LSTMs, Seq-to-Seq models and Attention

Srivatsan Srinivasan

Harvard University

srivatsansrinivasan@g.harvard.edu

January 24, 2019

RNNs and gradient issues

"Back in my college days at **France**, *I was living and Paris and was speaking* **French** *at the university."*

- $h_t = \sigma(Linear((x_t, h_{t-1}))$, $\quad \sigma -$ Activation
- Vanishing and Exploding gradients issues(same as very deep feedforward nets).

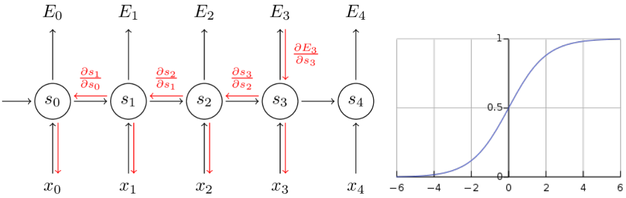


Figure: Back-prop on RNNs and Sigmoid activations.

LSTM - Long Short Term Memory Nets

- Forget Gate : $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
- Input Gate : $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
- Current Cell State (Context) : $\hat{C}_t = \sigma(W_c[h_{t-1}, x_t] + b_c)$
- Cell State(Context) : $C_t = f_t * C_{t-1} + i_t * \hat{C}_t$
- Output Gate : $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
- Hidden Layer Update : $h_t = o_t * tanh(C_t)$

h is overall state of what we have seen so far. C is selective memory of the past. Crudely think of it as a dairy of events (h) vs. what your memory remembers from these events (C).

Bi-lingual volunteers please

1. Read both these sentences once from left to right and try to translate them without looking at source.
2. Do the same as step 1 and the second time, you are allowed to peep at certain phrases while translating.

- *Sebastien lived in France.*
- *Back in Sebastien's days at France, he lived in the city of Paris, a city of great beauty and filled with love and beautiful art, and he spoke French, a language with great history and the national language of France.*

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

⇓

Fixed-Size Encoder (MLP, RNN, CNN)

Encoder(input) $\in \mathbb{R}^D$

⇓

Decoder

Decoder(Encoder(input))

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

Fixed-Size Encoder (MLP, RNN, CNN)

$\text{Encoder}(\text{input}) \in \mathbb{R}^D$

Decoder

$\text{Decoder}(\text{Encoder}(\text{input}))$

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

$$\Downarrow$$

Fixed-Size Encoder (MLP, RNN, CNN)

$$\text{Encoder}(\text{input}) \in \mathbb{R}^D$$

$$\Downarrow$$

Decoder

$$\text{Decoder}(\text{Encoder}(\text{input}))$$

Example: Neural Machine Translation (Sutskever et al., 2014)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $h_t = \mathrm{RNN}(x_t, h_{t-1})$ (Encoder RNN)
- $h_T =$ Last hidden state of RNN encoder (summary of source)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, h_T]))$
- Training: word-level maximum likelihood

$$\arg\max_{\theta} \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x})$$

Example: Neural Machine Translation (Sutskever et al., 2014)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $h_t = \mathrm{RNN}(x_t, h_{t-1})$ (Encoder RNN)
- $h_T =$ Last hidden state of RNN encoder (summary of source)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, h_T]))$
- Training: word-level maximum likelihood

$$\arg\max_{\theta} \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x})$$

Example: Neural Machine Translation (Sutskever et al., 2014)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $h_t = \mathrm{RNN}(x_t, h_{t-1})$ (Encoder RNN)
- $h_T =$ Last hidden state of RNN encoder (summary of source)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, h_T]))$
- Training: word-level maximum likelihood

$$\arg\max_{\theta} \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x})$$

Example: Neural Machine Translation (Sutskever et al., 2014)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $h_t = \mathrm{RNN}(x_t, h_{t-1})$ (Encoder RNN)
- $h_T =$ Last hidden state of RNN encoder (summary of source)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, h_T]))$
- Training: word-level maximum likelihood

$$\arg\max_{\theta} \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x})$$

Example: Neural Machine Translation (Sutskever et al., 2014)

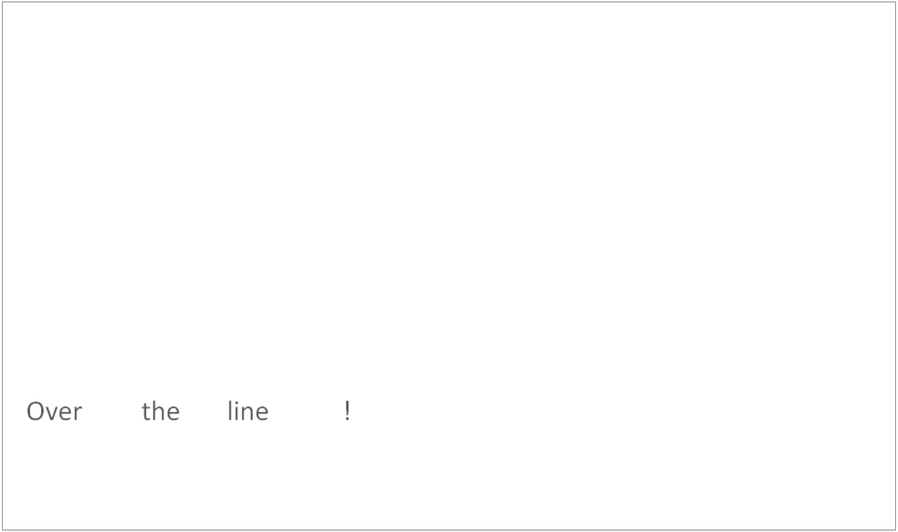
Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

