

# Memory Networks and Neural Turing Machines

Diego Marcheggiani

University of Amsterdam  
ILLC

Unsupervised Language Learning 2016

# Outline

Motivation

Memory Networks

End-to-end Memory Networks

Neural Turing Machines

# Outline

## Motivation

## Memory Networks

## End-to-end Memory Networks

## Neural Turing Machines

# Motivation

- ▶ Neural networks have hard times to capture long-range dependencies.

# Motivation

- ▶ Neural networks have hard times to capture long-range dependencies. Yes, even LSTMs.

# Motivation

- ▶ Neural networks have hard times to capture long-range dependencies. Yes, even LSTMs.
- ▶ Memory networks (MN) and Neural Turing machines (NTM) try to overcome this problem using an external memory.
  - ▶ MN are mainly motivated by the fact that it is hard to capture long-range dependencies,
  - ▶ while NTM are devised to perform program induction.

# Outline

Motivation

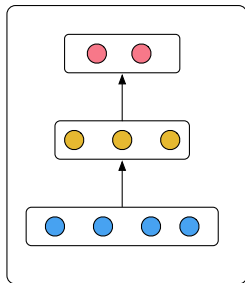
Memory Networks

End-to-end Memory Networks

Neural Turing Machines

# General idea

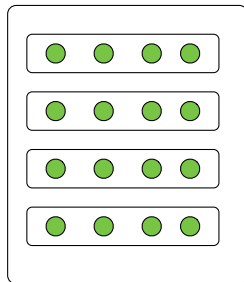
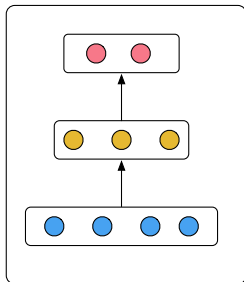
- ▶ We have a neural network, RNN, MLP, ...





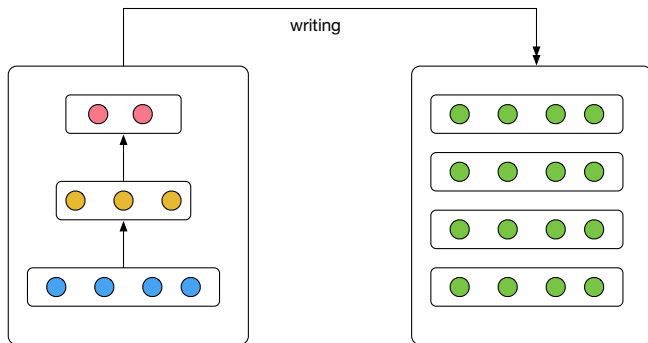
# General idea

- ▶ We have a neural network, RNN, MLP, ...
- ▶ An external memory.



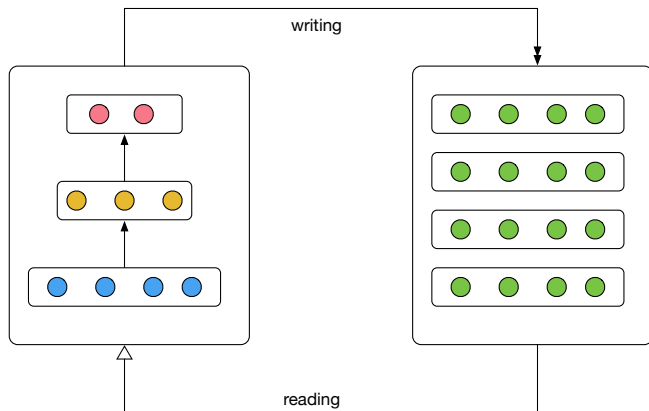
# General idea

- ▶ We have a neural network, RNN, MLP, does not matter.
- ▶ External memory where the neural network can write and read to.



# General idea

- ▶ We have a neural network, RNN, MLP, does not matter.
- ▶ External memory where the neural network can write and read to.



