어텐션 메커니즘

19조 김두진, 박경민, 유병관

2021년 10월 24일

AGENDA

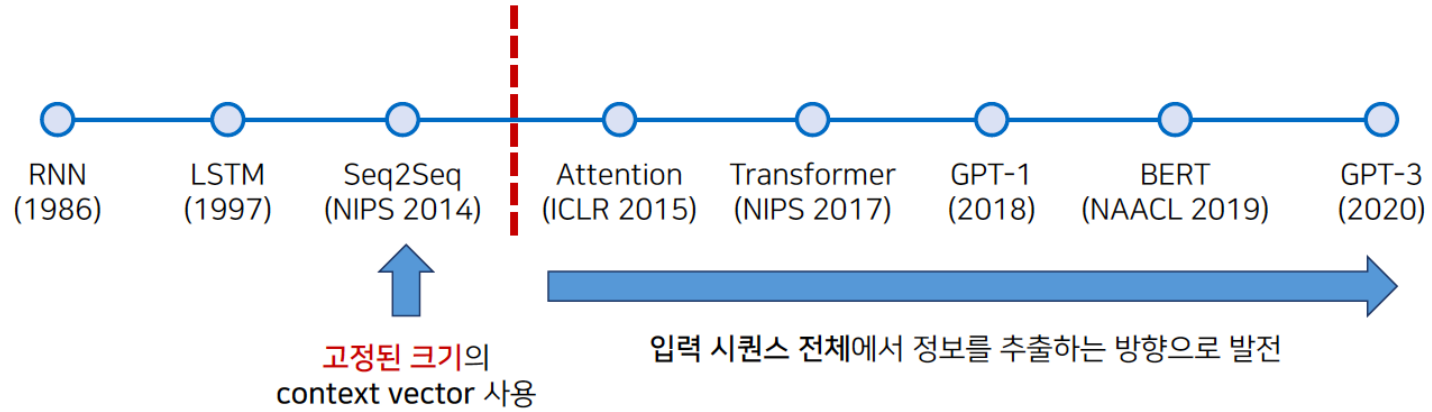
- ❏ 어텐션 소개
- ❏ 닷-프로덕션 어텐션
- ❏ 바다나우 어텐션
- ❏ 양방향 LSTM과 어텐션 메커니즘를 이용한 IMDB 리뷰 감성 분류하기

출처 : https://wikidocs.net/48948

어텐션 소개

딥러닝 기반의 기계 번역 발전 과정

- 2021년 기준으로 최신 고성능 모델들은 Transformer 아키텍처를 기반으로 하고 있습니다.
  - **GPT**: Transformer의 **디코더(Decoder)** 아키텍처를 활용
  - **BERT**: Transformer의 **인코더(Encoder)** 아키텍처를 활용

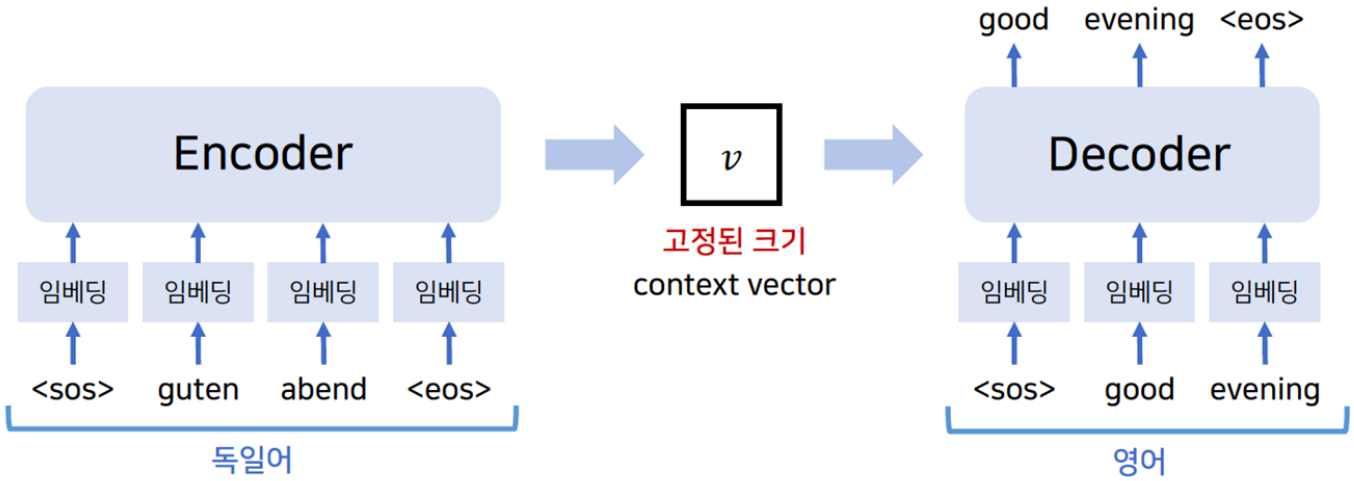


Sequence to Sequence Learning with Neural Networks (NIPS 2014) 나동빈

출처: https://www.youtube.com/watch?v=4DzKM0vgG1Y

Sequence to Sequence Learning with Neural Networks (NIPS 2014)

- 본 논문에서는 LSTM을 활용한 효율적인 **Seq2Seq** 기계 번역 아키텍처를 제안합니다.
  - **Seq2Seq**는 딥러닝 기반 기계 번역의 돌파구와 같은 역할을 수행했습니다.
  - **Transformer(2017)**가 나오기 전까지 state-of-the-art로 사용되었습니다.



출처: https://www.youtube.com/watch?v=4DzKM0vgG1Y

어텐션의 아이디어

SEQ2SEQ 모델의 문제점 :

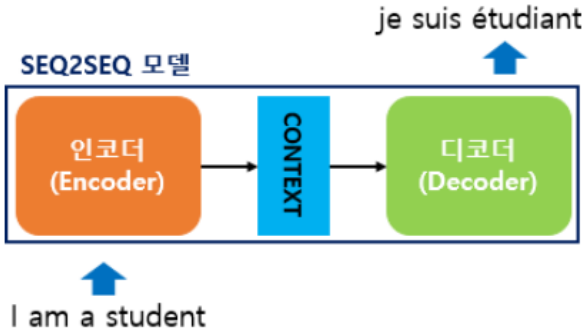
seq2seq 모델에는 크게 두 가지 문제가 있습니다.

첫째, 하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하니까 정보 손실이 발생합니다.

둘째, RNN의 고질적인 문제인 기울기 소실(Vanishing Gradient) 문제가 존재합니다.

즉, 결국 이는 기계 번역 분야에서 입력 문장이 길면 번역 품질이 떨어지는 현상으로 나타났습니다.

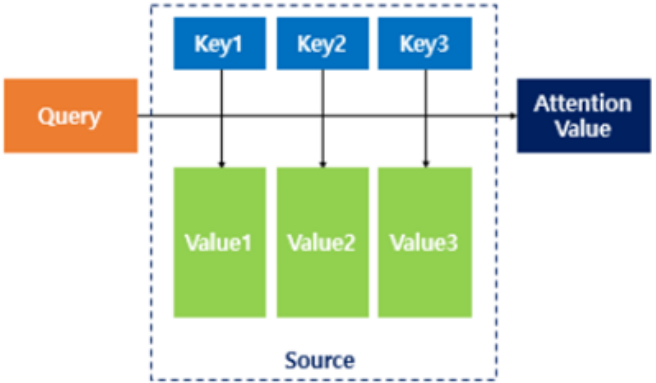
이를 위한 대안으로 입력 시퀀스가 길어지면 출력 시퀀스의 정확도가 떨어지는 것을 보정해주기 위한 등장한 기법인 **어텐션(attention)**을 소개합니다.



어텐션 함수

어텐션을 함수로 표현하면 주로 다음과 같이 표현됩니다.

`Attention(Q, K, V) = Attention Value`



지금부터 배우게 되는 **seq2seq + 어텐션** 모델에서 Q, K, V에 해당되는 각각의 Query, Keys, Values는 각각 다음과 같습니다.

```
Q = Query : t 시점의 디코더 셀에서의 은닉 상태
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들
```

어텐션의 아이디어

어텐션의 기본 아이디어는 디코더에서 **출력 단어를 예측하는 매 시점(time step)마다**, 인코더에서의 전체 입력 문장을 다시 한 번 참고한다는 점입니다. 단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야할 단어와 **연관이 있는 입력 단어 부분을** 좀 더 **집중(attention)해서** 보게 됩니다.

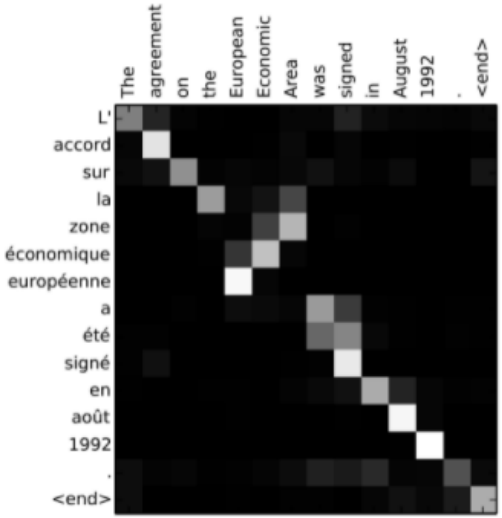


Figure 4. Attention visualization – example of the alignments between source and target sentences. Image is taken from (Bahdanau et al., 2015).

