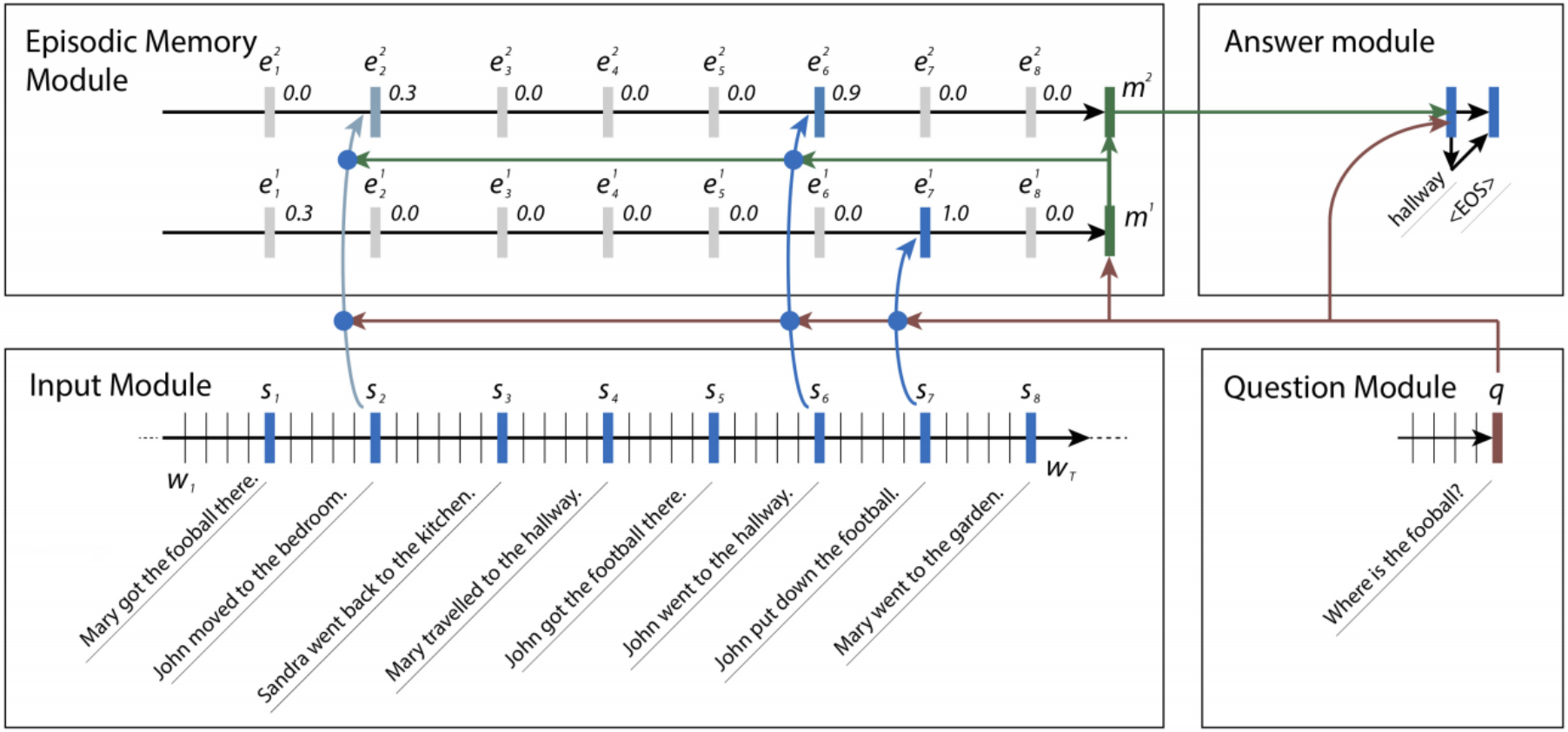
Neural Architectures with Memory

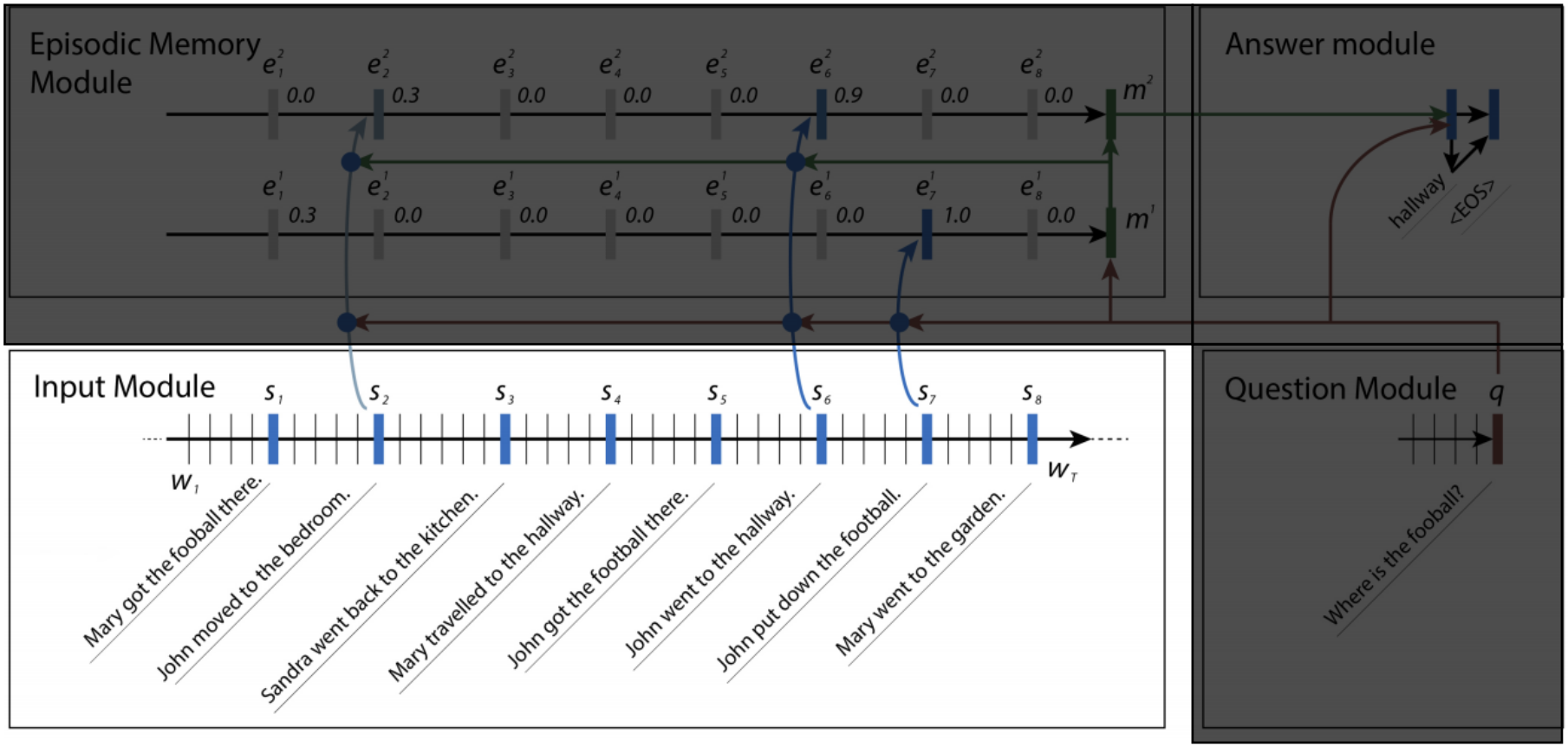
By Mohit Kumar(DeepMind)

