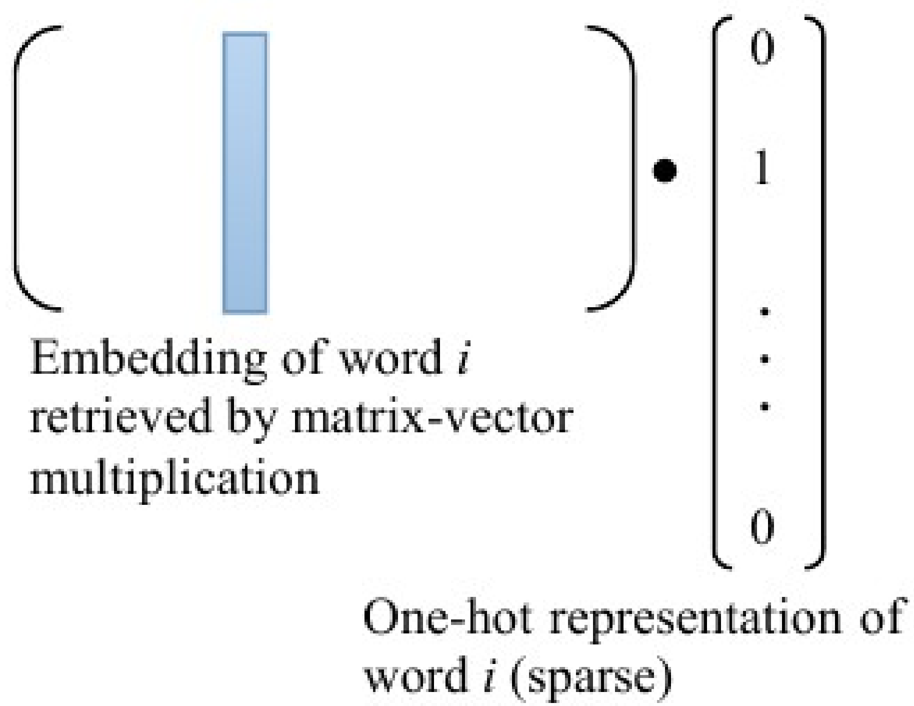
[2] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. JMLR, 2011.

[3] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013

[4] Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, Zhi Jin. "Compressing neural language models by sparse word representations." In ACL, 2016.

The Role of Word Embeddings?

- Word embeddings are essentially a connectional weight matrix, whose input is a one-hot vector.
- Implementing by a look-up table is much faster than matrix multiplication.
- Each column of the matrix corresponds to a word.

