Memory network

투빅스 8기 최수정

Contents

Unit 01 | Memory networks

배경

대부분의 모델들은 input이 매우 길어질 때 long-term memory component 를 잡아내고 inference와 매끄럽게 결합하는 것을 잘 못함

다음 word를 예측하는 RNN과 같은 모델들도 memory능력이 매우 작기 때문에 일을 수행하기 어려움

이런 문제를 해결하기 위해 Memory network를 제시하게 됨

Network의 구조

메모리 i개로 indexing된 메모리 m과 학습되는 I G O R로 이뤄짐

- I : input feature map. input을 feature representation으로 바꿔준다
- G : generalization. 새로운 input이 들어오면 이를 토대로 이전 memory를 update해준다.
- O : output feature map. 새로운 input과 현재의 memory를 토대로 새로운 output(feature representation)을 만든다
- R : response. output(feature representation)을 원하는 형태로 변환시킨 뒤 response를 한다.

Network의 구조 -모델의 작동방식

Input x 가 들어왔을 때

1. X를 feature representation I(x) 으로 전환한다.

2. 주어진 Input을 토대로 memory(i)를 update. $m_i = G(m_i, I(x), \boldsymbol{m}), \forall i$

3. Input과 memory로 output feature O를 계산 $o = O(I(x), \boldsymbol{m})$

4. 마지막으로 최종 response로 output feature O를 디코딩한다. $r = R(o)$

Network의 구조

I component : I는 pre-processing과정 (parsing, entity resolution)의 역할을 하거나 encoding으로 사용

G component : G의 가장 간단한 역할은 I(x)를 메모리 슬롯에 저장한다는 것이다. 원하는 memory의 크기만큼 update시켜줄 수 있다.

O component : memory를 읽고 어떤 memory가 관련되어 있는지를 계산한다.

R component : 이를 통해 나온 O를 토대로 최종 response를 한다 R은 output O에 대한 RNN decoder일 수 있다.

A Mem NN implementation for text

MEMORY NETWORK + NEURAL NETWORK = MEM NN

A Mem NN implementation for text

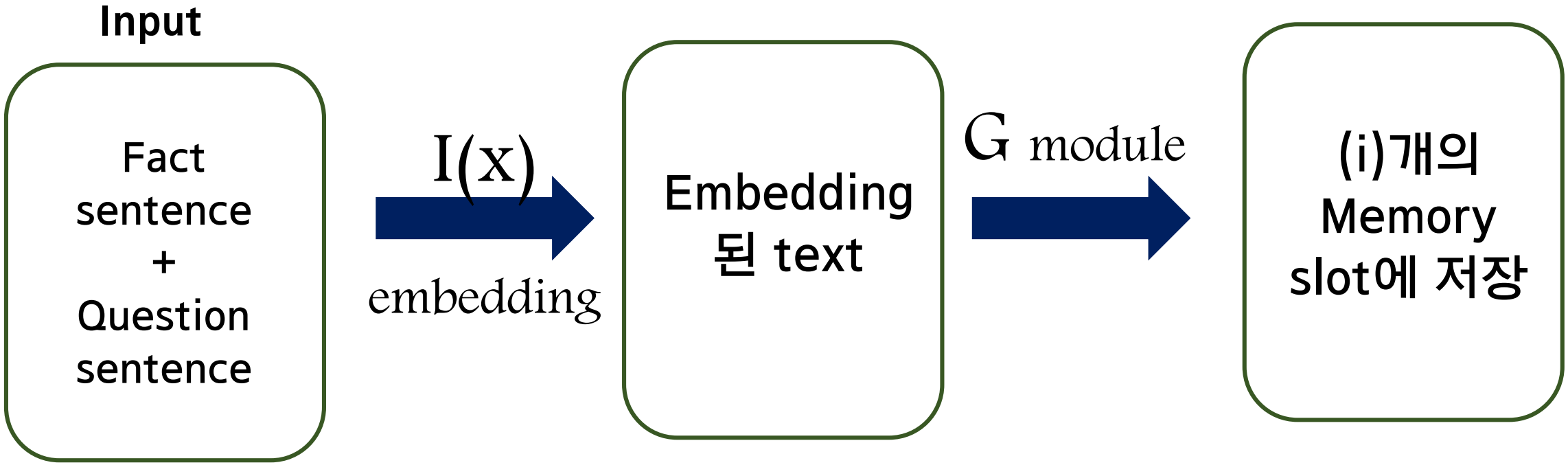
Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

Figure 3: An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 5.2 and had never seen many of these words before, e.g., Bilbo, Frodo and Gollum.

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

A Mem NN implementation for text

-Data들은 Story가 있다! -> 따라서 이 story를 바탕으로 답변이 나옴
-"시간 순서 " 로 data들이 들어오기 때문에 주변 단어들과 어느 정도 연관이 있다



A Mem NN implementation for text

Storing information into the memory

First, we store all the sentences in the memory m:

Memory slot m_i	Sentence
1	Joe went to the kitchen.
2	Fred went to the kitchen.
3	Joe picked up the milk.
4	Joe traveled to the office.
5	Joe left the milk.
6	Joe went to the bathroom.

A Mem NN implementation for text

-Inference의 core는 O, R에 있다 (O :어떤 메모리와 관련되어 있는지를 계산,
R: output feature O에 대한 decoder)

O는 input x에 대해 유의미한 k개의 메모리를 선정하여 output feature을 만든다.
(여기선 k를 2로 선정)

즉 여러 문장과 질문이 input으로 들어온다면, 그 질문에 도움될 만한 2개의 메모리,
여기서는 한문장이 한 메모리 슬롯에 들어가 있으므로 2개의 문장을 고르는 것!

