Neural Networks with Memory

Memory Networks, End-to-End Memory Networks, Condensed Memory Networks, Neural Turing Machine & Differential Neural Computer

Aaditya Prakash
Nov 5, 2018

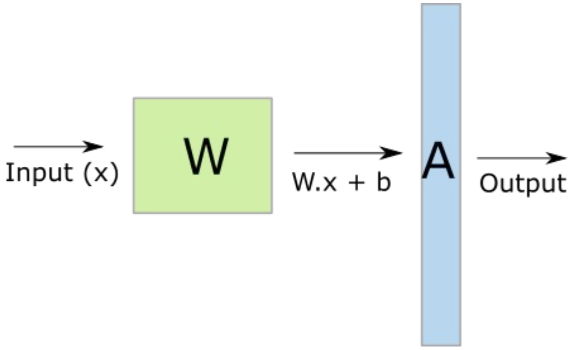
Why?

Siri, play me a good song

Sorry, I couldn’t find a ‘good song’ in your music.

Because AI is good only with context !

Neural Network - limitation



1. Not all problems can be mapped to y = f(x)
2. Need to remember external context
3. Need to know where to look for in the context
4. Need to know what to look for in the context
5. Need to know how to reason using external context
6. Need to handle potentially changing external context

Text source: Sumit Chopra FAIR, DLSS 2016

Why not RNN ?

1. Inherent temporal structure (not all knowledge is temporal)

2. Very slow (because of sequential learning, not embarrassingly parallel)

3. Cannot scale to large network (overhead in terms of learning)

4. Prone to “vanishing/exploding gradient” (multiply gradient many times)

5. Back-propagation in time is hard (always needs truncation and clipping)

Reasoning is hard

Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring. Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring. Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died. Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.

Where is the ring?

Where is Bilbo now?

Where is Frodo now?

Reasoning is hard

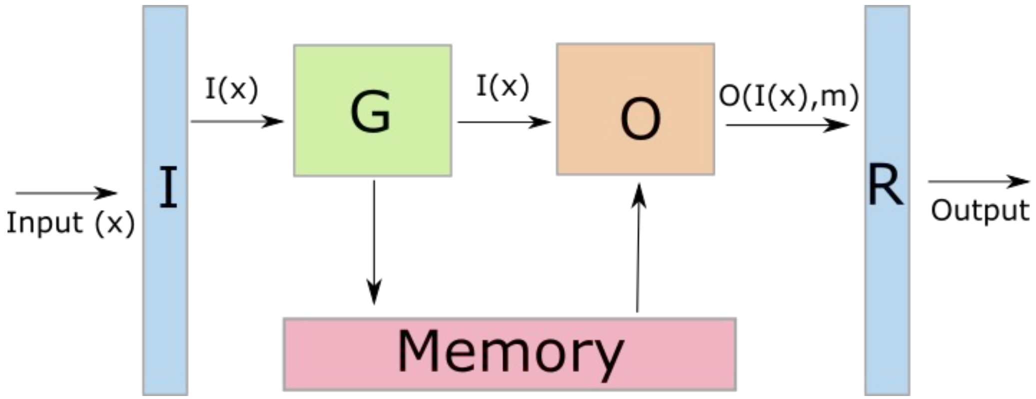
Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring.
Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring.
Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.

Where is the ring? A: Mount-Doom

Where is Bilbo now? A: Grey-havens

Where is Frodo now? A: Shire

Memory Networks



I Convert the input (x) to internal representation (aka feature space)
- learn embeddings (bow/gru/lstm), use pre-trained vectors (glove/w2v)

G Make a memory state (u), write initial data to the memory, update when necessary

O Generate the output state, which is combination of memory state and input

R Convert the output (O) into response as desired by the model

Memory Networks

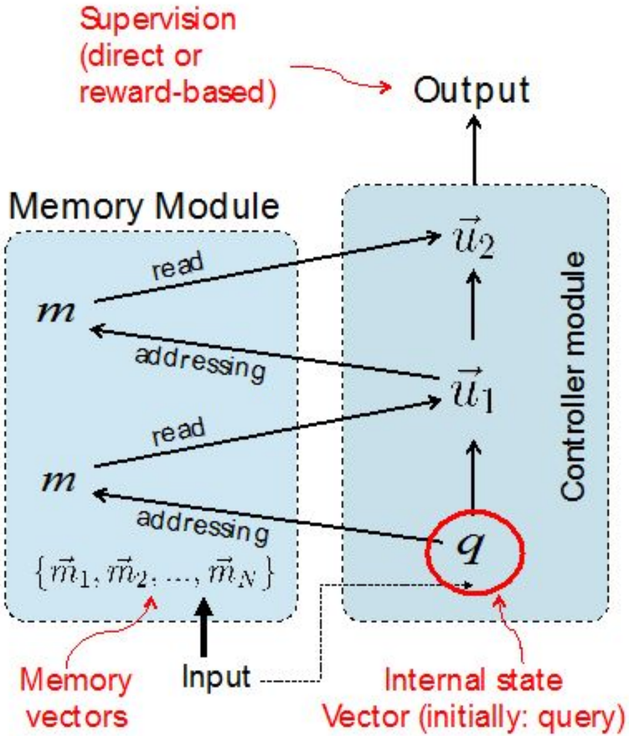


Figure: Saina Sukhbaatar

Memory Networks

bABI tasks ---

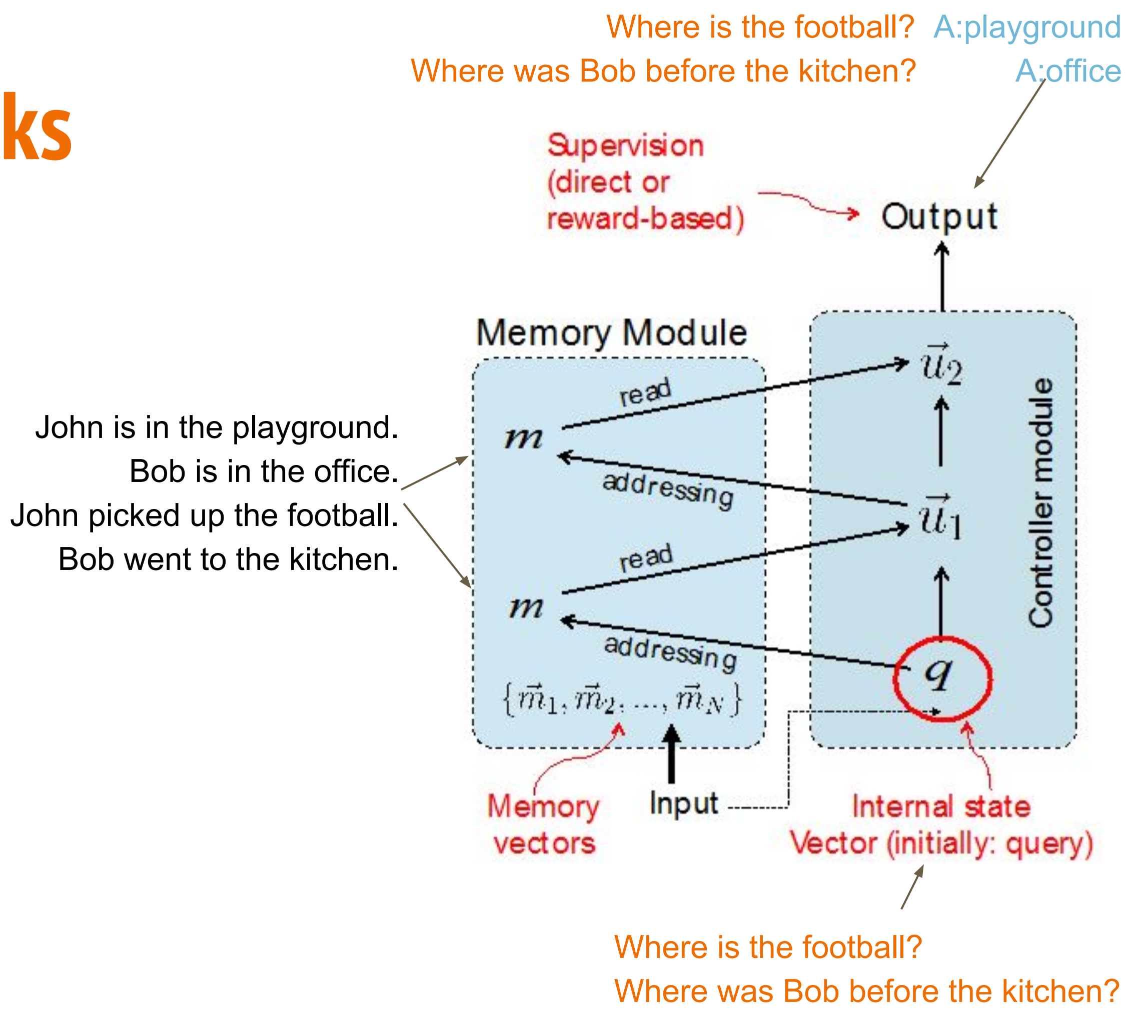
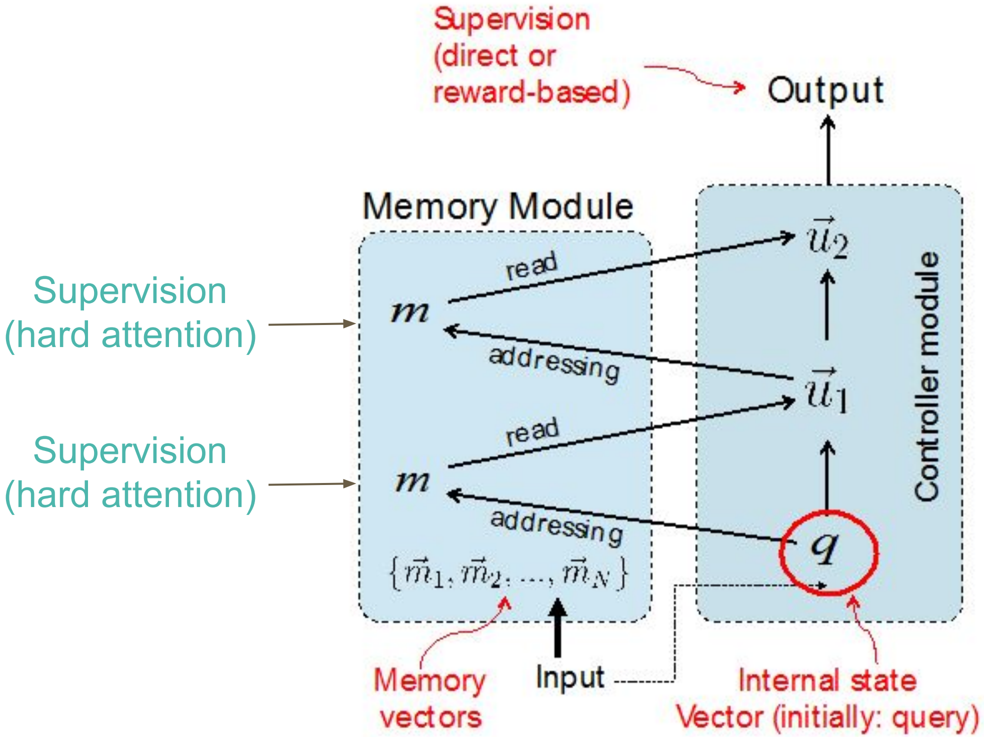


