

2018 투빅스 Conference

모델의 이해를 돕기 위한 가장 최적의 설명!

DR & QA Model Description

고려대학교 통계학과 | 장유영

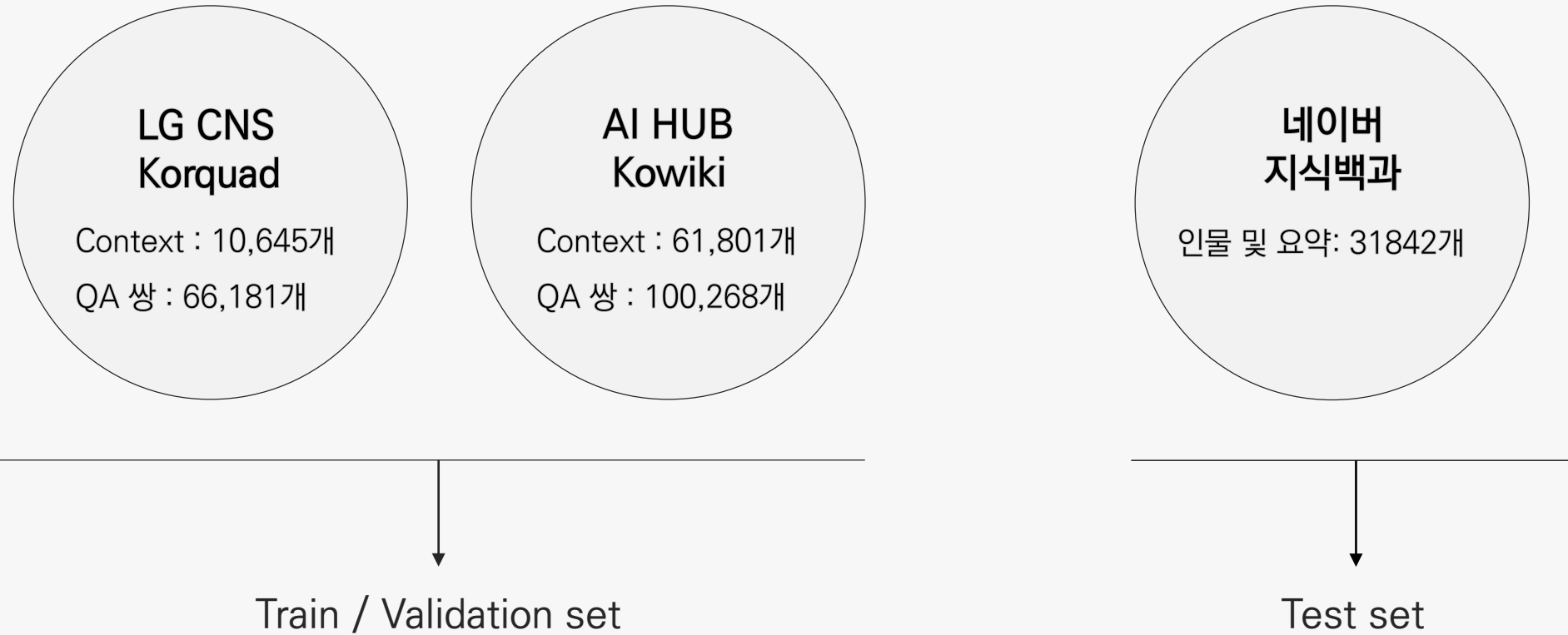
CONTENTS

I	Background	03
II	Document Retriever 설명	16
III	QA 모델 설명	19
IV	Code Review	25

```
35 self.logger = logging.getLogger(__name__)
36
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'), 'a')
39     self.file.seek(0)
40     self.fingerprints.update(requests_log)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('DEBUG', False)
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```

1. Background

데이터 수집 및 정제



Raw 데이터

id	(‘8_C58_wiki_327-1’,
context	<p>‘그러나 조조는 암살에 실패하여 진류로 달아나 각지의 제후들과 함께 반동탁 연합군을 일으켰다. 연합군에 밀려 장안으로 천도한 후에도 동탁의 행동이 달라지지 않자, 왕윤은 자신의 수양딸 초선으로 하여금 동탁과 여포의 사이를 이간질시키고 두 사람의 사이를 벌어지게 만들었다. 여포가 동탁을 죽일 결심을 굳히자, 이숙을 동탁에게 보내어 헌제가 동탁에게 선양하려 한다고 전하고 궁궐로 불러들여 동탁을 죽였다. 이것을 연환지계라 한다. 그러나 동탁의 잔당이 장안을 공격해 오자 장안을 빼앗기고 이각에게 붙잡혀 죽었다. 왕굉(왕굉 (장문)) 왕릉(왕릉 (조위)) 왕윤(王允, 137년 ~ 192년 7월 4일(음력 6월 7일))은 중국 후한 말의 정치가로, 자는 자사(子師)이며 병주(병주 (중국))(并州) 태원군(타이위안 시)(太原郡) 기현(祁縣) 사람이다. 여포(呂布)를 움직여 전횡을 일삼던 동탁(董卓)을 죽였으나, 동탁의 하인 이각과 곽사의 반격을 받고 헌제 앞에서 죽었다.’,</p>
question	— ‘동탁과 여포 사이를 이간질 한 사람은 누구지’,
answer	<p>／ ‘왕윤’,</p> <p>326,</p> <p>328)</p>

네이버 지식백과 Crawling 데이터

Title	Context	Summary
프레디 머큐리	1946년 탄자니아 잔지바르에서 태어났다. 어린 시절 인도로 이주해 학교를 다녔고, 12세 때 스쿨 밴드 헥틱스(The Hectics)를 결성해 클리프 리처드(Cliff Richard), 리틀 리처드(Little Richard)등 로큰롤 가수들의 노래를 커버했다. 1964년 영국으로 이주했고, 1960년대 후반 아이벡스(Ibex), 사우워 밀크 시(Sour Milk Sea) 등의 그룹에서 활동했다.	프레디 머큐리는 영국의 록 보컬리스트. 프레디 머큐리는 1946년 탄자니아 잔지바르에서 태어남. 프레디 머큐리는 1960년대 후반 아이벡스(Ibex), 사우워 밀크 시(Sour Milk Sea) 등의 그룹에서 활동함. 프레디 머큐리는 1985년 첫 번째 솔로 앨범 《미스터 배드 가이》(1985년 영국 앨범차트 6위), 1988년 스페인 출신의 소프라노 몽세라 카바예(Montserrat Caballé)와 함께 만든 앨범 《바르셀로나(Barcelona)》(1988년 영국 앨범차트 15위)를 연이어 발표함.
윤동주	만주 북간도의 명동촌(明東村)에서 태어났으며, 기독교인인 할아버지의 영향을 받았다. 본관은 파평(坡平)이며, 아버지는 윤영석(尹永錫), 어머니는 김룡(金龍)이다. 1931년(14세)에 명동(明東)소학교를 졸업하고, 한 때 중국인 관립학교인 대랍자(大拉子) 학교를 다니다 가족이 용정으로 이사하자 용정에 있는 은진(恩眞)중학교에 입학하였다(1933).1935년에 평양의 숭실(崇實)중학교로 전학하였으나, 학교에 신사참배 문제가 발생하여 폐쇄당하고 말았다.	윤동주는 기독교인인 할아버지의 영향을 받음. 윤동주는 1931년(14세)에 명동소학교를 졸업함. 윤동주는 다시 용정에 있는 광명학원의 중학부로 편입하여 거기서 졸업함. 윤동주는 1941년에는 서울의 연희전문학교 문과를 졸업함. 윤동주는 학업 도중 귀향하려던 시점에 항일운동을 했다는 혐의로 일본 경찰에 체포됨. 윤동주는 그의 시집은 본인이 직접 발간하지 못함.

