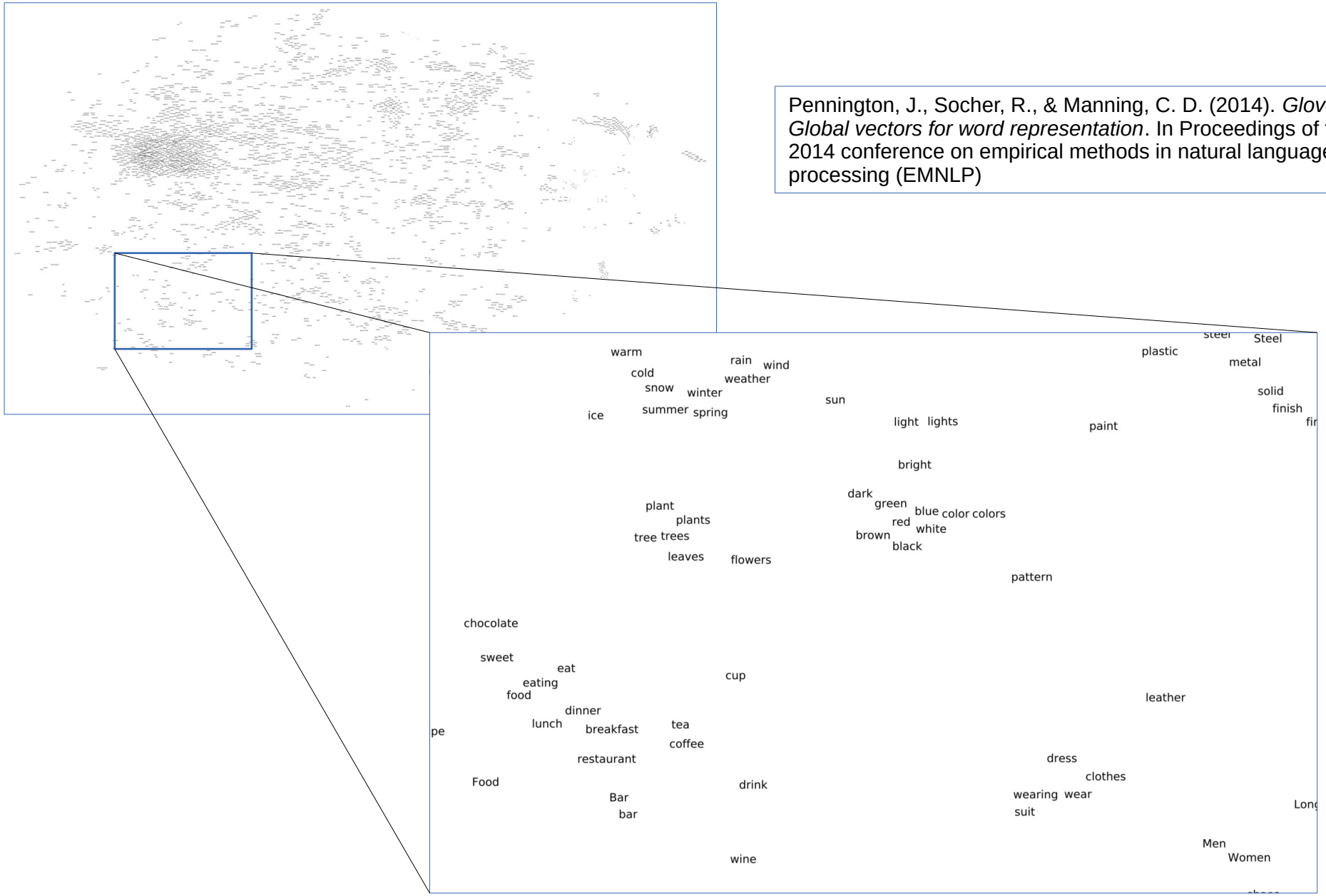
Recurrent Neural Networks and NLP

Nathan Sprague

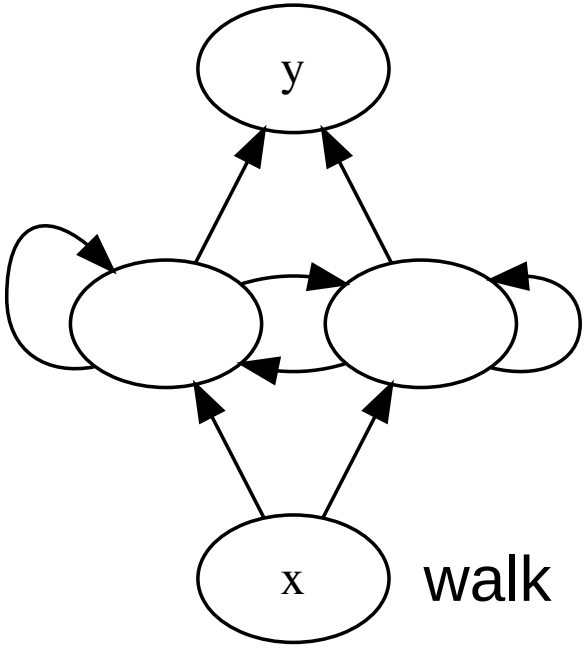
Word Embeddings



Pennington, J., Socher, R., & Manning, C. D. (2014). *Glove: Global vectors for word representation*. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)

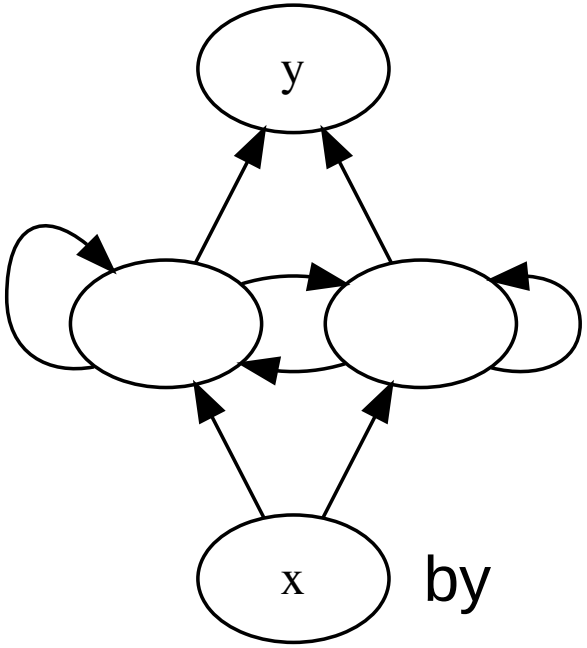
https://home.ttic.edu/~kgimpel/wordembviz/wordembviz.html