Figure: Saina Sukhbaatar

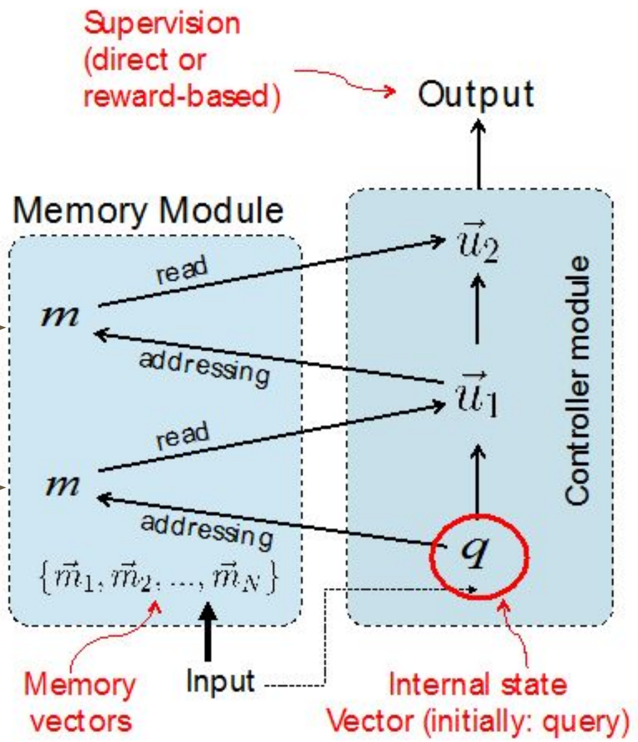
Hard Attention (Strong supervision)



Soft Attention (Weak supervision)

~~Supervision~~
(soft attention)

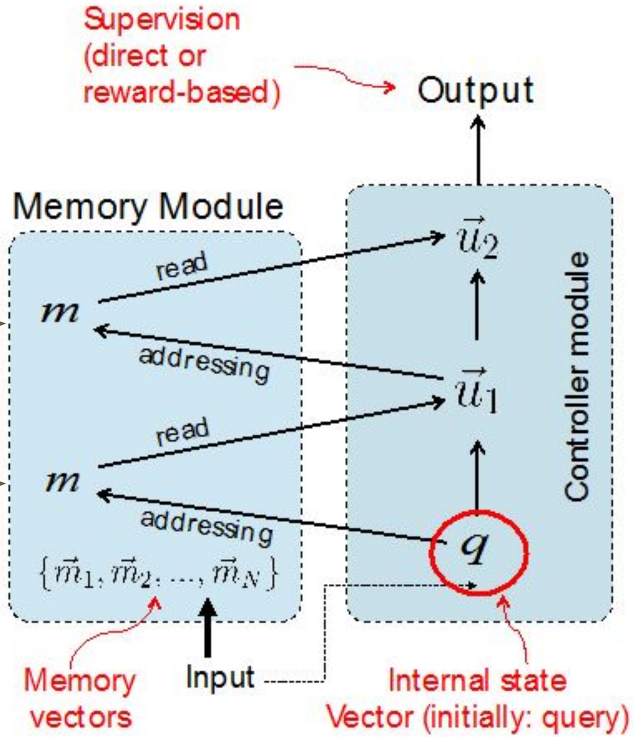
~~Supervision~~
(soft attention)



Soft Attention (Weak supervision)

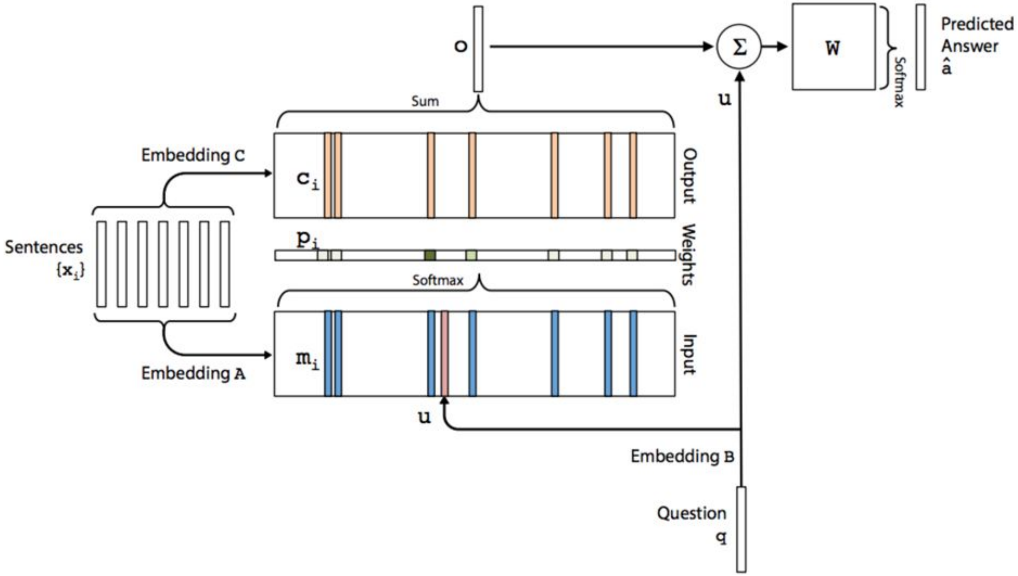
~~Supervision~~
(soft attention)

~~Supervision~~
(soft attention)