데이터 전처리 과정

Tokenizing

Twitter/Komoran/Kkma 중 모델에 적합한 Twitter 사용

Regular Expression

불용어가 많아 분석에 있어 필수적인 '~', '.', 한글, 영어, 숫자만을 추출함

Embedding

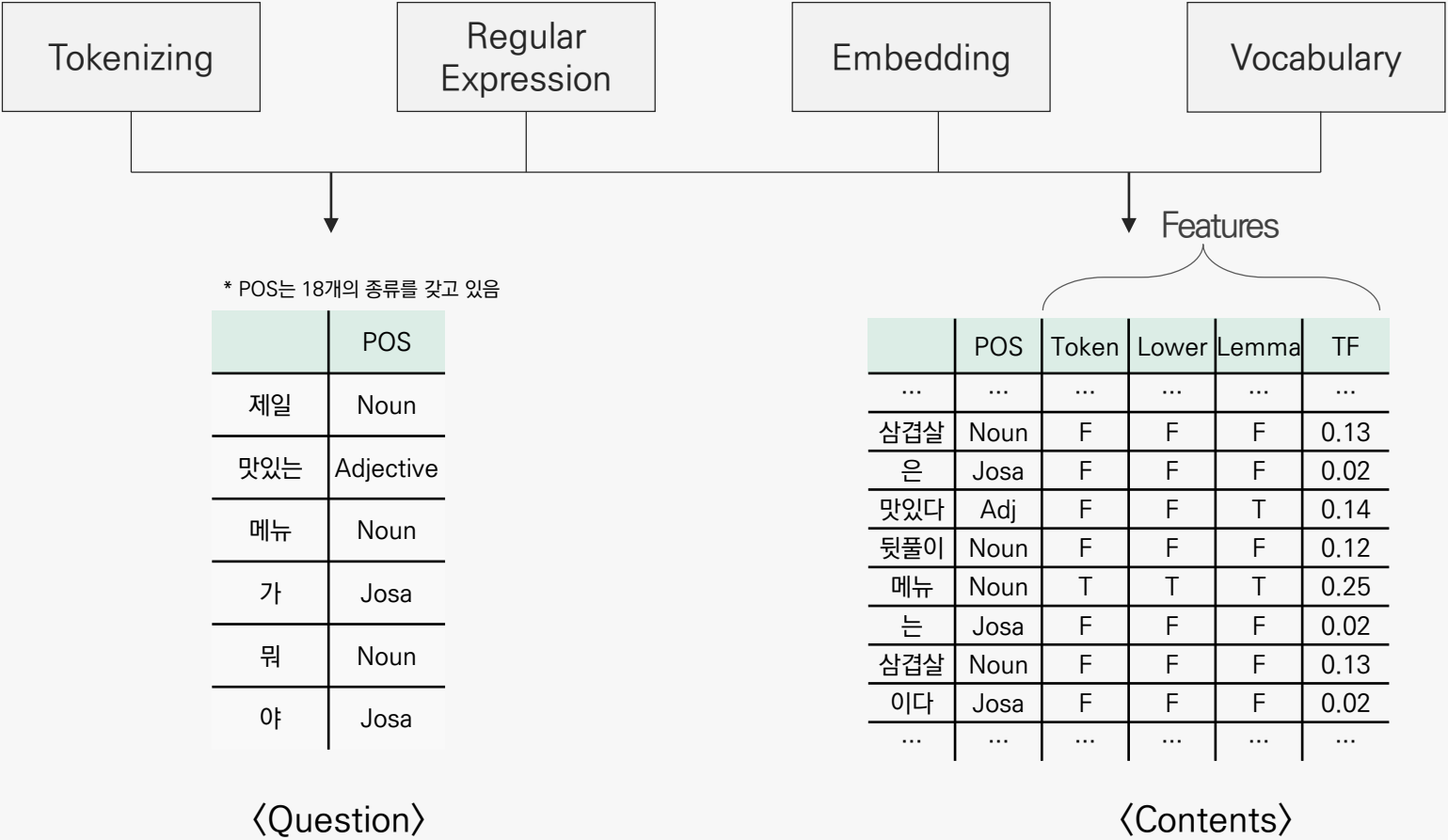
200차원, CBOW에 비해 300차원, Skipgram의 성능이 우수하였음

Vocabulary

...	Unk	Pad	배	가	고프	다	...
...	0	1	500	4056	331	6948	...

데이터 전처리 결과

* 이후 자주 나오는 질문 토큰 벡터에 대해서는 Fine tune



Paper Summary – A Structured Self-Attentive Sentence Embedding (2017)

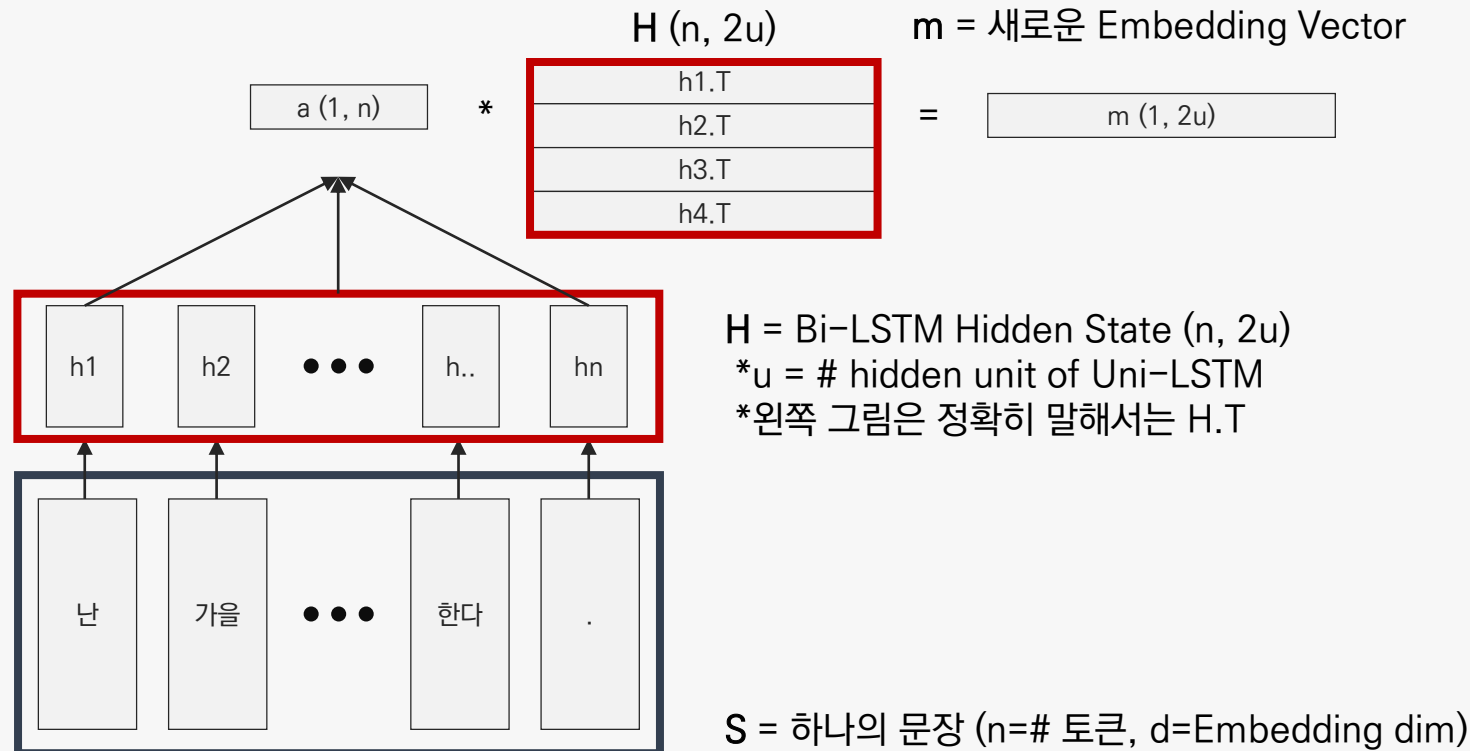
Abstract

본 논문은 self-attention 개념을 도입하여 해석 가능한 Sentence Embedding을 추출하기 위한 새로운 모델을 제시한다. 2차원 행렬(M)로 Embedding을 표현하며, 각 행은 한 문장의 다른 부분에 주목하는 역할을 수행하게 된다.

Summary

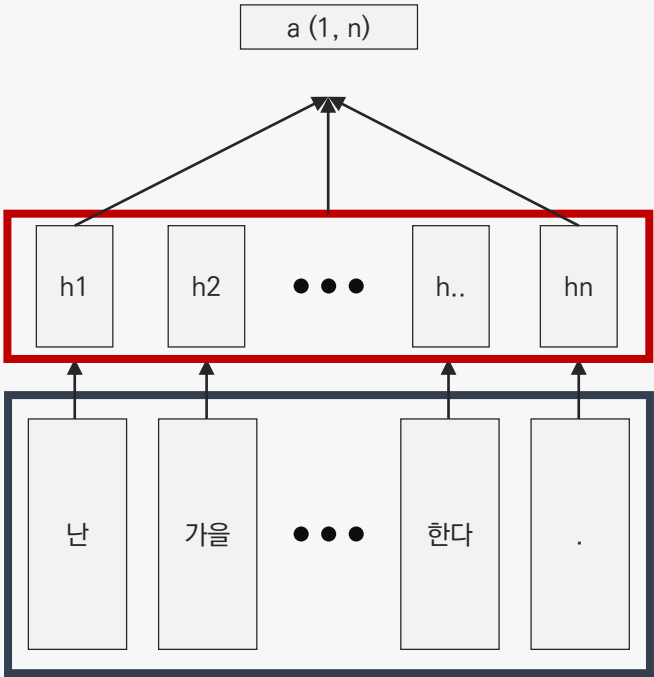
Task	Author Profiling, Sentiment Classification, Textual Entailment
Architecture	Bidirectional-LSTM + Self-Attention
Points	복수의 attention output 반환, Penalization Term 사용

Hidden State Matrix(H)는 각 구성 원소의 강조 포인트를 잡아(a) 자기자신(H)과 곱하여 새로운 Embedding 벡터인 m을 생성함



Hidden State Matrix(H)는 총 n개의 hidden state를 수직으로 쌓은
단방향 hidden state matrix를 수평으로 Stack한 행렬임

Hidden State Matrix 설명



$H(n, 2u)$

h1.T	h1.T
h2.T	h2.T
h3.T	h3.T
h4.T	h4.T

$\vec{h}_t(n, u)$ $\overleftarrow{h}_t(n, u)$

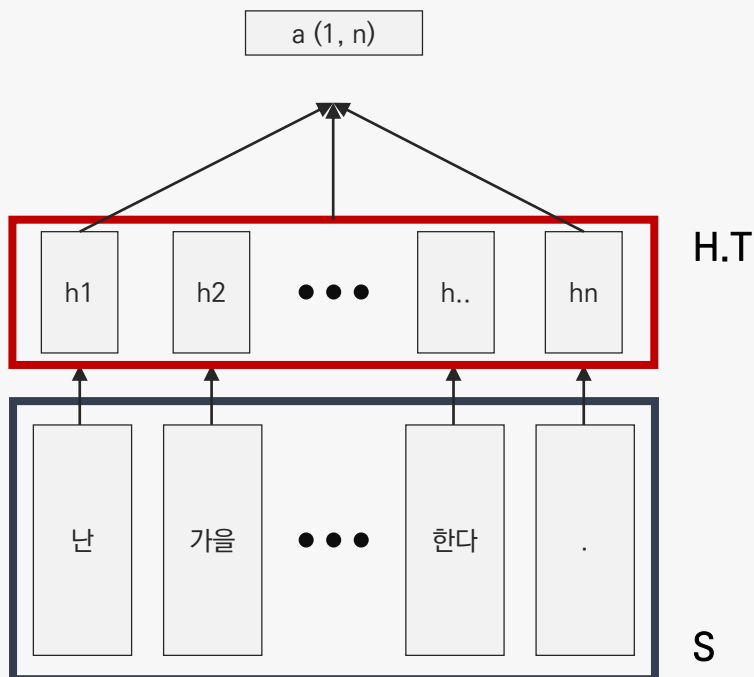
$$\vec{h}_t = LSTM(w_t, \vec{h}_{t-1}) \tag{2}$$
$$\overleftarrow{h}_t = \overleftarrow{LSTM}(w_t, \overleftarrow{h}_{t+1}) \tag{3}$$

And we concatenate each \vec{h}_t with \overleftarrow{h}_t to obtain a hidden state h_t . Let the hidden unit number for each unidirectional LSTM be u . For simplicity, we note all the n h_t s as H , who have the size n -by- $2u$.

$$H = (h_1, h_2, \dots h_n) \tag{4}$$

Hidden State Matrix(H)에서 중요한 근거 단어를 찾아내는 가중치 벡터 a는 아래와 같이 계산됨

가중치 벡터 a 설명



$$a = \underset{(1, n)}{\text{softmax}} \left(\underset{(1, d_a)}{w_{s2}} \underset{(d_a, 2u)}{\tanh} \left(\underset{(2u, n)}{W_{s1} H^T} \right) \right)$$

H를 Input으로 놓고 H 내부에 있는 n개의 hidden state vector(h_t)의 선형결합을 취한 뒤, tanh 함수를 통과시킨다.

W 벡터와 행렬은 모두 학습되는 가중치 행렬이다.