Dynamic Memory Networks – **The Beast**



Use RNNs, specifically GRUs for every module

DMN

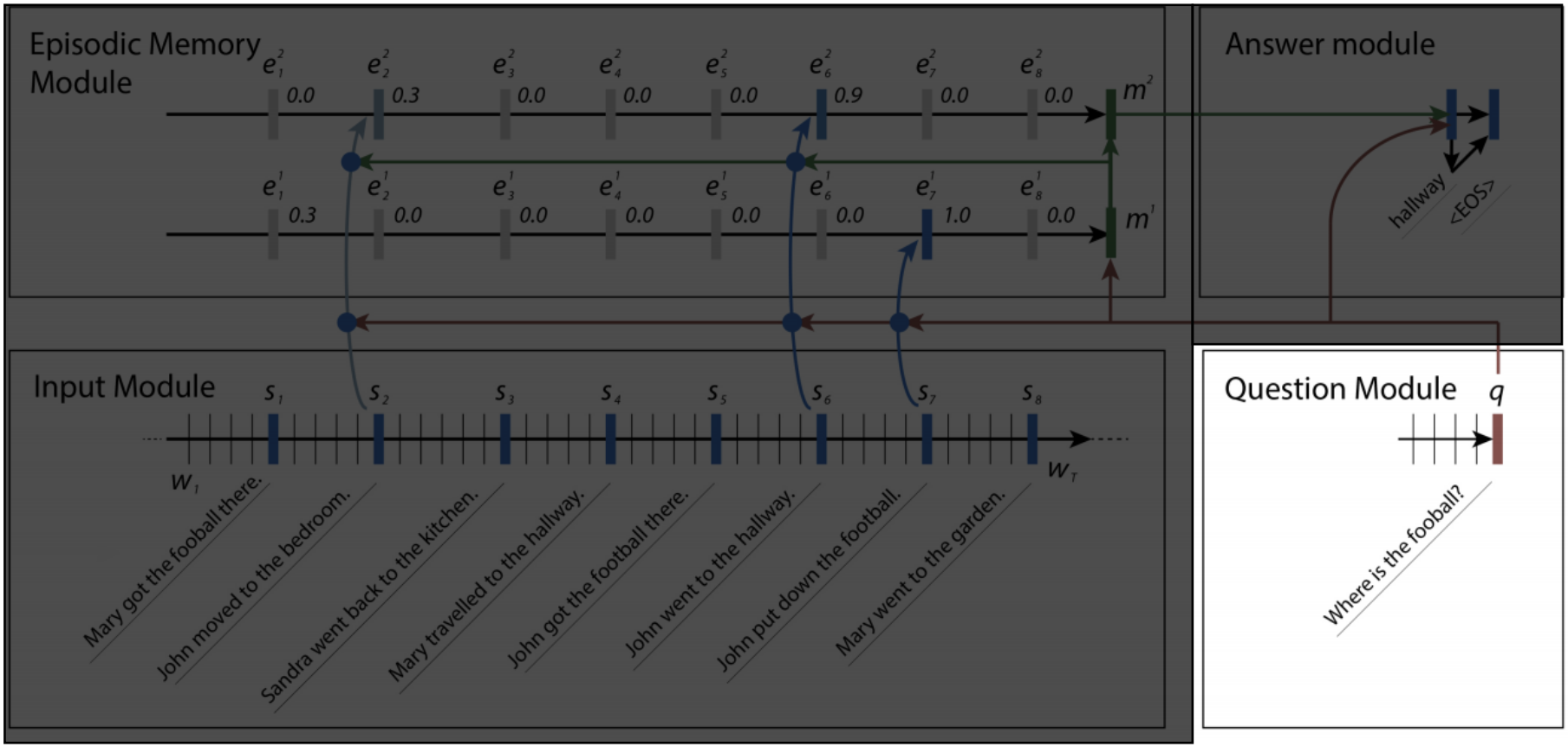


Final GRU Output for t^{th} sentence

Output for sentence

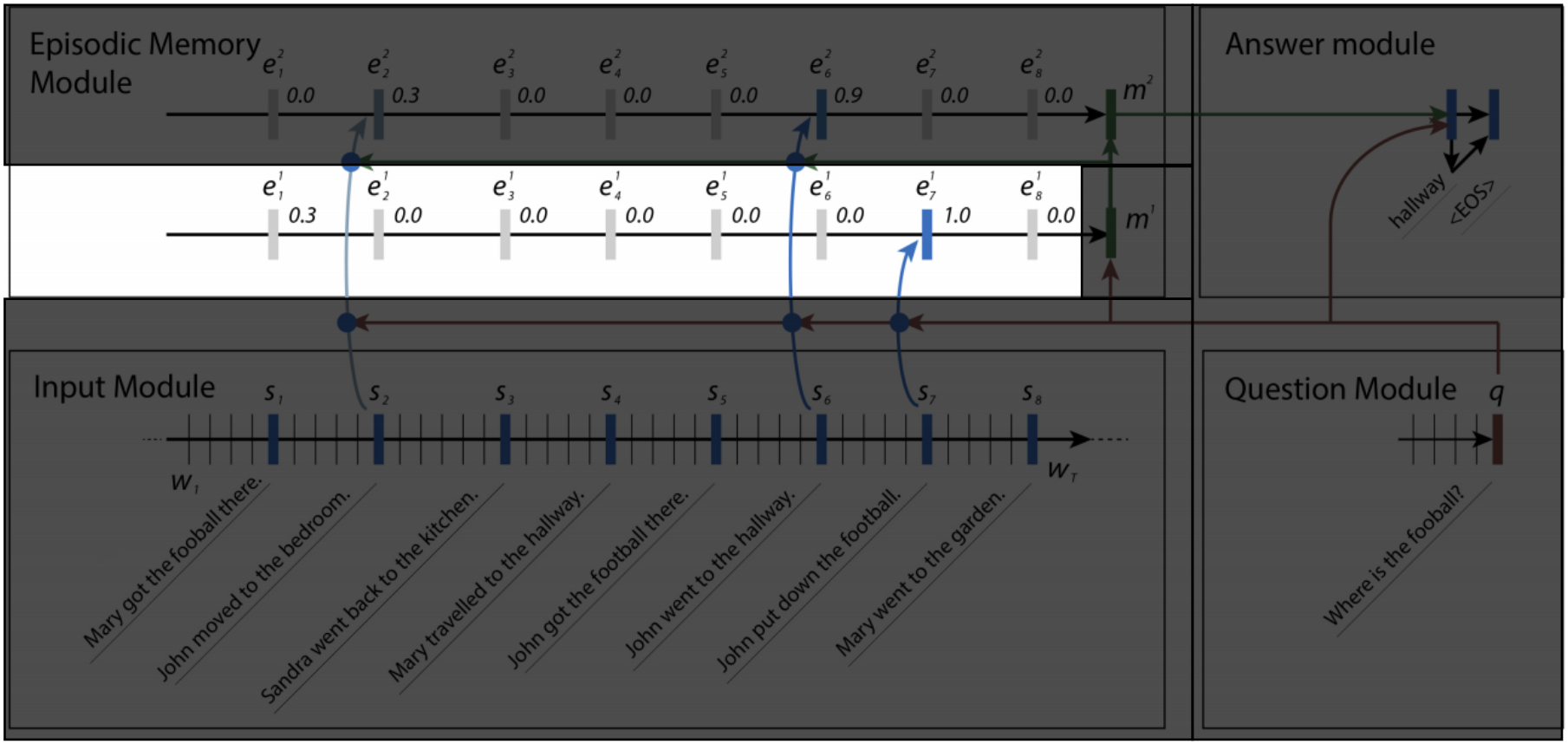
$$c_t = \mathrm{GRU}(w_t^i, c_t^{i-1})$$