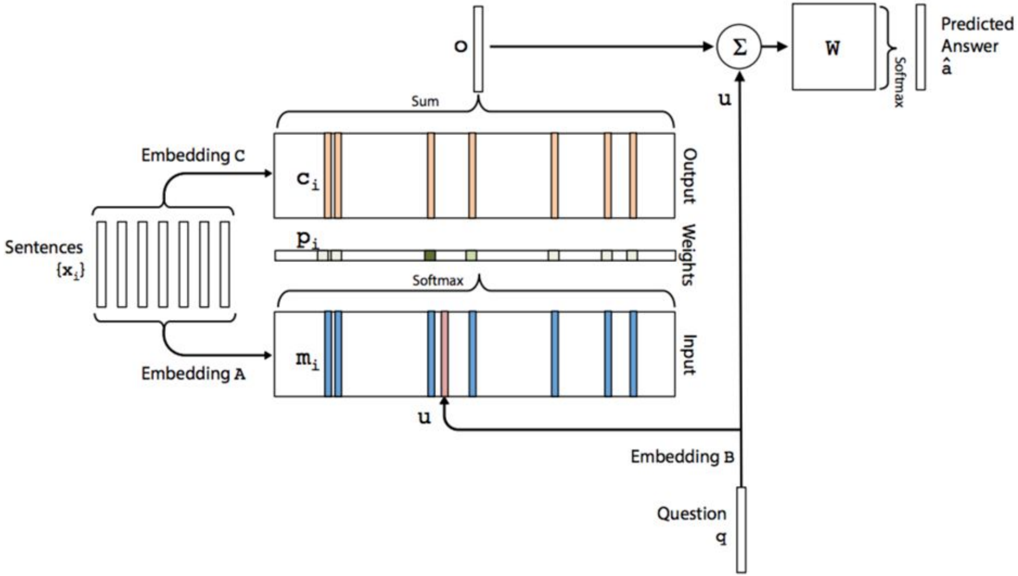


Now applicable to more problems where it is hard to get strong supervision

End to End Memory Networks



End to End Memory Networks



$$x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$$

$$m_i = \sum_j Ax_{ij}$$

$$p_i = softmax(u^T m_i)$$

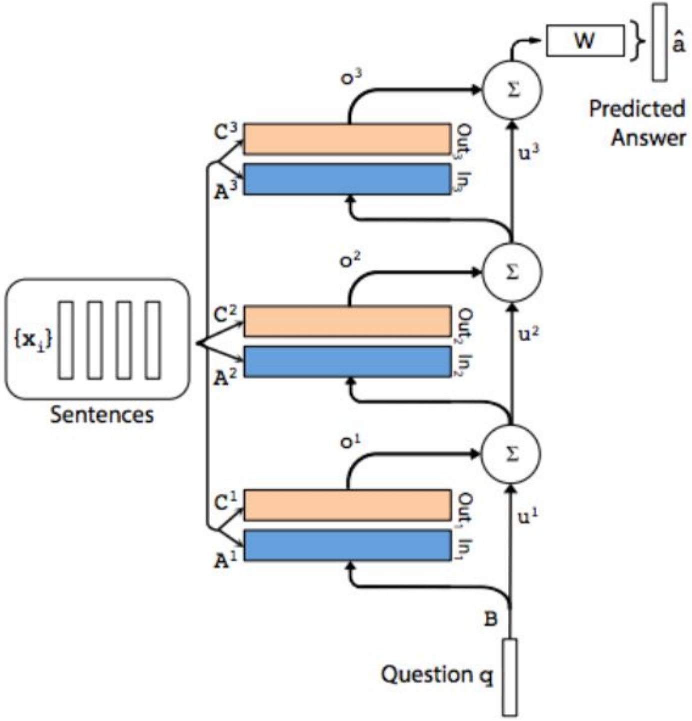
$$= softmax(q^T B^T \sum_j Ax_{ij})$$

$$c_i = \sum_j Cx_{ij}$$

$$o = \sum_i p_i c_i$$

$$a = softmax(W(o+u))$$

One hop is just not sufficient

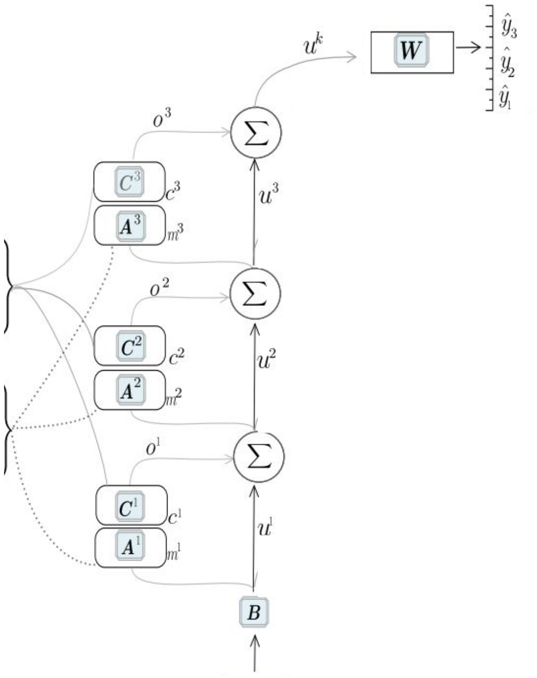


$$u^{k+1} = u^k + o^k$$

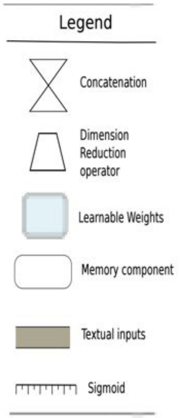
Each layer has its own embedding matrices A^k, C^k

$$\hat{a} = \mathrm{Softmax}(Wu^{K+1}) = \mathrm{Softmax}(W(o^K + u^K))$$

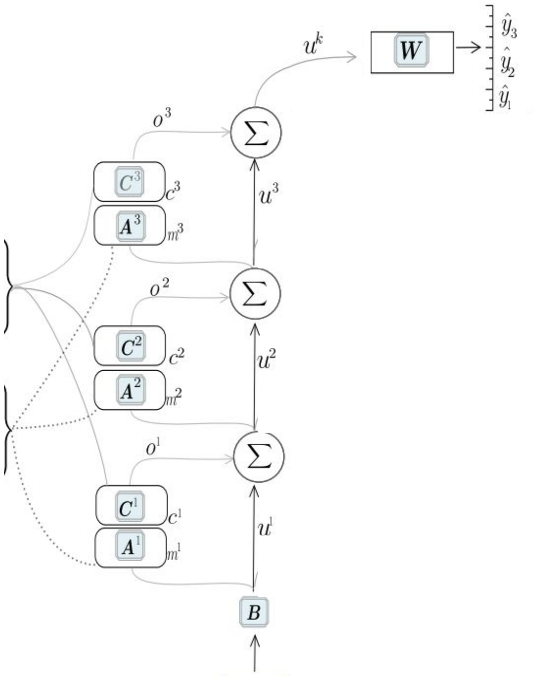
Condensed Memory Network [C-MemNN]



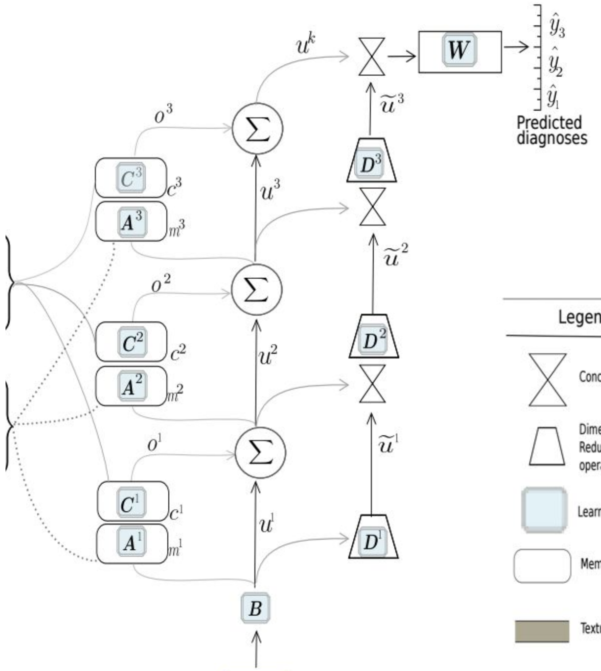
MemNN



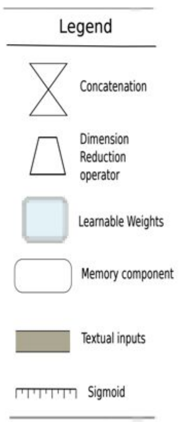
Condensed Memory Network [C-MemNN]