Softmax함수를 통과하여 a는 원소 총합이 1인 이산형 확률분포를 갖는 벡터가 된다.

이 a는 H에서 중요한 단어에 높은 가중치를 부여하는 (강조 역할을 하는) 가중치 벡터이다.

Input sentence에 대한 새로운 표현인 m 벡터는 한 번에 한 요소에만 집중할 수 있는 한계를 가지기 때문에, 여러 개의 m 벡터가 필요함

New Embedding 벡터 m 설명

$$\begin{array}{c} \mathbf{a} (1, n) \\ \boxed{0.0 \ 0.0 \ \mathbf{0.9} \ 0.1} \end{array} * \begin{array}{c} \mathbf{H} (n, 2u) \\ \begin{array}{|c|} \hline h1.T \\ \hline h2.T \\ \hline \mathbf{h3.T} \\ \hline h4.T \\ \hline \end{array} \end{array} = \begin{array}{c} \mathbf{m} = \text{새로운 Embedding Vector} \\ \boxed{\mathbf{m} (1, 2u)} \end{array}$$

가중치 벡터 a와 기존의 Hidden State Matrix H의 곱으로 생성된 새로운 Embedding Vector m은 input sentence에 대한 새로운 Representation이다.

이 벡터 m은 문장의 특정 구성 요소에만 집중한 형태이다.
(가장 중요한 정보, 근거 만을 집중적으로 담고 있다.)

그러나 한 문장 내에 중요한 단어는 단 1개가 아니라 여러 개일 수 있기 때문에
한 문장의 여러 부분에 집중할 수 있는 복수의 m이 필요하다.

복수의 m을 vertical하게 쌓은 M 행렬은 강조 포인트의 위치가 반영된 새로운 Embedding 행렬임

New Embedding 행렬 M 설명

$$A = \underset{(r, n)}{\text{softmax}} \left(\underset{(r, d_a)}{W_{s2}} \underset{(d_a, 2u)}{\tanh \left(\underset{(2u, n)}{W_{s1} H^T} \right)} \right) \quad * \text{ 총 } r \text{ 번 시행하여 } r \text{ 개의 } m \text{ 을 얻는 작업임}$$

A (r, n)				
0.0	0.0	0.9	0.1	
0.0	0.0	0.1	0.9	
0.7	0.2	0.0	0.1	
0.1	0.8	0.1	0.0	

*

H (n, 2u)
h1.T
h2.T
h3.T
h4.T

=

M (r, 2u)
m (1, 2u)
m (1, 2u)
m (1, 2u)
m (1, 2u)

가중치 행렬(annotation matrix) A와 LSTM hidden State Matrix H를 곱한
행렬인 M은 강조 포인트가 덧칠해진 새로운 Embedding 행렬임

이 M 행렬은 기존의 H를 대체하여 이후에 효과적인 Project 진행을 가능하게 함

페널티를 주어야만 A행렬을 구성할 때 행벡터들이 중복되지 않고 Input 문장 내에서 다양한 부분을 고려할 수 있음

New Embedding 행렬 M 설명

$$P = \| (AA^T - I) \|_F^2$$

(r, r) (r, n), (n, r) (r, r)

0.0	0.0	0.9	0.1
0.0	0.0	0.1	0.9
0.7	0.2	0.0	0.1
0.1	0.8	0.1	0.0

*

0.0	0.0	0.7	0.1
0.0	0.0	0.2	0.8
0.9	0.1	0.0	0.1
0.1	0.9	0.1	0.0

=

0.81	0.18	0.01	0.09
0.18	0.81	0.09	0.01
0.01	0.09	0.54	0.23
0.09	0.01	0.23	0.66

대각원소는 1에 가깝게 한다.
I의 대각원소인 1을 빼서 0으로 만들어준다.
비 대각원소는 0에 가깝게 한다.

Frobenius norm을 사용하여 새로운 Penalization Term을 생성함
이 P를 최소화하는 것이 목표임

A 행렬의 각 행 a_i 와 a_j 가 유사할 수록 P는 증가하므로
각 시행에서 나온 결과물인 a 벡터는 서로 다를 수록 좋음
(각각 문장의 다른 부분을 pinpoint하는 것이 좋음)