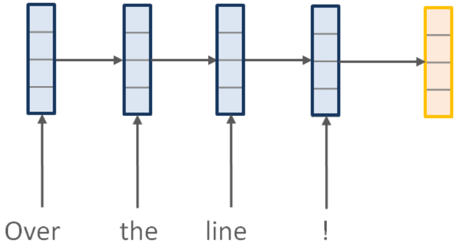
- $h_t = \mathrm{RNN}(x_t, h_{t-1})$ (Encoder RNN)
- $h_T =$ Last hidden state of RNN encoder (summary of source)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, h_T]))$
- Training: word-level maximum likelihood

$$\arg\max_{\theta} \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x})$$

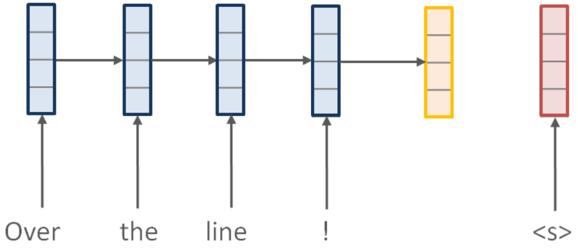
Example: Neural Machine Translation (Sutskever et al., 2014)



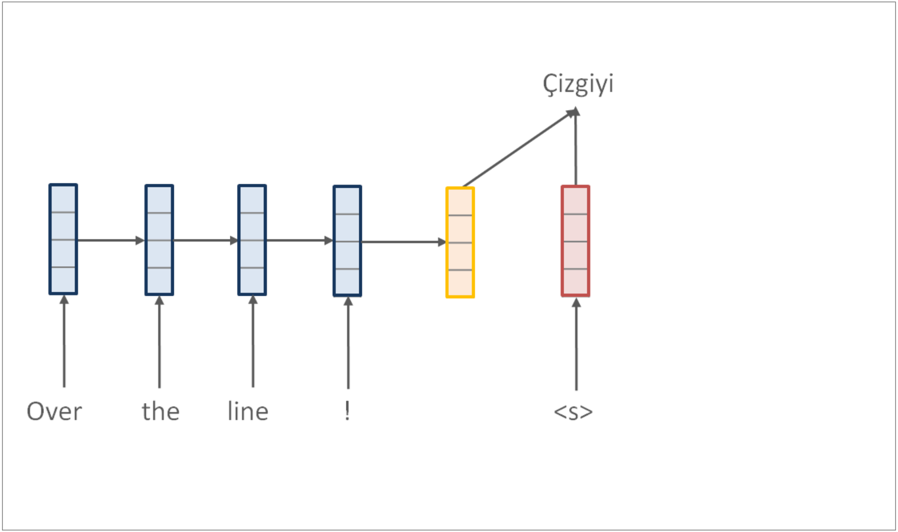
Example: Neural Machine Translation (Sutskever et al., 2014)



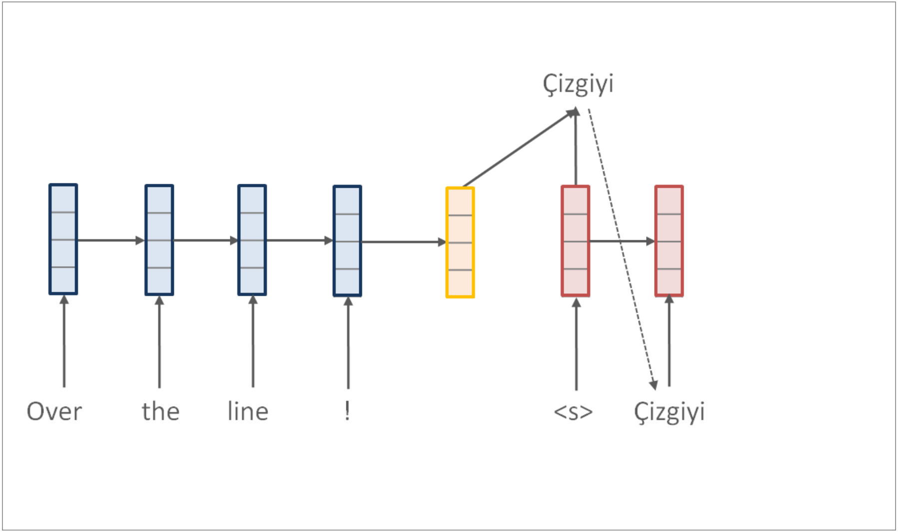
Example: Neural Machine Translation (Sutskever et al., 2014)