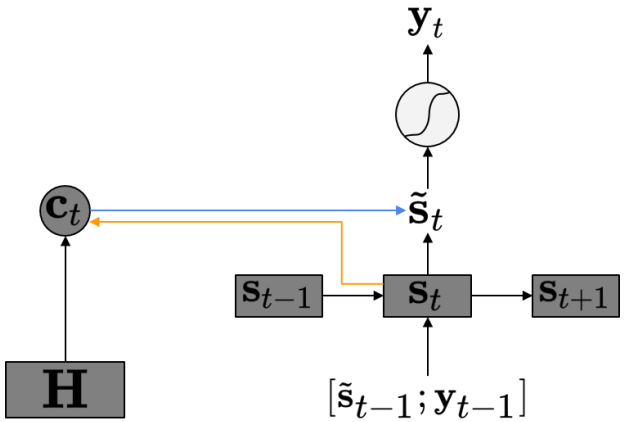
https://github.com/tensorflow/nmt#background-on-the-attention-mechanism

다양한 종류의 어텐션

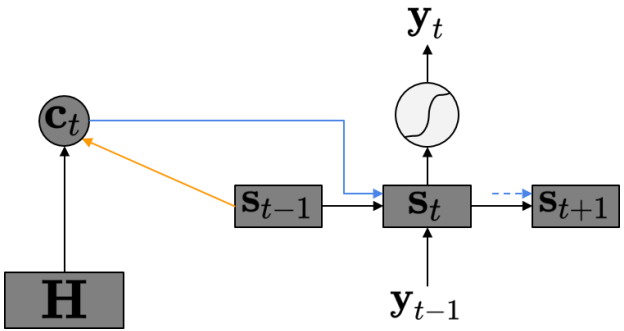
어텐션 스코어를 구하는 방법은 여러가지가 제시되어 있으며, 현재 제시된 여러 종류의 어텐션 스코어 함수는 다음과 같습니다.

이름	스코어 함수	Defined by
dot	$score(s_t,\ h_i) = s_t^T h_i$	Luong et al. (2015)
$scaled\ dot$	$score(s_t,\ h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Vaswani et al. (2017)
$general$	$score(s_t,\ h_i) = s_t^T W_a h_i$ // 단, W_a는 학습 가능한 가중치 행렬	Luong et al. (2015)
$concat$	$score(s_t,\ h_i) = W_a^T\ tanh(W_b[s_t; h_i])score(s_t,\ h_i) = W_a^T\ tanh(W_b s_t + W_c h_i)$	Bahdanau et al. (2015)
$location-base$	$\alpha_t = softmax(W_a s_t)$ // α_t 산출 시에 s_t만 사용하는 방법.	Luong et al. (2015)

Luong Attention vs. Bahdanau Attention



$$
\begin{aligned}
\mathbf{c}_t &= \sum_{j=1}^{T_\mathbf{x}} \mathbf{a}_{tj}\mathbf{h}_j \\
&= \mathbf{H}\mathbf{a}_t \\
\mathbf{a}_t &= \text{Softmax}\left((\text{Score}(\mathbf{s}_t, \mathbf{h}_j))_{j=1}^{T_\mathbf{x}}\right) \in \mathbb{R}^{T_\mathbf{x}} \\
\\
\widehat{\mathbf{y}}_t &= \text{Softmax}\left(\mathbf{W}_\mathbf{y}\tilde{\mathbf{s}}_t + \mathbf{b}_\mathbf{y}\right) \\
\tilde{\mathbf{s}}_t &= \tanh(\mathbf{W}_{\mathbf{ss}}\mathbf{s}_t + \mathbf{W}_{\mathbf{cs}}\mathbf{c}_t + \mathbf{b}_\mathbf{s})
\end{aligned}
$$



$$
\begin{aligned}
\mathbf{c}_t &= \sum_{j=1}^{T_\mathbf{x}} \mathbf{a}_{tj}\mathbf{h}_j \\
&= \mathbf{H}\mathbf{a}_t \\
\mathbf{a}_t &= \text{Softmax}\left((\text{Score}(\mathbf{s}_{t-1}, \mathbf{h}_j))_{j=1}^{T_\mathbf{x}}\right) \in \mathbb{R}^{T_\mathbf{x}} \\
\text{Score}(\mathbf{s}_{t-1}, \mathbf{h}_j) &= \mathbf{v}^\text{T}\tanh(\mathbf{W}_\mathbf{a}\mathbf{s}_{t-1} + \mathbf{U}_\mathbf{a}\mathbf{h}_j)
\end{aligned}
$$

https://hcnoh.github.io/2019-01-01-luong-attention

닷-프로덕션 어텐션

Luong Attention

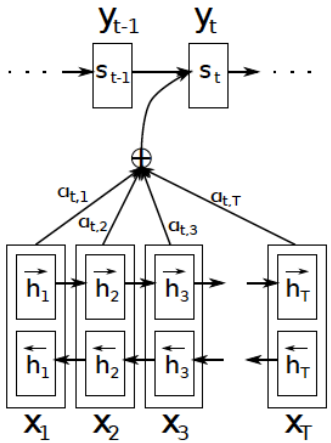
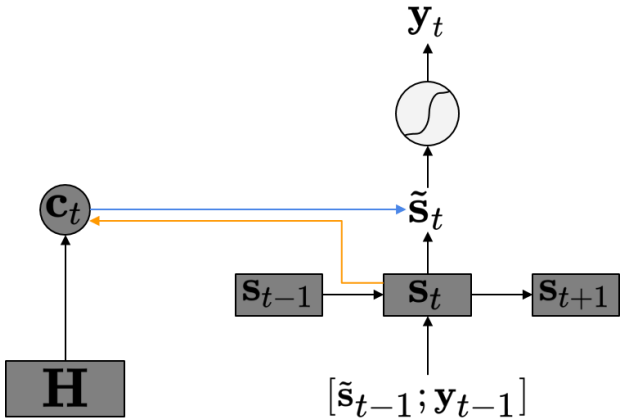


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



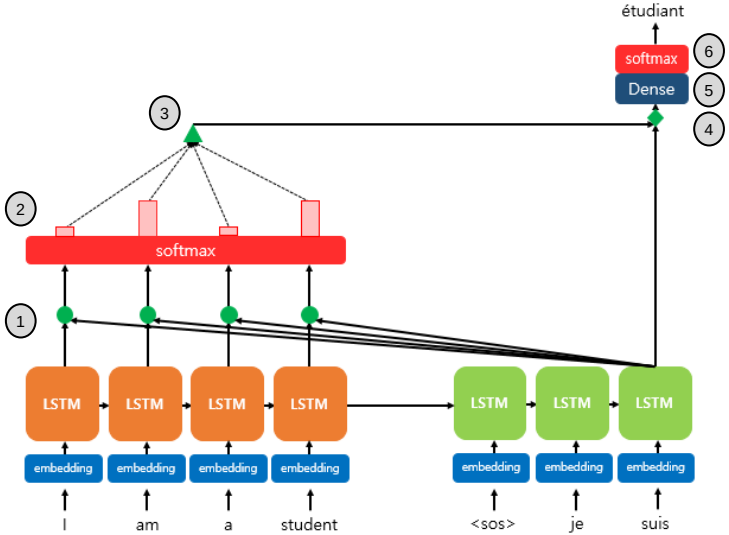
$$\hat{\mathbf{y}}_t = \text{Softmax}\left(\mathbf{W}_{\mathbf{y}}\tilde{\mathbf{s}}_t + \mathbf{b}_{\mathbf{y}}\right)$$
$$\tilde{\mathbf{s}}_t = \tanh(\mathbf{W}_{\mathbf{ss}}\mathbf{s}_t + \mathbf{W}_{\mathbf{cs}}\mathbf{c}_t + \mathbf{b}_{\mathbf{s}})$$

$$\mathbf{c}_t = \sum_{j=1}^{T_{\mathbf{x}}} \mathbf{a}_{tj}\mathbf{h}_j$$
$$= \mathbf{H}\mathbf{a}_t$$
$$\mathbf{a}_t = \text{Softmax}\left((\text{Score}(\mathbf{s}_t, \mathbf{h}_j))_{j=1}^{T_{\mathbf{x}}}\right) \in \mathbb{R}^{T_{\mathbf{x}}}$$

https://hcnoh.github.io/2019-01-01-luong-attention

닷-프로덕트 어센션

어텐션은 다양한 종류가 있는데 그 중에서도 가장 수식적으로 이해하기 쉽게 수식을 적용한 **닷-프로덕트 어텐션**(Dot-Product Attention)을 통해 어텐션을 이해해보도록 하겠습니다. seq2seq에서 사용되는 어텐션 중에서 닷-프로덕트 어텐션과 다른 어텐션의 차이는 주로 중간 수식의 차이로 **메커니즘 자체는 거의 유사**합니다.



1) 어텐션 스코어(Attention Score)를 구한다.

2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.

3) 각 인코더의 어텐션 가중치와 은닉 상태를 가중합하여 **어텐션 값** (Attention Value) 을 구한다.

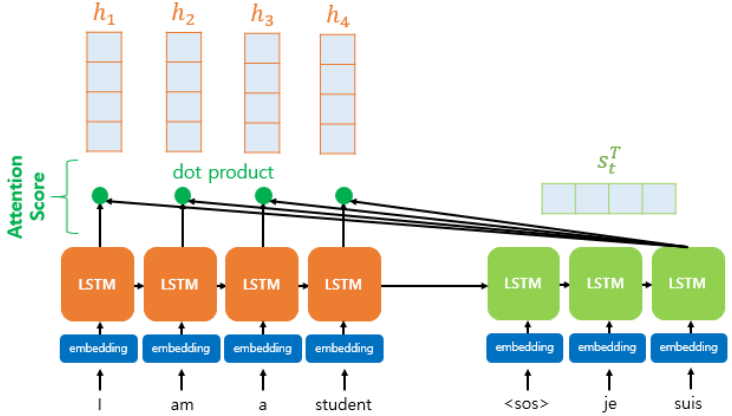
4) 어텐션 값과 디코더의 t 시점의 은닉 상태를 연결한다.(Concatenate)

5) 출력층 연산의 입력이 되는 $\tilde{s}^t$ 를 계산합니다.

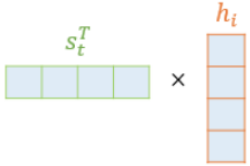
6) $\tilde{s}^t$ 를 출력층의 입력으로 사용합니다.