# Memory network components

- ▶ **Input feature map (I)**: transforms the input in a feature representation, e.g., bag of words

# Memory network components

- ▶ **Input feature map (I)**: transforms the input in a feature representation, e.g., bag of words
- ▶ **Generalization (G)**: writes the input, or a function of it, on the memory

# Memory network components

- ▶ **Input feature map (I)**: transforms the input in a feature representation, e.g., bag of words
- ▶ **Generalization (G)**: writes the input, or a function of it, on the memory
- ▶ **Output feature map (O)**: reads the most relevant memory slots

# Memory network components

- ▶ **Input feature map (I)**: transforms the input in a feature representation, e.g., bag of words
- ▶ **Generalization (G)**: writes the input, or a function of it, on the memory
- ▶ **Output feature map (O)**: reads the most relevant memory slots
- ▶ **Response (R)**: given the info read from the memory, returns the output

# Memory network components

- ▶ **Input feature map (I)**: transforms the input in a feature representation, e.g., bag of words
- ▶ **Generalization (G)**: writes the input, or a function of it, on the memory
- ▶ **Output feature map (O)**: reads the most relevant memory slots
- ▶ **Response (R)**: given the info read from the memory, returns the output

**Extremely general framework** which can be instantiated in several ways.



# Question answering

- ▶ Input text: Fred moved to the bedroom. Joe went to the kitchen. Joe took the milk. Dan journeyed to the bedroom.

# Question answering

- ▶ Input text: Fred moved to the bedroom. Joe went to the kitchen. Joe took the milk. Dan journeyed to the bedroom.
- ▶ Input question: Where is Dan now?

# Question answering

- ▶ Input text: Fred moved to the bedroom. Joe went to the kitchen. Joe took the milk. Dan journeyed to the bedroom.
- ▶ Input question: Where is Dan now?
- ▶ Output answer: bedroom

Let's see a simple instantiation of memory networks for QA.













# l component

Raw text sentence is transformed in its vector representation e.g., bag of words, with the component l.

Fred moved to the bedroom.  
Joe went to the kitchen.  
Joe took the milk.  
Dan journeyed to the bedroom.

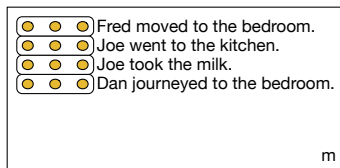
# I component

Raw text sentence is transformed in its vector representation e.g., bag of words, with the component I.

  	Fred moved to the bedroom.
  	Joe went to the kitchen.
  	Joe took the milk.
  	Dan journeyed to the bedroom.

# G component

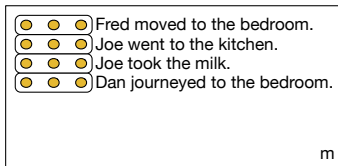
The sentences are then written to the memory sequentially, via the component G.



Notice that the memory is fixed in this approach once is written, it is not changed neither during learning nor during testing.

# I component

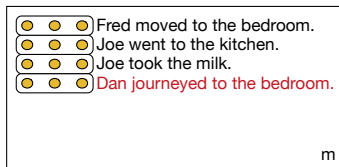
The question is transformed in its vector representation with the component I.



● ● ● Where is Dan now?

# O component

The best matching memory (supporting fact), according to the question is retrieved with the component O.



 Where is Dan now?

$$o_1 = O_1(q, \mathbf{m}) = \operatorname{argmax}_{i=1, \dots, N} s_O(q, \mathbf{m}_i)$$

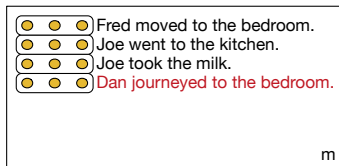
where the similarity function is defined as:

$$s_O(x, y) = x^T \cdot U_O^T \cdot U_O \cdot y$$



# R component

Given the supporting fact and the query, the best matching word in the dictionary is retrieved.



Where is Dan now?