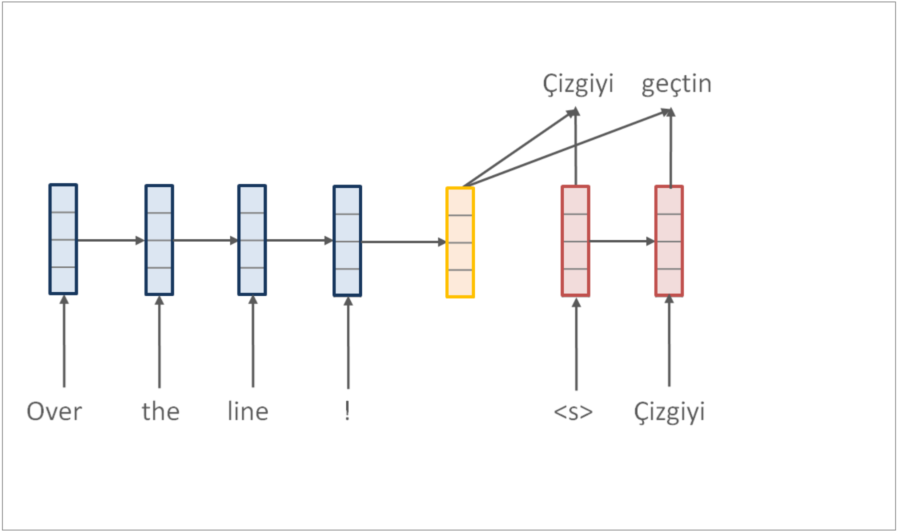
Example: Neural Machine Translation (Sutskever et al., 2014)



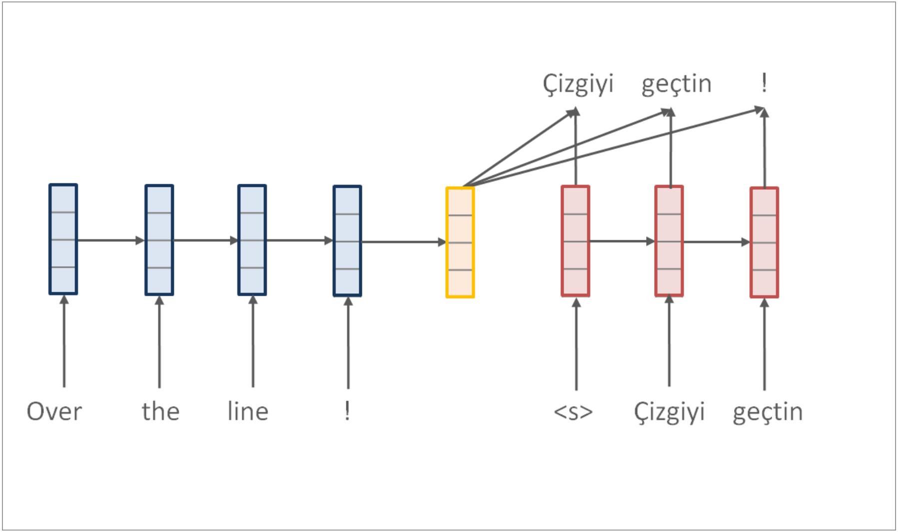
Example: Neural Machine Translation (Sutskever et al., 2014)



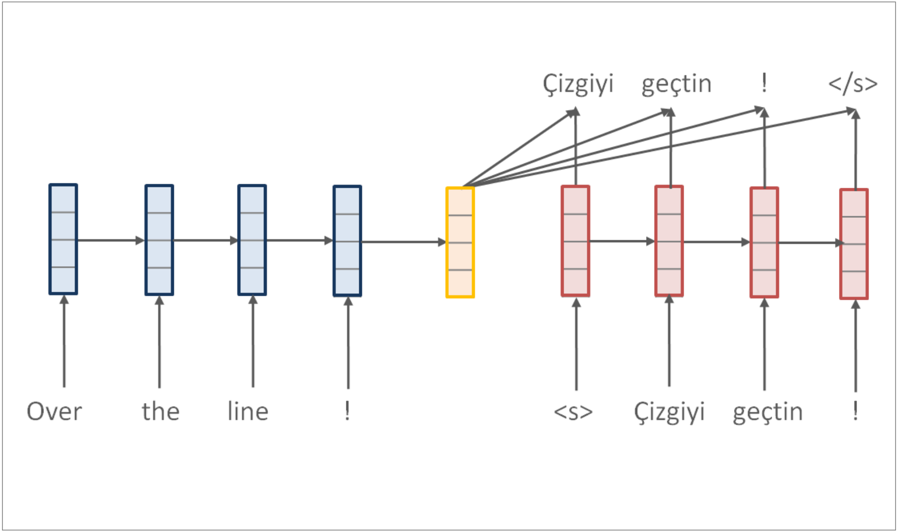
Example: Neural Machine Translation (Sutskever et al., 2014)



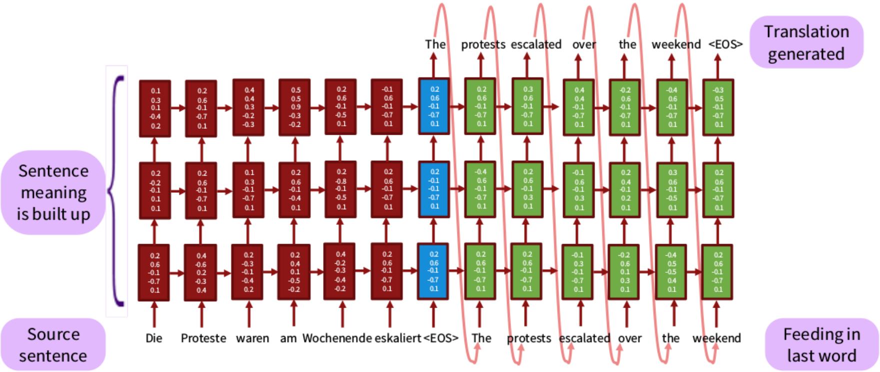
Example: Neural Machine Translation (Sutskever et al., 2014)



Example: Neural Machine Translation (Sutskever et al., 2014)



Example: Neural Machine Translation (Sutskever et al., 2014)



Communication Bottleneck

All input information communicated through fixed-size hidden vector.

$$\text{Encoder}(\text{input})$$

- Training: All gradients have to flow through single bottleneck.
- Test: All input encoded in single vector.

Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

Encoder(input) $= x_1, x_2, \ldots, x_T$

Attention Distribution "where" | Annotation Function "what"

Context Vector ("soft selection")

Decoder

Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

$$\text{Encoder}(\text{input}) = x_1, x_2, \ldots, x_T$$

Attention Distribution "where"

Annotation Function "what"

Context Vector ("soft selection")

Decoder

Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

$$\text{Encoder(input)} = x_1, x_2, \ldots, x_T$$

Attention Distribution
"where"

Annotation Function
"what"

Context Vector ("soft selection")

Decoder

Neural Attention

Input (sentence, image, etc.)

↓

Memory-Bank Encoder (MLP, RNN, CNN)

Encoder(input) $= x_1, x_2, \ldots, x_T$

↓

Attention Distribution "where"

Annotation Function "what"

↓

Context Vector ("soft selection")

↓

Decoder

Neural Attention

Input (sentence, image, etc.)

⇩

Memory-Bank Encoder (MLP, RNN, CNN)

Encoder(input) $= x_1, x_2, \ldots, x_T$

⇩

Attention Distribution "where" | Annotation Function "what"

⇩

Context Vector ("soft selection")

⇩

Decoder

Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $[h_1, \ldots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \dots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \dots, y_L]$

- $[h_1, \dots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $[h_1, \ldots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $[h_1, \ldots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $[h_1, \ldots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

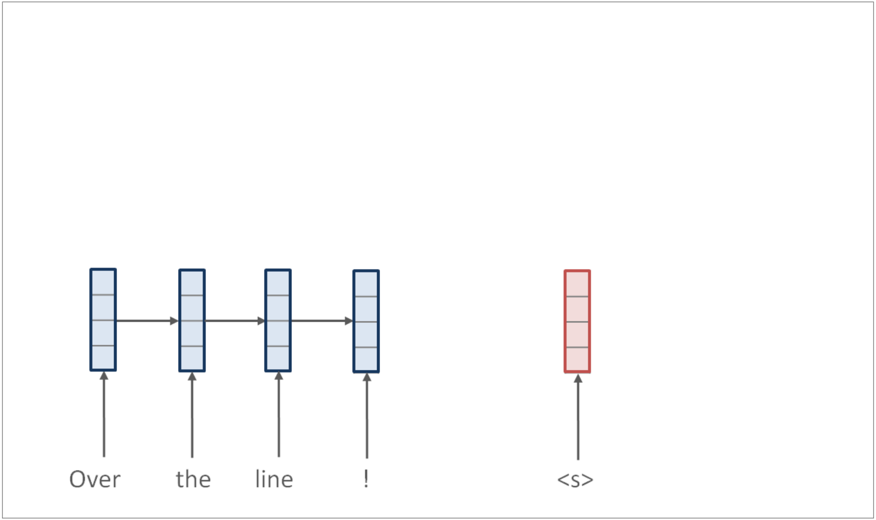
Attention-based Neural Machine Translation (Bahdanau et al., 2015)

Source sentence: $\mathbf{x} = [x_1, \ldots, x_T]$

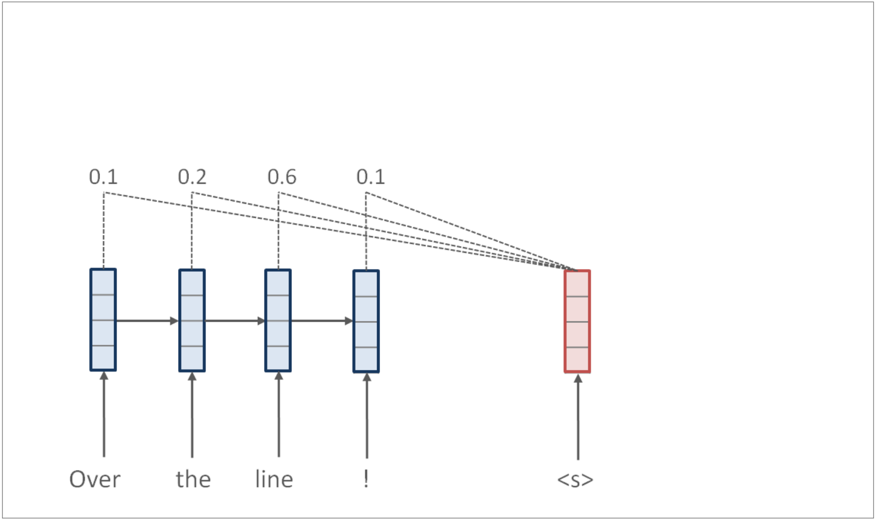
Target sentence: $\mathbf{y} = [y_1, \ldots, y_L]$

