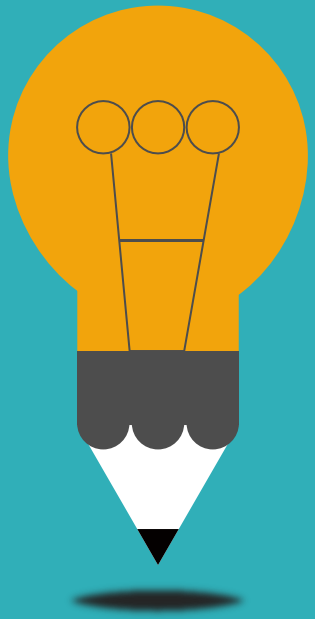
2018 Fall Text Seminar

Attention

[NMT by Jointly Learning to Align and Translate, 2015],
[A Structured Self-Attentive Sentence Embedding, 2017]



장유영

CONTENTS

Paper Summary

Abstract

본 논문은 타겟 단어를 예측하는 과정에 있어, 모델이 자동적으로 source sentence를 Search하는 과정을 통해 예측력을 향상시키는 방법에 대해 논의한다. (Basic Attention)

Summary

Task	영어 – 프랑스어 번역
Architecture	Bidirectional-RNN + Basic Attention
Results	기존의 Basic 인코더-디코더 모형의 한계를 극복함

Background

기존 인코더-디코더 모델의 한계점

기존 모델은 Source sentenc의 모든 정보를 fixed-length 벡터로 압축하는 방식을 취하는데,
이는 학습 데이터에 있었던 문장보다 **길이가 긴 문장을 만났을 때 모델이 대처하기가 어렵게**
만드는 요인이 되며, 긴 Input 문장을 만났을 때 성능 자체가 저하되는 문제점을 유발하였다.

본 논문에서 제시한 모델

Input 문장을 a sequence of vectors로 인코딩하고,
번역할 때(타겟 단어를 예측할 때) 이러한 벡터들 중 일부를 취함으로써
일괄적으로 문장 길이를 무시하여 모든 정보를 fixed-length 벡터로 만들지 않는다.

Basic RNN Encoder-Decoder 모델

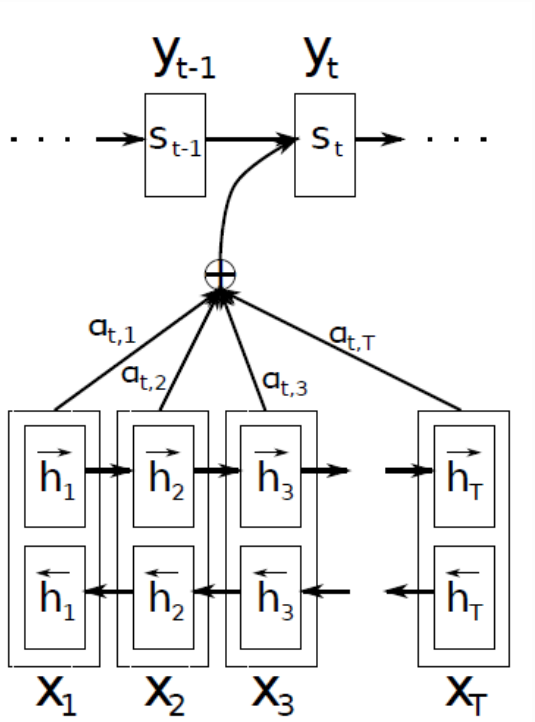
구성 요소 정리

Input Sentence	$\mathbf{x} = (x_1, \cdots, x_{T_x})$	(d_x, T_x)
Hidden State	$h_t = f(x_t, h_{t-1})$	(h, 1)
Context Vector	$c = q(\{h_1, \cdots, h_{T_x}\})$	(d_x, T_x)
Output Sentence	$\mathbf{y} = (y_1, \cdots, y_{T_y})$	(d_y, T_y)
Each Prediction	$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$	
Overall Predicting	$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c)$	