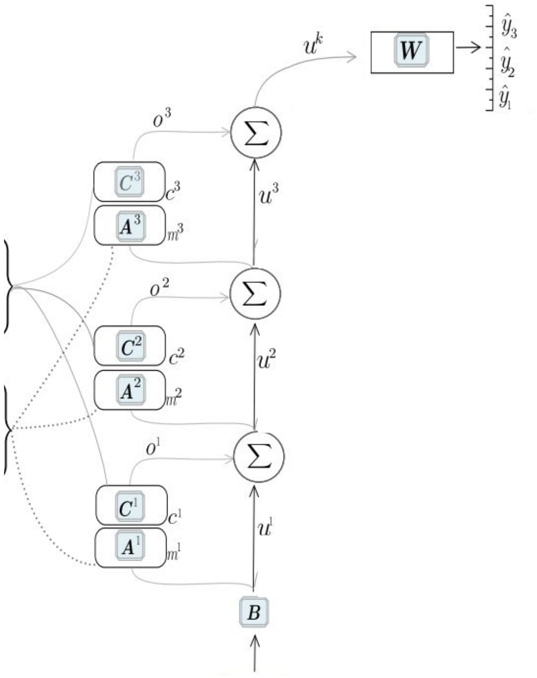
MemNN



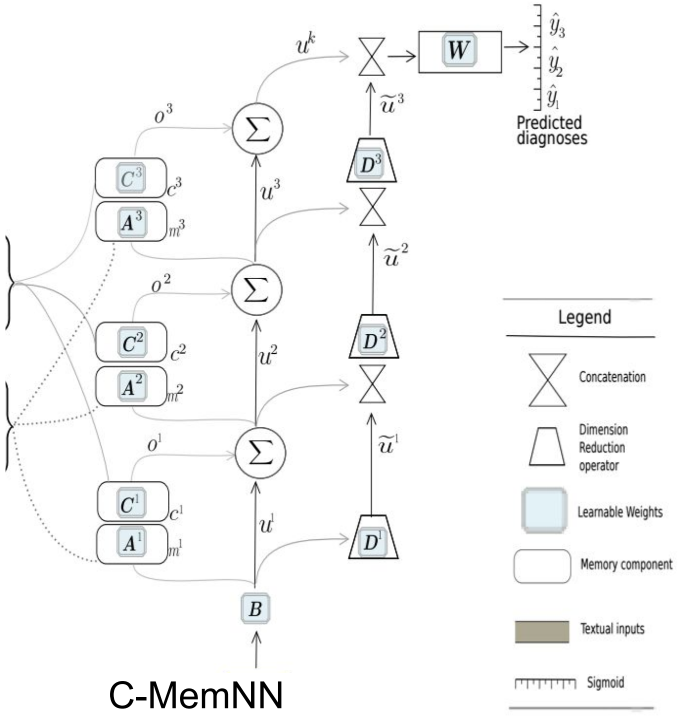
C-MemNN



Condensed Memory Network [C-MemNN]



MemNN



C-MemNN

C-MemNN

$$o^k = \sum_i \text{Addressing}(u^k, m_i^k) \cdot c_i^k$$

$$u^{k+1} = u^k + o^k$$

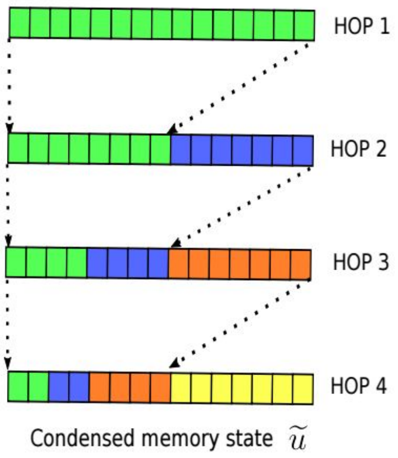
$$\tilde{u}^{k+1} = u^{k+1} \oplus \boldsymbol{D_1} \cdot \tilde{u}^k$$

$$\hat{y_r} = \arg\max_{r \epsilon R} \frac{1}{1 + e^{-1*(\tilde{u}^{k+1} \cdot \boldsymbol{W})}}$$

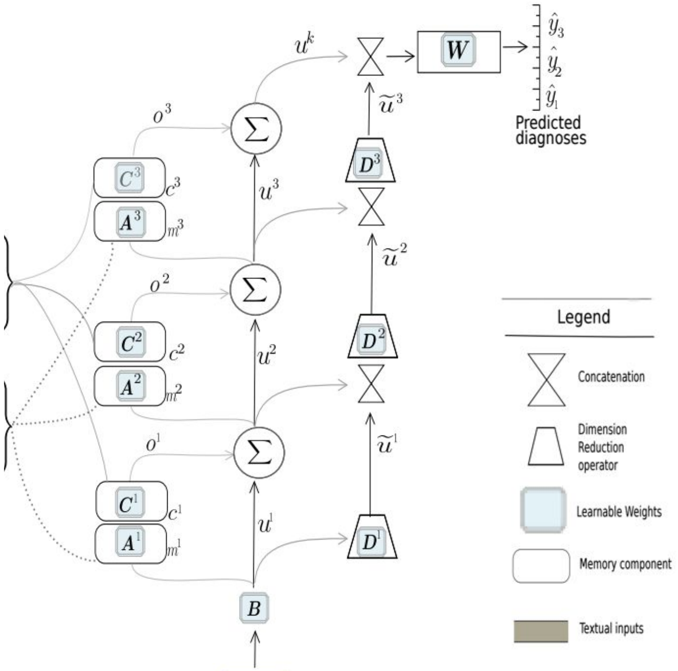
A-MemNN

$$\tilde{u}^{k+1} = \tilde{u}^k + \frac{\tilde{u}^{k-1}}{2} + \frac{\tilde{u}^{k-2}}{4} + \dots$$

Condensed Memory Network [C-MemNN]



Condensation of memory state - continuous hierarchy.



C-MemNN

C-MemNN

$$o^k = \sum_i \text{Addressing}(u^k, m_i^k) \cdot c_i^k$$

$$u^{k+1} = u^k + o^k$$

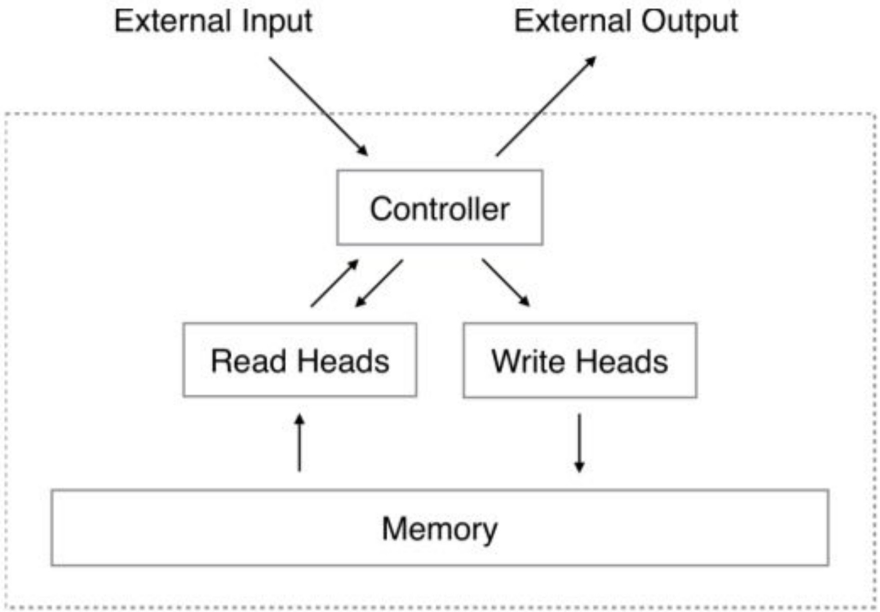
$$\widetilde{u}^{k+1} = u^{k+1} \oplus \boldsymbol{D_1} \cdot \widetilde{u}^k$$

$$\hat{y}_r = \arg\max_{r \epsilon R} \frac{1}{1 + e^{-1*(\widetilde{u}^{k+1} . \boldsymbol{W})}}$$

A-MemNN

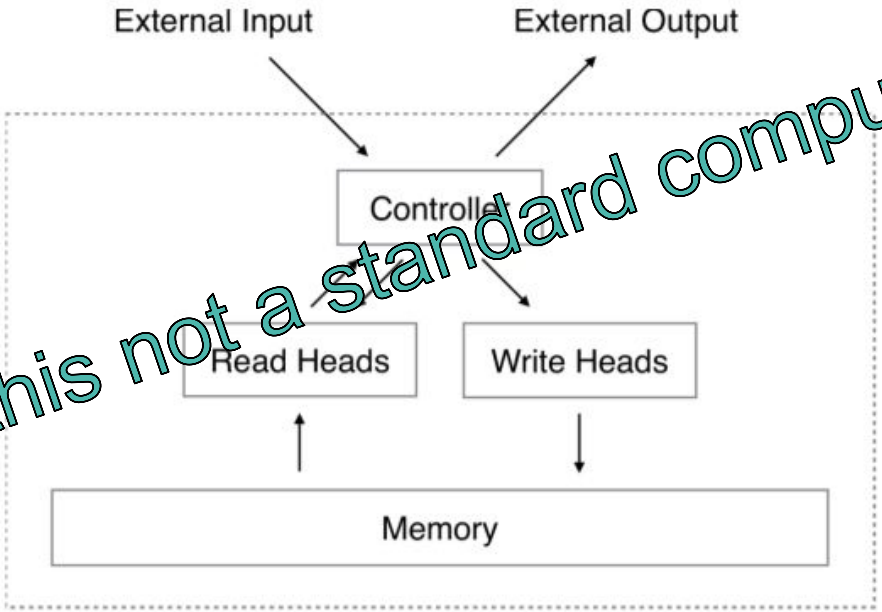
$$\widetilde{u}^{k+1} = \widetilde{u}^k + \frac{\widetilde{u}^{k-1}}{2} + \frac{\widetilde{u}^{k-2}}{4} + \dots$$

Neural Turing Machine



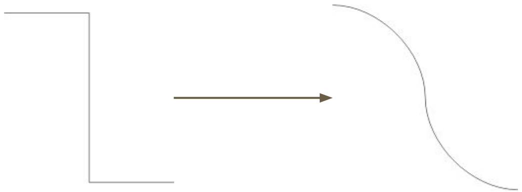
NTM can learn basic algorithms like sorting.