A Mem NN implementation for text

When K=1 Q와 가장 관련 있는 memory(supporting memory) o1

$$o_1 = O_1(x, \mathbf{m}) = argmax s_O(x, m_i), \forall i$$

A Mem NN implementation for text

When K=1 Q와 가장 관련 있는 memory(supporting memory) o1

$$o_1 = O_1(x, \mathbf{m}) = argmax s_O(x, m_i), \forall i$$

Input sentence x와 m(i) 쌍의
Match점수를 매기는 score함수

A Mem NN implementation for text

When K=2 Q와 2번째로 관련 있는 memory o2

$$o_2 = O_2(x, m) = argmaxs_O([x, m_{o_1}], m_i), \forall i \ \vdots$$

A Mem NN implementation for text

When K=2 Q와 2번째로 관련 있는 memory o2

$$o_2 = O_2(x, m) = argmax s_O([[x, m_{o_1}]], m_i), \forall i$$

다음 관련 있는 후보 memory는 이전 메모리(m(o1))와 input(x)의 list와의 match로써 S(o)함수 적용해 o2를 얻는다.

$s_O(x, m_i) + s_O(m_{o_1}, m_i)$을 최대화 하는 값을 찾는것과 같다.

A Mem NN implementation for text

R 로 들어가는 최종 output: $[x, m_{o1}, m_{o2}]$

-이제 최종 R이 response r 을 만든다. (선택 한 메모리 안에서 Answer로 내놓을 단어를 찾아야 한다!)

-이 논문에서는 single word 를 textual response를 limit함
(ex. Q:John은 어디에 있니? A:침실)

A Mem NN implementation for text

$[x, m_{o1}, m_{o2}]$ 의 조합을 평가하기 위해 ranking한다.

$$r = argmax_{w \in W} s_R([x, m_{o1}, m_{o2}], w)$$

Match를 점수매기는 score함수 Dictionary에 있는 모든 단어

결국, 앞에 했던 So와 비슷하게 Dic(W)에 있는 모든 단어 중 List $[x, m_{o1}, m_{o2}]$ 에 가장 적절하게 매칭될 만한 단어를 찾는 것!

A Mem NN implementation for text

지금까지 한 것을 예제에 대입해 이해해보기

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 question x='Where is the milk now?'

Memory slot m_i	Input	Sentence
1		Joe went to the kitchen.
2		Fred went to the kitchen.
3		Joe picked up the milk.
4		Joe traveled to the office.
5		Joe left the milk.
6		Joe went to the bathroom.

-O는 이 input x와 memory들간의 점수를 매기고 x와 가장 관련 있는 메모리를 m(o1)로 내보낸다.
(ex. m(o1) :'Joe left the milk')

- $[x, m_{o_1}]$ 에 가장 관련 있는 m(o2)를 찾는다. (ex. m(o2):'Joe travelled to the office'(우유를 방치하기 전 가장 최근에 간 장소))

- R은 $s_R([x, m_{o_1}, m_{o_2}], w)$ 에 따라 'office'를 출력하게 된다.

A Mem NN implementation for text

지금까지 한 것을 예제에 대입해 이해해보기

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 question x='Where is the milk now?'

Memory slot m_i	Input	Sentence
1		Joe went to the kitchen.
2		Fred went to the kitchen.
3		Joe picked up the milk.
4		Joe traveled to the office.
5		Joe left the milk.
6		Joe went to the bathroom.

$$o = [q, m_{o1}, m_{o2}] = [\text{"where is the milk now"}, \text{" Joe left the milk."}, \text{" Joe travelled to the office."}]$$

A Mem NN implementation for text

Encoding the input

We use bags of words to represent our input text. First, we start with a vocabulary of a size $|W|$.

To encode the query "where is the milk now" with bags of words:

Vocabulary	...	is	Joe	left	milk	now	office	the	to	travelled	where	...
where is the milk now	...	1	0	0	1	1	0	1	0	0	1	...

$$\text{"where is the milk now"} = (\ldots,1,0,0,1,1,0,1,0,0,1,\ldots)$$

For better performance, we use 3 different set of bags to encode q, m_{o_1} and m_{o_2} separately, i.e., we encode "Joe" in q as "Joe_1" and the same word in m_{o_1} as "Joe_2":

Compute the scoring function

To encode $[q, m_{o_1}, m_{o_2}]$ above, we use:

...	Joe_1	milk_1	...	Joe_2	milk_2	...	Joe_3	milk_3	...	
	0	1		1	1		1	0		

Hence, each sentence is converted to a bag of words of size $3|W|$.

A Mem NN implementation for text

Compute the scoring function

Compute the scoring function

We use word embeddings U to convert a sentence with a bag of words with size $3|W|$ into a dense word embedding with a size n. We evaluate both score functions s_o and s_r with:

$$s_o(x,y) = \Phi_x(x)^T U_o{}^T U_o \Phi_y(y)$$

$$s_r(x,y) = \Phi_x(x)^T U_r{}^T U_r \Phi_y(y)$$

which embedding U_o and U_r are trained with a margin loss function, and $\Phi(m_i)$ converts the sentence m_i into a bag of words.