```

35 self.logger = logging.getLogger(__name__)
36
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'), 'a')
39     self.file.seek(0)
40     self.fingerprints.update(requests_log)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('DEBUG', False)
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)

```

II. Document Retriever 설명



관련 정보 글 찾기 + 질문에 대한 답 찾기
Document Retriever + Question & Answering Model

사용자의 질문이 주어졌을 때, 정보 글(Document)과의 유사도 계산을 통해 가장 관련성이 높은 (유사도가 높은) 정보 글들을 출력하는 것이 Document Retriever의 역할임

질문, 정보 글 예시

유사도 계산

질문

카보베르데는 누가 발견했는가?

정보 글

카보베르데 공화국(카보베르데共和國,), 줄여서 카보베르데는 아프리카 서쪽 대서양에 있는 국가이다. 영어로 케이프베르데라는 이름으로도 알려져 있다. 포르투갈의 항해자들에 의해 발견되었다. ...

$$W_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

- TF-IDF: Term x within document y
- TF-IDF 기반의 코사인 유사도를 사용함
- 벡터 공간에서 각도를 보고 유사도를 판단하는 수치임
- 문서의 길이에 상관없이 정규화 가능

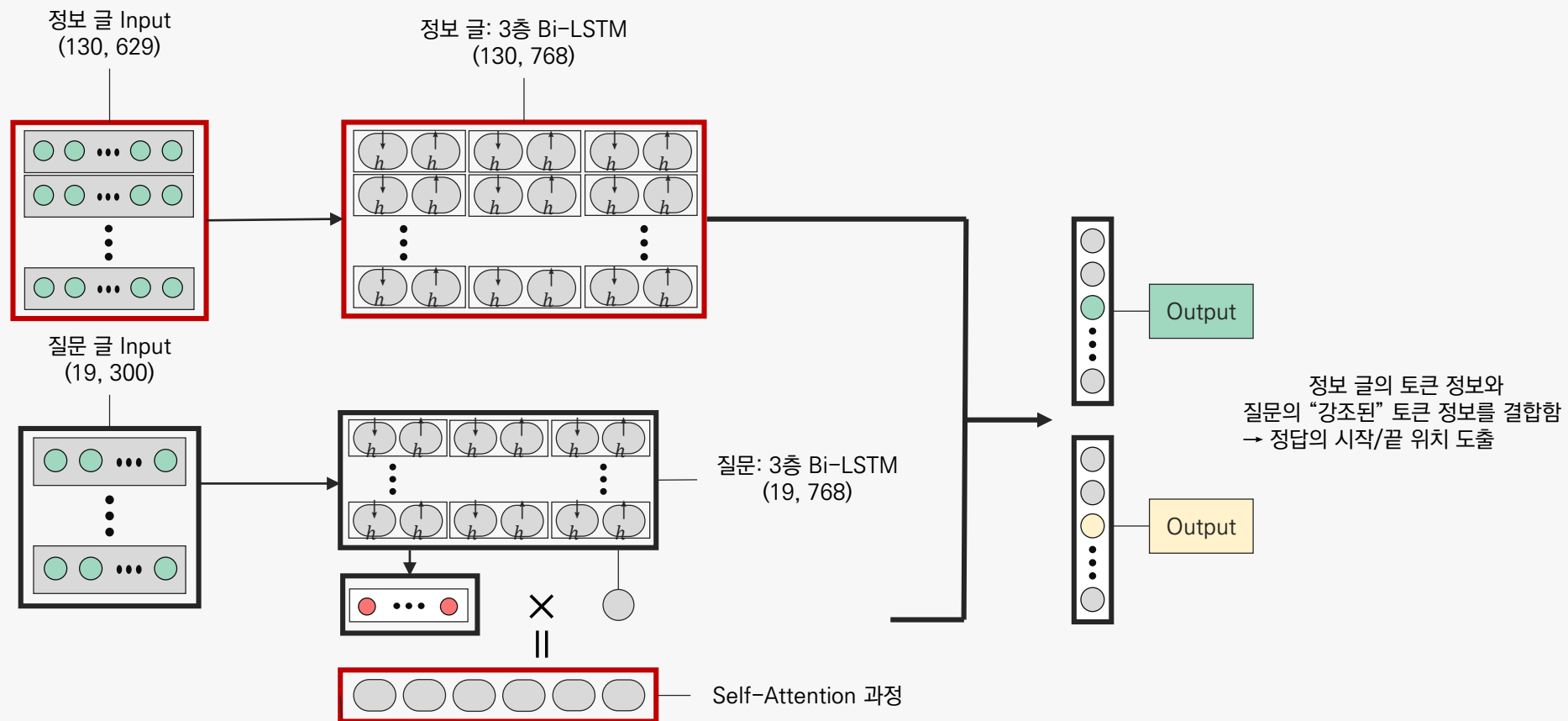
```

