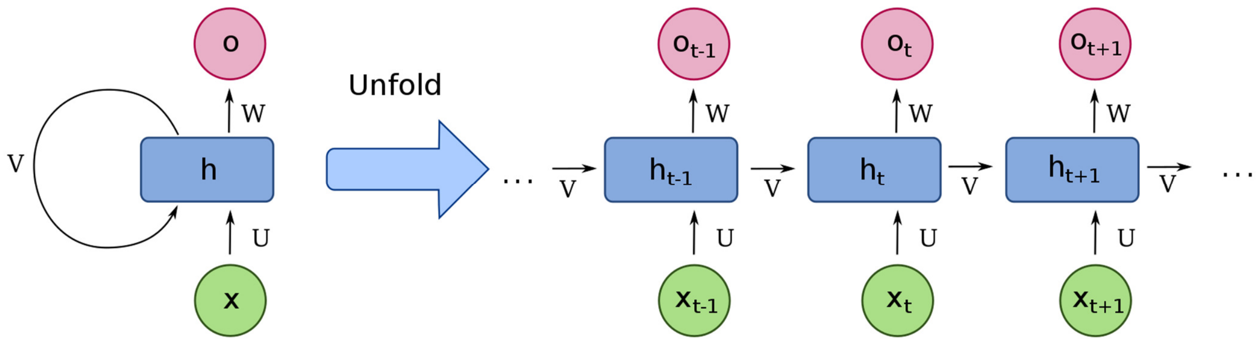
DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Redes recurrentes [RNNs]

Fernando Berzal, berzal@acm.org

Redes recurrentes

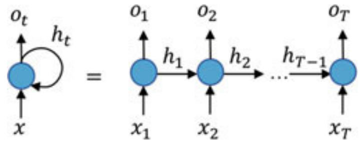


$$\mathbf{h}_t = \mathbf{o}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$$

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

Redes recurrentes

Redes recurrentes simples



- **Redes de Elman**
  Jeffrey L. Elman (1990): "Finding Structure in Time". *Cognitive Science*. 14(2):179–211.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

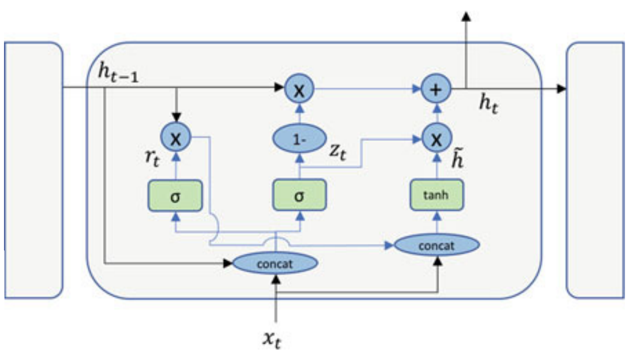
- **Redes de Jordan**
  Michael I. Jordan (1986): "Serial order: A parallel distributed processing approach", Technical Report 8604, Institute for Cognitive Science, UCSD

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

Redes recurrentes

GRU [Gated Recurrent Unit]

Kyunghyun Cho et al. (2014): "Learning phrase representations using RNN encoder-decoder for statistical machine translation". *arXiv:1406.1078* & EMNLP'2014

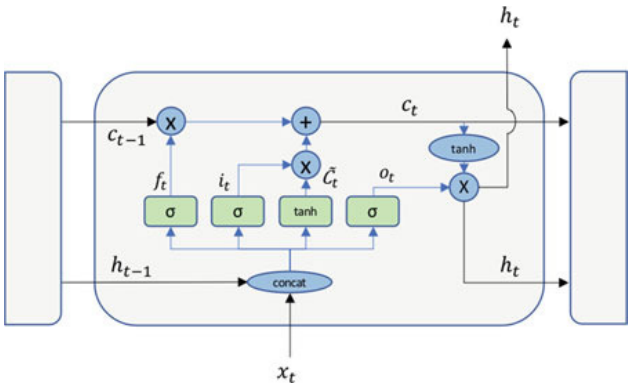


$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1})$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1})$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} \circ \mathbf{r}_t)$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t + \mathbf{z}_t * \mathbf{h}_{t-1}$$