Answer: bedroom

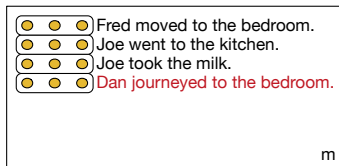
$$r = \operatorname{argmax}_{w \in W} s_R(q + \mathbf{m}_{o_1}, w)$$

where the similarity function is defined as below:

$$s_R(x, y) = x^T \cdot U_R^T \cdot U_R \cdot y$$

# R component

Given the supporting fact and the query, the best matching word in the dictionary is retrieved.



$$r = \operatorname{argmax}_{w \in W} s_R(q + \mathbf{m}_{o_1}, w)$$

where the similarity function is defined as below:

$$s_R(x, y) = x^T \cdot U_R^T \cdot U_R \cdot y$$

What about harder questions?

# Question Answering

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Fred moved to the bedroom.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Joe went to the kitchen.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Joe took the milk.
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Dan journeyed to the bedroom.

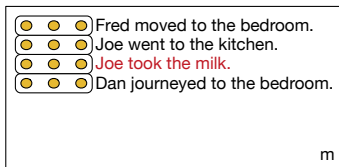
m




Where is the milk now?

# Question Answering

The best matching memory (supporting fact), according to the question is retrieved with the component  $O$ .



 Where is the milk now?

$$o_1 = O_1(q, \mathbf{m}) = \operatorname{argmax}_{i=1, \dots, N} s_O(q, \mathbf{m}_i)$$

where the similarity function is defined as:

$$s_O(x, y) = x^T \cdot U_O^T \cdot U_O \cdot y$$

# Question Answering

We also need another supporting fact

● ● ●	Fred moved to the bedroom.
● ● ●	Joe went to the kitchen.
● ● ●	Joe took the milk.
● ● ●	Dan journeyed to the bedroom.
m	



Where is the milk now?

$$o_2 = O_2(q + \mathbf{m}_{o_1}, \mathbf{m}) = \operatorname{argmax}_{i=1, \dots, N} s_O(q + \mathbf{m}_{o_1}, \mathbf{m}_i)$$

where the similarity function is defined as:

$$s_O(x, y) = x^T \cdot U_O^T \cdot U_O \cdot y$$

# Question Answering

Given the supporting facts and the query, the best matching word in the dictionary is retrieved.

Fred moved to the bedroom.

Joe went to the kitchen.

Joe took the milk.

Dan journeyed to the bedroom.

m

Where is the milk now?

Answer: kitchen

$$r = \operatorname{argmax}_{w \in W} s_r(q + o_1 + o_2, w)$$

where the similarity function is defined as below:

$$s_r(x, y) = x^T \cdot U_R^T \cdot U_R \cdot y$$

# Training

Training is then performed with a hinge loss and stochastic gradient descent (SGD).

$$\sum_{f \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_o(q, \mathbf{m}_{o_1}) + s_o(q, f)) +$$

# Training

Training is then performed with a hinge loss and stochastic gradient descent (SGD).

$$\sum_{f \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_o(q, \mathbf{m}_{o_1}) + s_o(q, f)) +$$

$$\sum_{f' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_o(q + \mathbf{m}_{o_1}, \mathbf{m}_{o_2}) + s_o(q + \mathbf{m}_{o_1}, f')) +$$



# Training

Training is then performed with a hinge loss and stochastic gradient descent (SGD).

$$\sum_{f \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_o(q, \mathbf{m}_{o_1}) + s_o(q, f)) +$$

$$\sum_{f' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_o(q + \mathbf{m}_{o_1}, \mathbf{m}_{o_2}) + s_o(q + \mathbf{m}_{o_1}, f')) +$$

$$\sum_{\hat{r} \neq r} \max(0, \gamma - s_R(q + \mathbf{m}_{o_1} + \mathbf{m}_{o_2}, r) + s_R(q + \mathbf{m}_{o_1} + \mathbf{m}_{o_2}, \hat{r}))$$

Negative sampling instead of sum.

# Experiments

Question answering with artificially generated data.

model	accuracy
RNN	17.8 %
LSTM	29.0 %
MN $k=1$	44.4 %
MN $k=2$	99.9 %

How many problems can you spot?

