Neural Networks and Deep Learning

Recurrent Neural Network II

Shumin Wu

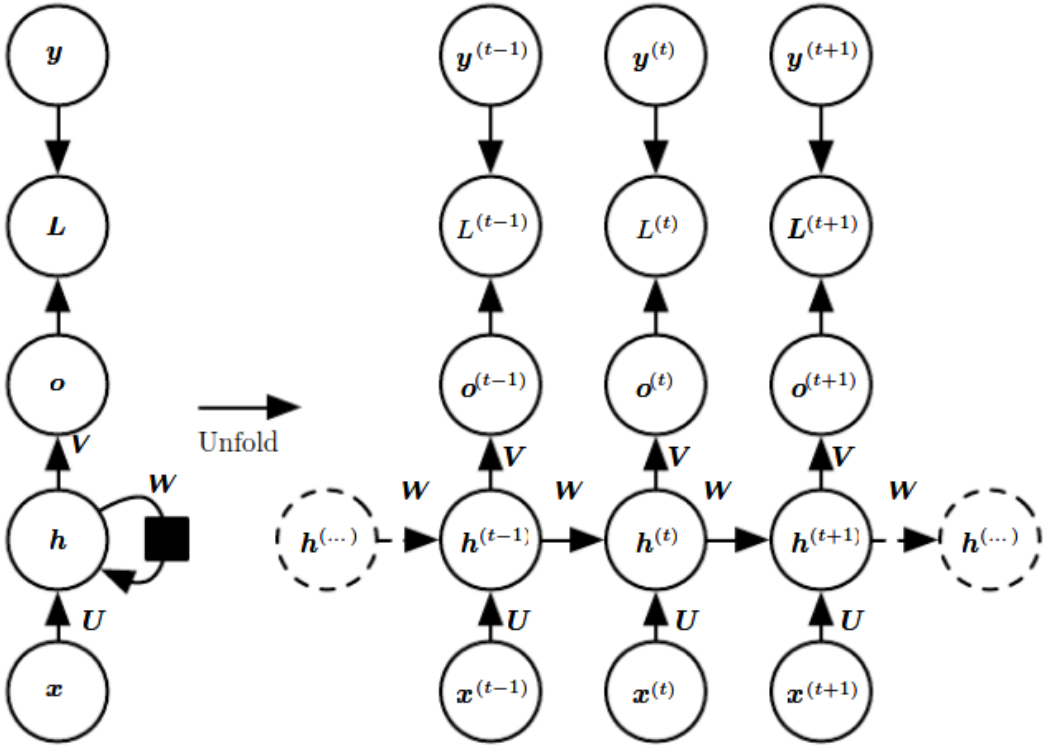
Department of Computer Science

shumin.wu@colorado.edu

February 24, 2020

Previous Lecture: RNN

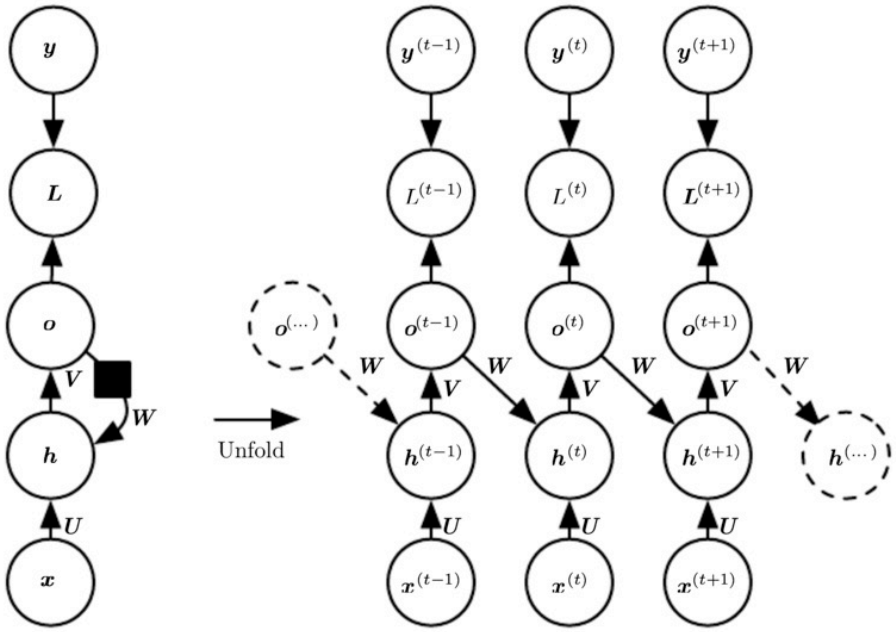
$$\alpha^t = Ux^t + Wh^{t-1} + b_\alpha$$
$$h^t = \tanh(\alpha^t)$$
$$o^t = Vh^t + b_o$$
$$\hat{y}^t = \text{softmax}(o^t)$$