Redes recurrentes

LSTM [Long Short-Term Memory]

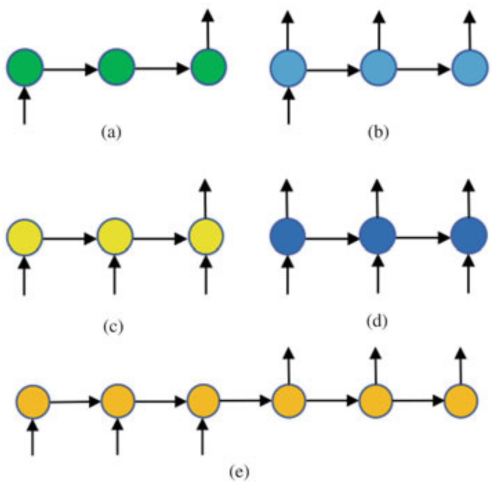
Sepp Hochreiter & Jürgen Schmidhuber (1997): "Long short-term memory". Neural Computation



$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1})$$
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

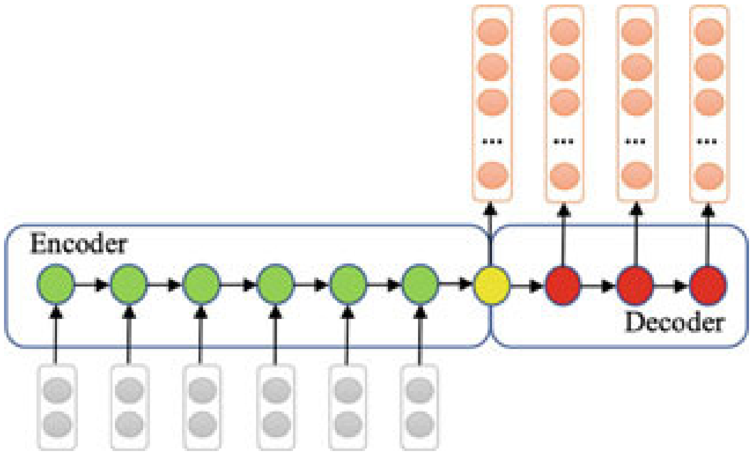
Redes recurrentes

Aplicaciones: Procesamiento de secuencias



Redes recurrentes

Aplicaciones: seq2seq



Entrenamiento

BPTT [Backpropagation through time]

- Secuencias de longitud fija [padding/truncation]

  e.g. Keras, TensorFlow

- Secuencias de longitud variable

  e.g. PyTorch, Chainer

Entrenamiento

Gradient clipping

- Norma L2

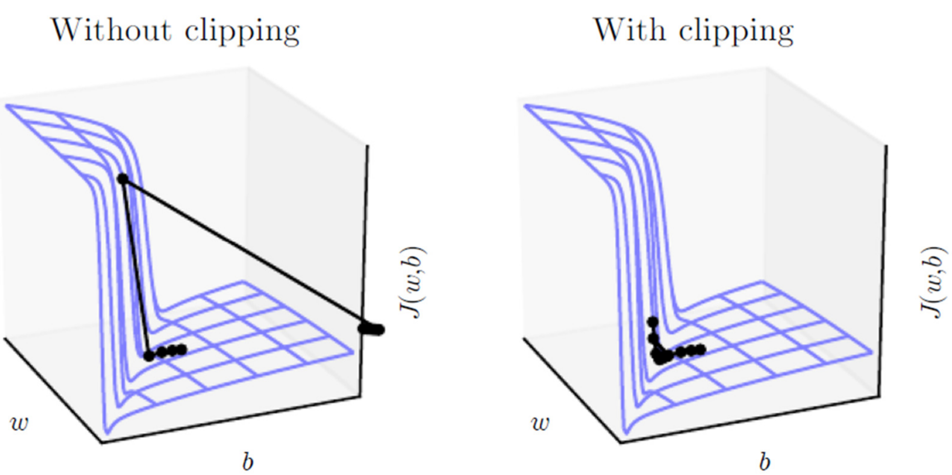
$$\nabla_{\text{new}} = \nabla_{\text{current}} \circ \frac{t}{L_2(\nabla)}$$

