Neural Architectures with Memory

Nitish Gupta, Shreya Rajpal

25th April, 2017

Story Comprehension

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk. Joe travelled to his office. Joe left the milk. Joe went to the bathroom.

Questions from Joe's angry mother:

Q1 : Where is Joe?

Q2 : Where is the milk now?

Q3 : Where was Joe before the office?

Dialogue System

Machine

Hello! What can I do for you today?

I'd like to reserve a table for 6.

Sure! When would you like that reservation?

At 7 PM, please.

Okay. What cuisine would you like?

Actually make that 7:30 PM

Updated! What cuisine?

Is there anything better than a medium rare steak?

Nothing at all! Blackdog has a 4.7 on Yelp.

Sounds perfect! Also, add one more person.

Reservation done for 7, 7:30pm at Blackdog. Enjoy!

Human

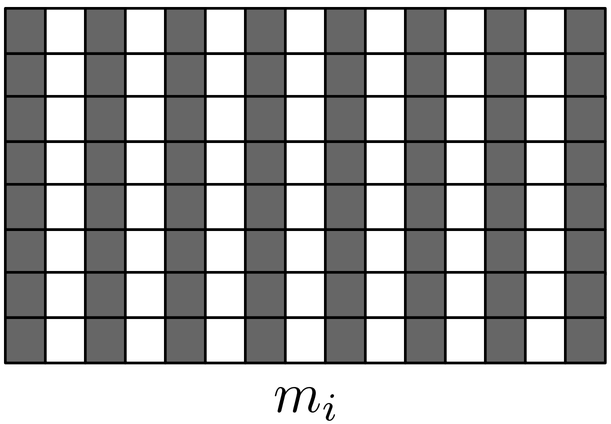
ML models need memory!

Deeper AI tasks require explicit memory and multi-hop reasoning over it

- RNNs have short memory
- Cannot increase memory without increasing number of parameters
- Need for compartmentalized memory
- Read/Write should be asynchronous

Memory Networks (MemNN)

- Class of Models with memory $\boldsymbol{m}$ - Array of objects $\boldsymbol{m_i}$



Each memory here is a dense vector

Four Components :

I - Input Feature Map : Input manipulation

G - Generalization : Memory Manipulation

O - Output Feature Map : Output representation generator

R - Response : Response Generator

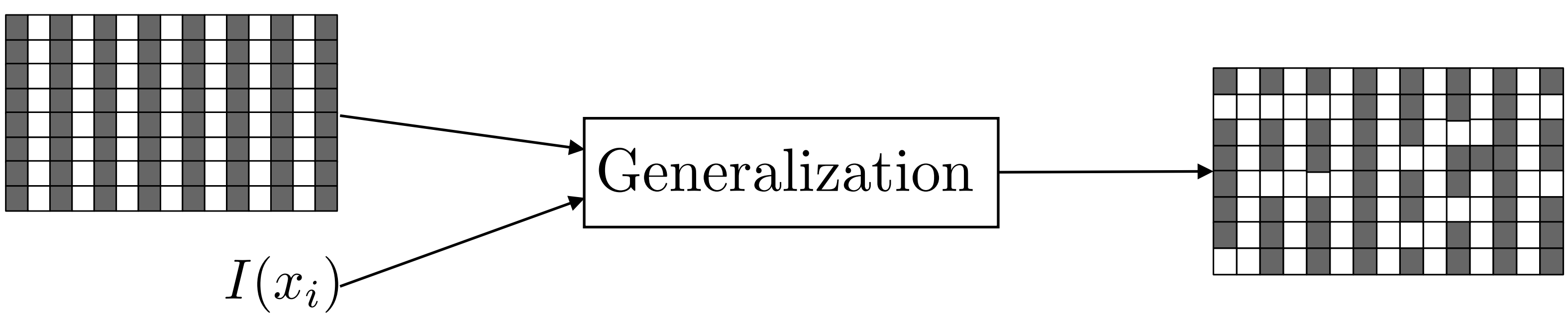
Memory Networks, Weston et. al., ICLR 2015

MemNN

1. **Input Feature Map**
   - Imagine input as a sequence of sentences x_i

$$x_i \longrightarrow \boxed{\text{Input Feature Map}} \longrightarrow I(x_i)$$

2. **Update Memories**

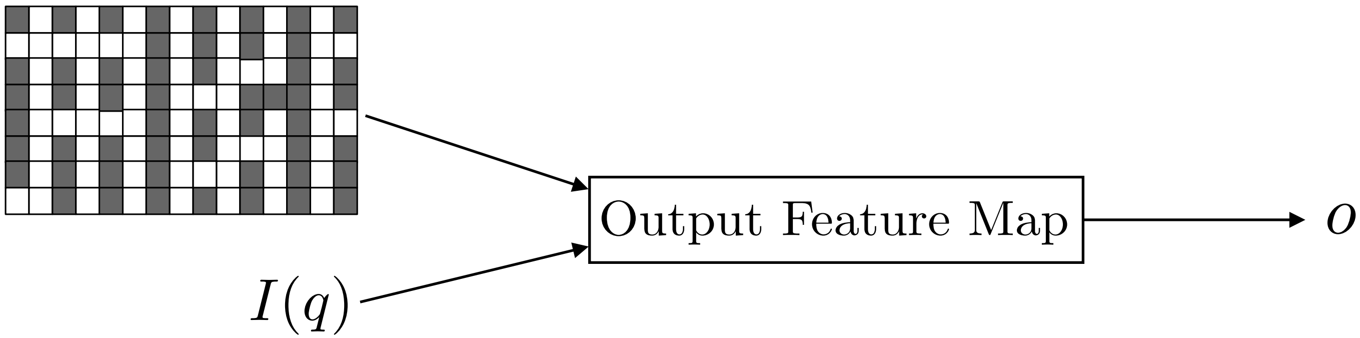


Memory Networks, Weston et. al., ICLR 2015

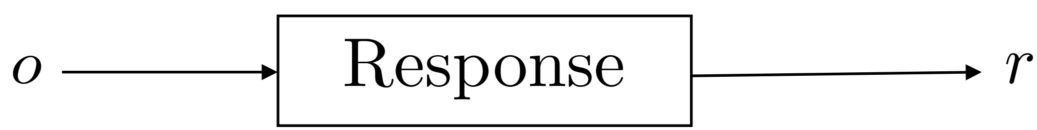
MemNN

3. Output Representation

- Say if q is a question, compute output representation



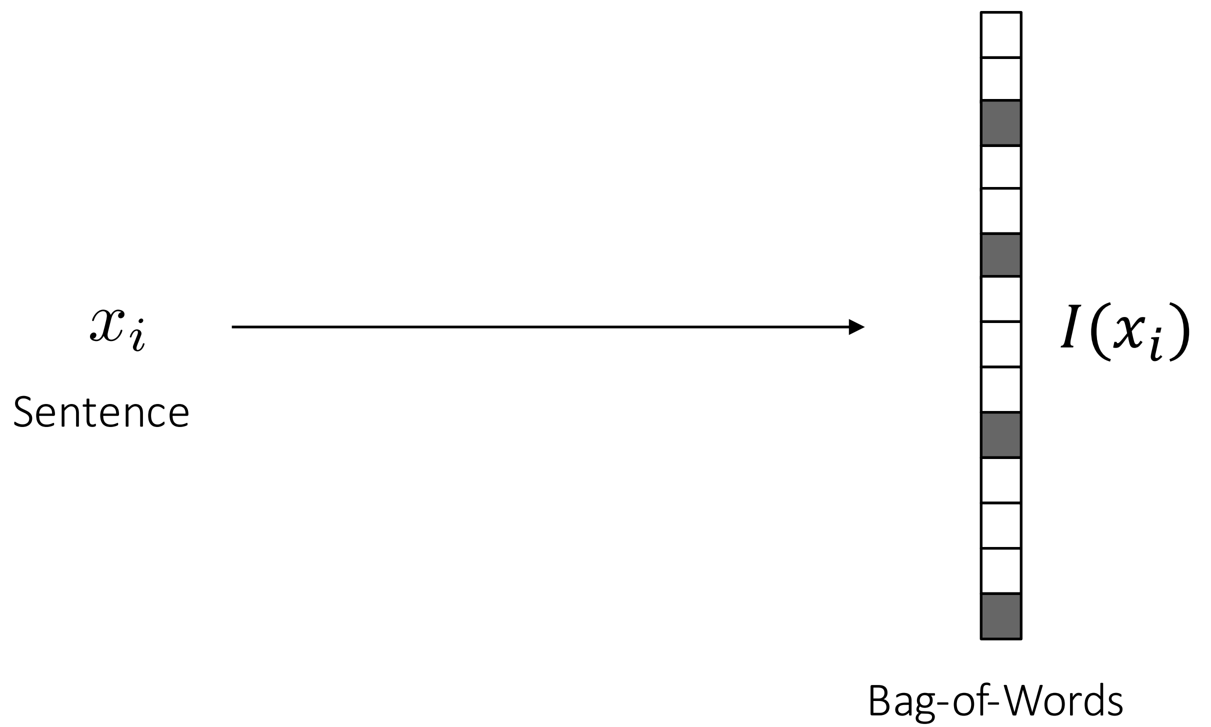
4. Generate Answer Response



Memory Networks, Weston et. al., ICLR 2015

Simple MemNN for Text

1. **Input Feature Map** - Bag-of-Words representation

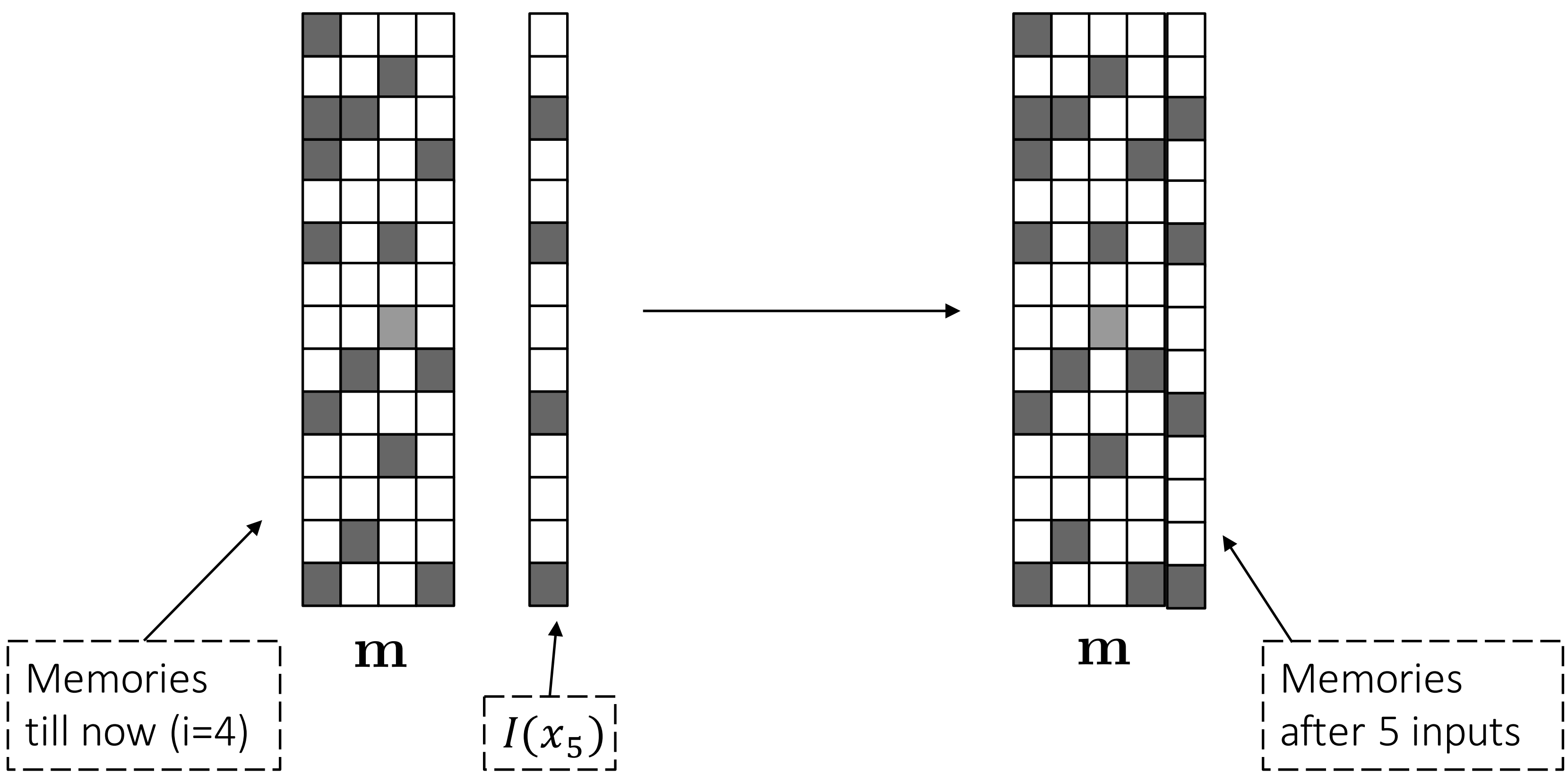


Memory Networks, Weston et. al., ICLR 2015

Simple MemNN for Text

2. **Generalization** : Store input in new memory

$$m_i = I(x_i)$$



Memory Networks, Weston et. al., ICLR 2015

Simple MemNN for Text

3. **Output:** Using $k = 2$ memory hops with query x

1st Max scoring memory index

$$o_1 = O_1(x, \mathbf{m}) = \underset{i=1,\dots,N}{argmax}\ s_O(I(x), m_i)$$

Score all memories against input

2nd Max scoring memory index

$$o_2 = O_2(x, \mathbf{m}) = \underset{i=1,\dots,N}{argmax}\ s_O([I(x), m_{o_1}], m_i)$$