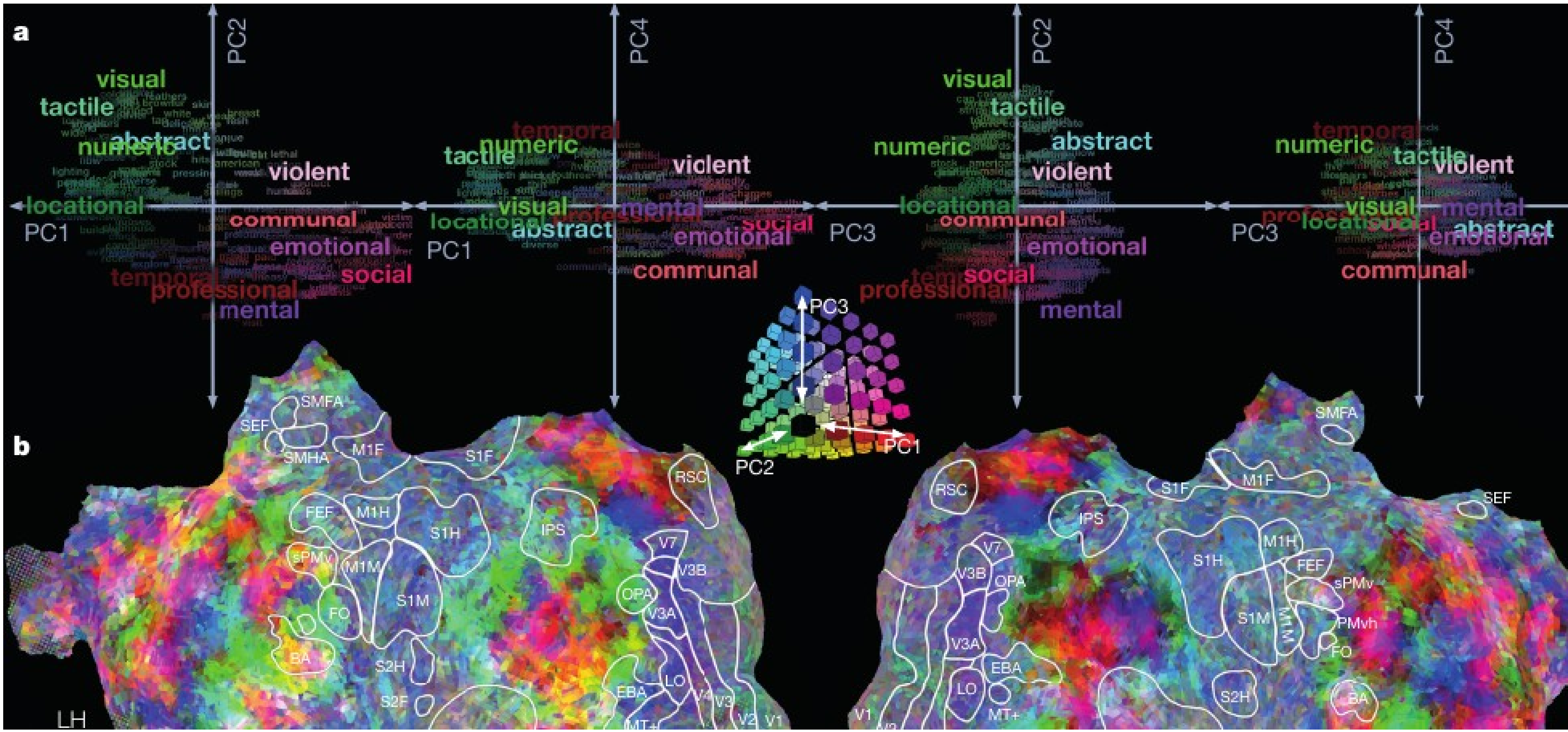


How can we use word embeddings?

- Embeddings demonstrate the internal structures of words
  - Relation represented by vector offset

    “man” – “woman” = “king” – “queen”
  - Word similarity
- Embeddings can serve as the initialization of almost every supervised task
  - A way of pretraining
  - **N.B.:** may not be useful when the training set is large enough

Word Embeddings in our Brain



Huth, Alexander G., et al. "Natural speech reveals the semantic maps that tile human cerebral cortex." Nature 532.7600 (2016): 453-458.

“Somatotopic Embeddings” in our Brain

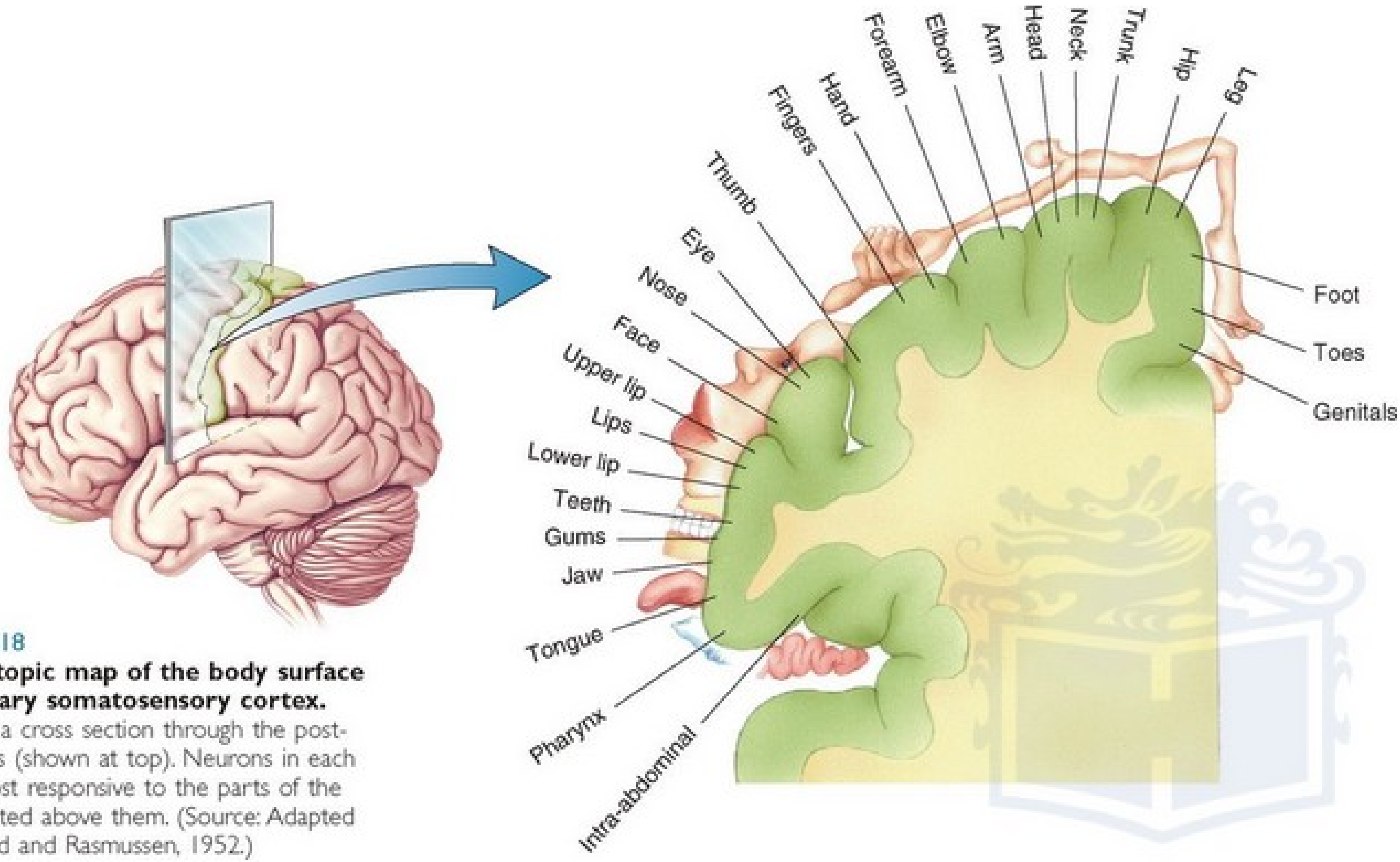


FIGURE 12.18

A somatotopic map of the body surface onto primary somatosensory cortex. This map is a cross section through the post-central gyrus (shown at top). Neurons in each area are most responsive to the parts of the body illustrated above them. (Source: Adapted from Penfield and Rasmussen, 1952.)

