Memory network 를 이용한 QA

긴 Statment를 잘 기억하는 메모리 네트워크

23조 : 박동호(발표자), 윤재두, 서혜영

CONTENTS

배경 설명

QA 모델

RNN의 한계

Memory network 종류

코드 설명

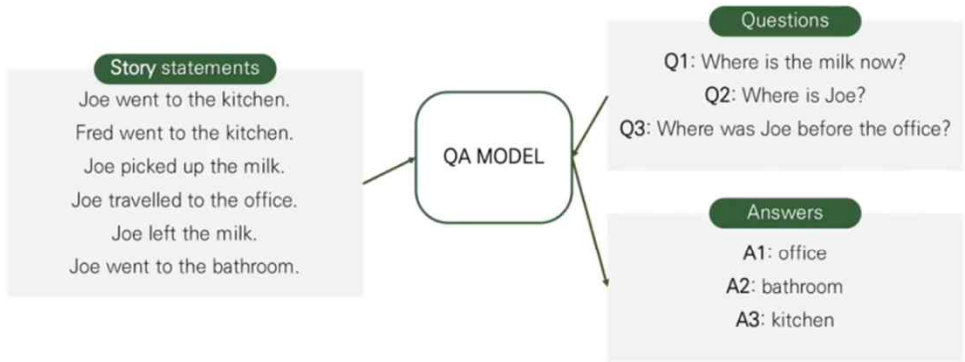
코드별 의미 파악

마무리

01. 배경 설명

Question-Answering 모델

- Question-Answering

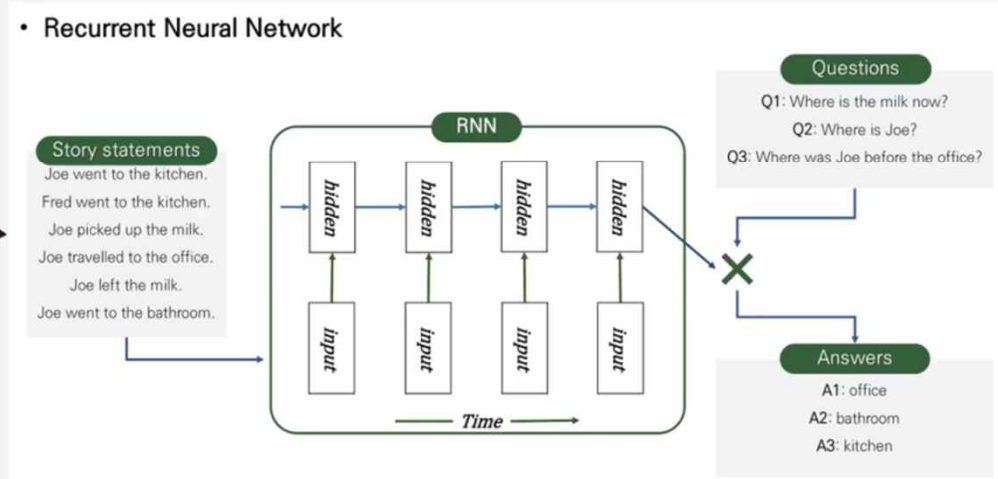


→ 사용자의 질문에 대한 답변이 될 수 있는 정답을 찾아내주는 모델

→ Story와 Question을 모델에 input후 Answer을 도출하는 구조

QA 모델에 RNN계열을 사용했었음.

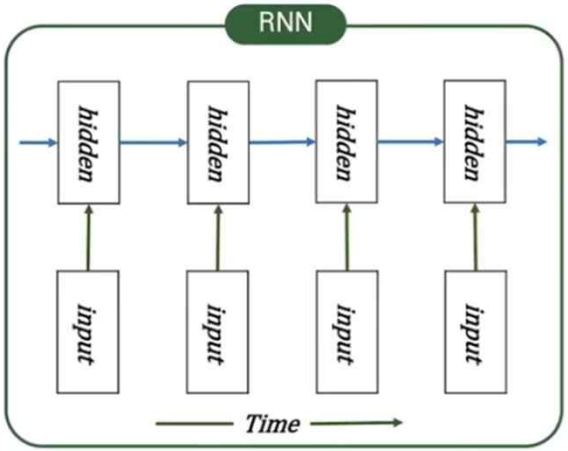
- Recurrent Neural Network



01. 배경 설명

출저 : 고려대학교 산업경영공학부 DSBA 연구실 Youtube (https://www.youtube.com/watch?v=CM5lRnNad2Q)

Recurrent Neural Network(RNN)의 한계

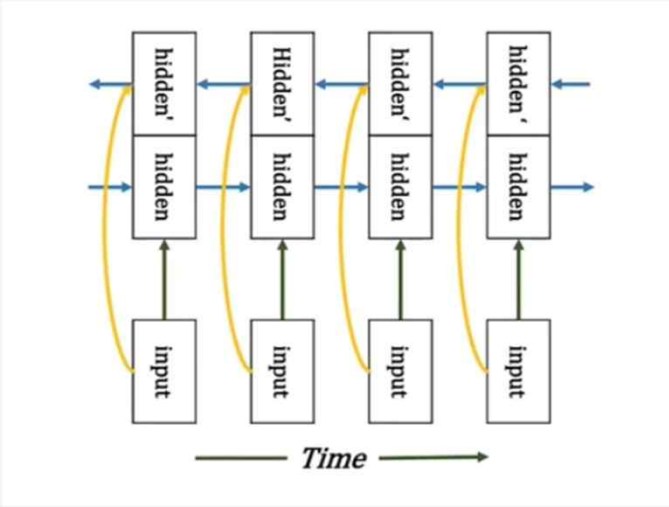


→ 순서를 기억하는데 사용되는 Sequence 모델

→ 저장공간 부족 (이전 시간의 정보를 hidden unit에 담기엔 어려움)

발전

Bi-Direction RNN



→ 양 방향으로 정보를 입력받게 하는 양방향 RNN

→ Long statement에 대해서 앞, 뒤 정보 기억 가능.

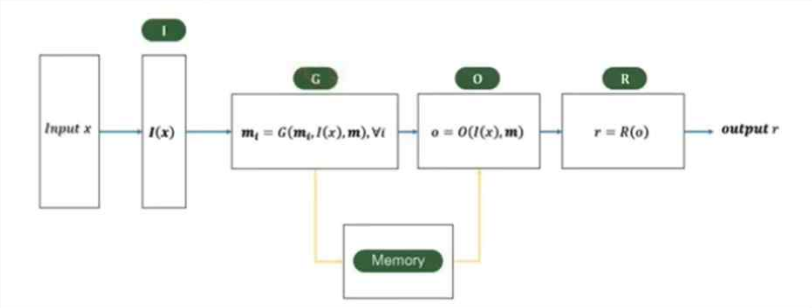
Memory network 논문 인용 출저 : https://arxiv.org/abs/1410.3916

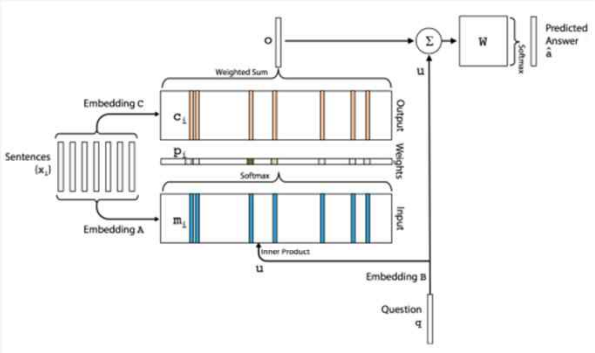
RNN's memory (encoded by hidden states and weights) is typically *too small,*
and is not compartmentalized enough to accurately remember facts from the past
(knowledge is compressed into dense vectors)

01. 배경 설명

출저 : https://reniew.github.io/46/

메모리 네트워크 (Memory Network)