- $[h_1, \ldots, h_T] = \mathrm{RNN}(\mathbf{x})$ (Memory/Annotation Function)
- $q_i = \mathrm{RNN}(y_l, q_{i-1})$ (Decoder RNN)
- $\alpha_{i,t} = \frac{\exp(q_i^\top h_t)}{\sum_{j=1}^{T} \exp(q_i^\top h_j)}$ (Attention distribution)
- $c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t$ (Context vector)
- $p(y_i | y_{<i}, \mathbf{x}) = \mathrm{softmax}(\mathrm{MLP}([q_i, c_i]))$

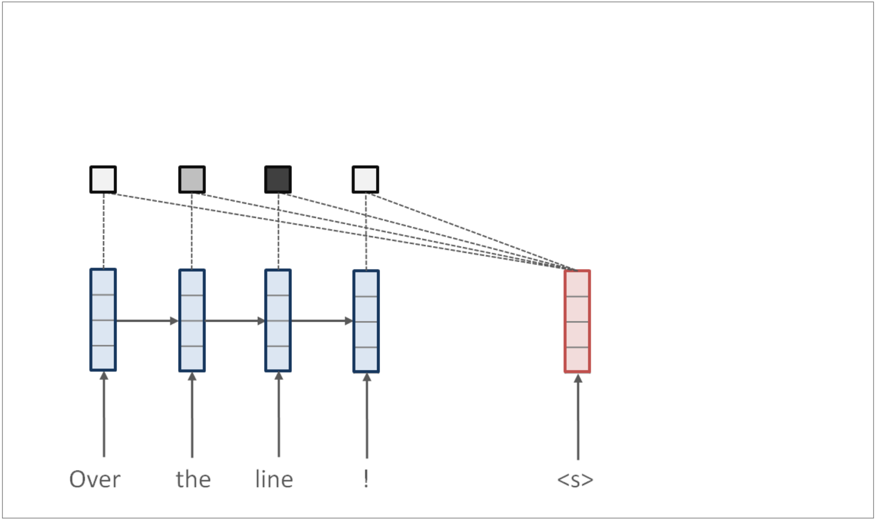
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



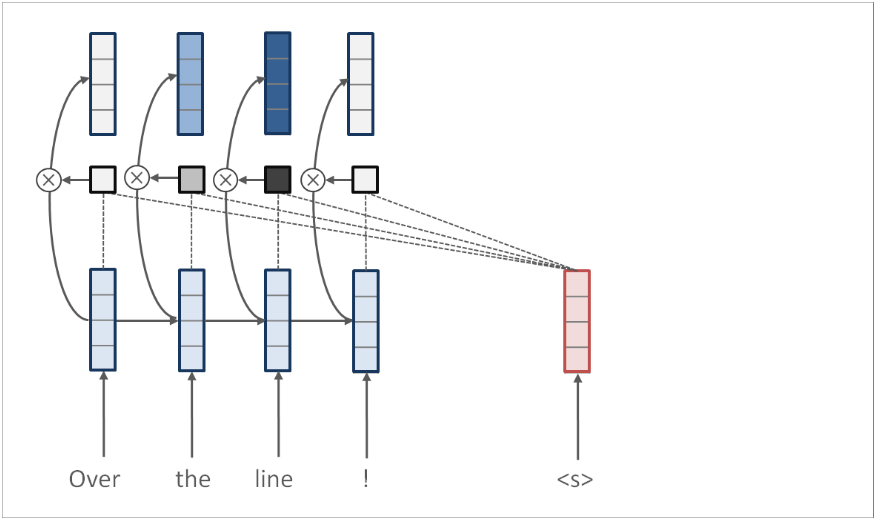
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



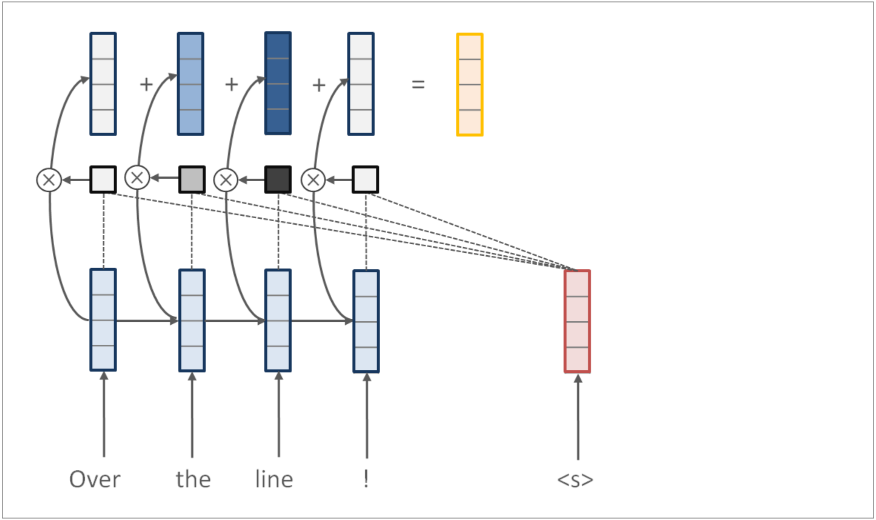
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