[8] Bear MF, Connors BW, Michael A. Paradiso. Neuroscience: Exploring the Brain. 2007

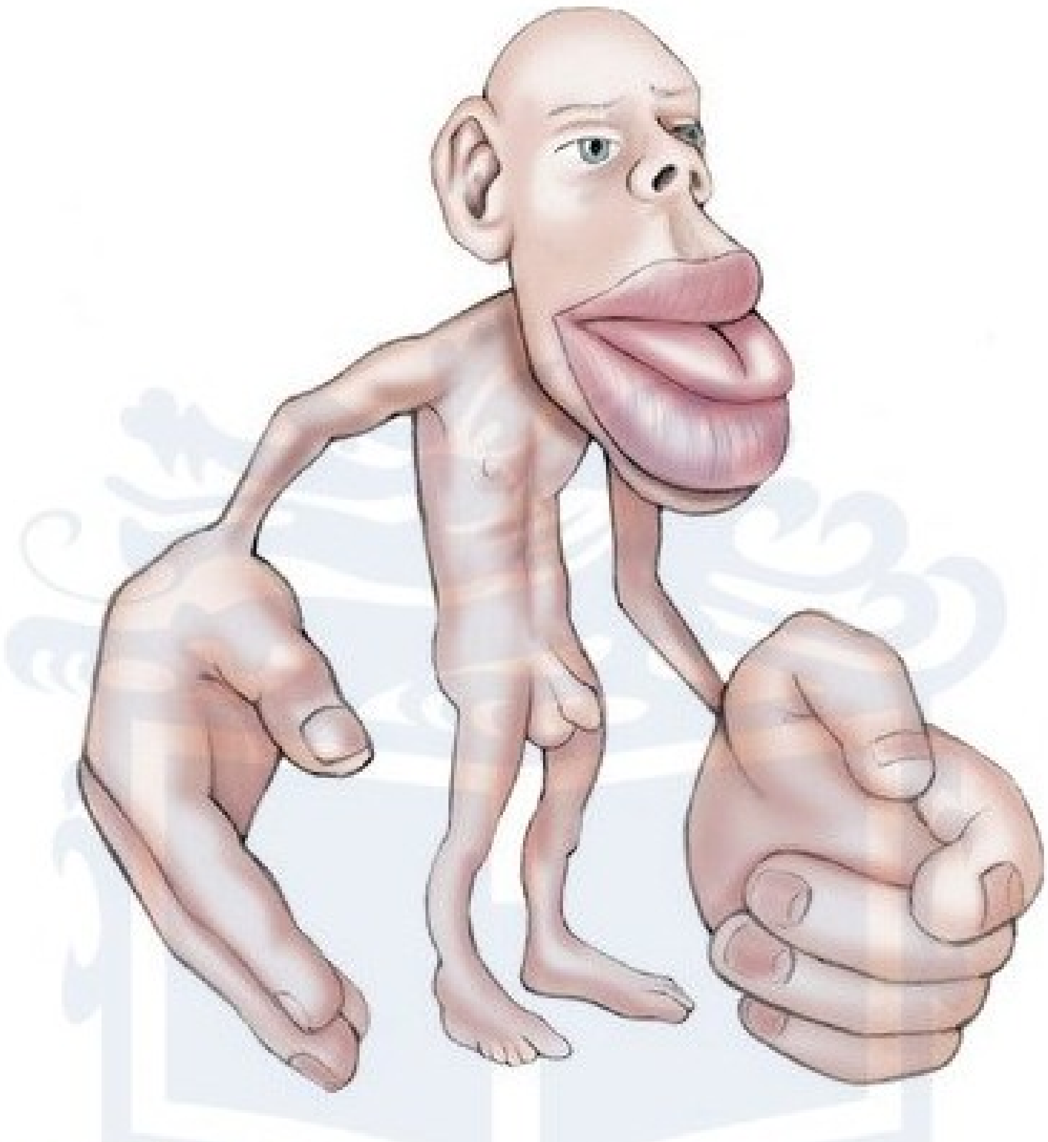
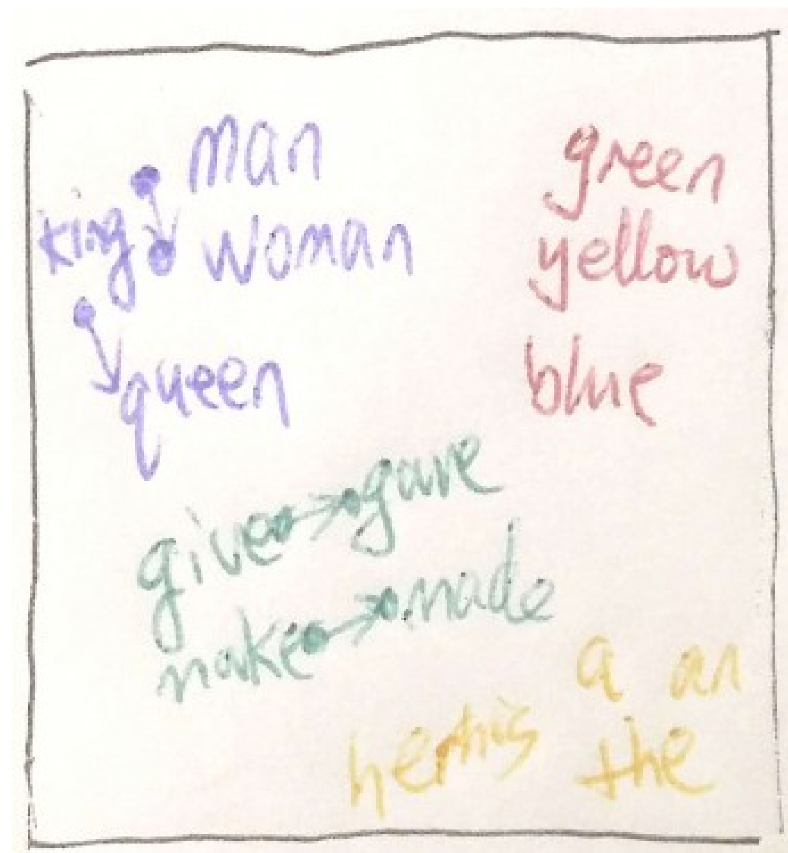