Score all memories against input & o_1

4. **Response** - Single Word Answer

Max scoring word

$$r = argmax_{w \in W}\ s_R([I(x), m_{o_1}, m_{o_2}], w)$$

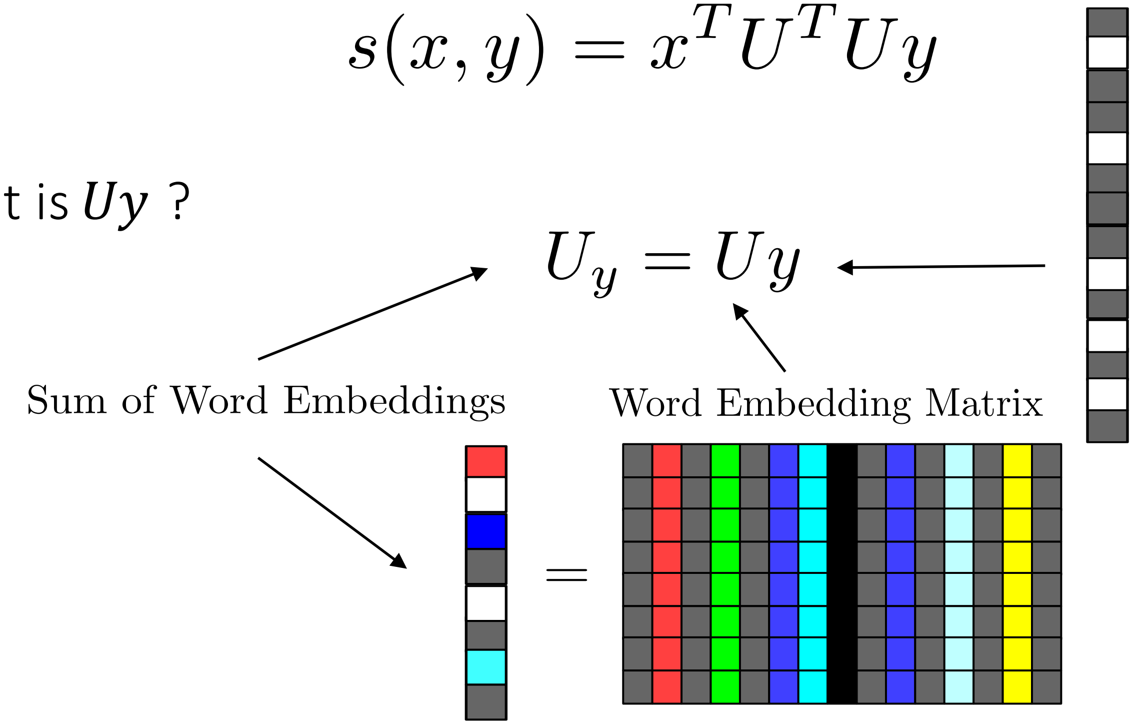
Score all words against query and 2 supporting memories

Scoring Function

- Scoring Function is an embedding model

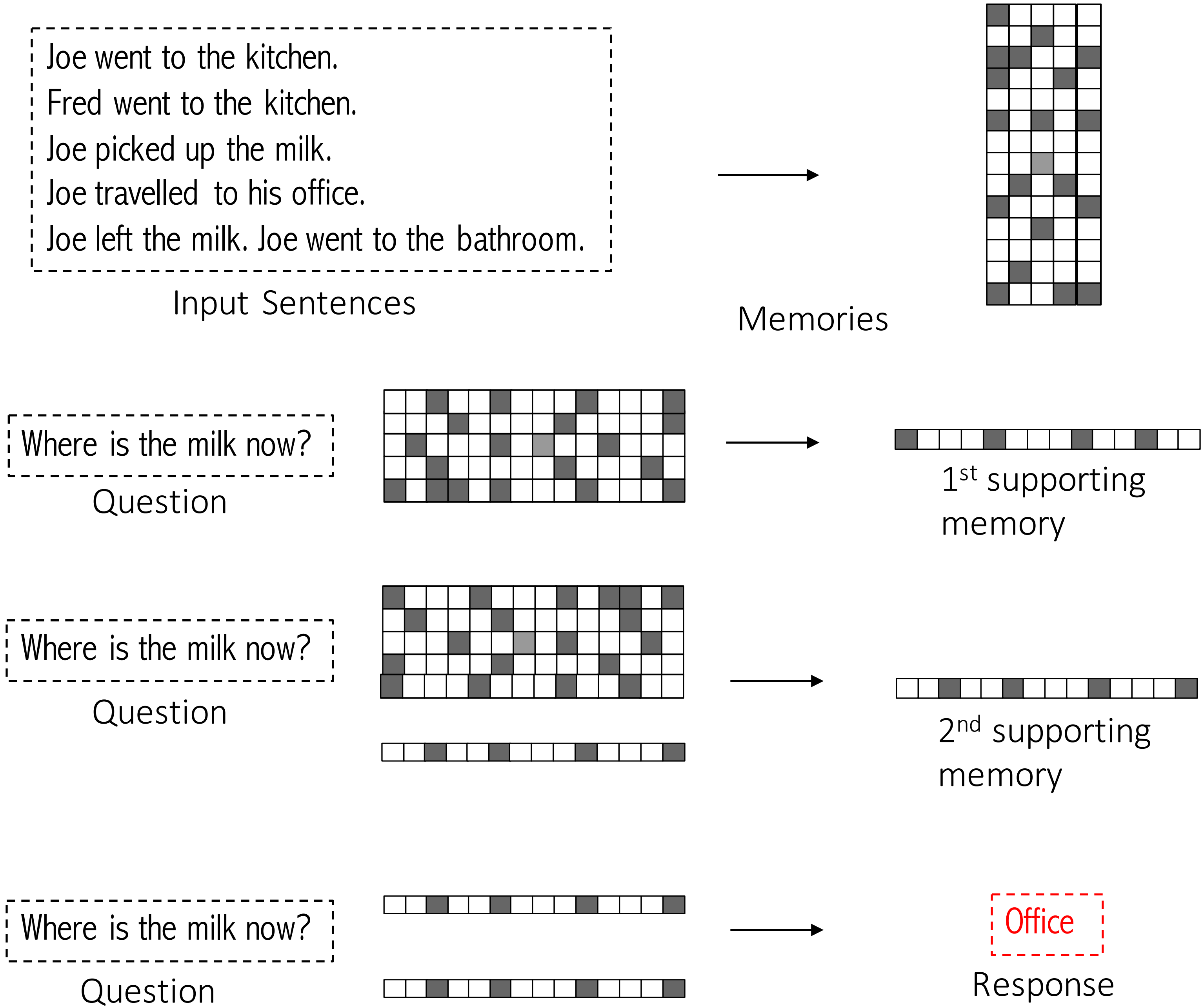
$$s(x, y) = x^T U^T U y$$

- What is Uy ?



Scoring Function is just dot-product between sum of word embeddings!!!

Memory Networks, Weston et. al., ICLR 2015



Memory Networks, Weston et. al., ICLR 2015

Training Objective

Score for true 1st memory

Score for a negative memory

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} max(0, \gamma - \boxed{s_O(x, \mathbf{m}_{o_1})} + \boxed{s_O(x, \bar{f})}) \; +$$

Memory Networks, Weston et. al., ICLR 2015

Training Objective

Score for true 2nd memory

Score for a negative memory

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} max(0, \gamma - s_O(x, \mathbf{m}_{o_1}) + s_O(x, \bar{f})) \ +$$

$$\sum_{\bar{f}' \neq \mathbf{m}_{o_2}} max(0, \gamma - \boxed{s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_{o_2})} + \boxed{s_O([x, \mathbf{m}_{o_1}], \bar{f}')}) \ +$$

Memory Networks, Weston et. al., ICLR 2015

Training Objective

Score for true response

Score for a negative response

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} max(0, \gamma - s_O(x, \mathbf{m}_{o_1}) + s_O(x, \bar{f})) \; +$$

$$\sum_{\bar{f}' \neq \mathbf{m}_{o_2}} max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_{o_2}) + s_O([x, \mathbf{m}_{o_1}], \bar{f}')) \; +$$

$$\sum_{\bar{r} \neq r} max(0, \gamma - \boxed{s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r)} + \boxed{s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r})})$$

Memory Networks, Weston et. al., ICLR 2015

Experiment

- Large – Scale QA
  - 14M Statements – *(subject, relation, object)*
  - Memory Hops; $k = 1$
  - Only re-ranked candidates from other system

Stored as memories

Output is highest scoring memory

Method	F1
Fader et. al. 2013	0.54
Bordes et. al. 2014b	0.73
Memory Networks (This work)	0.72

Why does Memory Network perform exactly as previous model?

Memory Networks, Weston et. al., ICLR 2015

Experiment

- Large – Scale QA
  - 14M Statements – *(subject, relation, object)*
  - Memory Hops; $k = 1$
  - Only re-ranked candidates from other system

USELESS EXPERIMENT

Method	F1
Fader et. al. 2013	0.54
Bordes et. al. 2014b	0.73
Memory Networks (This work)	0.72

Why does Memory Networks not perform as well?

Useful Experiment

- Simulated World QA
  - 4 characters, 3 objects, 5 rooms
  - 7k statements, 3k questions for training and same for testing
  - Difficulty 1 (5) – Entity in question is mentioned in last 1 (5) sentences
  - For $k = 2$, annotation has intermediate best *memories* as well

Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living_room.
Where is Dan? A: living room I believe
Where is Joe? A: the bathroom
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now ? A: the milk is in the kitchen
Where is Dan now? A: I think he is in the bedroom
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living_room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? A: I think Joe is in the office

Memory Networks, Weston et. al., ICLR 2015

Limitations

- Simple BOW representation
- Simulated Question Answering dataset is too trivial
- Strong supervision i.e. for intermediate memories is needed

Memory Networks, Weston et. al., ICLR 2015

End-to-End Memory Networks (MemN2N)

- What if the annotation is:
  - Input sentences $x_1, x_2, \ldots, x_n$
  - Query q
  - Answer a

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk. Joe travelled to his office. Joe left the milk. Joe went to the bathroom.

Where is the milk now?

Office

- Model performs by:
  - Generating memories from inputs
  - Transforming query into suitable representation
  - Process query and memories jointly using multiple hops to produce the answer
  - Backpropagate through the whole procedure

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

MemN2N

1. Convert input to memories $x_i \rightarrow m_i$

$$m_i = Ax_i$$

Sum of word-embeddings

BOW input

Word-Embedding Matrix

2. Transform query q into same representation space

$$u = Bq$$

3. Output Vectors $x_i \rightarrow c_i$

$$c_i = Cx_i$$

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

MemN2N

3. Scoring memories against query

$$p_i = \mathrm{Softmax}(u^T m_i)$$

Score for input/memory

Memories

Query (transformed)

4. Generate output

$$o = \sum_i p_i c_i$$

Weighted average of all inputs (transformed)

