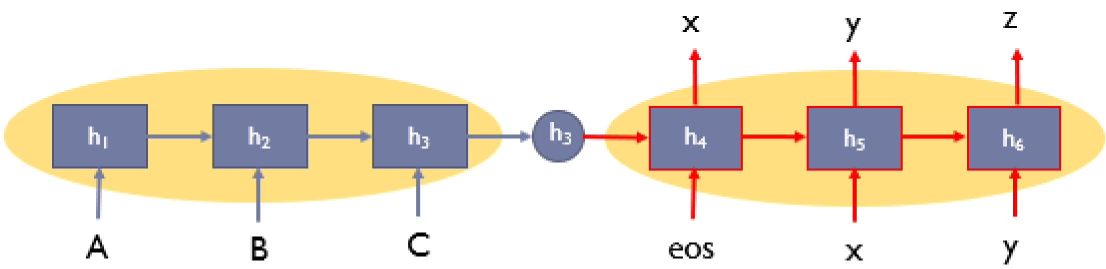
Attention Model

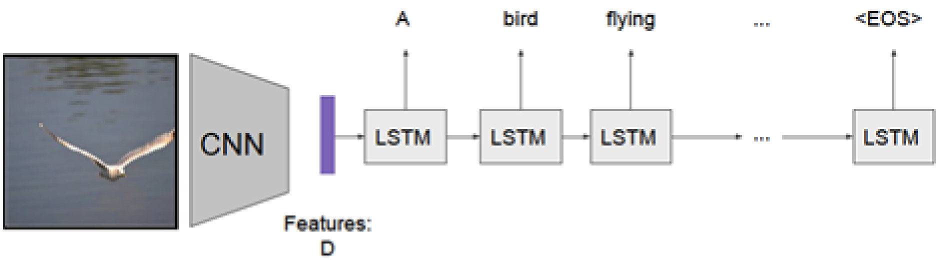
Sequence Generation

Encoder-Decoder Scheme

- Encoder : compress input sequence into one vector
    - h3 is the vector respresentation of the given sequence
- Decoder : uses this vector to generate output
    - It extracts necessary information only from the vector



- Rnns or CNNs can be used as Encoders
- RNNs are usually used as Decoders

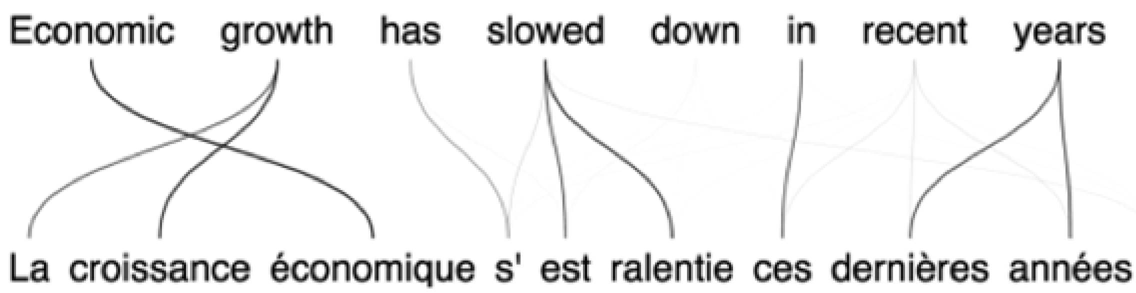


Challenges

- Hard for encoder to compress the whole source sentence into a single vector
- performance is degraded as the length of sentence increases
- A single vector may not enough for decoder to generate correct words

