Week 12: Memory networks

Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

16 May, 2019

Question answering (QA)

- Question answering: given a set (sequence) of facts and a question, predict the answer
- Facts (sentences), question (sentence) and answer (word)

Question answering (QA)

- Question answering: given a set (sequence) of facts and a question, predict the answer
- Facts (sentences), question (sentence) and answer (word)
- bAbI dataset: 20 toy tasks, with different types

Task 3: Three Supporting Facts

John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 20: Agent's Motivations

John is hungry.
John goes to the kitchen.
John grabbed the apple there.
Daniel is hungry.
Where does Daniel go? A:kitchen
Why did John go to the kitchen? A:hungry

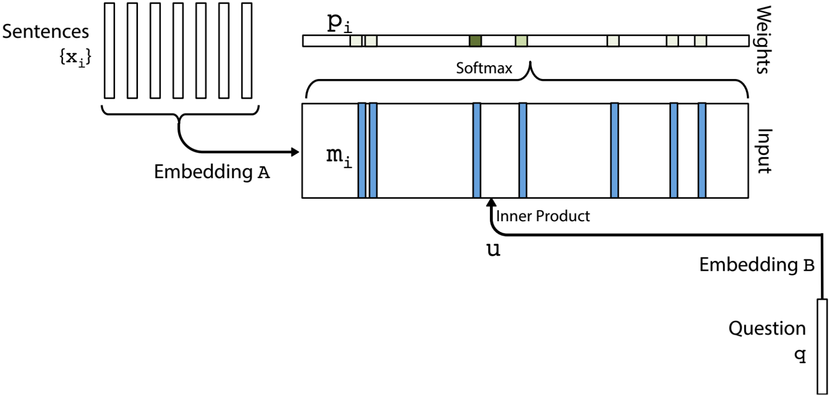
Table here from Weston et al, "Towards AI-complete question answering: a set of prerequisite toy tasks", arXiv, 2015

Figures in next 6 slides from Sukhbaatar, Szlam, Weston, Fergus, "End-to-end memory networks", arXiv, 2015

Memory network

- Embed a collection of facts (i.e., sentences)
- Compute an attention depending on a query (i.e., question)

$$p_i = \text{Softmax}(\mathbf{u}^T \mathbf{m}_i)$$

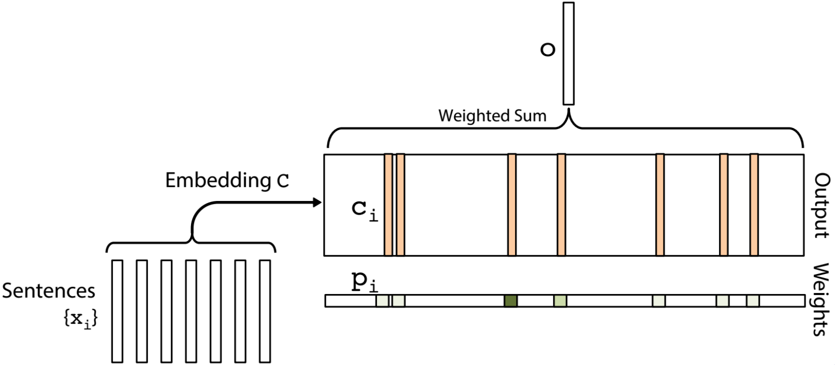


Memory network

- (Intermediate) output is weighted sum of the fact embeddings

$$\mathbf{o} = \sum_i p_i \mathbf{c}_i$$

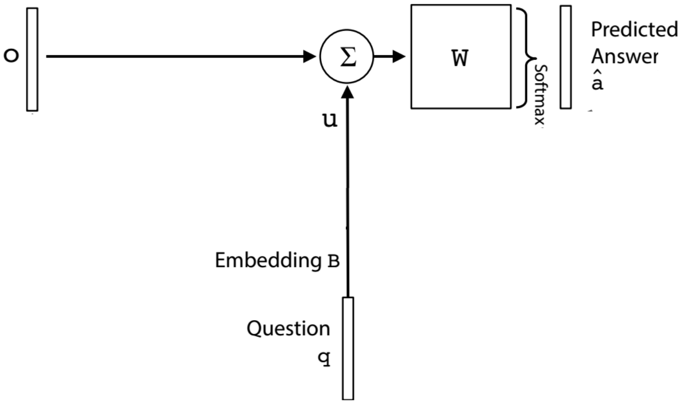
- Input embedding is different from output embedding



Memory network

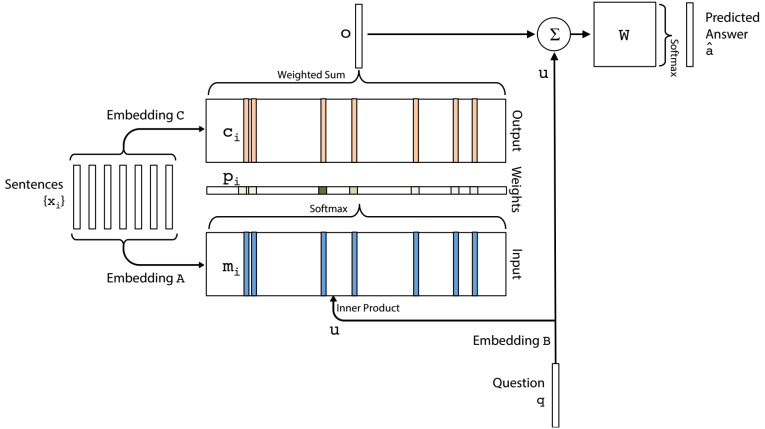
- The output $\mathbf{o}$ is then used as input to classifier
- Question embedding is also part of input to classifier

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



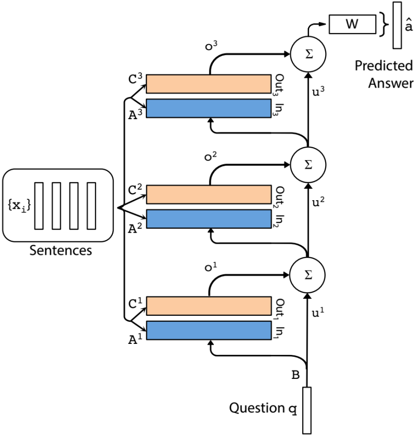
Memory network

- Model parameters: embeddings $\mathbf{A}$, $\mathbf{B}$,$\mathbf{C}$; dense output layer
- Trained by minimizing a cross-entropy loss between prediction $\hat{\mathbf{a}}$ and the true label $\mathbf{a}$