# How many problems can you spot?

- ▶ Single word as answer.

# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.

# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.
- ▶ The write component is somehow naive.

# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.
- ▶ The write component is somehow naive.
- ▶ Strongly

# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.
- ▶ The write component is somehow naive.
- ▶ Strongly fully



# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.
- ▶ The write component is somehow naive.
- ▶ Strongly fully extremely

# How many problems can you spot?

- ▶ Single word as answer.
- ▶ Need to iterate over the entire memory.
- ▶ The write component is somehow naive.
- ▶ Strongly fully extremely supervised.

# Outline

Motivation

Memory Networks

End-to-end Memory Networks

Neural Turing Machines

# Introduction


- ▶  $\text{argmax}$  is substituted by a soft attention mechanism
- ▶ less supervised, no need for annotated supporting facts


## QA example


Transform sentences in vector representation, write representations in the memory, transform query in vector representation.

 Where is Dan now?

 Fred moved to the bedroom.

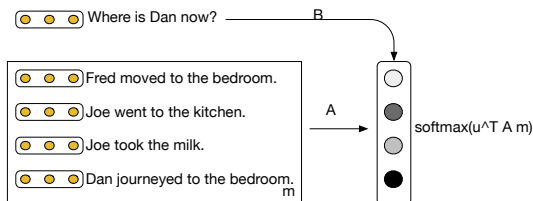
 Joe went to the kitchen.

 Joe took the milk.

 Dan journeyed to the bedroom.  
m

## QA example

For each sentence in memory calculate the "level of compatibility" between the sentence and the query - **soft attention**.

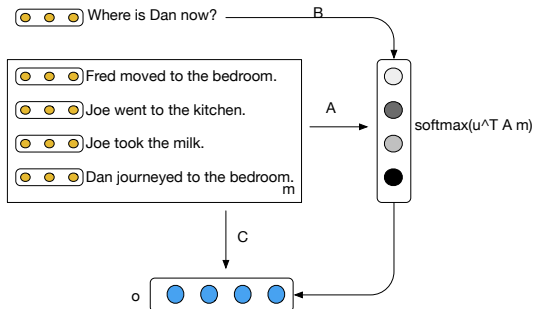


$$u = B \cdot q$$

$$p_i = \text{softmax}(u^T \cdot A \cdot \mathbf{m}_i)$$

## QA example

Calculate the weighted output representation.



$$c_i = C \cdot \mathbf{m}_i$$

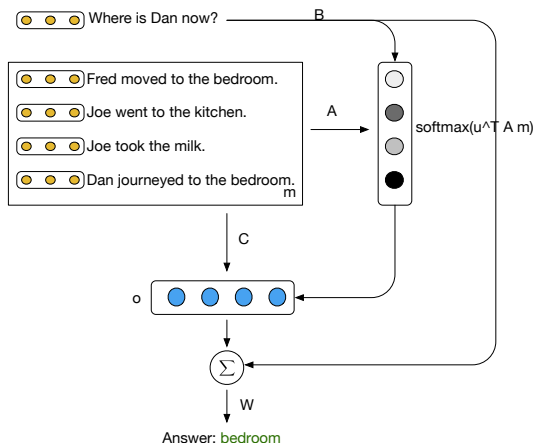
where  $c$  is the output memory representation

$$o = \sum_i (p_i c_i)$$

and  $o$  is the weighted output representation.

## QA example

Calculate the most likely answer given the query and the output memory representation.



$$r = \operatorname{argmax}_{w \in W} (w \cdot (o + u))$$



# QA example

As in the fully supervised case, we can perform multiple readings of the memory given the previous result.

Where is the milk now?

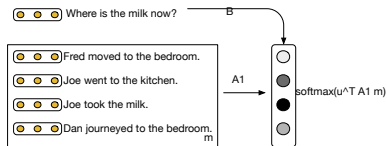
Fred moved to the bedroom.