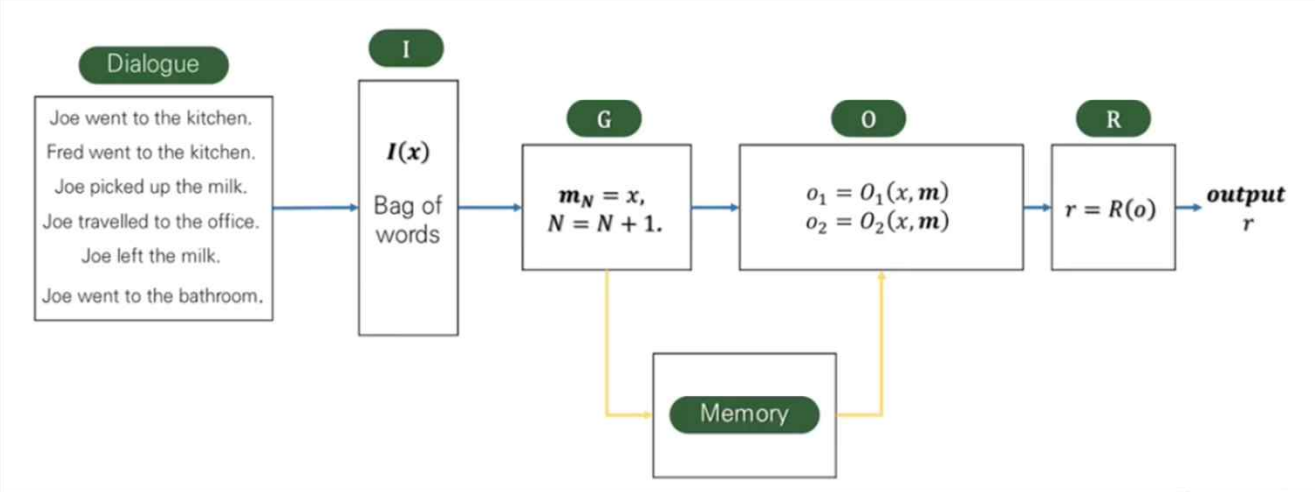


Memory Networks (2014년 10월 14일)
4가지 Components 존재 (I, G, O, R) > 기존 데이터로부터 각각의 결과를 도출해 훈련을 도와주며 결과를 내는 형식

End-to-End Memory network (2015년 3월 31일)
Component가 존재하지 않음. > 기존 데이터로부터 바로 결과를 도출하는 형식

01. 배경 설명

| 메모리 네트워크 (Memory Network)



※ Train, Test 별 각 요소들의 Update 여부

	Train	Test
Memory	O	O
I	O	X
G	O	X
O	O	X
R	O	X

Component 구성 요소

① I(input) component : input을 Feature Representation으로 encoding

② G(Generalization) component : 새로운 input이 들어오면 기존 Memory를 update.

③ O(Output) component : 질문과 가장 유사한 데이터를 Memory에서 찾음.

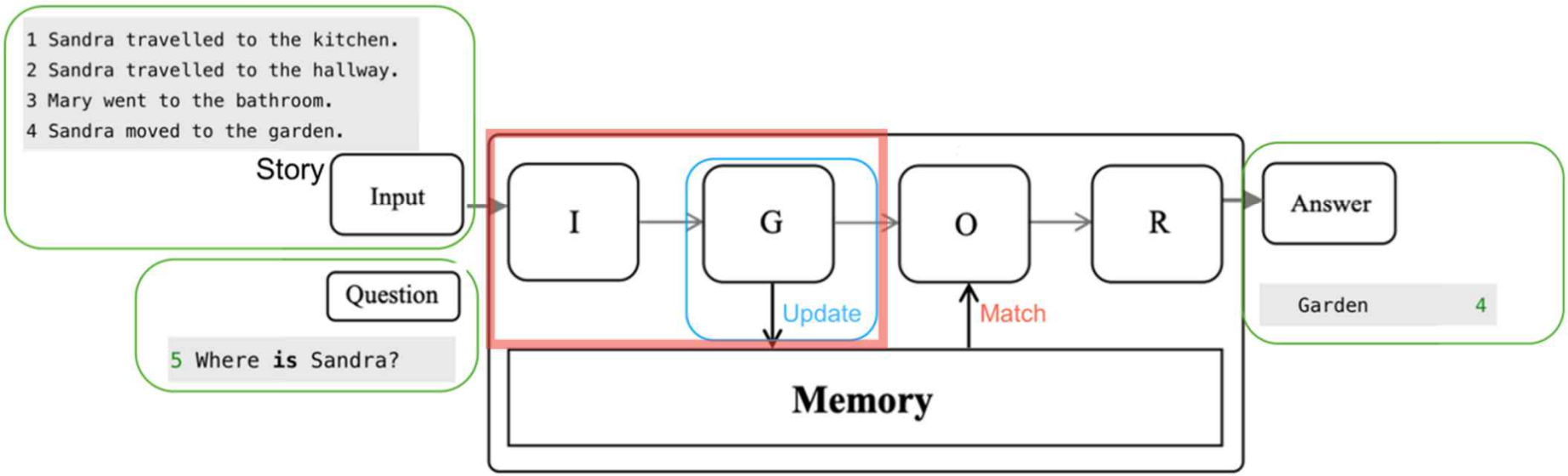
④ R(Response) component : O component의 결과물을 바탕으로 statement (words)를 도출.

➡ Memory 와 컴포넌트 4개 요소 존재.

01. 배경 설명

| 메모리 네트워크 (Memory Network)

1. I(Input) & G(Generalization) component



주어진 스토리를 Embedding (Input component) 후 메모리에 저장(Generalization component)

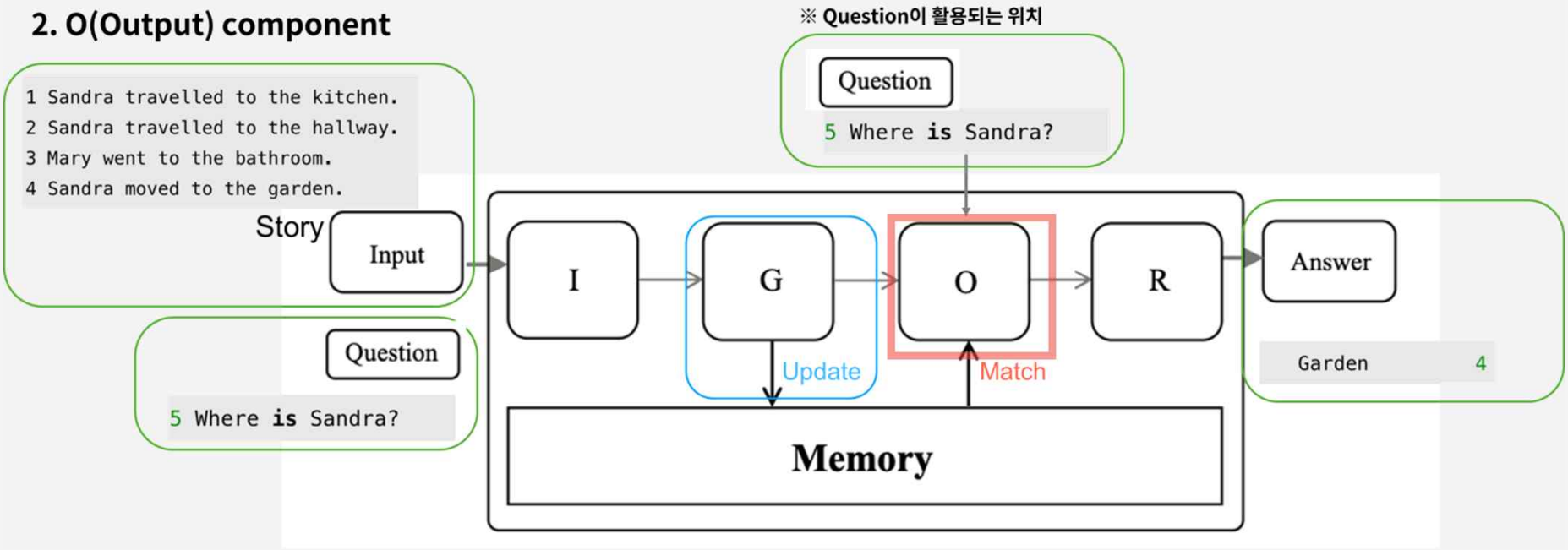
※ Input으로 들어가는 데이터가 Story일 수도 있고, Question일 수도 있다고 함.
Let us first assume this to be a sentence: either the statement of a fact, or a question to be answered by the system (later we will consider word-based input sequences).

※ 메모리가 꽉차면, 기존 메모리에 update 할 수 있다고 함.
If the memory becomes full, a procedure for "forgetting" could also be implemented by H as it chooses which memory is replaced, e.g., H could score the utility of each memory, and overwrite the least useful.
We have not explored this experimentally yet.

