# Attention

김동원

# 목차

Seq2Seq
Bahdanau Attention
Luong Attention

논문 요약 정리 노트

## Seq2seq모델(SEQUENCE TO SEQUENCE LEARNING WITH NEURAL NETOWRKS)

입력과 출력을 Sequence로 가지는 모델을 통해 언어모델을 구성한 방법에 관한 논문

DNN은 많은 분야에서 좋은 성능을 보였지만 시퀀스를 입력과 출력으로 가지는 모델에서는 사용하지 못하였지만, 논문에서 다중 레이어 LSTM을 이용한 모델을 제안하였다.

DNN은 입력과 타겟 데이터이 고정된 벡터의 크기를 가지고 있어서 적용이 어려웠는데, 이 모델은 LSTM을 이용하여 모델을 구현하였다.

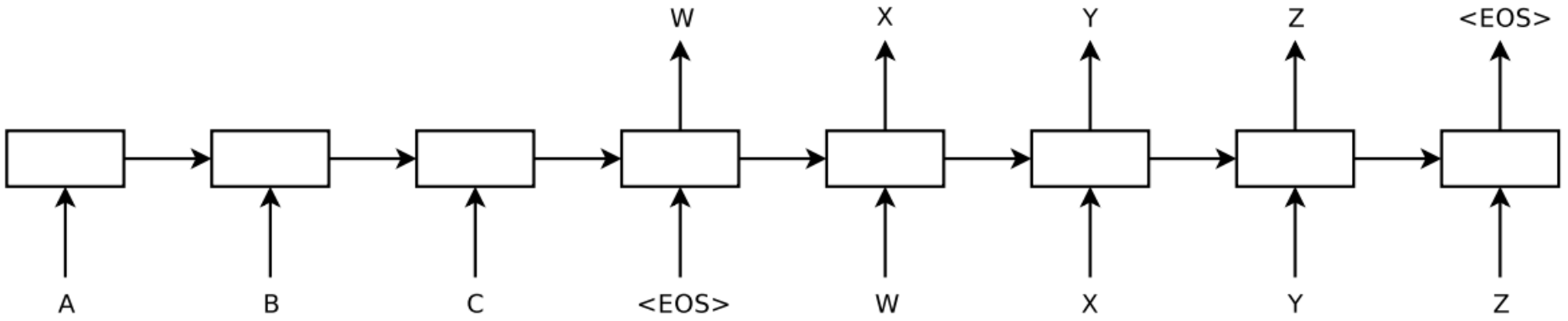


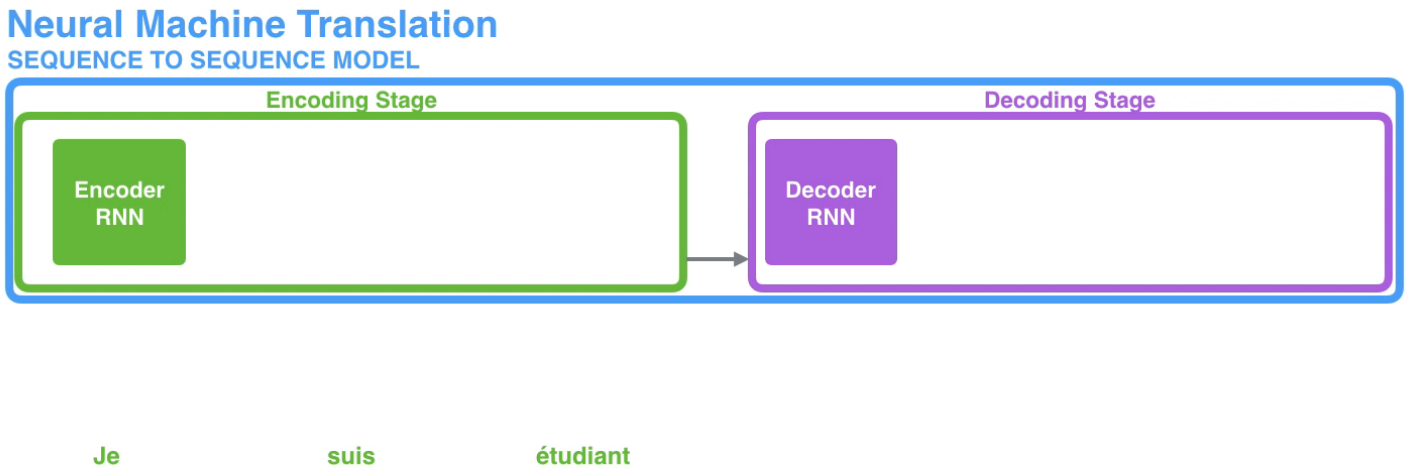

Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

## Seq2seq모델(SEQUENCE TO SEQUENCE LEARNING WITH NEURAL NETOWRKS)

RNN 구조를 입력을 hidden state 벡터를 통해 전달하고 이를 통해 출력을 생성해내는 방법을 사용하여 입력과 출력의 길이가 다른 문제를 해결할 수 있었지만 긴 문장에서의 성능이 떨어지고, 문장 내의 단어 간의 복잡한 관계를 파악하기 어려워 LSTM을 사용하였다.

$$h_t = \text{sigm}\left(W^{\text{hx}}x_t + W^{\text{hh}}h_{t-1}\right)$$
$$y_t = W^{\text{yh}}h_t$$

Hidden state는 이전 입력과 이전 hidden state를 통해 업데이트 되고 출력y는 hidden state를 통해 구할 수 있다. 마지막 입력을 받고 생성한 hidden state를 context벡터라고 한다. Context벡터를 통해 각 단어의 확률을 계산하고 이때 beam search를 사용하였다.