Neural Turing Machine

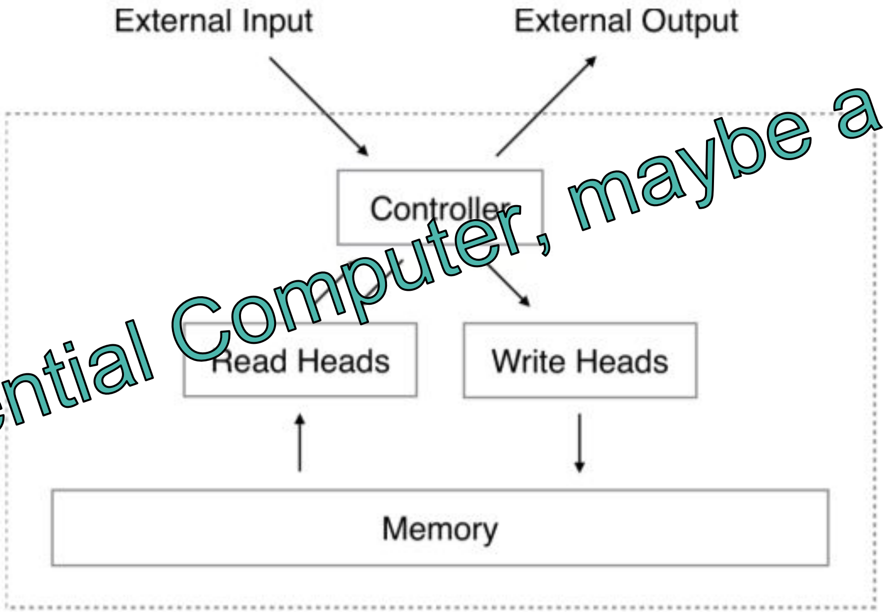


Is this not a standard computer?

Sharp functions made smooth. Now can be trained with 'backpropagation'

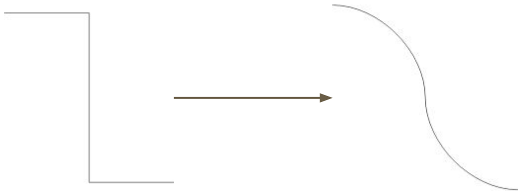


Neural Turing Machine



So Differential Computer, maybe a better name?

Sharp functions made smooth. Now can be trained with 'backpropagation'



Neural Turing Machine - Selective Attention

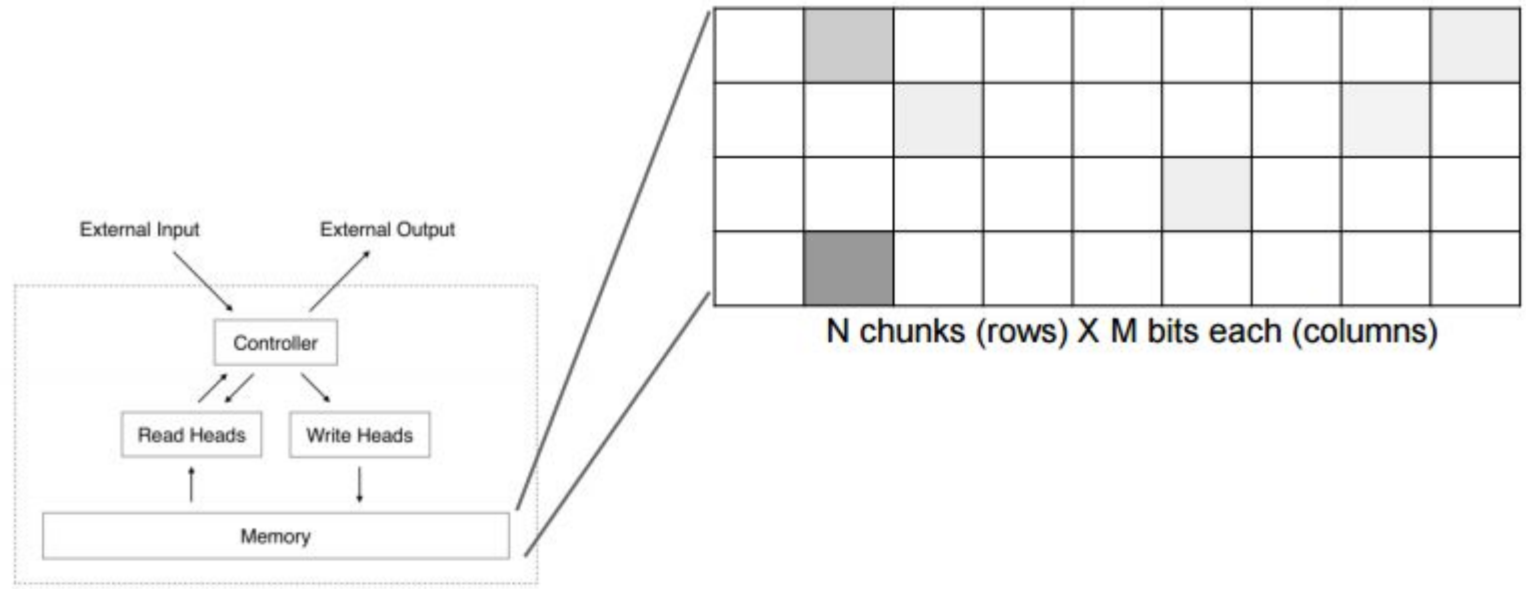
- Focus on the parts of the memory the network will read and write to:
  'An introspective attention model'

- Use the controller outputs to parameterise a distribution (weights) over the rows (slots) in the memory matrix

- Weights are defined by two main attention mechanism:
  - One based on content
  - One based on location

Source: Alex Graves

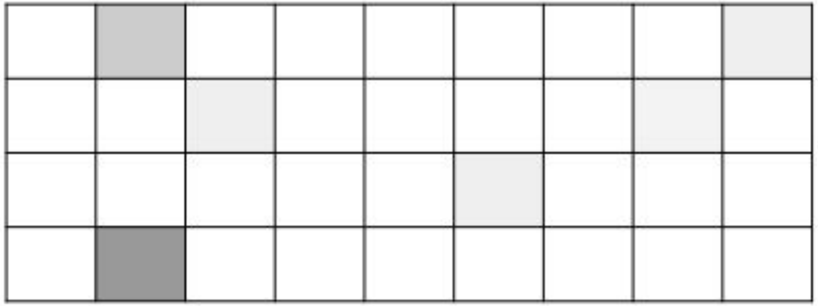
Neural Turing Machines - Memory



Neural Turing Machines - Memory

Read from memory (“blurry”)

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$$



N chunks (rows) X M bits each (columns)

Write to memory (“blurry”)

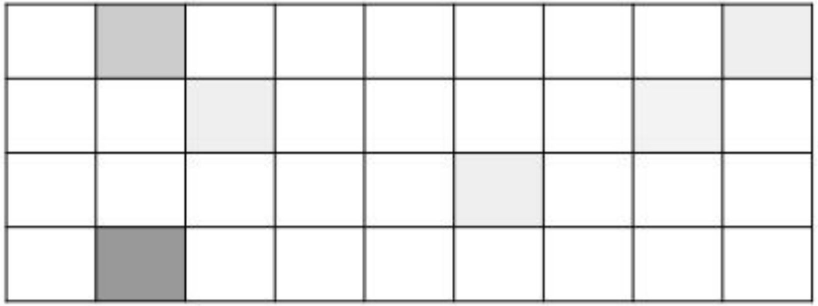
$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right]$$

$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Neural Turing Machines - Memory

Read from memory ("blurry")

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$$



N chunks (rows) X M bits each (columns)

Write to memory ("blurry")

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right]$$

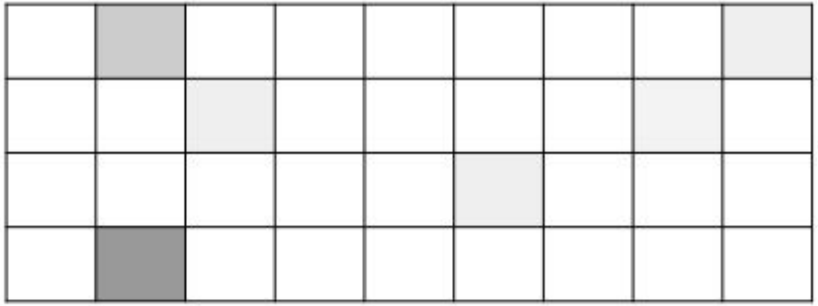
$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Forget gate !!
Déjà vu