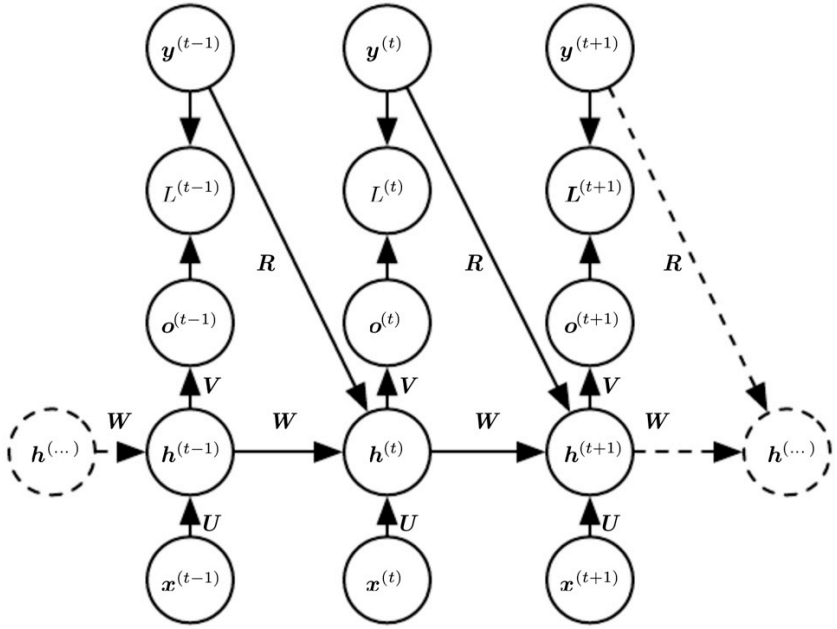
source: http://deeplearningbook.org

Previous Lecture: RNN

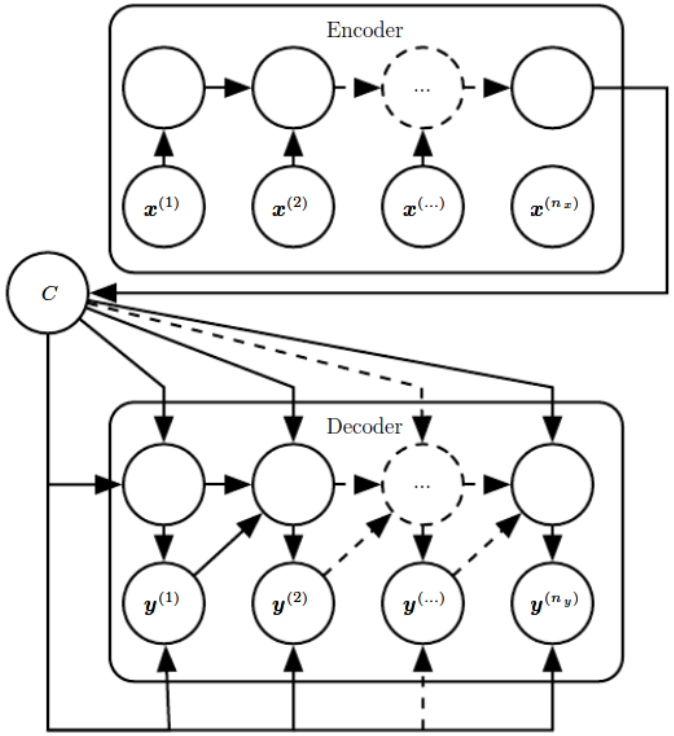
Output-to-hidden recurrence



Conditioned on previous outputs



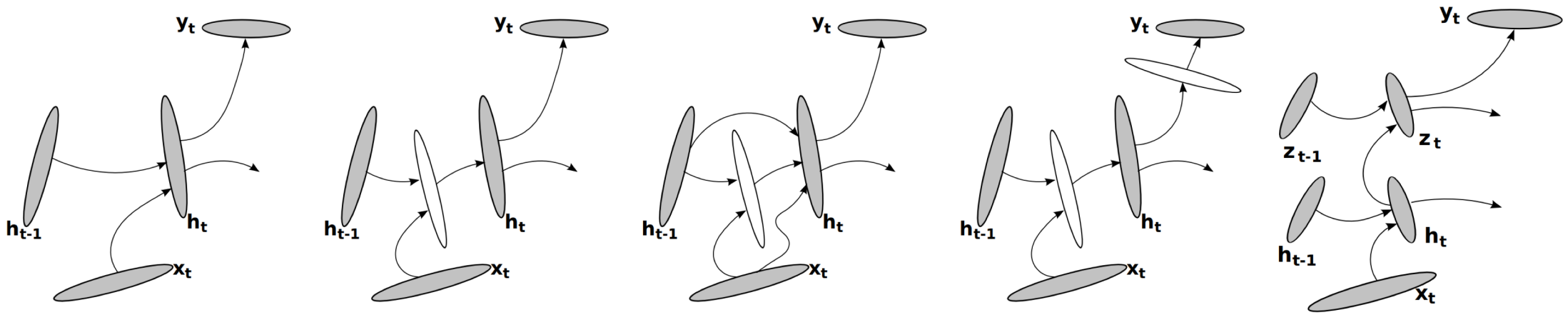
Encoder-decoder architecture



source: http://deeplearningbook.org

Deep RNN

1. Hidden recurrent state can be broken into groups organized hierarchically.
2. Each of U, V, W matrices can all be replaced with MLP.
3. Skip connections can be added to ease optimization.



(a) RNN (b) DT-RNN (b*) DT(S)-RNN (c) DOT-RNN (d) Stacked RNN

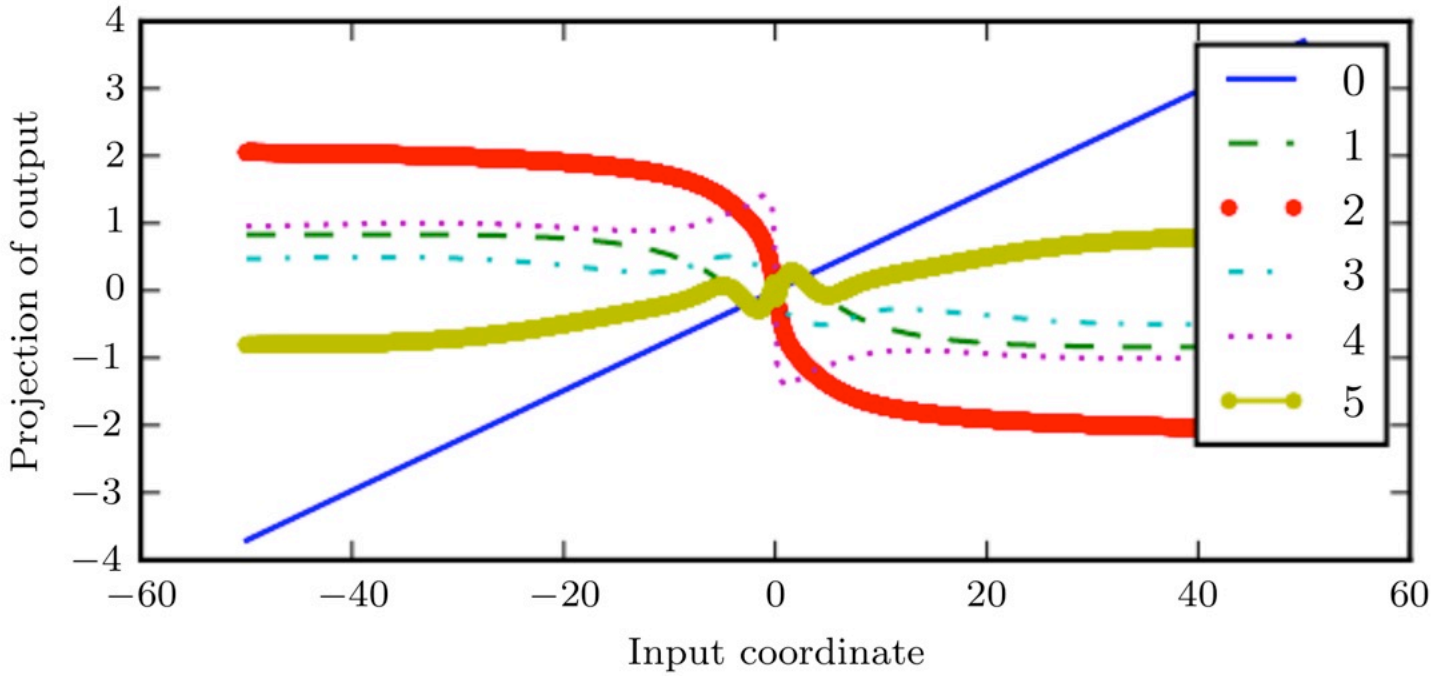
source: [Pascanu et al, 2014]