## Seq2seq모델(SEQUENCE TO SEQUENCE LEARNING WITH NEURAL NETOWRKS)

RNN 구조를 입력을 hidden state 벡터를 통해 전달하고 이를 통해 출력을 생성해내는 방법을 사용하여 입력과 출력의 길이가 다른 문제를 해결할 수 있었지만 긴 문장에서의 성능이 떨어지고, 문장 내의 단어 간의 복잡한 관계를 파악하기 어려워 LSTM을 사용하였다.

$$h_t = \text{sigm}\left(W^{\text{hx}}x_t + W^{\text{hh}}h_{t-1}\right)$$
$$y_t = W^{\text{yh}}h_t$$

Hidden state는 이전 입력과 이전 hidden state를 통해 업데이트 되고 출력y는 hidden state를 통해 구할 수 있다. 마지막 입력을 받고 생성한 hidden state를 context벡터라고 한다. Context벡터를 통해 각 단어의 확률을 계산하고 이때 beam search를 사용하였다.

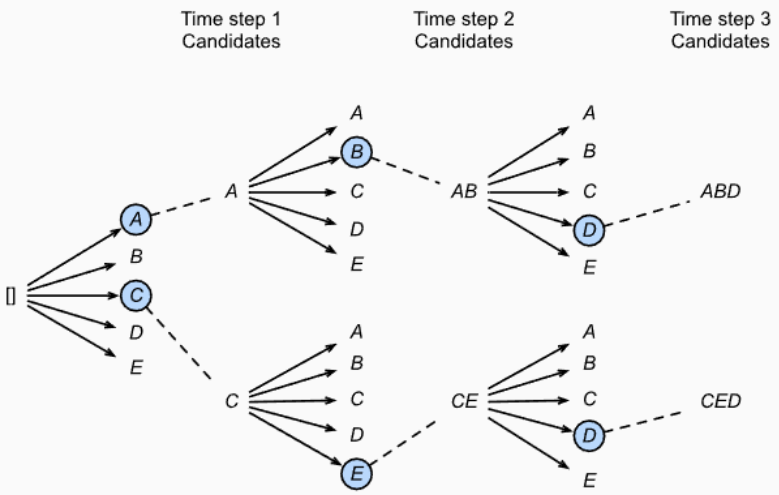



*Fig. 10.8.3* The process of beam search (beam size $= 2$; maximum length of an output sequence $= 3$). The candidate output sequences are $A$, $C$, $AB$, $CE$, $ABD$, and $CED$.

## Seq2seq모델(SEQUENCE TO SEQUENCE LEARNING WITH NEURAL NETOWRKS)

1. 두가지의 다른 LSTM을 사용해야 한다.
   - 하나는 입력데이터를 위해서 사용하고, 다른 하나는 출력 데이터를 위해 사용한다.
2. 깊은 LSTM구조가 얕은 LSTM 구조보다 성능이 좋다.
   - 4개의 레이어로 LSTM을 구성하였다.
3. 입력을 역순으로 주어줬을 경우 성능이 더 좋았다.