- Rango fijo

$$\nabla_{\text{new}} = \begin{cases} t_{\min} & \text{if } \nabla < t_{\min} \\ \nabla & \\ t_{\max} & \text{if } \nabla > t_{\max} \end{cases}$$

Entrenamiento

Gradient clipping



Entrenamiento

Técnicas de regularización

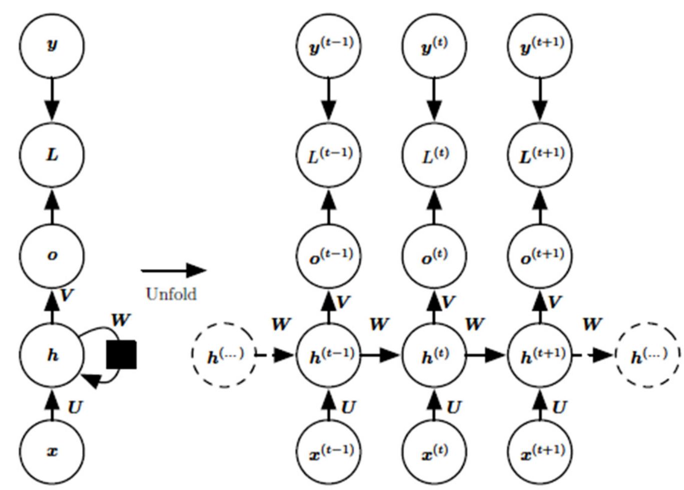
- Recurrent dropout
  Stanislau Semeniuta, Aliaksei Severyn & Erhardt Barth (2016): "Recurrent Dropout without Memory Loss", arXiv:1603.05118

- Variational dropout
  Yarin Gal & Zoubin Ghahramani (2016): "A theoretically grounded application of dropout in recurrent neural networks", NIPS'2016.

- Zoneout
  David Krueger et al. (2016): "Zoneout: Regularizing RNNs by randomly preserving hidden activations", arXiv:1606.01305