RNN Challenge of Long-Term Dependencies

"In practice, the experiments in Bengio et al. (1994) show that as we increase the span of the dependencies that need to be captured, gradient-based optimization becomes increasingly difficult, with the probability of successful training of a traditional RNN via SGD rapidly reaching 0 for sequences of only length 10 or 20."

Nonlinearities and Instability

Repeatedly applying a non-linear function (tanh with linear tails) results in many small derivatives, some large ones, and many changes in the sign of derivatives.



source: http://deeplearningbook.org

RNN Vanishing/Exploding Gradients

Consider the recurrence of an RNN without respect to the input or activation:

$$\mathbf{h}^{(t)} = \mathbf{W}^T \mathbf{h}^{(t-1)}$$

This reduces to:

$$\mathbf{h}^{(t)} = (\mathbf{W}^{(t)})^T \mathbf{h}^{(0)}$$

If $\mathbf{W}$ lends itself to eigen decomposition s.t. $\mathbf{W} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathbf{T}}$, with $\mathbf{Q}$ being orthogonal, then $\mathbf{h}^{(t)}$ can be written as:

$$\mathbf{h}^{(t)} = \mathbf{Q}^T \boldsymbol{\Lambda}^t \mathbf{Q} \mathbf{h}^{(0)}$$

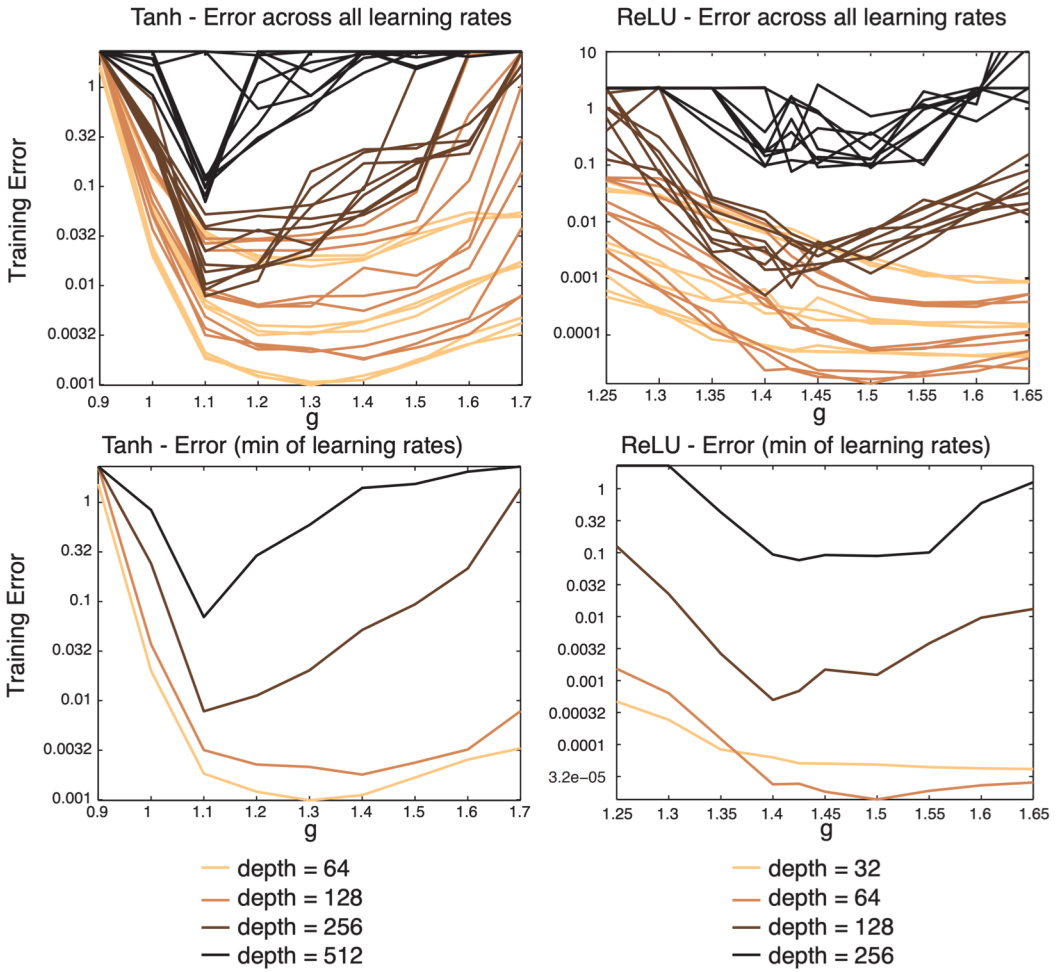
What is the implication of $\boldsymbol{\Lambda}^t$ for the training of RNNs?