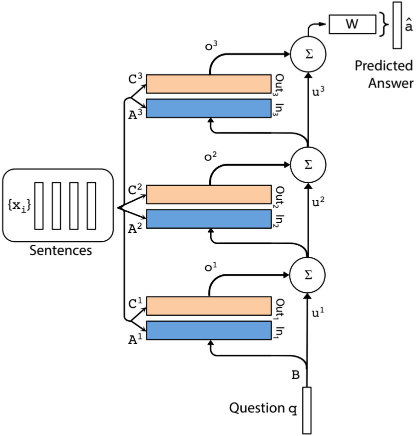
Memory network

- Several 'hops' of 'reasoning' to produce the right answer



Memory network

- Several 'hops' of 'reasoning' to produce the right answer
- Embedding layers can be tied: $\mathbf{A}_1 = \mathbf{A}_2 = \mathbf{A}_3$, $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C}_3$, making it similar to a RNN



Memory network

- Trained end-to-end from (Facts+Questions) to (Answers)
- Attented fact at each hop may be changed during inference

Story (2: 2 supporting facts)	Support	Hop 1	Hop 2	Hop 3
John dropped the milk.		0.06	0.00	0.00
John took the milk there.	yes	0.88	1.00	0.00
Sandra went back to the bathroom.		0.00	0.00	0.00
John moved to the hallway.	yes	0.00	0.00	1.00
Mary went back to the bedroom.		0.00	0.00	0.00
Where is the milk? Answer: hallway	**Prediction: hallway**			

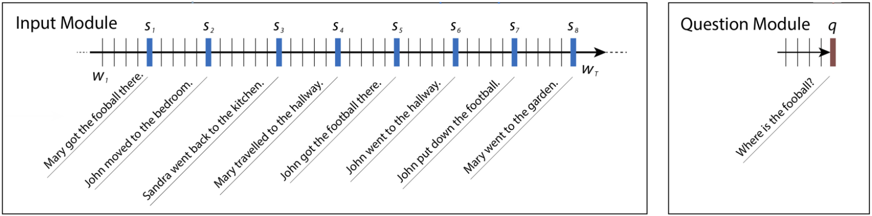
Story (18: size reasoning)	Support	Hop 1	Hop 2	Hop 3
The suitcase is bigger than the chest.	yes	0.00	0.88	0.00
The box is bigger than the chocolate.		0.04	0.05	0.10
The chest is bigger than the chocolate.	yes	0.17	0.07	0.90
The chest fits inside the container.		0.00	0.00	0.00
The chest fits inside the box.		0.00	0.00	0.00
Does the suitcase fit in the chocolate? Answer: no	**Prediction: no**			

Limitation of the memory network

It did not consider temporal order between sentences

Dynamic memory network

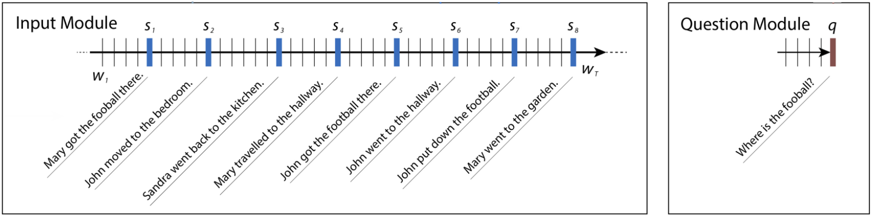
- Multiple facts (sentences) are connected (by 'end-of-sentence' token) into a single longer sequence
- A GRU RNN generates representation $\mathbf{S}_i$ for each sentence



Figures here and next 5 slides from Kumar et al., "Ask me anything: dynamic memory networks for natural language processing", ICML, 2016

Dynamic memory network

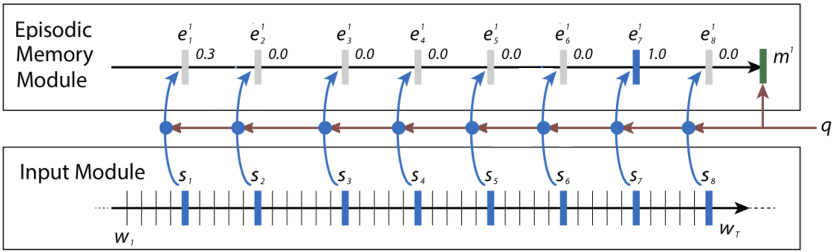
- Multiple facts (sentences) are connected (by 'end-of-sentence' token) into a single longer sequence
- A GRU RNN generates representation $\mathbf{S}_i$ for each sentence
- 2^{nd} GRU generates representation $\mathbf{q}$ for Question sentence



Figures here and next 5 slides from Kumar et al., "Ask me anything: dynamic memory networks for natural language processing", ICML, 2016

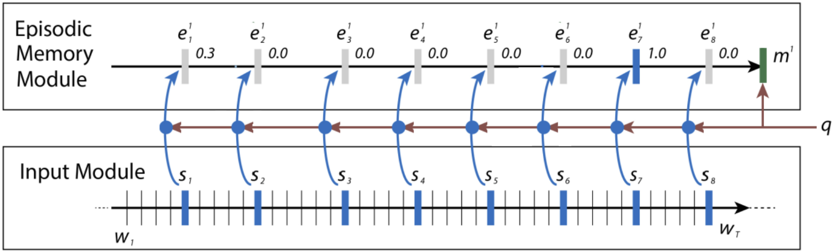
Dynamic memory network

- Attention (blue arrows & values nearby) by a 2-layer network



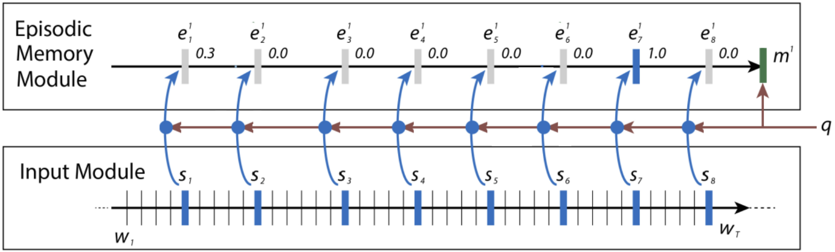
Dynamic memory network

- Attention (blue arrows & values nearby) by a 2-layer network
- Episode $\mathbf{e}_t$ by a modified 3^{rd} GRU, with attention as gate to tune GRU output
- Episode at final time step represents attended (relevant) facts



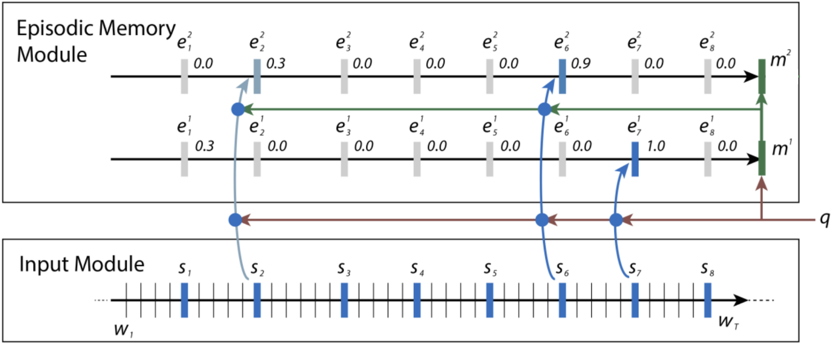
Dynamic memory network

- Attention (blue arrows & values nearby) by a 2-layer network
- Episode $\mathbf{e}_t$ by a modified 3^{rd} GRU, with attention as gate to tune GRU output
- Episode at final time step represents attended (relevant) facts
- 4^{th} GRU generates episode memory $\mathbf{m}$ ($\mathbf{e}$ and $\mathbf{q}$ as input)