닷-프로덕트 어센션

1) 어텐션 스코어(Attention Score)를 구한다.



닷-프로덕트 어텐션에서는 이 스코어 값을 구하기 위해 s^t 를 전치(transpose) 하고 각 은닉 상태와 내적(dot product)을 수행합니다. 즉, 모든 어텐션 스코어 값은 스칼라입니다. 예를 들어 s^t 과 인코더의 i번째 은닉 상태의 어텐션 스코어의 계산 방법은 아래와 같습니다.



어텐션 스코어 함수를 정의해보면 다음과 같습니다.

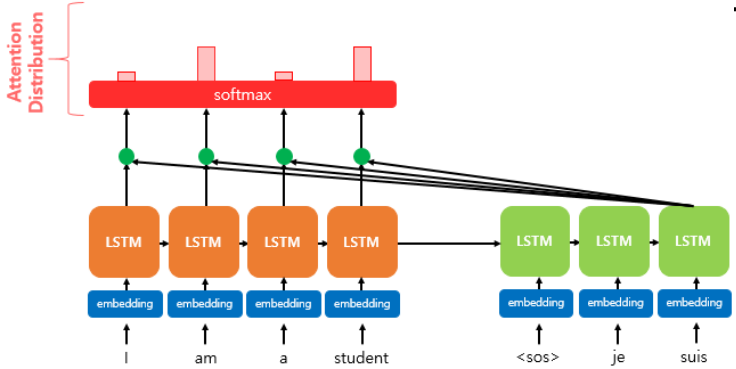
$$score(s_t,\ h_i) = s_t^T h_i$$

s_t와 인코더의 모든 은닉 상태의 어텐션 스코어의 모음값을 e^t라고 정의하겠습니다. e^t의 수식은 다음과 같습니다.

$$e^t = [s_t^T h_1, \ldots, s_t^T h_N]$$

닷-프로덕트 어센션

2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.



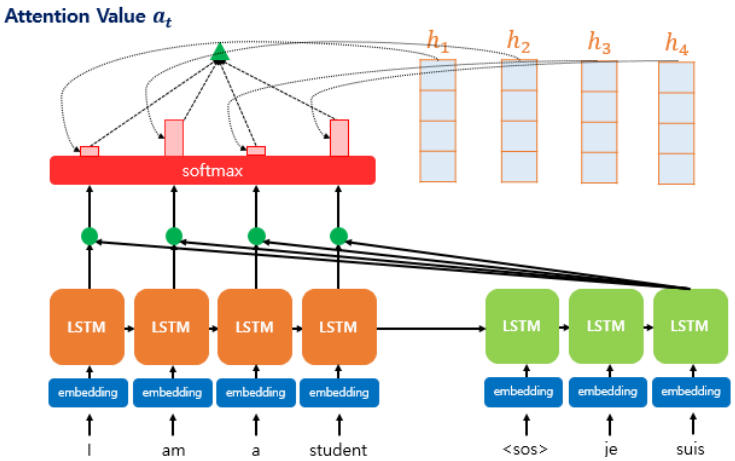
e^t 에 소프트맥스 함수를 적용하여, 모든 값을 합하면 1이 되는 확률 분포를 얻어냅니다. 이를 **어텐션 분포 (Attention Distribution)**라고 하며, 각각의 값은 **어텐션 가중치 (Attention Weight)**라고 합니다.

$$e^t = [s_t^T h_1, \ldots, s_t^T h_N]$$

디코더의 시점 t에서의 어텐션 가중치의 모음값인 **어텐션 분포**를 α^t 이라고 할 때, α^t 을 식으로 정의하면 다음과 같습니다.

$$\alpha^t = softmax(e^t)$$

닷-프로덕트 어센션



가중합하여 어텐션 값(Attention Value)을 구한다

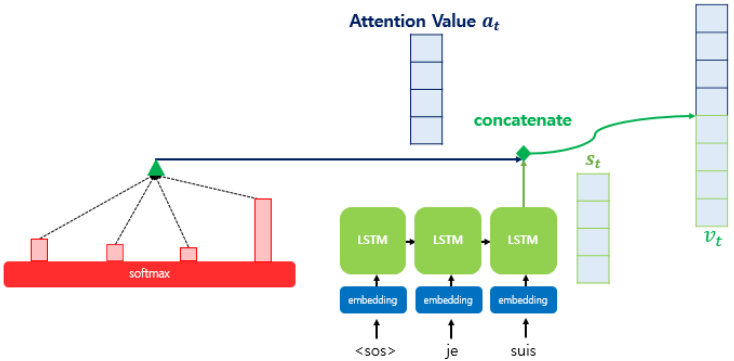
이제 지금까지 준비해온 정보들을 하나로 합치는 단계입니다. 어텐션의 최종 결과값을 얻기 위해서 각 인코더의 은닉 상태와 어텐션 가중치값들을 곱하고, 최종적으로 모두 더합니다. 요약하면 가중합(Weighted Sum)을 한다고 말할 수도 있겠습니다. 아래는 어텐션의 최종 결과. 즉, 어텐션 함수의 출력값인 **어텐션 값(Attention Value)** a_t 에 대한 식을 보여줍니다

$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i$$

이러한 어텐션 값 a_t 은 종종 인코더의 문맥을 포함하고 있다고하여, **컨텍스트 벡터 (context vector) 라고도 불립니다**. 앞서 배운 가장 기본적인 seq2seq에서는 인코더의 마지막 은닉 상태를 컨텍스트 벡터라고 부르는 것과 대조됩니다.

닷-프로덕트 어센션

4) 어텐션 값과 디코더의 t 시점의 은닉 상태를 연결한다.(Concatenate)

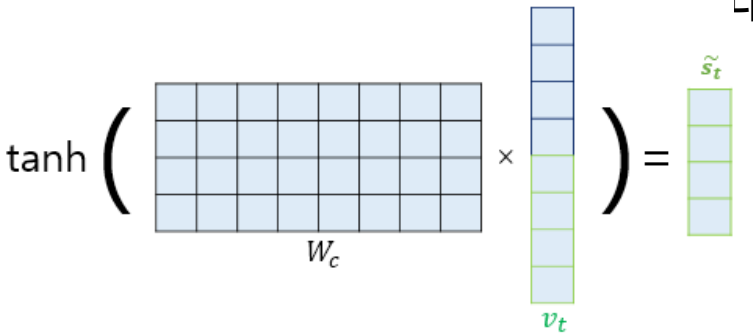


이제 어텐션 함수의 최종값인 어텐션 값 a_t 을 구했습니다. 앞서 어텐션 메커니즘이 들어간 t시점의 은닉 상태를 구하는 방법의 식으로 다음과 같은 식을 소개한 바 있습니다. 사실 어텐션 값이 구해지면 어텐션 메커니즘은 a_t 를 s_t 와 결합(concatenate)하여 하나의 벡터로 만드는 작업을 수행합니다. 이를 v_t 라고 정의해보겠습니다. 그리고 이 v_t 를 y^ 예측 연산의 입력으로 사용하므로서 인코더로부터 얻은 정보를 활용하여 y^ 를 좀 더 잘 예측할 수 있게 됩니다. 이것이 어텐션 메커니즘의 핵심입니다.

$$v_t = [a_t; s_t]$$

닷-프로덕트 어센션

5) 출력층 연산의 입력이 되는 $\tilde{s}_t$를 계산합니다.



논문에서는 v_t 를 바로 출력층으로 보내기 전에 신경망 연산을 한 번 더 추가하였습니다. 가중치 행렬과 곱한 후에 하이퍼볼릭탄젠트 함수를 지나도록 하여 출력층 연산을 위한 새로운 벡터인 $\tilde{s}_t$ 를 얻습니다. 어텐션 메커니즘을 사용하지 않는 seq2seq에서는 출력층의 입력이 t시점의 은닉 상태인 s_t 였던 반면, 어텐션 메커니즘에서는 출력층의 입력이 $\tilde{s}_t$ 가 되는 셈입니다.

$$\tilde{s}_t = \tanh\left(\mathbf{W_c}[a_t; s_t] + b_c\right)$$

식에서 W_c 는 학습 가능한 가중치 행렬, b_c 는 편향입니다. 그림에서 편향은 생략했습니다.

닷-프로덕트 어센션

6) $\tilde{s}_t$ 를 출력층의 입력으로 사용합니다

$\tilde{s}_t$를 출력층의 입력으로 사용하여 예측 벡터를 얻습니다.

$$\hat{y}_t = \text{Softmax}\left(W_y \tilde{s}_t + b_y\right)$$

저자 참고 자료

- ✓ 어텐션을 소개한 논문 : https://arxiv.org/pdf/1409.0473.pdf
- ✓ 추천하는 참고 자료 : http://docs.likejazz.com/attention/
- ✓ https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2015/slides/lec14.neubig.seq_to_seq.pdf
- ✓ https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
- ✓ https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

바다나우 어텐션
(Badana Attention)

바다나우 목차

1. 바다나우 어텐션과 닷 프로덕트 어텐션

1.컨텍스트 벡터

•바다나우: 컨텍스트 벡터를 구할 때 이전 시점의 은닉 상태 s_{t-1}를 사용한다.

•루옹: 컨텍스트 벡터를 구할 때 현재 시점의 은닉 상태 s_t를 사용한다.

2.출력 y_t

•바다나우: 현재 시점의 은닉 상태 s_t로부터 출력이 나온다.

•루옹: 현재 시점의 은닉 상태 s_t는 RNN의 은닉 상태 역할만 하고, 새로운 벡터 $s\sim t$를 사용한다.

3.computation path

•바다나우: 디코더의 은닉 상태 st를 구할 때 컨텍스트 벡터가 사용되므로 RNN의 재귀 연산이 수행될 때 컨텍스트 벡터가 구해질 때까지 기다려야 한다.