01. 배경 설명

| 메모리 네트워크 (Memory Network)

2. O(Output) component



Question과 Memory에 있는 데이터를 비교해서 question과 가장 관련이 있는 Story 추출

※ Supporting Memory(여러 문장 중 질문과 가장 유사하다고 판단되는 문장)을 K번 추출해내어 이후 R모듈로 보낼 수 있음.

ex.) K = 2 (2개의 Supporting memory를 사용)

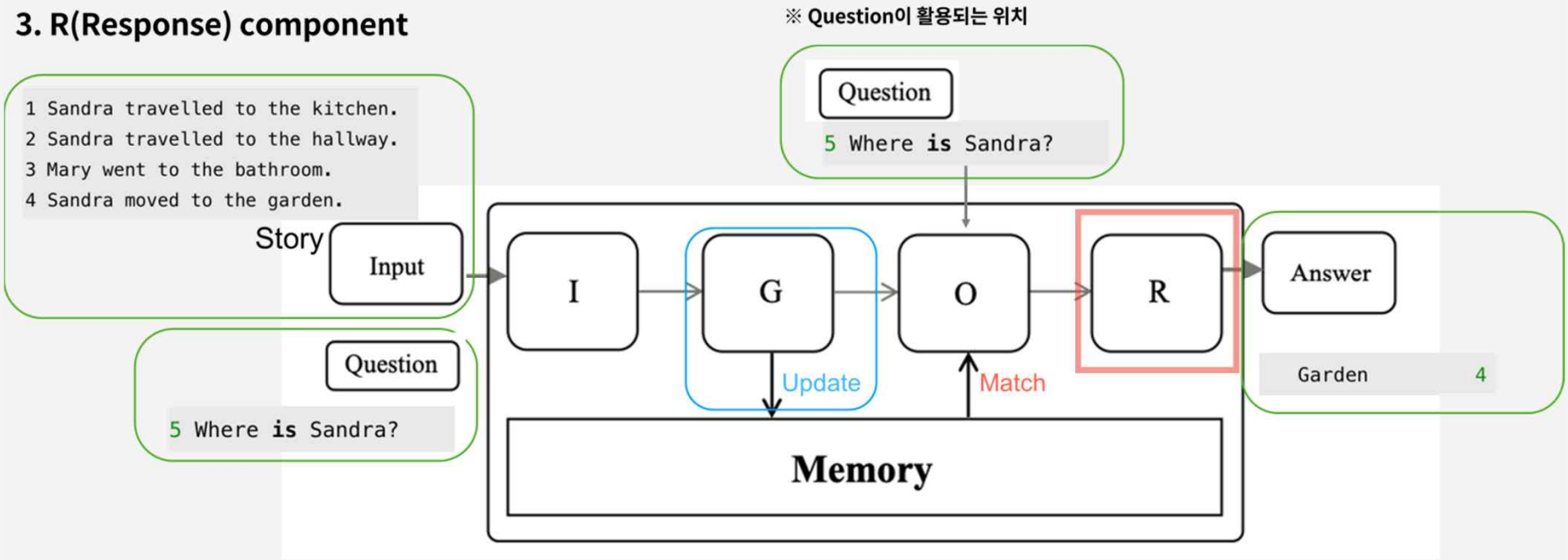
$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

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

01. 배경 설명

| 메모리 네트워크 (Memory Network)

3. R(Response) component



➡ **O(Output) 모듈의 값을 받아서 Text(ex. sentence, word)를 출력.**

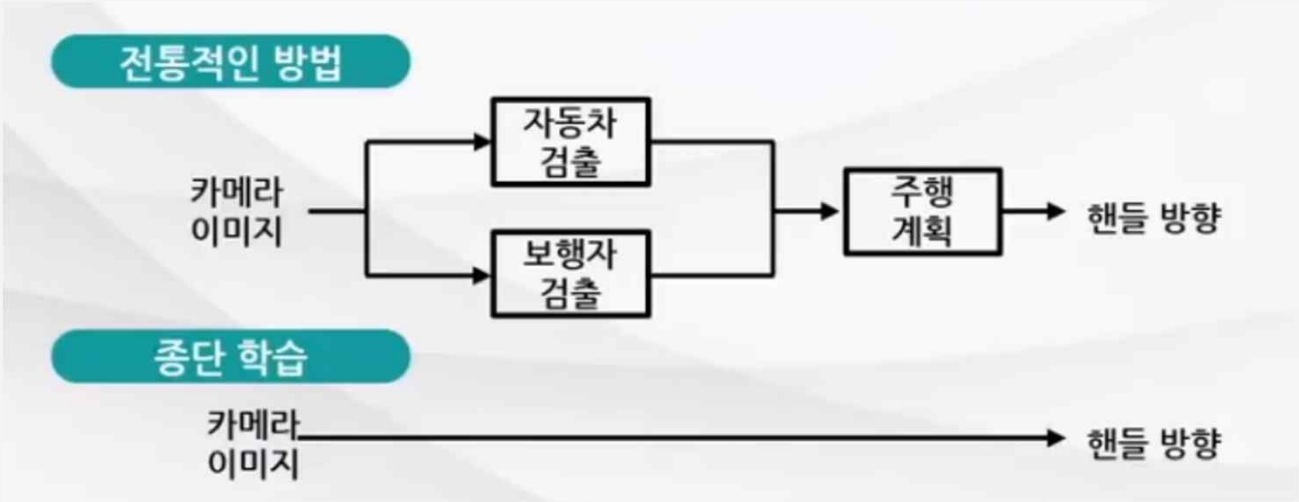
※ O 모듈은 질문과 관련있는 데이터를 찾고, 이후 R모듈이 실제 Text를 출력하는 방식임.

01. 배경 설명

출저 : T아카데미 (https://www.youtube.com/watch?v=vDQf7lcenfI&t=756s)

End-to-End Memory Network

기존 Memory network의 단점 : 학습하는 과정에서 가장 유사한 데이터를 찾는 연산 작업이 진행되기 때문에, Backpropagation 잘 이루어지지 않음.



학습 과정 중간에 연산작업 없이 끝에서 끝까지 (End-to-End) 매끄럽게 진행되는 방향으로 발전.
즉, End-to-End 학습 모델로 발전시킴.