Vanishing/Exploding Gradients in FFN

Consider d_{th} layer of a Feed-Forward Network with the form:

$$\mathbf{h}_d = f(g\mathbf{W}_d\mathbf{h}_{d-1} + b_d)$$

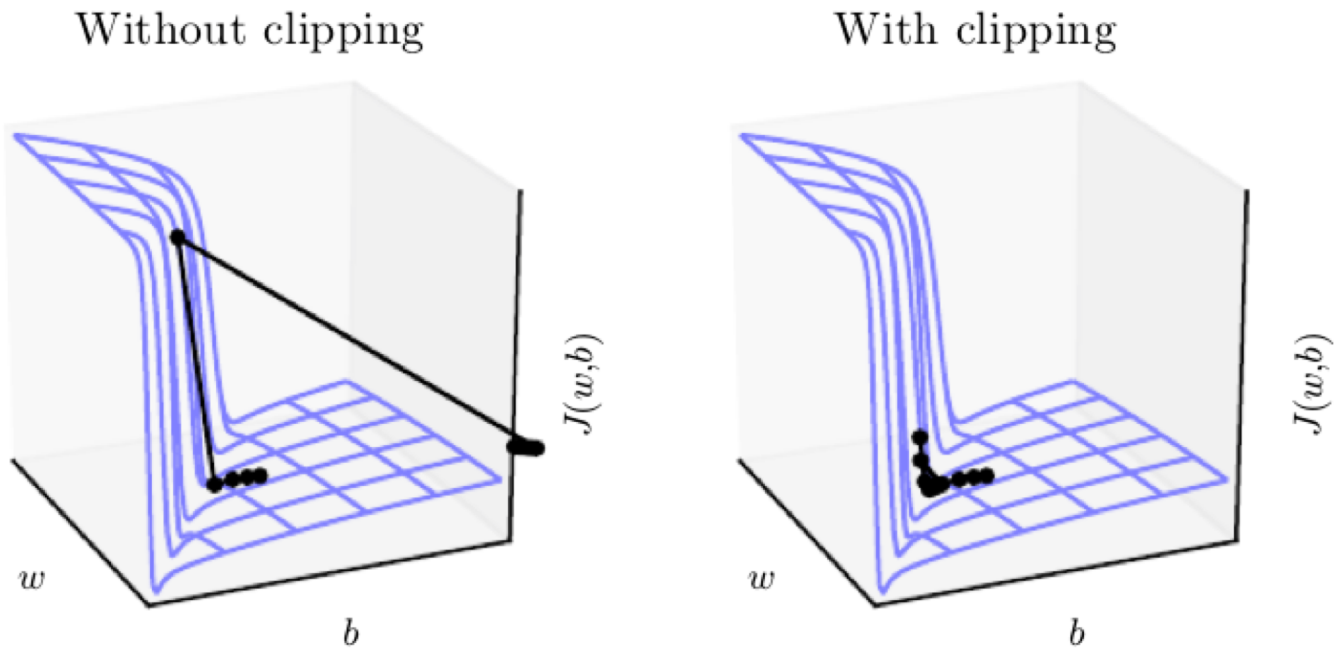
Where each $\mathbf{W}_i$ are i.i.d. with zero mean and unit variance. Then careful sampling of $\mathbf{W}_i$ can produce a network w/ neither vanishing nor exploding gradients.



source: [Sussillo and Abbott, 2014]

Gradient Clipping

$$\text{if } ||g|| > v \textbf{ do } g \leftarrow \frac{gv}{||g||}$$



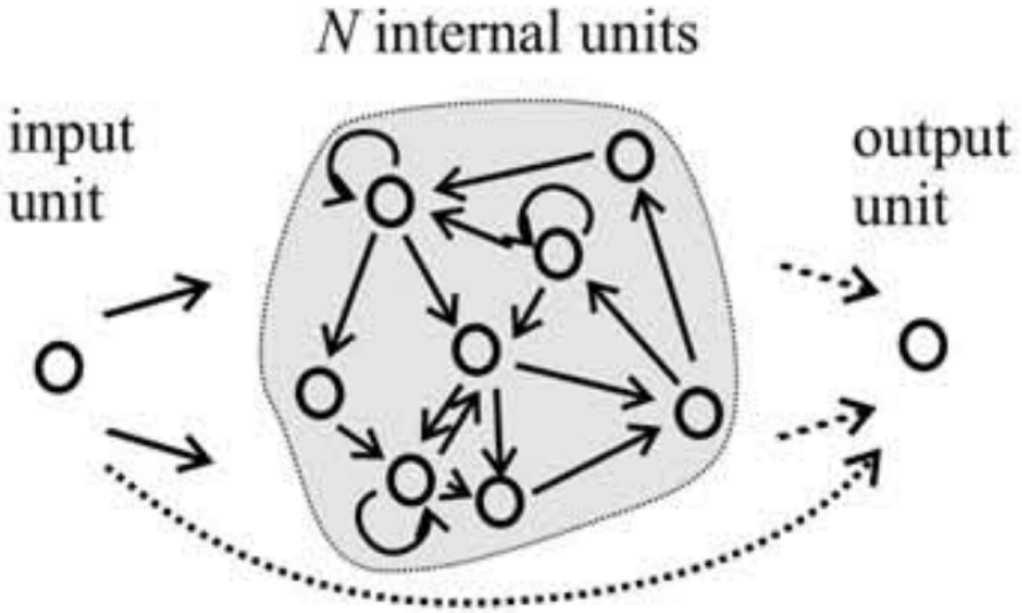
source: http://deeplearningbook.org

Echo State Network

Make $\mathbf{W}$ a hyperparameter of the model (akin to SVM). Pick $\mathbf{W}$ with a spectral radius (largest eigenvalue of $\mathbf{W}$) around 3 and train only the output layer.

Dotted lines are trainable paths.