•루옹: 출력을 구하는 부분과 RNN의 재귀 연산 부분이 분리되어 계산이 빨라졌다.

4.인코더의 은닉 상태 사용

•바다나우: 인코더의 모든 은닉 상태의 벡터를 본다.

•루옹: 특정 하이퍼파라미터 D에 대해 (2D+1)개의 부분집합 벡터만 본다.

1. 바다나우 어텐션과 닷 프로덕트 어텐션

Attention(Q, K, V) = Attention Value

t = 어텐션 메커니즘이 수행되는 디코더 셀의 현재 시점을 의미.
Q = Query : t-1 시점의 디코더 셀에서의 은닉 상태
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

닷-프로덕트 어텐션과 달리 바다나우 어텐션에서의

Query는 **이전 시점 t-1에서의 은닉 상태** 임을 유의해야 합니다.

1. 바다나우 어텐션과 닷 프로덕트 어텐션

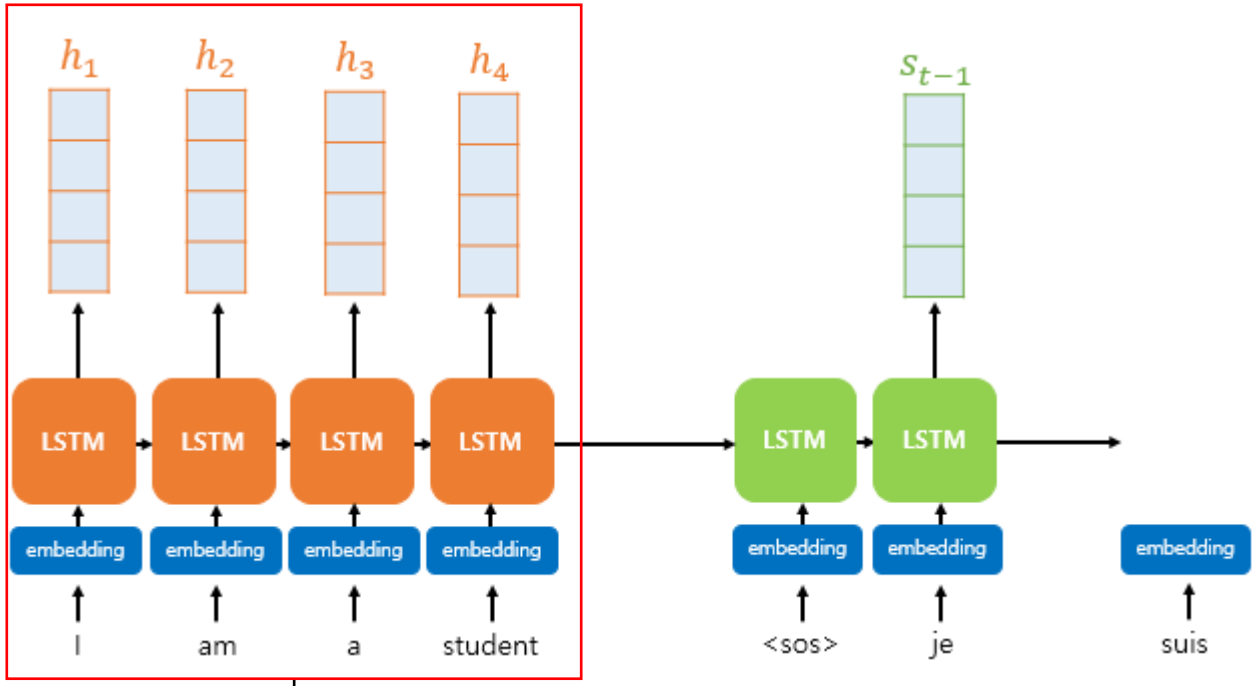
이름	스코어 함수	Defined by
dot	$score(s_t,\ h_i) = s_t^T h_i$	Luong et al. (2015)
$scaled\ dot$	$score(s_t,\ h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Vaswani et al. (2017)
$general$	$score(s_t,\ h_i) = s_t^T W_a h_i$ // 단, W_a는 학습 가능한 가중치 행렬	Luong et al. (2015)
$concat$	$score(s_t,\ h_i) = W_a^T\ tanh(W_b[s_t; h_i])$ $score(s_t,\ h_i) = W_a^T\ tanh(W_b s_t + W_c h_i)$	Bahdanau et al. (2015)
$location - base$	$\alpha_t = softmax(W_a s_t)$ // α_t 산출 시에 s_t만 사용하는 방법.	Luong et al. (2015)

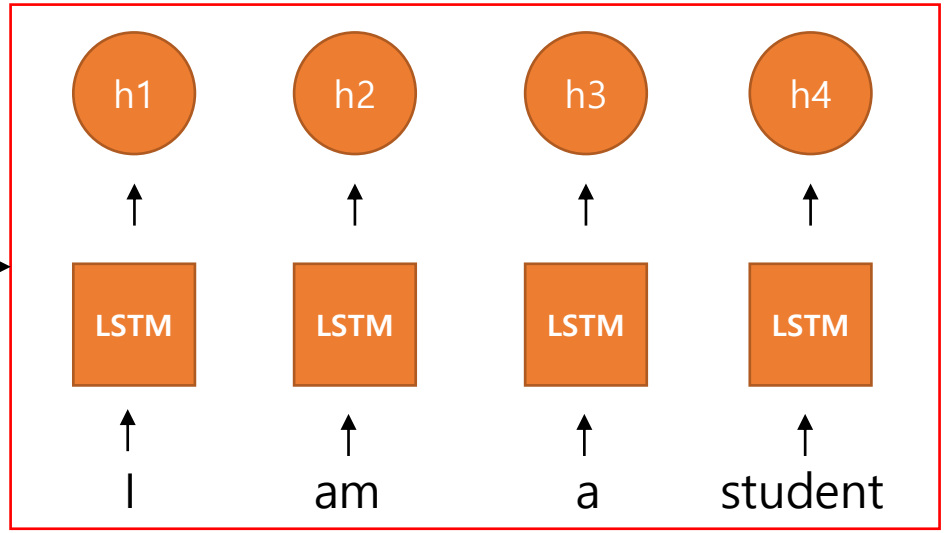
닷 – 프로덕트 어텐션(Dot-Product Attention) - dot

바다나우 어텐션(Bahdanau Attention - Global Attention) - concat

2. 바다나우 어텐션 (Bahdanau Attention)

1) Attention score를 구한다.

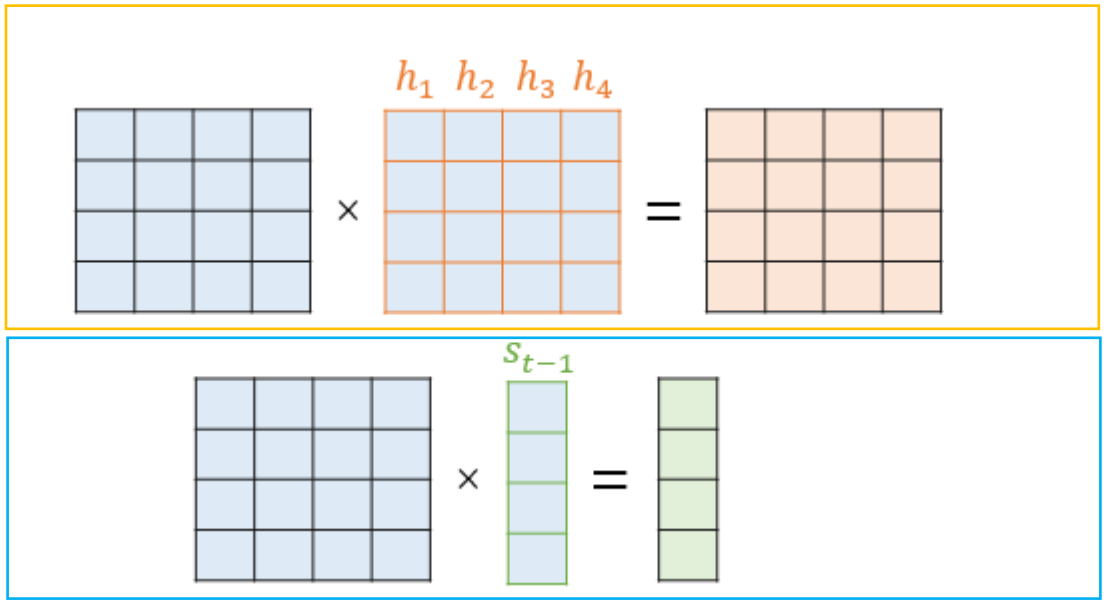




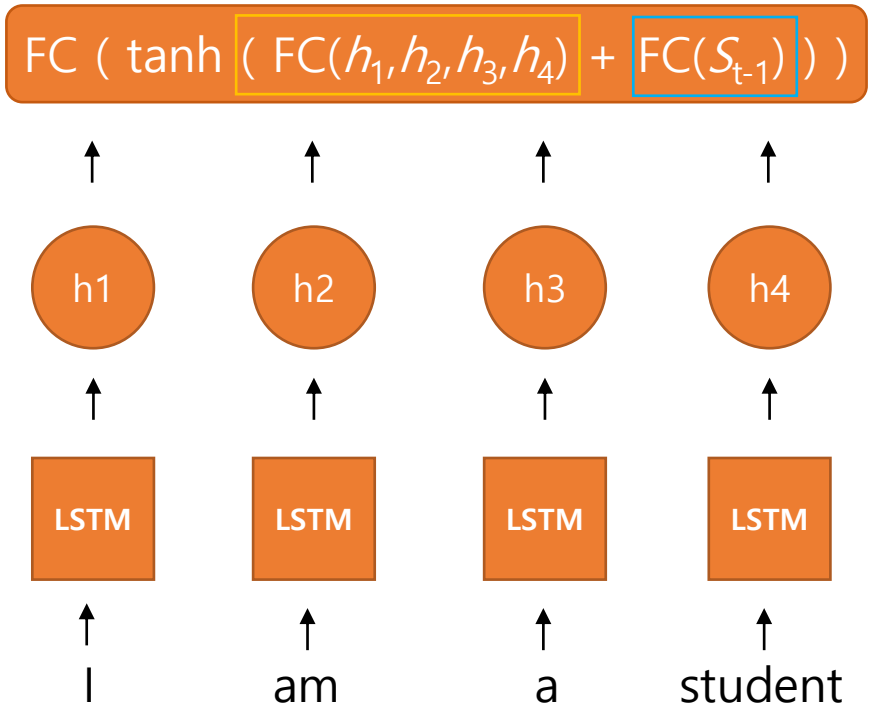
2. 바다나우 어텐션 (Bahdanau Attention)

1) Attention score를 구한다.