Joe went to the kitchen.

Joe took the milk.

Dan journeyed to the bedroom.  
m

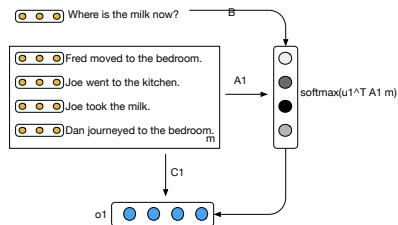
# QA example



$$u^1 = B \cdot q$$

$$p_i^1 = \text{softmax}(u^{1T} \cdot A^1 \cdot m_i)$$

# QA example



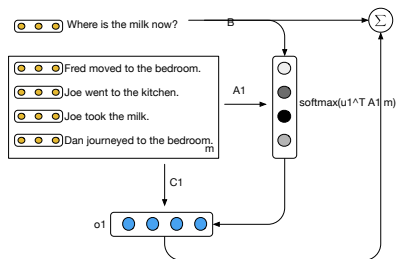
$$c_i^1 = C^1 \cdot \mathbf{m}_i$$

where  $c^1$  is the output memory representation at the first hop

$$o^1 = \sum_i (p_i^1 c_i^1)$$

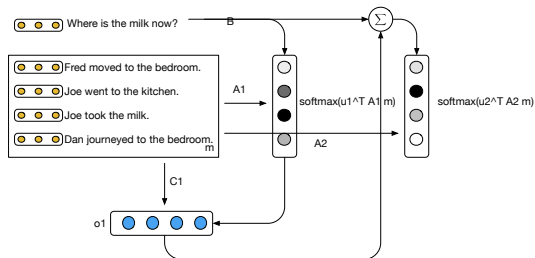
and  $o^1$  is the weighted output representation at the first hop.

# QA example



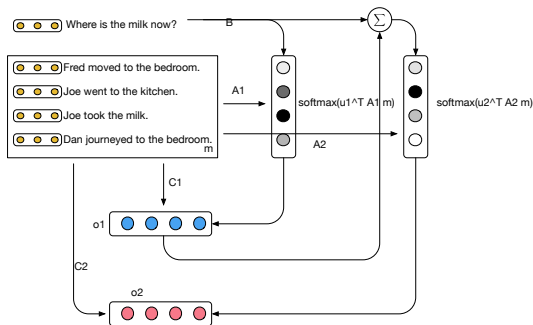
$$u^2 = (u^1 + o^1)$$

# QA example



$$p_i^2 = \text{softmax}(u^{2T} \cdot A^2 \cdot m_i)$$

## QA example



$$c_i^2 = C^2 \cdot m_i$$

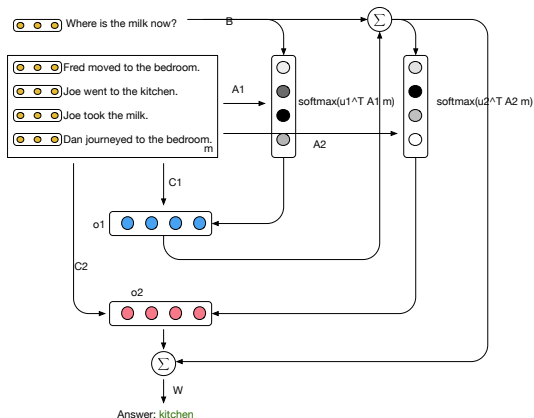
where  $c^2$  is the output memory representation at the second hop

$$o^2 = \sum_i (p_i^2 c_i^2)$$

and  $o^2$  is the weighted output representation at the second hop.