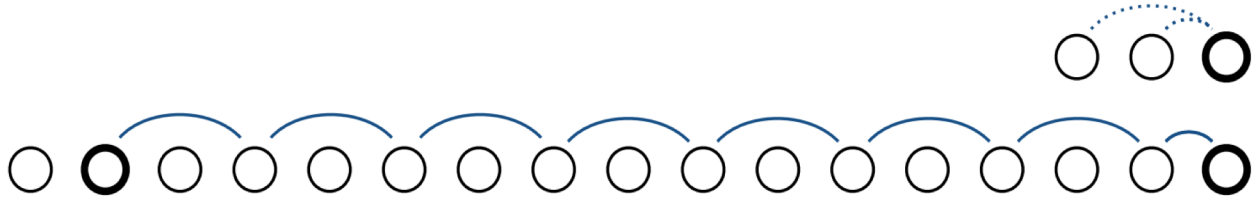
Also a good way to initialize trainable $\mathbf{W}$ (spectral radius ≈ 1.2)



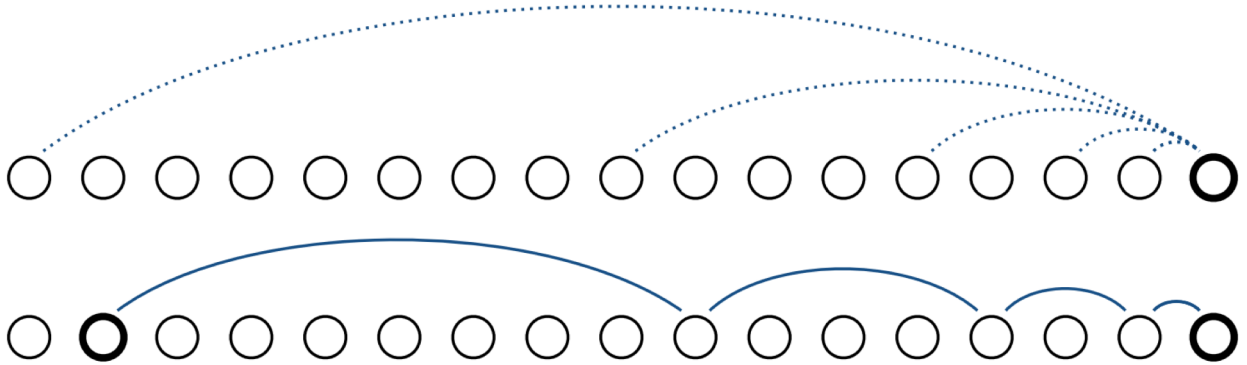
source: [Herbert Jaeger, 2003]

RNN Distant Connections

Addition/in lieu of constant step connections:



Addition/in lieu of exponential back-off step connections:



source: [DiPietro et al, 2018]

RNN Leaky Units

Objective: obtaining paths on which the product of derivatives is ≈1:

Design units with linear self-connections and a weight α near one on these connections.

$$h^t = \alpha h^{t-1} + (1 - \alpha)v^t$$

Can be seen as a moving running average h^t of some value v^t.

Leaky units – hidden units with linear self-connections – can behave similarly to moving averages. First proposed by M. C. Mozer, 1992 for music composition.

Gated RNN

Plain RNN:

$$h^t = \tanh(Ux^t + Wh^{t-1} + b_\alpha)$$

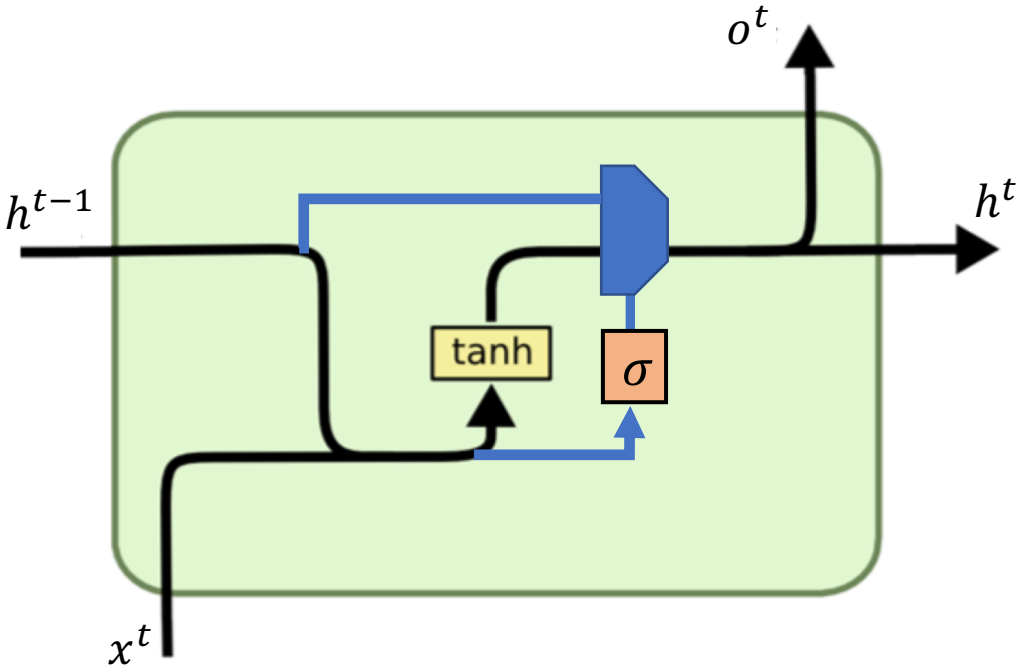
RNN with update gate/multiplexer:

$$\tilde{h}^t = \tanh(U_\alpha x^t + W_\alpha h^{t-1} + b_\alpha)$$

$$u^t = \sigma(U_u x^t + W_u h^{t-1} + b_u)$$

$$h^t = u^t \circ \tilde{h}^t + (1 - u^t) \circ h^{t-1}$$

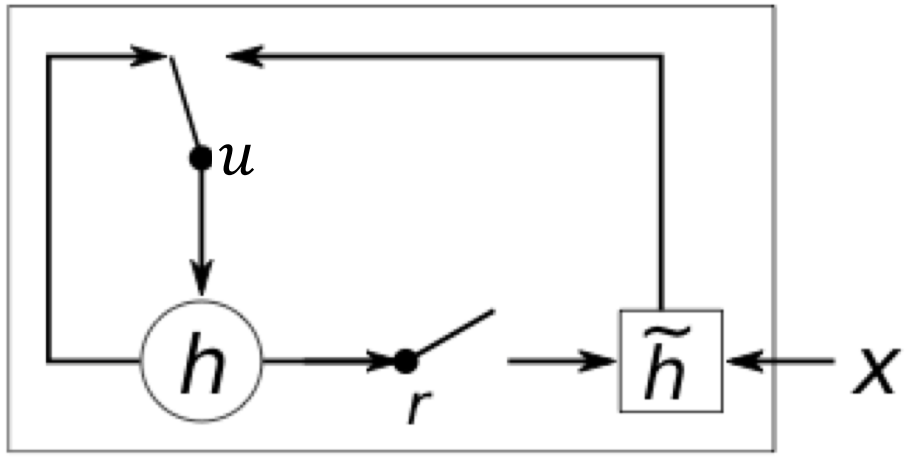
One gate/multiplexer per dimension



Gated Recurrent Unit (GRU)

Has separate update (u) and reset (r) gates:

- Update gate: decides whether the hidden state is to be updated with a new hidden state candidate
- Reset gate: decides whether the previous hidden state should be ignored



source: Cho et al, 2014

$$u_i^{(t)} = \sigma(b_i^z + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t-1)})$$

$$r_i^{(t)} = \sigma(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t-1)})$$

$$h_i^{(t)} = u_i^{(t)} h_i^{(t-1)} + (1 - u_i^{(t)})\sigma(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} r_j^{(t)} h_j^{(t-1)})$$

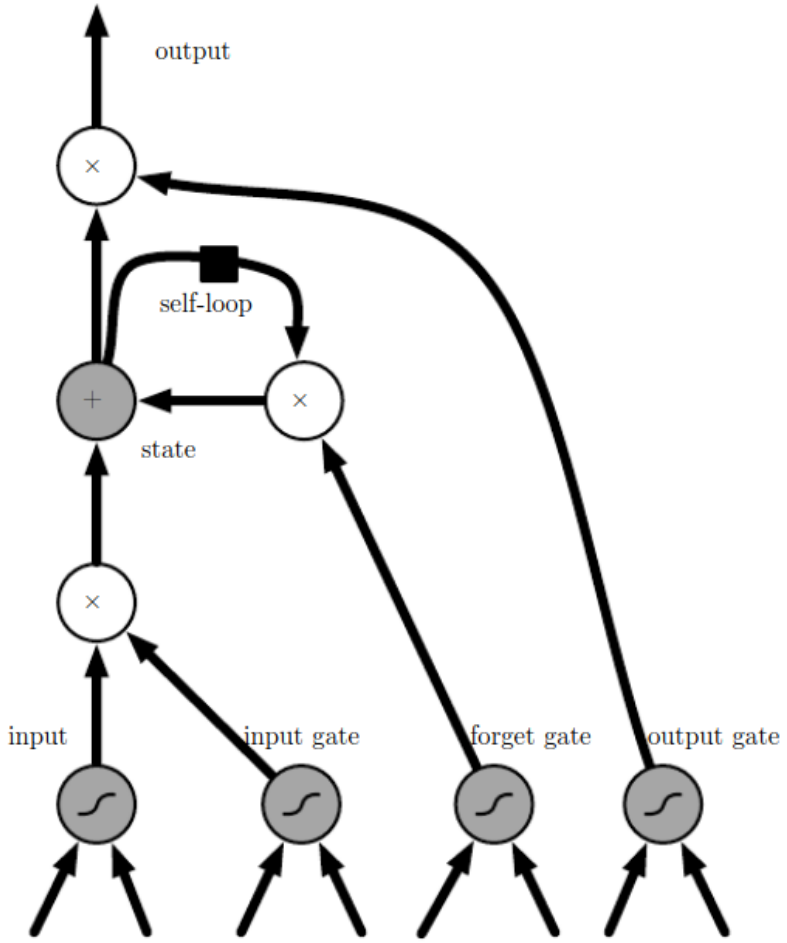
Long Short-Term Memory (LSTM)

Precedes GRU (Hochreiter & Schmidhuber, 1997)
Has an additional state unit and 3 gates:

- Input gate: controls what gets fed from the input for updating the state unit
- Forget gate: controls what is forgotten from the old state
- Output gate: controls what is outputted from the state unit to the hidden unit

The state unit may optionally influence all 3 gates (in addition to the input and the hidden unit).

LSTM is more powerful than GRU and has shown to work better in practice.

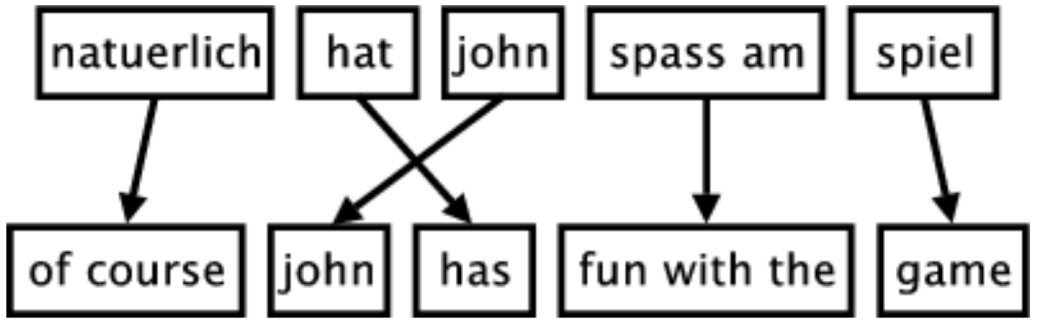


source: http://deeplearningbook.org

Phrase-based Machine Translation

Steps

- Word alignment (unsupervised)
- Phrase candidate building
- Phrase substitution + reordering
  - Beam search based on substitution probability + language model