학습 설정

-학습 전 파라미터들은 -0.08과 0.08의 값으로 uniform 분포를 따르게 설정하고
-SGD를 사용, learning rate는 0.7로 설정하고 학습 후 5epoch이후로 계속 줄여 나감
-LSTM은 기울기 소실 문제보다 폭주에 더 신경 써야 하므로 어느 기준을 넘으면 clipping
-입력 문장의 크기가 다르므로 batch낭비되는 경우를 줄이기 위해 일정하게 길이를 맞추는 방법 사용
-데이터를 병렬적으로 처리하여 GPU가 매초 6300개의 단어를 처리할 수 있도록 설정

## **Bahdanau Attention**(Neural Machine Translation By Jointly Learning To Align And Translate)

이 논문에서는 이전의 encoder와 decoder로 이루어진 seq2seq 모델에서 고정된 context벡터에 문장을 압축 시키는 것과 LSTM을 사용하여도 장기기억 소실로 인해 긴 문장 번역시 성능저하 문제를 해결하기 위해 attention매커니즘을 도입한 모델을 제시

각 hidden state는 각 입력의 큰 영향을 받기 때문에 context 벡터 구성 시 모든 hidden state의 가중합을 이용하는 방법이다.

The context vector $c_i$ is, then, computed as a weighted sum of these annotations $h_i$:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

The weight $\alpha_{ij}$ of each annotation $h_j$ is computed by

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

where

$$e_{ij} = a(s_{i-1}, h_j)$$

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

I번째 출력의 context 벡터는 encoder의 모든 hidden state와 디코더의 i-1번째의 hidden state를 사용하여 $e_{ij}$를 구성하고 $\alpha_{ij}$ 를 구하여 구성한다.