Recurrent Neural Networks



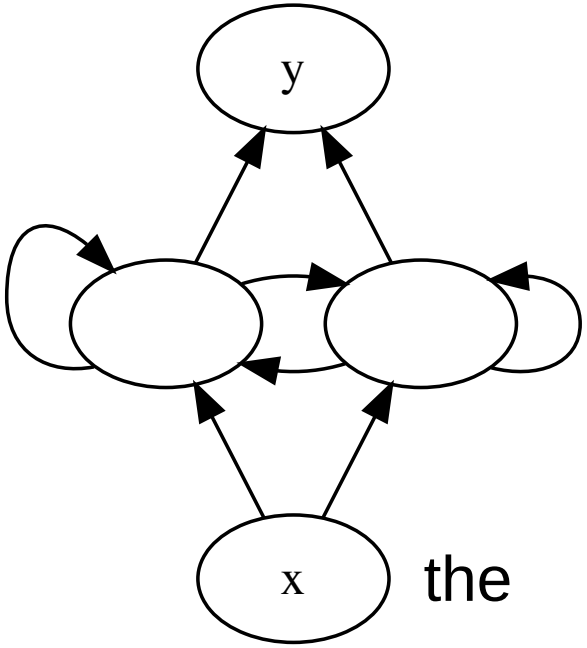
walk by the river

Recurrent Neural Networks

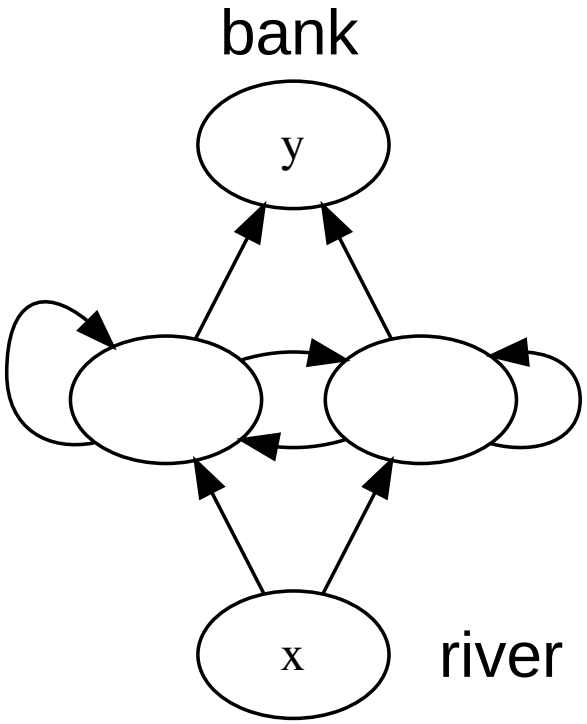


by the river

Recurrent Neural Networks

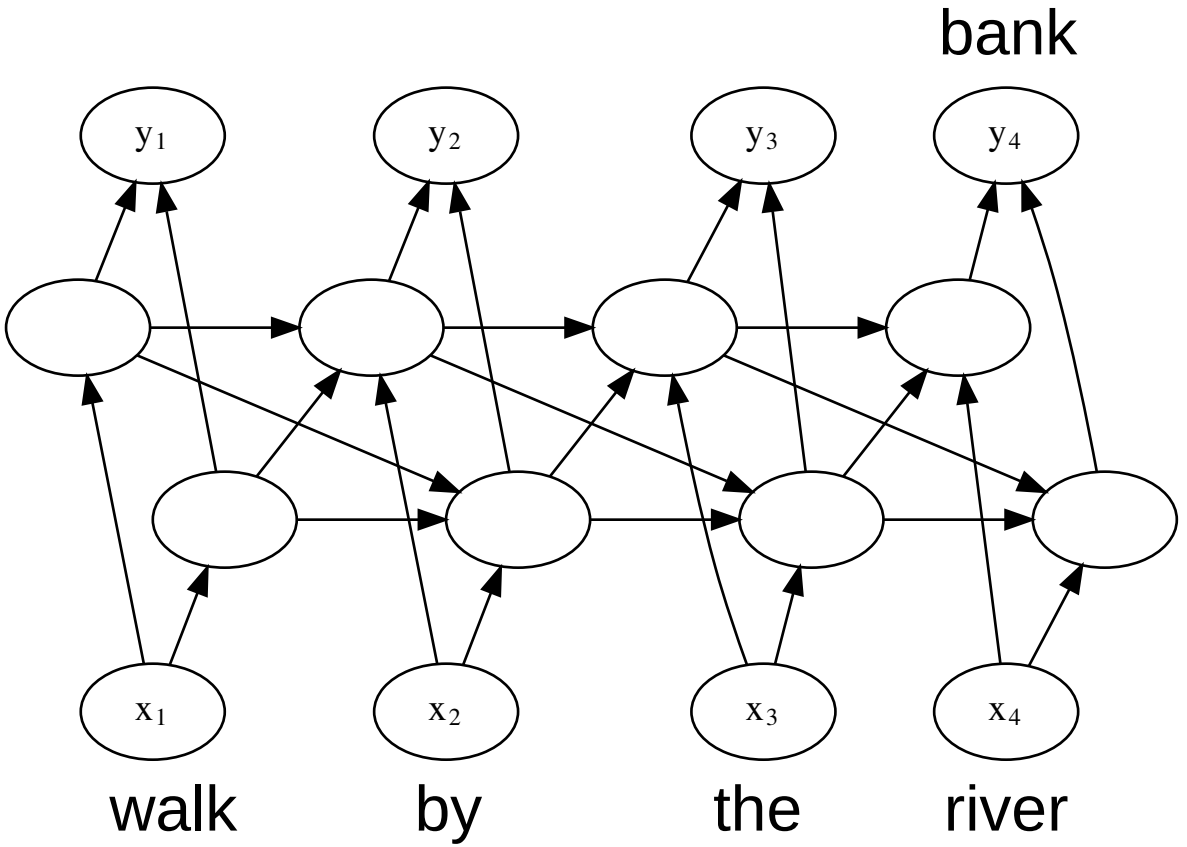