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

MemN2N

5. Generating Response

$$\hat{a} = \text{Softmax}(W(u + o))$$

Distribution over response words

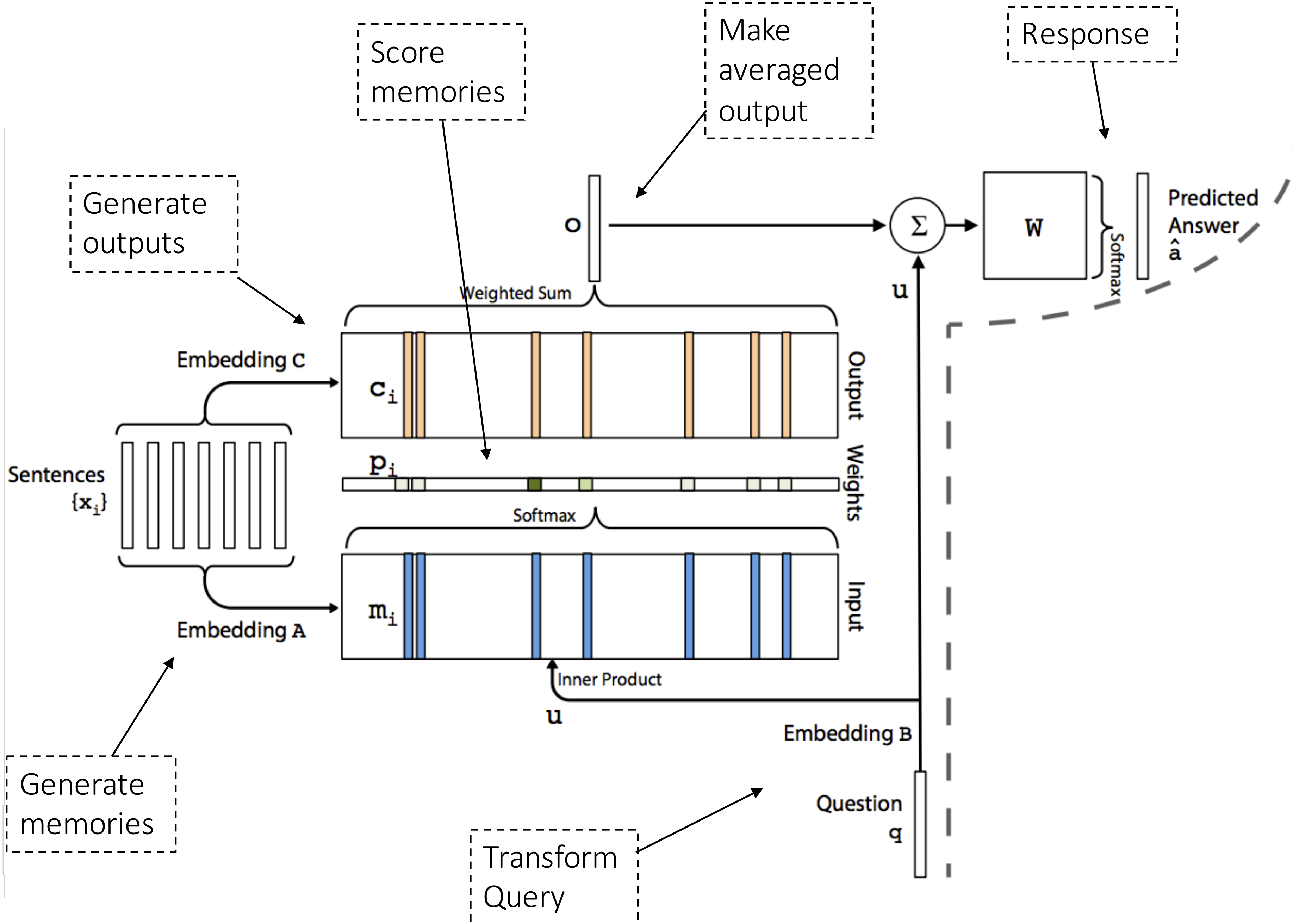
Query

Averaged-output

Training Objective – Maximum Likelihood / Cross Entropy

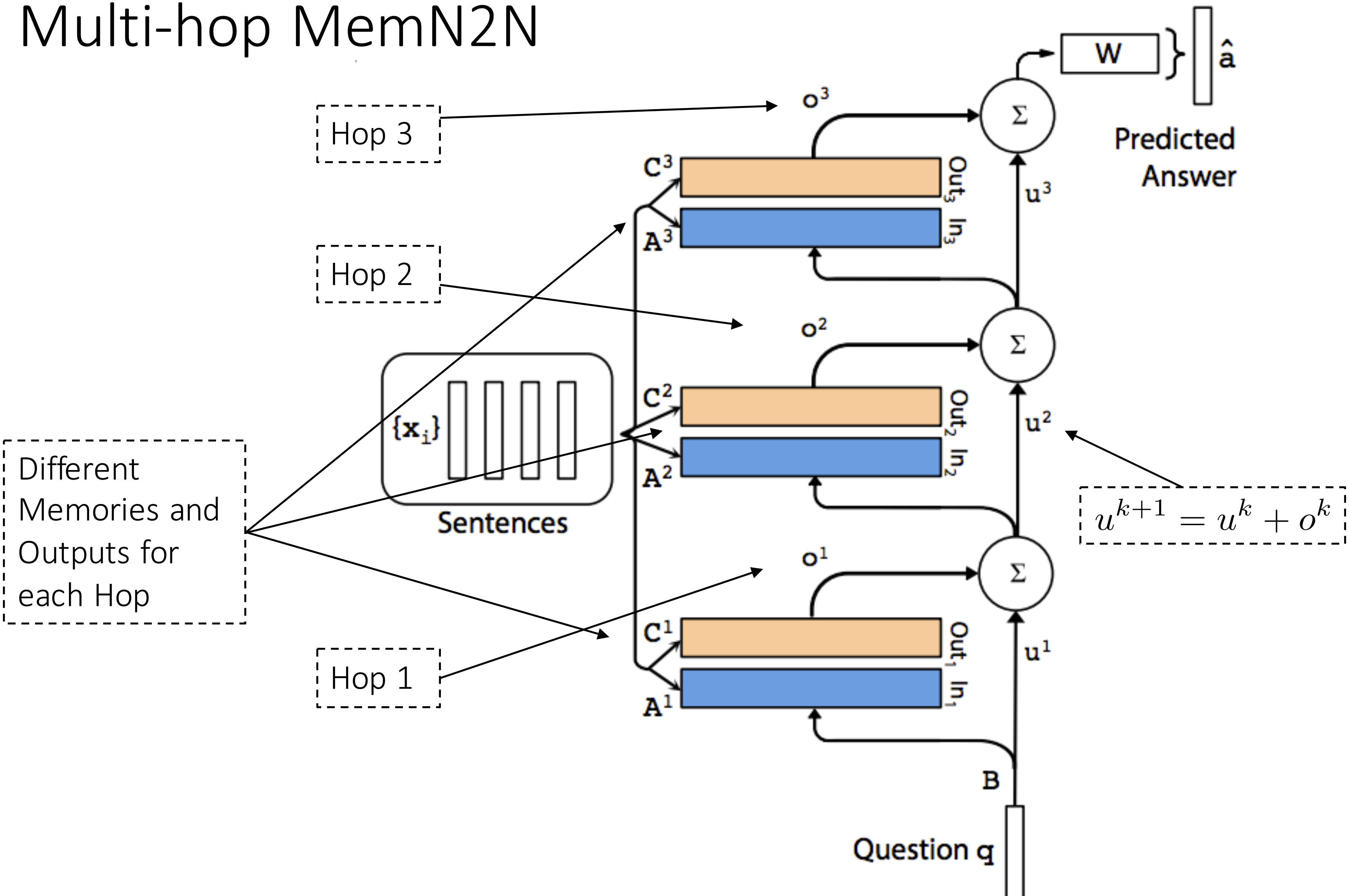
$$\hat{\Theta} = \text{argmax} \sum_{s=1}^{N} \log P(\hat{a}_s)$$

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015



End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

Multi-hop MemN2N



End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

Experiments

- Simulated World QA
  - 20 Tasks from bAbI dataset - 1K and 10K instances per task
  - Vocabulary = 177 words only!!!!!
  - 60 epochs
  - Learning Rate annealing
  - Linear Start with different learning rate
  - *“Model diverged very often, hence trained multiple models”*

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

Story (1: 1 supporting fact)	Support	Hop 1	Hop 2	Hop 3
Daniel went to the bathroom.		0.00	0.00	0.03
Mary travelled to the hallway.		0.00	0.00	0.00
John went to the bedroom.		0.37	0.02	0.00
John travelled to the bathroom.	yes	0.60	0.98	0.96
Mary went to the office.		0.01	0.00	0.00
Where is John? Answer: bathroom Prediction: bathroom				

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

	MemNN	MemN2N
Error % (1k)	6.7	12.4
Error % (10k)	3.2	7.5

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

Movie Trivia Time!

- Which was *Stanley Kubricks*'s first movie?

  Fear and Desire

- When did *2001:A Space Odyssey* release?

  1968

- After *The Shining*, which movie did its director direct?

  Full Metal Jacket

Subject | Relation | Object

(2001:a_space_odyssey, *directed_by*, stanley_kubrick)

(fear_and_dark, *directed_by*, stanley_kubrick)

…

(fear_and_dark, *released_in*, 1953)

(full_metal_jacket, *released_in*, 1987)

…

(2001:a_space_odyssey, *released_in*, 1968)

…

(the_shining, *directed_by*, stanley_kubrick)

…

(AI:artificial_intelligence, *written_by*, stanley_kubrick)

Knowledge Base

Knowledge Base?

(2001:a_space_odyssey, *directed_by*, stanley_kubrick)

(fear_and_dark, *directed_by*, stanley_kubrick)

...

(fear_and_dark, *released_in*, 1953)

(full_metal_jacket, *released_in*, 1987)

...

(2001:a_space_odyssey, *released_in*, 1968)

...

(the_shining, *directed_by*, stanley_kubrick)

...

(AI:artificial_intelligence, *written_by*, stanley_kubrick)

Incomplete!

Textual Knowledge?

WIKIPEDIA

Too Challenging!

Combine using Memory Networks?

Key-Value MemNNs for Reading Documents

- Structured Memories as Key-Value Pairs
  - Regular MemNNs have single vector for each memory
  - Key more related to question and values to answer

$$\text{Memories} = (k_1, v_1),\ (k_2, v_2),\ \ldots,\ (k_N, v_N)$$

($\boldsymbol{k}$: *Kubrick's first movie was,* $\boldsymbol{v}$: *Fear and Dark*)

Keys and Values can be Words, Sentences, Vectors etc.

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN

1. Retrieve relevant memories using Hashing Techniques

$$q \longrightarrow \boxed{\begin{array}{c}(k_1, v_1)\\ (k_2, v_2)\\ \cdots\\ (k_N, v_N)\end{array}} \longrightarrow \boxed{\begin{array}{c}(k_{h_1}, v_{h_1})\\ (k_{h_2}, v_{h_2})\\ \cdots\\ (k_{h_N}, v_{h_N})\end{array}}$$

Retrieved Relevant Memories

All Memories

Use inverted index, locality sensitive hashing, something sensible

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN

2. Score Memory-Keys

$$\boxed{p_i = \text{Softmax}(u^T m_i)}$$

$$p_{h_i} = \text{Softmax}(A\Phi_Q(q) \cdot A\Phi_K(k_{h_i}))$$

Distribution over Memory-Keys

Dot-Prod

Sum of Embeddings

BOW

Key

3. Generate Output

$$\boxed{o = \sum_i p_i c_i}$$

$$o = \sum_i p_{h_i} A\Phi_V(v_{h_i})$$

Weighted average of Memory-values

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN - Multiple Hops

In the j^{th} hop:

Query representation :

$$q_j = R_j(A\Phi_Q(q_{j-1}) + o)$$

Key Addressing

$$p_{h_i} = \text{Softmax}(A\Phi_Q(q_j) \cdot A\Phi_K(k_{h_i}))$$

Generate Response

$$\hat{a} = \text{Softmax}(A\Phi_Q(q_{H+1}) \cdot B\Phi_Y(y_i))$$

Final Hop

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN – What to store in memories?

1. KB Based :

Key: (subject, relation); Value: Object

K: (2001:a_space_odyssey, *directed_by)*; V: stanley_kubrick

2. Document Based

For each entity in document, extract 5-word window around it

Key: window; Value: Entity

K: screenplay written by and; V: Hampton

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN – Experiments

- WikiMovies Benchmark
    - Total 100K QA-pairs
    - 10% for testing

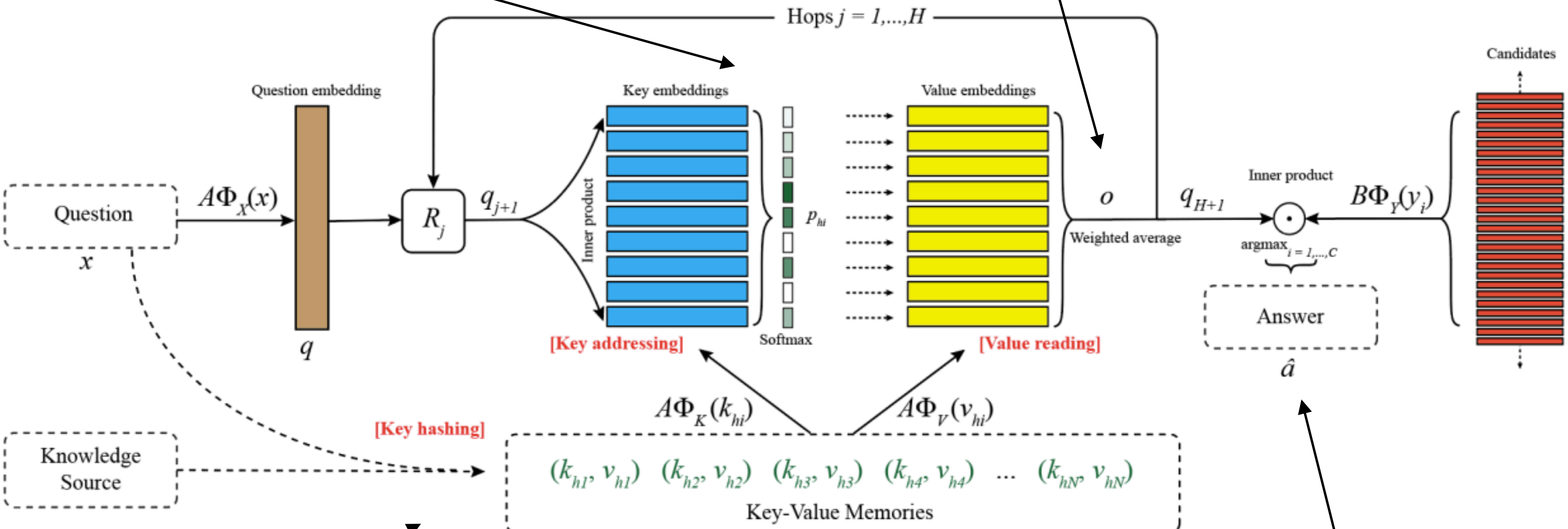
Method	KB	Doc
E2E Memory Network	78.5	69.9
Key-Value Memory Network	93.9	76.2