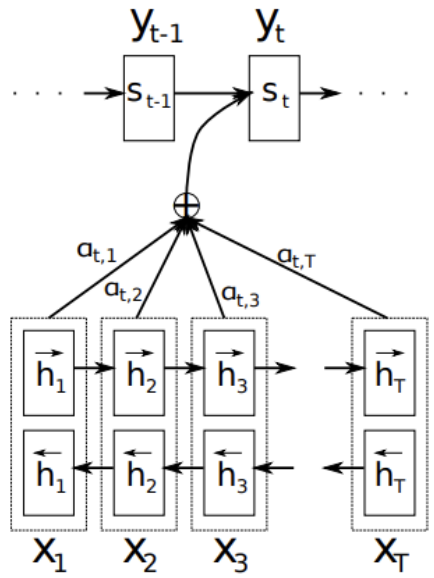


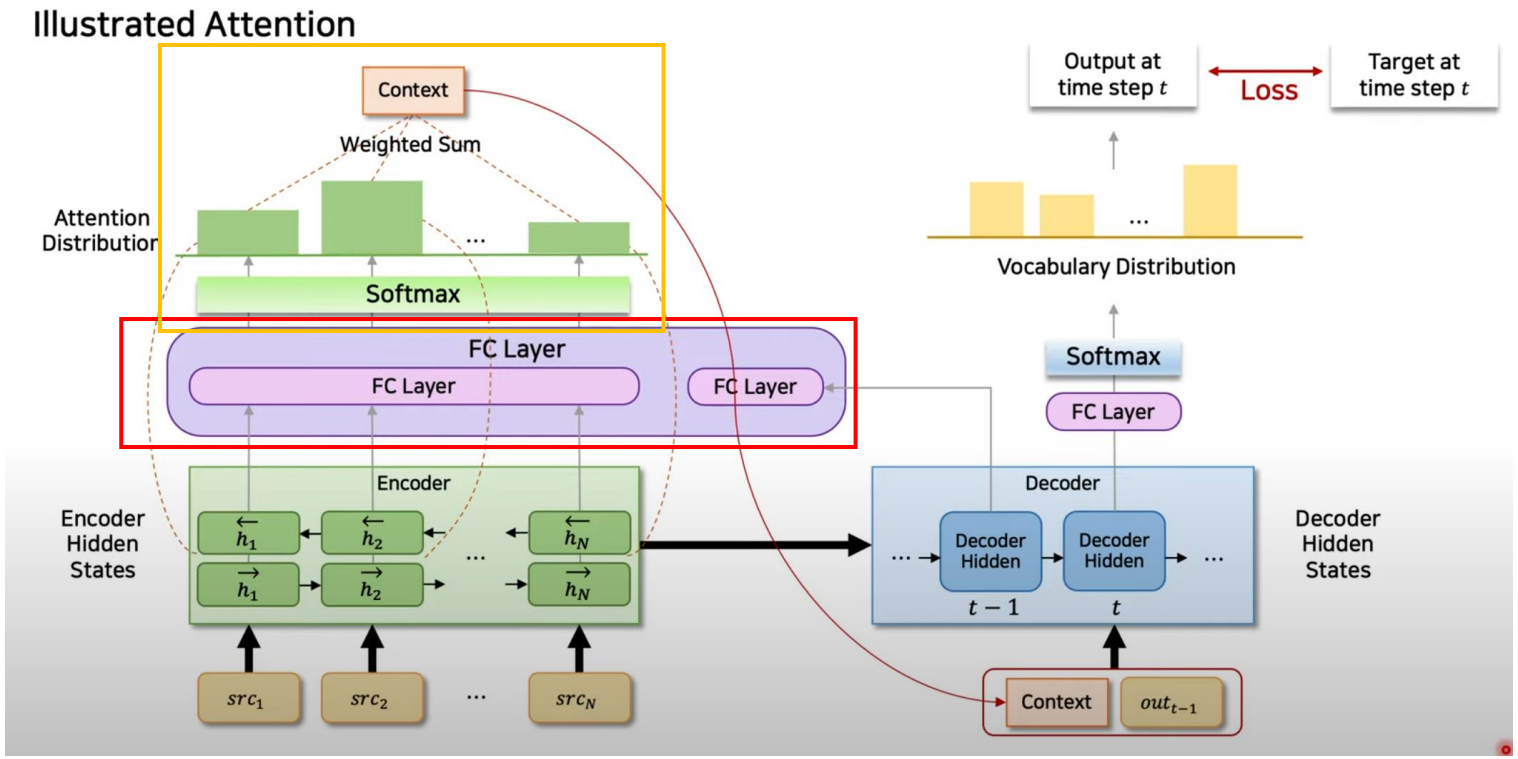

Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$.

Encoder는 BiRNN이라는 것을 사용하게 되는데 BiRNN이란 입력의 순방향과 역방향을 고려하여 hidden state를 구하여 합치는 방법이다.

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$

## **Bahdanau Attention**(Neural Machine Translation By Jointly Learning To Align And Translate)

T번째의 출력을 계산할 경우를 설명하는 그림으로 t-1번째 decode의 hidden state와 encoder의 hidden state각각 비선형 변환을 시키고 이후 합쳐서 입력 벡터와 같은 크기의 비선형 변환을 시켜주면 입력 벡터의 가중치를 구할 수 있게 된다. 이를 바탕으로 hidden state와 가중합을 통해 contex벡터를 구하고 이를 t번째 decode의 입력으로 같이 주어줘서 출력 단어의 벡터들 중 가장 높은 확률을 가진 단어를 선택




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

The context vector $c_i$ is, then, computed as a weighted sum of these annotations $h_i$:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight $\alpha_{ij}$ of each annotation $h_j$ is computed by

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

where

$$e_{ij} = a(s_{i-1}, h_j)$$

## **Bahdanau Attention(Neural Machine Translation By Jointly Learning To Align And Translate)**

단어를 대치하는 방법인 hard alignment는 번역에 도움이 되지 않지만, 논문에서 채택한 soft alignmnet방법은 같은 단어라도 다른 의미로 해석하는 등의 번역에서의 용의성과 문장의 길이가 다른 경우에도 좋은 결과를 보여줬다. 또한 attention 매커니즘을 사용하므로 encode가 고정된 context벡터에 완벽히 압축하지 않아도 돼 이전 seq2seq모델의 병목 과정을 해결할 수 있었다.