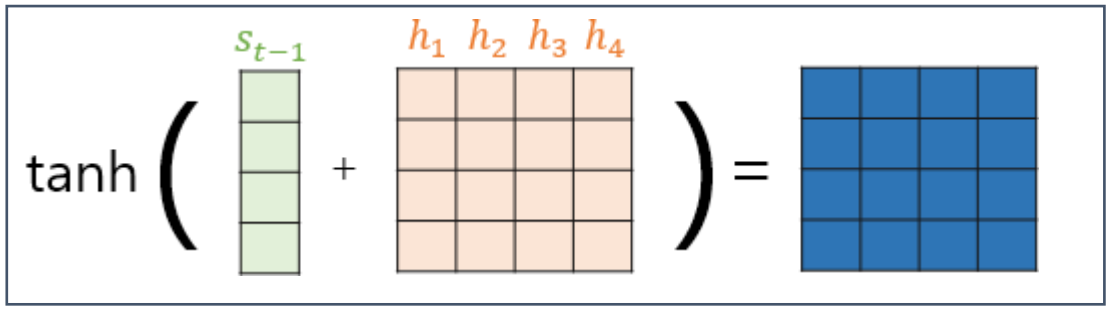
FC (학습 가능한 가중치 행렬과 연산을 의미)



$$score(s_{t-1}, h_i) = W_a^T \tanh(W_b s_{t-1} + W_c h_i)$$



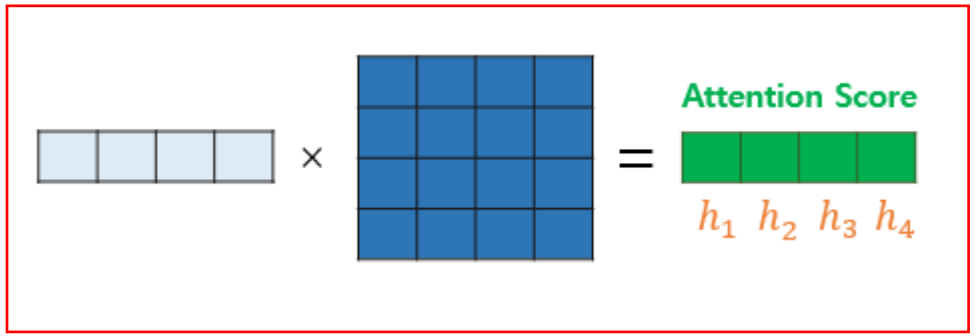
2. 바다나우 어텐션 (Bahdanau Attention)



$\tanh(W_b s_{t-1}+W_c H)$

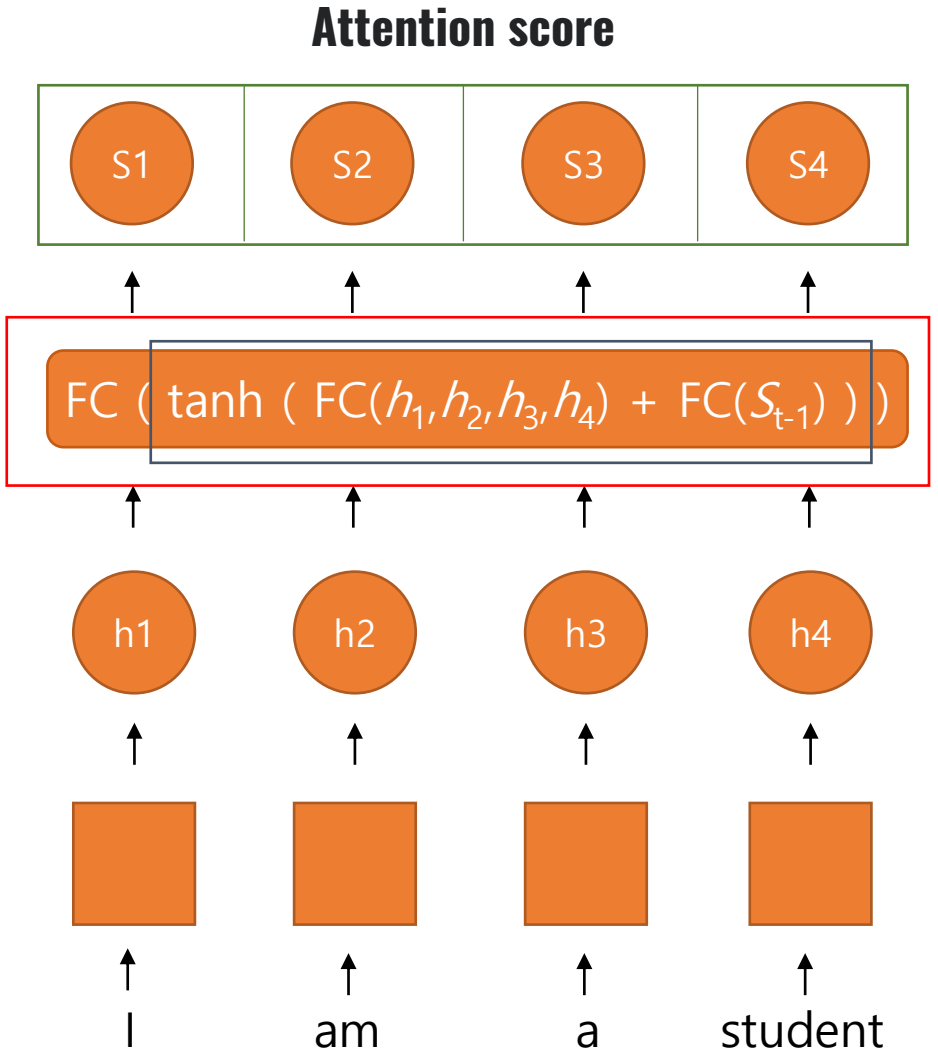
이제 W_a^T와 곱하여 s_{t-1}와 h_1,h_2,h_3,h_4의 유사도가 기록된

어텐션 스코어 벡터 e_t를 얻습니다.



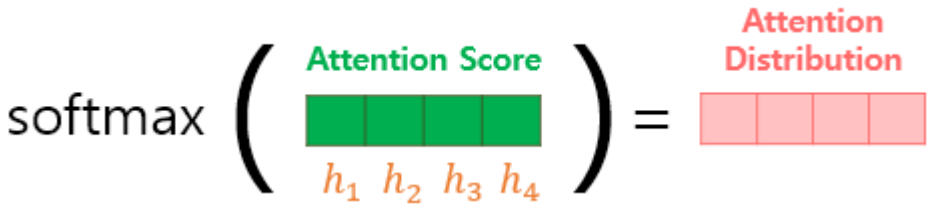
$e_t = W_a^T \tanh(W_b s_{t-1}+W_c H)$

1) Attention score를 구한다.

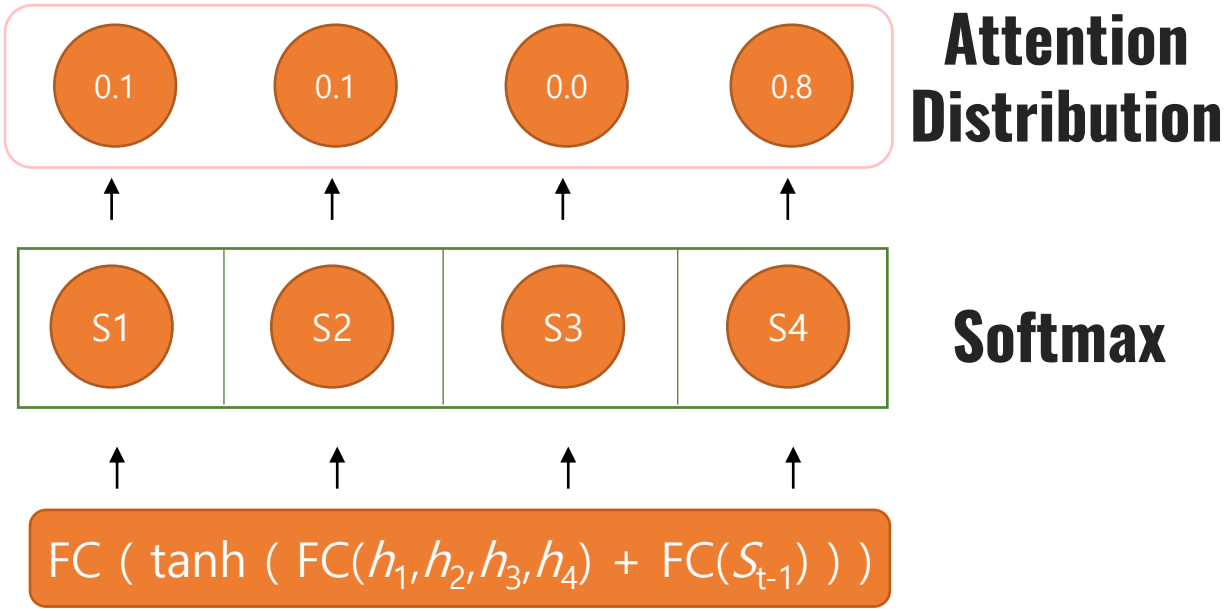


2. 바다나우 어텐션 (Bahdanau Attention)

2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.