Observation

- At every step, all the inputs are not equally useful



- **Inputs relevant to the context may be more useful**

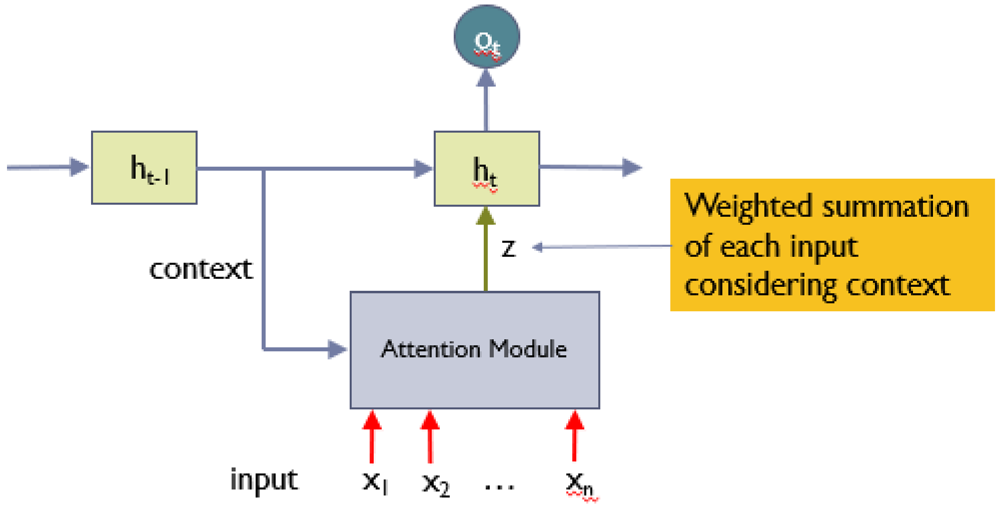
Attention Model

앞서 배운 seq2seq 모델은 **인코더**에서 입력 시퀀스를 컨텍스트 벡터라는 하나의 고정된 크기의 벡터 표현으로 압축하고, **디코더**는 이 컨텍스트 벡터를 통해서 출력 시퀀스를 만들어냈습니다.

하지만 이러한 RNN에 기반한 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.

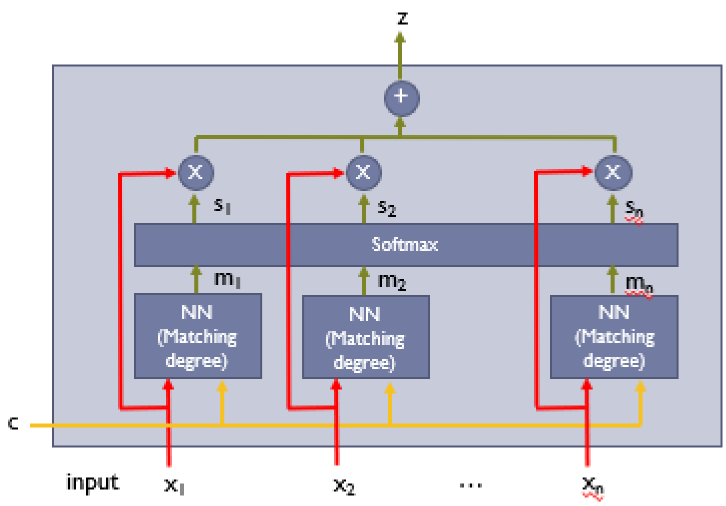
Overview



이 attention module은 decoder 부분에 있다.

Attention Module

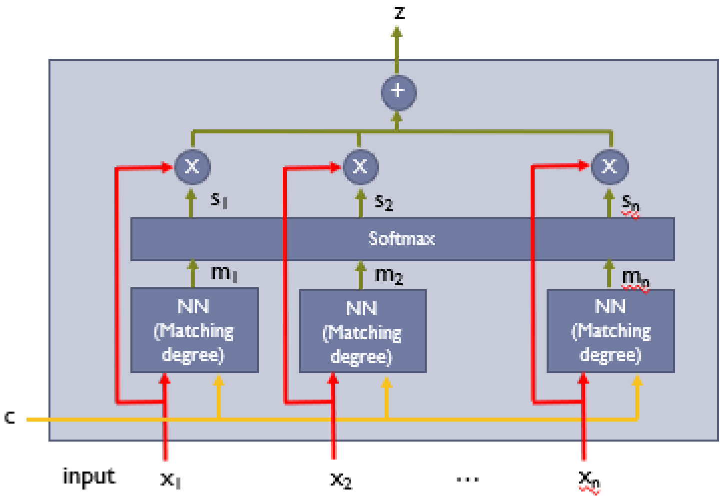
- All inputs share the same NN for matching degree