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

U 는 n X D의 embedding matrix.
n은 임베딩 dimension, D는 feature의 수를 의미

Φ_x, Φ_y 는 차원 feature space로 매핑된 text의 embedding feature representation이다.
(가장 간단한 매핑은 bag of words이다.)

$$D = |W|$$

$D = 3|W|$ 논문에서는 dictionary내의 모든 word가 3개의 representation을 갖는다는 의미로 사용

A Mem NN implementation for text

Compute the scoring function

Compute the scoring function

We use word embeddings U to convert a sentence with a bag of words with size $3|W|$ into a dense word embedding with a size n. We evaluate both score functions s_o and s_r with:

$$s_o(x,y) = \Phi_x(x)^T U_o{}^T U_o \Phi_y(y)$$

$$s_r(x,y) = \Phi_x(x)^T U_r{}^T U_r \Phi_y(y)$$

which embedding U_o and U_r are trained with a margin loss function, and $\Phi(m_i)$ converts the sentence m_i into a bag of words.

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

U 는 n X D의 embedding matrix.
n은 임베딩 dimension, D는 feature의 수를 의미

Φ_x, Φ_y 는 차원 feature space로 매핑된 text의 embedding feature representation이다.
(가장 간단한 매핑은 bag of words이다.)

$$D = |W|$$

결국 x와 y의 내적이니
내적으로 유사도를 구하는 것..

$D = 3|W|$ 논문에서는 dictionary내의 모든 word가 3개의 representation을 갖는다는 의미로 사용

A Mem NN implementation for text *Training*

Fully supervised방식!

Input에 대한 desired response가 있고 input에 관련되어 있는 메모리 sentence 역시 **Labeling**

훈련 데이터 전체에 대한 손실값
-margin ranking loss and SGD

X와 그에 대한 true response r, supporting sentence m(o1), m(o2)을 알고 있을 때!

$$\sum_{\bar{f} \neq m_{o_1}} \max(0, \gamma - s_o(x, m_{o_1}) + s_o(x, \bar{f}))+$$

$$\sum_{\bar{f} \neq m_{o_2}} \max(0, \gamma - s_o([x, m_{o_1}], m_{o_2}) + s_o([x, m_{o_1}], \bar{f}'))+$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_r([x, m_{o_1}, m_{o_2}], r) + s_r([x, m_{o_1}, m_{o_2}], \bar{r})$$

$\bar{f}, \bar{f}', \bar{r}$: 잘못 된 답

맞는 답과 잘못된 답의 점수 차이를 gamma이상 차이나게 하는 것이 목표!
->맞는 답 점수가 잘못된 답의 점수보다 gamma이상 차이나면 손실값이 0
->결국, 맞는 답의 match값을 더 크게 키우려고 하는 것

A Mem NN implementation for text *Training*

Fully supervised방식!

Input에 대한 desired response가 있고 input에 관련되어 있는 메모리 sentence 역시 **Labeling**

훈련 데이터 전체에 대한 손실값
-margin ranking loss and SGD

$$\sum_{\bar{f} \neq m_{o_1}} \max(0, \gamma - s_o(x, m_{o_1}) + s_o(x, \bar{f}))+$$

이 부분이 (-)가 되는게 좋은 거

$$\sum_{\bar{f} \neq m_{o_2}} \max(0, \gamma - s_o([x, m_{o_1}], m_{o_2}) + s_o([x, m_{o_1}], \bar{f}'))+$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_r([x, m_{o_1}, m_{o_2}], r) + s_r([x, m_{o_1}, m_{o_2}], \bar{r})$$

X와 그에 대한 true response r, supporting sentence m(o1), m(o2)을 알고 있을 때!

$\bar{f}, \bar{f}', \bar{r}$: 잘못 된 답

맞는 답과 잘못된 답의 점수 차이를 gamma이상 차이나게 하는 것이 목표!
->맞는 답 점수가 잘못된 답의 점수보다 gamma이상 차이나면 손실값이 0
->결국, 맞는 답의 match값을 더 크게 키우려고 하는 것

A Mem NN implementation for text *Training*

Fully supervised방식!

Input에 대한 desired response가 있고 input에 관련되어 있는 메모리 sentence 역시 **Labeling**

훈련 데이터 전체에 대한 손실값
-margin ranking loss and SGD

X와 그에 대한 true response r, supporting sentence m(o1), m(o2)을 알고 있을 때!

$$\sum_{\bar{f} \neq m_{o_1}} \max(0, \gamma - s_o(x, m_{o_1}) + s_o(x, \bar{f}))+$$

$\bar{f}, \bar{f}', \bar{r}$: 잘못 된 답

$$\sum_{\bar{f} \neq m_{o_2}} \max(0, \gamma - s_o([x, m_{o_1}], m_{o_2}) + s_o([x, m_{o_1}], \bar{f}'))+$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_r([x, m_{o_1}, m_{o_2}], r) + s_r([x, m_{o_1}, m_{o_2}], \bar{r})$$

-SGD를 사용해서 모든 training set 에 대해 $\bar{f}, \bar{f}', \bar{r}$ 을 계산하는게 아니라 sample로써 계산

-또한 So, Sr 의 parameter인 각각의 U (embedding matrix)는 GD를 이용해 학습한다.

A Mem NN implementation for text

1. Input이 sentence level이 아닌 word level일 때
->뭐가 fact input이고 Q인지 구분 X
-> segmentation함수로 seq를 학습해 classify를 함

2. 메모리가 매우 매우 클 때
->input I(x)를 hassing 해 bucket에 나누어 메모리의 score 매김
->또는 워드벡터들을K-means로 클러스터링 후 K개의 bucket으로 분류 후 처리

A Mem NN implementation for text *Result*

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)

Table 3: Test accuracy on the simulation QA task.