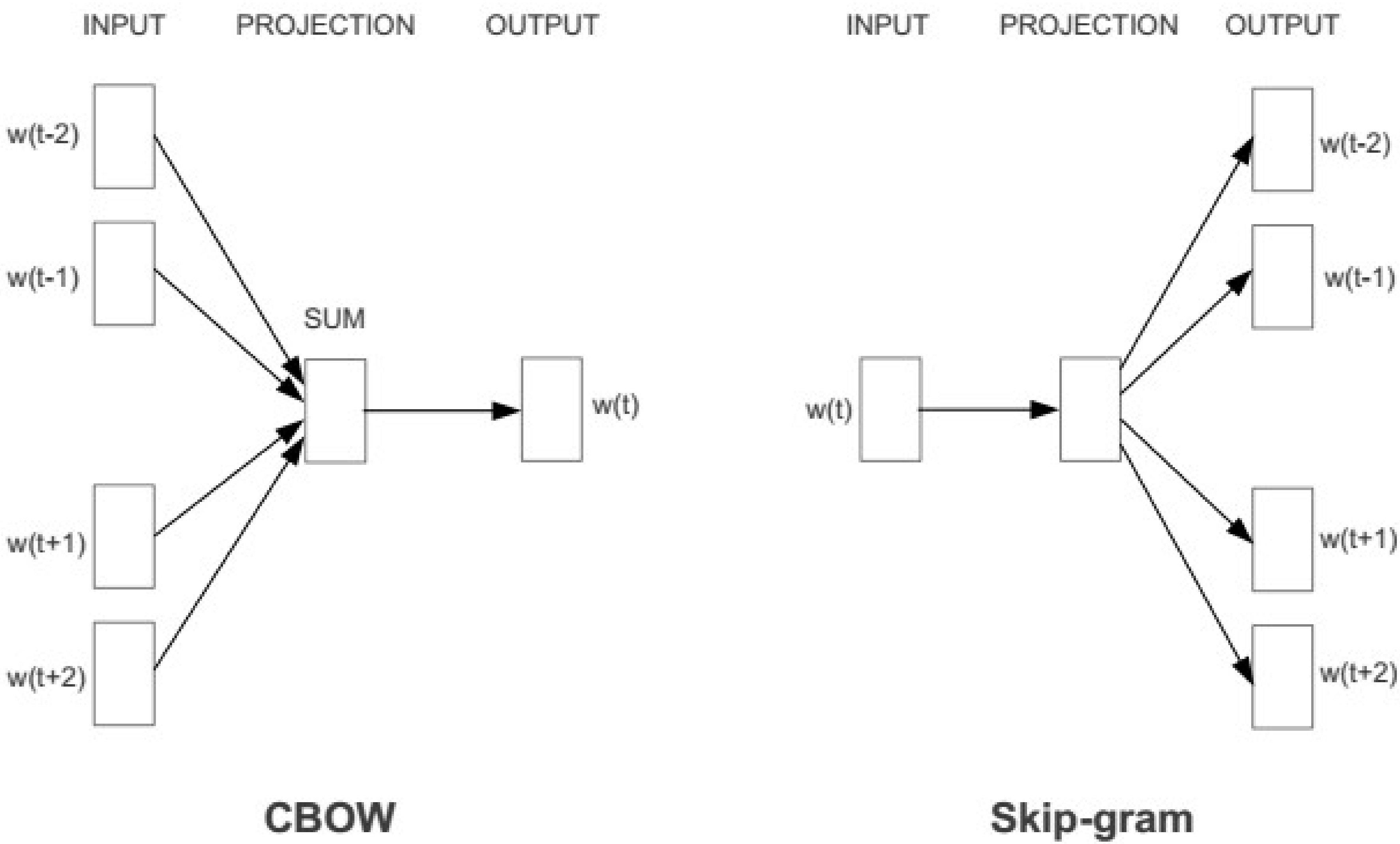
FIGURE 12.19
The homunculus.