## QA example

As in the fully supervised case we can perform multiple reading of the memory given the previous result.



$$r = \operatorname{argmax}_{w \in W} (w \cdot (o^2 + u^2))$$

# Experiments

Question answering toy tasks, Weston et al. (2016).

model	mean error
Strongly sup. MN	6.7 %
LSTM	51.3 %
EEMN $k=1$	25.8 %
EEMN $k=2$	15.6 %
EEMN $k=3$	13.3 %



# Outline

Motivation

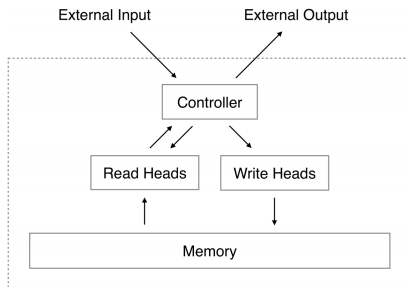
Memory Networks

End-to-end Memory Networks

Neural Turing Machines

# Introduction

- ▶ As Turing machines, NTMs have a controller, a memory, a write head, and a read head. NTMs are differentiable.
- ▶ Differently from Memory Networks,
  - ▶ the attention mechanism of NTMs is more sophisticated.
  - ▶ NTMs are already equipped for rewriting the memory.



# Read head

- ▶ The memory can be updated during training and testing, at each time  $t$  we have a memory  $M_t$ .

# Read head

- ▶ The memory can be updated during training and testing, at each time  $t$  we have a memory  $M_t$ .
- ▶  $\mathbf{w}_t^r$  is the weighting vector over the memory at time  $t$  - It is constrained to be a probability distribution.

# Read head

- ▶ The memory can be updated during training and testing, at each time  $t$  we have a memory  $M_t$ .
- ▶  $\mathbf{w}_t^r$  is the weighting vector over the memory at time  $t$  - It is constrained to be a probability distribution.
- ▶ The read vector is calculated as:

$$\mathbf{r}_t = \sum_i w_t(i) M_t(i)$$

$\mathbf{w}_t^r$  is emitted by the controller.

# Write head

- ▶  $\mathbf{w}_t^w$  is the write weighting vector (emitted by the controller).
- ▶  $M_t(i)$  represents the memory location  $i$  at time step  $t$ .

# Write head

- ▶  $\mathbf{w}_t^w$  is the write weighting vector (emitted by the controller).
- ▶  $M_t(i)$  represents the memory location  $i$  at time step  $t$ .
- ▶ write operation is composed by two parts:
- ▶ erase part
  - ▶ the controller emits an **erase** vector  $\mathbf{e}_t$  in the range  $(0,1)$

$$\tilde{M}_t(i) = M_{t-1}(i)[\mathbf{1} - w_t(i)\mathbf{e}_t]$$

# Write head

- ▶  $\mathbf{w}_t^w$  is the write weighting vector (emitted by the controller).
- ▶  $M_t(i)$  represents the memory location  $i$  at time step  $t$ .

- ▶ write operation is composed by two parts:
- ▶ erase part
  - ▶ the controller emits an **erase** vector  $\mathbf{e}_t$  in the range  $(0,1)$

$$\tilde{M}_t(i) = M_{t-1}(i)[\mathbf{1} - w_t(i)\mathbf{e}_t]$$

- ▶ add part
  - ▶ the controller emits an **add** vector  $\mathbf{a}_t$

$$M_t(i) = \tilde{M}_t(i) + w_t(i)\mathbf{a}_t$$

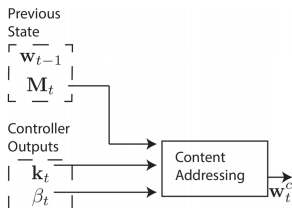


# Addressing mechanism

How do we get  $\mathbf{w}_t$ ?