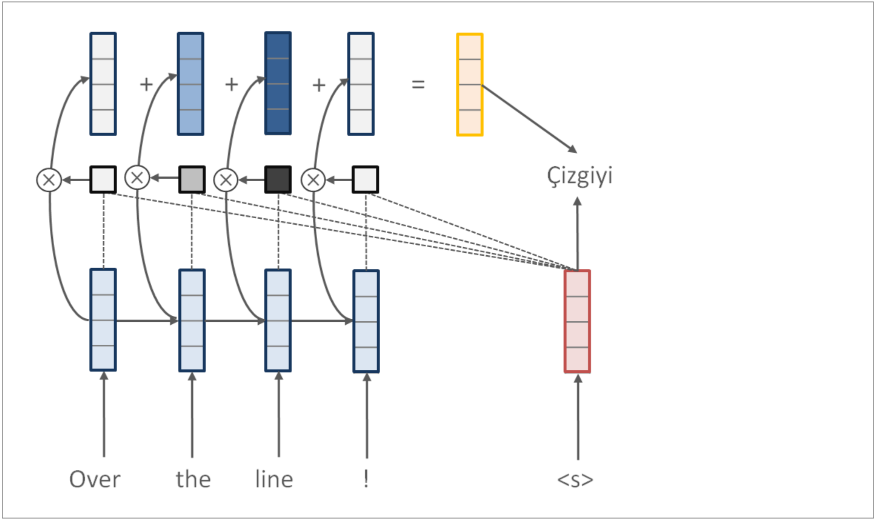
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



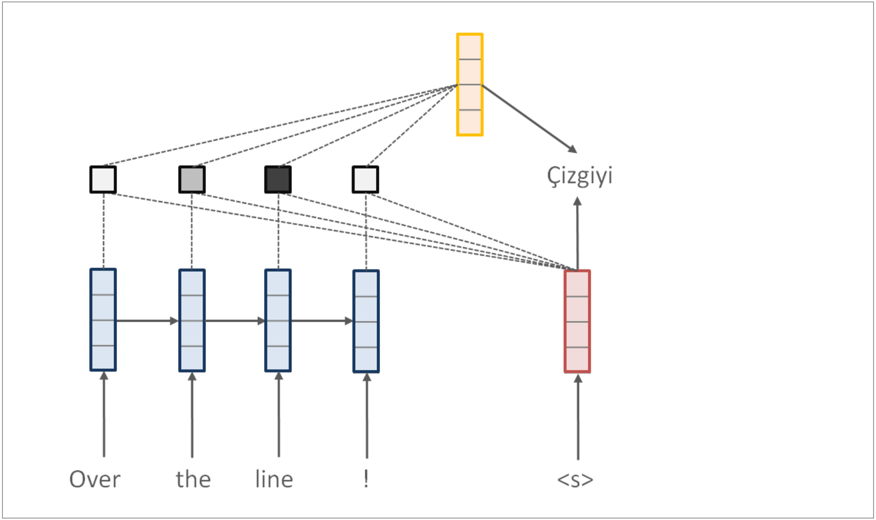
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



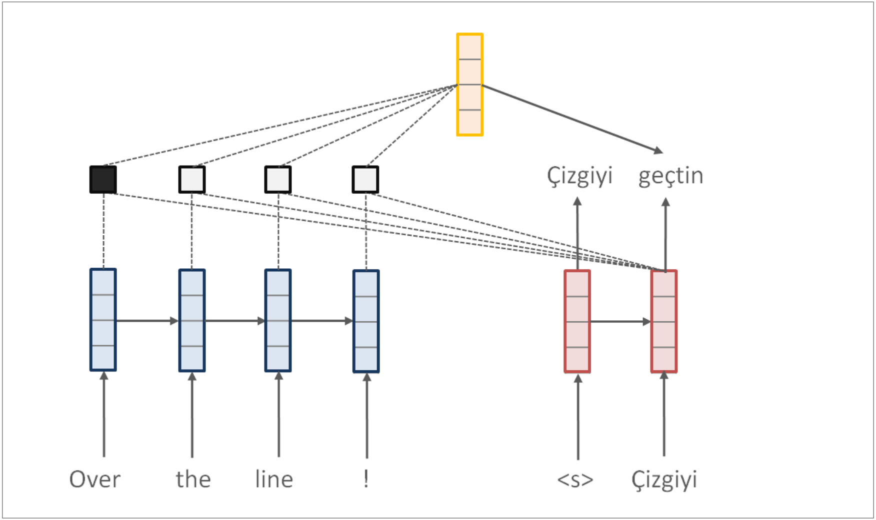
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



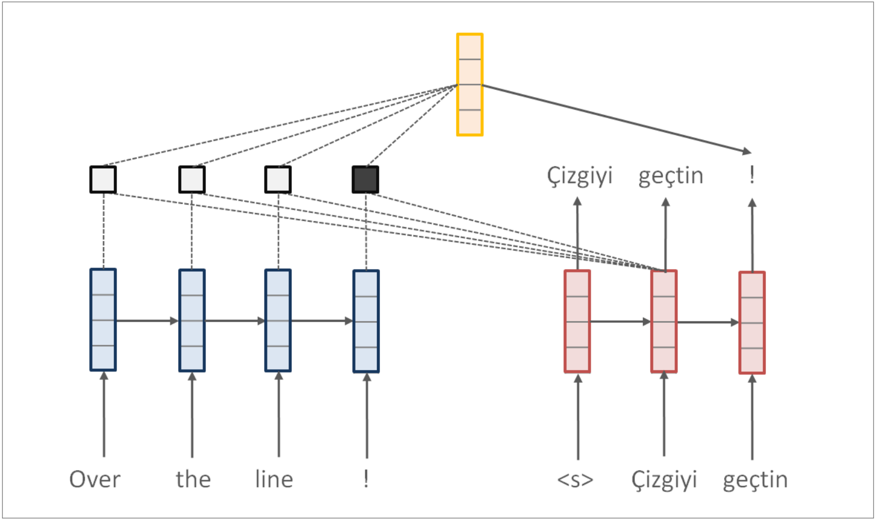
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