step 1 : Evaluation Matching Degree

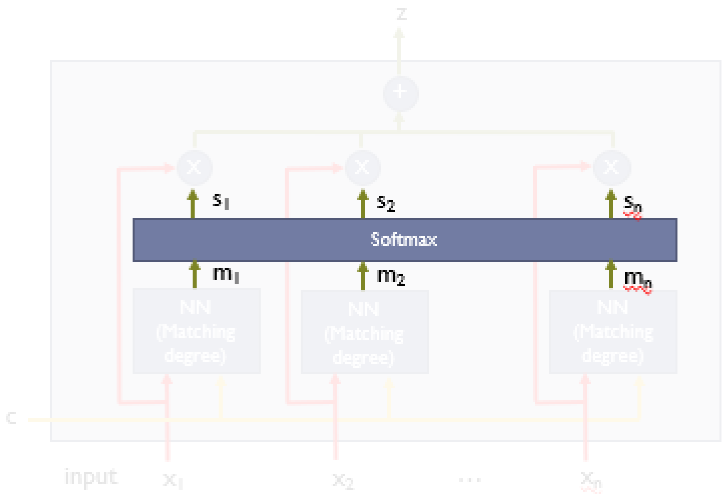
- Evaluating matching degree of each input to the context
  - Produce scalar matching degree(Higher value is higher attention)

- All inputs share the same NN
- m들은 0~1사이값 (따라서 activation func는 sigmoid)