35 self.logger = logging.getLogger(__name__)
36
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'), 'a')
39     self.file.seek(0)
40     self.fingerprints.update(requests_log)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('SUPERFUTUR_DEBUG')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)

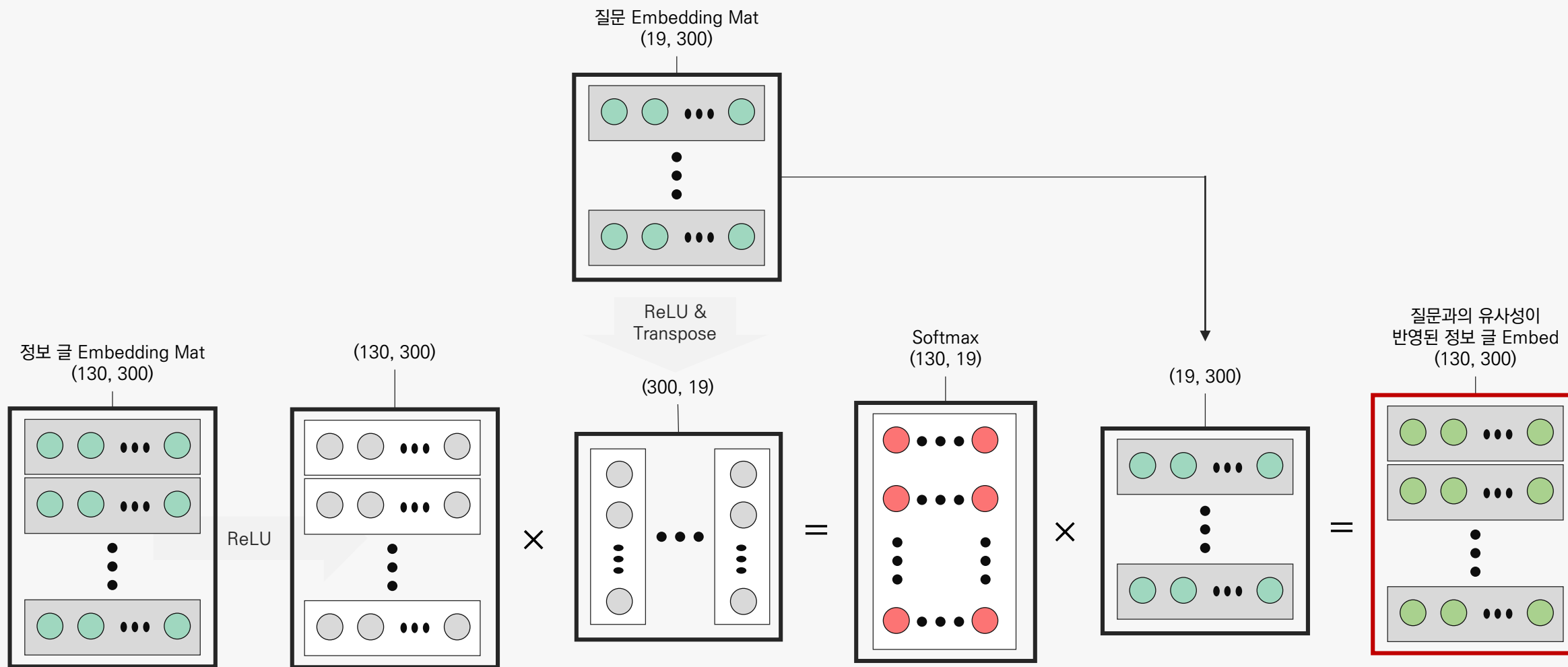
```

III. QA 모델 설명

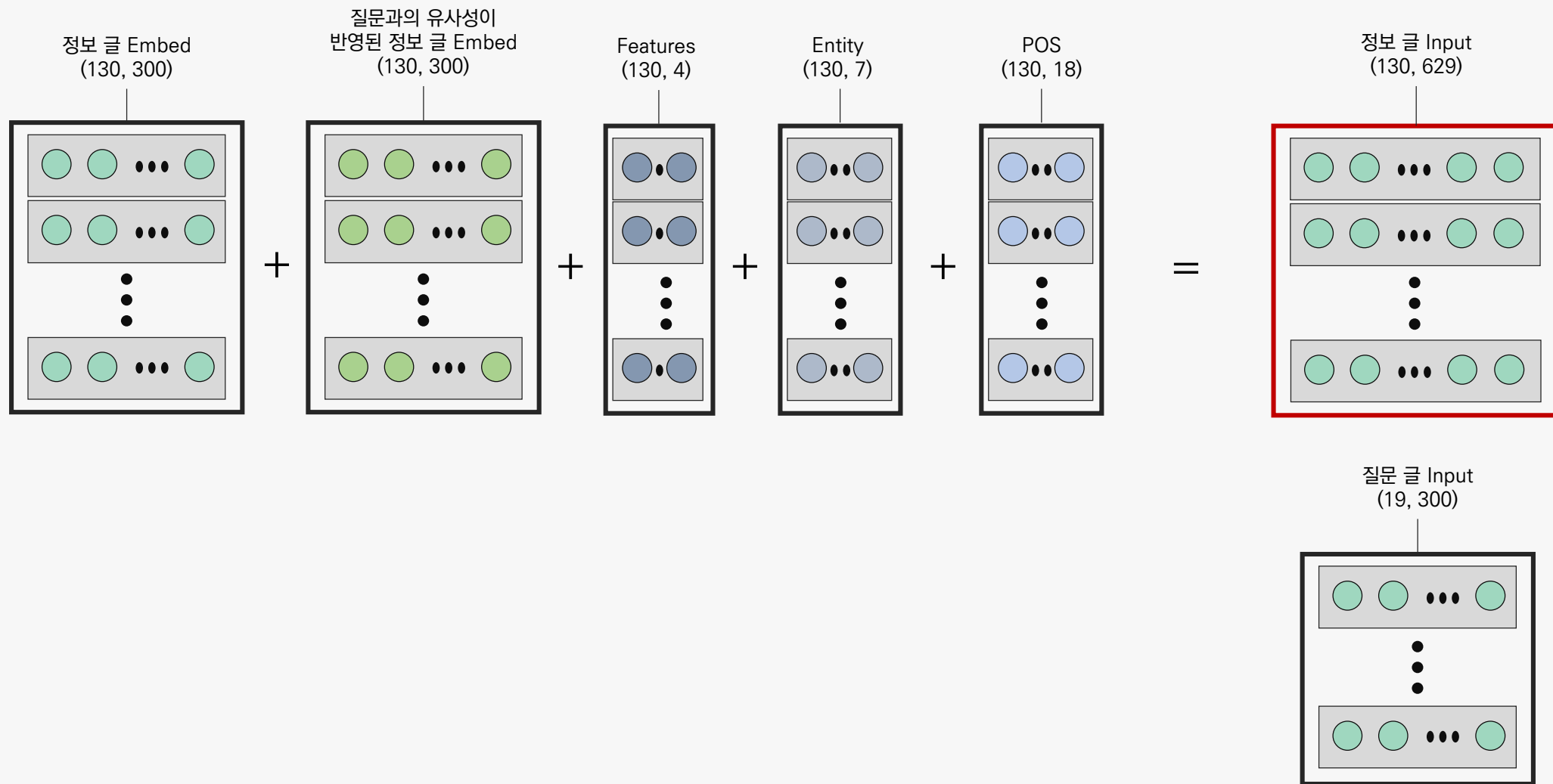