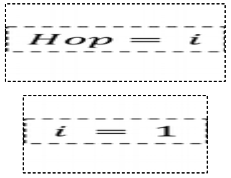
DMN



$$q = \mathrm{GRU}(q_w^i, q^{i-1})$$

DMN

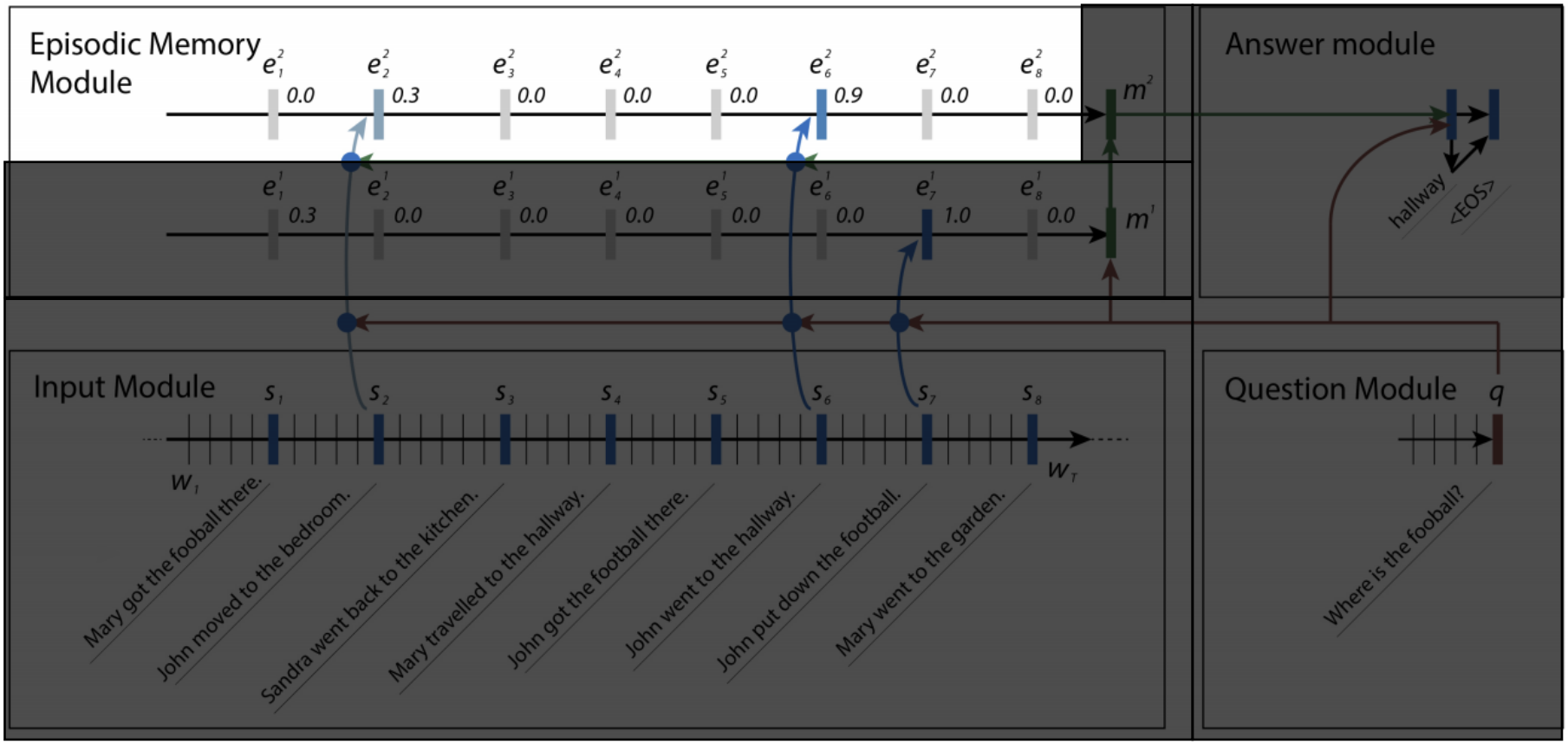




$$h_t^i = g_t^i \mathrm{GRU}(c_t, h_{t-1}^i) + (1 - g_t^i) h_{t-1}^i$$

$$e^i = h_{T_C}^i$$

DMN



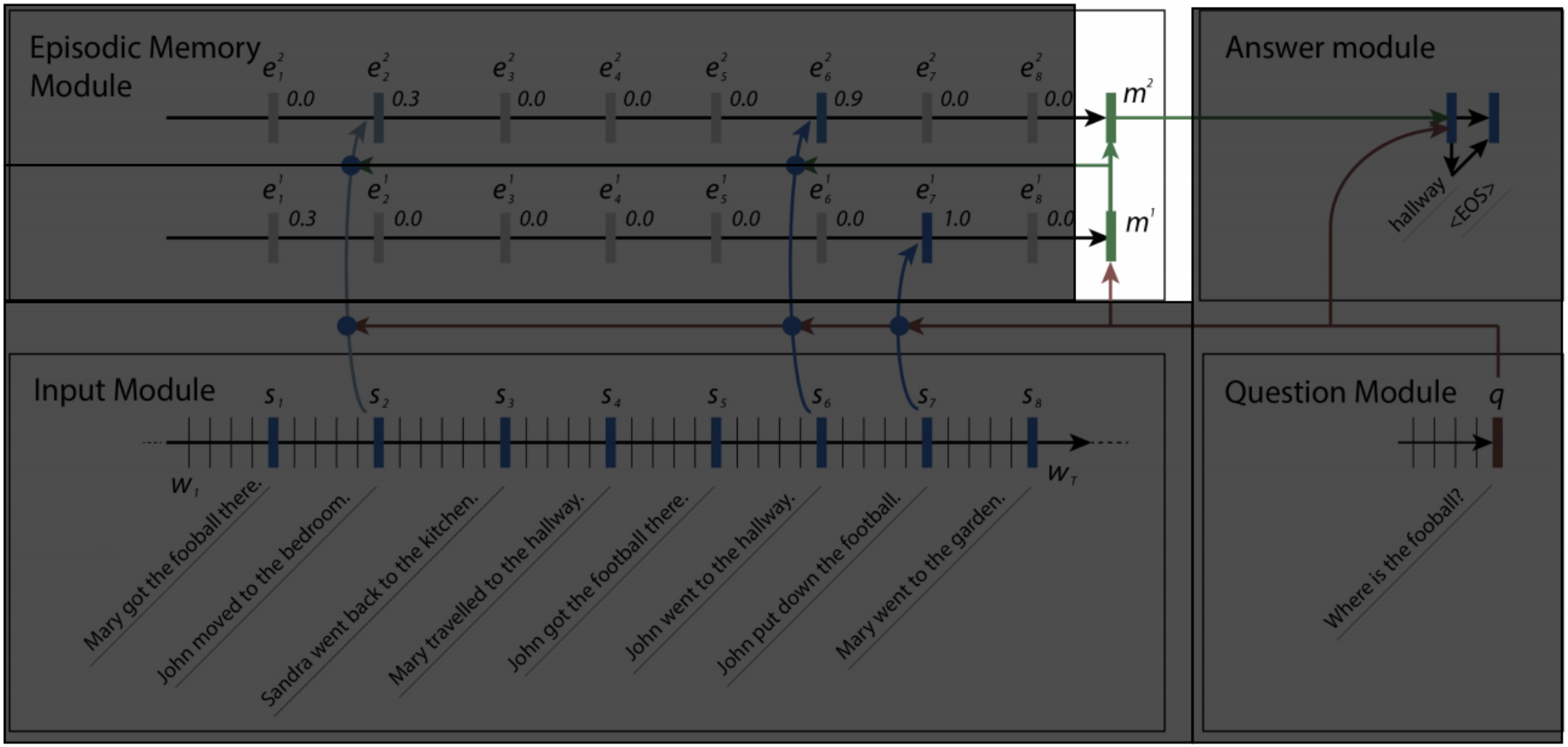
$Hop = i$

$i = 2$

$$h_t^i = g_t^i \mathrm{GRU}(c_t, h_{t-1}^i) + (1 - g_t^i) h_{t-1}^i$$

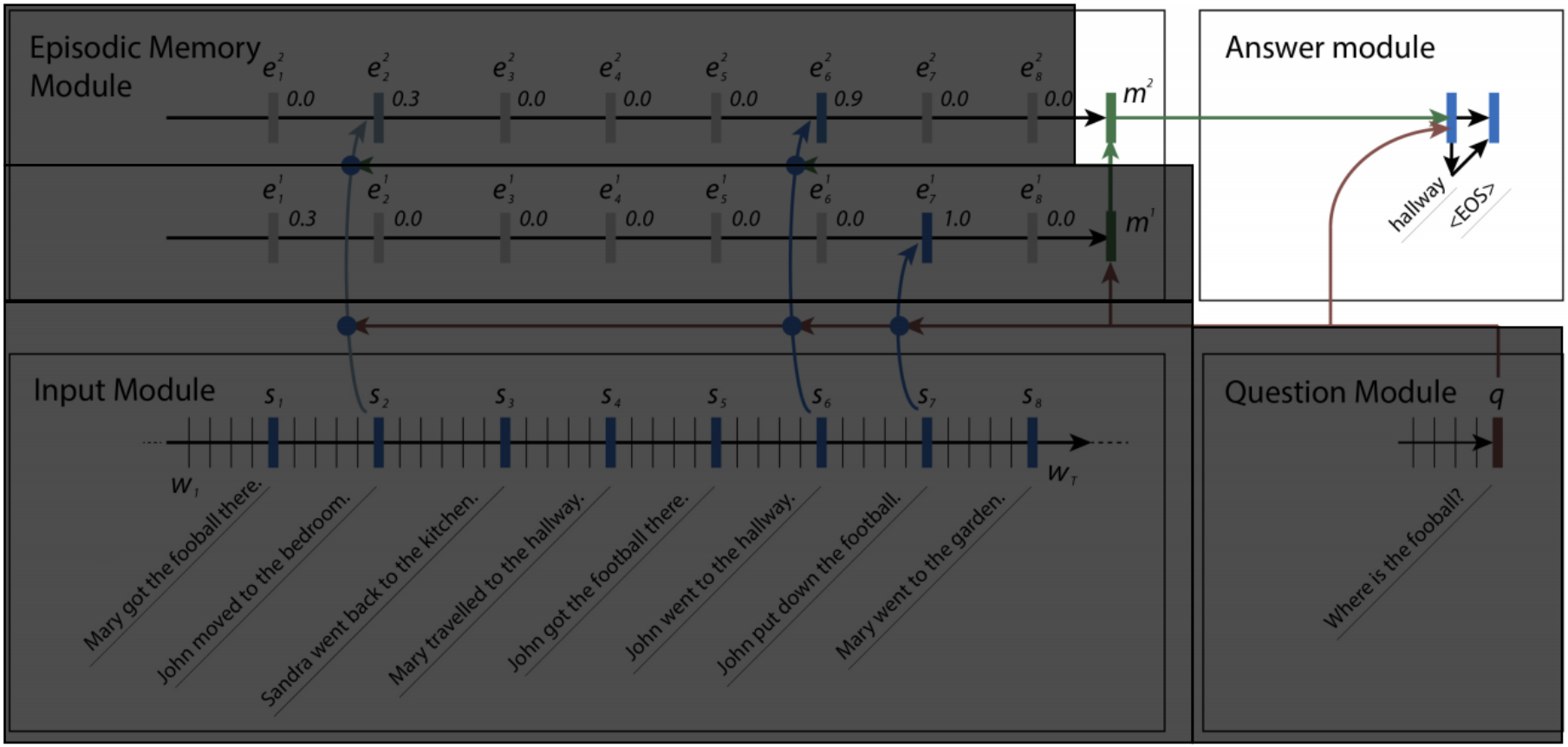
$$e^i = h_{T_C}^i$$

DMN



$$m^i = \mathrm{GRU}(e^i, m^{i-1})$$

DMN

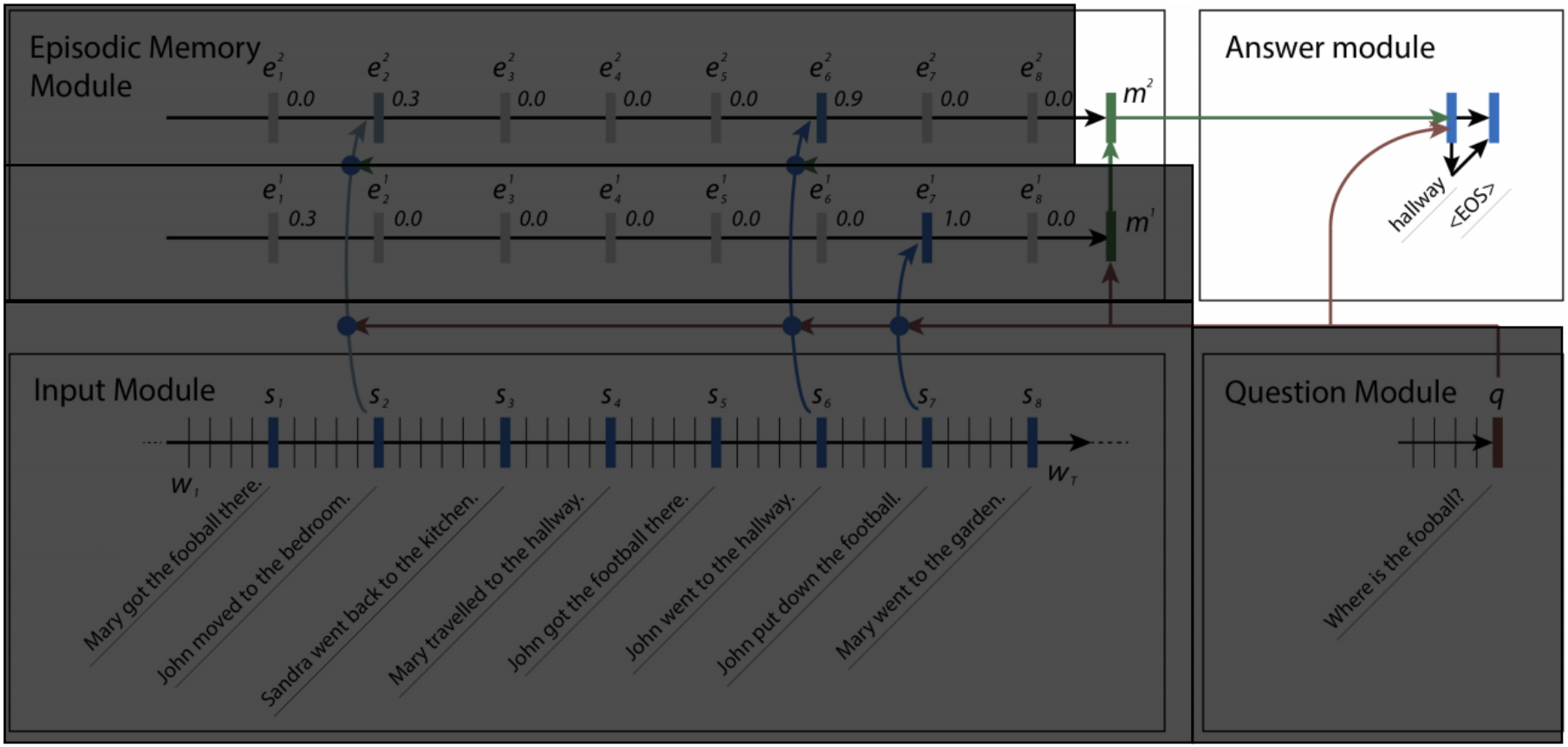


$$y_t = \mathrm{Softmax}(W^{(a)}\alpha_t) \qquad \alpha_0 = m^{T_m}$$

$$\alpha_t = \mathrm{GRU}([y_{t-1}, q], \alpha_{t-1})$$

Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, Kumar et. al. ICML 2016

DMN

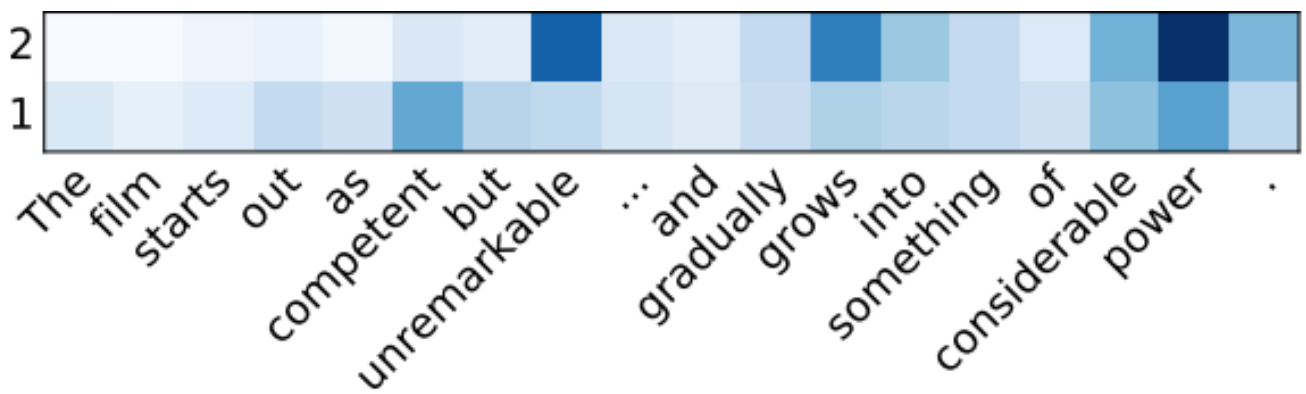


How many GRUs were used with 2 hops?