[8] Bear MF, Connors BW, Michael A. Paradiso. Neuroscience: Exploring the Brain. 2007

Deep neural networks:
To be, or not to be? That is the question.



CBOW, SkipGram (word2vec)



[6] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013

Hierarchical Softmax and Negative Contrastive Estimation

- HS

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I} \right)$$

- NCE

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

[6] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013

Tricks in Training Word Embeddings

- The # of negative samples?
  - The more, the better.
- The distribution from which negative samples are generated? Should negative samples be close to positive samples?
  - The closer, the better.

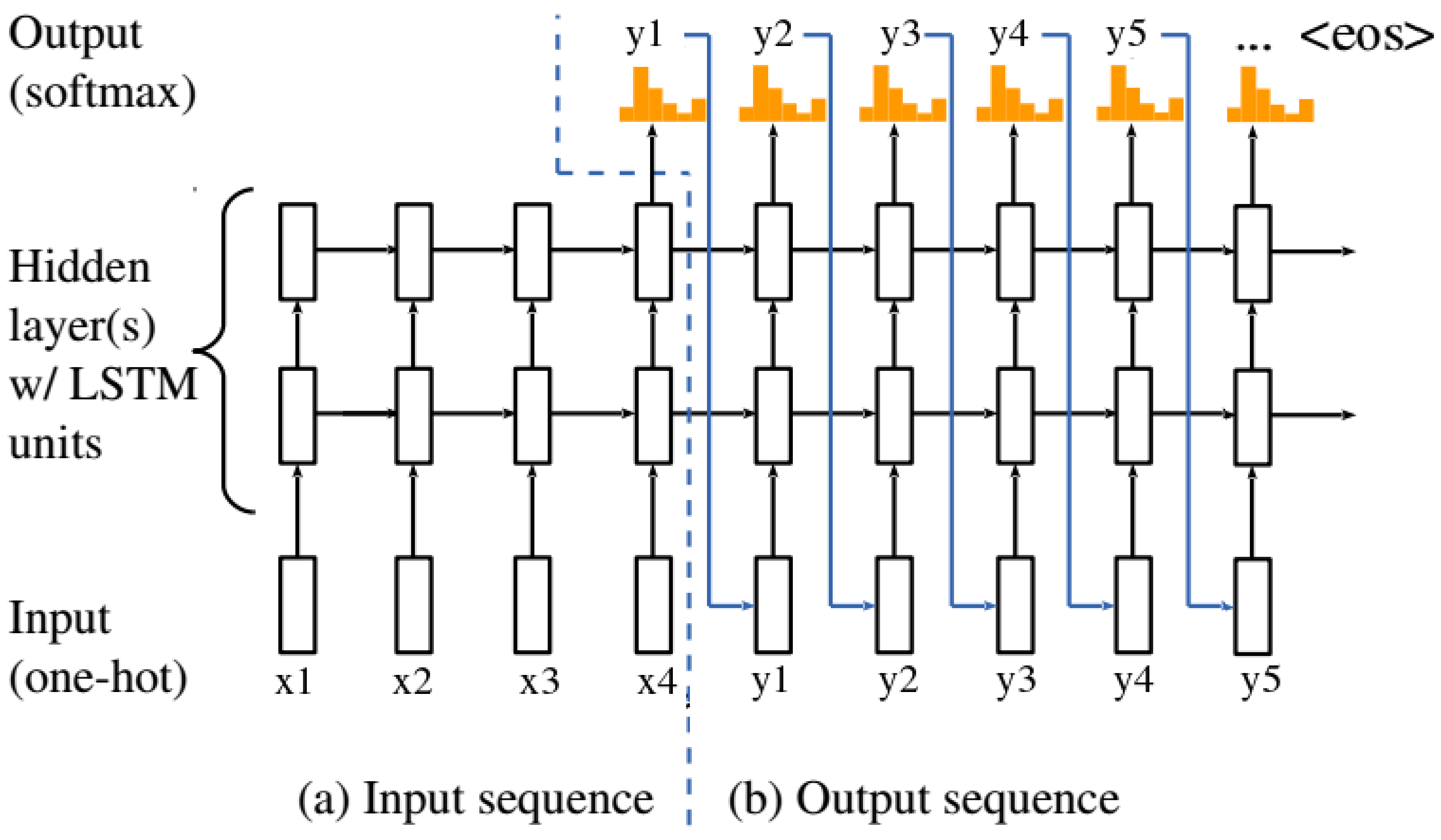
- Full softmax vs. NCE vs. HS vs. hinge loss?