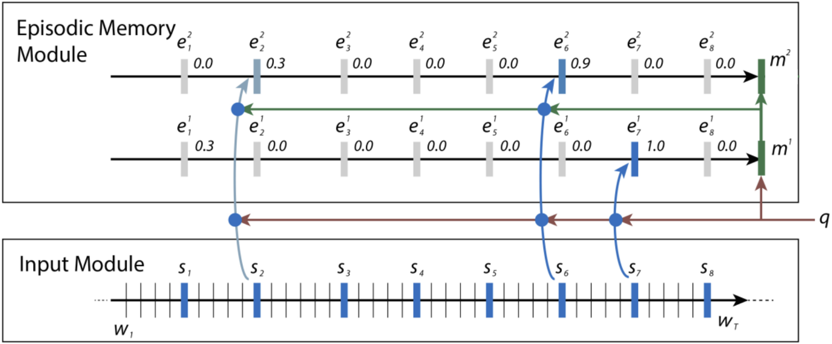
Dynamic memory network

- There are multiple ‘passes’ (iterations) in episodic memory



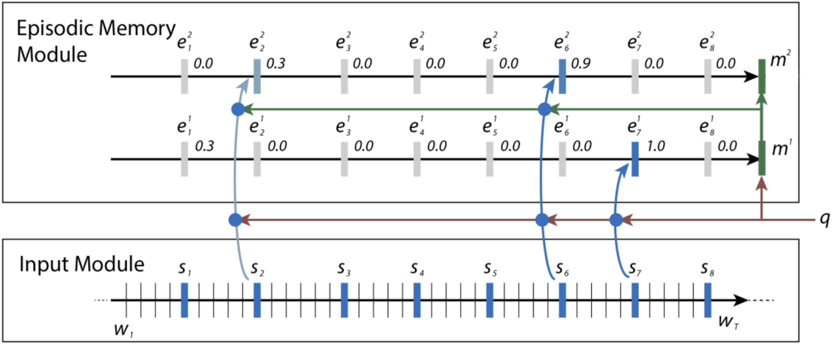
Dynamic memory network

- There are multiple 'passes' (iterations) in episodic memory
- In 2^{nd} pass: attention takes prev episodic memory $\mathbf{m}^1$ as input



Dynamic memory network

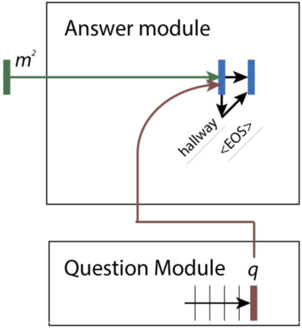
- There are multiple 'passes' (iterations) in episodic memory
- In 2^{nd} pass: attention takes prev episodic memory $\mathbf{m}^1$ as input
- $\mathbf{m}^1$ as input to 4^{th} GRU for updated episodic memory $\mathbf{m}^2$
- $\mathbf{m}$ should contain all information to answer question $\mathbf{q}$
- Note: only higher attention (blue arrows) were drawn



Dynamic memory network

- 5^{th} GRU generates answer; episodic memory $\mathbf{m}$ as initial state
- Concatenate last generated word and question vector as input

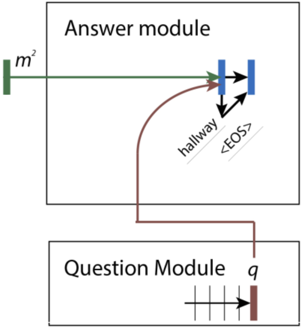
$$
\begin{aligned}
y_t &= \text{softmax}(W^{(a)} a_t) \\
a_t &= GRU([y_{t-1}, q], a_{t-1})
\end{aligned}
$$



Dynamic memory network

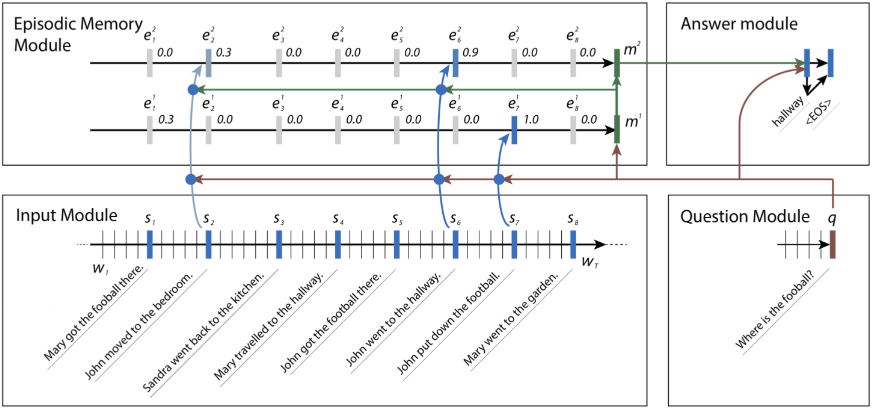
- 5^{th} GRU generates answer; episodic memory $\mathbf{m}$ as initial state
- Concatenate last generated word and question vector as input
- Trained with cross-entropy loss between (predicted single $\mathbf{y}_t$ or sequence $\{\mathbf{y}_t\}$) and (correct word/sequence)

$$
\begin{aligned}
y_t &= \mathrm{softmax}(W^{(a)} a_t) \\
a_t &= GRU([y_{t-1}, q], a_{t-1})
\end{aligned}
$$



Dynamic memory network

- DMN capture information of position and temporality
- The entire DMN can be trained end-to-end



Dynamic memory network

- Episodic memory module helps iteratively retrieve and store facts, gradually incorporating more relevant information

Max passes	task 3 three-facts	task 7 count	task 8 lists/sets
0 pass	0	48.8	33.6
1 pass	0	48.8	54.0
2 pass	16.7	49.1	55.6
3 pass	64.7	83.4	83.4
5 pass	**95.2**	**96.9**	**96.5**

Limitations of dynamic memory network

- DMN requires supporting facts (i.e., the facts that are relevant for answering a particular question) are labeled during training
- DMN did not work well on bAbI-10k without supporting facts