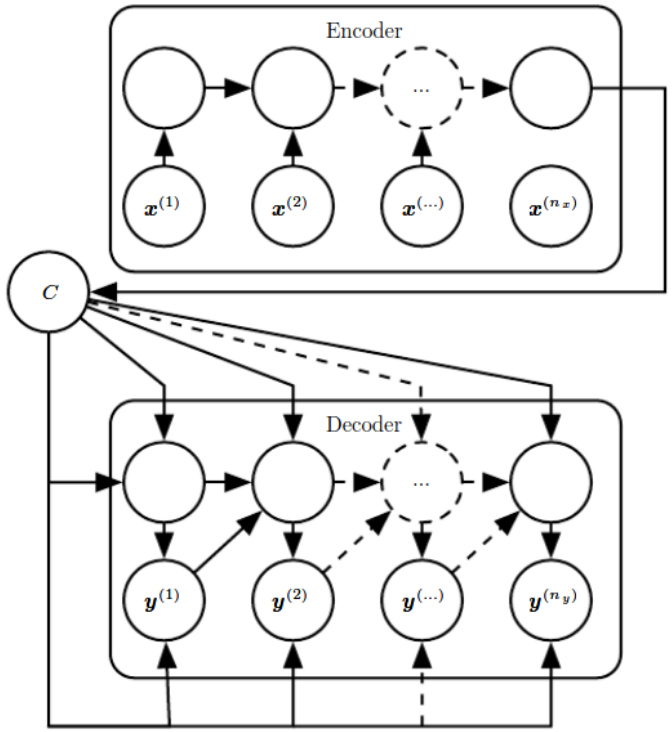
Source: http://www.statmt.org/

RNN MT (Bahdanau et al, 2014)

- Uses RNN encoder-decoder architecture
  - Bidirectional RNN on the source sequence
  - Unidirectional RNN on the output sequence
    - Output dependent context derived from the source encoding
  - GRU for long term dependency
- Joint learning of word alignment and translation
  - Implicit alignment through learned “attention” to the source sequence

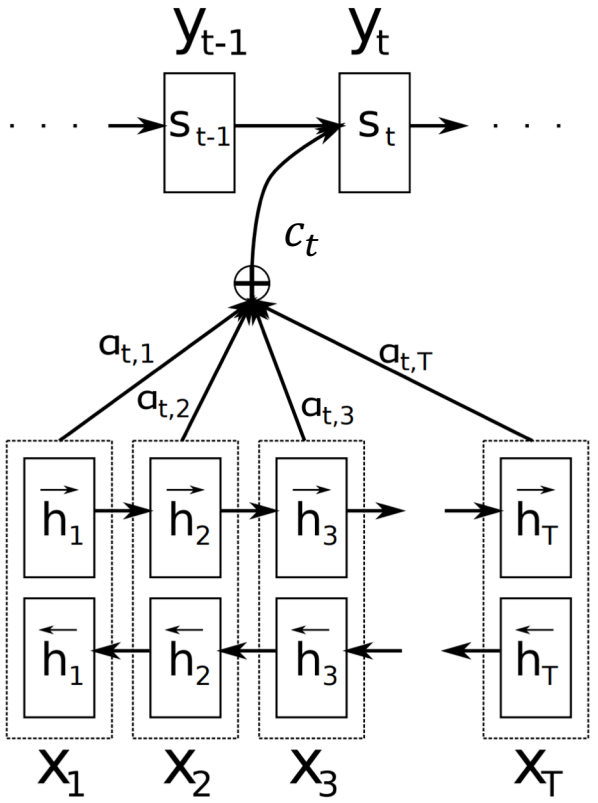
RNN Joint Alignment and Translation

Conventional RNN encoder-decoder with fixed context (C) vector:



source: http://deeplearningbook.org

Joint alignment and translation:



source: Bahdanau et al, 2014

i: output index
j: source index

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

$$\alpha_{ij} = \text{SOFTMAX}(a(s_{i-1}, h_j))$$

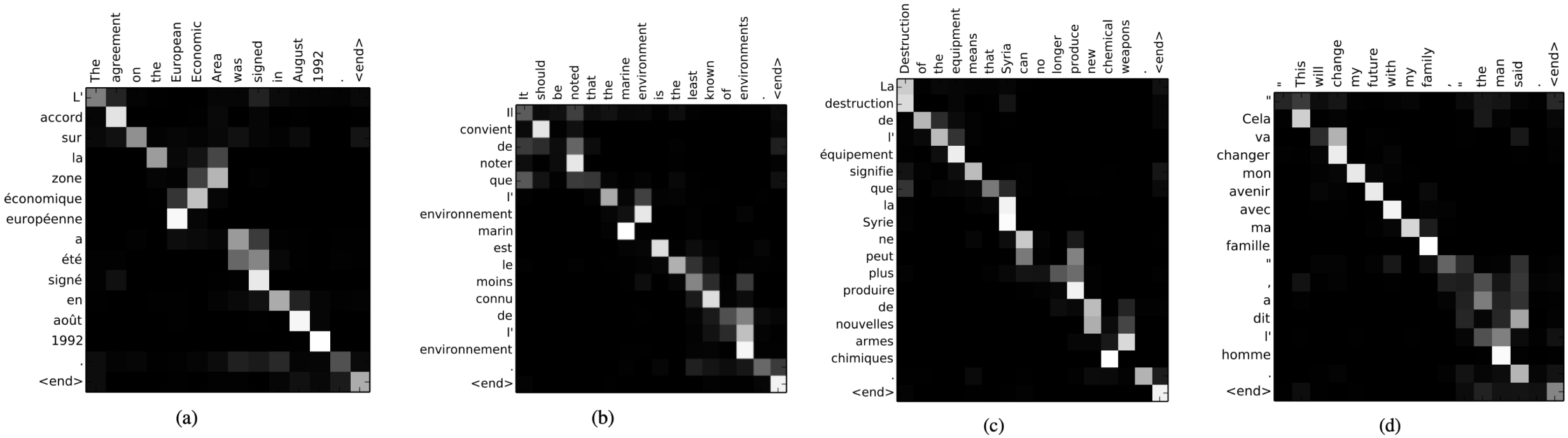
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$a(s_{i-1}, h_j)$ can be interpreted as an alignment or "attention" model (what source tokens the output need to correctly emit the token).