Recurrent Neural Networks



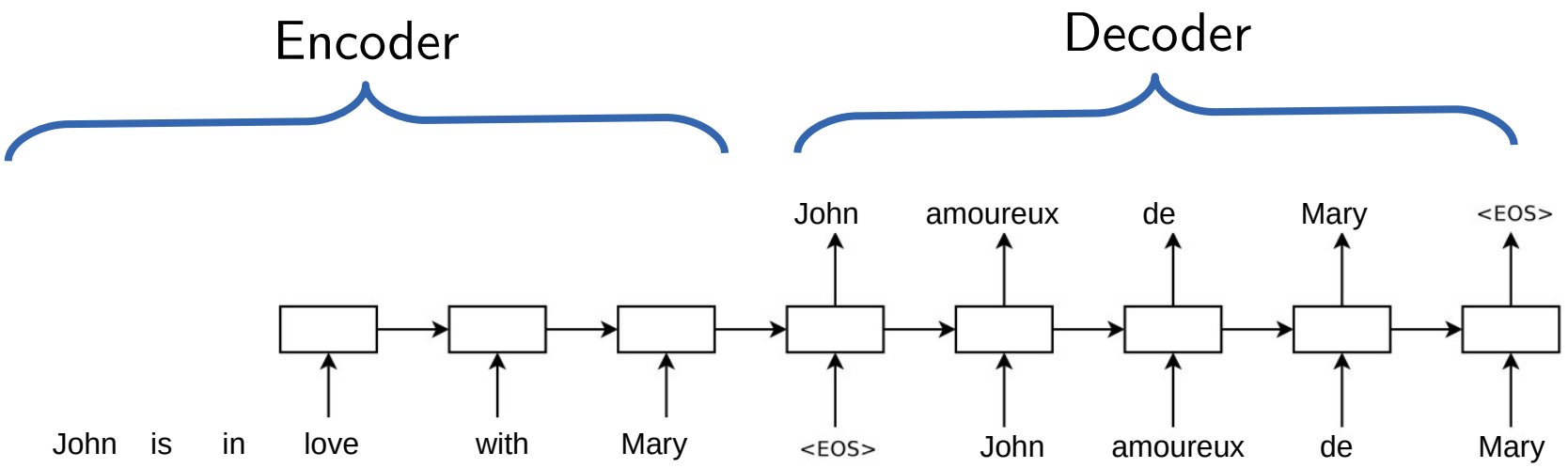
Backpropagation in RNN's

- We can still do backpropagation by unrolling the network in time:



Translation: Encoder/Decoder Models

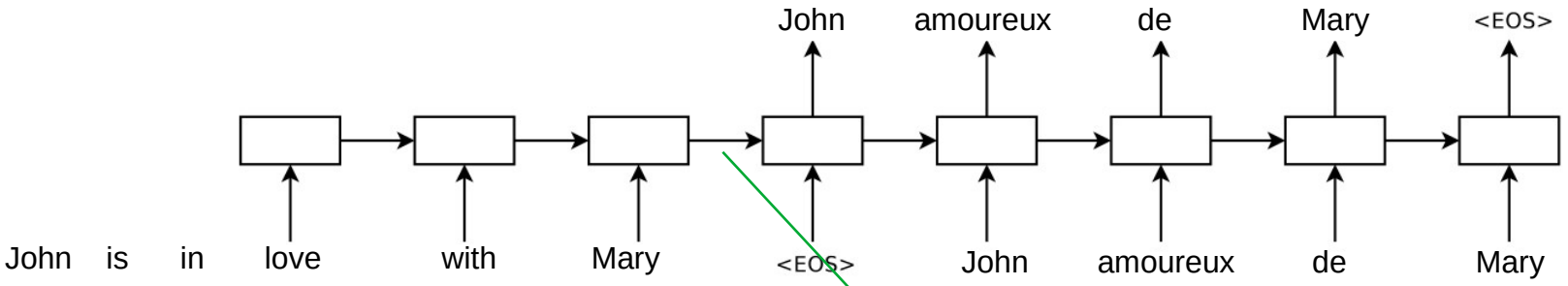