Outline

- **Preliminary**
  - Word embeddings
  - **Sequence-to-sequence generation**
- Indifferentiability, solutions, and applications
- A case study in semantic parsing

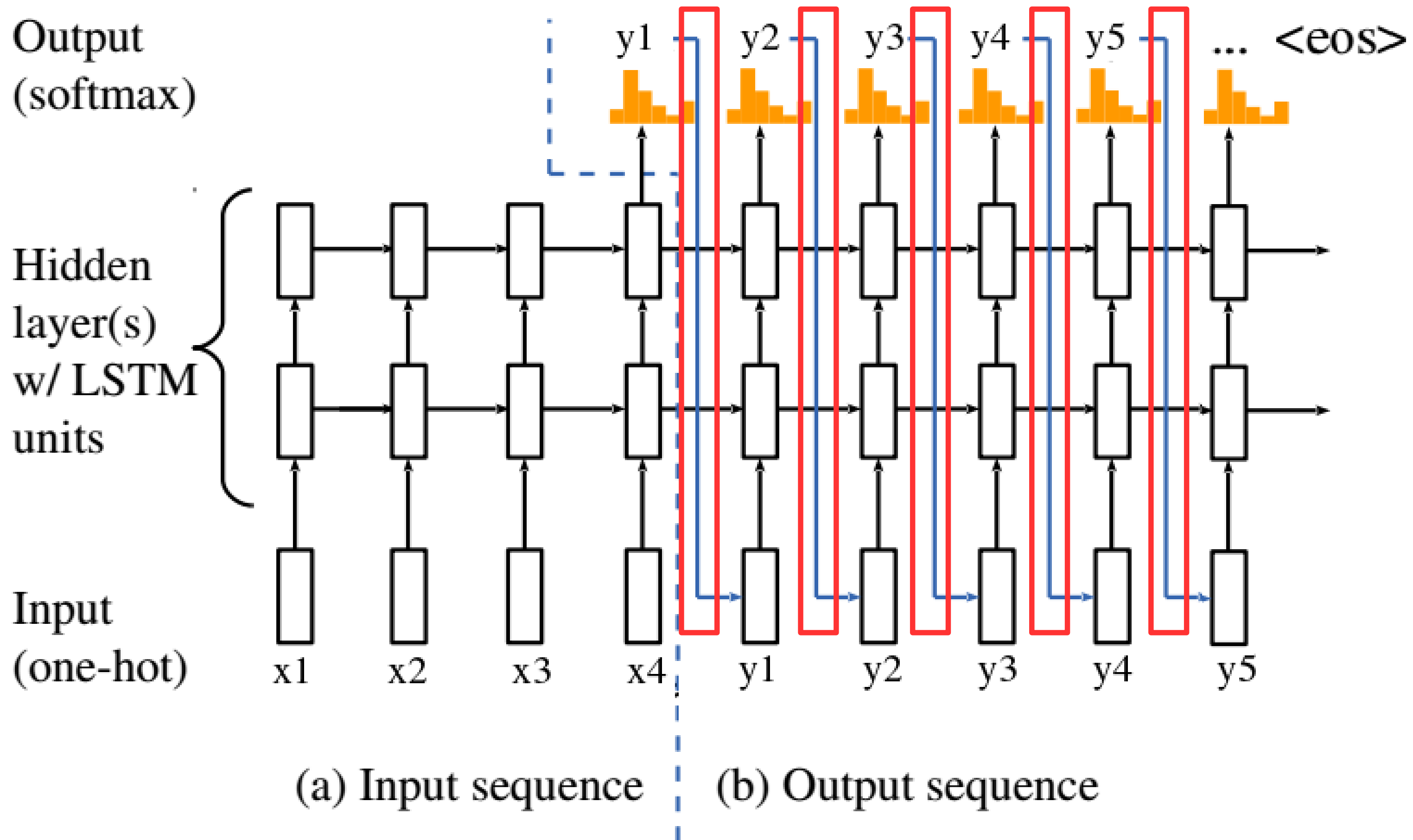
Seq2Seq Networks

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." NIPS. 2014

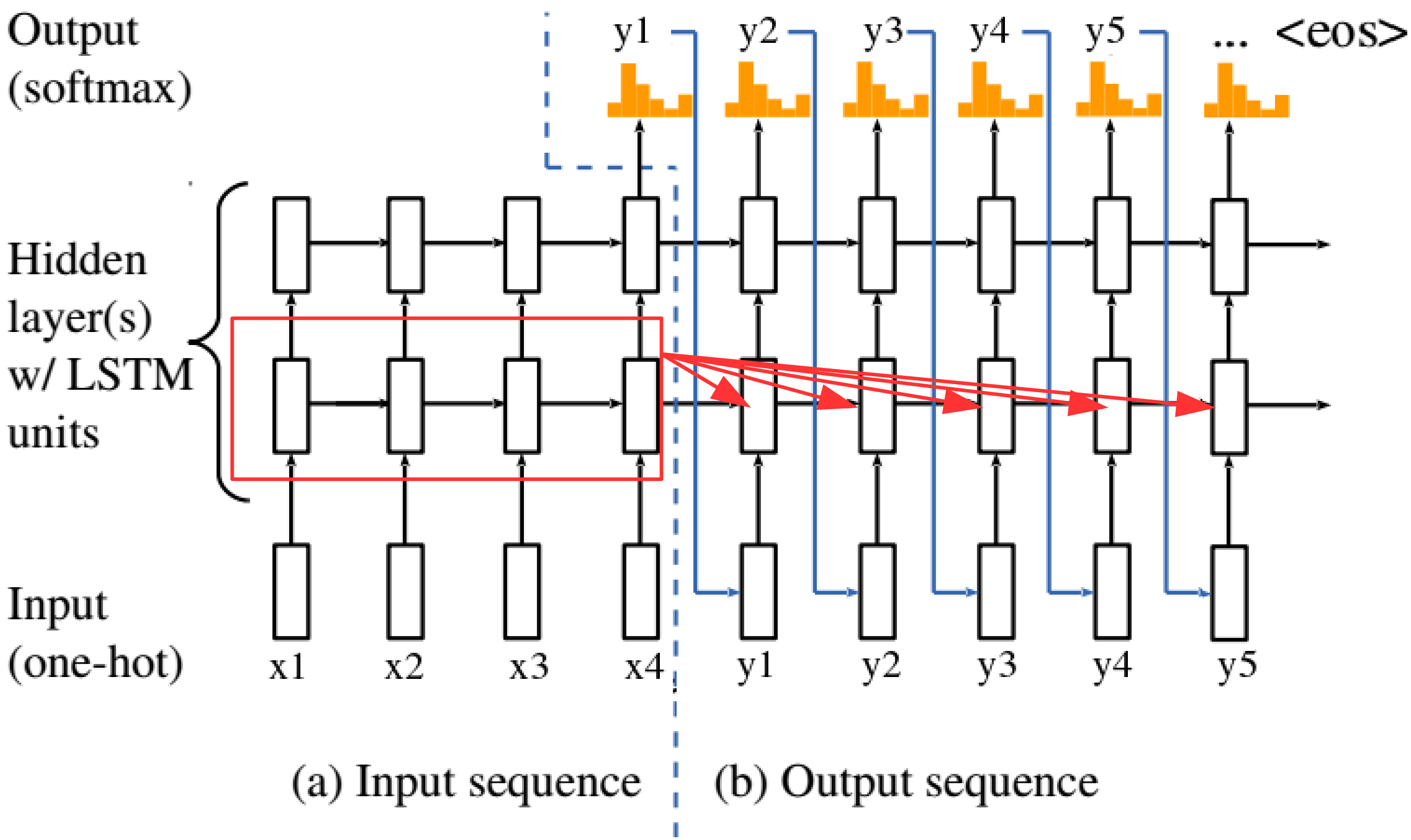