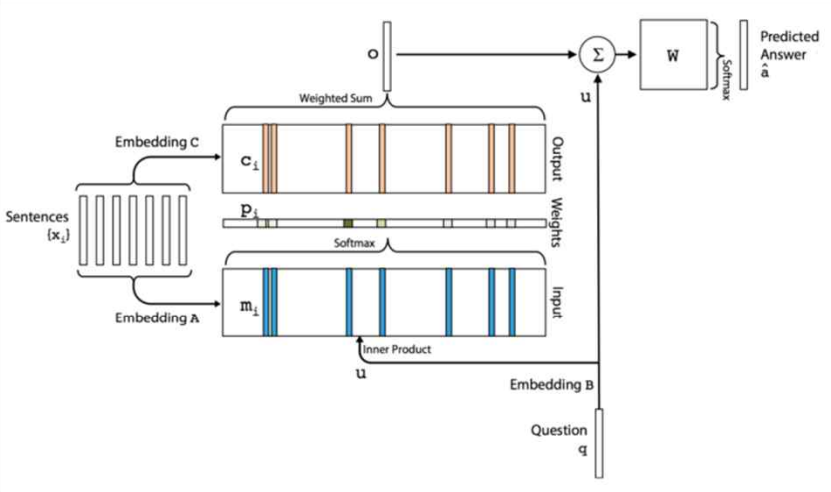
※ End-to-End 학습 모델

내부적 특징들을 스스로 학습하고 다양한 중간 단계의 과정들을 학습 알고리즘 하나로 통합하여 전체적인 작업을 End-to-End로 수행하는 모델

01. 배경 설명

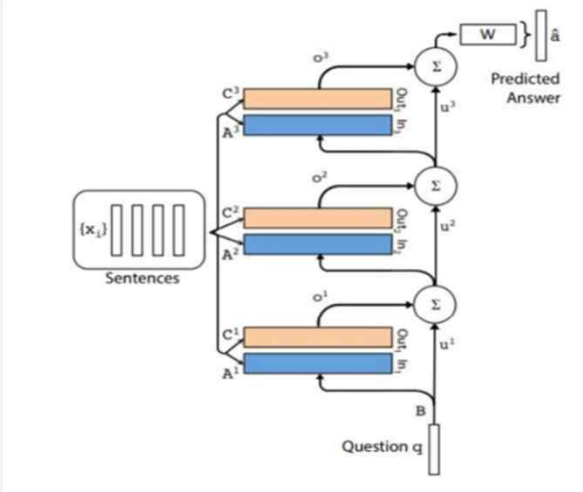
End-to-End Memory Network

Single Layer



단순한 문제(ex. 예/아니오) 해결할 때

Multiple Layers

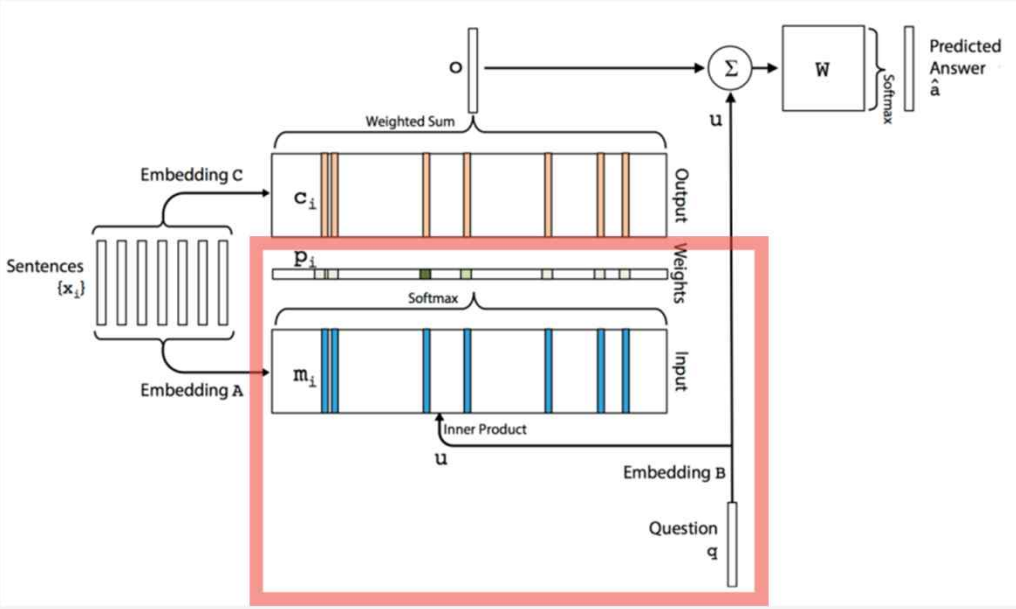


복잡한 문제(ex. 추론) 해결할 때.

01. 배경 설명

End-to-End Memory Network

1. Single Layer



1) Embedding 수행

- Sentence 와 Question에 대하여 Embedding 실시.

(Embedding A) (Embedding B)

2) 연산 작업 및 가중치(Pi) 도출

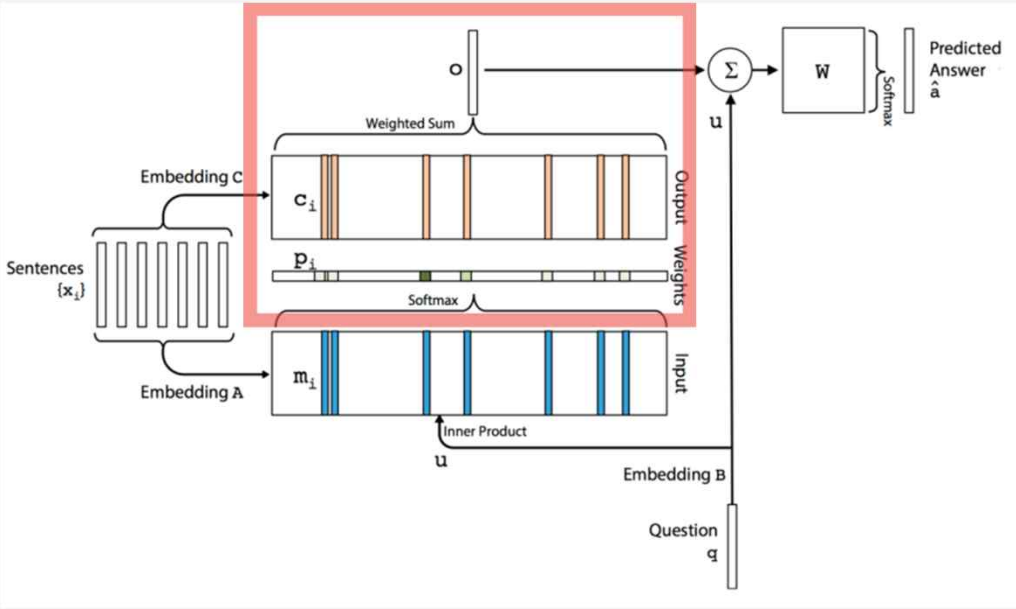
- Embedding A, B 내적 후 softmax함수를 통해 가중치(Pi) 산출.

※ Pi : Question과의 유사도 (가중치) ~ "0과 1사이의 값, 모든 값의 합은 1"

01. 배경 설명

End-to-End Memory Network

1. Single Layer



1) Pi(Question과의 유사도)를 반영

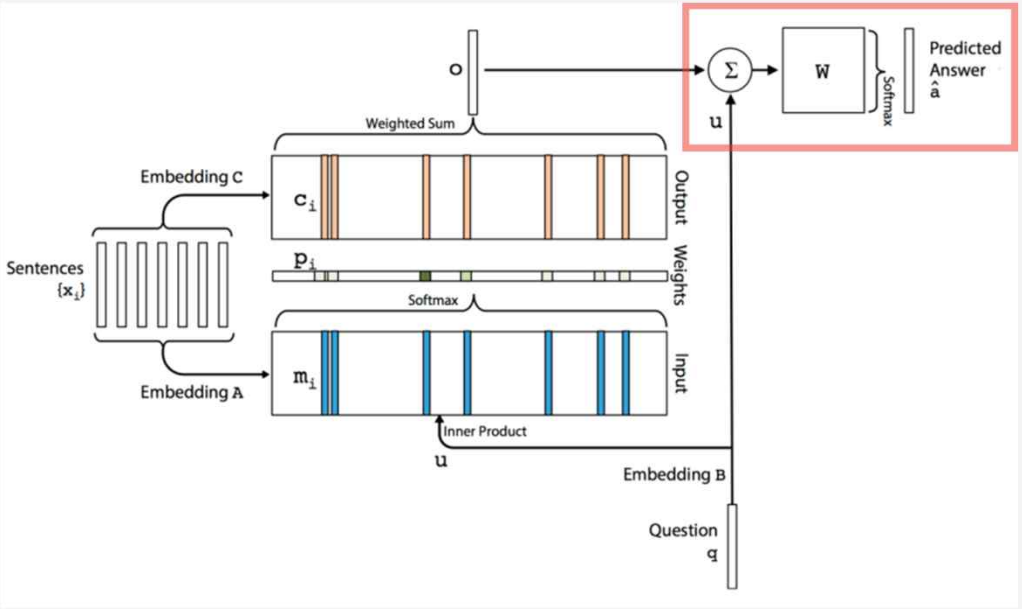
- Sentence에 대하여 Embeddding 다시 수행.(Embedding C)

- Embedding C와 Pi를 Weight sum(가중평균) 실시.
(결과값 O 산출)

01. 배경 설명

End-to-End Memory Network

1. Single Layer



1) 최종 결과값 도출

① O(Pi, Ci 가중평균)와 Embedding B(question)을 더함(Add)
※ 이어붙이기(concat)이 아니라, 값을 더하는 것임.

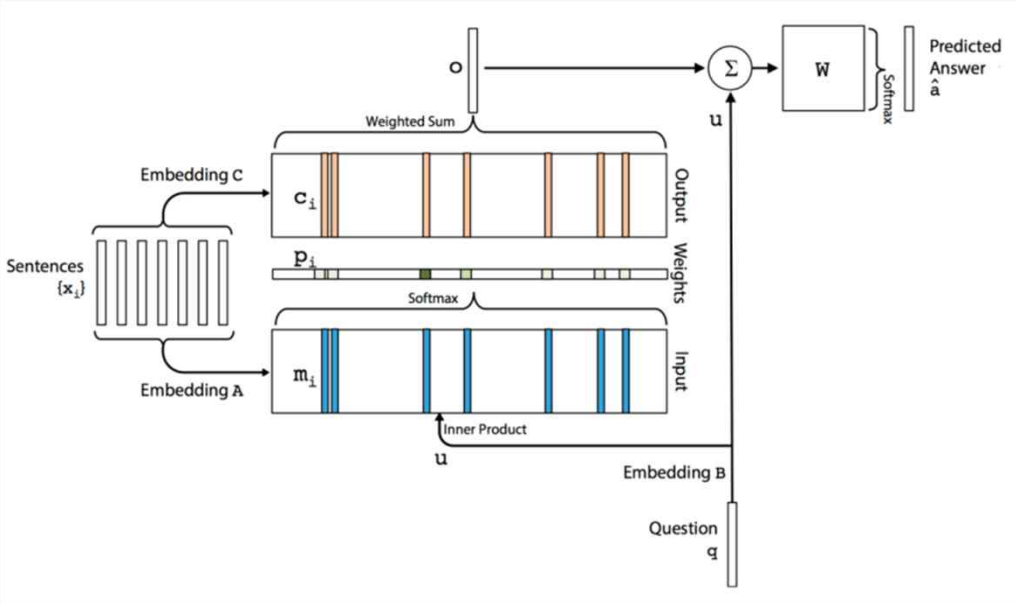
② ①값에 가중치행렬(W) 곱한 후 Softmax함수 이용하여 결과값 도출

□ 훈련 과정 추가 설명
- Parameter matrix : A, B, C, W
- loss function : Cross-entropy loss
- update : SGD