Limitations of dynamic memory network

- DMN requires supporting facts (i.e., the facts that are relevant for answering a particular question) are labeled during training
- DMN did not work well on bAbI-10k without supporting facts
- Reason 1: when encoding a fact, not use future sentences
- Reason 2: in input module, word-level GRU does not allow for distant supporting sentences to interact with each other

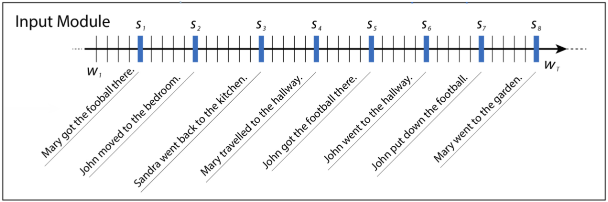
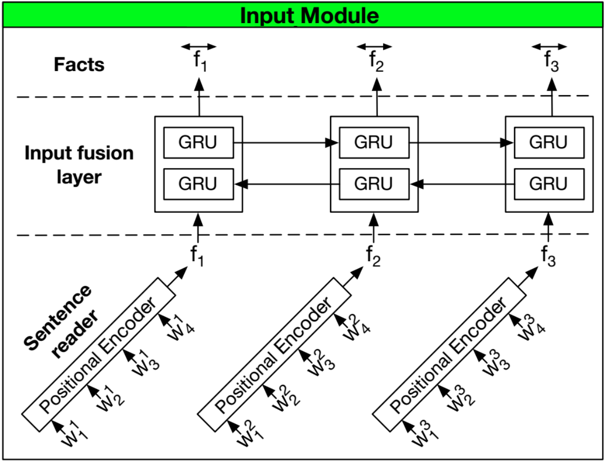


Figure in next 2 slides from Xiong, Merity, Socher, "Dynamic memory networks for visual and textual question answering", ICML, 2016

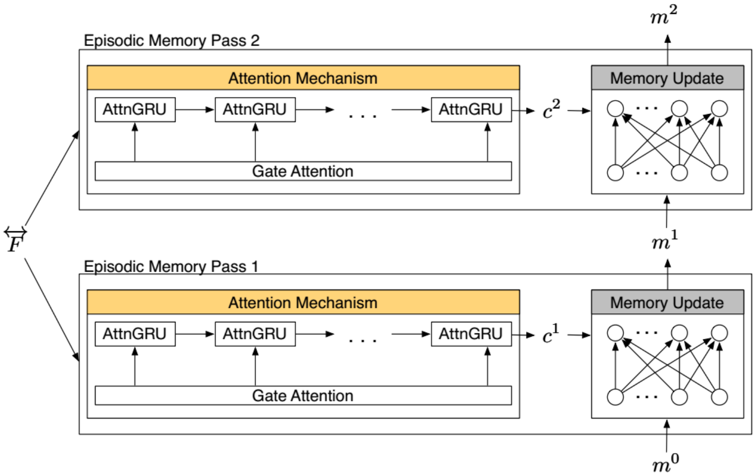
DMN+: improved dynamic memory network

- Two-level encoding of facts (sequences)
- Input fusion layer allows information flow between sentences



DMN+: improved dynamic memory network

- Attention score is used inside GRU at each time step
- Memory update: ReLU of linear embedding, unique per pass



Extending memory networks

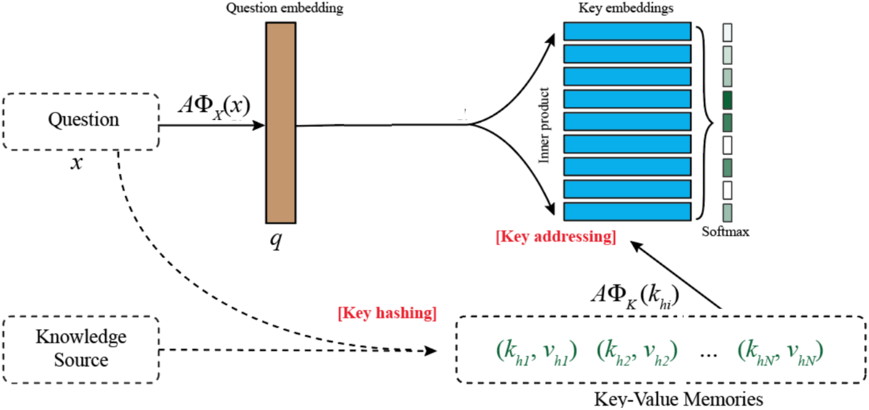
There are other ways to extend memory networks!

Key-value memory network

- Key-value memory: e.g., key is a window of words in doc, value is the centre word in window or title of doc, etc.

Key-value memory network

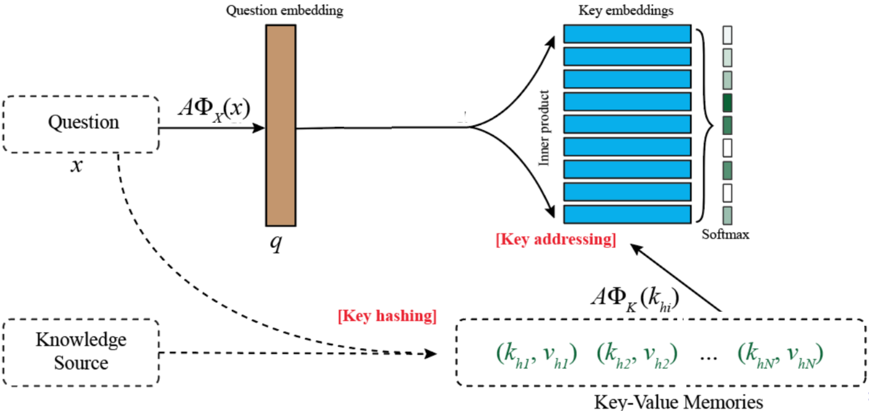
- Key-value memory: e.g., key is a window of words in doc, value is the centre word in window or title of doc, etc.
- Key hashing for subset where Key shares word with Question



Key-value memory network

- Φ_X, Φ_K: predefined sentence coding; $\mathbf{A}$: embedding
- Key addressing computes an attention depending on Question

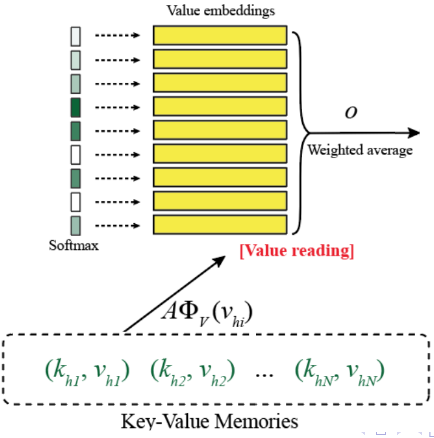
$$p_{h_i} \quad = \quad \text{Softmax}(\mathbf{A}\boldsymbol{\Phi}_X(\mathbf{x}) \cdot \mathbf{A}\boldsymbol{\Phi}_K(\mathbf{k}_{h_i}))$$