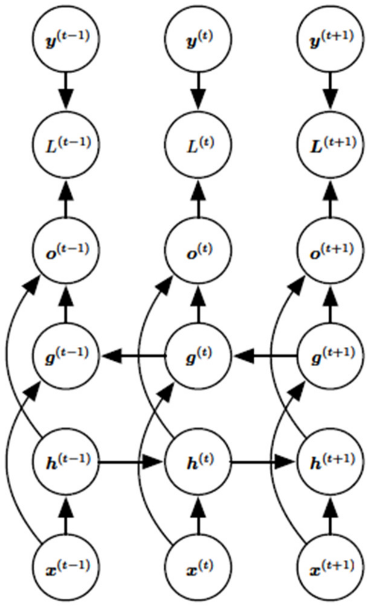
Redes recurrentes

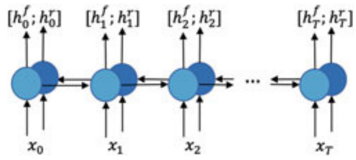
Grafo de cómputo



Redes recurrentes

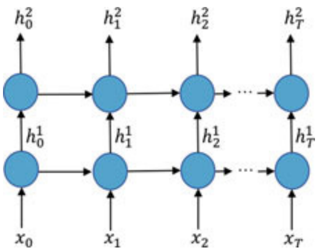
Redes bidireccionales

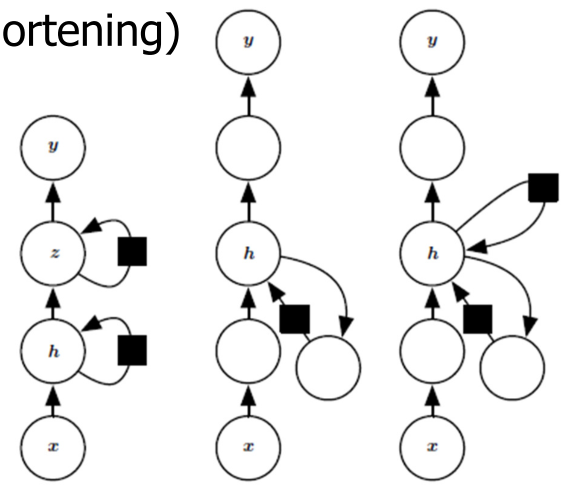




Deep RNNs

- **Stacked RNNs** (hierarchical)
- **Deep Transitions** (deeper computation)
- **Skip connections** (path shortening)

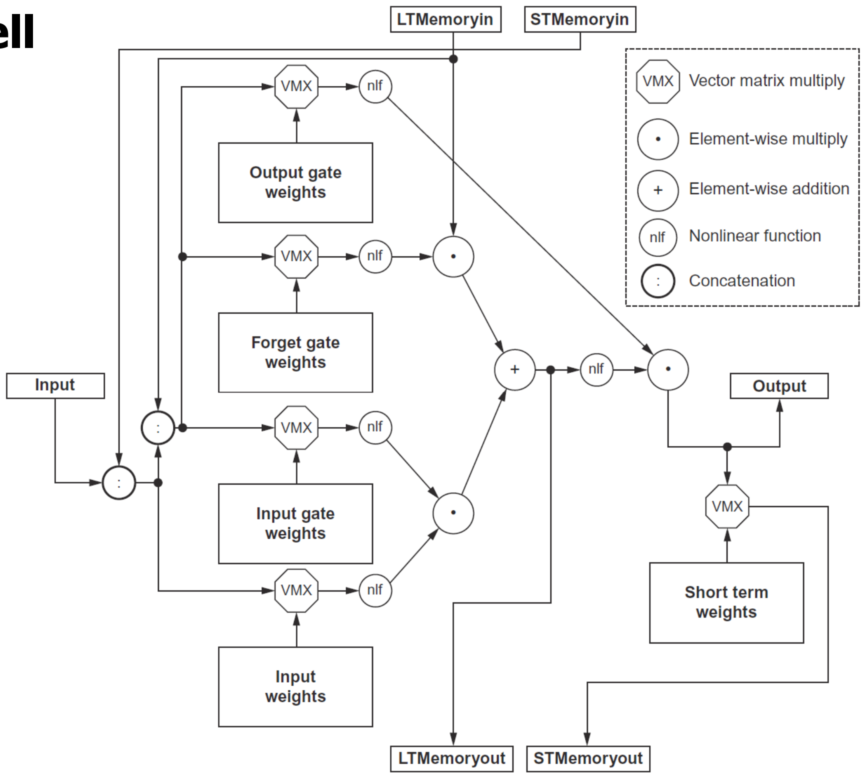




Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio: "How to Construct Deep Recurrent Neural Networks", ICLR'2014