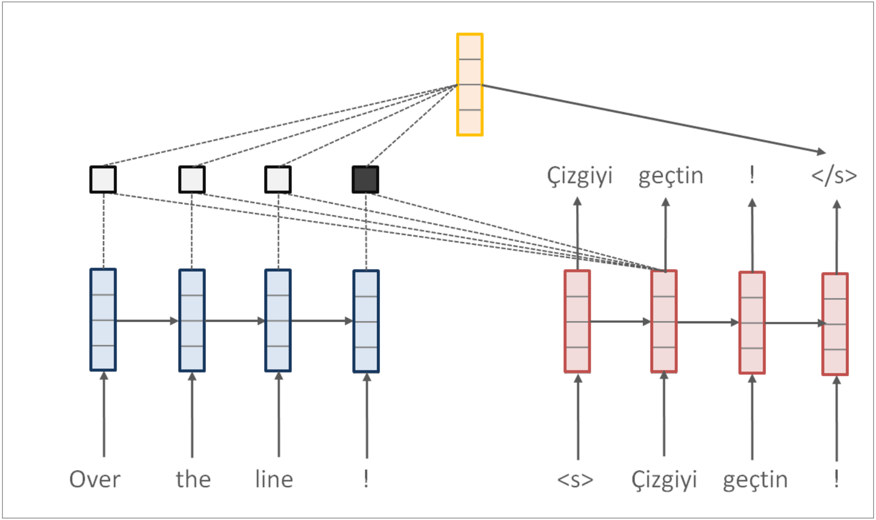
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



Attention-based Neural Machine Translation (Bahdanau et al., 2015)



Attention-based Neural Machine Translation (Bahdanau et al., 2015)



Attention Networks: Notation

$x_1, \ldots, x_T$	Memory bank
q	Query
z	Memory selection ("where")
$p(z = i \mid x, q; \theta)$	Attention distribution
$f(x, z)$	Annotation function ("what")
$c = \mathbb{E}_z[f(x, z)]$	Context vector ("soft selection")

End-to-End Requirements:

1. Need to compute attention distribution $p(z = i \mid x, q; \theta)$
2. Need to backpropagate to learn parameters θ

Attention Networks: Notation

$x_1, \ldots, x_T$	Memory bank
q	Query
z	Memory selection ("where")
$p(z = i \mid x, q; \theta)$	Attention distribution
$f(x, z)$	Annotation function ("what")
$c = \mathbb{E}_z[f(x, z)]$	Context vector ("soft selection")

End-to-End Requirements:

1. Need to compute attention distribution $p(z = i \mid x, q; \theta)$
2. Need to backpropagate to learn parameters θ

Attention Networks: Machine Translation

$x_1, \dots, x_T$	Memory bank	Source RNN hidden states
q	Query	Decoder hidden state
z	Memory selection	Source position $\{1, \dots, T\}$
$p(z = i \mid x, q; \theta)$	Attention distribution	$\mathrm{softmax}(x_i^\top q)$
$f(x, z)$	Annotation function	Memory at time z, i.e. x_z
$c = \mathbb{E}_z[f(x, z)]$	Context vector	$\sum_{i=1}^{T} p(z = i \mid x, q) x_i$

End-to-End Requirements:

1. Need to compute attention $p(z = i \mid x, q; \theta)$
   $\implies$ softmax function
2. Need to backpropagate to learn parameters θ
   $\implies$ Backprop through softmax function

Attention Networks: Machine Translation

$x_1, \ldots, x_T$	Memory bank	Source RNN hidden states
q	Query	Decoder hidden state
z	Memory selection	Source position $\{1, \ldots, T\}$
$p(z = i \mid x, q; \theta)$	Attention distribution	$\mathrm{softmax}(x_i^\top q)$
$f(x, z)$	Annotation function	Memory at time z, i.e. x_z
$c = \mathbb{E}_z[f(x, z)]$	Context vector	$\sum_{i=1}^{T} p(z = i \mid x, q) x_i$

End-to-End Requirements:

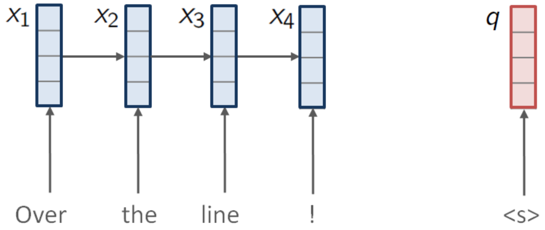
1. Need to compute attention $p(z = i \mid x, q; \theta)$

   $\Longrightarrow$ softmax function

2. Need to backpropagate to learn parameters θ

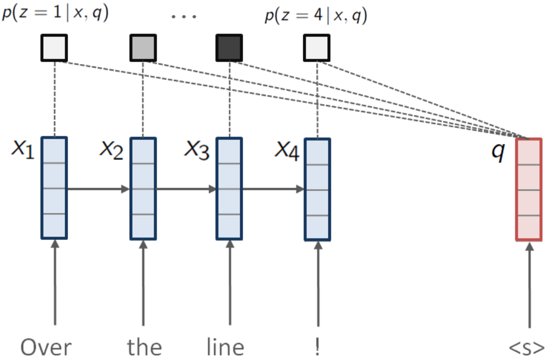
   $\Longrightarrow$ Backprop through softmax function