Neural Turing Machines - Memory

Read from memory ("blurry")

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$$



N chunks (rows) X M bits each (columns)

Write to memory ("blurry")

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right]$$
$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Forget gate !!
Déjà vu

LSTM

Neural Turing Machines - Memory

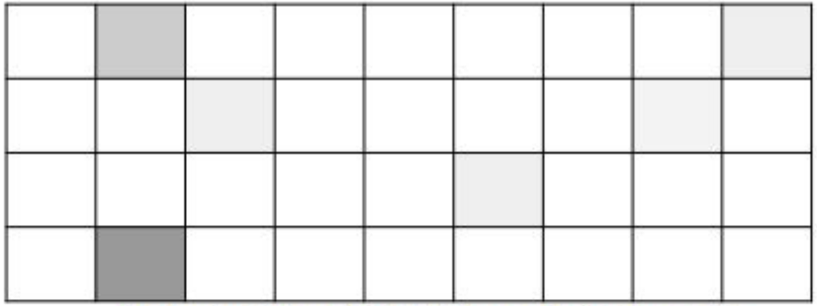
Addressing by content (similarity)

$$w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}$$

$$K\big[\mathbf{u}, \mathbf{v}\big] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

Addressing by location (shift)

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j) \qquad w_t(i) \longleftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$



N chunks (rows) X M bits each (columns)

Neural Turing Machines - Memory

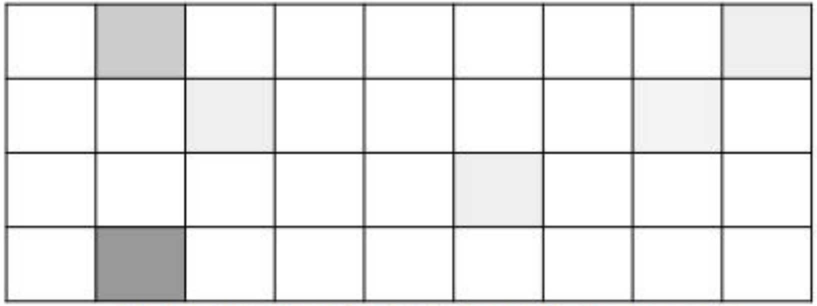
Addressing by content (similarity)

$$w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}$$

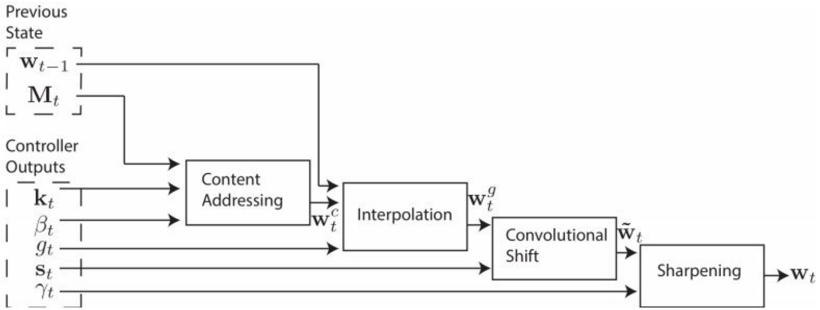
$$K\big[\mathbf{u}, \mathbf{v}\big] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

Addressing by location (shift)

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j) \qquad w_t(i) \longleftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$



N chunks (rows) X M bits each (columns)

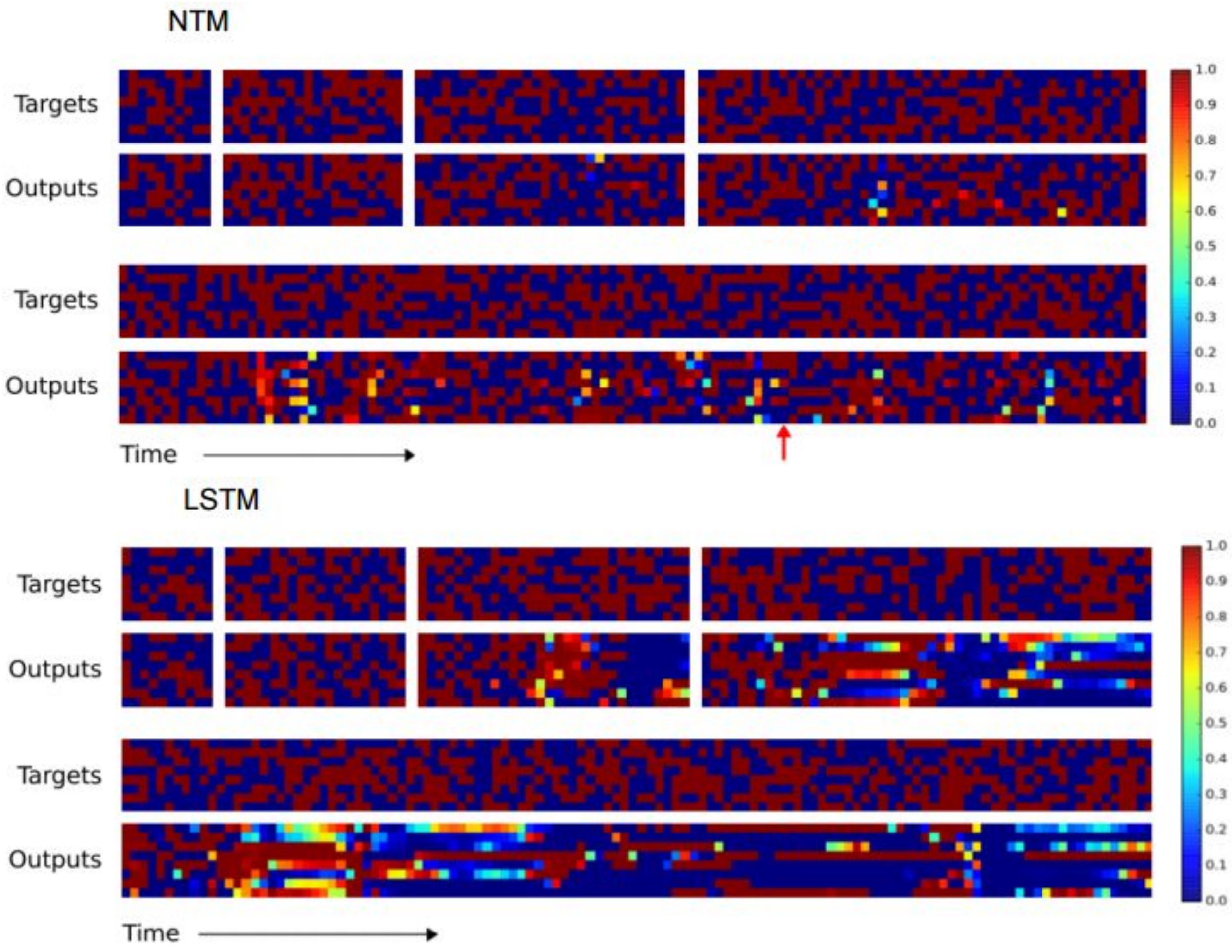


The key vector k_t, & key strength β_t, are used to perform content-based addressing of the memory matrix M_t. The resulting content-based weighting is interpolated with the weighting from the previous time step based on the value of the interpolation gate g_t. The shift weighting s_t, determines whether & by how much the weighting is rotated. Depending on γ_t, the weighting is sharpened and used for memory access.

NTM - Copy

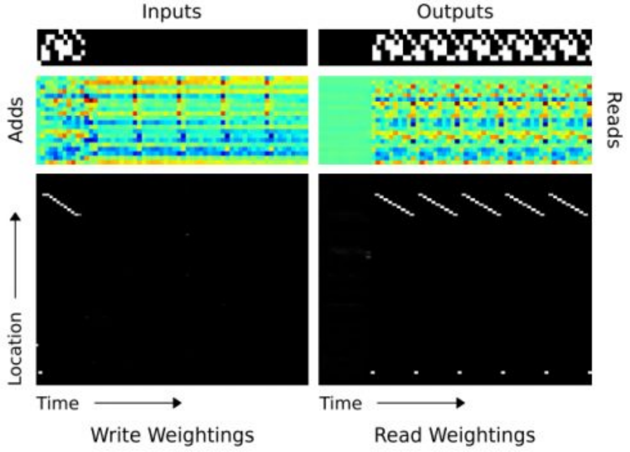
- NTM can learn basic algorithms like copy, loop, sort, associative recall, N-gram inference.