QA 모델 전체 구조



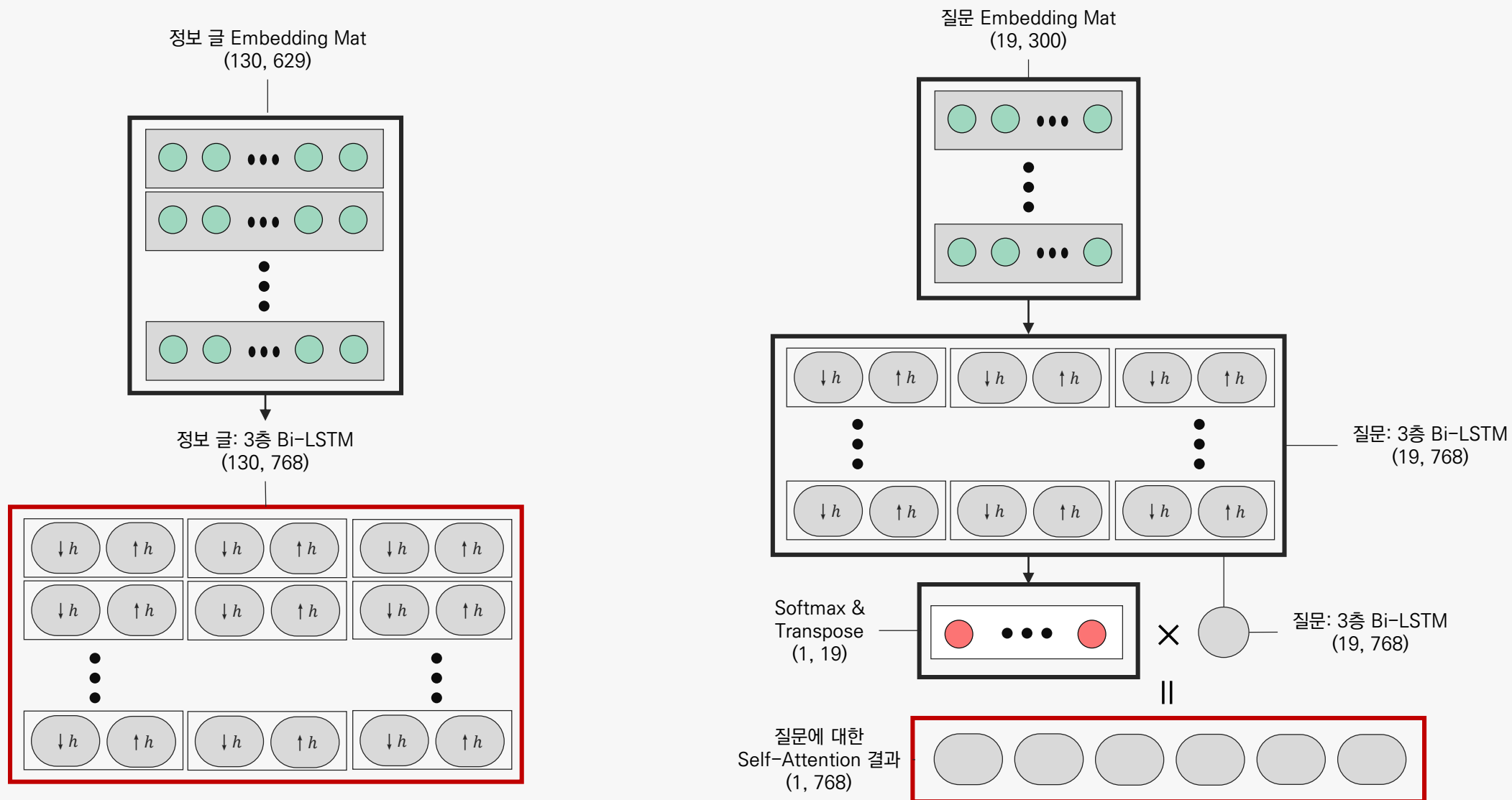
Input Data 준비 과정 (1)



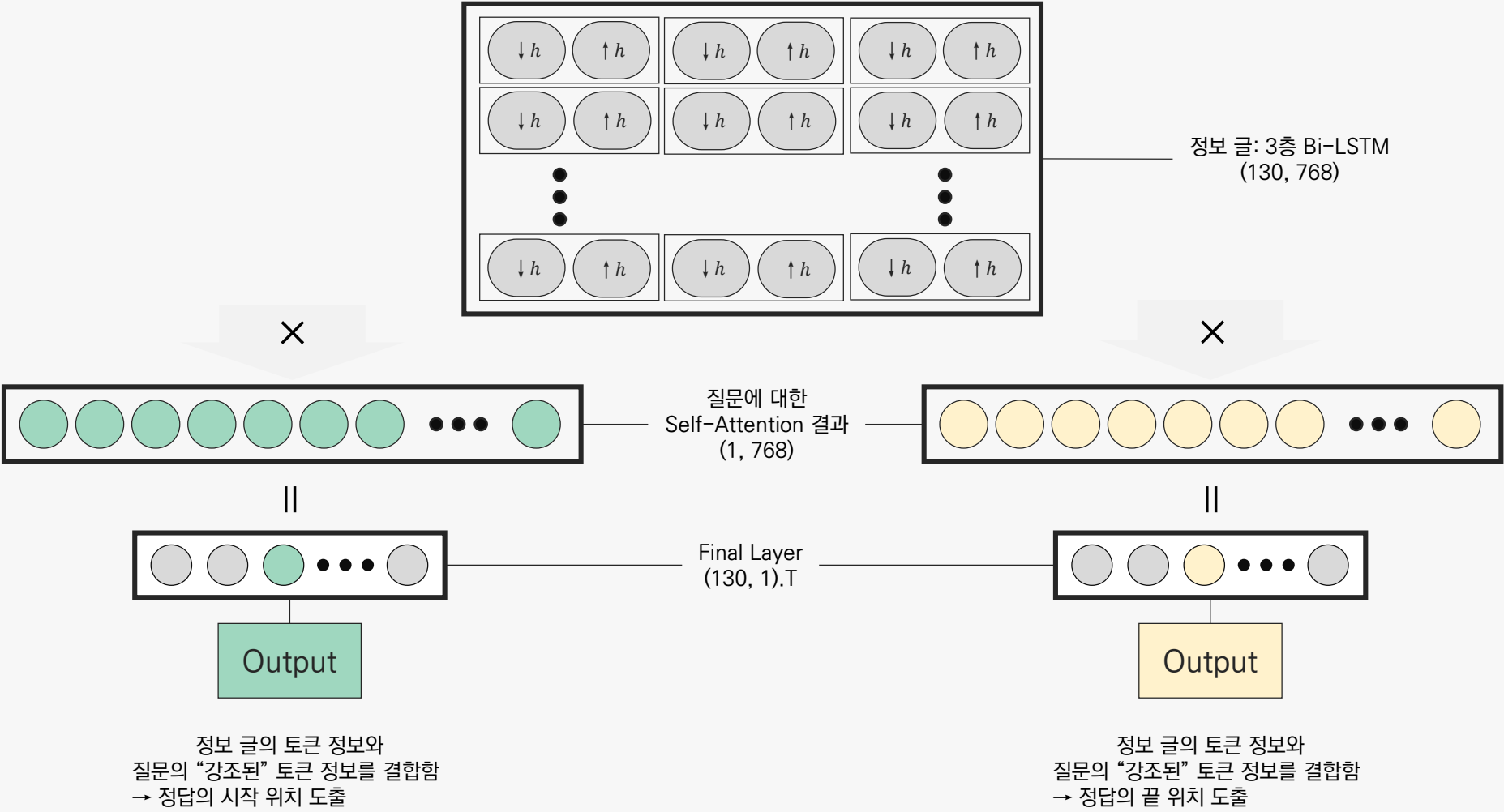
Input Data 준비 과정 (2)



Main Process



Output 출력



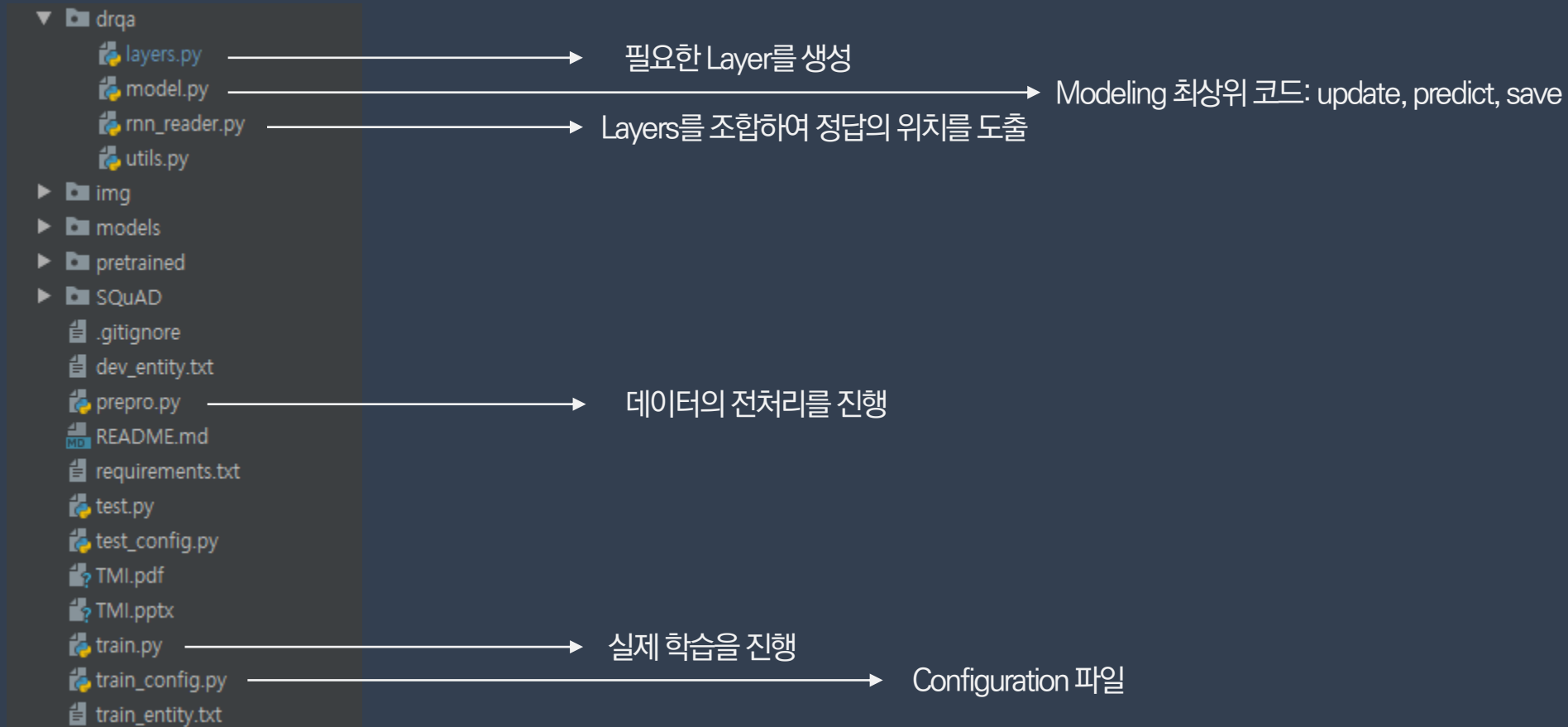