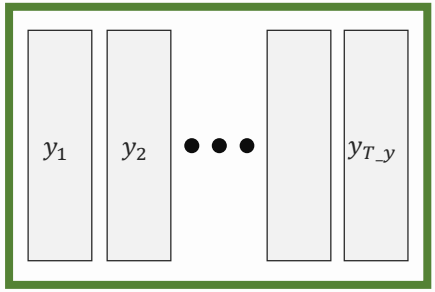
f, q, g는 비선형 함수

디코더 Hidden State

모델 Architecture – Input, Output Dimension



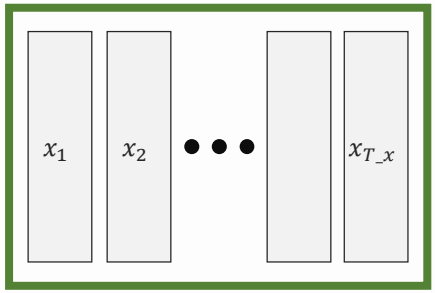
Y



k_y = target 언어의 vocab_size

T_y = target 문장의 길이

X



k_x = source 언어의 vocab_size

T_x = source 문장의 길이

모델 Architecture – Context Vector 계산 법

학습 parameters

Attention Score; 스칼라

$$e_{ij} = a(s_{i-1}, h_j) = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

(1, n`) (n`, n)(n, 1) (n`, 2n) (2n, 1)

Attention Weight; 확률

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

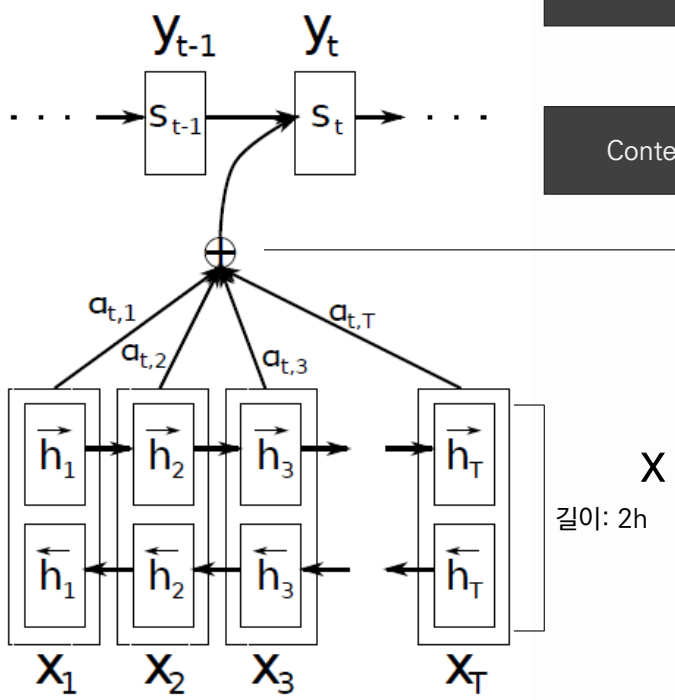
s_{i-1}과 h_j의 관련성

Context Vector; (2h, 1) 벡터

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

weighted sum

* 여기서 i = t



X

길이: 2h

$C_t = C_i$: Decoder time i에 영향을 미치는 Context Vector

C_i 는 j번 째 hidden state인 $h_j(j^{th}$ annotation)의 가중 선형 결합이며,
일종의 Expected Annotation with probabilities $\alpha_{i,j}$ 이다.

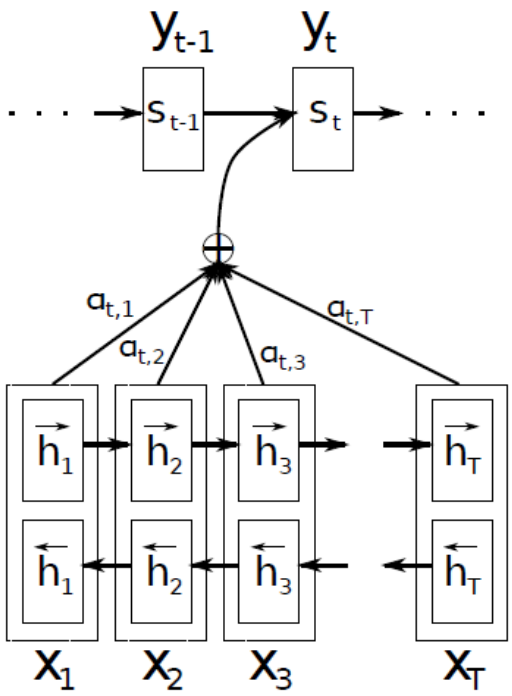
$\alpha_{i,j}$ 는 j^{th} annotation과 i-1번 째까지의 Decoder hidden state의 상관 정도를 나타내며
또한 target word y_i가 source word x_j와 aligned to, translated from될 확률임.
결국 $\alpha_{i,j}$ 는 h_j의 중요성을 결정한다고 볼 수 있다.