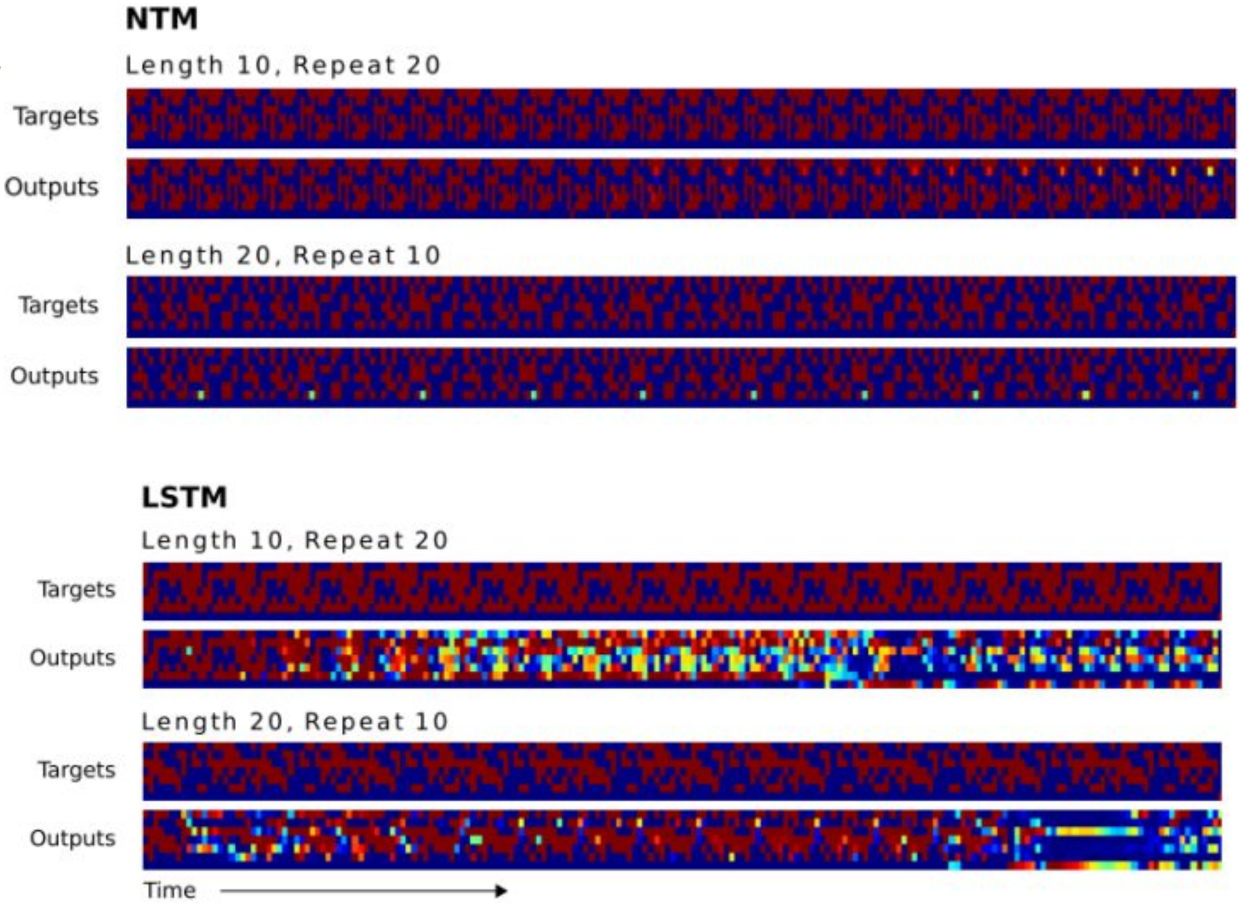
- A simple task of ‘copying’ the input back can be hard, when the time period becomes very long.



NTM - Mult Copy

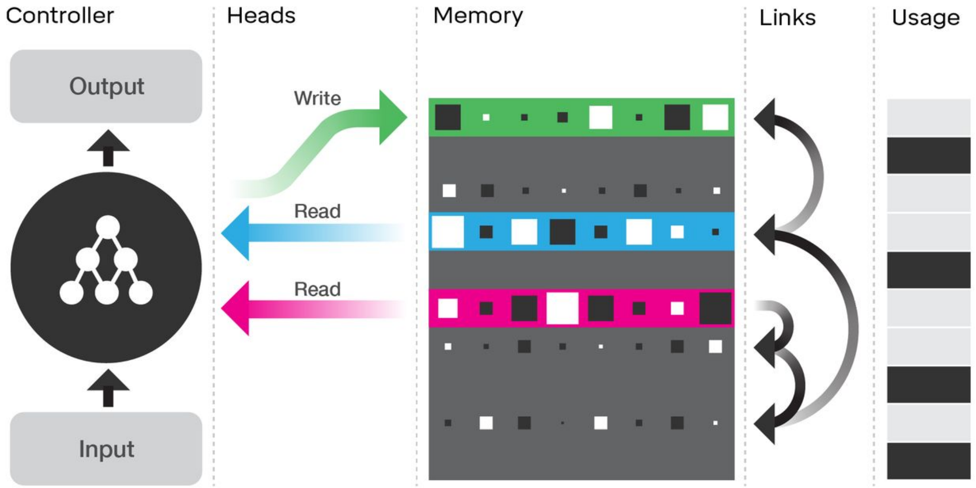
Copying the same sequence many times.



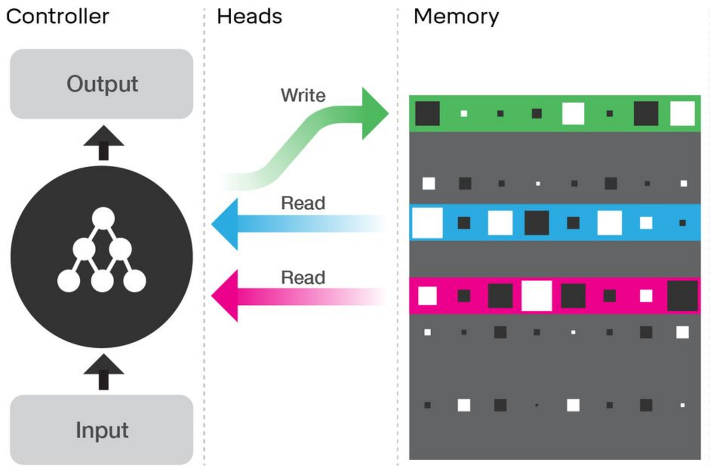


Differential Neural Computer

Architecture of DNC

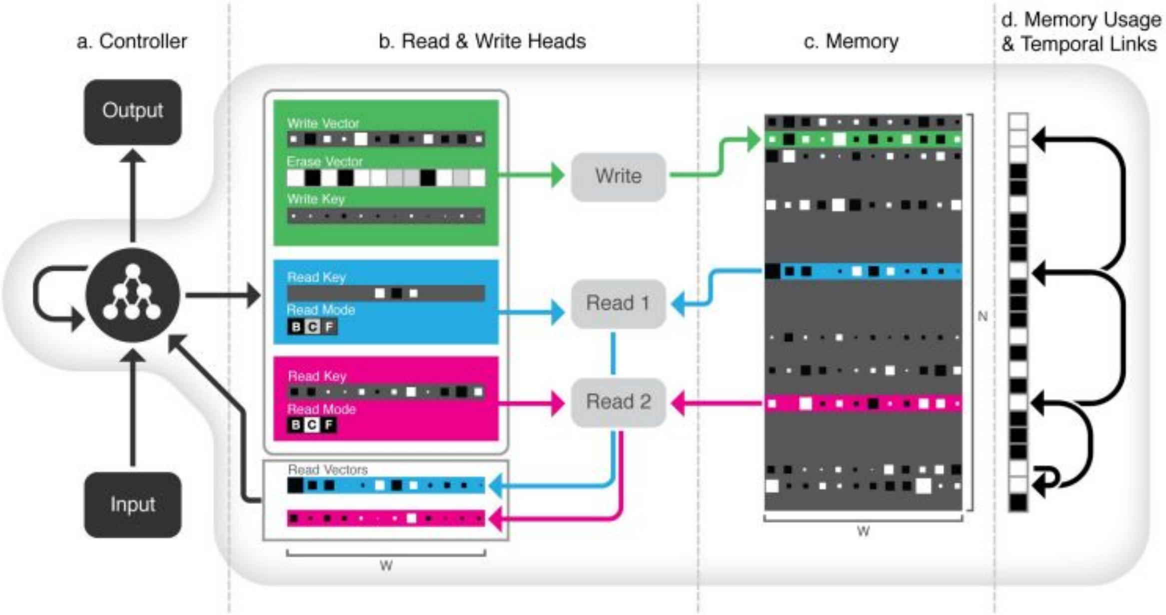


Architecture of NTM



Differential Neural Computer

Architecture of DNC



- NTM was able to retrieve memories in order of their index but not in order in which they were written
- Preserving temporal order is necessary for many tasks (e.g. sequence of instructions) and appears to play an important role in human cognition
- DNC tries to iterate through memories in the order they were written (or rewritten)
- A precedence weighting (p_t) keeps track of which locations were most recently written to:

$$\boldsymbol{p}_t = \left(1 - \sum_i \boldsymbol{w}_t^{\mathrm{w}}[i]\right)\boldsymbol{p}_{t-1} + \boldsymbol{w}_t^{\mathrm{w}}$$