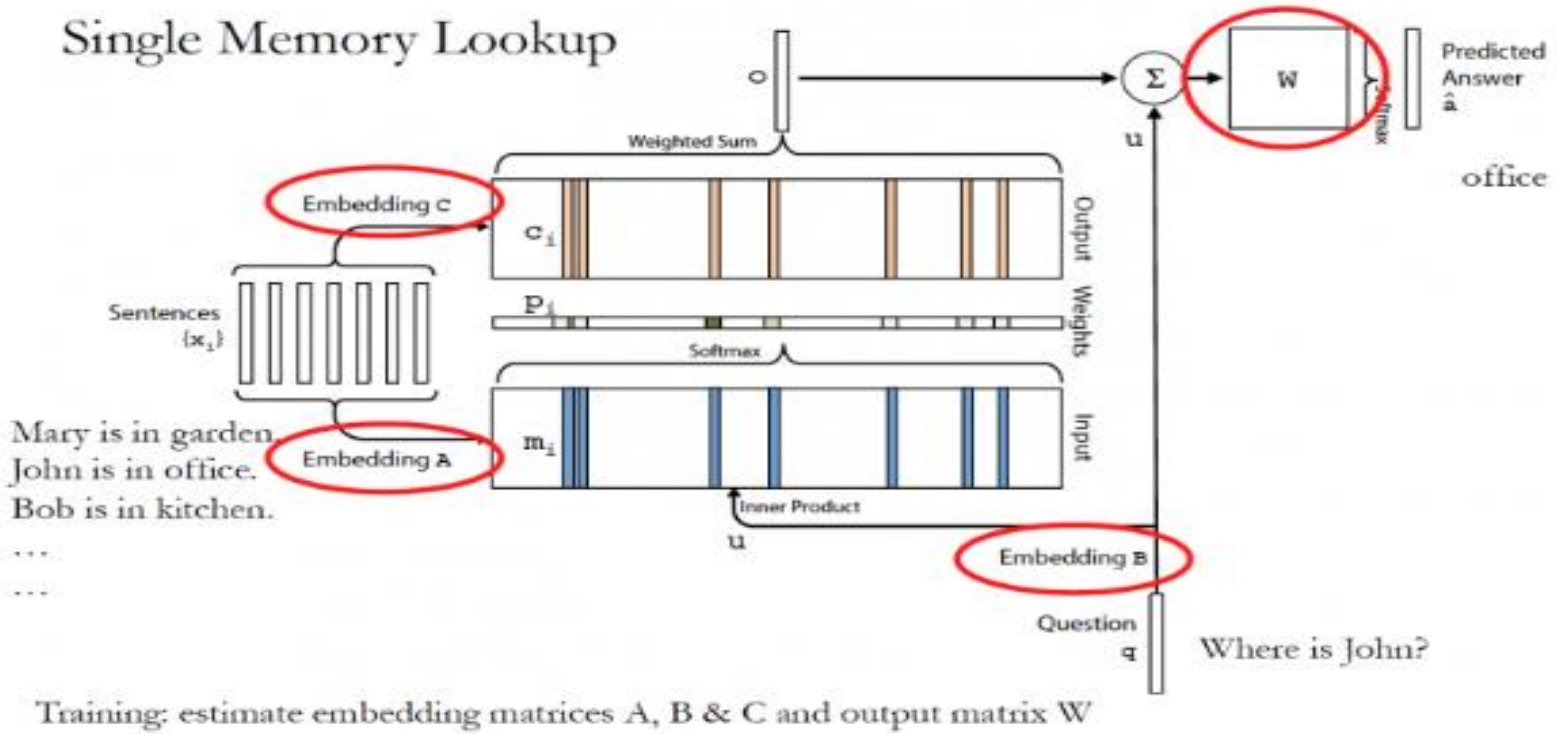
	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