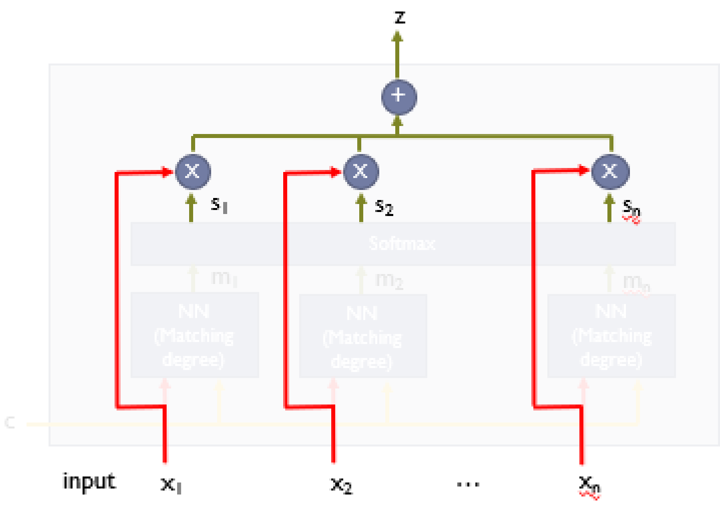
Step 2 : Normalizing Matching Degree

$$s_i = \frac{\exp(m_i)}{\sum_j exp(m_j)}$$



Step 3 : Agngregating Inputs

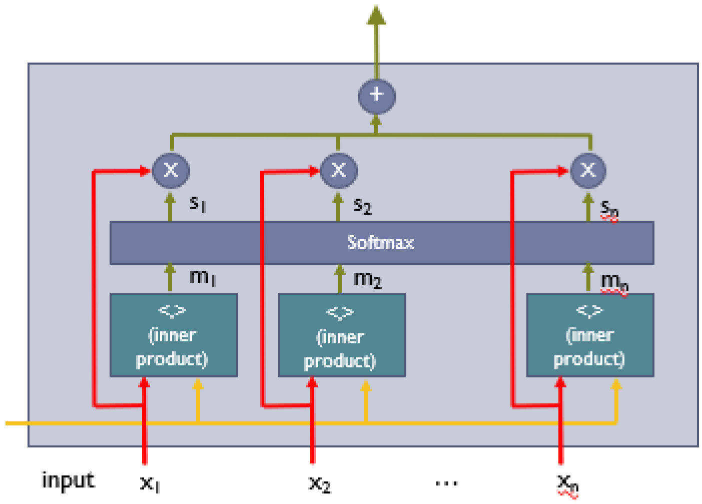
- Each input is scaled by si and summed up into z
- z is the input focused on the current context



이 z가 그대로 ht에 가는게 아니라 여러가지 방법이 있다.

Variation

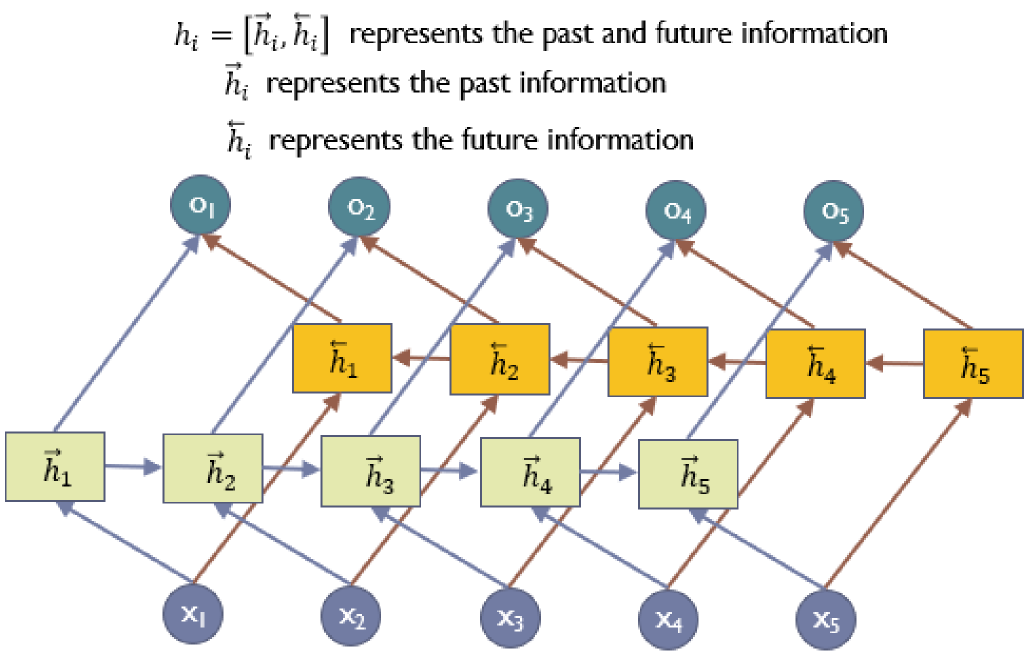
- Matching NN can be replaced with the inner products of inputs and context