Key-value memory network

- Value reading outputs weighted sum of value embeddings

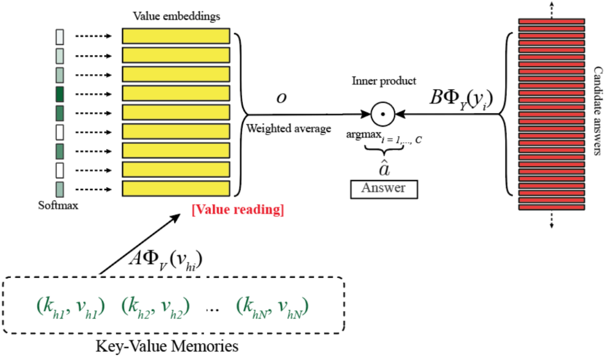
$$\mathbf{o} = \sum_i p_{h_i} \mathbf{A}\Phi_V(\mathbf{v}_{h_i})$$



Key-value memory network

- Select from candidate answers $\{\mathbf{y}_i\}$ which are all possible answer words or sentences by matching $\mathbf{o}$ (which contains answer information):

$$\hat{a} \quad = \quad \arg \max_{i=1,\ldots,C} \text{Softmax}(\mathbf{o}^T \mathbf{B} \mathbf{\Phi}_Y(\mathbf{y}_i))$$

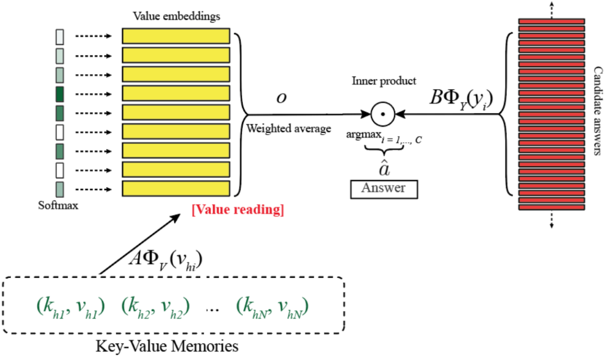


Key-value memory network

- Select from candidate answers $\{\mathbf{y}_i\}$ which are all possible answer words or sentences by matching $\mathbf{o}$ (which contains answer information):

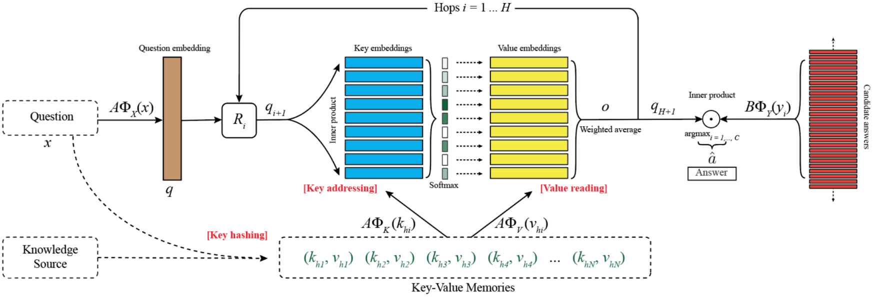
$$\hat{a} \quad = \quad \arg \max_{i=1,\ldots,C} \text{Softmax}(\mathbf{o}^T \mathbf{B}\mathbf{\Phi}_Y(\mathbf{y}_i))$$

- This is another difference from standard memory network



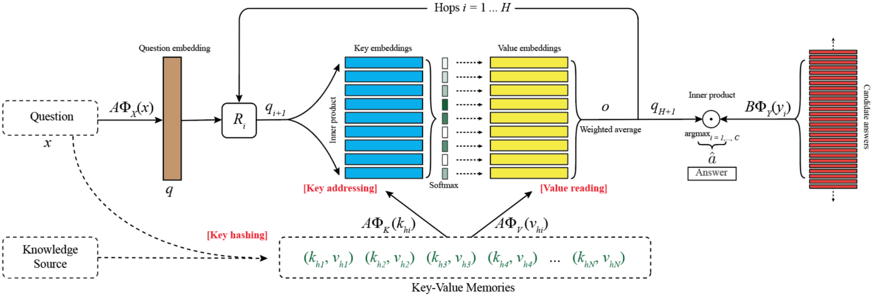
Key-value memory network

- Addressing and reading can be repeated ('Hops'), each time with updated query (question embedding) $\mathbf{q}_{i+1} = \mathbf{R}_i(\mathbf{q}_i + \mathbf{o}_i)$



Key-value memory network

- Addressing and reading can be repeated ('Hops'), each time with updated query (question embedding) $\mathbf{q}_{i+1} = \mathbf{R}_i(\mathbf{q}_i + \mathbf{o}_i)$
- Model parameters: $\mathbf{A}, \mathbf{B}, \mathbf{R}_1, \ldots, \mathbf{R}_H$
- Train: cross entropy loss between predicted and true answers



Key-value memory network

- Model is better than others on WikiQA and MOVIEQA sets

Doc: Wikipedia Article for Blade Runner (partially shown)
Blade Runner is a 1982 American neo-noir dystopian science fiction film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, Sean Young, and Edward James Olmos. The screenplay, written by Hampton Fancher and David Peoples, is a modified film adaptation of the 1968 novel "Do Androids Dream of Electric Sheep?" by Philip K. Dick. The film depicts a dystopian Los Angeles in November 2019 in which genetically engineered replicants, which are visually indistinguishable from adult humans, are manufactured by the powerful Tyrell Corporation as well as by other "mega-corporations" around the world. Their use on Earth is banned and replicants are exclusively used for dangerous, menial, or leisure work on off-world colonies. Replicants who defy the ban and return to Earth are hunted down and "retired" by special police operatives known as "Blade Runners". . . .
Questions for Blade Runner (subset)
Ridley Scott directed which films? What year was the movie Blade Runner released? Who is the writer of the film Blade Runner? Which films can be described by dystopian? Which movies was Philip K. Dick the writer of? Can you describe movie Blade Runner in a few words?

How does human read and understand a story?

- The ability to maintain and update the state of the world is a key feature of intelligent agents!
  1. ‘Mary picked up the ball.’
  2. ‘Mary went to the garden.’