A Mem N2N – End to End Memory NN

그 전 Mem NN은 스토리 선택 모듈을 학습시키기 위해 질의에 대한
적절한 스토리 정보를 사람이 추가 작성

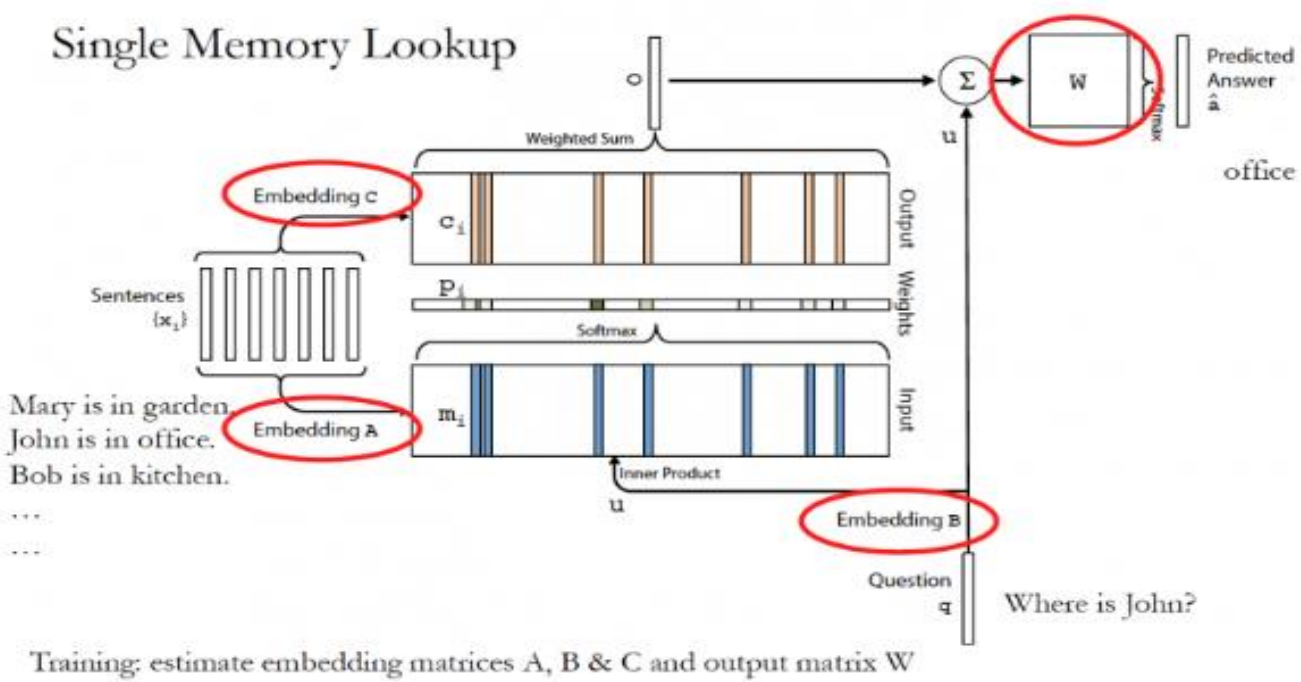
But, End to End는 훈련과정에서 지도학습이 필요 X

A Mem N2N – End to End Memory NN



Training: estimate embedding matrices A, B & C and output matrix W

A Mem N2N – End to End Memory NN



Training: estimate embedding matrices A, B & C and output matrix W

1.Input : contex문장들 (sentence {x(i)})와 질문 q
2. T(c)개의 단어가 포함된 문장 sentence I 를

$x_i = [x_{i1}, x_{i2}, \cdots, x_{iT_c}]$ 로 표현 후,

Bow방식으로 인코딩 후 vector로 바꿔준다.
=> 단어사전 index
3. 이렇게 구성된 여러 문장들의 집합을 context라 함.

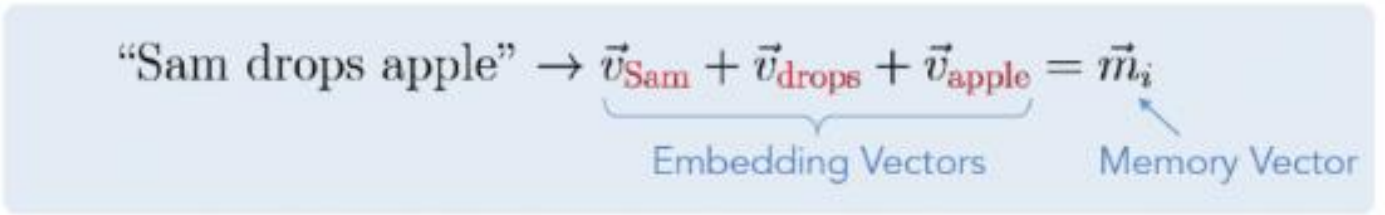
$$Context = [x_1, x_2, \cdots, x_n] \quad (n \times T_c)$$

4.T(q)개의 단어가 포함된 질문 question q도 마찬가지로 Bow방식으로 인코딩

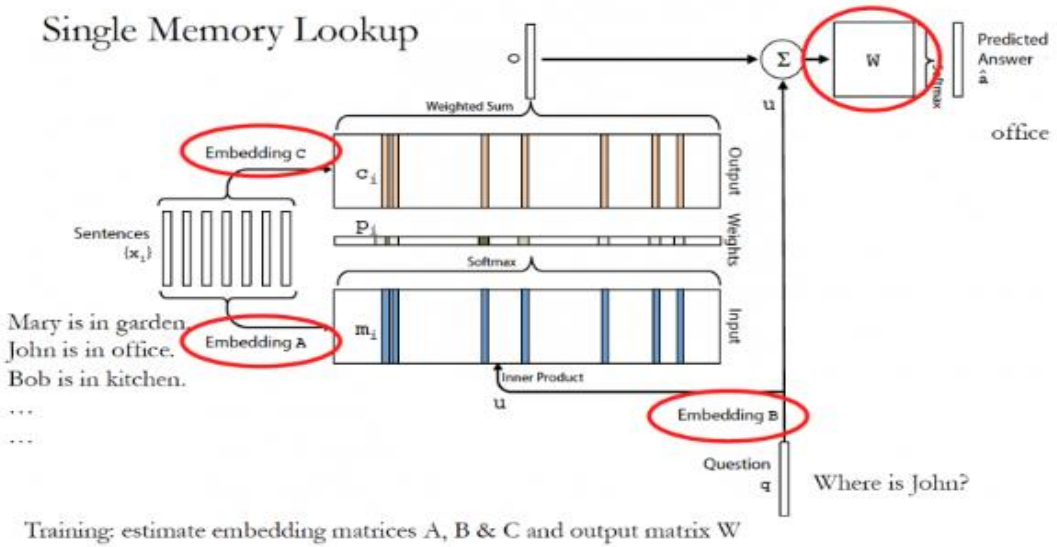
A Mem N2N – End to End Memory NN

-하나의 문장 x(i) 의 각각의 단어에 Embedding matrix A를 곱하고
-각각의 단어를 Embedding Vectors로 변환하고
-이를 모두 더하여 메모리 벡터(Memory Vector) m(i)를 구한 후,
-총 n개의 메모리가 만들어짐.

$$m_i = \sum_j A x_{ij} \tag{1}$$



A Mem N2N – End to End Memory NN



1-3. 질문 q도 같은 방식으로 Embedding matrix B를 곱하여 각각의 단어를 Embedding Vectors u로 변환한다.

$$u = \sum_j Bq_j \tag{2}$$

1-4. 1-2와 같이 Context 문장들의 각각의 단어에 다시 Embedding matrix C를 곱하여 각각의 단어를 Embedding Vectors c_i로 변환한다.

$$c_i = \sum_j Cx_{ij} \tag{3}$$

A Mem N2N – End to End Memory NN

이후 Context와 Question의 유사성(match)를 구하기 위해
추출한 Q에 대한 embedding vector u (1 x d) 와
Context 문장들로부터 만든 메모리 m (i x d) 에 Softmax!
이러한 결과로 input에 대한 확률을 도출 해낼 수 있다.