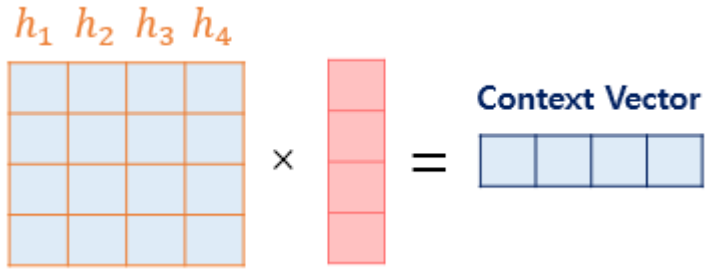


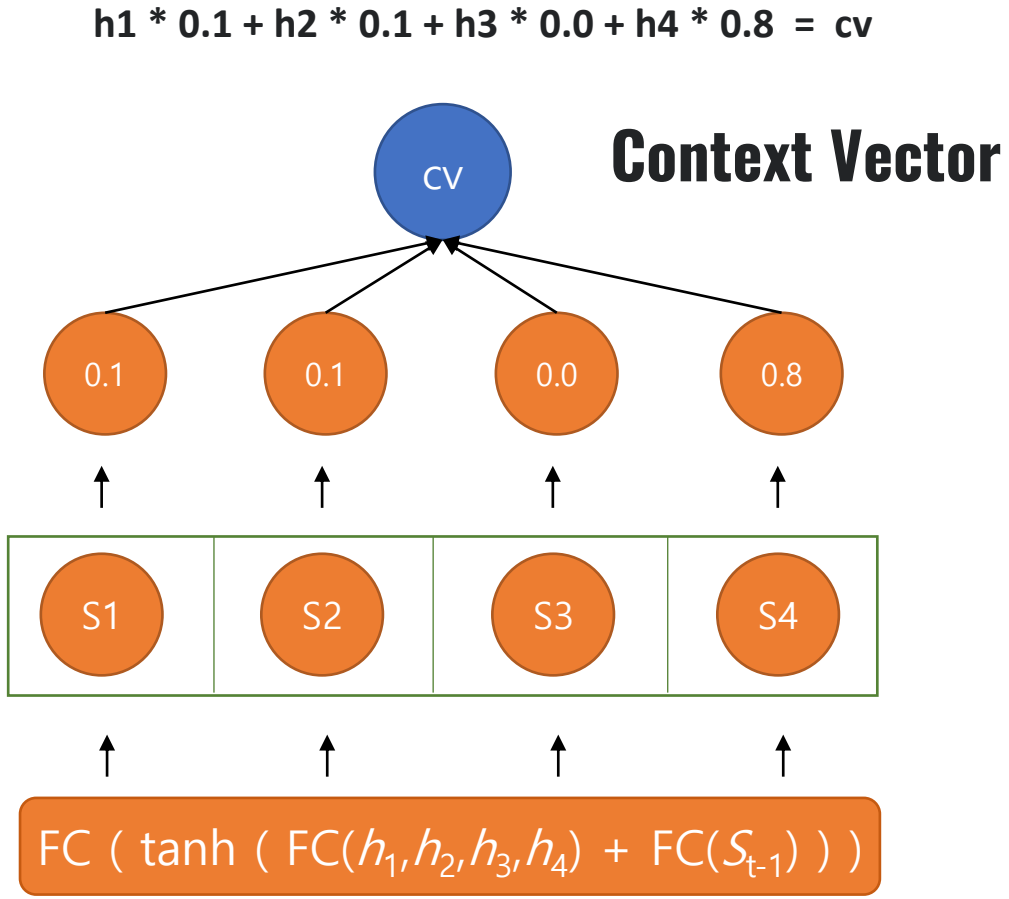
e_t에 Softmax 함수를 적용하여,
모든 값을 합하면 1이 되는 확률 분포를 얻어냅니다.
이를 **어텐션 분포(Attention Distribution)**라고 하며,
각각의 값은 **어텐션 가중치(Attention Weight)**라고 합니다.



2. 바다나우 어텐션 (Bahdanau Attention)

3) 각 인코더의 어텐션 가중치와 은닉 상태를 가중 합하여 어텐션 값을 구한다.





지금까지 준비해온 정보들을 하나로 합치는 단계입니다.

어텐션의 최종 결과값을 얻기 위해서 각 인코더의 은닉 상태와 어텐션 가중치 값들을 곱하고, 최종적으로 모두 더합니다.

요약하면 가중 합(Weighted Sum)을 한다고 말할 수도 있겠습니다.

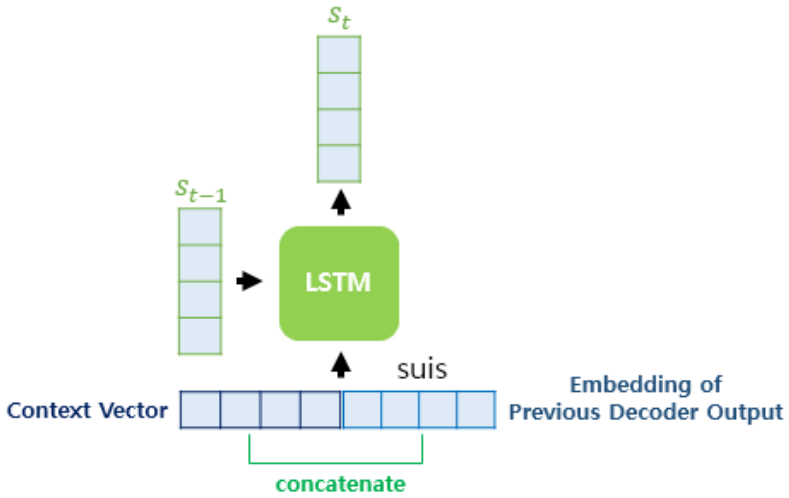
이 벡터는 인코더의 문맥을 포함하고 있다고하여,

컨텍스트 벡터(context vector)라고 부릅니다.

2. 바다나우 어텐션의 구조

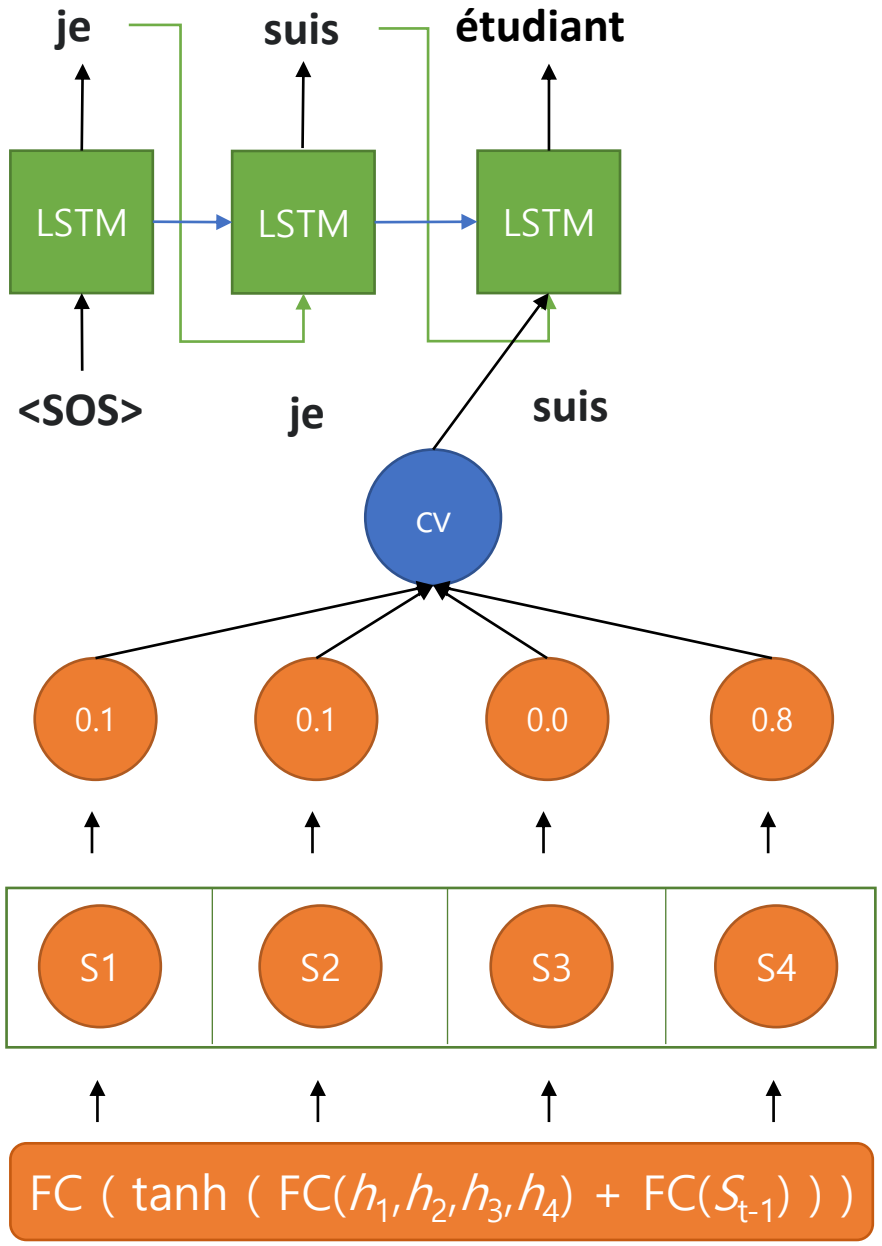
4) 컨텍스트 벡터로부터 s_t를 구합니다.

어텐션 메커니즘에서는 컨텍스트 벡터와 현재 시점의 입력인
단어의 **임베딩 벡터를 연결(concatenate)**하고,
현재 시점의 새로운 입력으로 사용하는 모습을 보여줍니다.
그리고 이전 시점의 셀로부터 전달받은 은닉 상태 s_{t-1}와
현재 시점의 새로운 입력으로부터 s_t를 구합니다.