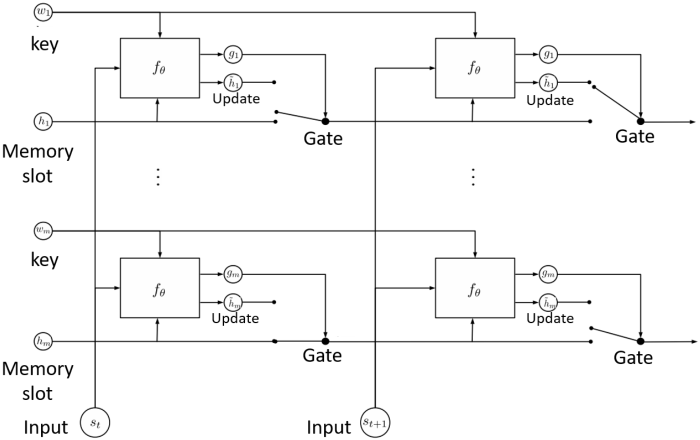
How does human read and understand a story?

- The ability to maintain and update the state of the world is a key feature of intelligent agents!
  1. 'Mary picked up the ball.'
  2. 'Mary went to the garden.'
- Explicitly maintain a set of high-level concepts or entities together with their properties, which are updated as new information is received

Figures in previous 6 slides from Miller et al., "Key-value memory networks for directly reading documents", arXiv, 2016
Figures in next 6 slides from Henaff, Weston, Szlam, Bordes, LeCun, "Tracking the world state with recurrent entity networks", ICLR, 2017

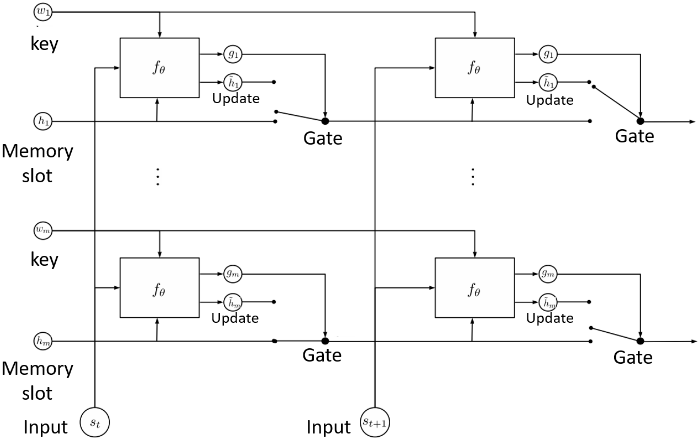
Recurrent entity network

- Multiple ($5 \sim 20$) dynamic memory cells: (key $\mathbf{w}_j$, value $\mathbf{h}_j$)'s



Recurrent entity network

- Multiple $(5 \sim 20)$ dynamic memory cells: (key $\mathbf{w}_j$, value $\mathbf{h}_j$)'s
- Novelty 1: each cell is associated with its own 'processor', a gated recurrent network that update cell value given an input



Recurrent entity network: dynamic memory

- Each cell stores state of one entity (e.g., objects, persons)
- The j^{th} network independently uses a key $\mathbf{w}_j$ and the t^{th} input sentence $\mathbf{s}_t$ to update the memory $\mathbf{h}_j$ at time step t

$$
\begin{aligned}
\mathbf{g}_{j,t} &= \sigma(\mathbf{s}_t^T \mathbf{h}_{j,t-1} + \mathbf{s}_t^T \mathbf{w}_j) \\
\tilde{\mathbf{h}}_{j,t} &= \phi(\mathbf{U}\mathbf{h}_{j,t-1} + \mathbf{V}\mathbf{w}_j + \mathbf{W}\mathbf{s}_t) \\
\mathbf{h}_{j,t} &= \mathbf{h}_{j,t-1} + \mathbf{g}_{j,t} \odot \tilde{\mathbf{h}}_{j,t} \\
\mathbf{h}_{j,t} &\leftarrow \frac{\mathbf{h}_{j,t}}{\|\mathbf{h}_{j,t}\|}
\end{aligned}
$$

Recurrent entity network: dynamic memory

- Each cell stores state of one entity (e.g., objects, persons)
- The j^{th} network independently uses a key $\mathbf{w}_j$ and the t^{th} input sentence $\mathbf{s}_t$ to update the memory $\mathbf{h}_j$ at time step t
- Keys $\{\mathbf{w}_j\}$ can be learned as parameters, or entity words in doc or candidate answers
- $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ are shared by all gated recurrent networks
- L_2 norm helps forget old memory

$$
\begin{aligned}
\mathbf{g}_{j,t} &= \sigma(\mathbf{s}_t^T \mathbf{h}_{j,t-1} + \mathbf{s}_t^T \mathbf{w}_j) \\
\tilde{\mathbf{h}}_{j,t} &= \phi(\mathbf{U}\mathbf{h}_{j,t-1} + \mathbf{V}\mathbf{w}_j + \mathbf{W}\mathbf{s}_t) \\
\mathbf{h}_{j,t} &= \mathbf{h}_{j,t-1} + \mathbf{g}_{j,t} \odot \tilde{\mathbf{h}}_{j,t} \\
\mathbf{h}_{j,t} &\leftarrow \frac{\mathbf{h}_{j,t}}{\|\mathbf{h}_{j,t}\|}
\end{aligned}
$$

Recurrent entity network: gate function

- Novelty 2: input $\mathbf{s}_t$ directly interacts with $\mathbf{w}_j$ and memory $\mathbf{h}_j$

$$\mathbf{g}_{j,t} = \sigma(\mathbf{s}_t^T \mathbf{h}_{j,t-1} + \mathbf{s}_t^T \mathbf{w}_j)$$

Recurrent entity network: gate function

- Novelty 2: input $\mathbf{s}_t$ directly interacts with $\mathbf{w}_j$ and memory $\mathbf{h}_j$
- The gate function $\mathbf{g}_{j,t}$ contains two terms
- 'Content' term $\mathbf{s}_t^T \mathbf{h}_{j,t-1}$ cause gate to open for memory slots whose content $\mathbf{h}_{j,t-1}$ matches the input
- 'Location' term $\mathbf{s}_t^T \mathbf{w}_j$ cause gate to open for memory slots whose key $\mathbf{w}_j$ matches the input

$$\mathbf{g}_{j,t} = \sigma(\mathbf{s}_t^T \mathbf{h}_{j,t-1} + \mathbf{s}_t^T \mathbf{w}_j)$$

Recurrent entity network: gate function

- Novelty 2: input $\mathbf{s}_t$ directly interacts with $\mathbf{w}_j$ and memory $\mathbf{h}_j$
- The gate function $\mathbf{g}_{j,t}$ contains two terms
- 'Content' term $\mathbf{s}_t^T \mathbf{h}_{j,t-1}$ cause gate to open for memory slots whose content $\mathbf{h}_{j,t-1}$ matches the input
- 'Location' term $\mathbf{s}_t^T \mathbf{w}_j$ cause gate to open for memory slots whose key $\mathbf{w}_j$ matches the input

$$\mathbf{g}_{j,t} = \sigma(\mathbf{s}_t^T \mathbf{h}_{j,t-1} + \mathbf{s}_t^T \mathbf{w}_j)$$

- Memories are updated only when new information relevant to their concepts are received, and remains otherwise unchanged

$$\mathbf{h}_{j,t} = \mathbf{h}_{j,t-1} + \mathbf{g}_{j,t} \odot \tilde{\mathbf{h}}_{j,t}$$

An example of how the model work

1^{st}: 'Mary picked up the ball.'; 2^{nd}: 'Mary went to the garden.'