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN

Score relevant Memory-Keys

Generate Output using Averaged Memory-Values

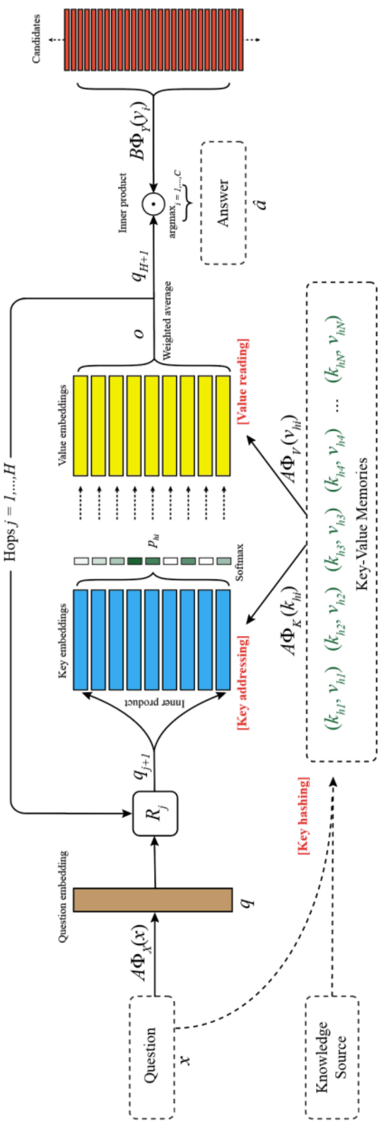


Retrieve relevant Memories

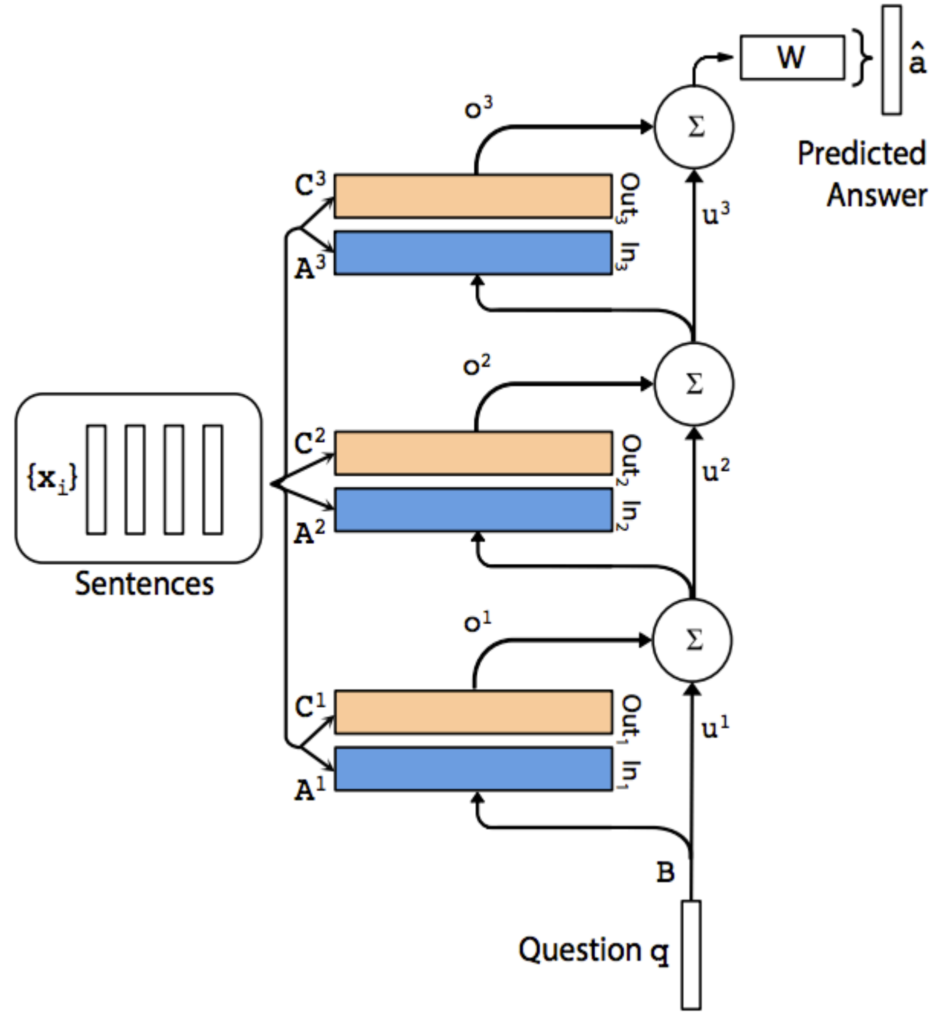
Generate Response

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

KV-MemNN



≡

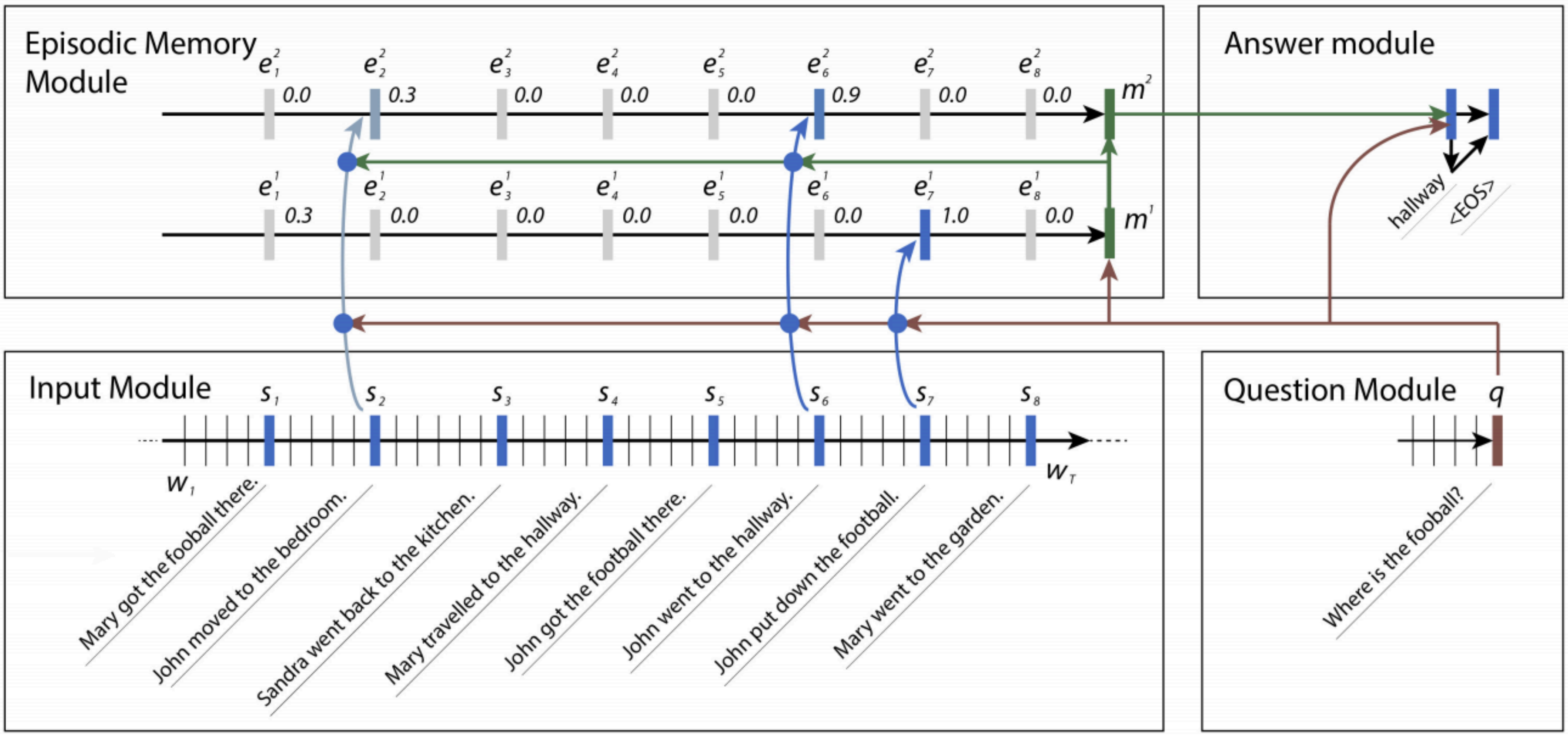


Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

CNN : Computer Vision :: ___RNN___ : NLP

Key-Value Memory Networks for Directly Reading Documents, Miller et. al., EMNLP 2016