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

첫 번째 ram에서는 q와 문장들이 얼마나 잘 일치하는지를 p(i)로 나타낸
다 (i번째 문장과 q와 얼마나 일치하는지 나타내주는 정수)

A Mem N2N – End to End Memory NN

1-6. Attention 정도를 나타내는 p_i와 Context 문장로부터 만든 Embedding Vector c_i를 곱한뒤 모두 더해서 output o를 구한다.

$$o = \sum_i p_i c_i \tag{5}$$

2번째 Ram에서는 response vector o를 구하기 위해 가중 평균합을 해준다.
⇒모든 문장들이 동일하게 반영되는게 아니라 적절한 정보를 가지고 있는 문장이 크게 반영됨.

1-7. 마지막으로, output o와 질문으로부터 추출한 Embedding Vector u에 가중치값 W를 곱하여 더한뒤에 Softmax Function을 적용하여 답$\hat{a}$을 추론한다.

$$\hat{a} = Softmax(W(o+u)) \tag{6}$$

최종적으로 이들을 바탕으로 답변 선택!

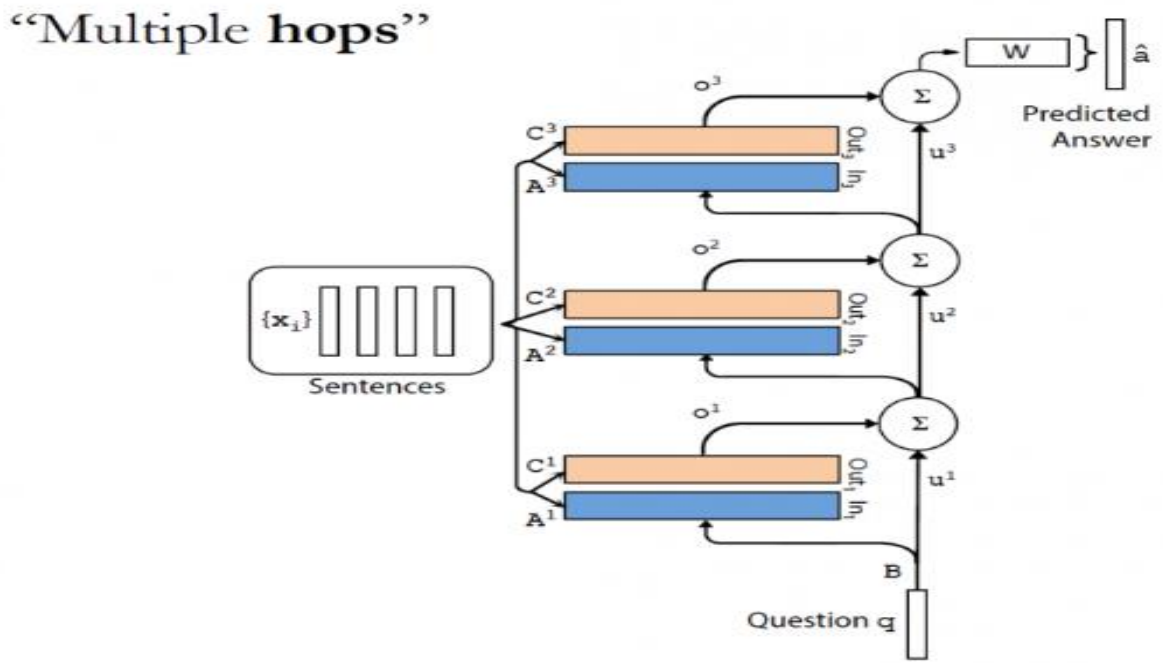
A Mem N2N – End to End Memory NN *Weight Updating*

Loss Function은 standard cross-entropy loss를
사용하여 예측치 a와 정답인 true값 a간의 오차를 최소화 시켜준다.