2h: Encoder Hidden Unit 개수

n: Decoder Hidden Unit 개수

i = Decoder의 time, j = Encoder의 time

모델 Architecture – Decoder Hidden State s_i



Context Vector; (2h, 1) 벡터

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

Decoder Hidden State; (n, 1) 벡터

GRU와 유사함

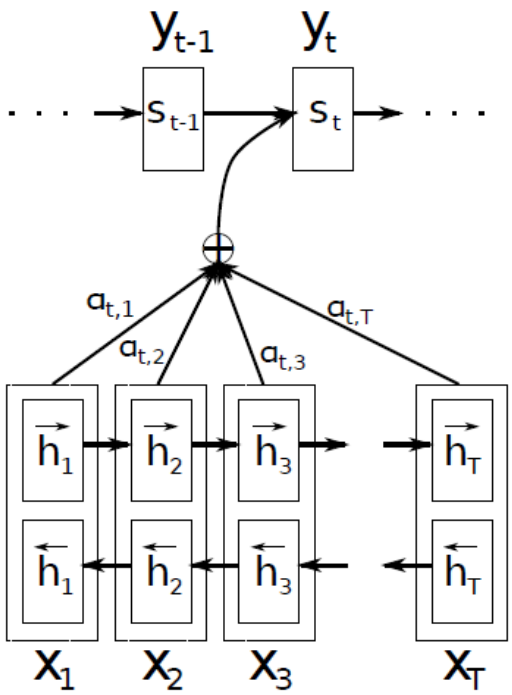
$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$s_i = (1 - z_i) \circ s_{i-1} + z_i \circ \tilde{s}_i,$$

$$\tilde{s}_i = \tanh\left(WEy_{i-1} + U\left[r_i \circ s_{i-1}\right] + Cc_i\right)$$
$$z_i = \sigma\left(W_z Ey_{i-1} + U_z s_{i-1} + C_z c_i\right)$$
$$r_i = \sigma\left(W_r Ey_{i-1} + U_r s_{i-1} + C_r c_i\right)$$

최종적으로 target word y_i는 어떻게 출력되는가?

모델 Architecture – 최종 Output 산출

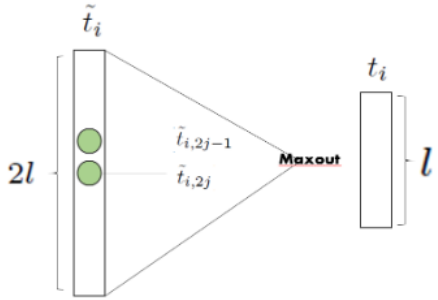


Target Word 등장 확률

$$p(y_i|s_i, y_{i-1}, c_i) \propto \exp\left(y_i^\top W_o t_i\right)$$

$(1, k_y)\ (k_y, l)\ (l, 1)$

이에 비례함



$$t_i = \left[\max\left\{\tilde{t}_{i,2j-1}, \tilde{t}_{i,2j}\right\}\right]_{j=1,\ldots,l}^\top$$

$$\tilde{t}_i = U_o s_{i-1} + V_o E y_{i-1} + C_o c_i.$$

$(2l, n)\ (n, 1)$ $(2l, m)\ (m, 1)$ $(2l, 2m)\ (2n, 1)$

t_i는 Single Maxout Hidden Layer의 Output이며,
j=1일 때의 candidate, j=2일 때의 candidate 중 최대값(max)을 고르는 것처럼
Candidate들을 쌍으로 2개씩 비교하여 최대값들을 모아 구성된다.
이를 이용하여 y_i의 등장 확률을 구할 수 있으며 이 확률을 통해 실제 target word를 출력한다.

최종적으로 target word y_i는 Decoder Hidden State s_i와
이전 target word y_{i-1}, Context Vector C_i를 모두 고려하여 산출된다.