An example of how the model work

1^{st}: 'Mary picked up the ball.'; 2^{nd}: 'Mary went to the garden.'

- Two gated networks (memories) for states of 'Mary' and 'ball'.
- Key $\mathbf{w}_1$ for entity 'Mary'; $\mathbf{w}_2$ for entity 'Ball'

An example of how the model work

1^{st}: 'Mary picked up the ball.'; 2^{nd}: 'Mary went to the garden.'

- Two gated networks (memories) for states of 'Mary' and 'ball'.
- Key $\mathbf{w}_1$ for entity 'Mary'; $\mathbf{w}_2$ for entity 'Ball'

1^{st} time step: $\mathbf{s}_1$ encoding 'Mary picked up the ball.'

- Location term $\mathbf{s}_1^T\mathbf{w}_1$ would open gates of memory 'Mary'
- Location term $\mathbf{s}_1^T\mathbf{w}_2$ would open gates of memory 'ball'.
- Updated state $\mathbf{h}_{1,1}$ of 'Mary' entity: she is carrying the ball
- Updated state $\mathbf{h}_{2,1}$ of 'ball' entity: it is carried by Mary

An example of how the model work

1^{st}: 'Mary picked up the ball.'; 2^{nd}: 'Mary went to the garden.'

- Two gated networks (memories) for states of 'Mary' and 'ball'.
- Key $\mathbf{w}_1$ for entity 'Mary'; $\mathbf{w}_2$ for entity 'Ball'

1^{st} time step: $\mathbf{s}_1$ encoding 'Mary picked up the ball.'

- Location term $\mathbf{s}_1^T \mathbf{w}_1$ would open gates of memory 'Mary'
- Location term $\mathbf{s}_1^T \mathbf{w}_2$ would open gates of memory 'ball'.
- Updated state $\mathbf{h}_{1,1}$ of 'Mary' entity: she is carrying the ball
- Updated state $\mathbf{h}_{2,1}$ of 'ball' entity: it is carried by Mary

2^{nd} time step: $\mathbf{s}_2$ encoding 'Mary went to the garden.'

- Both content term $\mathbf{s}_2^T \mathbf{h}_{1,1}$ and location term $\mathbf{s}_2^T \mathbf{w}_1$ help update 'Mary' entity: she is now in the garden
- Content term $\mathbf{s}_2^T \mathbf{h}_{2,1}$ would open gate of 'ball', even though the word 'ball' does not occur in the second sentence.
- Why: infor for 'Mary' is in 'ball' memory from 1^{st} time step,

Recurrent entity network: input and output modules

Input module:

- $\mathbf{e}_i$: i^{th} word in t^{th} input sequence; $\mathbf{f}_i$: model parameters

$$\mathbf{s}_t = \Sigma_i \mathbf{f}_i \odot \mathbf{e}_i$$

Output module:

- $\mathbf{q}$: query (question) vector; $\mathbf{R}$ and $\mathbf{H}$: model parameters
- Attention mechanism (p_j) is adopted as well

$$\begin{aligned} p_j &= \text{Softmax}(\mathbf{q}^T \mathbf{h}_j) \\ \mathbf{u} &= \Sigma_j p_j \mathbf{h}_j \\ \mathbf{y} &= \mathbf{R}\,\phi(\mathbf{q}_j + \mathbf{H}\mathbf{u}) \end{aligned}$$

Model (including previous gated networks) trained end-to-end

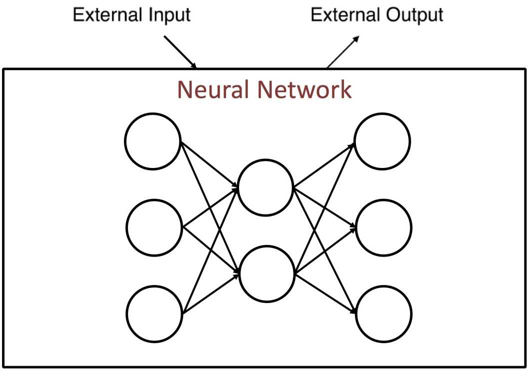
Recurrent entity network: result

- The model solved all 20 bAbI question-answering tasks

Task	NTM	D-NTM	MemN2N	DNC	DMN+	EntNet
1: 1 supporting fact	31.5	4.4	0	0	0	0
2: 2 supporting facts	54.5	27.5	0.3	0.4	0.3	0.1
3: 3 supporting facts	43.9	71.3	2.1	1.8	1.1	4.1
4: 2 argument relations	0	0	0	0	0	0
5: 3 argument relations	0.8	1.7	0.8	0.8	0.5	0.3
6: yes/no questions	17.1	1.5	0.1	0	0	0.2
7: counting	17.8	6.0	2.0	0.6	2.4	0
8: lists/sets	13.8	1.7	0.9	0.3	0.0	0.5
9: simple negation	16.4	0.6	0.3	0.2	0.0	0.1
10: indefinite knowledge	16.6	19.8	0	0.2	0	0.6
11: basic coreference	15.2	0	0.0	0	0.0	0.3
12: conjunction	8.9	6.2	0	0	0.2	0
13: compound coreference	7.4	7.5	0	0	0	1.3
14: time reasoning	24.2	17.5	0.2	0.4	0.2	0
15: basic deduction	47.0	0	0	0	0	0
16: basic induction	53.6	49.6	51.8	55.1	45.3	0.2
17: positional reasoning	25.5	1.2	18.6	12.0	4.2	0.5
18: size reasoning	2.2	0.2	5.3	0.8	2.1	0.3
19: path finding	4.3	39.5	2.3	3.9	0.0	2.3
20: agent's motivation	1.5	0	0	0	0	0
Failed Tasks ($>5\%$ error):	16	9	3	2	1	**0**
Mean Error:	20.1	12.8	4.2	3.8	2.8	**0.5**

Before neural Turing machine

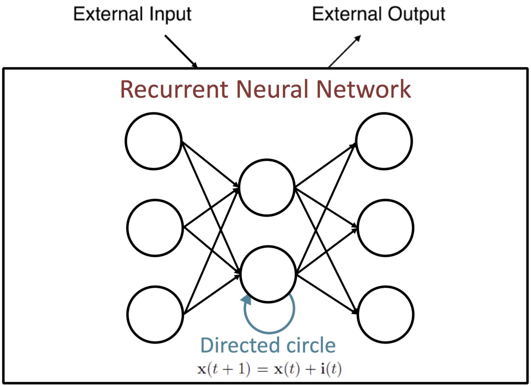
- Feedforward networks process input, then generate output



Figures here and in next 4 slides from https://jamiekang.github.io/

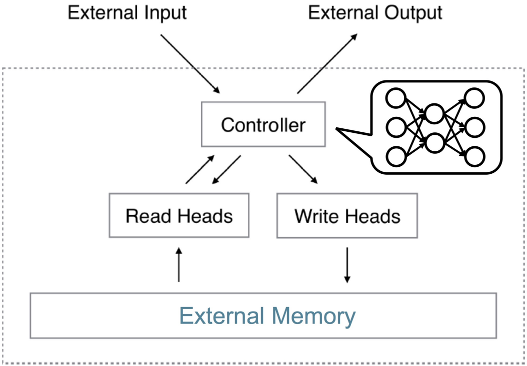
Before neural Turing machine

- RNN did similarly, with recurrent connections inside



Neural Turing machine

- Neural Turing machine uses controller (either feed-forward or RNN) to read from and write to external memory, then generate output from the read result.