01. 배경 설명

End-to-End Memory Network

1. Single Layer



※ 논문에서의 연산 과정 별 Matrix 크기

Single Layer Version

Input: Story (x) Query (q)

$$\text{Story } (x_i) \xrightarrow[V\times s]{x_{ij}} \text{Embedding } \mathbf{A} \xrightarrow[d\times s]{\sum_j Ax_{ij}} m_i \quad (1\times s,\ d\times V,\ d\times 1)$$

$$\text{Query } (q) \xrightarrow[V\times q]{q_j} \text{Embedding } \mathbf{B} \xrightarrow[d\times q]{\sum_j Bq_j} u \quad (1\times q,\ d\times V,\ d\times 1)$$

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

$$\text{Story } (x_i) \xrightarrow[V\times s]{x_{ij}} \text{Embedding } \mathbf{C} \xrightarrow[d\times s]{\sum_j Cx_{ij}} c_i \quad (1\times s,\ d\times V,\ d\times 1)$$

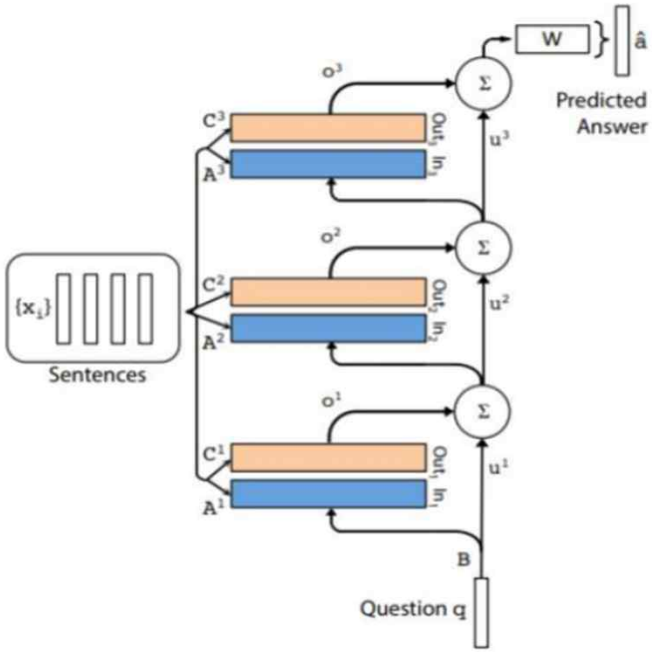
$$o = \sum_i p_i c_i = cp \quad (d\times 1,\ (d\times m)(m\times 1))$$

$$\hat{a} = \text{Softmax}(W(o+u)) \quad (V\times 1,\ (V\times d)\ (d\times 1))$$

01. 배경 설명

| End-to-End Memory Network

2. Multiple layers



1) 한 종류의 데이터 셋으로만 풀 수 있는 문제가 아닐 경우 활용
(ex. 추론 문제)

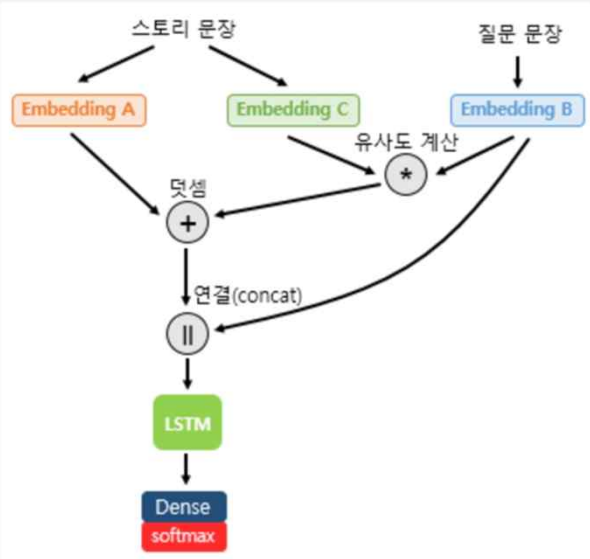
2) 이전 Single Layer의 출력값이 다음 layer의 input값.
layer를 거칠때마다 Question을 Update시킴.

3) layer마다의 연산 과정은 Single layer 참고.

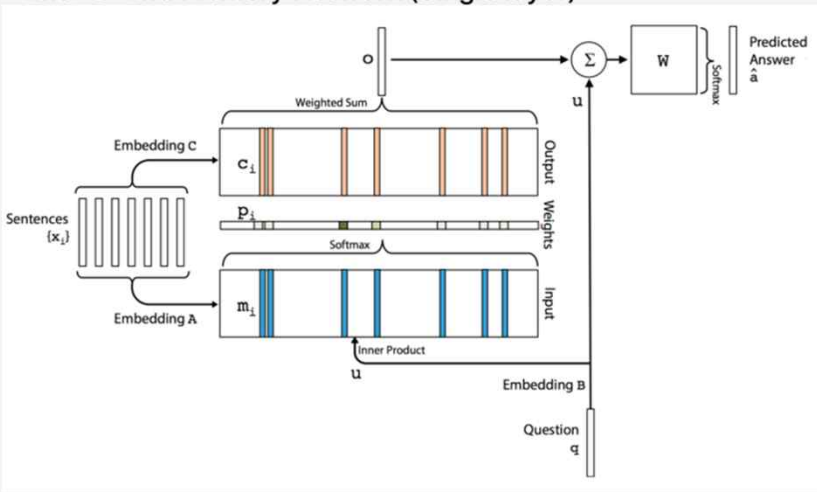
01. 배경 설명

책 기반 Memory Network 구조

※ 책에서 제시된 Memory Network 구조



※ End-to-End Memory Network (Single layer)



책의 Memory Network 구조 : End-to-End 형식으로 판단

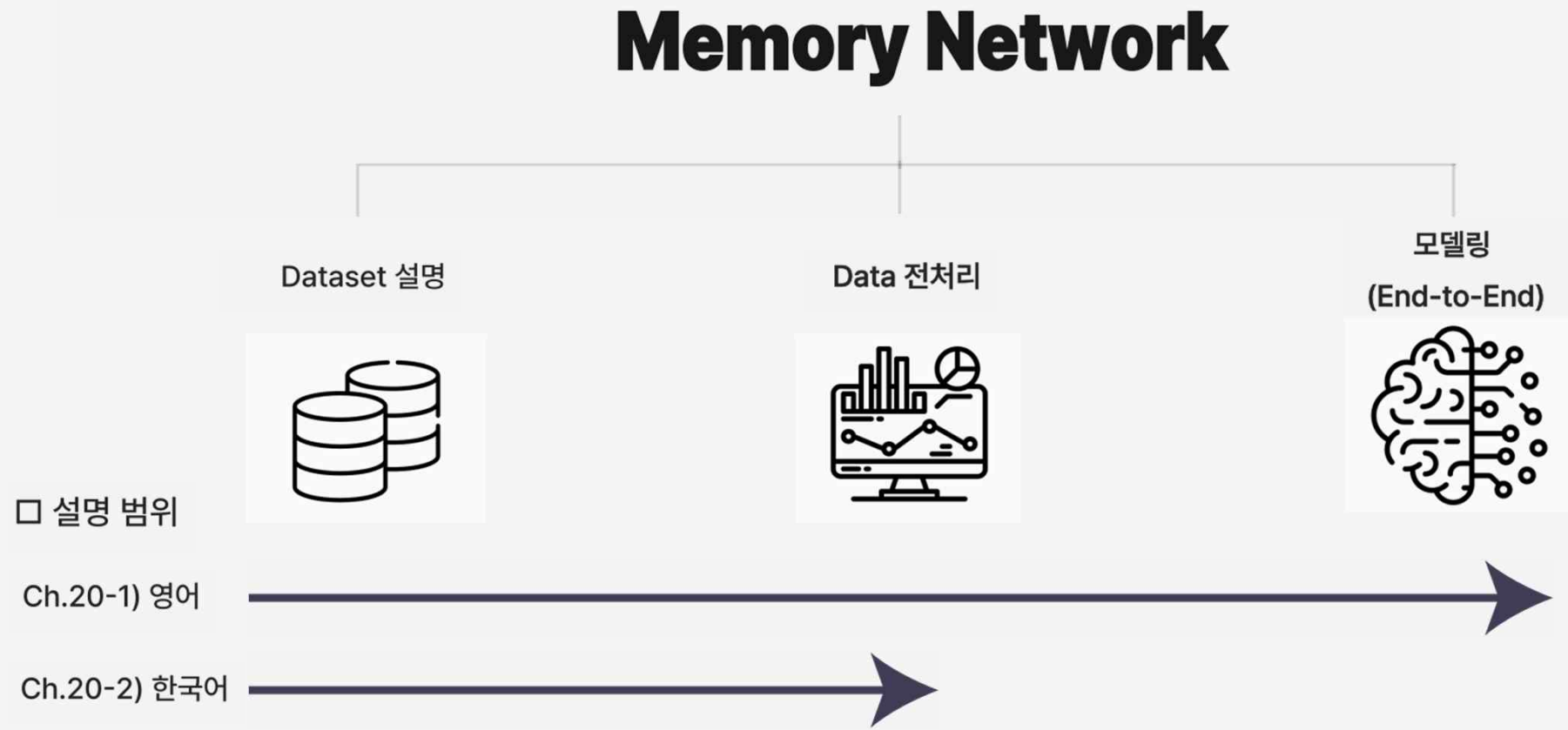
※ End-to-End 형식으로 판단 내린 근거.