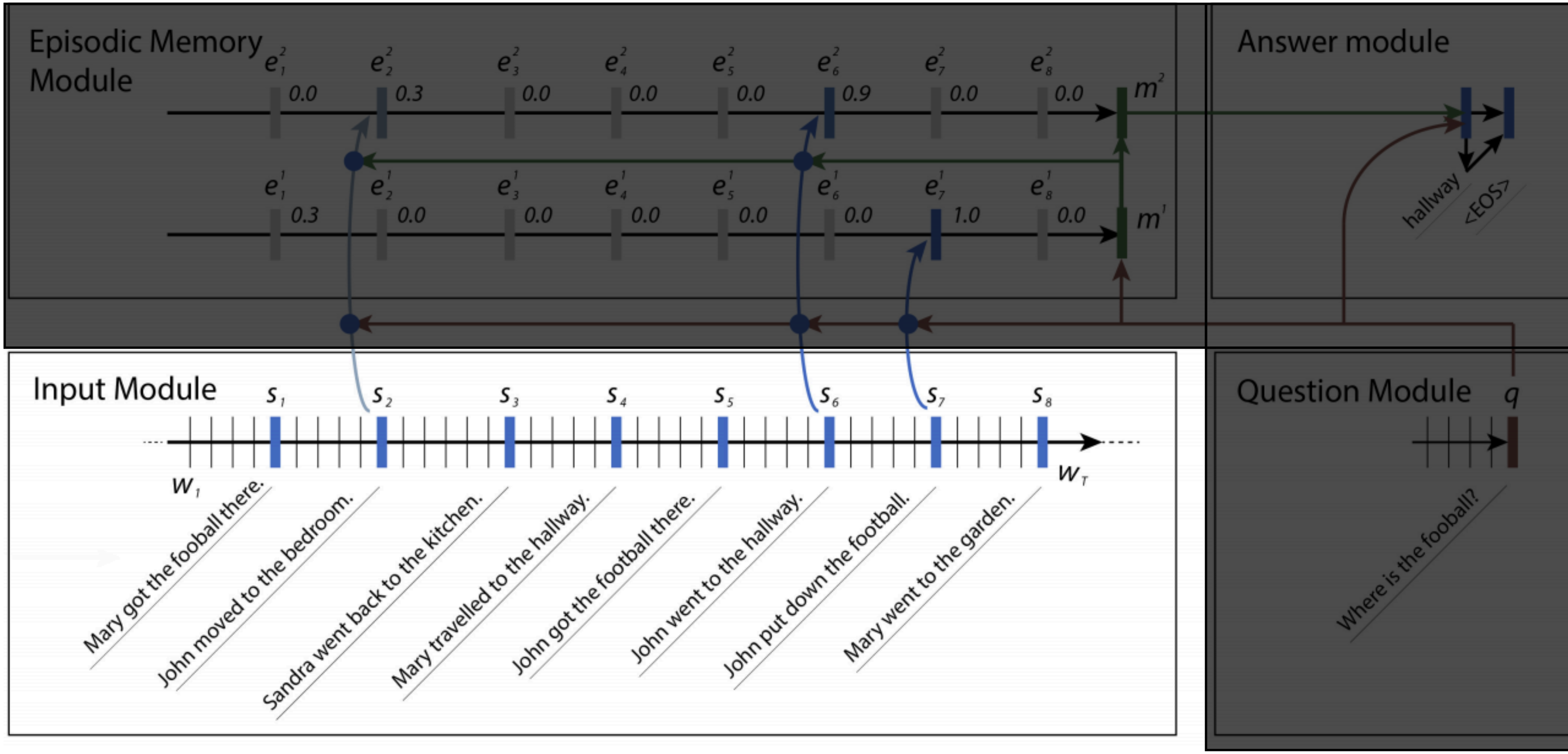
Dynamic Memory Networks – **The Beast**



Use RNNs, specifically GRUs for every module

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

DMN

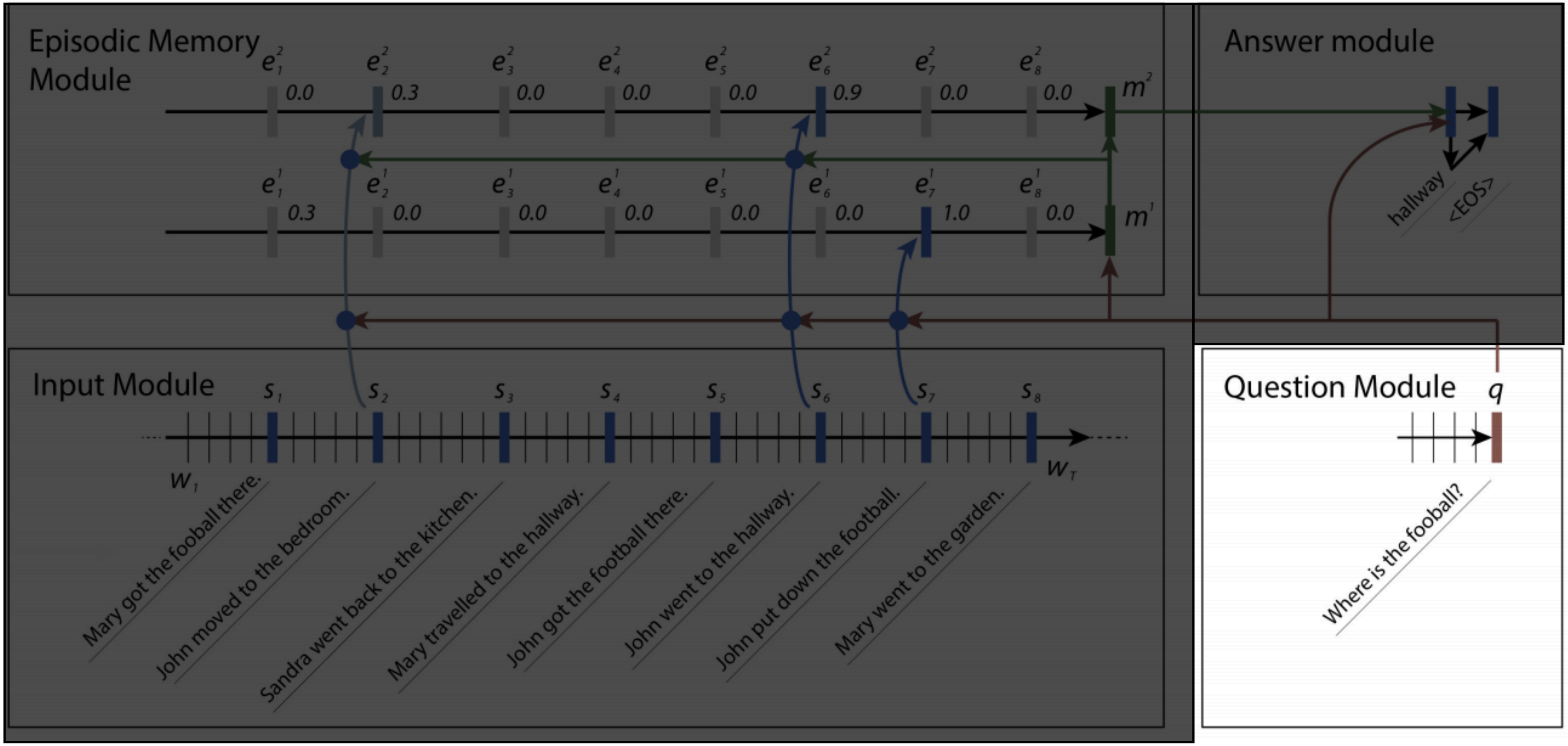


Final GRU Output for t^{th} sentence

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

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

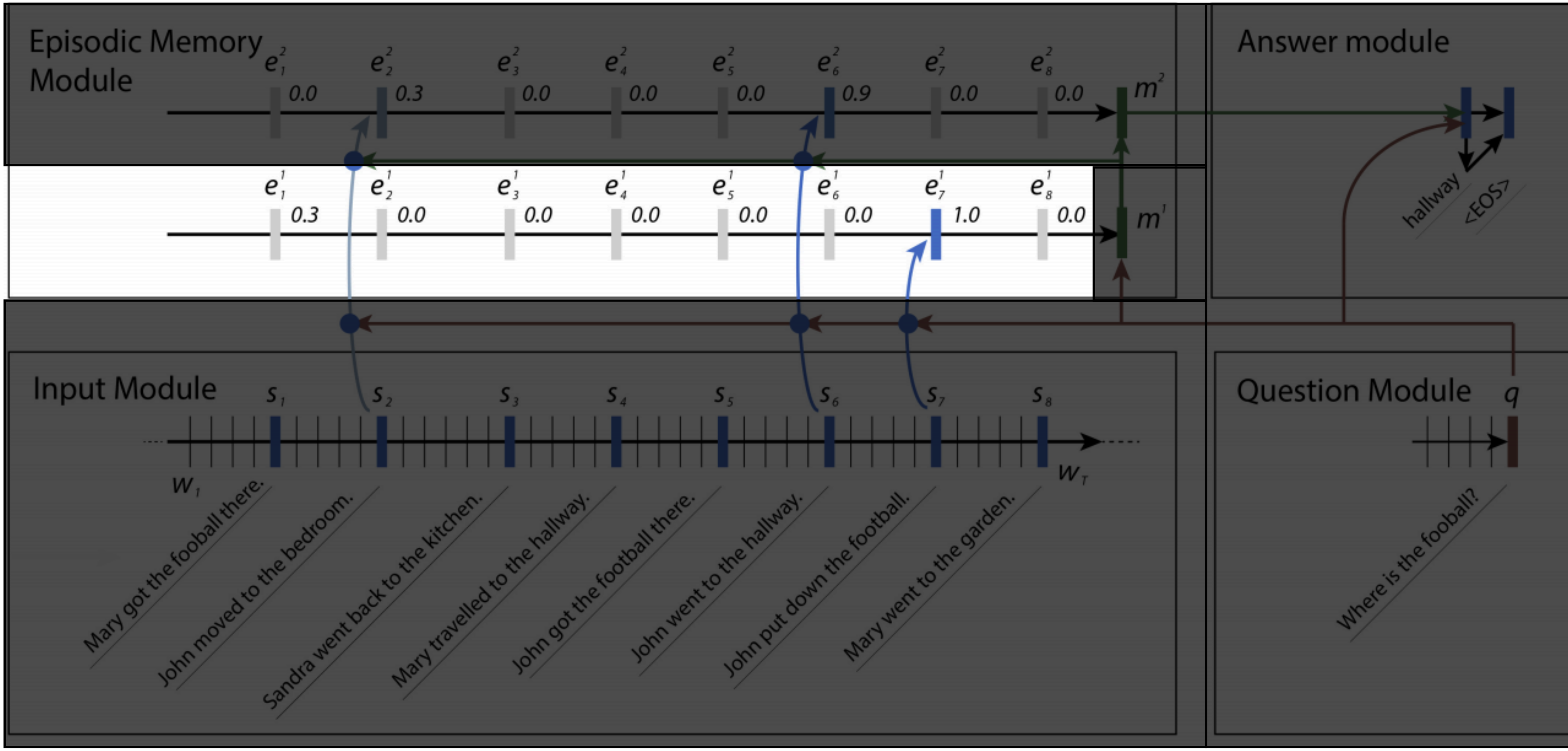
DMN



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

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

DMN



$$Hop = i$$

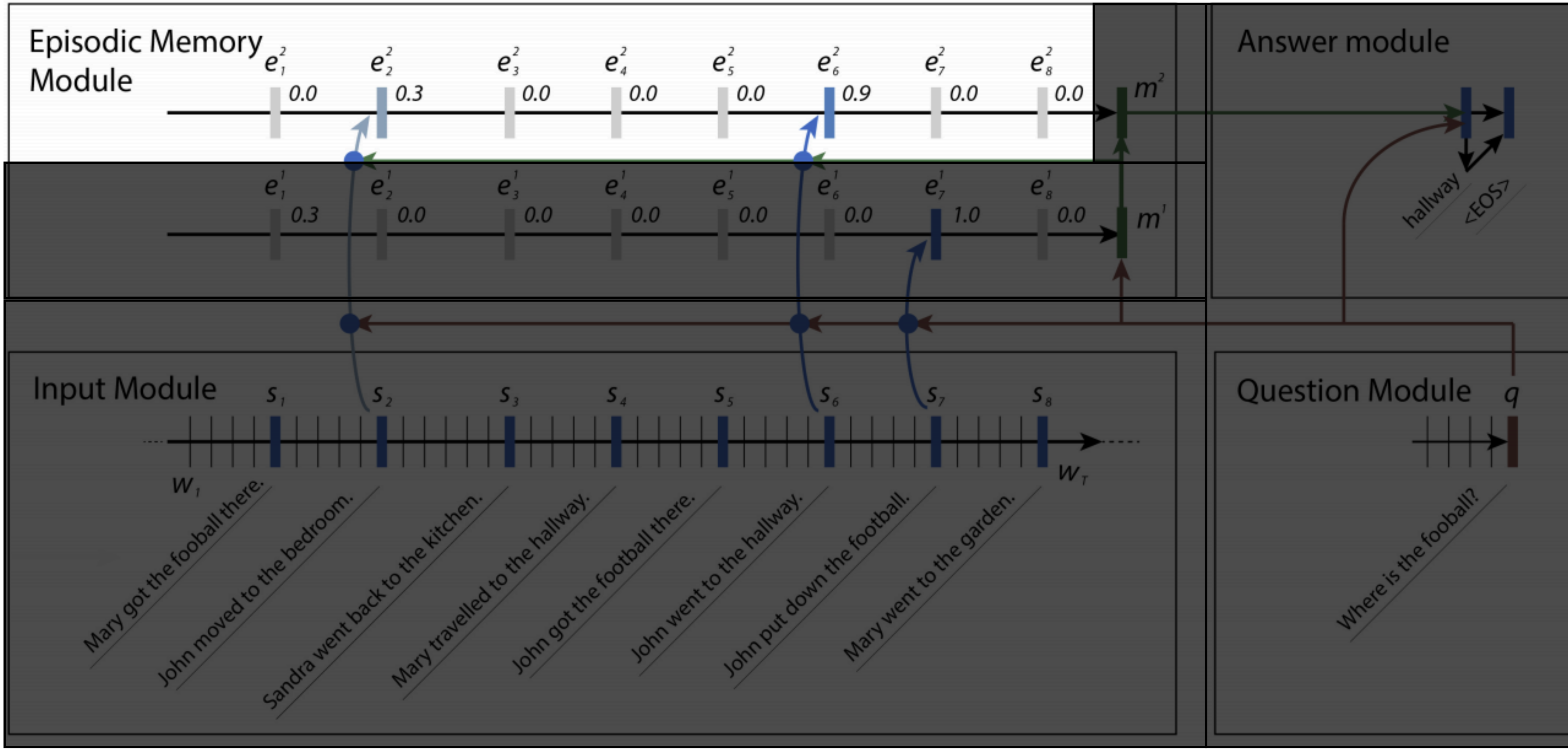
$$i = 1$$

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

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

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

DMN



$Hop = \ i$

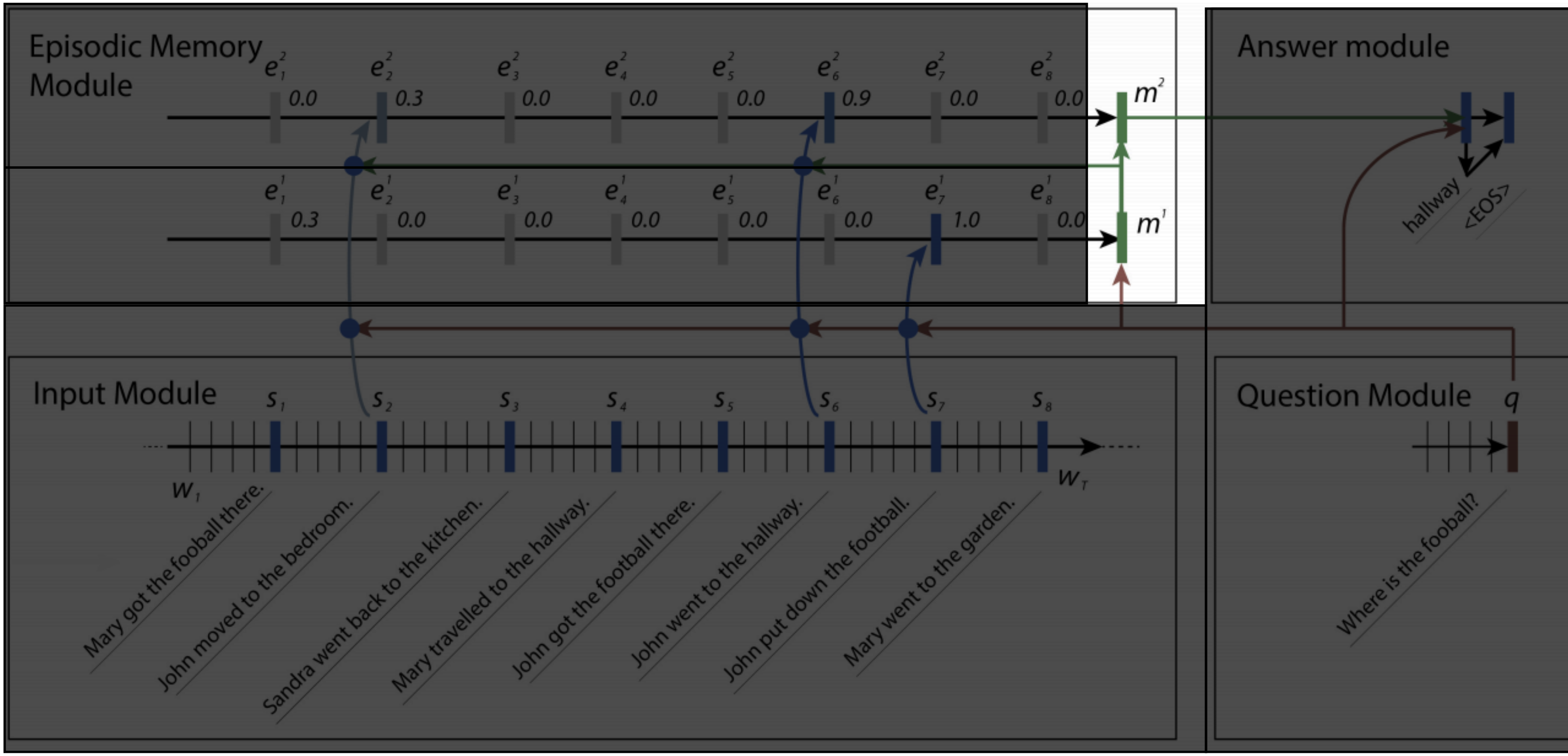
$i \ = 2$

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

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

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

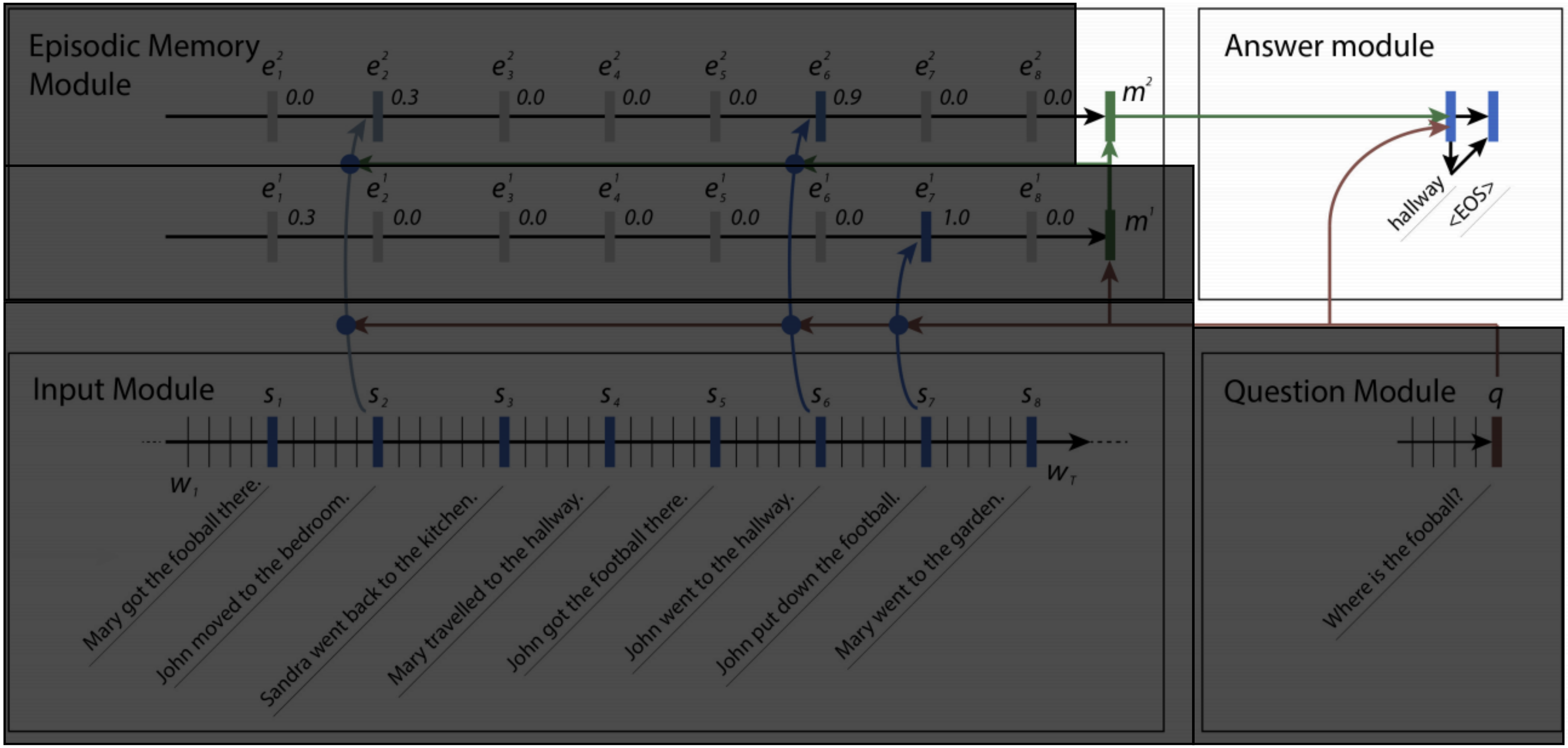
DMN



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

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

DMN

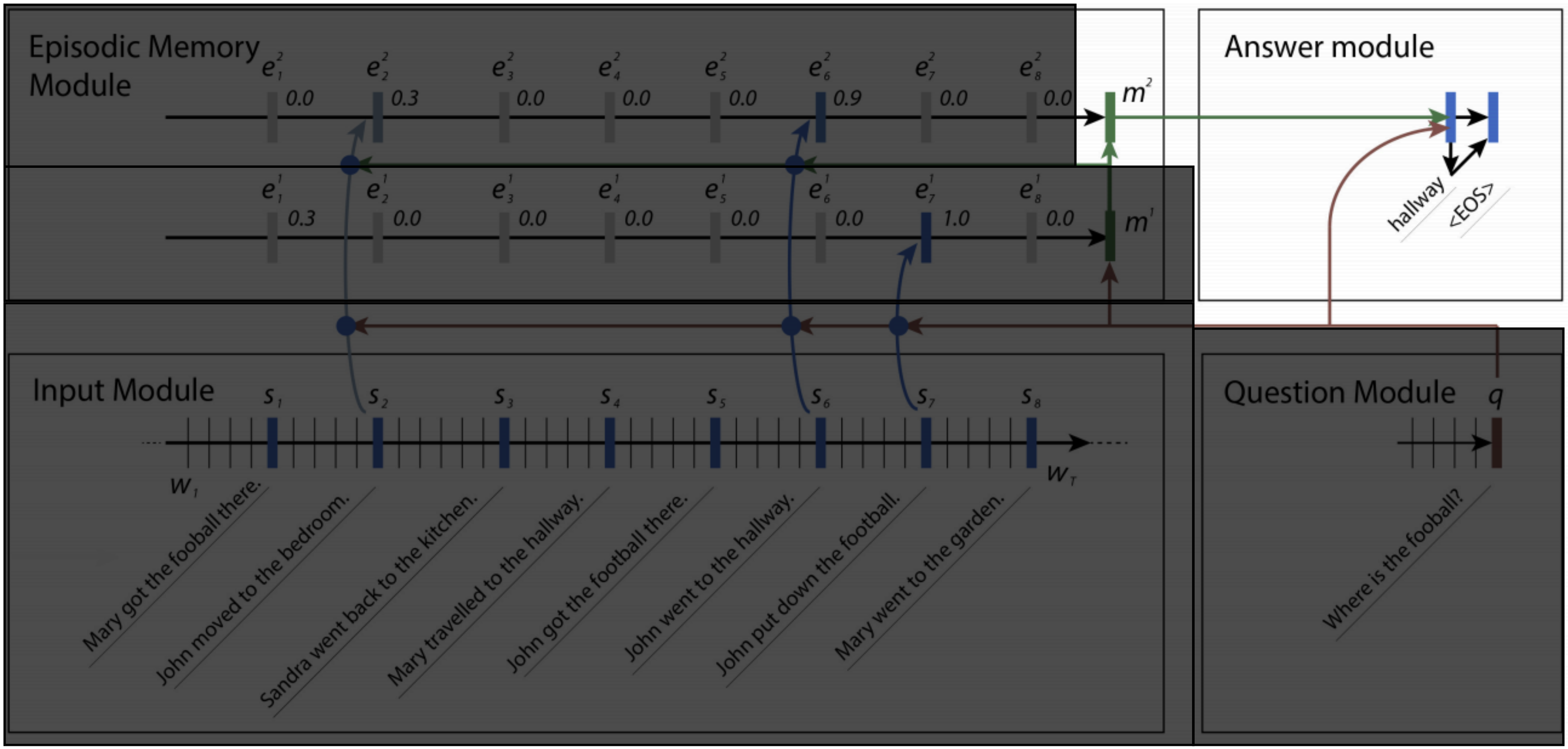


$$y_t = \text{Softmax}(W^{(a)} \alpha_t) \qquad \alpha_0 = m^{T_m}$$

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

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

DMN



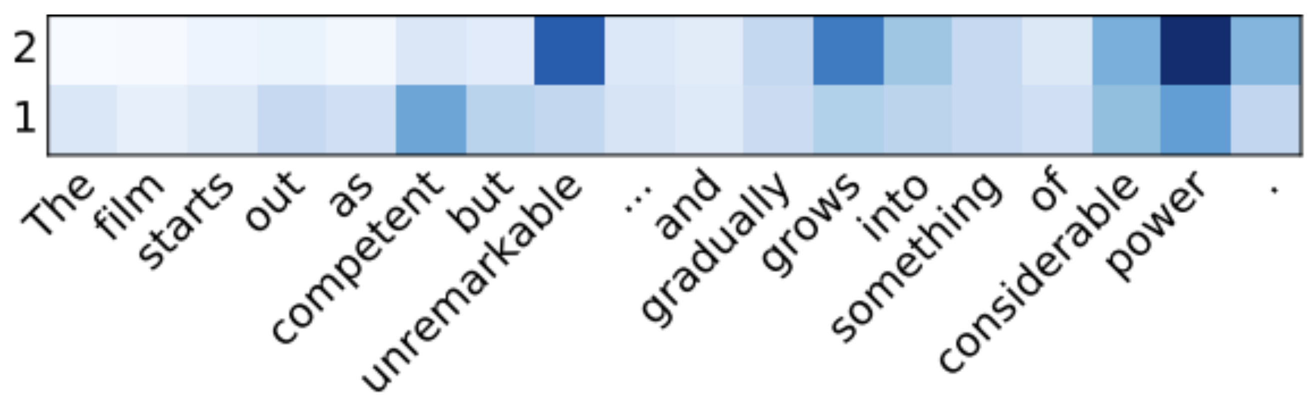
How many GRUs were used with 2 hops?