Neural Turing machine: reading and writing

- Memory $\mathbf{M}$ is $N \times M$: N locations, M elements per location
- $\mathbf{w}_t = (w_t(0), \ldots, w_t(N-1))$: reading weight, from read head
- $\mathbf{r}_t$: weighted row vectors in memory; returned by read head

$$\sum_i w_t(i) = 1, \quad 0 \leq w_t(i) \leq 1, \forall i$$

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

Neural Turing machine: reading and writing

- Memory $\mathbf{M}$ is $N \times M$: N locations, M elements per location
- $\mathbf{w}_t = (w_t(0), \ldots, w_t(N-1))$: reading weight, from read head
- $\mathbf{r}_t$: weighted row vectors in memory; returned by read head

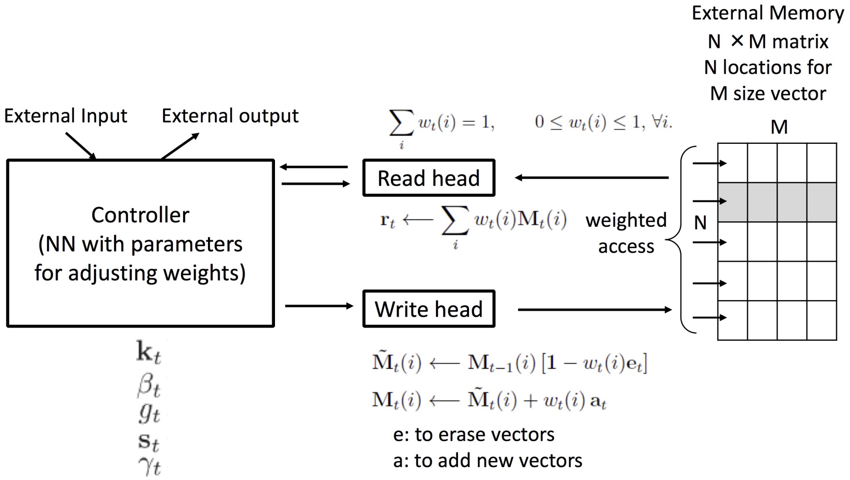
$$\sum_i w_t(i) = 1, \quad 0 \leq w_t(i) \leq 1, \; \forall i$$

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

- From write head: weight $\mathbf{w}_t$, erase vector $\mathbf{e}_t$, add vector $\mathbf{a}_t$
- Fine-grained control over elements in each memory location

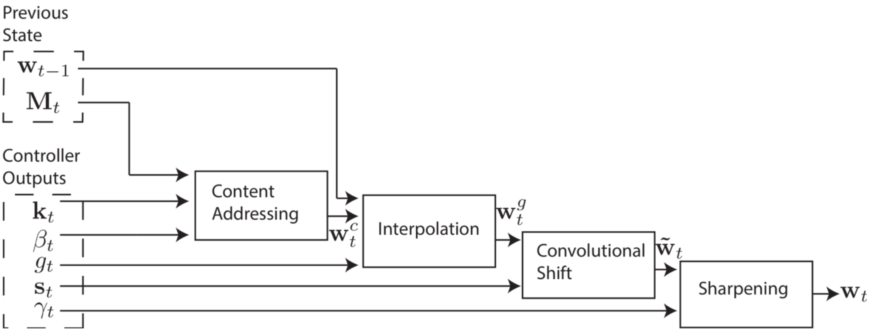
$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i) \left[\mathbf{1} - w_t(i)\mathbf{e}_t\right]$$
$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\, \mathbf{a}_t$$

Neural Turing machine: reading and writing



How to generate read weight and write information?

- Content-based addressing: attended to locations where memory content is similar to values $\mathbf{k}_t$ emitted by controller
- Location-based: by rotationally shifting elements of a weight
- Can be combined, also with interpolation and sharpening



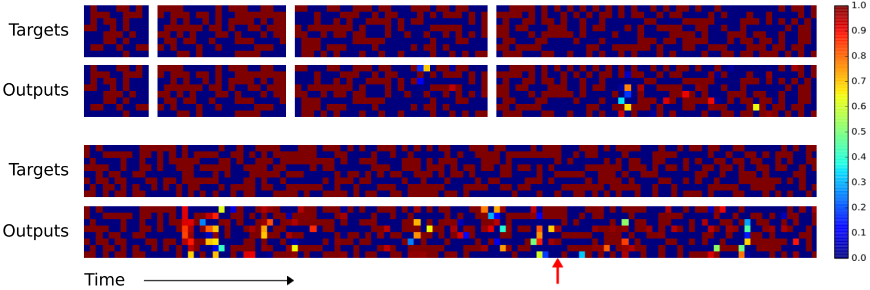
Figures here and in next slide from Graves, Wayne, Danihelka, "Neural Turing machines", arXiv, 2014

Neural Turing machine: result

- Model parameters are mainly in Controller; trained end-to-end
- Trained to copy sequence (length up to 20) of 8-bit random vectors; tested on sequences of length 10, 20, 30, 50, 120

Neural Turing machine: result

- Model parameters are mainly in Controller; trained end-to-end
- Trained to copy sequence (length up to 20) of 8-bit random vectors; tested on sequences of length 10, 20, 30, 50, 120
- NTM can copy longer sequences (below); while LSTM not
- NTM can infer simple algorithms like copying, sorting, etc.



Summary

- Memory networks often contain external memory module
- Memory networks can be trained end-to-end
- Dynamic memory networks consider sentence order
- Key-value network is more flexible than memory networks
- Recurrent entity networks can directly update entities' states
- Neural Turing machine can infer simple algorithms
- Focus is how to generate and use memory information

Further reading:

- Graves et al., 'Hybrid computing using a neural network with dynamic external memory', Nature, 2016
- Chandar et al., 'Hierarchical memory networks', ICLR, 2017