Differential Neural Computer

$\boldsymbol{p}_t$ is then used to update a *temporal link matrix* (L_t), defining the order locations were written in:

$$L_t[i,j] = (1 - \boldsymbol{w}_t^{\mathrm{w}}[i] - \boldsymbol{w}_t^{\mathrm{w}}[j])L_{t-1}[i,j] + \boldsymbol{w}_t^{\mathrm{w}}[i]\boldsymbol{p}_{t-1}[j]$$

The controller can use L_t to retrieve the write before (b^i_t) or after (f^i_t) the last read location ($\boldsymbol{w}^{r,i}_{t-1}$), allowing it to iterate forwards or backwards in time

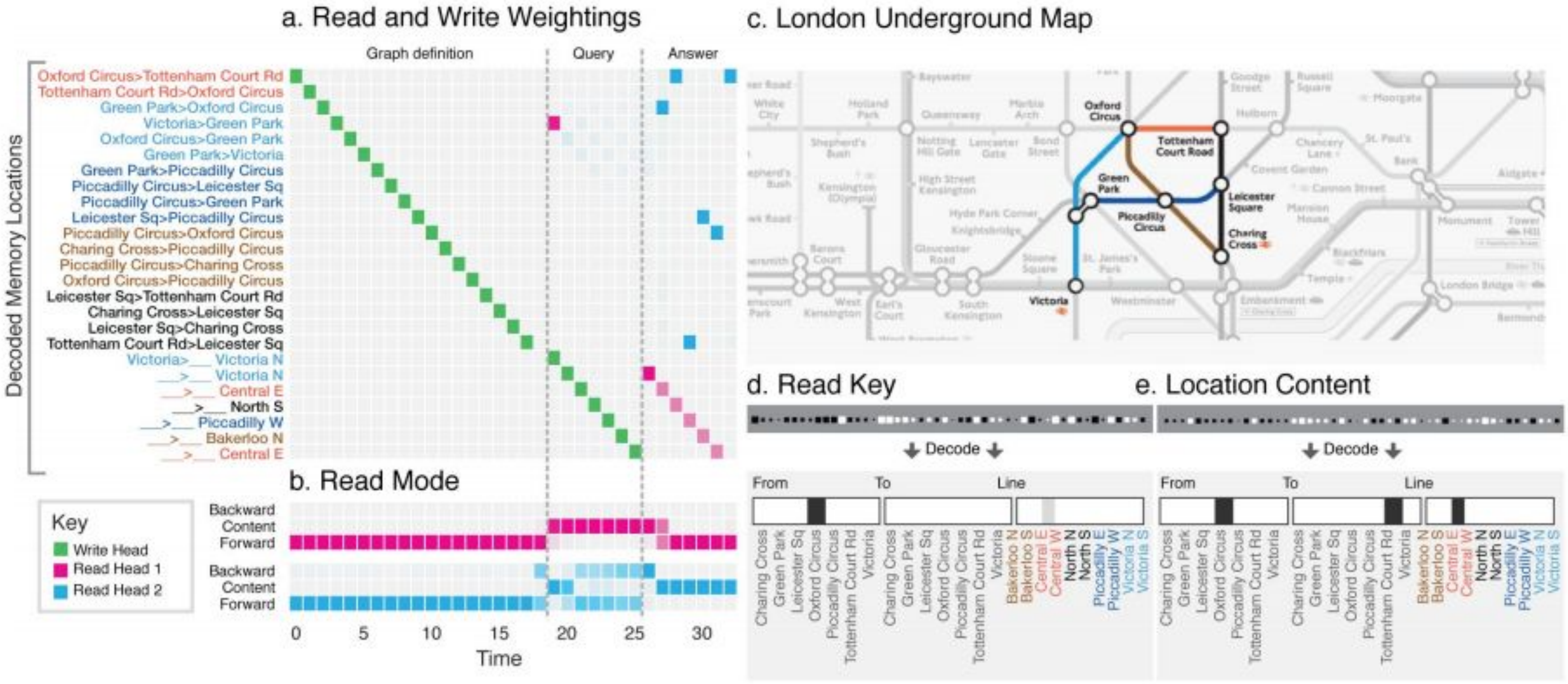
$$\boldsymbol{b}_t^i = \hat{L}_t^\top \boldsymbol{w}_{t-1}^{\mathrm{r},i} \qquad \boldsymbol{f}_t^i = \hat{L}_t \boldsymbol{w}_{t-1}^{\mathrm{r},i}$$

Finally three-way gates ($\boldsymbol{\pi}^i_t$) are used to interpolate among iterating forwards, backwards, or reading by content:

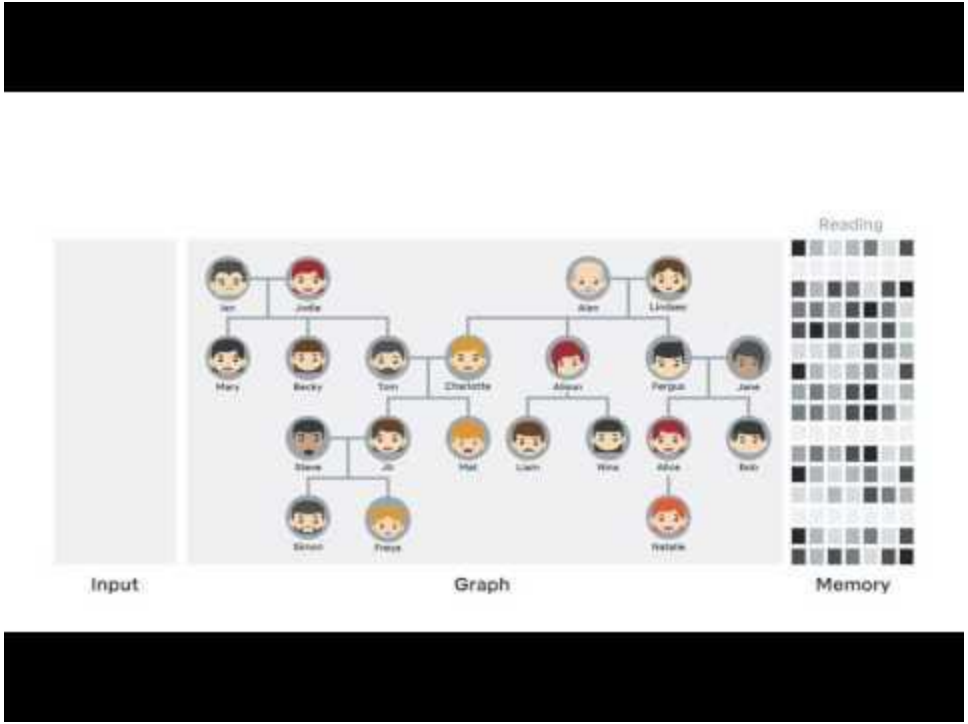
$$\boldsymbol{w}_t^{\mathrm{r},i} = \boldsymbol{\pi}_t^i[1]\boldsymbol{b}_t^i + \boldsymbol{\pi}_t^i[2]\boldsymbol{c}_t^{\mathrm{r},i} + \boldsymbol{\pi}_t^i[3]\boldsymbol{f}_t^i$$

- RNNs are great at processing sequences: text, audio, time-series…
- But graph-structured data is even more general: maps, molecules, parse-trees, knowledge graphs, social networks…
- DNC is able to interpret and answer questions about graphs, even if presented in sequential form
- Started with explicit graphs, but ultimately interested in implicit graphs: relations in natural language, scene parsing, agent's surroundings…

How DNC Reads Graphs



Differential Neural Computer



Relevant papers

1. J. Weston, S. Chopra, A. Bordes. Memory Networks. ICLR 2015 (and arXiv:1410.3916).
2. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. End-To-End Memory Networks. NIPS 2015
3. J. Weston, et al. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks
4. A. Bordes, N. Usunier, S. Chopra, J. Weston. Large-scale Simple Question Answering with Memory Networks
5. Alex Graves, et al Neural Turing Machine. Nature 2015
6. Prakash, et al. Condensed Memory Networks. AAAI 2017
7. Alex Graves, et al Differential Neural Computers. Nature 2017

Source materials (also good for tutorials)

1. Jason Weston, Memory Networks
2. Daniel Shank, NTM
3. DeepMind, DNC
4. J. Schmidhuber, How to learn an algorithm
5. Alex Graves, RNN Symposium on DNC