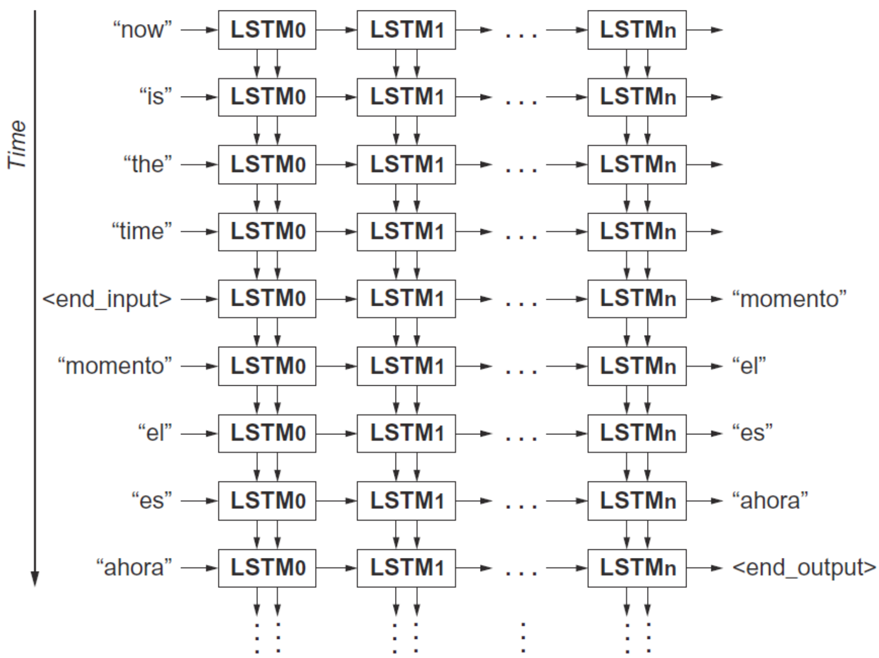
LSTM

LSTM cell



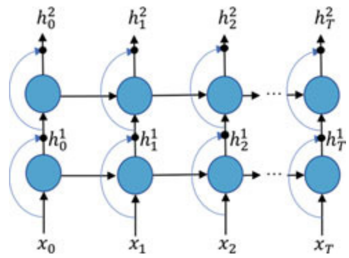
LSTM

Traductor neuronal basado en LSTM



Residual LSTM

Aaditya Prakash et al. (2016):
“Neural Paraphrase Generation with Stacked Residual LSTM Networks”.
arXiv:1610.03098

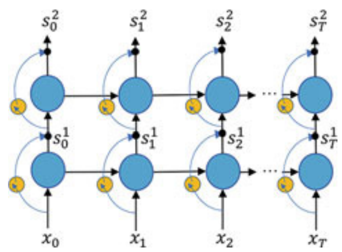


$$\mathbf{h}_t = \mathbf{o}_t \cdot (\mathbf{W}_p \cdot \tanh(\mathbf{c}_t) + \mathbf{W}_h \mathbf{x}_t)$$

Recurrent highway networks

RHN

Julian G. Zilly et al. (2016):
“Recurrent Highway Networks”.
arXiv:1607.03474



$$\mathbf{s}_t^{(l)} = \mathbf{h}_t^{(l)} \cdot \mathbf{t}_t^{(l)} + \mathbf{s}_t^{(l-1)} \cdot \mathbf{c}_t^{(l)}$$

$$\mathbf{h}_t^{(l)} = \tanh\left(\mathbf{W}_H \mathbf{x}_t \mathbb{1}_{\{l=1\}} + \mathbf{R}_{H^l} \mathbf{s}_t^{(l-1)} + \mathbf{b}_{H^l}\right)$$

$$\mathbf{t}_t^{(l)} = \sigma\left(\mathbf{W}_T \mathbf{x}_t \mathbb{1}_{\{l=1\}} + \mathbf{R}_{T^l} \mathbf{s}_t^{(l-1)} + \mathbf{b}_{T^l}\right)$$

$$\mathbf{c}_t^{(l)} = \sigma\left(\mathbf{W}_C \mathbf{x}_t \mathbb{1}_{\{l=1\}} + \mathbf{R}_{C^l} \mathbf{s}_t^{(l-1)} + \mathbf{b}_{C^l}\right)$$