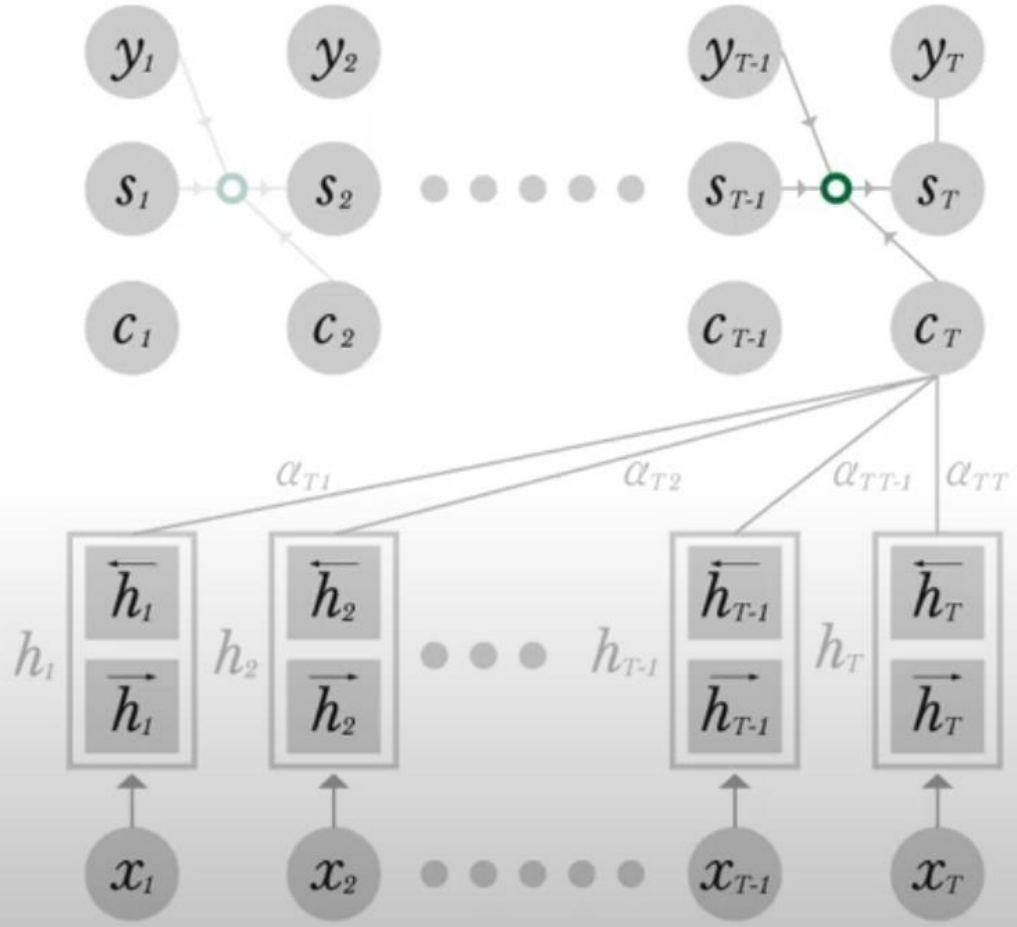


tanh(concat (cv , <suis>))

이후에는 어텐션 메커니즘을 사용하지 않는 경우와 동일합니다.
s_t는 출력층으로 전달되어 현재 시점의 예측 값을 구하게 됩니다.



3. 수식으로 보는 바다나우 어텐션



y_T : *Output* *Decoder* $p(y_T \mid y_{T-1}, \cdots, y_{T-1}, X) = p(y_{T-1}, s_T, c_T)$

$$s_T = f(s_{T-1}, y_{T-1}, c_T)$$

$$c_T = \sum_{j=1}^{T} \alpha_{Tj} h_j$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})} \quad | \quad e_{ij} = a(s_{i-1}, h_j)$$

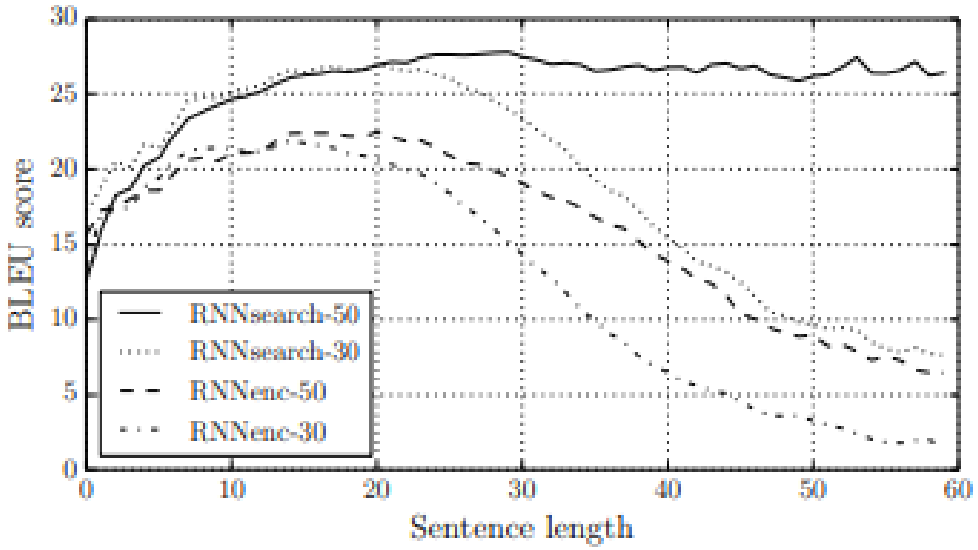
where a: alignment model (parameterized)

$$h_j = [\overrightarrow{h_j}^T ; \overleftarrow{h_j}^T]^T$$ *concatenating the forward/backward hidden state*

x_T : *Input*

https://www.youtube.com/watch?v=6aouXD8WMVQ&ab_channel=10mindeeplearning

4. 바다나우 어텐션의 프랑스어 기계번역 결과



Sentence에 따른 BLEU값 비교

참고 영상 자료 :

https://www.youtube.com/watch?v=WsQLdu2JMgI&ab_channel=MinsukHeo%ED%97%88%EB%AF%BC%EC%84%9D
https://www.youtube.com/watch?v=6aouXD8WMVQ&ab_channel=10mindeeplearning
https://www.youtube.com/watch?v=p3jmVkUMMuw&ab_channel=TmaxAI
https://www.youtube.com/watch?v=SysgYptB198&ab_channel=DeepLearningAI

참고 문서 자료 :

https://hcnoh.github.io/2018-12-11-bahdanau-attention
https://medium.com/analytics-vidhya/neural-machine-translation-using-bahdanau-attention-mechanism-d496c9be30c3
https://eda-ai-lab.tistory.com/157

양방향 LSTM과 어텐션 메커니즘
(BiLSTM with Attention mechanism)

개요

- **단방향 LSTM으로 텍스트 분류를 수행할 수도 있지만 때로는 양방향 LSTM을 사용하는 것이 더 강력합니다.**
- 여기에 추가적으로 어텐션 메커니즘(바다나우 어텐션)을 사용합니다.
- 양방향 LSTM과 어텐션 메커니즘으로 IMDB 리뷰 감성 분류하기를 수행해봅시다.

1. IMDB 리뷰 데이터 전처리하기

IMDB 리뷰 데이터 전처리하기

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

- 필요한 모듈 import 합니다.
- IMDB 리뷰 데이터는 앞서 텍스트 분류하기 챕터에서 다룬 바 있으므로 데이터에 대한 상세 설명은 생략합니다.

```
vocab_size = 10000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = vocab_size)
```

- 최대 단어 개수를 10,000개로 제한하고 훈련 데이터와 테스트 데이터를 받아옵니다.
- 훈련 데이터와 이에 대한 레이블이 각각 X_train, y_train에 테스트 데이터와 이에 대한 레이블이 각각 X_test, y_test에 저장되었습니다.

IMDB 리뷰 데이터 전처리하기

```
print('리뷰의 최대 길이 : {}'.format(max(len(l) for l in X_train)))
print('리뷰의 평균 길이 : {}'.format(sum(map(len, X_train))/len(X_train)))
```

```
리뷰의 최대 길이 : 2494
리뷰의 평균 길이 : 238.71364
```

- IMDB 리뷰 데이터는 이미 정수 인코딩이 된 상태이므로 남은 전처리는 패딩 뿐입니다.
- 리뷰의 최대 길이와 평균 길이를 확인해봅니다.

```
max_len = 500
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

- 평균 길이보다는 조금 크게 데이터를 패딩합니다.
- 훈련용 리뷰 데이터와 리뷰의 길이가 모두 500이 되었습니다.

2. 바다나우 어텐션
(Bahdanau Attention)

바다나우 어텐션

- 여기서 사용할 어텐션은 바다나우 어텐션(Bahdanau Attention)입니다.
- 이를 이해하기 위해 앞서 배운 닷-프로덕트 어텐션과 어텐션 스코어 함수를 상기해봅니다.

- **어텐션 스코어 함수란, 주어진 query와 모든 key에 대해서 유사도를 측정하는 함수를 말합니다.**

바다나우 어텐션 스코어 함수

- 닷-프로덕트 어텐션에서는 query와 key의 유사도를 구하는 방법이 내적(dot product)이었습니다.

$$score(query,\ key) = query^T key$$

- 바다나우 어텐션 스코어 함수는 아래와 같습니다.

$$score(query,\ key) = V^T tanh(W_1 key + W_2 query)$$

텍스트 분류에서의 어텐션 메커니즘

- 그런데 텍스트 분류에서 어텐션 메커니즘을 사용하는 이유는 무엇일까요?

-> RNN의 마지막 은닉 상태는 예측을 위해 사용됩니다.
그런데 이 RNN의 마지막 은닉 상태는 몇 가지 유용한 정보들을 손실한 상태입니다.
그래서 RNN이 time step을 지나며 손실했던 정보들을 다시 참고하고자 합니다.

-> **이는 다시 말해 RNN의 모든 은닉 상태들을 다시 한 번 참고하겠다는 것입니다.**
그리고 이를 위해 어텐션 메커니즘을 사용하는 것입니다.

바다나우 어텐션 구현