Feeding back the output

- Mode selection
- Potential chaotic behaviors



Attention Mechanisms



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR, 2014.

Context Vector

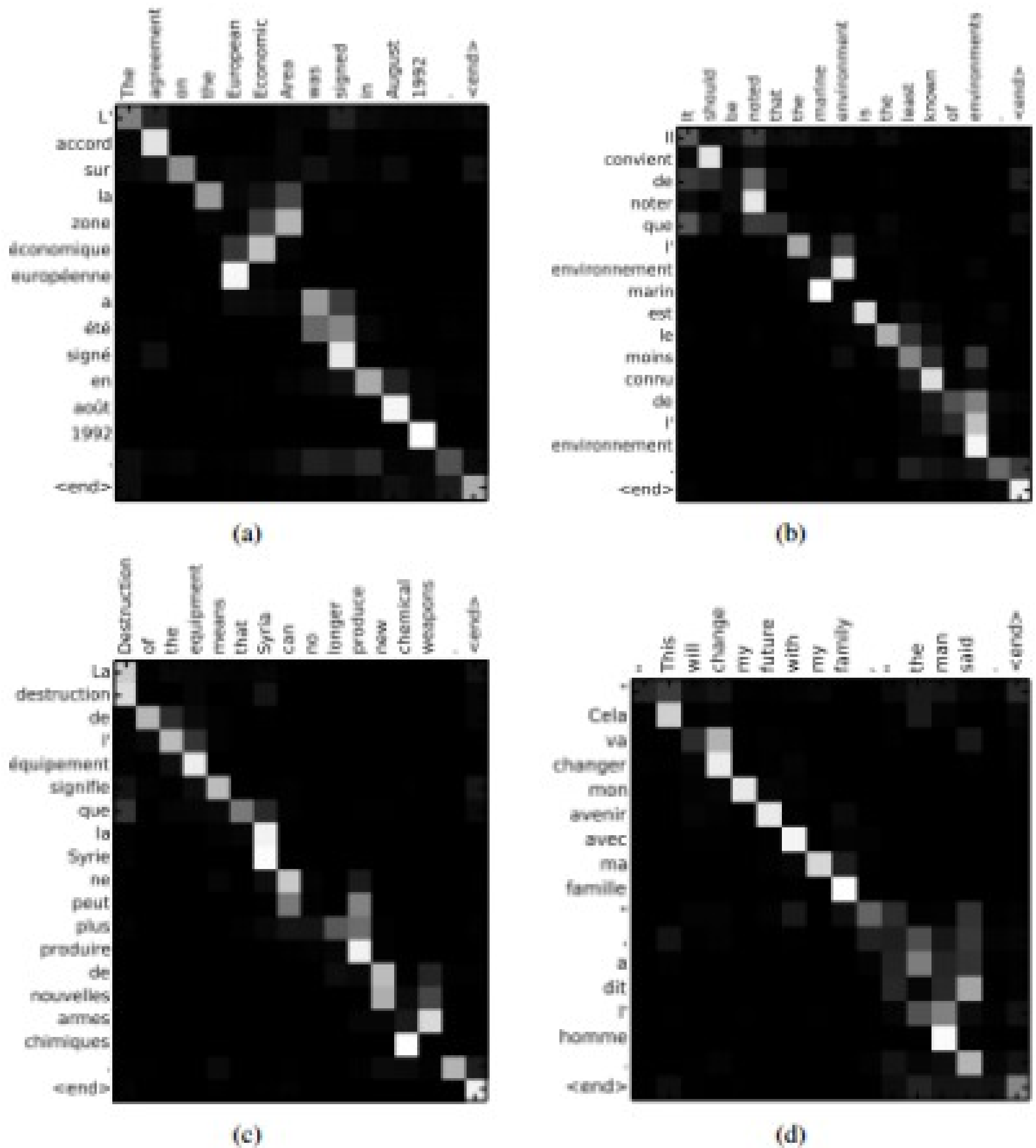
The context vector $\boldsymbol{c}$ is a combination of the input sequence's states $$\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{c}_i$$