업데이트 되는 건 embedding matrix A B C 와 W

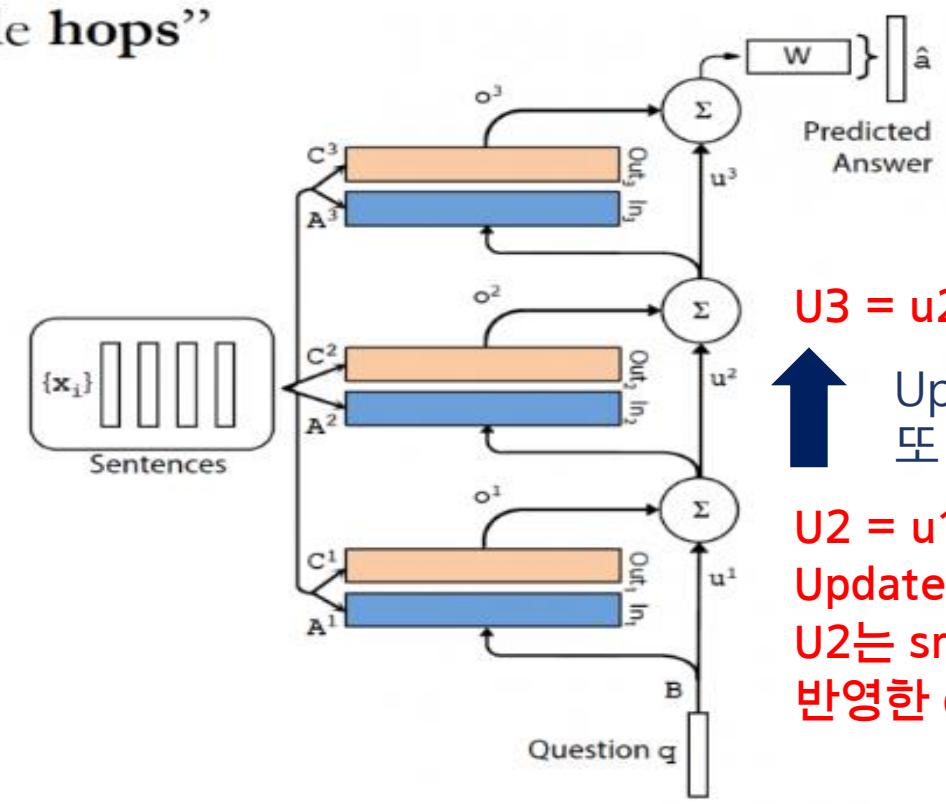
A Mem N2N – End to End Memory NN *Multiple layers*

만약 하나의 문장만이 아니라 여러 문장을 가지고 추론해야 할 때는?
위와 같은 단계를 1hop이라 하고 multiple layers K hops까지 확장시킨다.



A Mem N2N – End to End Memory NN

Multiple layers



U3 = u2+o2

Update 된 q(u2) 가지고
또 다시 o2를 구한다.

U2 = u1+o1
Update된 q!
U2는 srory를
반영한 q가 된다.

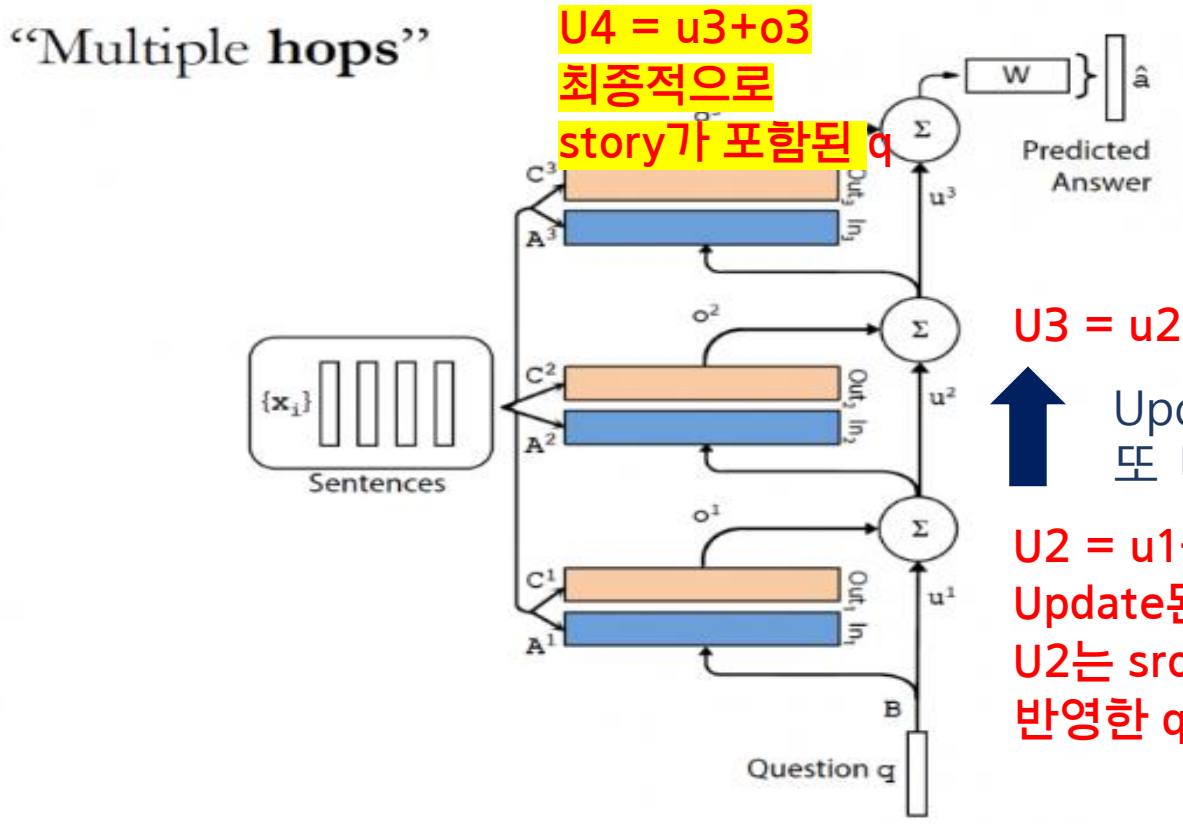
=>이 과정을 여러번

첫번째, k번째 layer에서 나온 output으로 나온 o^k과 input u^k 는 합쳐져서 새로운 input u^{k+1} 가 되어서 k + 1 layer로 들어가게 된다.