- Original encoder/decoder architecture:

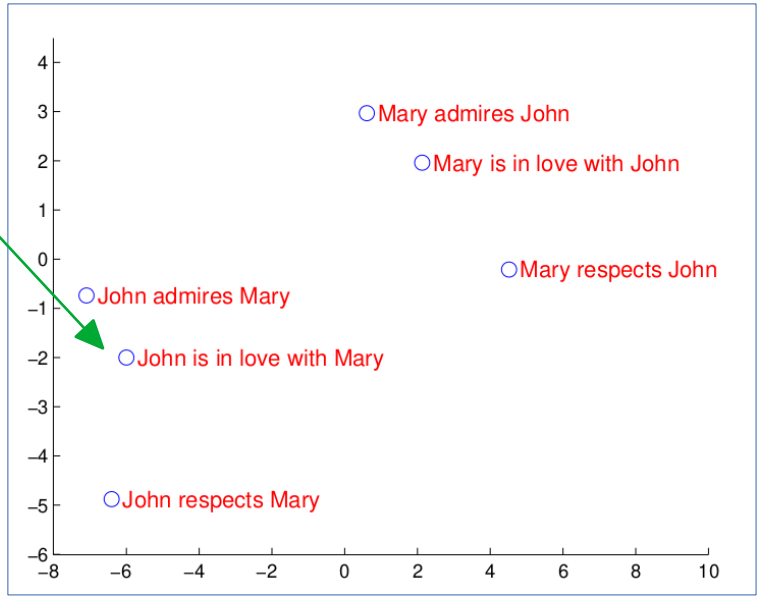


I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to Sequence Learning with Neural Networks*, in Advances in Neural Information Processing Systems 27, 2014, pp. 3104–3112.