- ▶ content-based addressing
- ▶ location-based addressing

# Content-based addressing

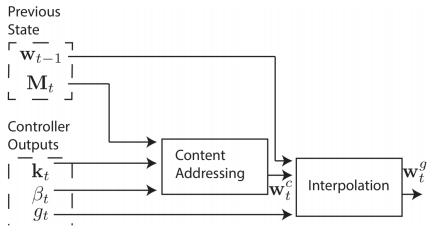


- ▶ the controller emits a key vector  $\mathbf{k}_t$
- ▶ the key vector is compared to the memory via cosine similarity  $K[\cdot, \cdot]$
- ▶

$$w_t(i) = \frac{\exp(\beta_t \cdot K[\mathbf{k}_t, M_t(i)])}{\sum_j \exp(\beta_t \cdot K[\mathbf{k}_t, M_t(j)])}$$

- ▶  $\beta_t$  is a scalar emitted by the controller that attenuates or amplifies the precision of the focus

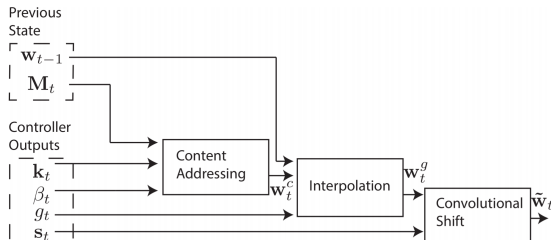
# Location-based addressing



Interpolation gate  $g_t$ , decides how much of the content-based weighting is preserved.

$$\mathbf{w}_t^g = g_t \mathbf{w}_t^c + (1 - g_t) \mathbf{w}_{t-1}$$

# Location-based addressing

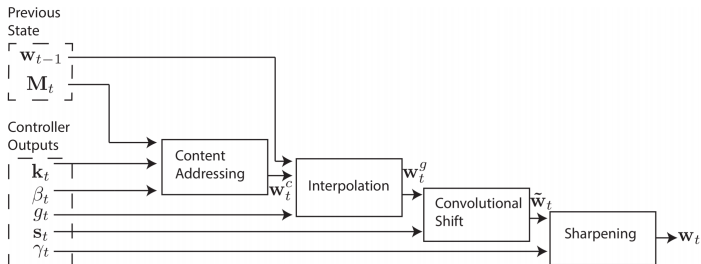


Convolutional shift (as in Turing machines)

$$\tilde{w}_t(i) = \sum_{j=0}^{N-1} w_t^g(j) s_t(i-j)$$

the shift weighting  $\mathbf{s}_t$  is emitted by the controller and is a distribution over possible shifts.

# Location-based addressing



Sharpening,

$$w_t(i) = \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$

this operation is useful when the shift weighting is not sharp.

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t$$



# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t, g_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t, \mathbf{g}_t, \mathbf{s}_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t, \mathbf{g}_t, \mathbf{s}_t, \gamma_t$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t, \mathbf{g}_t, \mathbf{s}_t, \gamma_t\})^r$$

# Controller network

- ▶ It can be a recurrent or a feedforward neural network.
- ▶ it takes as input a vector  $\mathbf{x}_t$  and the memory  $M_t$
- ▶ the output is

$$y_t = (\mathbf{a}_t, \mathbf{e}_t, \{\mathbf{k}_t, \beta_t, \mathbf{g}_t, \mathbf{s}_t, \gamma_t\}^r, \{\mathbf{k}_t, \beta_t, \mathbf{g}_t, \mathbf{s}_t, \gamma_t\}^w)$$

- ▶ the emissions for the write and read head, and the erase and add vectors are adjusted to meet the constraints.

# Experiments

Can a neural network learn procedures/programs?

- ▶ copy task
- ▶ repeat copy
- ▶ associative recall
- ▶ sorting

On all these tasks NTM outperforms LSTM.

# Copy task demo

`https:  
//thumbs.gfycat.com/WelllitInferiorAndeancondor-mobile.mp4`



# Extensions

Program induction papers:

- ▶ Neural Programmer: Inducing Latent Programs with Gradient Descent
- ▶ Neural Programmer-Interpreters
- ▶ Reinforcement Learning Neural Turing Machines - Revised
- ▶ Neural Random-Access Machines
- ▶ Neural GPUs Learn Algorithms

Memory networks extensions:

- ▶ Ask Me Anything: Dynamic Memory Networks for Natural Language Processing
- ▶ The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations

# Lecture recap

- ▶ Memory networks
- ▶ End-to-end memory networks
- ▶ Neural Turing machines