DMN – Qualitative Results

Question: Where was Mary before the Bedroom?
Answer: Cinema.

Facts	Episode 1	Episode 2	Episode 3
Yesterday Julie traveled to the school.			
Yesterday Marie went to the cinema.		■	
This morning Julie traveled to the kitchen.			
Bill went back to the cinema yesterday.			
Mary went to the bedroom this morning.	■		
Julie went back to the bedroom this afternoon.			
[done reading]			■



Algorithm Learning

Neural Turing Machine

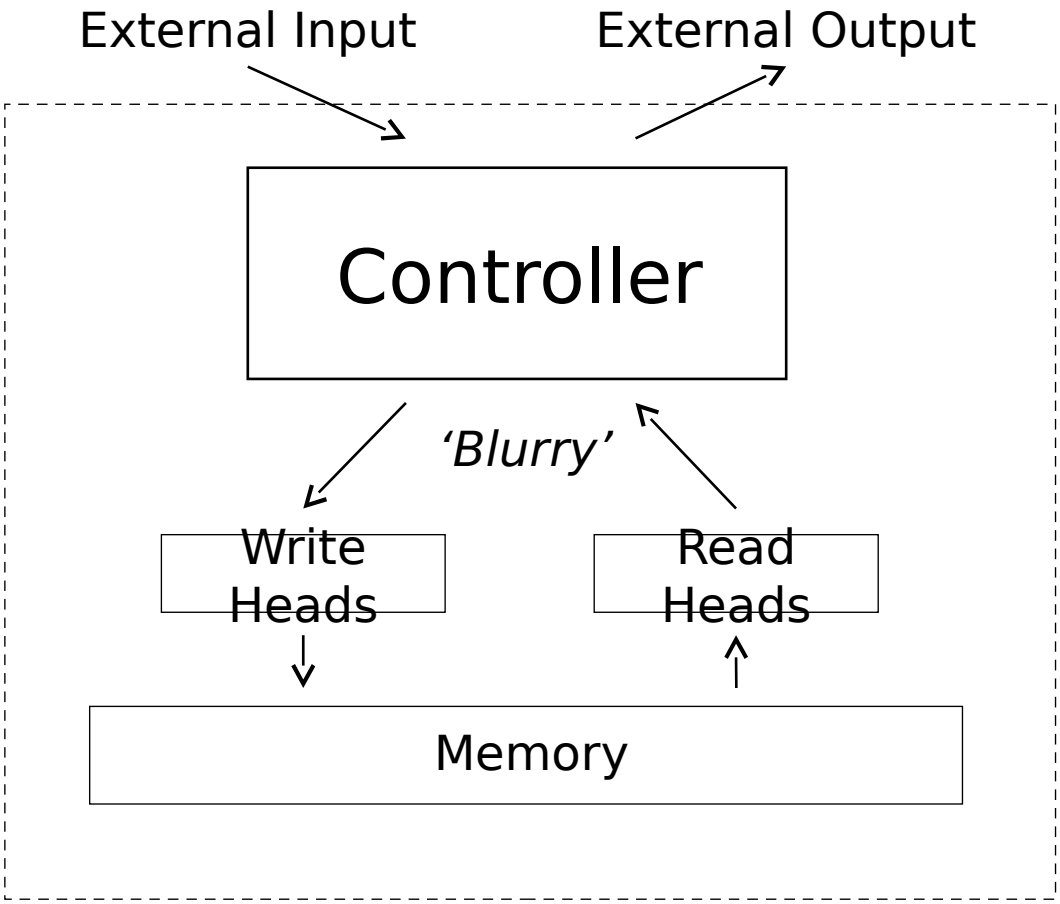
Copy Task: Implement the Algorithm

Given a list of numbers at input, reproduce the list at output

Neural Turing Machine Learns:

1. What to write to memory
2. When to write to memory
3. When to stop writing
4. Which memory cell to read from
5. How to convert result of read into final output

Neural Turing Machines



Neural Turing Machines

‘Blurry’ Memory Addressing (at time instant ‘t’)

$$\sum_{i=0}^{N-1} w_t(i) = 1$$

Soft Attention (Lectures 2, 3, 20, 24)

w_t

M_t

N

M

$w_t(0)$ = 0.1	$w_t(1)$ = 0.2	$w_t(2)$ = 0.5	$w_t(3)$ = 0.1	$w_t(4)$ = 0.1

Neural Turing Machines

More formally,

Blurry Read Operation

Given: M_t (memory matrix) of size NxM

w_t (weight vector) of length N

t (time index)

$$r_t = \sum_{i=0}^{N-1} w_t(i)\mathbf{M}_t(i)$$

Neural Turing Machines: Blurry Writes

Blurry Write Operation

Decomposed into <u>blurry erase</u> + <u>blurry add</u>

Given: M_t (memory matrix) of size NxM

w_t (weight vector) of length N

t (time index)

e_t (erase vector) of length M

a_t (add vector) of length M

$$\mathbf{M}_t(i) = \underbrace{\mathbf{M}_{t-1}(i)(1 - w_t(i)\mathbf{e}_t)}_{\text{Erase Component}} + \underbrace{w_t(i)\mathbf{a}_t}_{\text{Add Component}}$$

Neural Turing Machines: Erase

$$\mathbf{M}_t(i) = \boxed{\mathbf{M}_{t-1}(i)(1 - w_t(i)\mathbf{e}_t)}$$

1xN

Mx1

	$w_1(0)$ = 0.1	$w_1(1)$ = 0.2	$w_1(2)$ = 0.5	$w_1(3)$ = 0.1	$w_1(4)$ = 0.1	e_1
	5	7	9	2	12	1.0
	11	6	3	1	2	0.7
M_0 ⇨	3	7	3	10	6	0.2
	4	2	5	9	9	0.5
	3	5	12	8	4	0.0

Neural Turing Machines: Erase

$$\mathbf{M}_t(i) = \boxed{\mathbf{M}_{t-1}(i)(1 - w_t(i)\mathbf{e}_t)}$$

$w_1(0)$ = 0.1	$w_1(1)$ = 0.2	$w_1(2)$ = 0.5	$w_1(3)$ = 0.1	$w_1(4)$ = 0.1
4.5	5.6	4.5	1.8	10.8
10.23	5.16	1.95	0.93	1.86
2.94	6.72	2.7	9.8	5.88
3.8	1.8	3.75	8.55	8.55
3	5	12	8	4

Neural Turing Machines: Addition

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i)(1 - w_t(i)\mathbf{e}_t) + \boxed{w_t(i)\mathbf{a}_t}$$

$w_1(0)$ = 0.1	$w_1(1)$ = 0.2	$w_1(2)$ = 0.5	$w_1(3)$ = 0.1	$w_1(4)$ = 0.1
4.5	5.6	4.5	1.8	10.8
10.23	5.16	1.95	0.93	1.86
2.94	6.72	2.7	9.8	5.88
3.8	1.8	3.75	8.55	8.55
3	5	12	8	4

a_1
3
4
-2
0
2

Neural Turing Machines: Blurry Writes

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i)(1 - w_t(i)\mathbf{e}_t) + w_t(i)\mathbf{a}_t$$

M_1 ⇨

4.8	6.2	6	2.1	11.1
10.63	5.96	3.95	1.33	2.26
2.74	6.32	1.7	9.6	5.68
3.8	1.8	3.75	8.55	8.55
3.2	5.4	13	8.2	4.2

Neural Turing Machines: Attention Model

Generating w_t

Content Based

Example: QA Task

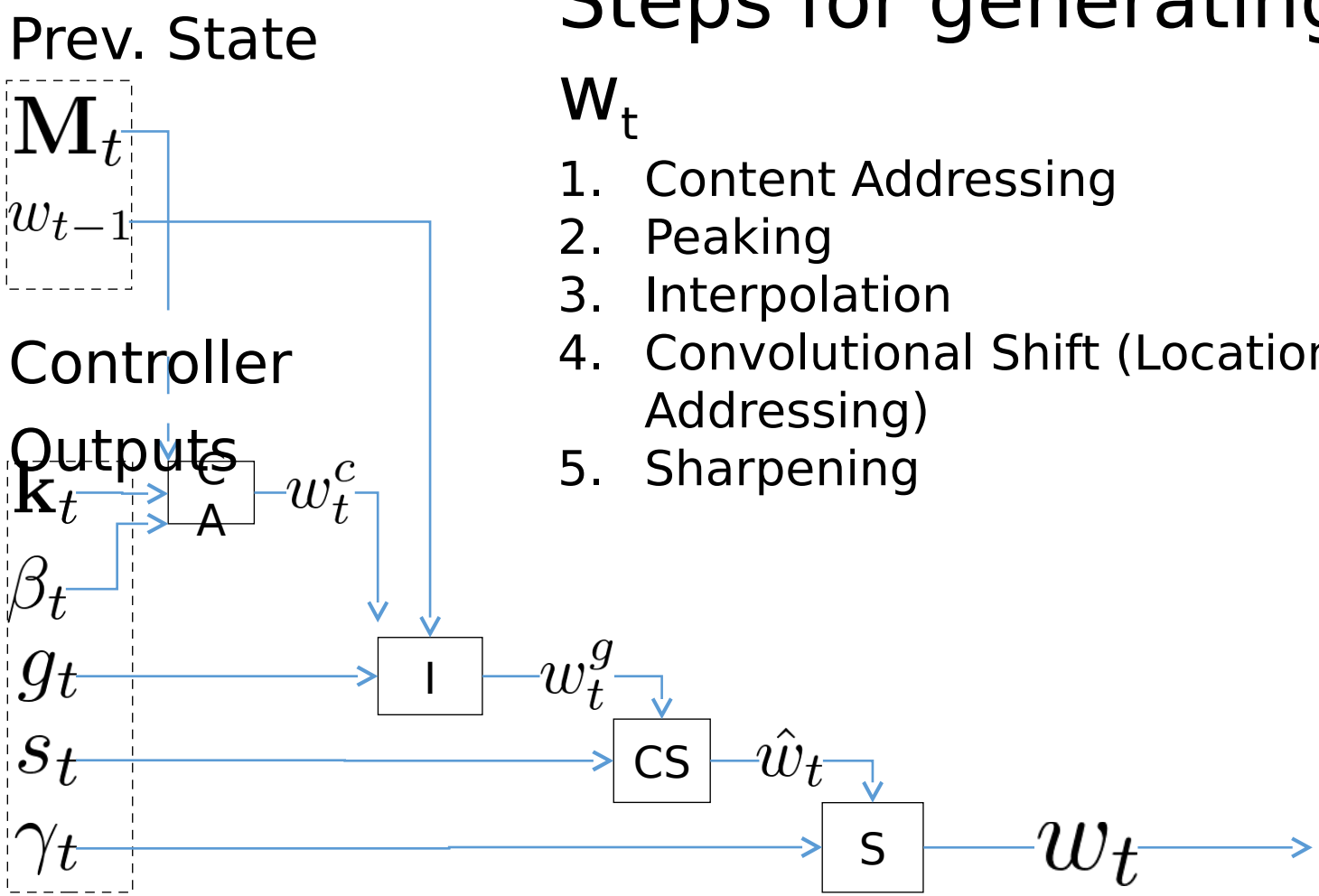
- Score sentences by similarity with Question
- Weights as softmax of similarity scores