CONTENTS

Paper Summary

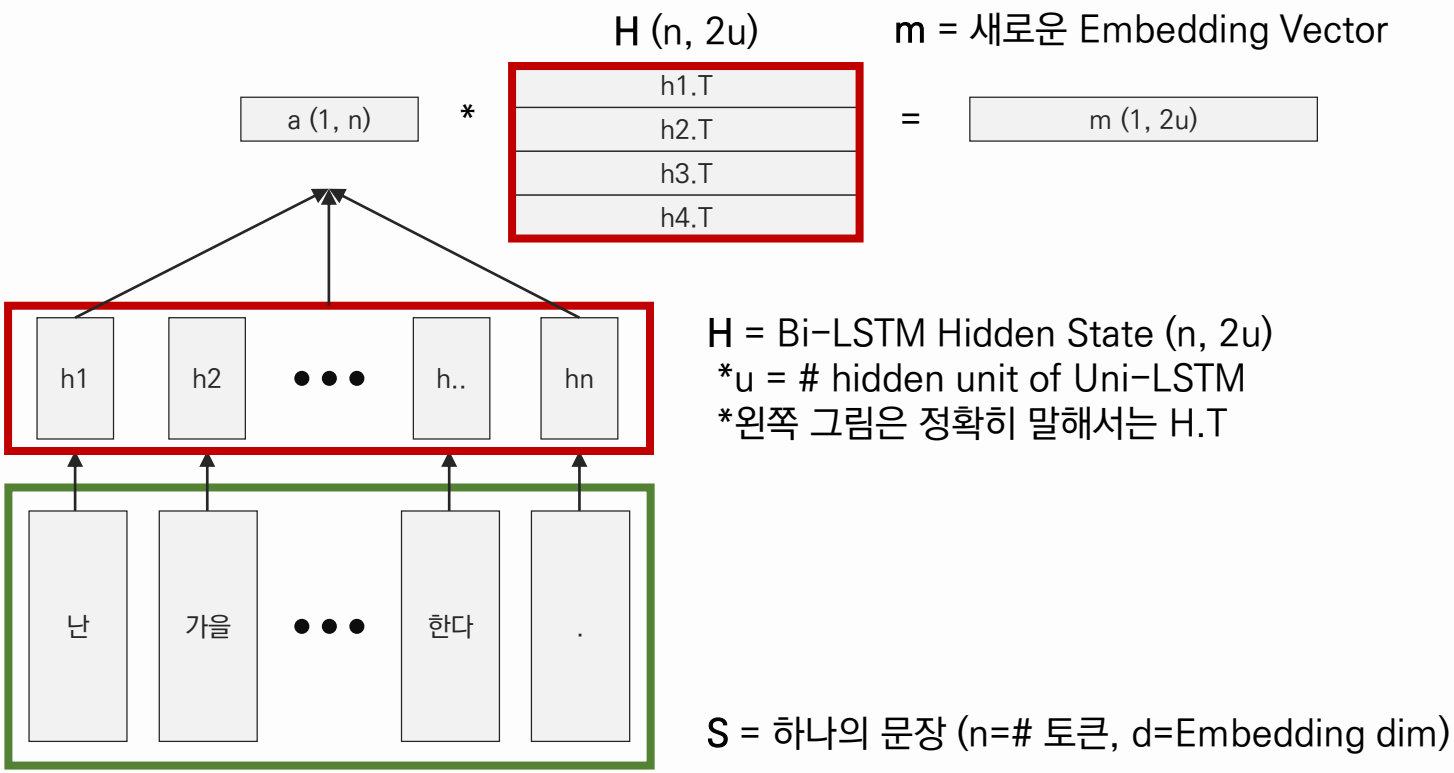
Abstract

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

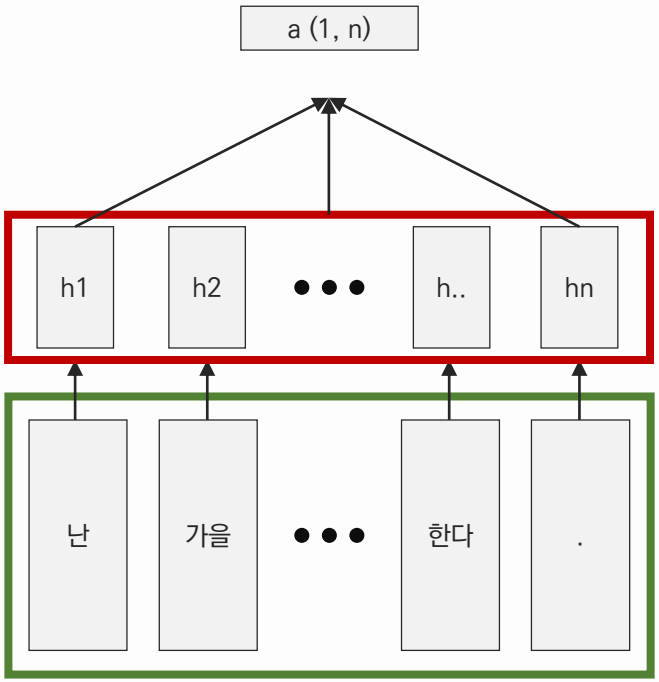


H = Bi-LSTM Hidden State (n, 2u)
*u = # hidden unit of Uni-LSTM
*왼쪽 그림은 정확히 말해서는 H.T

S = 하나의 문장 (n=# 토큰, d=Embedding dim)

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

$\overrightarrow{h_t}$ (n, u) $\overleftarrow{h_t}$ (n, u)

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}}) \quad (2)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t+1}}) \quad (3)$$