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

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]			



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

Algorithm Learning

Neural Turing Machine

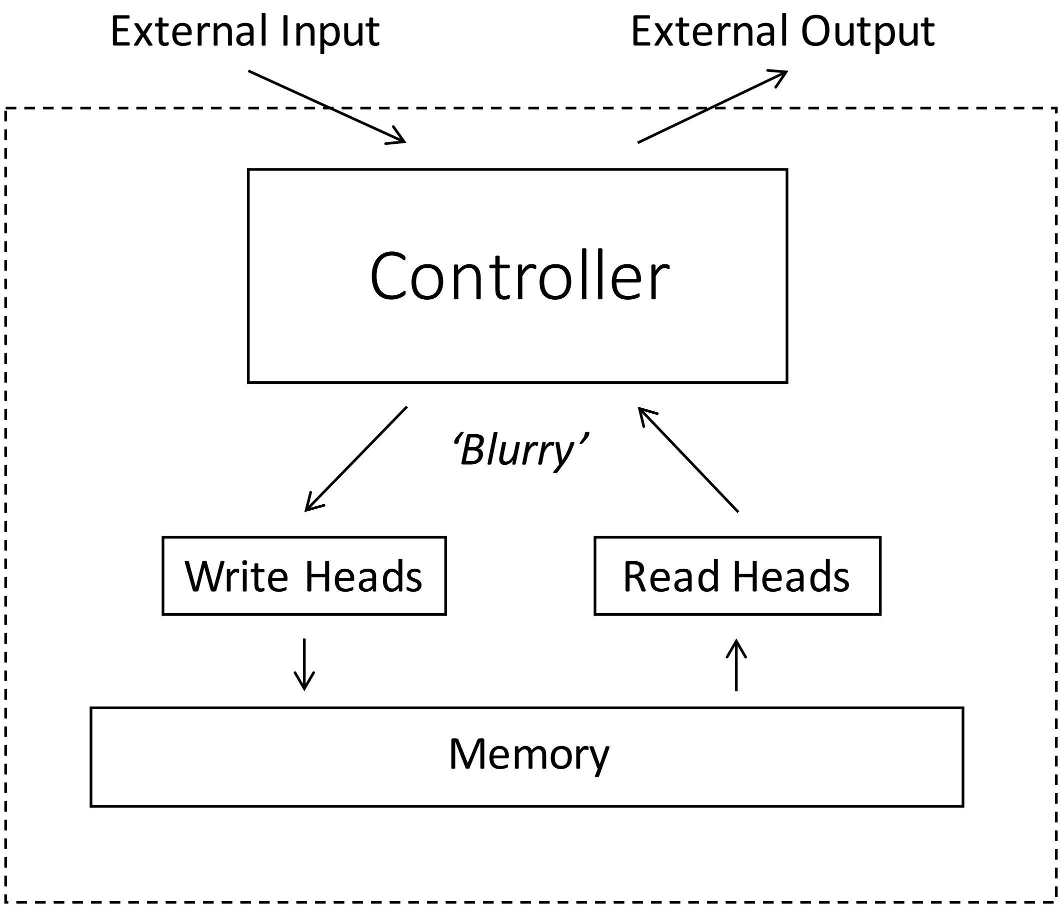
Copy Task: Implement the Algorithm

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



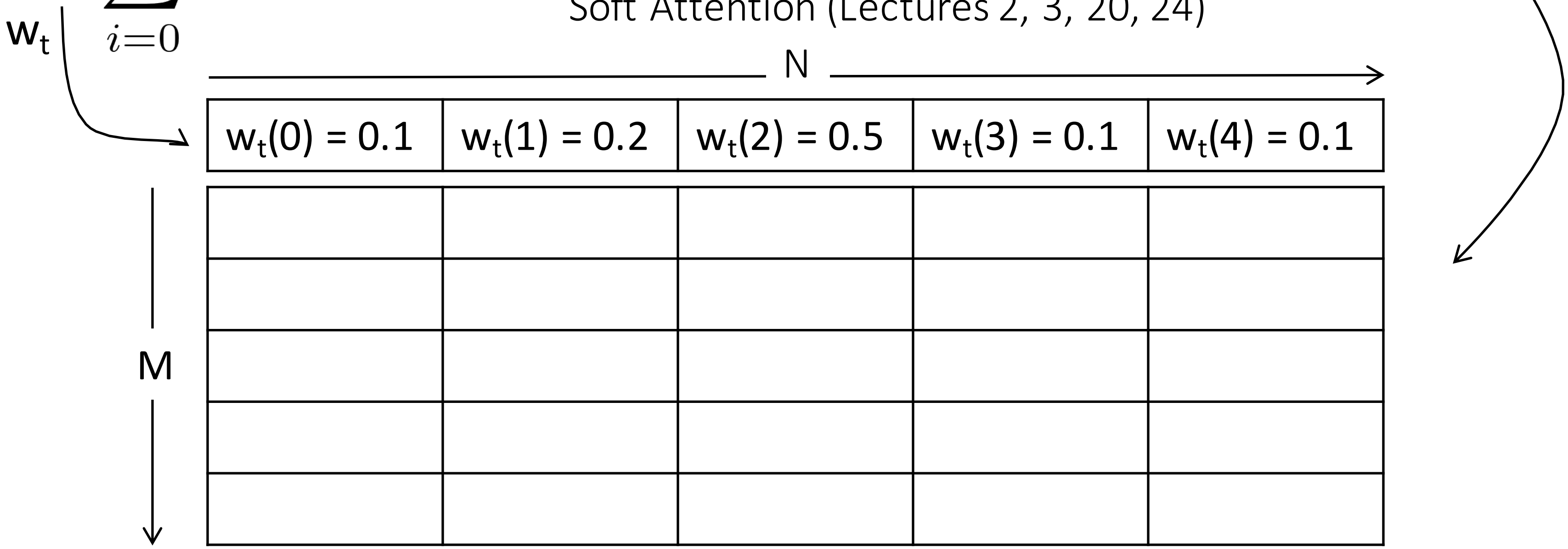
Neural Turing Machines, Graves et. al., arXiv:1410.5401

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)



Neural Turing Machines, Graves et. al., arXiv:1410.5401

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, Graves et. al., arXiv:1410.5401

Neural Turing Machines: Blurry Writes

Blurry Write Operation

Decomposed into blurry erase + blurry add

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, Graves et. al., arXiv:1410.5401

Neural Turing Machines: Erase

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

1xN

$w_1(0) = 0.1$	$w_1(1) = 0.2$	$w_1(2) = 0.5$	$w_1(3) = 0.1$	$w_1(4) = 0.1$

M_0

5	7	9	2	12
11	6	3	1	2
3	7	3	10	6
4	2	5	9	9
3	5	12	8	4

Mx1

e_1
1.0
0.7
0.2
0.5
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, Graves et. al., arXiv:1410.5401

Neural Turing Machines: Addition

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

$w_1(0) = 0.1$	$w_1(1) = 0.2$	$w_1(2) = 0.5$	$w_1(3) = 0.1$	$w_1(4) = 0.1$	a_1
4.5	5.6	4.5	1.8	10.8	3
10.23	5.16	1.95	0.93	1.86	4
2.94	6.72	2.7	9.8	5.88	-2
3.8	1.8	3.75	8.55	8.55	0
3	5	12	8	4	2

Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machines: Blurry Writes

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

$M_1 \Rightarrow$

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, Graves et. al., arXiv:1410.5401

Neural Turing Machines: Demo

Demonstration: Training on Copy Task

Figure from [Snips AI's Medium Post]

Neural Turing Machines, Graves et. al., arXiv:1410.5401

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

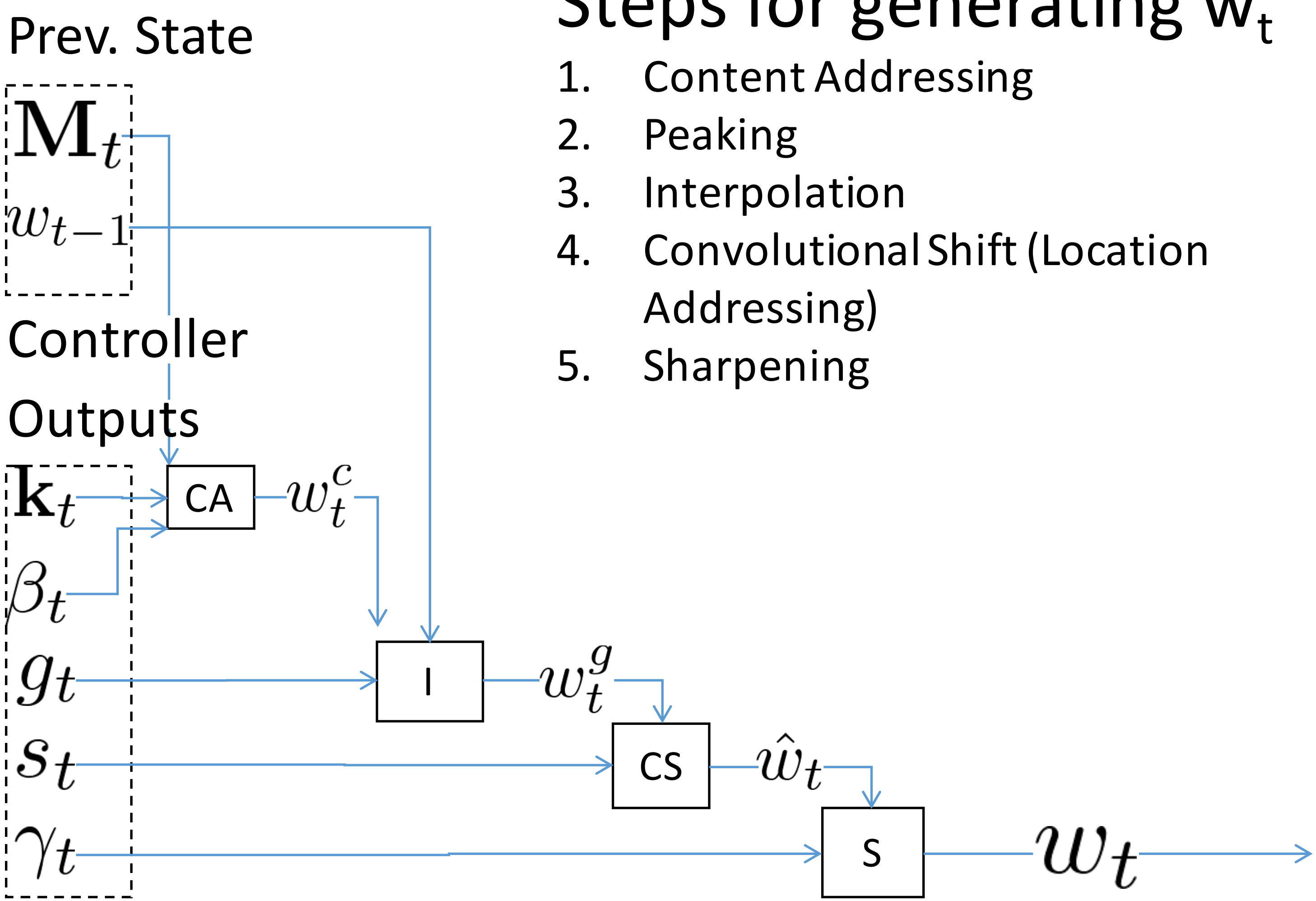
- Move to address (i+1) after writing to index (i)
- Weights ≈ Transition probabilities

Neural Turing Machines, Graves et. al., arXiv:1410.5401

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 Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Attention Model