```
class BahdanauAttention(tf.keras.Model):
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = Dense(units)
    self.W2 = Dense(units)
    self.V = Dense(1)

  def call(self, values, query): # 단, key와 value는 같음
    # query shape == (batch_size, hidden size)
    # hidden_with_time_axis shape == (batch_size, 1, hidden size)
    # score 계산을 위해 뒤에서 할 덧셈을 위해서 차원을 변경해줍니다.
    hidden_with_time_axis = tf.expand_dims(query, 1)

    # score shape == (batch_size, max_length, 1)
    # we get 1 at the last axis because we are applying score to self.V
    # the shape of the tensor before applying self.V is (batch_size, max_length, units)
    score = self.V(tf.nn.tanh(
        self.W1(values) + self.W2(hidden_with_time_axis)))

    # attention_weights shape == (batch_size, max_length, 1)
    attention_weights = tf.nn.softmax(score, axis=1)

    # context_vector shape after sum == (batch_size, hidden_size)
    context_vector = attention_weights * values
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights
```

- BahdanauAttention 모델은 뉴럴 네트워크를 통해 score를 구하는 방식입니다.

- 그래서 Dense 레이어를 2개 거치고 하나로 모아서 softmax를 거쳐 attention_weight를 계산합니다.

- attention_weight를 value(현재 hidden state)와 곱해서 context_vector를 구합니다.

3. 양방향 LSTM + 어텐션 메커니즘

(BiLSTM with Attention Mechanism)

실습

양방향 LSTM + 어텐션 메커니즘 실습

```
from tensorflow.keras.layers import Dense, Embedding, Bidirectional, LSTM, Concatenate, Dropout
from tensorflow.keras import Input, Model
from tensorflow.keras import optimizers
import os
```

- 필요한 라이브러리와 모듈을 import 합니다.
- 여기서는 **텐서플로우의 케라스 함수형 API**를 사용하여 모델을 설계합니다.

양방향 LSTM + 어텐션 메커니즘 실습

```
sequence_input = Input(shape=(max_len,), dtype='int32')
embedded_sequences = Embedding(vocab_size, 128, input_length=max_len, mask_zero = True)(sequence_input)
```

- 우선 입력층과 임베딩층을 설계합니다.
- 10,000 개의 단어들을 128차원의 벡터로 임베딩하도록 설계합니다.

```
lstm = Bidirectional(LSTM(64, dropout=0.5, return_sequences = True))(embedded_sequences)
```

- 이제 양방향 LSTM을 설계합니다.
- **단, 여기서는 양방향 LSTM을 두 층을 사용합니다.**
- 우선 첫 번째 층에 두번째 층을 쌓을 예정이므로 return_sequences를 True로 해주어야 합니다.

```
lstm, forward_h, forward_c, backward_h, backward_c = Bidirectional \
  (LSTM(64, dropout=0.5, return_sequences=True, return_state=True))(lstm)
```

- 두번째 층을 설계합니다. 상태를 리턴받아야 하므로 return_state를 True로 해줍니다.

양방향 LSTM + 어텐션 메커니즘 실습

- 각 상태의 크기(shape)를 출력해봅니다.

```
print(lstm.shape, forward_h.shape, forward_c.shape, backward_h.shape, backward_c.shape)
```

```
(None, 500, 128) (None, 64) (None, 64) (None, 64) (None, 64)
```

- **양방향 LSTM이기에** 순방향 LSTM의 은닉 상태와 셀 상태를 forward_h, forward_c 에 저장하고, 역방향 LSTM의 은닉 상태와 셀 상태를 backward_h, backward_c 에 저장합니다.

- 각 은닉 상태나 셀 상태의 경우에는 128차원을 가지는데, LSTM의 경우에는 (500 x 128)의 크기를 가집니다.
- **forward 방향과 backward 방향이 연결된 hidden state 벡터가 모든 시점에 대해서 존재함을 의미합니다.**

양방향 LSTM + 어텐션 메커니즘 실습

```
state_h = Concatenate()([forward_h, backward_h]) # 은닉 상태
state_c = Concatenate()([forward_c, backward_c]) # 셀 상태
```

- 양방향 LSTM의 은닉 상태와 셀 상태를 사용하려면 두 방향의 LSTM의 상태들을 연결(concatenate)해주면 됩니다.

```
attention = BahdanauAttention(64) # 가중치 크기 정의
context_vector, attention_weights = attention(lstm, state_h)
```

- 앞서 언급한 바다나우 어텐션을 이용한 은닉 상태를 사용합니다.
- 이를 입력으로 하는 컨텍스트 벡터를 얻습니다.

양방향 LSTM + 어텐션 메커니즘 실습

```
dense1 = Dense(20, activation="relu")(context_vector)
dropout = Dropout(0.5)(dense1)
output = Dense(1, activation="sigmoid")(dropout)
model = Model(inputs=sequence_input, outputs=output)
```

- 컨텍스트 벡터를 밀집층(dense layer)에 통과 시키고, **이진 분류이므로 최종 출력층에 1개의 뉴런을 배치하고, 활성화 함수로 시그모이드 함수를 사용합니다.**

양방향 LSTM + 어텐션 메커니즘 실습

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- 옵티마이저로 adam을 사용하고 모델을 컴파일합니다.
- 시그모이드 함수를 사용하므로 손실 함수로는 binary_crossentropy를 사용합니다.

```
history = model.fit(X_train, y_train, epochs = 3, batch_size = 256, validation_data=(X_test, y_test), verbose=1)
```

- 이제 모델을 훈련합니다.
- 검증 데이터로 테스트 데이터를 사용하여 에포크가 끝날 때마다 테스트 데이터에 대한 정확도를 출력하도록 합니다.

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/3
25000/25000 [==============================] - 566s 23ms/sample - loss: 0.4941 - accuracy: 0.7570 - val_loss: 0.3110 - val_accu
racy: 0.8721
Epoch 2/3
25000/25000 [==============================] - 541s 22ms/sample - loss: 0.2530 - accuracy: 0.9074 - val_loss: 0.2852 - val_accu
racy: 0.8835
Epoch 3/3
25000/25000 [==============================] - 543s 22ms/sample - loss: 0.1901 - accuracy: 0.9352 - val_loss: 0.3375 - val_accu
racy: 0.8793
```

양방향 LSTM + 어텐션 메커니즘 실습

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

```
25000/25000 [==============================] - 183s 7ms/sample - loss: 0.1901 - acc: 0.8793
테스트 정확도: 0.8793
```

- 87.93%의 정확도를 얻었습니다.

- 위의 모델을 네이버 영화 리뷰 분류하기에도 수행해봅니다.
- 저자의 경우에는 86%의 정확도를 얻었습니다.

양방향 LSTM + 어텐션 메커니즘 실습 (개인 실습 결과)

- 저자와 같이 본 챕터의 모델을 데이터만 바꿔 사용하였습니다.

- IMDB 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘

=> 87.66%

- 네이버 영화 리뷰 실습 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘

=> 84.47%

- 네이버 쇼핑 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘

=> 91.6%

- 한국어 스팀 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘

=> 77.64%

양방향 LSTM + 어텐션 메커니즘 실습 (코드 공유)

- IMDB 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘
=> https://colab.research.google.com/drive/1fzEhl-jswtzvAcQxK94ZJkGJcfc18_O4?usp=sharing

- 네이버 영화 리뷰 실습 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘
=> https://colab.research.google.com/drive/1AT-bjNbx1TgWxGd-GAuxpRL-2JuUCqpR?usp=sharing

- 네이버 쇼핑 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘
=> https://colab.research.google.com/drive/1cT7mHDz7fTEGyXT75ZKjLF11mralATZc?usp=sharing

- 한국어 스팀 리뷰 데이터 + 양방향 LSTM + 바다나우 어텐션 메커니즘
=> https://colab.research.google.com/drive/1aGCat7sQj8IdFrjkNusArvQCK1vZ-5Qt?usp=sharing

참고 자료 1

- NTM with Attention : https://www.tensorflow.org/tutorials/text/nmt_with_attention
- Text classification using BiLSTM with attention : https://androidkt.com/text-classification-using-attention-mechanism-in-keras/
- https://matthewmcateer.me/blog/getting-started-with-attention-for-classification/
- https://machinetalk.org/2019/03/29/neural-machine-translation-with-attention-mechanism/
- https://github.com/zdqzyx/NLP/blob/master/TextClassification/imp_by_tensorflow2/TextBiRNNAtt/attention.py
- https://gist.github.com/cbaziotis/7ef97ccf71cbc14366835198c09809d2

참고 자료 2

- 10min deep learning 유투브 십분딥러닝_12_어텐션(Attention Mechanism)
- 허민석님 유투브 [딥러닝 기계번역] 시퀀스 투 시퀀스 + 어텐션 모델
- 허민석님 유투브 [딥러닝 기계번역] 트랜스포머 (어텐션 이즈 올 유 니드)
- 나동빈님 유투브 [딥러닝 기계 번역] Seq2Seq: Sequence to Sequence Learning with Neural Networks (꼼꼼한 딥러닝 논문 리뷰와 코드 실습)
- 나동빈님 유투브 [딥러닝 기계 번역] Transformer: Attention Is All You Need (꼼꼼한 딥러닝 논문 리뷰와 코드 실습)
- 이수안컴퓨터연구소 유투브 RNN 인코더, 디코더, Seq2Seq, Attention
- 고려대학교 산업경영공학부 DSBA 연구실 유투브 [Paper Review] Attention is All You Need (Transformer)

감사합니다

이상 19조 발표를 마칩니다.