Translation: Encoder/Decoder Models

- Original encoder/decoder architecture:



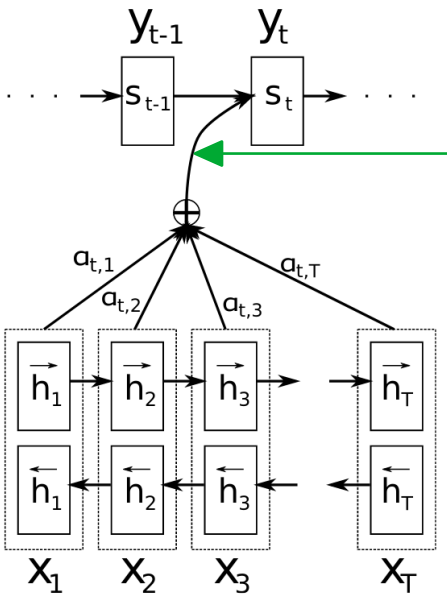


I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to Sequence Learning with Neural Networks*, in Advances in Neural Information Processing Systems 27, 2014, pp. 3104–3112.

Attention

Decoder

Encoder (Bi-directional LSTM)



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

Decoder receives weighted sum of *all* annotations

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Weights are calculated using "softmax" over the output of an alignment model

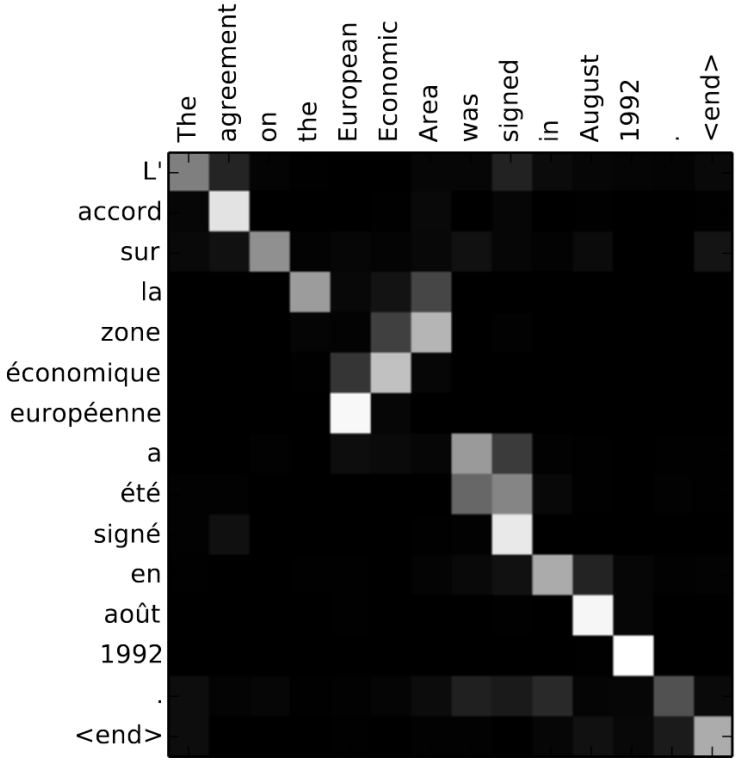
$$e_{ij} = a(s_{i-1}, h_j)$$

Alignment model is a simple feedforward network

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015, January). Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015.

Attention: Alignment Example

- English to French translation
- Each “pixel” shows the corresponding α_{ij}



Bahdanau, D., Cho, K. H., & Bengio, Y. (2015, January). Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015.

LSTM Networks

- Problem: a plain RNN must carry information across many time steps
  - Each step mixes in the new input: multiply, add, squash
  - Old information washes out (and so do the gradients)
- LSTM: add a *cell state* that is changed only by gated, element-wise updates
  - Information can persist across many steps
- Learned gates decide what to forget, write, and output
- GRU: a simpler variant with no separate cell state

S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation 9(8), 1997, pp. 1735–1780.

Understanding LSTMs

- Christopher Olah's walkthrough, with diagrams:
  - colah.github.io: Understanding LSTM Networks
- In PyTorch, `nn.LSTM` and `nn.GRU` are near drop-in replacements for `nn.RNN`
  - Exception: `nn.LSTM`'s hidden state is a pair, (h, c)