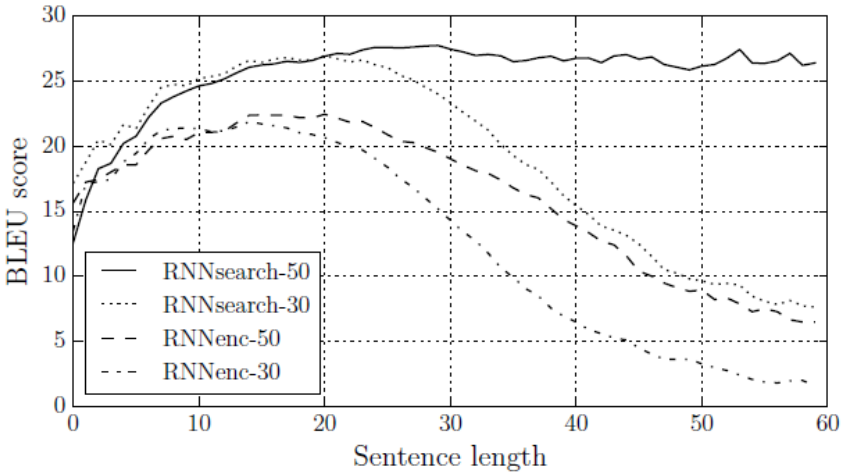


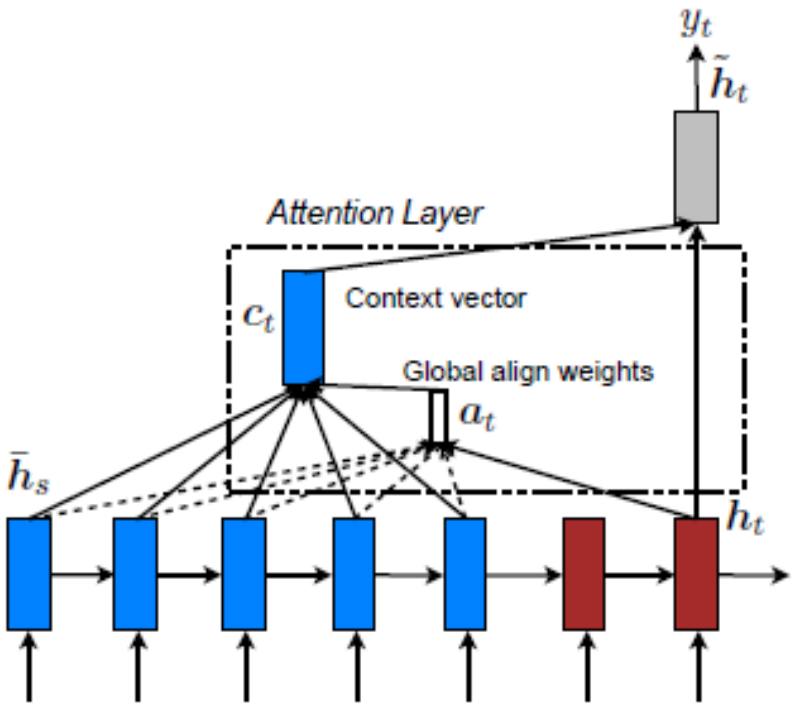

Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

긴 문장을 학습한 경우에는 긴 문장에서도 좋은 성능을 보였다. 또한 프랑스-영어 번역에서 어순이 다른 경우에도 좋은 성능을 보였다.

## Luong Attention(Effective Approaches to Attention-based Neural Machine Translation)

2015년 Bahdanau가 제시한 것과 비슷한 attention기반의 언어모델을 제안하였는데 모델 구성이 단순하고 성능차이가 비슷하여 Luong attention을 많이 사용한다.

Luong attention은 global과 local방법이 있는데, global 방식은 입력 문장의 모든 것을 고려하는 방법이고, local 방식은 입력된 문장의 부분 집합만 사용하여 attentional 벡터를 구성하는 방식이다.




Global attention은 t 시점의 t시점의 hidden state와 입력의 hidden state의 유사도를 구하고 softmax한 값을 가중치로 하여 가중합을 계산하여 context벡터를 구하고

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned} \qquad \text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{W}_a \boldsymbol{h}_t) \qquad \textit{location} \qquad (8)$$

Context벡터와 t시점의 hidden state를 바탕으로 예측하게 된다.