① 기존 Sentence에서 임베딩 2번.

② 유사도 계산 방식

02. 코드 설명

| Chapter 20. 질의 응답(Question Answering, QA) 설명 개요



02. 코드 설명

데이터셋 설명 (Babi Project)

Format

```
ID text
ID text
ID text
ID question[tab]answer[tab]supporting_fact ID.
...
```

*BaBI 데이터셋 특징 설명(간략)

1) 20 BaBI tasks

2) Story + Question + Answering

3) 비교적 간단하고 적은 양의 단어로 이루어진 쉬운 문장들

4) 각 tasks 별 서로 추론이 필요

→ supporting_fact ID는 훈련 과정에서 생략.

Example

```
1 Mary moved to the bathroom.
2 John went to the hallway.
3 Where is Mary?	bathroom	1
4 Daniel went back to the hallway.
5 Sandra moved to the garden.
6 Where is Daniel?	hallway	4
7 John moved to the office.
8 Sandra journeyed to the bathroom.
9 Where is Daniel?	hallway	4
10 Mary moved to the hallway.
11 Daniel travelled to the office.
12 Where is Daniel?	office	11
13 John went back to the garden.
14 John moved to the bedroom.
15 Where is Sandra?	bathroom	8
1 Sandra travelled to the office.
2 Sandra went to the bathroom.
3 Where is Sandra?	bathroom	2
```

02. 코드 설명

| Ch.20-1 영어 _ Data 전처리

데이터 Load

```
path = get_file('babi-tasks-v1-2.tar.gz', origin='https://s3.amazonaws.com/text-datasets/'
                'babi_tasks_1-20_v1-2.tar.gz')
```

```
with tarfile.open(path) as tar:
 tar.extractall()
 tar.close()

DATA_DIR = 'tasks_1-20_v1-2/en-10k'
TRAIN_FILE = os.path.join(DATA_DIR, "qa1_single-supporting-fact_train.txt")
TEST_FILE = os.path.join(DATA_DIR, "qa1_single-supporting-fact_test.txt")
```

```
i = 0
lines = open(TRAIN_FILE , "rb")
for line in lines:
    line = line.decode("utf-8").strip()
    # lno, text = line.split(" ", 1) # ID와 TEXT 분리
    i = i + 1
    print(line)
    if i == 20:
      break
```

스토리, 질문 , 답변 분리

```
def read_data(dir):
    stories, questions, answers = [], [], [] # 각각 스토리, 질문, 답변을 저장할 예정
    story_temp = [] # 현재 시점의 스토리 임시 저장
    lines = open(dir, "rb")

    for line in lines:
        line = line.decode("utf-8") # b' 제거
        line = line.strip() # '\n' 제거
        idx, text = line.split(" ", 1) # 맨 앞에 있는 id number 분리
        # 여기까지는 모든 줄에 적용되는 전처리

        if int(idx) == 1:
            story_temp = []

        if "\t" in text: # 현재 읽는 줄이 질문 (tab) 답변 (tab)인 경우
            question, answer, _ = text.split("\t") # 질문과 답변을 각각 저장
            stories.append([x for x in story_temp if x]) # 지금까지의 누적 스토리를 스토리에 저장
            questions.append(question)
            answers.append(answer)

        else: # 현재 읽는 줄이 스토리인 경우
            story_temp.append(text) # 임시 저장

    lines.close()
    return stories, questions, answers
```

02. 코드 설명

Ch.20-1 영어 _ Data 전처리

토큰화 작업

```python
def tokenize(sent):
    return [ x.strip() for x in re.split('(\W+)', sent) if x and x.strip()]
```

```python
def preprocess_data(train_data, test_data):
    counter = FreqDist()

    # 두 문장의 story를 하나의 문장으로 통합하는 함수
    flatten = lambda data: reduce(lambda x, y: x + y, data)

    # 각 샘플의 길이를 저장하는 리스트
    story_len = []
    question_len = []

    for stories, questions, answers in [train_data, test_data]:
        for story in stories:
            stories = tokenize(flatten(story)) # 스토리의 문장들을 펼친 후 토큰화
            story_len.append(len(stories)) # 각 story의 길이 저장
            for word in stories: # 단어 집합에 단어 추가
                counter[word] += 1
        for question in questions:
            question = tokenize(question)
            question_len.append(len(question))
            for word in question:
                counter[word] += 1
        for answer in answers:
            answer = tokenize(answer)
            for word in answer:
                counter[word] += 1

    # 단어 집합 생성
    word2idx = {word : (idx + 1) for idx, (word, _) in enumerate(counter.most_common())}
    idx2word = {idx : word for word, idx in word2idx.items()}

    # 가장 긴 샘플의 길이
    story_max_len = np.max(story_len)
    question_max_len = np.max(question_len)

    return word2idx, idx2word, story_max_len, question_max_len
```

단어 집합 생성

```
{'to': 1, 'the': 2, '.': 3, 'went': 4, 'Sandra': 5, 'John': 6, 'Daniel': 7, 'Mary': 8, 'travelled': 9,
```

스토리와 질문 Max length 구하기

```python
print('스토리의 최대 길이 :',story_max_len)
print('질문의 최대 길이 :',question_max_len)
```

```
스토리의 최대 길이 : 68
질문의 최대 길이 : 4
```

02. 코드 설명

Ch.20-1 영어 _ Data 전처리

정수 인코딩 및 패딩

```
def vectorize(data, word2idx, story_maxlen, question_maxlen):
    Xs, Xq, Y = [], [], []
    flatten = lambda data: reduce(lambda x, y: x + y, data)

    stories, questions, answers = data
    for story, question, answer in zip(stories, questions, answers):
        xs = [word2idx[w] for w in tokenize(flatten(story))]
        xq = [word2idx[w] for w in tokenize(question)]
        Xs.append(xs)
        Xq.append(xq)
        Y.append(word2idx[answer])

        # 스토리와 질문은 각각의 최대 길이로 패딩
        # 정답은 원-핫 인코딩
    return pad_sequences(Xs, maxlen=story_maxlen),\
           pad_sequences(Xq, maxlen=question_maxlen),\
           to_categorical(Y, num_classes=len(word2idx) + 1)
```

```
Xstrain, Xqtrain, Ytrain = vectorize(train_data, word2idx, story_max_len, question_max_len)
Xstest, Xqtest, Ytest = vectorize(test_data, word2idx, story_max_len, question_max_len)
```

```
print(Xstrain.shape, Xqtrain.shape, Ytrain.shape, Xstest.shape, Xqtest.shape, Ytest.shape)
```

```
(10000, 68) (10000, 4) (10000, 22) (1000, 68) (1000, 4) (1000, 22)
```

```
Xstrain[3576]
```

```
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  6,  4, 11,  1,  2, 13,  3,  8,  4,
        1,  2, 17,  3,  5,  4, 11,  1,  2, 18,  3,  6,  9,  1,  2, 18,  3],
      dtype=int32)
```

```
Xqtrain[3576]
```

```
array([19, 20,  6, 21], dtype=int32)
```

```
Ytrain[3576]
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0.], dtype=float32)
```

02. 코드 설명

| Ch.20-1 영어 _ 모델링

구현 시 사용한 Library

```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Permute, dot, add, concatenate
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input, Activation
```

Input 데이터 받는 변수 생성.

```
# 플레이스 홀더. 입력을 담는 변수
input_sequence = Input((story_max_len,))
question = Input((question_max_len,))

print('Stories :', input_sequence)
print('Question:', question)
```

모델 관련 하이퍼파라미터 설정

```
# 에포크 횟수
train_epochs = 120
# 배치 크기
batch_size = 32
# 임베딩 크기
embed_size = 50
# LSTM의 크기
lstm_size = 64
# 과적합 방지 기법인 드롭아웃 적용 비율
dropout_rate = 0.30
```