Can be jointly trained w/ (multi-layer) feed-forward network.

Word Alignments

English to French alignments:



source: [Bahdanau et al, 2014]

Joint Alignment and Translation Results

BLEU performance:

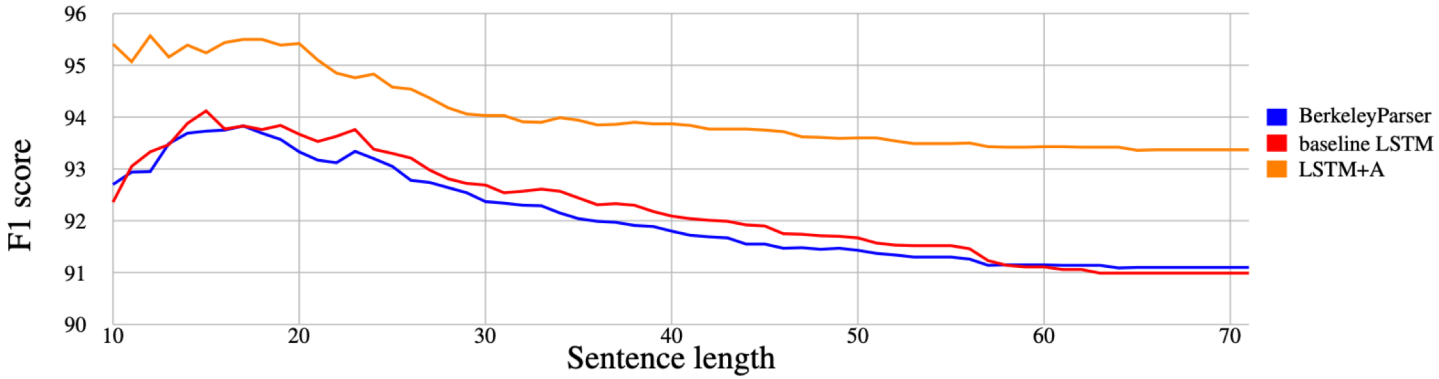
- RNNencdec: w/o attention
- RNNSearch: BiRNN + attention
- Number: max training sentence length
- All/No UNK: whether translation involves unknown (training) words
- Moses: phrase-based MT

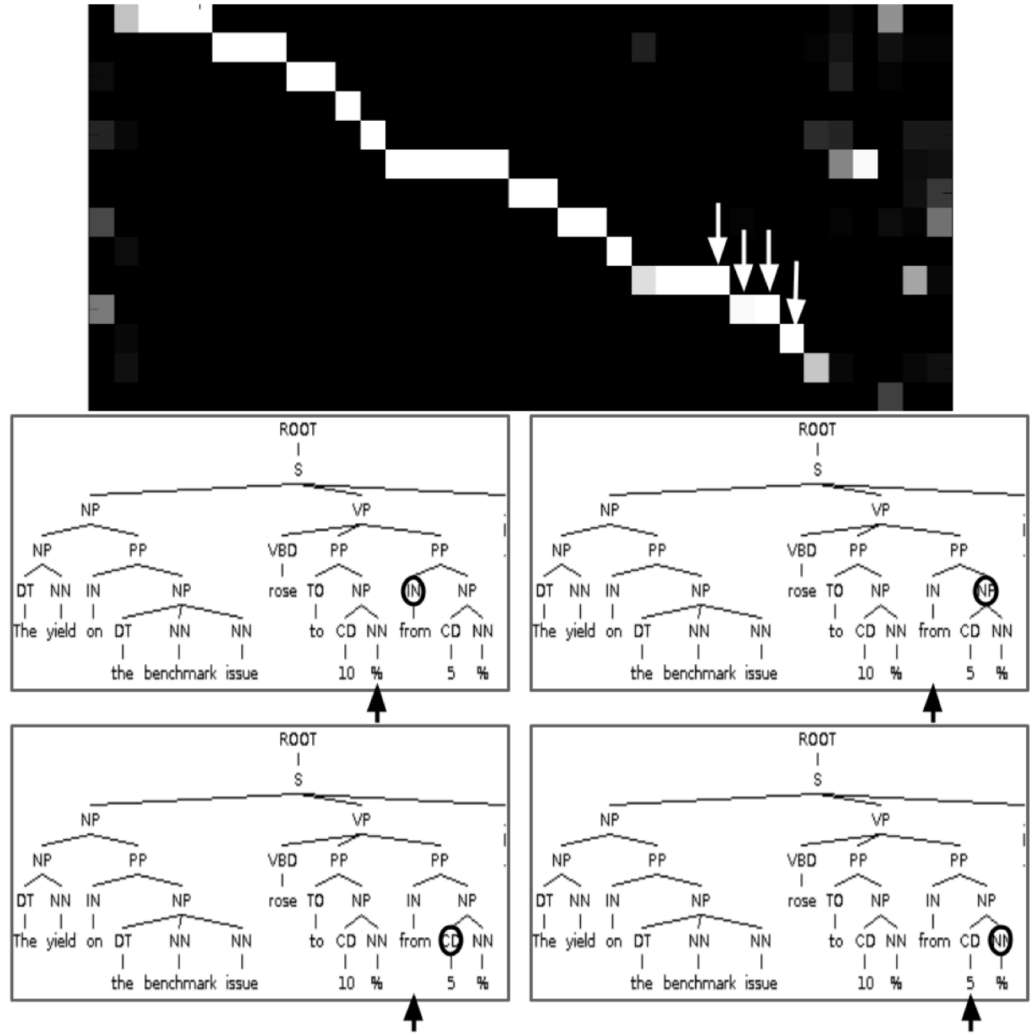
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

source: Bahdanau et al, 2014

Attention for Constituent Parsing

Uses Attention mechanism of Bahdanau et al, 2014.
Less performance degradation on longer sentences:





source: Vinyals et al, 2015

Attention for Image Caption (Correct)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

source: Xu et al., 2015

Attention for Image Caption (Mistakes)

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

source: Xu et al., 2015