Location Based

Example: Copy Task

- Move to address (i+1) after writing to index (i)
- Weights ≈ Transition probabilities

Neural Turing Machine: Attention Model



Steps for generating w_t

1. Content Addressing
2. Peaking
3. Interpolation
4. Convolutional Shift (Location Addressing)
5. Sharpening

Neural Turing Machine: Attention Model

Prev. State

$\mathbf{M}_t$

Controller

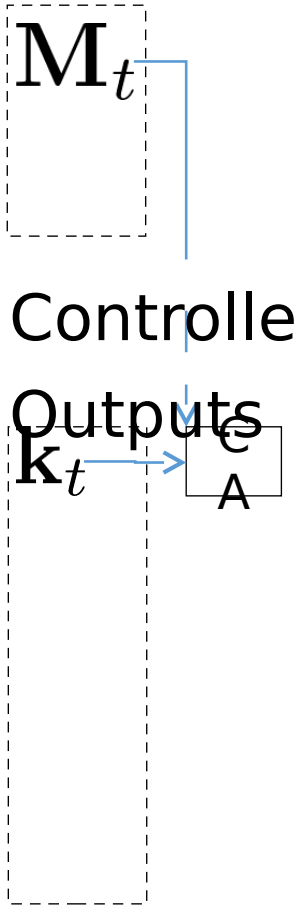
Outputs

$\mathbf{k}_t$

$\mathbf{k}_t$:Vector (length M) produced by Controller

Neural Turing Machine: Attention Model

Prev. State

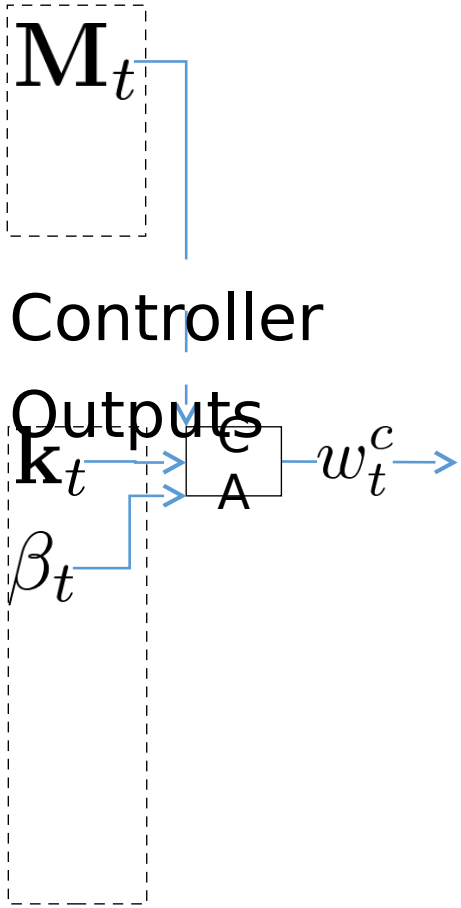


<u>Step 1: Content Addressing (CA)</u>

$$w_t^c(i) = \frac{exp < \mathbf{M}_t(i), \mathbf{k}_t >}{\sum_i exp < \mathbf{M}_t(i), \mathbf{k}_t >}$$

Neural Turing Machine: Attention Model

Prev. State

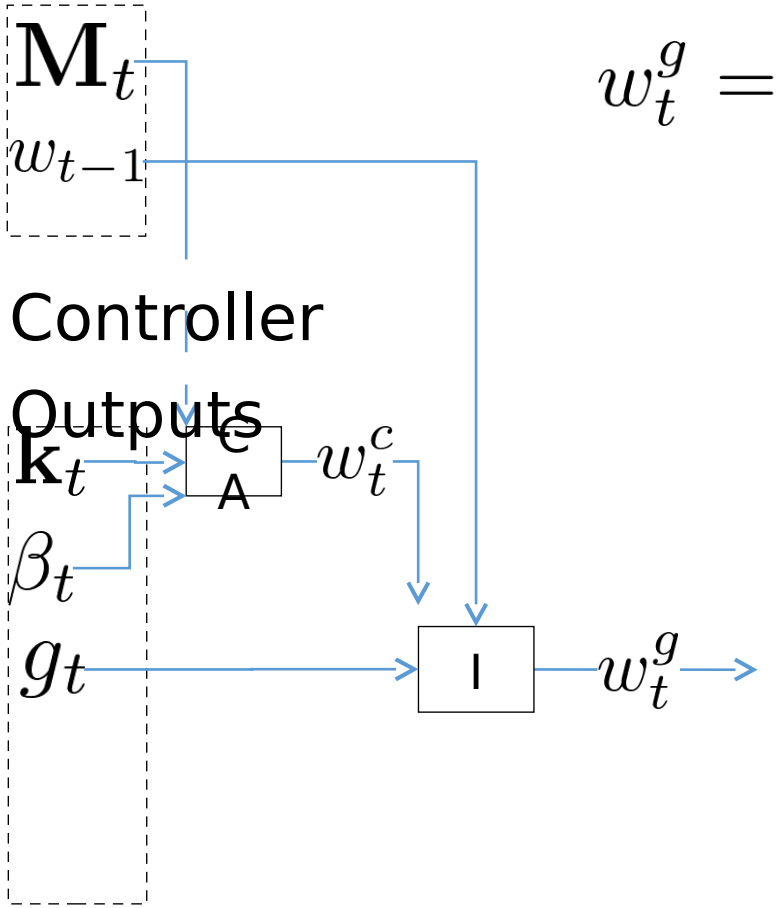


<u>Step 2: Peaking</u>

$$w_t^c(i) = \frac{exp(\beta_t(< \mathbf{M}_t(i), \mathbf{k}_t >))}{\sum_i exp(\beta_t(< \mathbf{M}_t(i), \mathbf{k}_t >))}$$

Neural Turing Machine: Attention Model

Prev. State

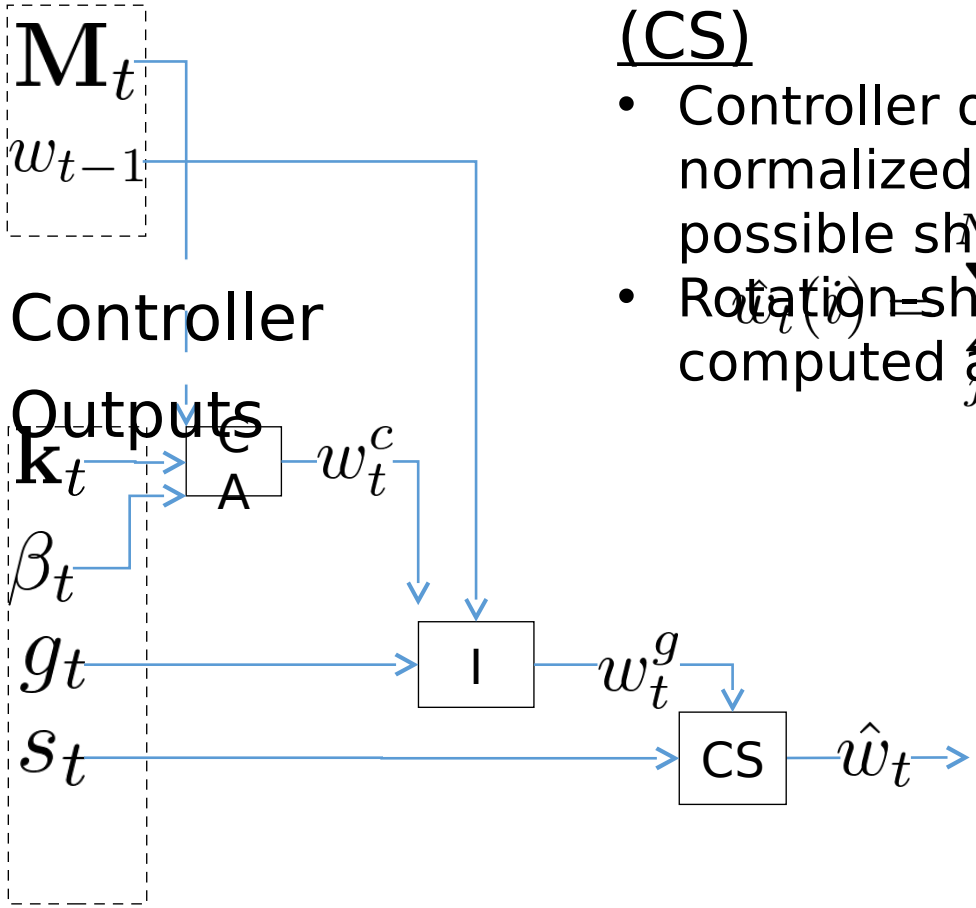


<u>Step 3: Interpolation (I)</u>

$$w_t^g = g_t w_t^c + (1 - g_t) w_{t-1}$$

Neural Turing Machine: Attention Model

Prev. State



<u>Step 4: Convolutional Shift (CS)</u>

- Controller outputs s_t, a normalized distribution over all N possible shifts
- Rotation-shifted weights computed as:

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

Neural Turing Machine: Attention Model

Prev. State

$\mathbf{M}_t$

w_{t-1}

Controller

Outputs

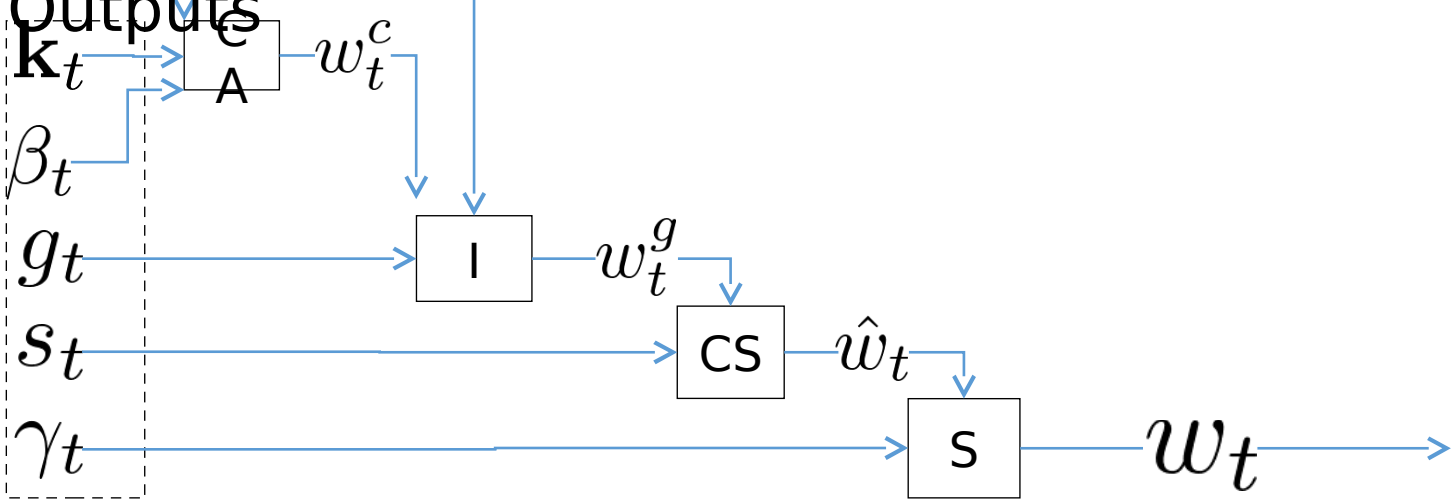
$\mathbf{k}_t$

β_t

g_t

s_t

γ_t



Step 5: Sharpening (S)

- Uses γ_t to sharpen as:

$$w_t(i) = \frac{\hat{w}(i)^{\gamma_t}}{\sum_i \hat{w}(i)^{\gamma_t}}$$

Neural Turing Machine: Controller Design

- Feed-forward: faster, more transparency & interpretability about function learnt

- LSTM: more expressive power, doesn't limit the number of computations per time step

Both are end-to-end differentiable!

1. Reading/Writing -> Convex Sums
2. w_t generation -> Smooth
3. Controller Networks

Neural Turing Machine: Network Overview

Unrolled Feed-forward Controller

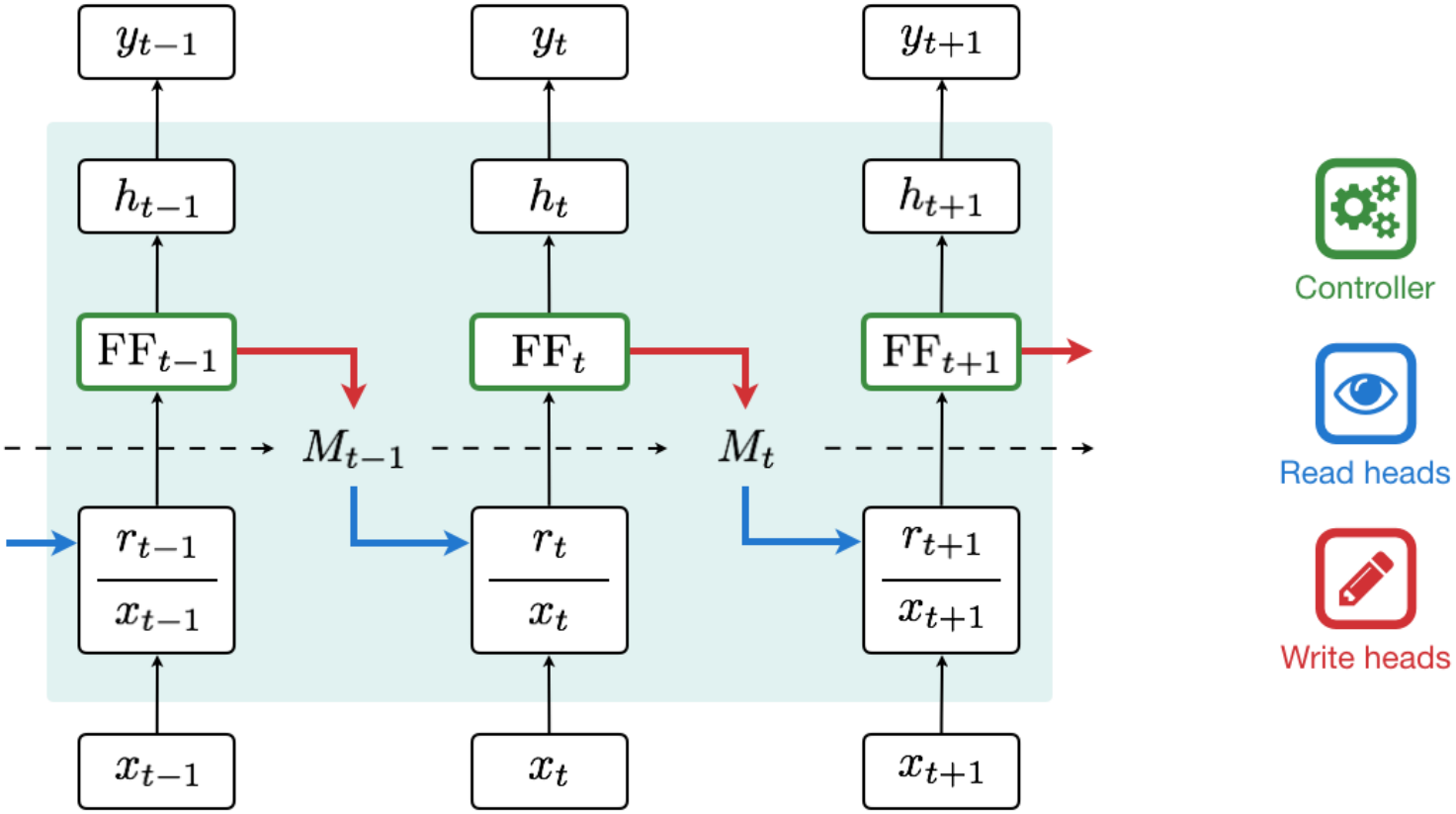


Figure from
[Snips AI's Medium Post]

Neural Turing Machines vs. MemNNs

MemNNs

- Memory is static, with focus on retrieving (reading) information from memory

NTMs

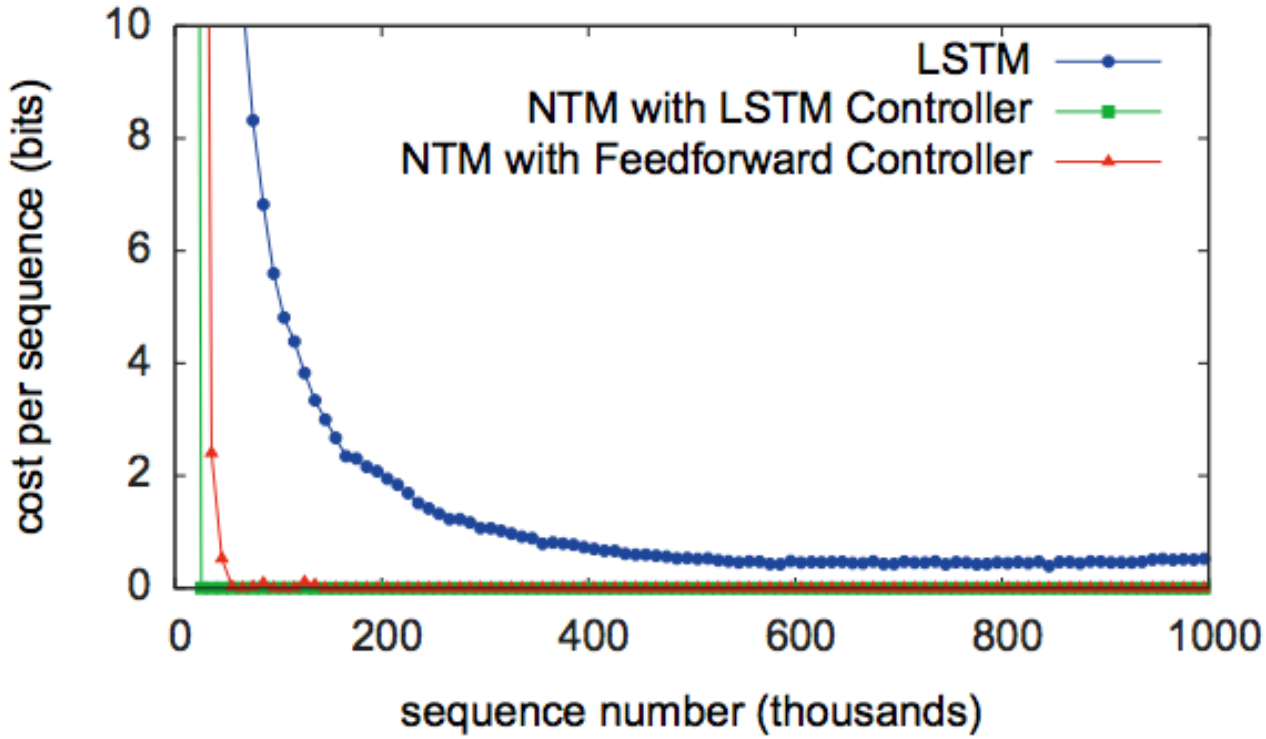
- Memory is continuously written to and read from, with network learning when to perform memory read and write