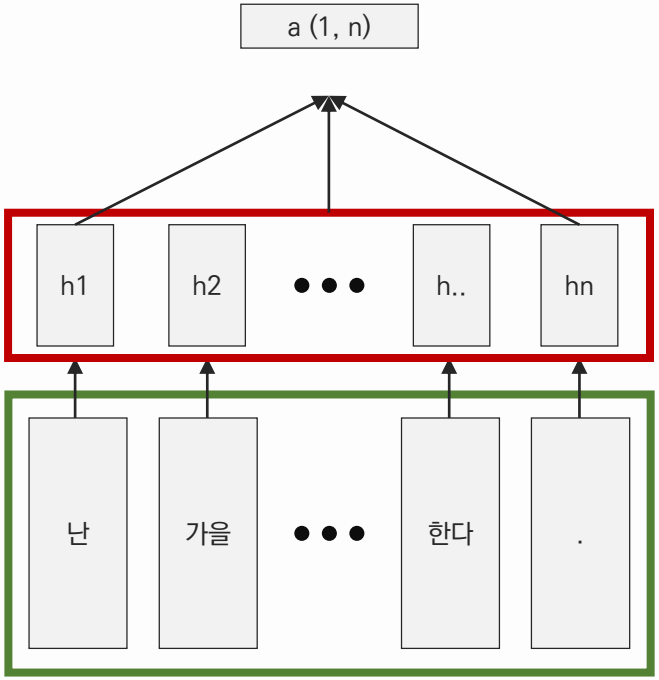
And we concatenate each $\overrightarrow{h_t}$ with $\overleftarrow{h_t}$ to obtain a hidden state h_t. Let the hidden unit number for each unidirectional LSTM be u. For simplicity, we note all the n h_ts as H, who have the size n-by-$2u$.

$$H = (\mathbf{h_1}, \mathbf{h_2}, \cdots \mathbf{h_n}) \quad (4)$$

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

가중치 벡터 a 설명

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



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

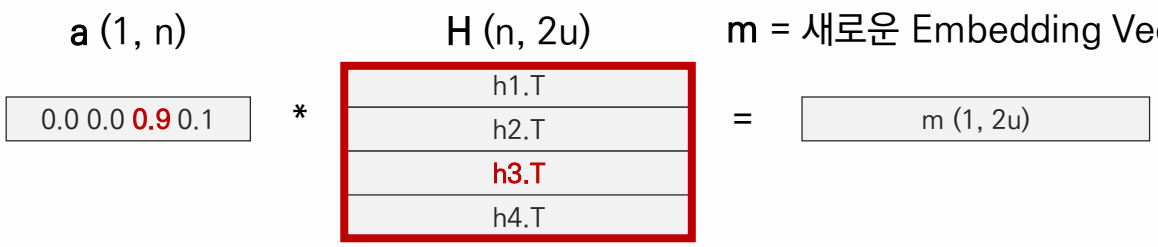
W 벡터와 행렬은 모두 학습되는 가중치 행렬이다.

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

이 a는 H에서 중요한 단어에 높은 가중치를 부여하는 (강조 역할을 하는) **가중치 벡터**이다.

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

New Embedding 벡터 m 설명



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 m**이 필요하다.

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

New Embedding 행렬 M 설명

$$A = softmax\left(W_{s2}tanh\left(W_{s1}H^T\right)\right)$$

(r, n) (r, d_a) (d_a, 2u) (2u, n)

* 총 r번 시행하여 r개의 m을 얻는 작업임

A (r, n)		H (n, 2u)		M (r, 2u)
0.0 0.0 **0.9** 0.1	*	h1.T	=	m (1, 2u)
0.0 0.0 0.1 **0.9**		h2.T		m (1, 2u)
0.7 0.2 0.0 0.1		h3.T		m (1, 2u)
0.1 **0.8** 0.1 0.0		h4.T		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

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

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

A (r, n) A^T (r, n)

0.0	0.0	**0.9**	0.1		0.0	0.0	**0.7**	0.1		**0.81**	0.18	0.01	0.09
0.0	0.0	0.1	**0.9**		0.0	0.0	0.2	**0.8**		0.18	**0.81**	0.09	0.01
0.7	0.2	0.0	0.1	*	**0.9**	0.1	0.0	0.1	=	0.01	0.09	**0.54**	0.23
0.1	**0.8**	0.1	0.0		0.1	**0.9**	0.1	0.0		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_i와 a_j가 유사할 수록 P는 증가하므로
각 시행에서 나온 결과물인 a 벡터는 서로 다를 수록 좋음
(각각 문장의 다른 부분을 pinpoint하는 것이 좋음)

End of Document