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

A Mem N2N – End to End Memory NN

Multiple layers



U3 = u2+o2

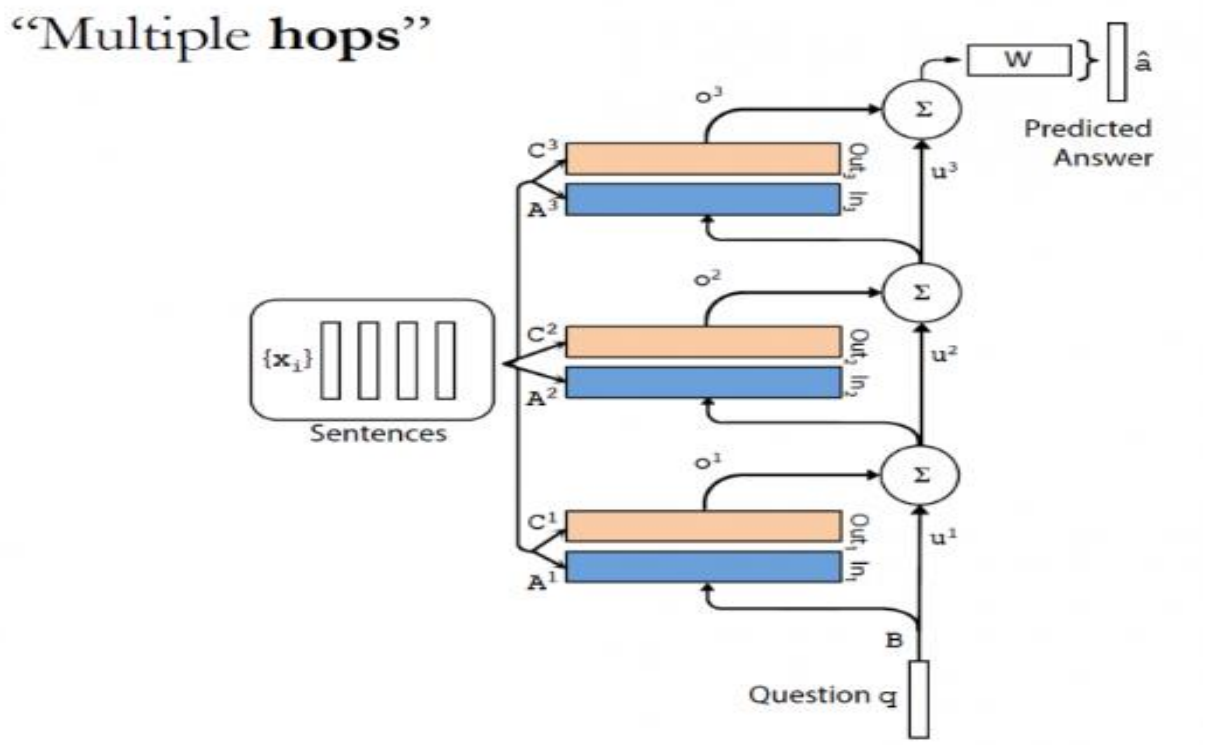
Update 된 q(u2) 가지고
또 다시 o2를 구한다.

U2 = u1+o1
Update된 q!
U2는 srory를
반영한 q가 된다.

- Network의 마지막 부분에서만 W를 곱해서 Softmax 로 출력한다.

$$\hat{a} = Softmax(Wu^{K+1})$$

A Mem N2N – End to End Memory NN



Multiple layers

각 layer마다 input에 embed로 사용된 embedding matrices A(k)와 C(k) 존재
=>쉽게 트레이닝하고 parameter개수를 줄이기 위해 제약이 존재

두 가지 가중치 버전

1. Adjacent: k_{th} output layer embedding matrix가 다음 input layer의 embedding matrix가 된다. 예를 들면, $A^{k+1} = C^k$. 또한, 두 가지 제약 조건을 추가했는데, (a) answer prediction matrix가 최종 output embedding과 같고 ($W^T = C^K$), (b) question embedding 과 첫번째 layer의 input embedding과 같게 했다($B = A^1$).

2. Layer-wise (RNN - like): Input 과 Output embedding들이 layer마다 다 같다. 예를 들면, $A^1 = A^2 = \cdots = A^K$ 과 $C^1 = C^2 = \cdots = C^K$ 같은 것들. 또한, hops간 u를 업데이트하기위한 linear mapping H 를 추가하는 것이 도움이 된다는 것을 알아냈다. $u^{k+1} = Hu^k + o^k$.

A Mem N2N – End to End Memory NN

- 예시1:

> Sam walks into the kitchen.
>
> Sam picks up an apple.
>
> Sam walks into the bedroom.
>
> Sam drops the apple.
>
> Q: Where is the apple?
>
> A. Bedroom

- 예시2:

> Brian is a lion.
>
> Julius is white.
>
> Julius is a lion.
>
> Bernhard is green.
>
> Q: What color is Brian?
>
> A. White

- 예시3:

> Mary journeyed to the den.
>
> Mary went back to the kitchen.
>
> John journeyed to the bedroom.
>
> Mary discarded the milk.
>
> Q: Where was the milk before the den?
>
> A. Hallway

bAbI task

http://solaris33.pythonanywhere.com/

Unit 05 | | reference

https://arxiv.org/pdf/1503.08895.pdf
https://arxiv.org/pdf/1410.3916.pdf
https://simonjisu.github.io/datascience/2017/08/04/E2EMN.html
https://www.youtube.com/watch?v=vDQf7lcenfI
https://jhui.github.io/2017/03/15/Memory-network/
http://solarisailab.com/archives/690
https://camo.githubusercontent.com/b563d59313f34e16a1fedabf235c568aa45d029b/687474703a2f2f692e696d6775722e636f6d2f6d4b745a376b422e676966
https://godongyoung.github.io/2018/03/11/Memory-Networks-paper-review.html
http://solaris33.pythonanywhere.com/