```

35 self.logger = logging.getLogger(__name__)
36
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'), 'a')
39     self.file.seek(0)
40     self.fingerprints.update(requests_log)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('DEBUG', False)
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)

```

IV. Code Review

DrQA 전체 프로젝트 구조



StackedBRNN: 질문, 정보 글 3층 BiLSTM

```
class StackedBRNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers,
                 dropout_rate=0, dropout_output=False,
                 rnn_type=nn.LSTM,
                 concat_layers=False, padding=False):
        super(StackedBRNN, self).__init__()
        self.padding = padding
        self.dropout_output = dropout_output
        self.dropout_rate = dropout_rate
        self.num_layers = num_layers
        self.concat_layers = concat_layers

    # 아래 rnns에 layers가 쌓이게 된다.
    self.rnns = nn.ModuleList()
    for i in range(num_layers):
        input_size = input_size if i == 0 else 2 * hidden_size
        self.rnns.append(rnn_type(input_size, hidden_size,
                                   num_layers=1,
                                   bidirectional=True))
```

```
def forward(self, x, x_mask):
    # 패딩이 필요 없는 경우
    if x_mask.data.sum() == 0:
        return self._forward_unpadded(x, x_mask)
    # 패딩을 적용하였거나, Validation Set을 다루는 경우
    if self.padding or not self.training:
        return self._forward_padded(x, x_mask)
    # 그 외의 경우
    return self._forward_unpadded(x, x_mask)
```

QA 모델 상의 Main Process를 담당하는 부분임

StackedBRNN 인스턴스 생성에 필요한 매개변수는 아래와 같이 rnn_reader.py에 저장됨

```
# RNN document encoder
self.doc_rnn = layers.StackedBRNN(
    input_size=doc_input_size,
    hidden_size=opt['hidden_size'],
    num_layers=opt['doc_layers'],
    dropout_rate=opt['dropout_rnn'],
    dropout_output=opt['dropout_rnn_output'],
    concat_layers=opt['concat_rnn_layers'],
    rnn_type=self.RNN_TYPES[opt['rnn_type']],
    padding=opt['rnn_padding'],
)

# RNN question encoder
self.question_rnn = layers.StackedBRNN(
    input_size=opt['embedding_dim'],
    hidden_size=opt['hidden_size'],
    num_layers=opt['question_layers'],
    dropout_rate=opt['dropout_rnn'],
    dropout_output=opt['dropout_rnn_output'],
    concat_layers=opt['concat_rnn_layers'],
    rnn_type=self.RNN_TYPES[opt['rnn_type']],
    padding=opt['rnn_padding'],
)
```

LinearSeqAttn: Linear Self-Attention 적용

```
class LinearSeqAttn(nn.Module):
    """Self attention over a sequence:
    *  $o_i = \text{softmax}(Wx_i)$  for  $x_i$  in  $X$ .
    """
    def __init__(self, input_size):
        super(LinearSeqAttn, self).__init__()
        self.linear = nn.Linear(input_size, 1)

    def forward(self, x, x_mask):
        """
         $x = \text{batch} * \text{len} * \text{hdim}$ 
         $x\_mask = \text{batch} * \text{len}$ 
        """
        x_flat = x.contiguous().view(-1, x.size(-1))
        scores = self.linear(x_flat).view(x.size(0), x.size(1))
        scores.data.masked_fill_(x_mask.data, -float('inf'))
        alpha = F.softmax(scores, dim=1)
        return alpha
```

1차적으로 질문에 대한 Self-Attention 결과를
도출하기 위해 사용되는 코드임


아래 Code를 통해 “질문에 대한 Attention 결과”를 도출함

rnn_reader.py

```
# Encode question with RNN + merge hiddens
question_hiddens = self.question_rnn(x2_emb, x2_mask)
if self.opt['question_merge'] == 'avg':
    q_merge_weights = layers.uniform_weights(question_hiddens, x2_mask)
elif self.opt['question_merge'] == 'self_attn':
    q_merge_weights = self.self_attn(question_hiddens, x2_mask)
question_hidden = layers.weighted_avg(question_hiddens, q_merge_weights)
```

layers.py

```
def weighted_avg(x, weights):
    """x = batch * len * d
    weights = batch * len
    """
    return weights.unsqueeze(1).bmm(x).squeeze(1)
```



아래는 정답의 시작/끝 위치를 도출하기 위한 준비 과정임

rnn_reader.py

```
# Start, End 포인트를 생성하기 위한 Bilinear Attention
self.start_attn = layers.BilinearSeqAttn(
    doc_hidden_size,
    question_hidden_size,
)
self.end_attn = layers.BilinearSeqAttn(
    doc_hidden_size,
    question_hidden_size,
)
```

layers.py

```
class BilinearSeqAttn(nn.Module):
    def __init__(self, x_size, y_size, identity=False):
        super(BilinearSeqAttn, self).__init__()
        if not identity:
            self.linear = nn.Linear(y_size, x_size)
        else:
            self.linear = None

    def forward(self, x, y, x_mask):

        Wy = self.linear(y) \n
        if self.linear is not None else y
        xWy = x.bmm(Wy.unsqueeze(2)).squeeze(2)
        xWy.data.masked_fill_(x_mask.data, -float('inf'))
        if self.training:
            alpha = F.log_softmax(xWy, dim=1)
        else:
            alpha = F.softmax(xWy, dim=1)
        return alpha
```

정보 글의 토큰 정보와
질문의 “강조된” 토큰 정보를 결합하여 결과를 도출함

아래와 같이 정답의 시작/끝 위치를 도출함

rnn_reader.py

```
start_scores =
    self.start_attn(doc_hiddens, question_hidden, x1_mask)
end_scores =
    self.end_attn(doc_hiddens, question_hidden, x1_mask)
return start_scores, end_scores
```

layers.py

```
def forward(self, x, y, x_mask):
    """
    x = batch * len * h1
    y = batch * h2
    x_mask = batch * len
    """
    Wy = self.linear(y) if self.linear is not None else y
    xWy = x.bmm(Wy.unsqueeze(2)).squeeze(2)
    xWy.data.masked_fill_(x_mask.data, -float('inf'))
    if self.training:
        # In training we output log-softmax for NLL
        alpha = F.log_softmax(xWy, dim=1)
    else:
        # ...Otherwise 0-1 probabilities
        alpha = F.softmax(xWy, dim=1)
    return alpha
```


model.py 내 DocReaderModel에서 최종적으로 업데이트가 이루어짐

```
def update(self, ex):
    self.network.train()      # 학습 모드

    # GPU로 넘김
    inputs = [e.to(self.device) for e in ex[:7]]
    target_s, target_e = ex[7].to(self.device), ex[8].to(self.device)

    # Run forward
    score_s, score_e = self.network(*inputs)

    # Loss와 Accuracy를 계산함
    loss = F.nll_loss(score_s, target_s) + F.nll_loss(score_e, target_e)
    self.train_loss.update(loss.item())

    # Gradients를 초기화 하고 Run Backward / Clip Gradients
    self.optimizer.zero_grad()
    loss.backward()

    torch.nn.utils.clip_grad_norm_(self.network.parameters(), self.opt['grad_clipping'])

    # 파라미터 업데이트
    self.optimizer.step()
    self.updates += 1
```

End of Document