Prev. State

$\mathbf{M}_t$

Controller

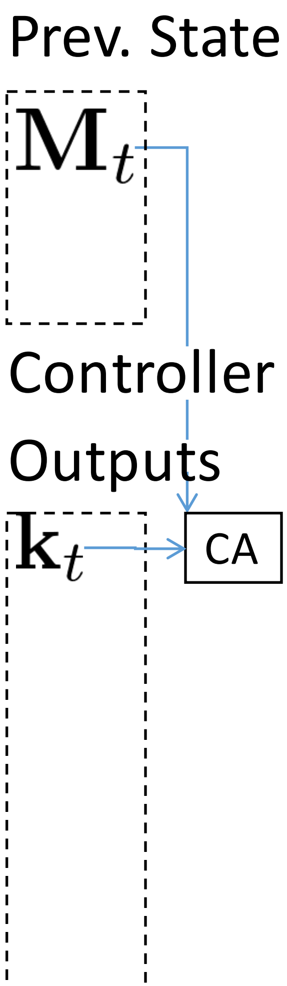
Outputs

$\mathbf{k}_t$

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

Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Attention Model

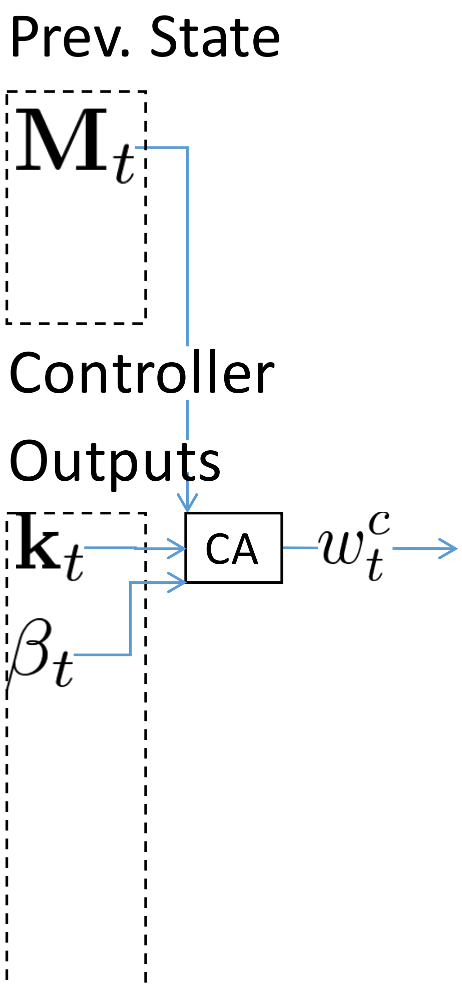


<u>Step 1: Content Addressing (CA)</u>

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

Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Attention Model



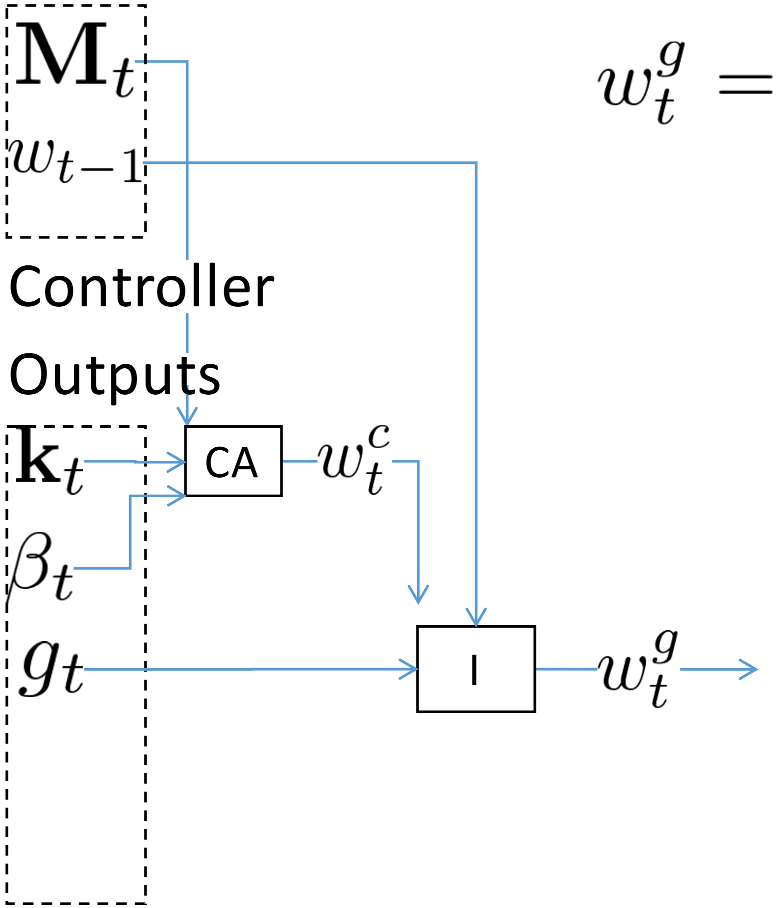
Step 2: Peaking

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

Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Attention Model

Prev. State



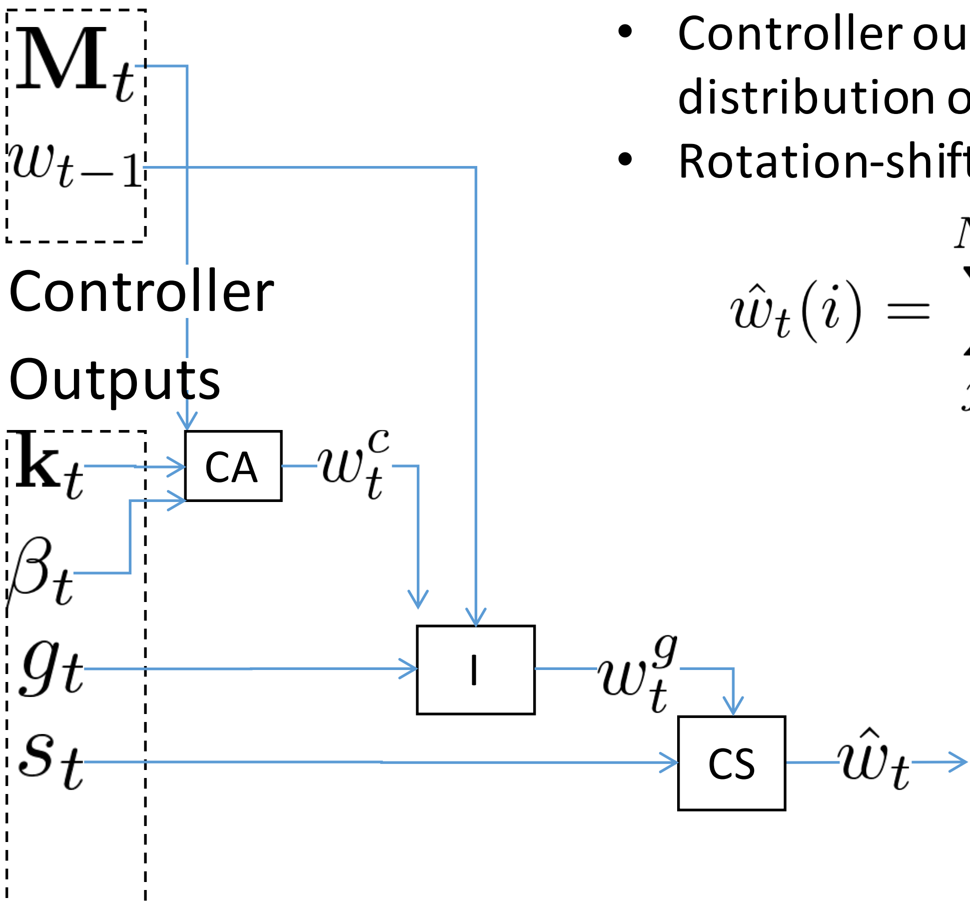
Step 3: Interpolation (I)

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

Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Attention Model

Prev. State

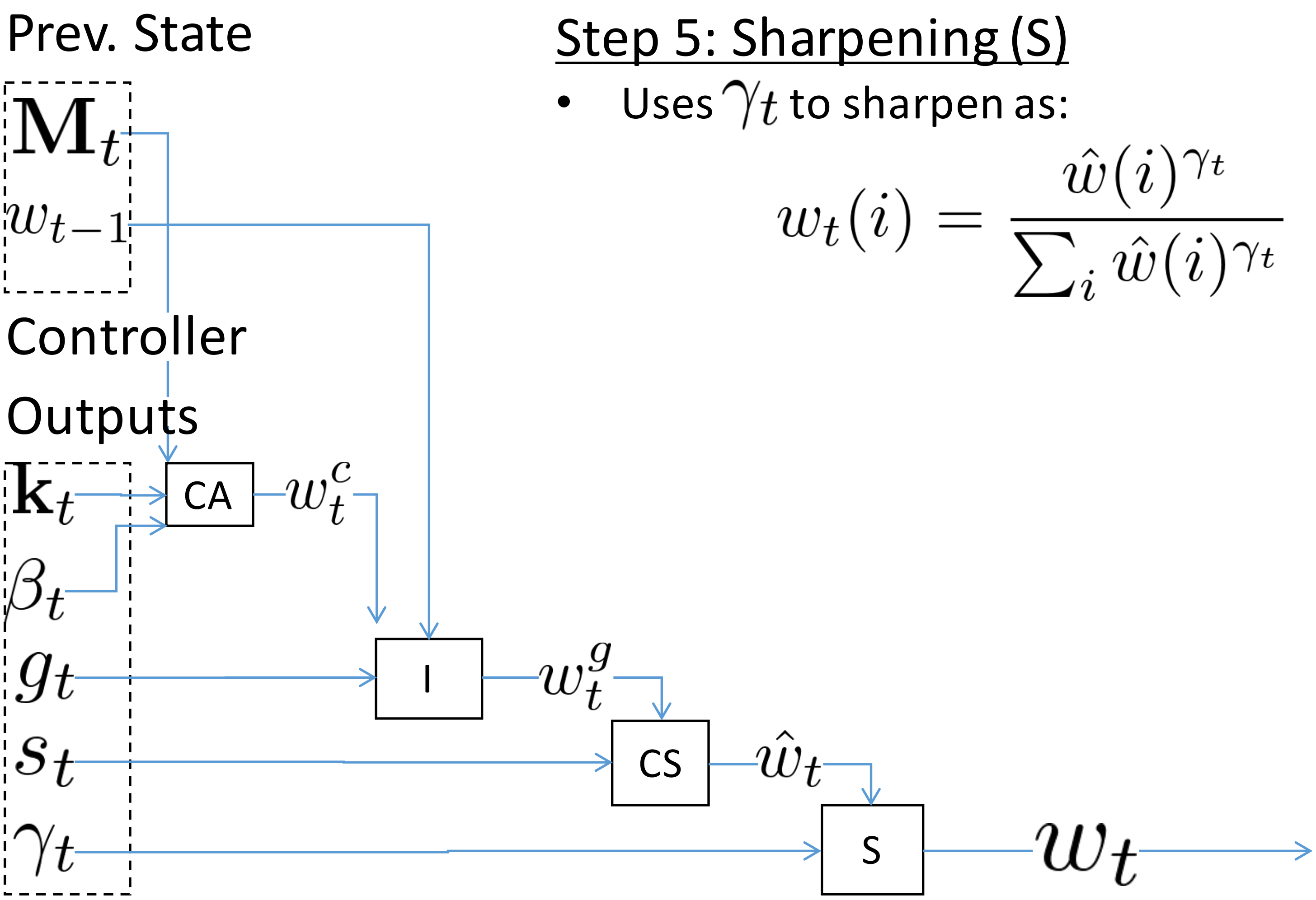


Step 4: Convolutional Shift (CS)

- 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



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 Machines, Graves et. al., arXiv:1410.5401

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 Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machine: Network Overview

Unrolled Feed-forward Controller

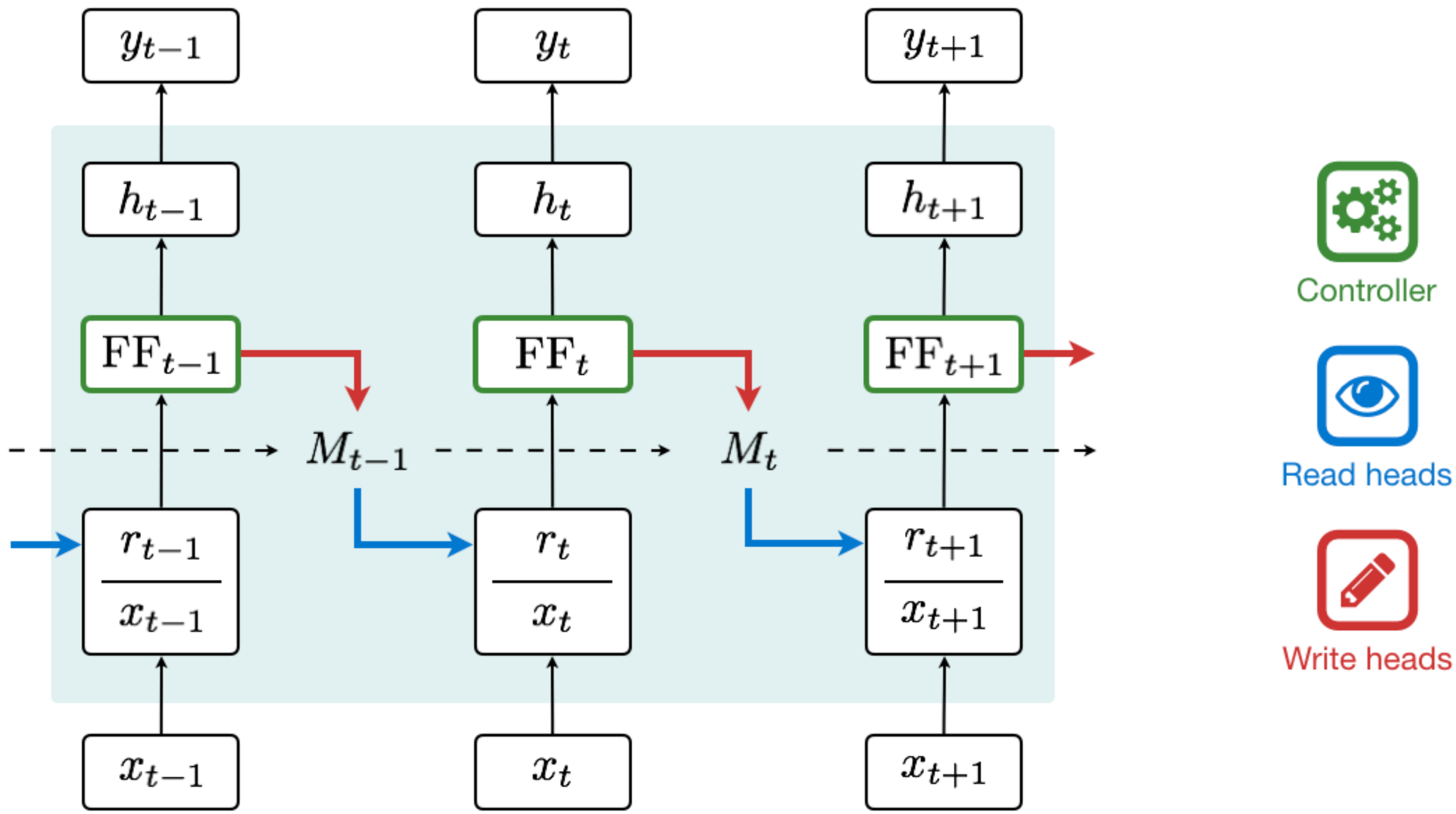


Figure from [Snips AI's Medium Post]()