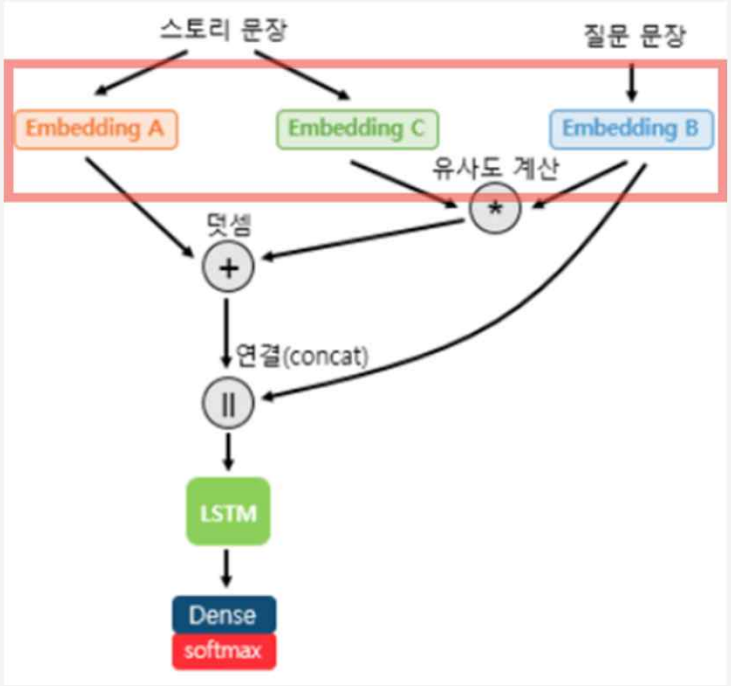
02. 코드 설명

Ch.20-1 영어 _ 모델링

```
# 스토리를 위한 첫번째 임베딩. 그림에서의 Embedding A
input_encoder_m = Sequential()
input_encoder_m.add(Embedding(input_dim=vocab_size,
                              output_dim=embed_size))
input_encoder_m.add(Dropout(dropout_rate))
# 결과 : (samples, story_max_len, embedding_dim) / 샘플의 수, 문장의 최대 길이, 임베딩 벡터의 차원

# 스토리를 위한 두번째 임베딩. 그림에서의 Embedding C
# 임베딩 벡터의 차원을 question_max_len(질문의 최대 길이)로 한다.
input_encoder_c = Sequential()
input_encoder_c.add(Embedding(input_dim=vocab_size,
                              output_dim=question_max_len))
input_encoder_c.add(Dropout(dropout_rate))
# 결과 : (samples, story_max_len, question_max_len) / 샘플의 수, 문장의 최대 길이, 질문의 최대 길이(임베딩 벡터의 차원)
```

```
# 질문을 위한 임베딩. 그림에서의 Embedding B
question_encoder = Sequential()
question_encoder.add(Embedding(input_dim=vocab_size,
                               output_dim=embed_size,
                               input_length=question_max_len))
question_encoder.add(Dropout(dropout_rate))
# 결과 : (samples, question_max_len, embedding_dim) / 샘플의 수, 질문의 최대 길이, 임베딩 벡터의 차원
```



02. 코드 설명

Ch.20-1 영어 _ 모델링

유사도 계산 (Key & Query)

```
# 스토리 단어들과 질문 단어들 간의 유사도를 구하는 과정
# 유사도는 내적을 사용한다.
match = dot([input_encoded_m, question_encoded], axes=-1, normalize=False)
match = Activation('softmax')(match)
print('Match shape', match)
# 결과 : (samples, story_maxlen, question_max_len) / 샘플의 수, 문장의 최대 길이, 질문의 최대 길이
```

```
Match shape KerasTensor(type_spec=TensorSpec(shape=(None, 68, 4), dtype=tf.float32, name=None),
```

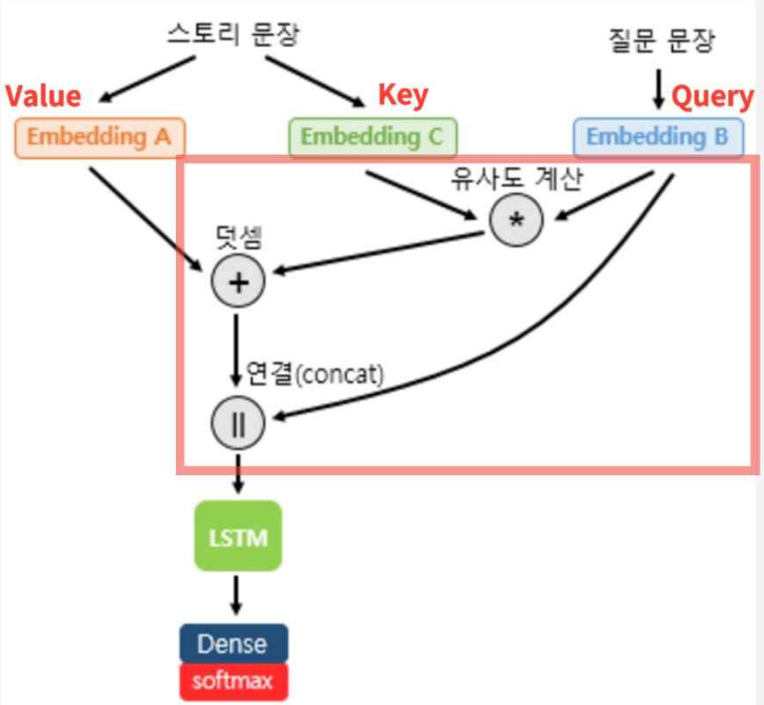
덧셈 & 연결

```
# 유사도가 반영된 어텐션 분포 행렬과 임베딩 c를 거친 스토리 행렬을 더한다.
# 두 행렬 모두 크기는 (68, 4)이다.
# 이로부터 얻은 행렬은 어텐션 값 행렬(Attention Value Matrix)이다.
response = add([match, input_encoded_c])
```

```
# 질문 행렬은 (4, 50)의 크기를 가진다.
# 하지만 어텐션 값 행렬의 크기는 (68, 4)이다.
# 이 두 개를 연결시켜주기 위해서 어텐션 값 행렬의 크기를 (4, 68)로 변환해준다.
response = Permute((2, 1))(response)  # (samples, question_max_len, story_max_len)
print('Response shape', response)

# 질문 행렬과 어텐션 값 행렬을 연결한다.
# (4, 118)의 크기를 가진다.
answer = concatenate([response, question_encoded])
print('Answer shape', answer)
```

```
Response shape KerasTensor(type_spec=TensorSpec(shape=(None, 4, 68), dtype=tf.float32, name=None),
Answer shape KerasTensor(type_spec=TensorSpec(shape=(None, 4, 118), dtype=tf.float32, name=None),
```



※ Attention 메커니즘과 유사

Attention score(query, key) = queryT*key

02. 코드 설명

Ch.20-1 영어 _ 모델링

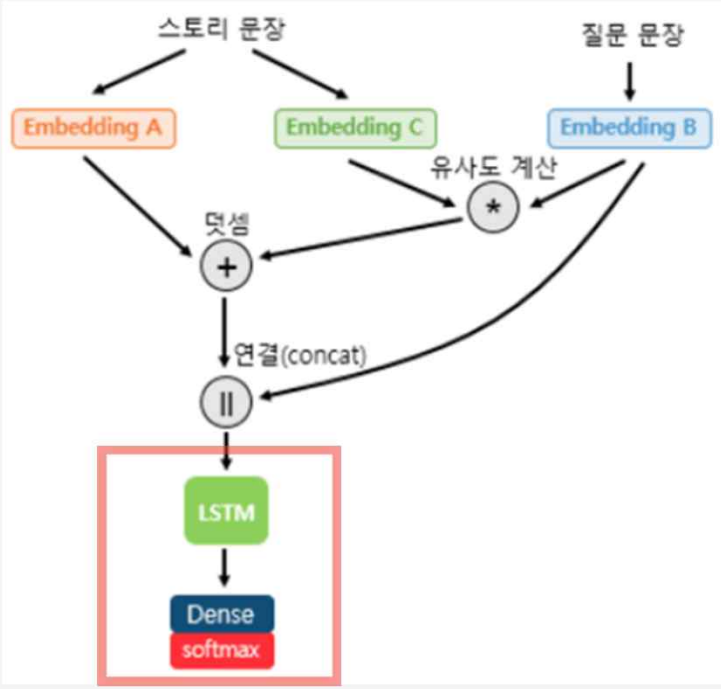
```
answer = LSTM(lstm_size)(answer)
answer = Dropout(dropout_rate)(answer)
answer = Dense(vocab_size)(answer)
answer = Activation('softmax')(answer)
```

```
# build the final model
model = Model([input_sequence, question], answer)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['acc'])

print(model.summary())

# start training the model
history = model.fit([Xstrain, Xqtrain],
         Ytrain, batch_size, train_epochs,
         validation_data=([Xstest, Xqtest], Ytest))

# save model
model.save('model.h5')
```