where the coefficient α_i is related to

- The local context $\boldsymbol{c}_i$, and
- The last output state $\boldsymbol{h}_{t-1}$
- α_i is normalized

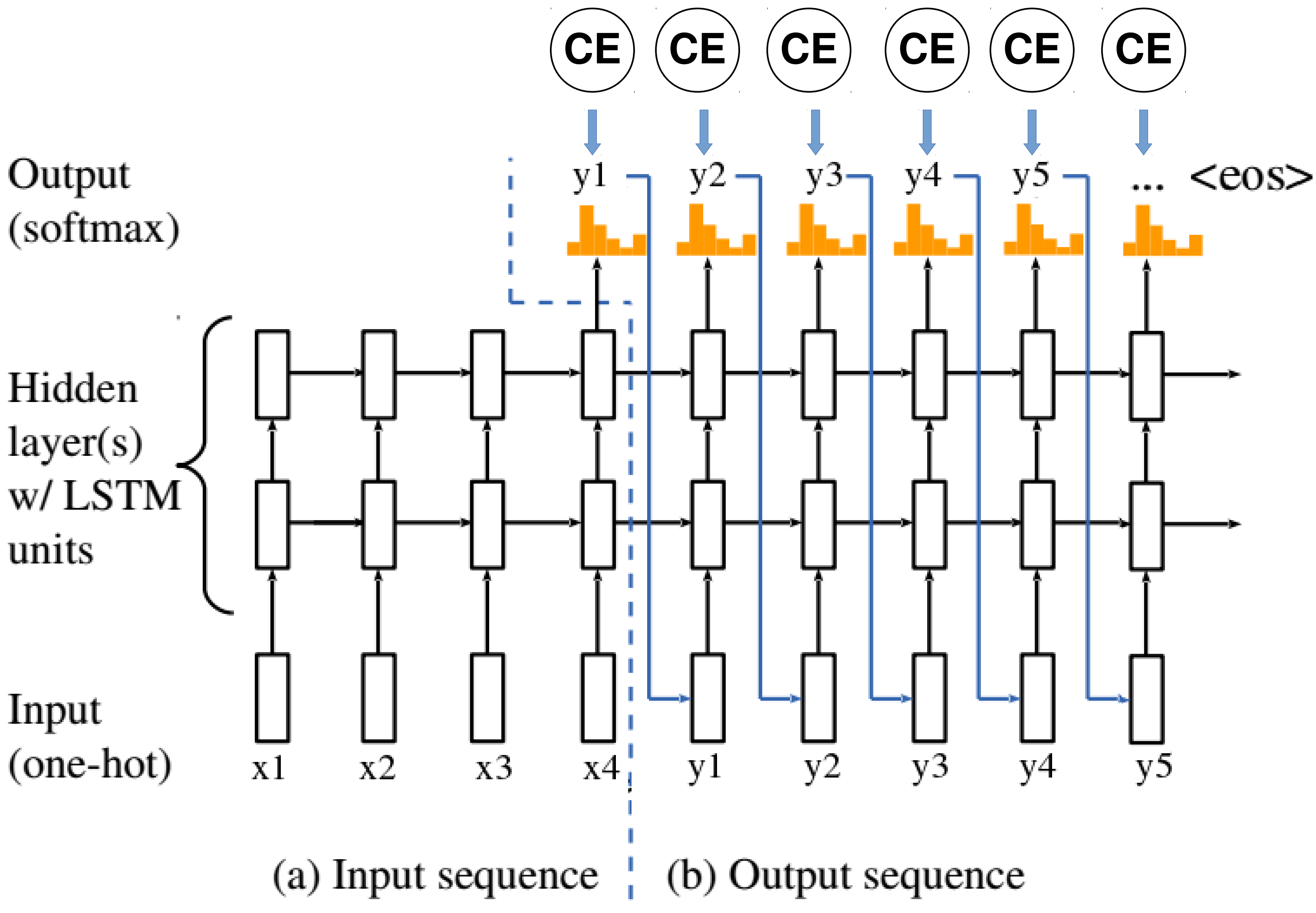
$$\alpha_i = \frac{\exp\{\tilde{\alpha}_i\}}{\sum_j \exp\{\tilde{\alpha}_j\}} \qquad \tilde{\alpha}_i = W[\boldsymbol{h}_{t-1}; \boldsymbol{c}_i]$$

Attention as Alignment



Training

- Step-by-step training



Indifferentiability