Memory Networks

- Structure

I: (input feature map) – converts the incoming input to the internal feature representation.

G: (generalization) – updates old memories given the new input. We call this generalization as there is an opportunity for the network to compress and generalize its memories at this stage for some intended future use.

O: (output feature map) – produces a new output (in the feature representation space), given the new input and the current memory state.

R: (response) – converts the output into the response format desired. For example, a textual response or an action.

Memory Networks

- Structure

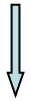
1. Convert x to an internal feature representation $I(x)$.
2. Update memories $\mathbf{m}_i$ given the new input: $\mathbf{m}_i = G(\mathbf{m}_i, I(x), \mathbf{m}), \ \forall i.$
3. Compute output features o given the new input and the memory: $o = O(I(x), \mathbf{m})$.
4. Finally, decode output features o to give the final response: $r = R(o)$.

Memory Networks

- Example

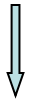
Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

Input x : Where is the milk now?



I(x): feature embedding

m_{o1}="Jone left the milk"
m_{o2}="Joe travelled to the office"



r = "office"

$$s(x, y) = \Phi_x(x)^\top U^\top U \Phi_y(y).$$

$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

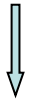
$$r = \text{argmax}_{w \in W} s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

$$\sum_{\bar{f} \neq f_1} \max(0, \gamma - s_O(x, f_1) + s_O(x, \bar{f})) +$$

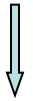
$$\sum_{\bar{f}' \neq f_2} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], f_2]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}'])) +$$

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

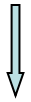
Input x : Where is the milk now?



I(x): feature embedding



m_{o1}="Jone left the milk"
m_{o2}="Joe travelled to the office"



r = "office"

$$s(x, y) = \Phi_x(x)^\top U^\top U \Phi_y(y).$$

$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

$$r = \text{argmax}_{w \in W} s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

In the case of employing an RNN for the R component of our MemNN (instead of using a single word response as above) we replace the last term with the standard log likelihood used in a language modeling task, where the RNN is fed the sequence $[x, o_1, o_2, r]$. At test time we output its prediction r given $[x, o_1, o_2]$. In contrast the absolute simplest model, that of using $k = 1$ and outputting the located memory m_{o_1} as response r, would only use the first term to train.

Memory Networks

- Experiment1:large scale QA

1. consists of 14M statements, stored as (subject, relation, object) triples
2. training combines pseudo-labeled QA pairs made of aquestion and an associated triple, and 35M pairs of paraphrased questions from WikiAnswers like “Who wrote the Winnie the Pooh books?” and “Who is poohs creator?”

Table 1: Results on the large-scale QA task of (Fader et al., 2013).

Method	F1
(Fader et al., 2013)	0.54
(Bordes et al., 2014b)	0.73
MemNN (embedding only)	0.72
MemNN (with BoW features)	0.82

Memory Networks

- Experiment1:large scale QA

1. consists of 14M statements, stored as (subject, relation, object) triples
2. training combines pseudo-labeled QA pairs made of aquestion and an associated triple, and 35M pairs of paraphrased questions from WikiAnswers like "Who wrote the Winnie the Pooh books?" and "Who is poohs creator?"

Table 2: Memory hashing results on the large-scale QA task of (Fader et al., 2013).

Method	Embedding F1	Embedding + BoW F1	Candidates (speedup)
MemNN (no hashing)	0.72	0.82	14M (0x)
MemNN (word hash)	0.63	0.68	13k (1000x)
MemNN (cluster hash)	0.71	0.80	177k (80x)

Memory Networks

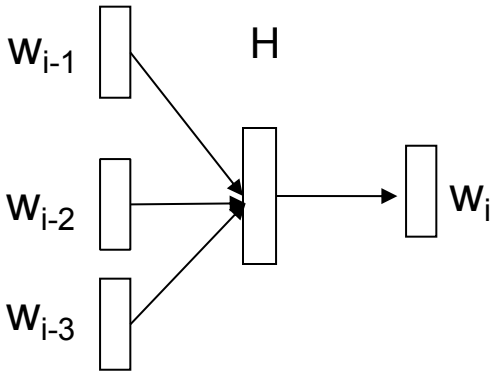
- Experiment1:Simulated world QA

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

Table 3: Test accuracy on the simulation QA task.

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

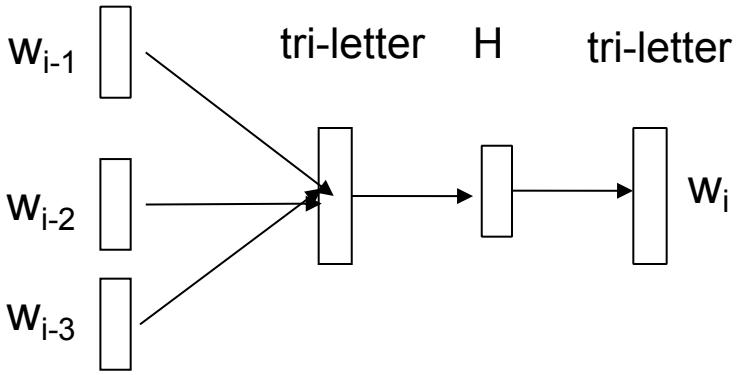
large scale lm



$$p(w_i \mid w_{i-1}w_{i-2}w_{i-3}) = f(w_{i-1}w_{i-2}w_{i-3}) \bullet w_i$$

$$o = \max(p(w_i))$$

Character lm



$$p(w_i \mid w_{i-1}, w_{i-2}, w_{i-3}) = f(hash(w_{i-1}, w_{i-2}, w_{i-3})) \bullet hash(w_i)$$