02. 코드 설명

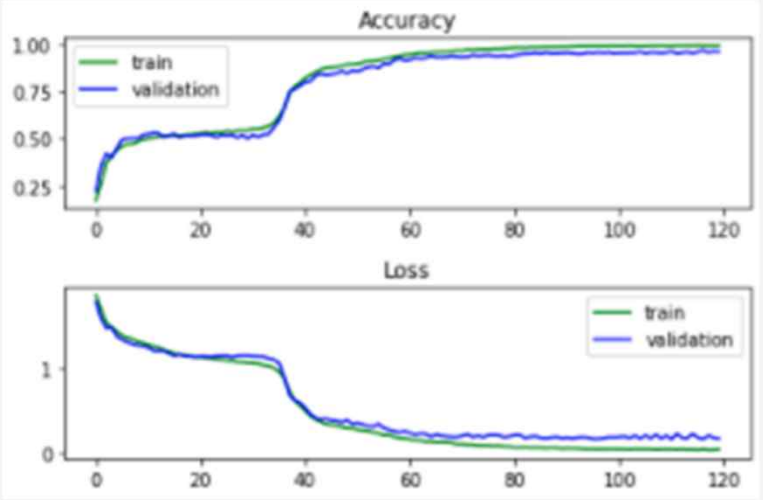
| Ch.20-1 영어 _ 모델링

학습결과 Tracking & 평가

```
# plot accuracy and loss plot
plt.subplot(211)
plt.title("Accuracy")
plt.plot(history.history["acc"], color="g", label="train")
plt.plot(history.history["val_acc"], color="b", label="validation")
plt.legend(loc="best")

plt.subplot(212)
plt.title("Loss")
plt.plot(history.history["loss"], color="g", label="train")
plt.plot(history.history["val_loss"], color="b", label="validation")
plt.legend(loc="best")

plt.tight_layout()
plt.show()
```




```
print("\n 테스트 정확도: %.4f" % (model.evaluate([Xstest, Xqtest], Ytest)[1]))
```

```
32/32 [==============================] - 0s 3ms/step - loss: 0.1666 - acc: 0.9580

 테스트 정확도: 0.9580
```

02. 코드 설명

| Ch.20-1 영어 _ 모델링

예측 사례 (임의로 1개 지정)

```
ytest = np.argmax(Ytest, axis=1)

Ytest_ = model.predict([Xstest, Xqtest])
ytest_ = np.argmax(Ytest_, axis=1)

test = list(ytest_)
test_text = list(map(lambda x : idx2word[x],test))

print("스토리 1가지 : '\n' {}".format(test_stories[957]))
print("질문 1가지 : {}".format(test_questions[957]))
print("그에 대한 답변 : {}".format(test_text[957]))
```

```
스토리 1가지 : '
' ['John went back to the office.', 'Daniel went to the bathroom.', 'John journeyed to the kitchen.', 'Mary went to the bathroom.',
'Sandra moved to the garden.', 'Daniel went back to the bedroom.']
질문 1가지 : Where is Sandra?
그에 대한 답변 : garden
```

※ 추가 10개 사례 비교

```
질문                    | 실제값   | 예측값
-----------------------------------------------
Where is John?          | hallway | hallway
Where is Mary?          | bathroom | bathroom
Where is Sandra?        | kitchen | kitchen
Where is Sandra?        | hallway | hallway
Where is Sandra?        | kitchen | kitchen
Where is Sandra?        | hallway | hallway
Where is Sandra?        | garden | garden
Where is Daniel?        | hallway | hallway
Where is Sandra?        | office | office
Where is Daniel?        | office | office
```

02. 코드 설명

| Ch.20-2 한국어 _ Data 전처리

형태소 분석기 라이브러리

```
1 from ckonlpy.tag import Twitter
```

```
1 twitter = Twitter()
```

```
/usr/local/lib/python3.6/dist-packages/konlpy/tag/_okt.py:16: UserWarning: "Twitter" has changed to "Okt" since KoNLPy v0.4.5.
  warn('"Twitter" has changed to "Okt" since KoNLPy v0.4.5.')
```

```
1 print(twitter.morphs('은경이는 화장실로 이동했습니다.'))
2 print(twitter.morphs('경임이는 정원으로 가버렸습니다.'))
3 print(twitter.morphs('수종이는 복도로 뛰어갔습니다.'))
4 print(twitter.morphs('필웅이는 부엌으로 복귀했습니다.'))
5 print(twitter.morphs('수종이는 사무실로 갔습니다.'))
6 print(twitter.morphs('은경이는 침실로 갔습니다.'))
```

```
['은', '경이', '는', '화장실', '로', '이동', '했습니다', '.']
['경', '임', '이', '는', '정원', '으로', '가버렸습니다', '.']
['수종', '이', '는', '복도', '로', '뛰어갔습니다', '.']
['필웅이', '는', '부엌', '으로', '복귀', '했습니다', '.']
['수종', '이', '는', '사무실', '로', '갔습니다', '.']
['은', '경이', '는', '침실', '로', '갔습니다', '.']
```

명사 추가작업.

```
1 twitter.add_dictionary('은경이', 'Noun')
2 twitter.add_dictionary('경임이', 'Noun')
3 twitter.add_dictionary('수종이', 'Noun')
```

```
1 print(twitter.morphs('은경이는 화장실로 이동했습니다.'))
2 print(twitter.morphs('경임이는 정원으로 가버렸습니다.'))
3 print(twitter.morphs('수종이는 복도로 뛰어갔습니다.'))
4 print(twitter.morphs('필웅이는 부엌으로 복귀했습니다.'))
5 print(twitter.morphs('수종이는 사무실로 갔습니다.'))
6 print(twitter.morphs('은경이는 침실로 갔습니다.'))
```

```
['은경이', '는', '화장실', '로', '이동', '했습니다', '.']
['경임이', '는', '정원', '으로', '가버렸습니다', '.']
['수종이', '는', '복도', '로', '뛰어갔습니다', '.']
['필웅이', '는', '부엌', '으로', '복귀', '했습니다', '.']
['수종이', '는', '사무실', '로', '갔습니다', '.']
['은경이', '는', '침실', '로', '갔습니다', '.']
```

02. 코드 설명

| Ch.20-2 한국어 _ 전처리 & 모델 구현

토큰화 ~ 모델 구현

02. 코드 설명

| Ch.20-2 한국어 _ 전처리 & 모델 구현

결과 예측

```
NUM_DISPLAY = 30

print("{:18}|{:5}|{}".format("질문", "실제값", "예측값"))
print(39 * "-")

for i in range(NUM_DISPLAY):
    question = " ".join([idx2word[x] for x in Xqtest[i].tolist()])
    label = idx2word[ytest[i]]
    prediction = idx2word[ytest_[i]]
    print("{:20}: {:7} {}".format(question, label, prediction))
```

```
질문                  |실제값   |예측값
---------------------------------------
은경이 는 어디 야 ?          : 복도    복도
필웅이 는 어디 야 ?          : 화장실   화장실
경임이 는 어디 야 ?          : 부엌    부엌
경임이 는 어디 야 ?          : 복도    복도
경임이 는 어디 야 ?          : 부엌    부엌
경임이 는 어디 야 ?          : 복도    복도
경임이 는 어디 야 ?          : 정원    정원
수종이 는 어디 야 ?          : 복도    복도
경임이 는 어디 야 ?          : 사무실   복도
수종이 는 어디 야 ?          : 사무실   복도
필웅이 는 어디 야 ?          : 부엌    부엌
필웅이 는 어디 야 ?          : 정원    정원
수종이 는 어디 야 ?          : 사무실   사무실
필웅이 는 어디 야 ?          : 침실    침실
필웅이 는 어디 야 ?          : 침실    침실
은경이 는 어디 야 ?          : 부엌    부엌
은경이 는 어디 야 ?          : 정원    정원
은경이 는 어디 야 ?          : 부엌    부엌
수종이 는 어디 야 ?          : 사무실   부엌
은경이 는 어디 야 ?          : 부엌    정원
필웅이 는 어디 야 ?          : 복도    복도
은경이 는 어디 야 ?          : 사무실   사무실
은경이 는 어디 야 ?          : 사무실   사무실
경임이 는 어디 야 ?          : 복도    사무실
수종이 는 어디 야 ?          : 침실    침실
경임이 는 어디 야 ?          : 침실    침실
필웅이 는 어디 야 ?          : 침실    침실
수종이 는 어디 야 ?          : 부엌    부엌
수종이 는 어디 야 ?          : 부엌    부엌
수종이 는 어디 야 ?          : 부엌    복도
```

03. 마무리

감사합니다