Neural Turing Machines, Graves et. al., arXiv:1410.5401

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

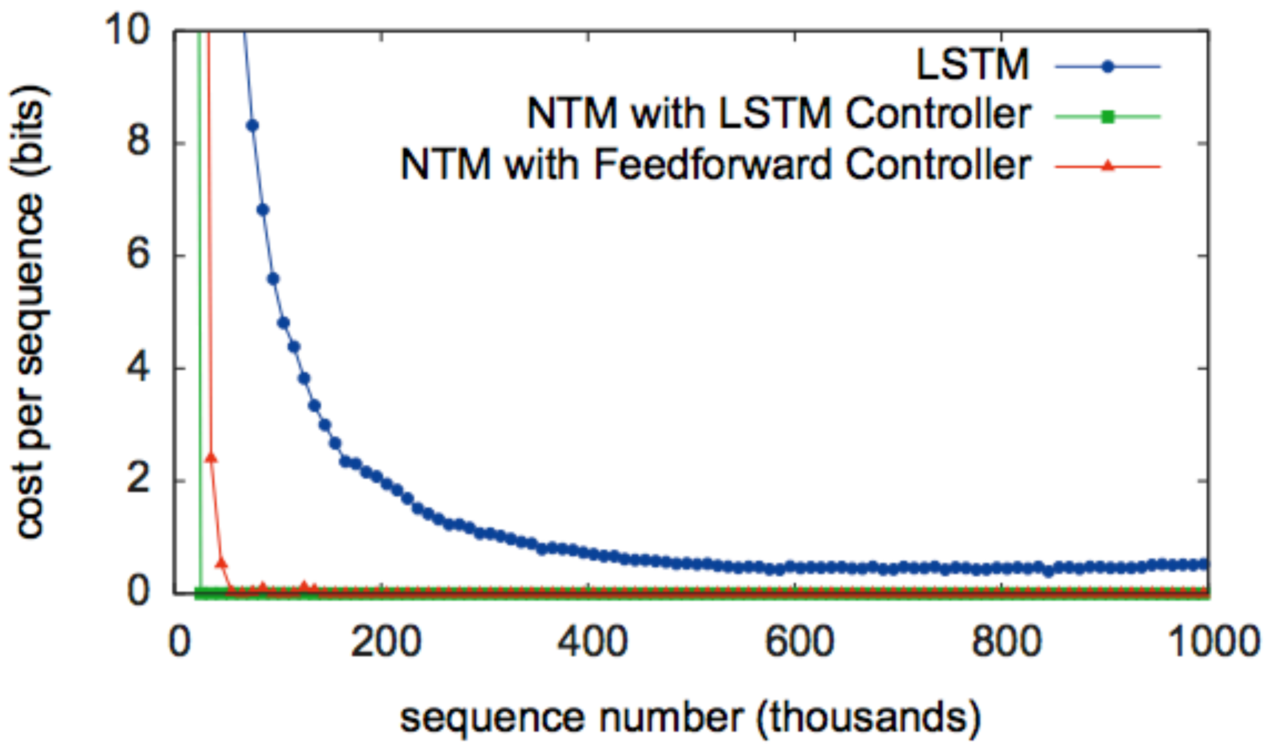
Neural Turing Machines: Experiments

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, Graves et. al., arXiv:1410.5401

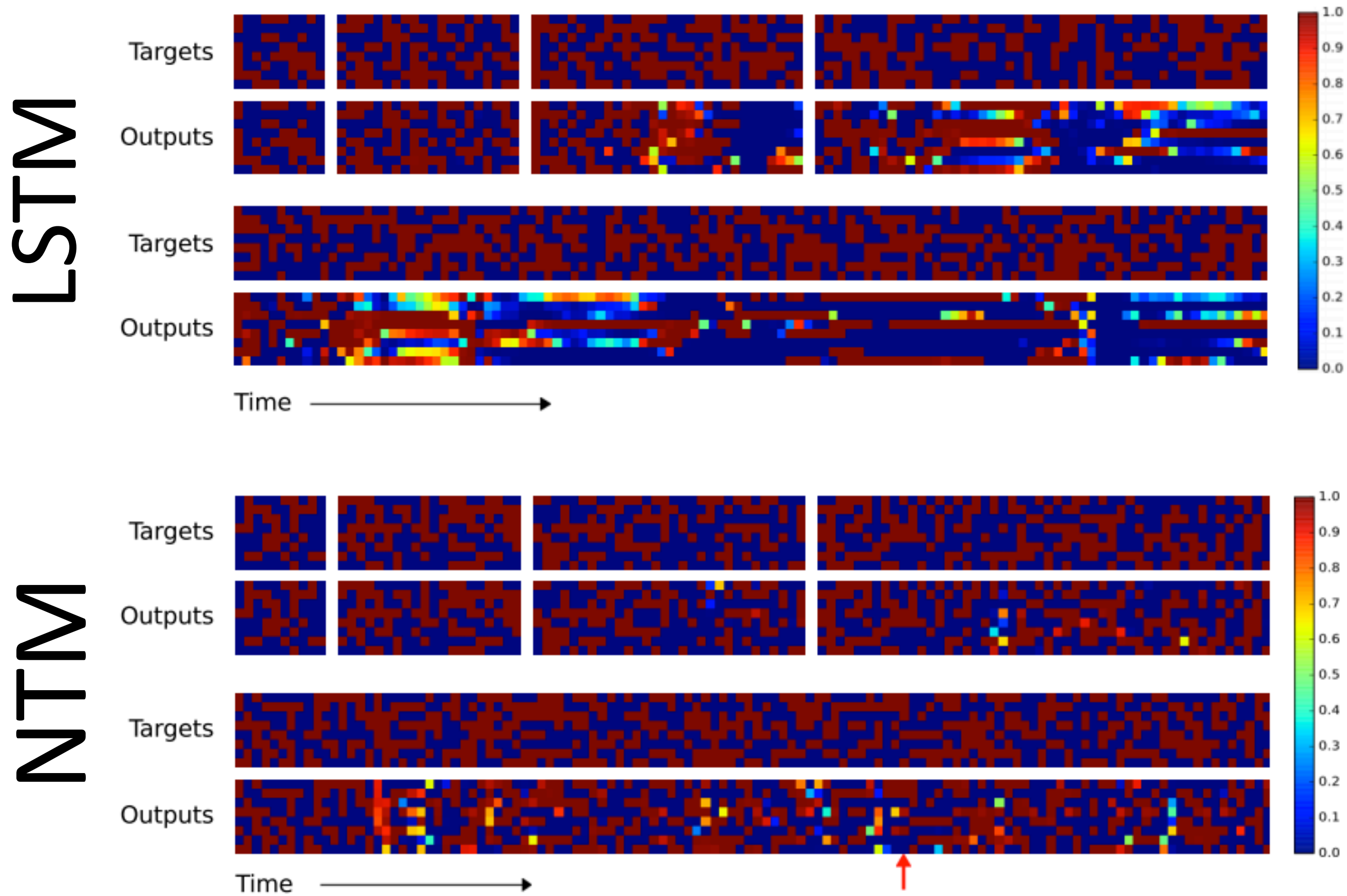
Neural Turing Machines: 'Copy' Learning Curve

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



Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machines: ‘Copy’ Performance



Neural Turing Machines, Graves et. al., arXiv:1410.5401

Neural Turing Machines triggered an outbreak of Memory Architectures!

Stack Augmented Recurrent Networks

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

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**cccca**bc**c**
$n \in [1, k]$, $X \to nXn$, $X \to =$	$(k = 2)$ 12=**21**2122=**2212**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

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%

Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes, Gulchere et. al., arXiv:1607.00036

Stack Augmented Recurrent Networks

- Blurry 'push' and 'pop' on stack. E.g.:

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

- 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

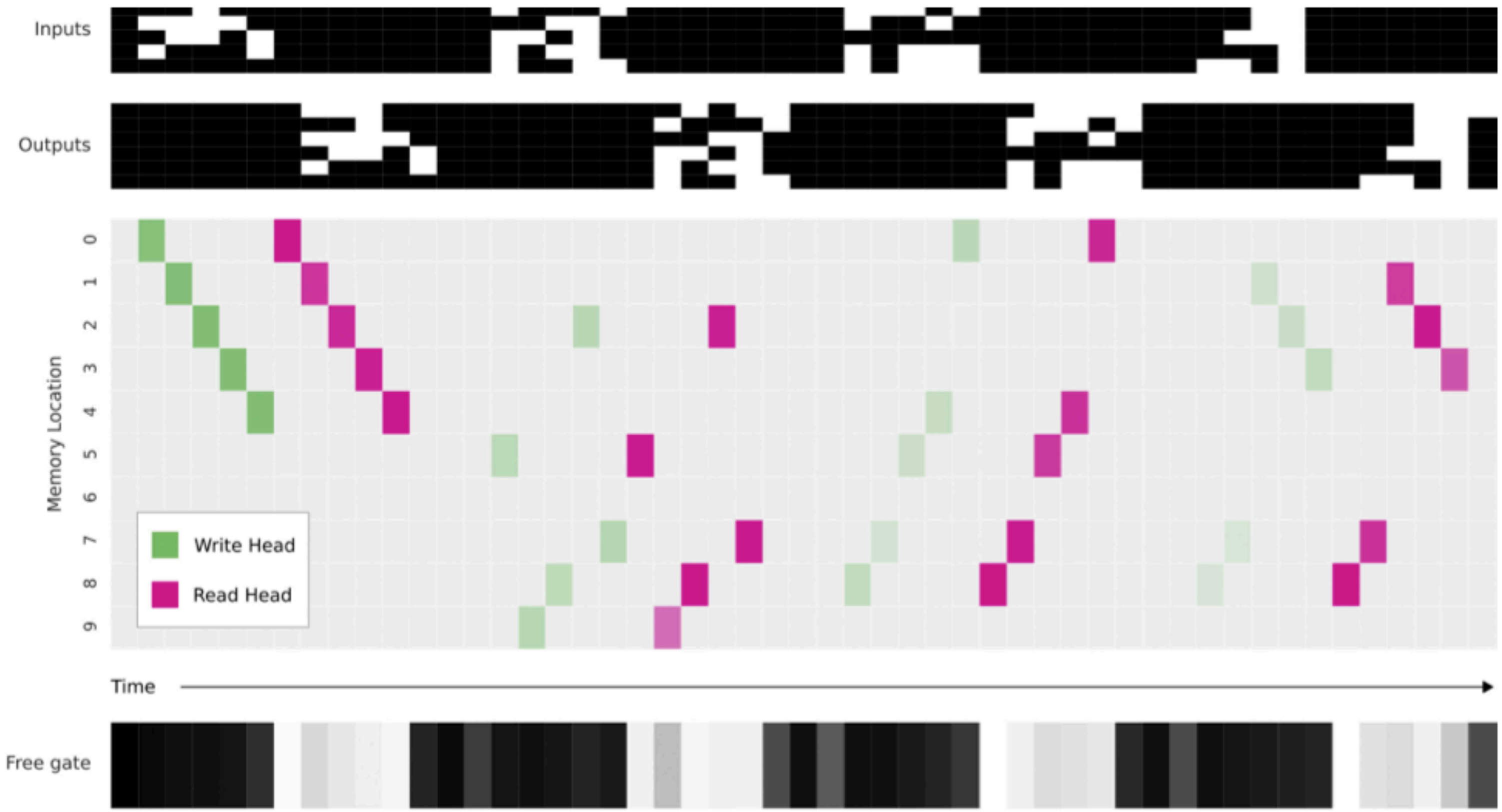
Hybrid computing using a neural network with dynamic external memory, Graves et. al., Nature *vol.* 538

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

Hybrid 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	DNC
• Large contiguous blocks of memory needed	• Memory locations non-contiguous, usage-based
• No way to free up memory cells after writing	• Regular de-allotment based on usage-tracking

Hybrid 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

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

DNC: Experiments

Graph Tasks

Training over 3 phases:

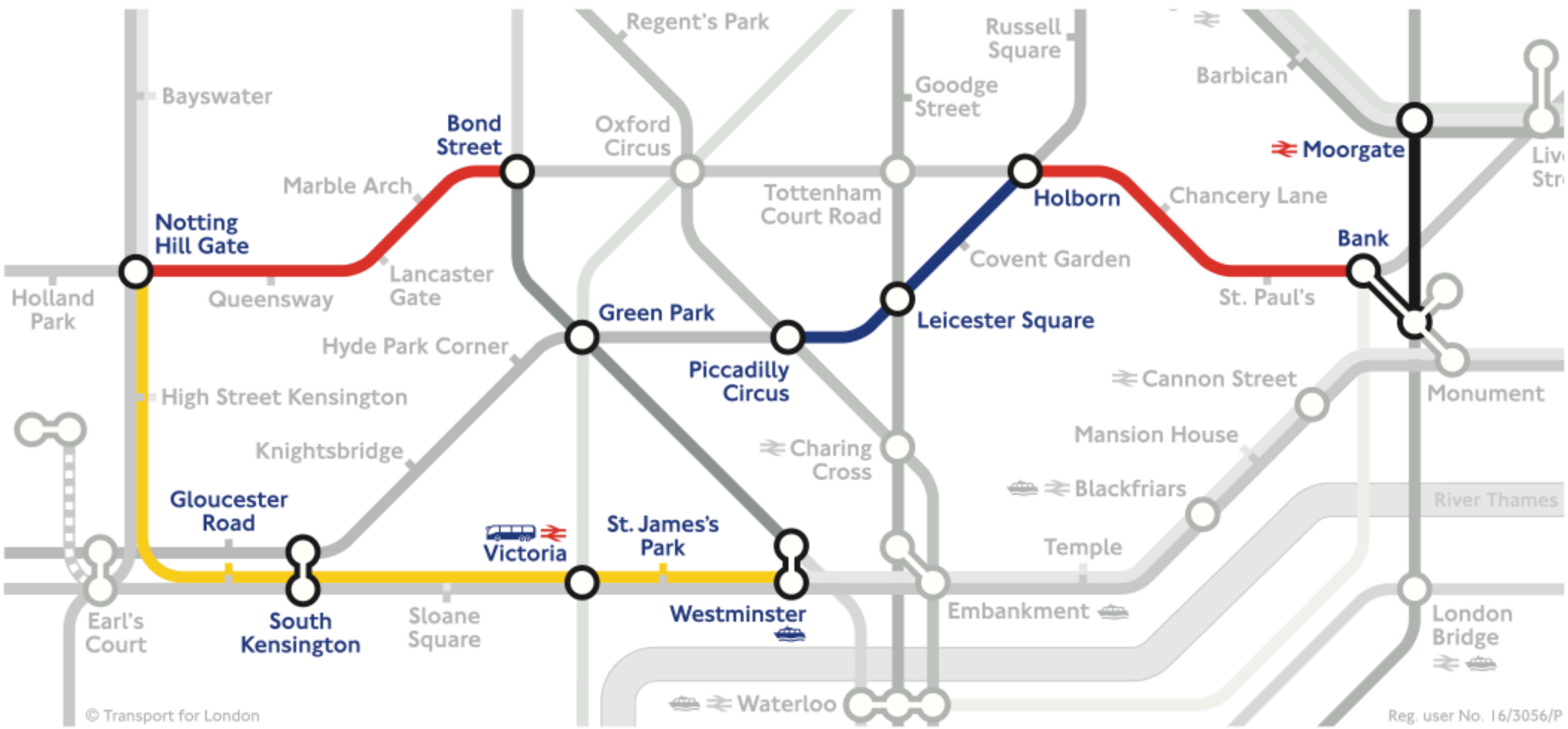
- Graph description phase: (source, edge, destination) tuples fed into the graph
- Query phase: Shortest path (source, ____, destination), Inference (source, hybrid relation, ___), Traversal (source, relation, relation …, ___)
- Answer phase: Target responses provided at output

Trained on random graphs of maximum size 1000

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

DNC: Experiments

Graph Tasks: London Underground



Hybrid 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)

Hybrid 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

Hybrid 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

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

DNC: Experiments

Graph Tasks: Freya’s Family Tree

DeepMind

Differentiable Neural Computer
Family tree inference task
(artistic rendering)

Hybrid 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.

Hybrid 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**