Neural Turing Machines: Experiments

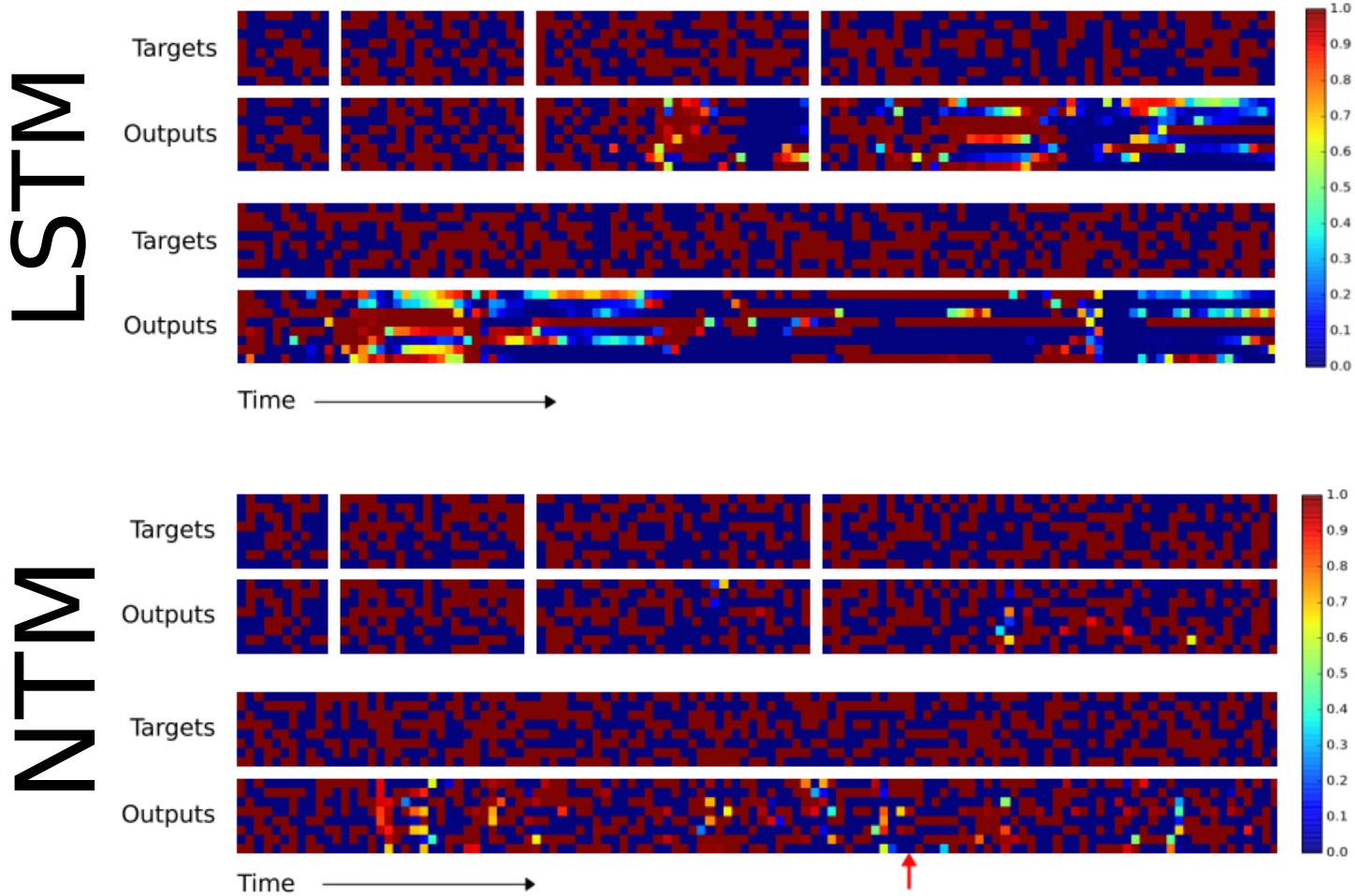
Task	**Network Size**		**Number of Parameters**	
	NTM w/ LSTM*	**LSTM**	**NTM w/ LSTM**	**LSTM**
Copy	3 x 100	3 x 256	67K	1.3M
Repeat Copy	3 x 100	3 x 512	66K	5.3M
Associative	3 x 100	3 x 256	70K	1.3M
N-grams	3 x 100	3 x 128	61K	330K
Priority Sort	2 x 100	3 x 128	269K	385K

Neural Turing Machines: ‘Copy’ Learning Curve

Trained on 8-bit sequences, 1<= sequence length <= 20



Neural Turing Machines: ‘Copy’ Performance



Neural Turing Machines triggered an outbreak of Memory Architectures!

Dynamic Neural Turing Machines

Experimented with addressing schemes

- Dynamic Addresses: Addresses of memory locations learnt in training – allows non-linear location-based addressing
- Least recently used weighting: Prefer least recently used memory locations + interpolate with content-based addressing
- Discrete Addressing: Sample the memory location from the content-based distribution to obtain a one-hot address
- Multi-step Addressing: Allows multiple hops over memory

Results: bAbI QA task

	Location NTM	**Content NTM**		**Soft DNTM**	**Discrete DNTM**
1-step	31.4%	33.6%		29.5%	27.9%
3-step	32.8%	32.7%		24.2%	21.7%

Stack Augmented Recurrent Networks

Learn algorithms based on stack implementations (e.g. learning fixed sequence g

Sequence generator	Example
$\{a^n b^n \mid n > 0\}$	aab**ba**aab**bba**b**a**aaaab**bbbb**
$\{a^n b^n c^n \mid n > 0\}$	aaab**bbccca**b**ca**aaaab**bbbbccccc**
$\{a^n b^n c^n d^n \mid n > 0\}$	aab**bccdda**aab**bbcccddda**b**cd**
$\{a^n b^{2n} \mid n > 0\}$	aab**bbba**aab**bbbbba**b**b**
$\{a^n b^m c^{n+m} \mid n, m > 0\}$	aabc**cca**aabbc**ccccca**bc**c**
$n \in [1, k],\ X \rightarrow nXn,\ X \rightarrow =$	$(k = 2)$ 12=**21**2122=**221**11121=**12111**

Uses a stack data structure to store memory (as opposed to a memory matrix)

Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets, Joulin et. al., arXiv:1503.01007

Stack Augmented Recurrent Networks

- Blurry ‘push’ and ‘pop’ on stack. E.g.:

$$s_t[0] = a[\texttt{Push}](h_t) + a[\texttt{Pop}]s_{t-1}[1]$$

- Some results:

method	$a^n b^n$	$a^n b^n c^n$	$a^n b^n c^n d^n$	$a^n b^{2n}$	$a^n b^m c^{n+m}$
RNN	25%	23.3%	13.3%	23.3%	33.3%
LSTM	100%	100%	68.3%	75%	100%
List RNN 40+5	100%	33.3%	100%	100%	100%
Stack RNN 40+10	100%	100%	100%	100%	43.3%
Stack RNN 40+10 + rounding	100%	100%	100%	100%	100%

Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets, Joulin et. al., arXiv:1503.01007

Differentiable Neural Computers

Advanced addressing mechanisms:

- Content Based Addressing
- Temporal Addressing
  - Maintains notion of sequence in addressing
  - Temporal Link Matrix *L* (size NxN), L[*i,j*] = degree to which location *I* was written to after location *j.*
- Usage Based Addressing

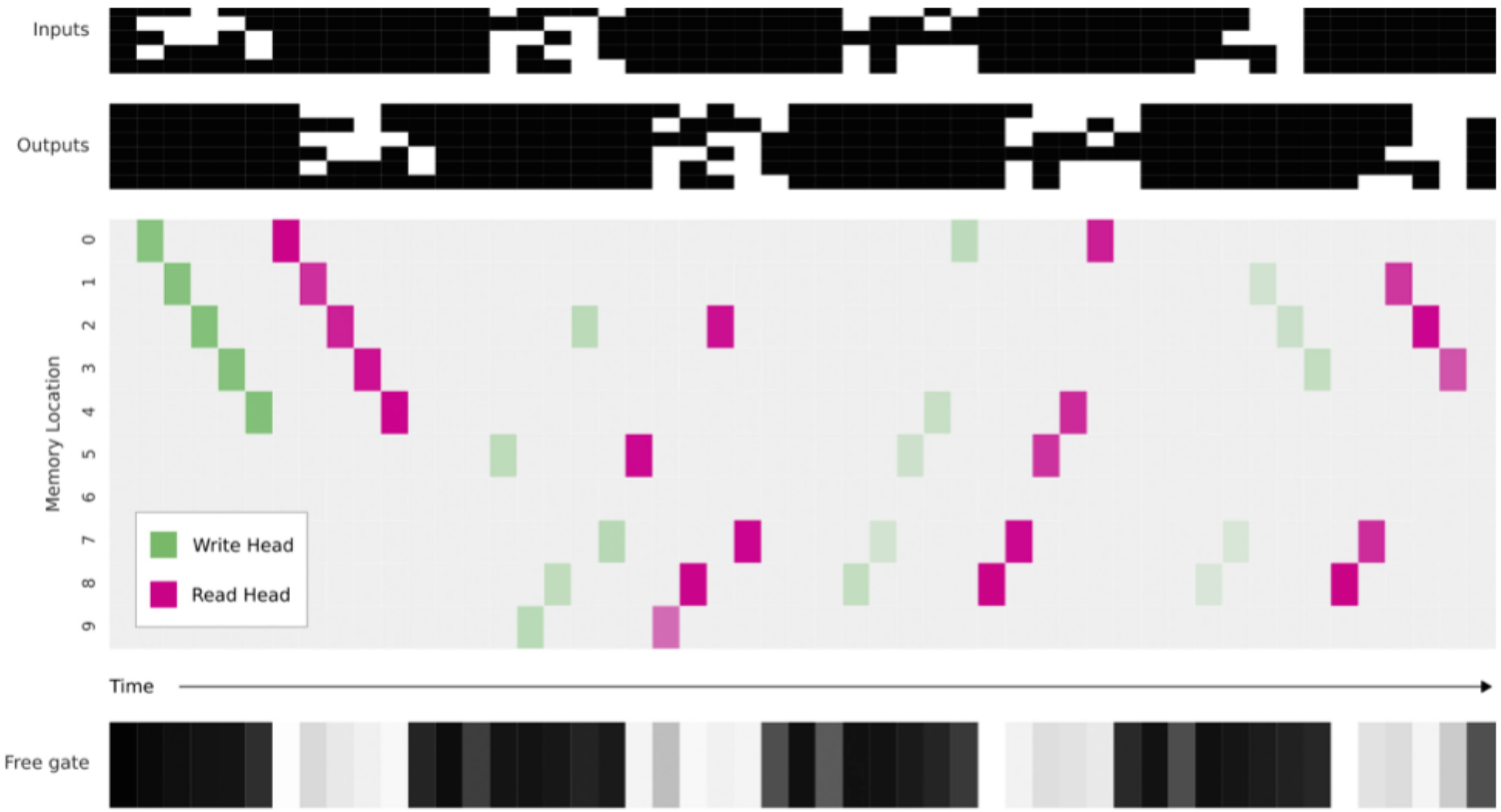
ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Usage Based Addressing

- Writing increases usage of cell, reading decreases usage of cell
- Least used location has highest usage-based weighting
- Interpolate b/w usage & content based weights for final write weights

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Example



Hybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Improvements over NTMs

NTM

- Large contiguous blocks of memory needed
- No way to free up memory cells after writing