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

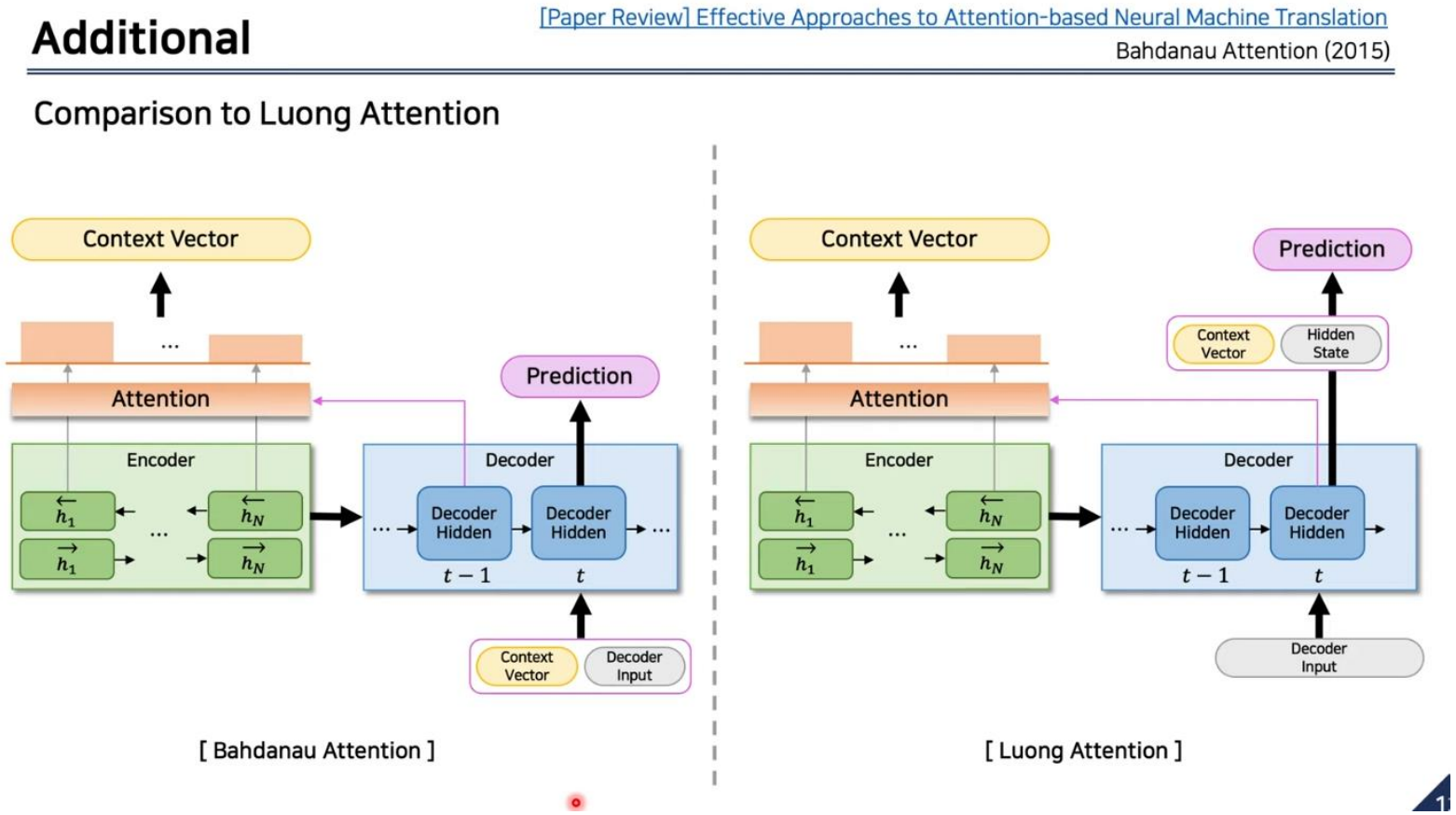
## Luong Attention(Effective Approaches to Attention-based Neural Machine Translation)

Global attention방법은 bahdanau 방법과 비슷한 접근법이지만 몇가지 다른 점이 있다.

-T시점의 출력을 결정하기 위해 bahdanau는 t-1번 째의 hidden state를 사용하였고, luong은 t시점의 hidden state를 사용하여 계산 과정이 간단하다.

-Luong논문에서는 단순한 LSTM을 여러 층으로 쌓은 모델에서 가장 위층의 LSTM 유닛의 hidden state만 사용하였다.

-bahdanau는 추가적인 deep out, max-out을 위한 추가적인 층을 만들어 계산이 비효율적이다.




Bahdanau는 h_t -1-〉 a_t -〉 c_t -〉 h_t

Luong은 h_t -〉 a_t -〉 c_t -〉 $\widetilde{h_t}$ 과정을거치게 되는데, Bahdanau의 과정에서는 deep-output과maxout layer와 같은 추가적인 비선형 변환 연산을 수행하므로 비효율적이다.