Attention Networks: Machine Translation



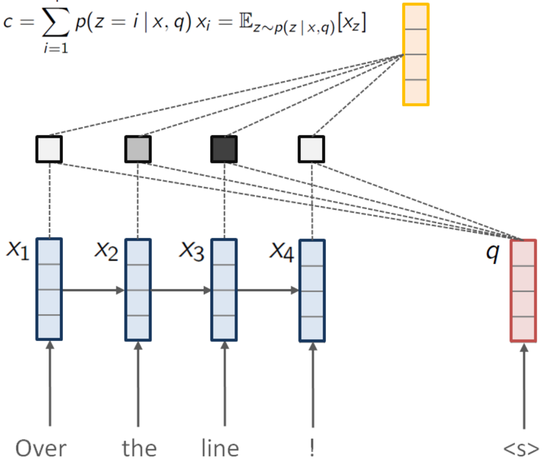
Attention Networks: Machine Translation

$$p(z = i \mid x, q) = \text{softmax}(x_i^\top q) = \frac{\exp(x_i^\top q)}{\sum_{k=1}^{4} \exp(x_k^\top q)}$$

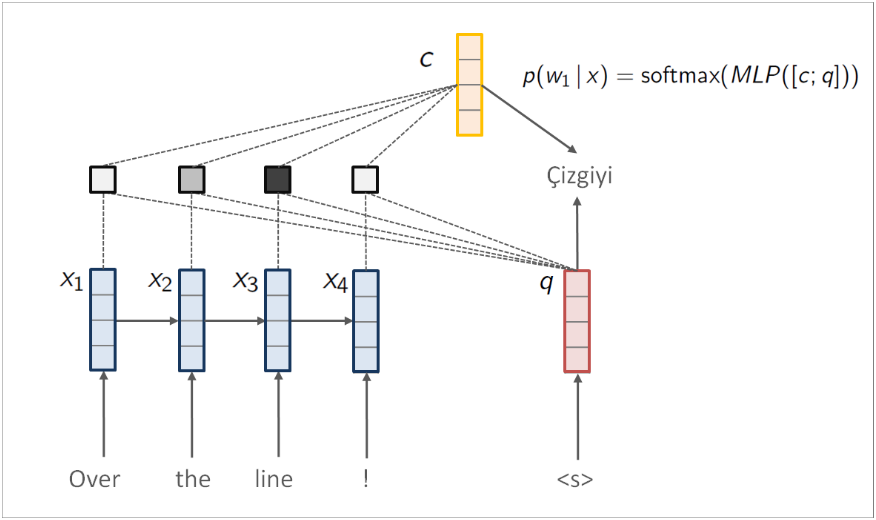


Attention Networks: Machine Translation

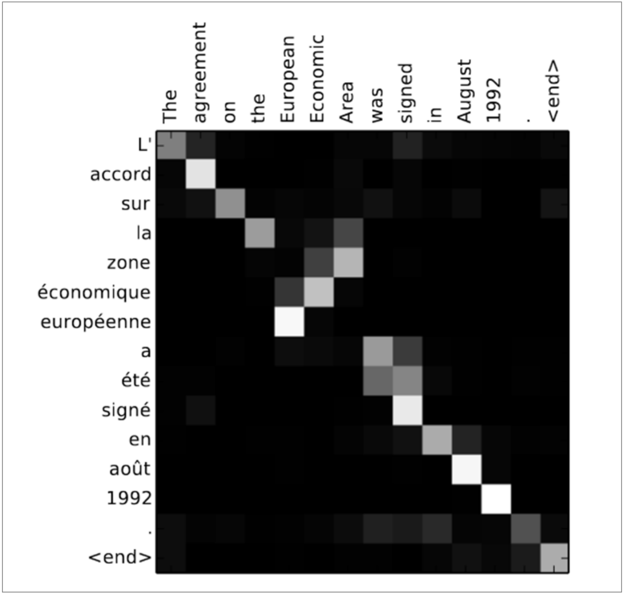
$$c = \sum_{i=1}^{4} p(z = i \mid x, q)\, x_i = \mathbb{E}_{z \sim p(z \mid x, q)}[x_z]$$



Attention Networks: Machine Translation



Attention Visualization



References I

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2015). Listen, Attend and Spell. *arXiv:1508.01211*.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-Based Models for Speech Recognition. In *Proceedings of NIPS*.

Deng, Y., Kanervisto, A., and Rush, A. M. (2016). What You Get Is What You See: A Visual Markup Decompiler. *arXiv:1609.04938*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. *arXiv:1410.5401*.

References II

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature*.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In *Proceedings of NIPS*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*.

Parikh, A. P., Tackstrom, O., Das, D., and Uszkoreit, J. (2016). A Decomposable Attention Model for Natural Language Inference. In *Proceedings of EMNLP*.

References III

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kocisky, T., and Blunsom, P. (2016). Reasoning about Entailment with Neural Attention. In *Proceedings of ICLR*.

Rush, A. M., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-To-End Memory Networks. In *Proceedings of NIPS*.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015a). Pointer Networks. In *Proceedings of NIPS*.

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015b). Grammar as a Foreign Language. In *Proceedings of NIPS*.

References IV

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICML*.