DNC

- Memory locations non-contiguous, usage-based
- Regular de-allotment based on usage-tracking

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks

Graph Representation: (source, edge, destination) tuples

Types of tasks:

- Traversal: Perform walk on graph given source, list of edges
- Shortest Path: Given source, destination
- Inference: Given source, relation over edges; find destination

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks

Training over 3 phases:

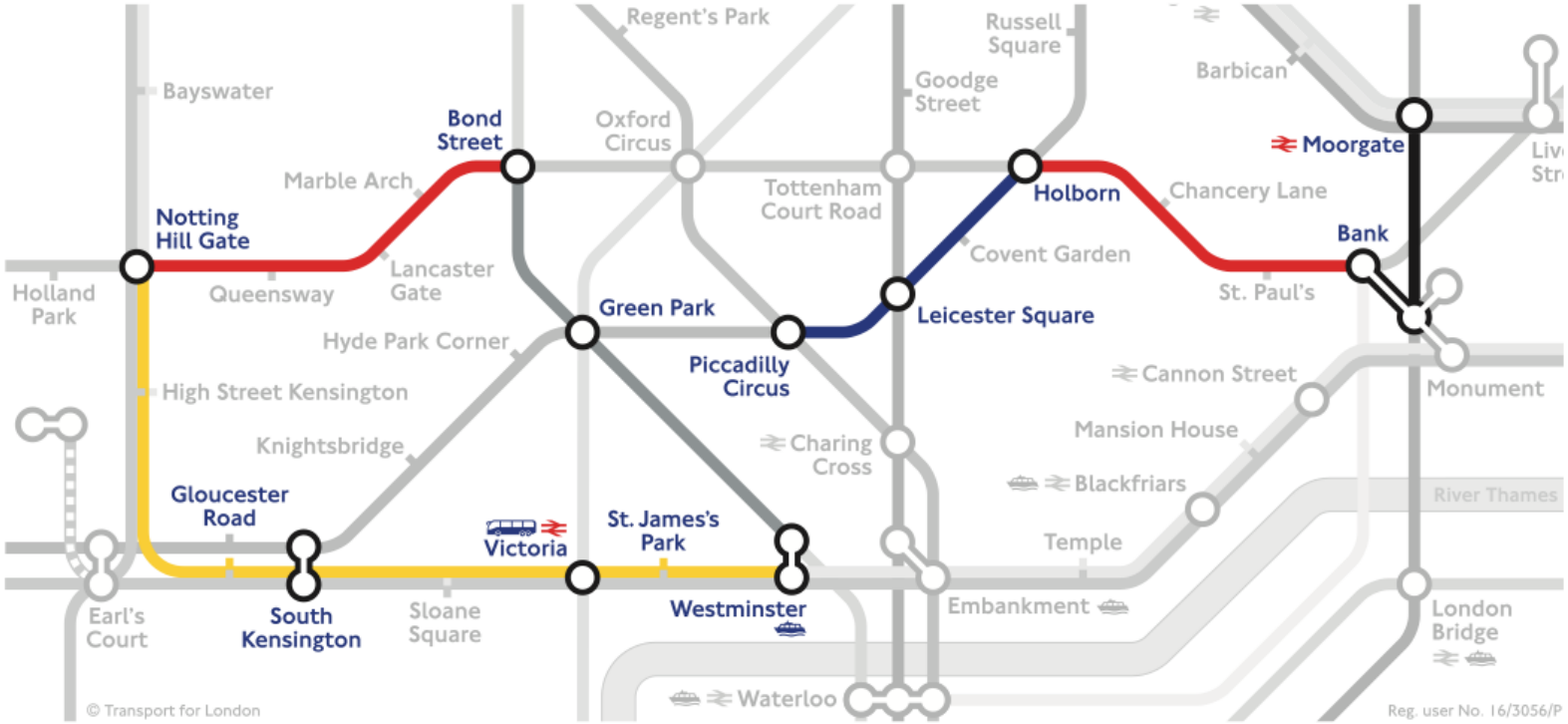
- Graph description phase: (source, edge, destination) tuples fed into the graph
- Query phase: Shortest path (source, ____, destination), Inference (source, hybrid relation, ___), Traversal (source, relation, relation …, ___)
- Answer phase: Target responses provided at output

Trained on random graphs of maximum size 1000

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks: London Underground



ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks: London Underground

Input Phase

(OxfordCircus, TottenhamCtRd, Central)
(TottenhamCtRd, OxfordCircus, Central)
(BakerSt, Marylebone, Circle)
(BakerSt, Marylebone, Bakerloo)
(BakerSt, OxfordCircus, Bakerloo)
⋮
(LeicesterSq, CharingCross, Northern)
(TottenhamCtRd, LeicesterSq, Northern)
(OxfordCircus, PiccadillyCircus, Bakerloo)
(OxfordCircus, NottingHillGate, Central)
(OxfordCircus, Euston, Victoria)

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks: London Underground

Traversal Task

(BondSt, _, Central),
(_, _, Circle), (_, _, Circle),
(_, _, Circle), (_, _, Circle),
(_, _, Jubilee), (_, _, Jubilee),

Query Phase

(BondSt, NottingHillGate, Central)
(NottingHillGate, GloucesterRd, Circle)
⋮
(Westminster, GreenPark, Jubilee)
(GreenPark, BondSt, Jubilee)

Answer Phase

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments

Graph Tasks: London Underground

Shortest Path Task

(Moorgate, PiccadillyCircus, _)

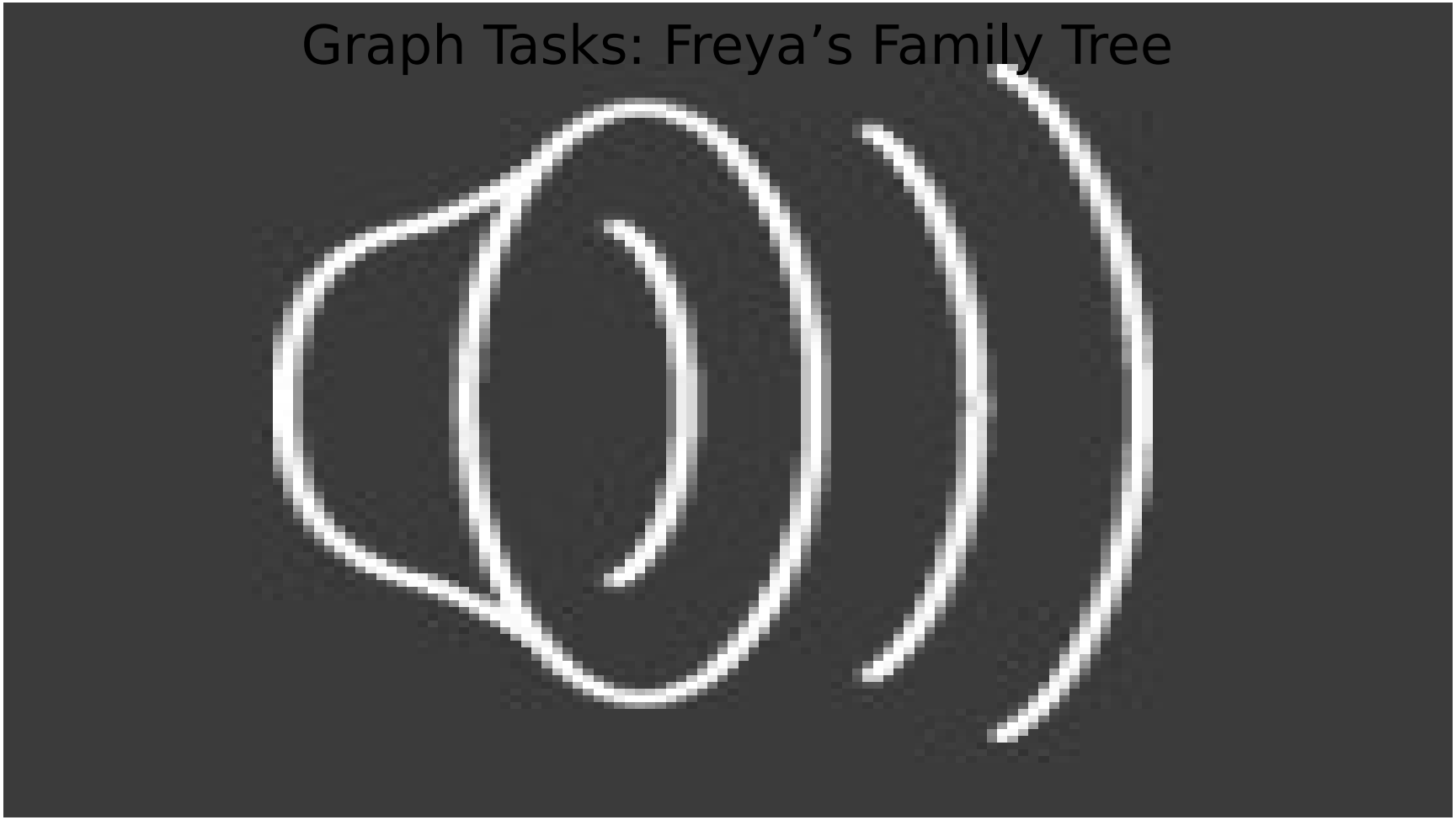
(Moorgate, Bank, Northern)
(Bank, Holborn, Central)
(Holborn, LeicesterSq, Piccadilly)
(LeicesterSq, PiccadillyCircus, Piccadilly)

Query Phase

Answer Phase

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

DNC: Experiments



ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

Conclusion

- Machine Learning models require memory and multi-hop reasoning to perform AI tasks better
- Memory Networks for Text are an interesting direction but very simple
- Generic architectures with memory, such as Neural Turing Machine, limited applications shown
- Future directions should be focusing on applying generic neural models with memory to more AI Tasks.

ybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

Reading List

- Karol Kurach, Marcin Andrychowicz & Ilya Sutskever **Neural Random-Access Machines**, ICLR, 2016
- Emilio Parisotto & Ruslan Salakhutdinov **Neural Map: Structured Memory for Deep Reinforcement Learning**, ArXiv, 2017
- Pritzel et. al. **Neural Episodic Control**, ArXiv, 2017
- Oriol Vinyals,Meire Fortunato, Navdeep Jaitly **Pointer Networks**, ArXiv, 2017
- Jack W Rae et al., **Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes, ArXiv 2016**
- Antoine Bordes, Y-Lan Boureau, Jason Weston, **Learning End-to-End Goal-Oriented Dialog, ICLR 2017**
- Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, Honglak Lee, **Control of Memory, Active Perception, and Action in Minecraft, ICML 2016**
- Wojciech Zaremba, Ilya Sutskever, **Reinforcement Learning Neural Turing Machines, ArXiv 2016**