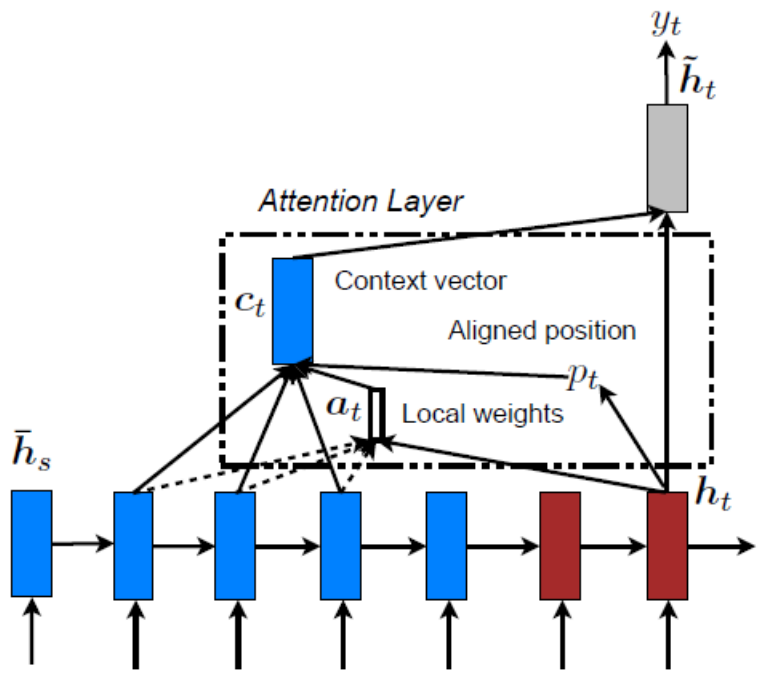
## Luong Attention(Effective Approaches to Attention-based Neural Machine Translation)

Local attention은 global attention에서 모든 입력에 대해 연산하는 문제점을 해결하기 위해 window를 통해 입력의 몇몇 hidden state만 고려하는 방법이다.

Window를 적용시킬 위치 P_t에 대해 window 크기 D를 바탕으로 [P_t-D,P_t+D]의 hidden state를 이용하여 연산하게 된다.

이 아이디어는 이미지 자막 생성에 관한 논문에서 soft attention을 적용하여 모든 이미지 패치에 대해 고려했을 경우와, hard attention을 이용하여 정해진 하나의 이미지 패치에 적용하였을 경우의 상충관계를 보고 이를 융합한 방법을 고안한 것이다.

| | window | 연산량 | 미분 |
|---|---|---|---|
| soft | 모든 단어 | 복잡 | o |
| hard | 1개 | 덜 복잡 | x |
| local | 여러 개 | 덜 복잡 | o |




Local attention의 경우 참조하는 hidden state의 길이가 정해져 있으므로 a_t(s)는 고정된 벡터이고 global attention 보다 짧아 지게 된다.

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$
$$= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

## Luong Attention(Effective Approaches to Attention-based Neural Machine Translation)

P_t는 window를 적용할 입력 hidden state의 위치를 나타내게 되는데 이를 계산하기 위한 두가지 방법이 존재한다.

**단조로운 정렬(local-m):** 입력과 출력의 복잡한 관계가 없고 비슷한 위치의 입출력이 비슷한 의미를 가진 경우를 가정

P_t = t이고 a_t(s)는 이전과 같다.

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

**예측 정렬(local-p):** 단조로운 정렬의 가정 대신 P_t의 위치를 예측하는 방법으로 W_p, v_p의 파라미터를 추가하여 P_t의 위치를 학습하고 예측하는 방법이다. a_t(s)는 주변단어들이 중요하다는 가정으로 정규분포를 따르게 정렬하여 이전의 a_t(s)의 표준편차를 D/2로 가지게 변경하여 사용하였다.(P_t는 실수, s는 정수이다)

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W}_p \boldsymbol{h}_t)), \qquad \boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