Más variantes

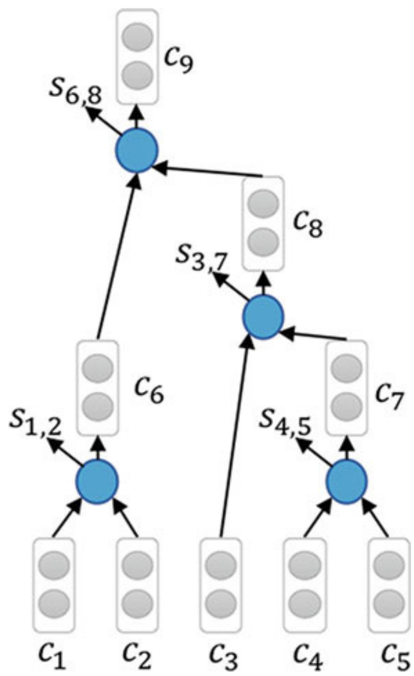
Optimizaciones para mejorar su eficiencia (eliminando dependencias secuenciales):

- SRU [Semi-Recurrent Unit]
  Tao Lei, Yu Zhang & Yoav Artzi (2017):
  "Training RNNs as Fast as CNNs", arXiv:1709.02755

- QRNN [Quasi-Recurrent Neural Network]
  James Bradbury et al. (2016):
  "Quasi-Recurrent Neural Networks". arXiv:1611.01576

RecNN: Redes recursivas

$$s_{ij} = \mathbf{U}\dot{p}(\mathbf{c}_i,\mathbf{c}_j)$$
$$p(\mathbf{c}_i,\mathbf{c}_j) = f(W[\mathbf{c}_i;\mathbf{c}_j]+\mathbf{b})$$

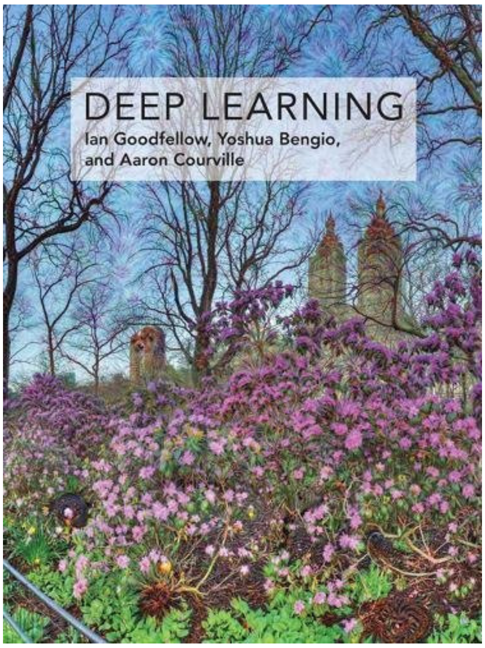


Christoph Goller & Andreas Kuchler:
"Learning task-dependent distributed representations by backpropagation through structure". ICNN'1996

Bibliografía

Lecturas recomendadas

Ian Goodfellow,
Yoshua Bengio
& Aaron Courville:
Deep Learning
MIT Press, 2016
ISBN 0262035618

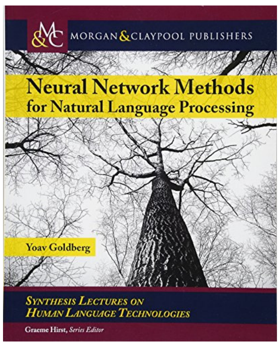


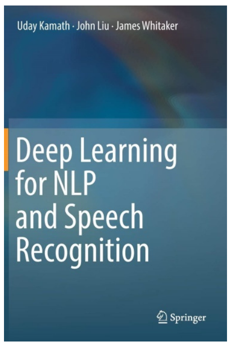
http://www.deeplearningbook.org

Bibliografía

Procesamiento del Lenguaje Natural

NLP





- Yoav Goldberg:
  Neural Network Methods in Natural Language Processing
  Morgan & Claypool Publishers, 2017
  ISBN 1627052984
  https://doi.org/10.2200/S00762ED1V01Y201703HLT037

- Uday Kamath, John Liu & James Whitaker:
  Deep Learning for NLP and Speech Recognition
  Springer, 2019
  ISBN 3030145956
  http://link.springer.com/978-3-030-14595-8