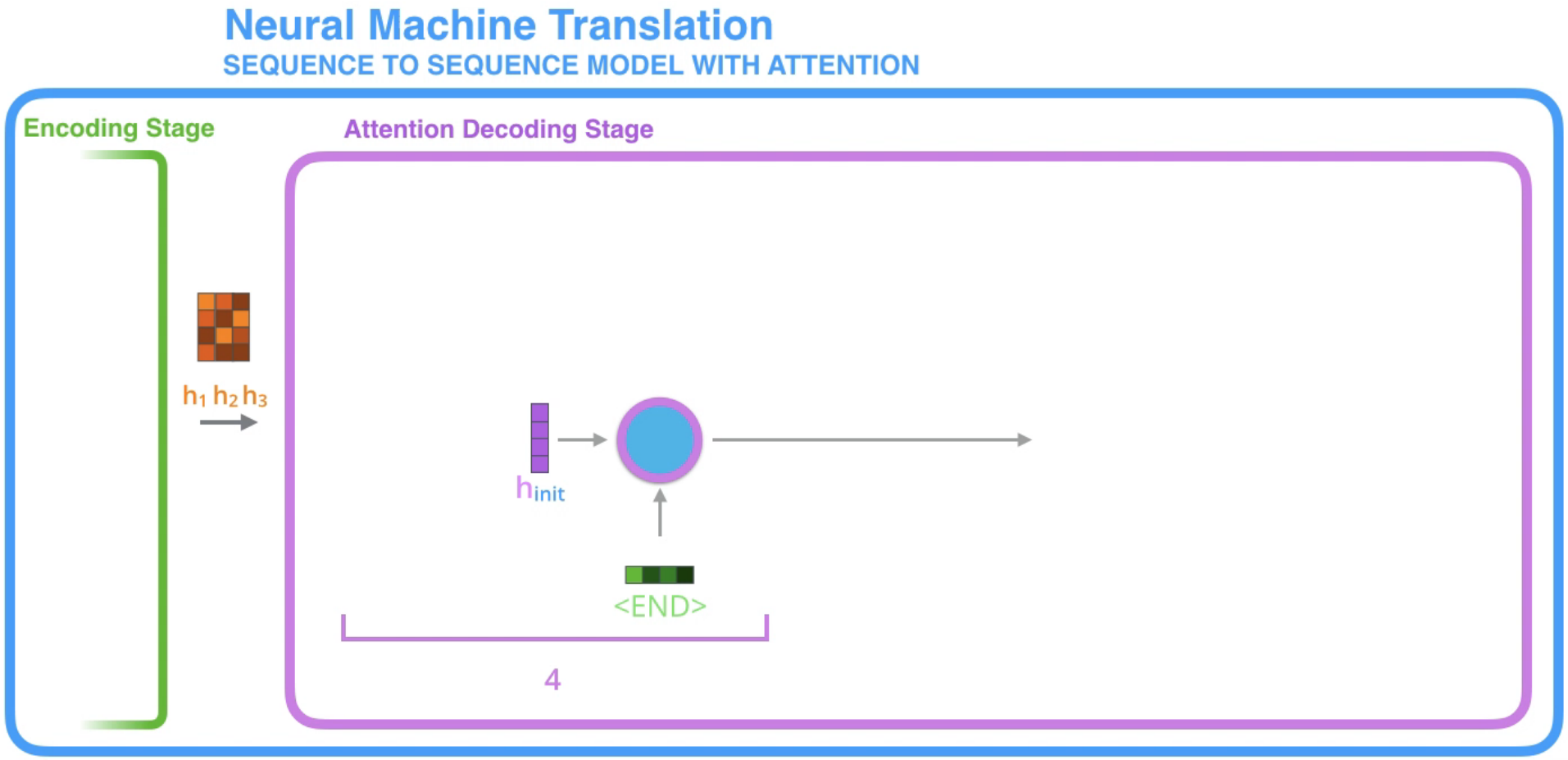
## Luong Attention(Effective Approaches to Attention-based Neural Machine Translation)

결과

-역순 입력, dropout, global attention, input feeding을 적용한 경우 계속해서 성능은 좋아졌다

| System | Ppl | BLEU |
|---|---|---|
| Winning WMT'14 system – *phrase-based + large LM* (Buck et al., 2014) | | 20.7 |
| *Existing NMT systems* | | |
| RNNsearch (Jean et al., 2015) | | 16.5 |
| RNNsearch + unk replace (Jean et al., 2015) | | 19.0 |
| RNNsearch + unk replace + large vocab + *ensemble* 8 models (Jean et al., 2015) | | **21.6** |
| *Our NMT systems* | | |
| Base | 10.6 | 11.3 |
| Base + reverse | 9.9 | 12.6 (+*1.3*) |
| Base + reverse + dropout | 8.1 | 14.0 (+*1.4*) |
| Base + reverse + dropout + global attention (*location*) | 7.3 | 16.8 (+*2.8*) |
| Base + reverse + dropout + global attention (*location*) + feed input | 6.4 | 18.1 (+*1.3*) |
| Base + reverse + dropout + local-p attention (*general*) + feed input | 5.9 | 19.0 (+*0.9*) |
| Base + reverse + dropout + local-p attention (*general*) + feed input + unk replace | | 20.9 (+*1.9*) |
| *Ensemble* 8 models + unk replace | | **23.0** (+*2.1*) |

## Seq2seq model with attention 설명 이미지



## 참고자료


정민성 – Attention papers 1 (Bahdanau& Luong attention)

[Paper Review] Neural Machine Translation by Jointly Learning to Align and Translate

Jay Alammer – attention