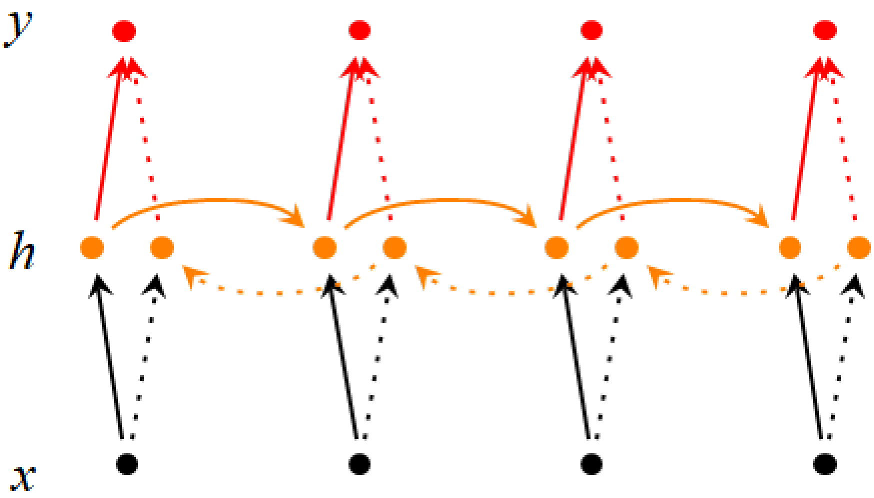


내적값은 두 벡터의 similarity와 비례한다.

백터의 내적이므로 학습이 따로 필요없다.

Bidirectional LSTM



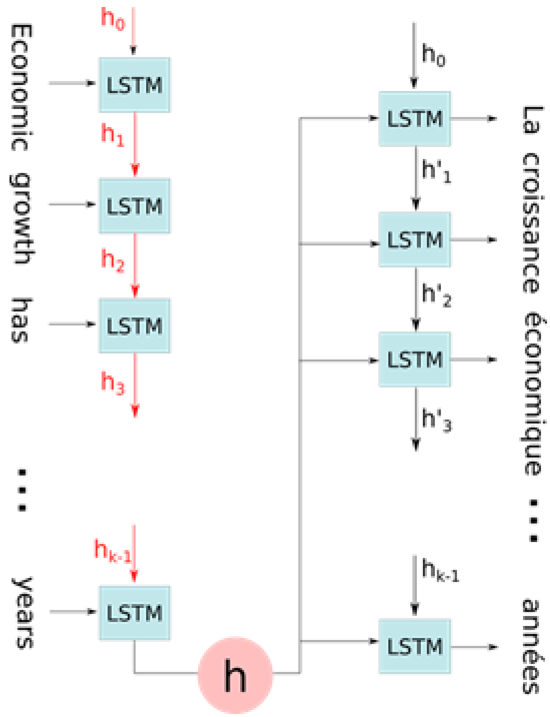


$$\overrightarrow{h}_t = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

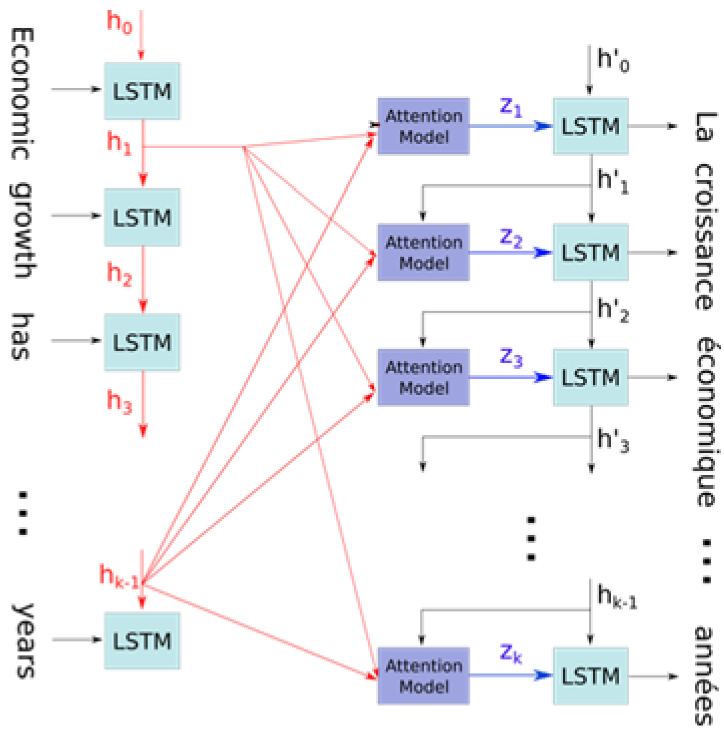
$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t] + c)$$

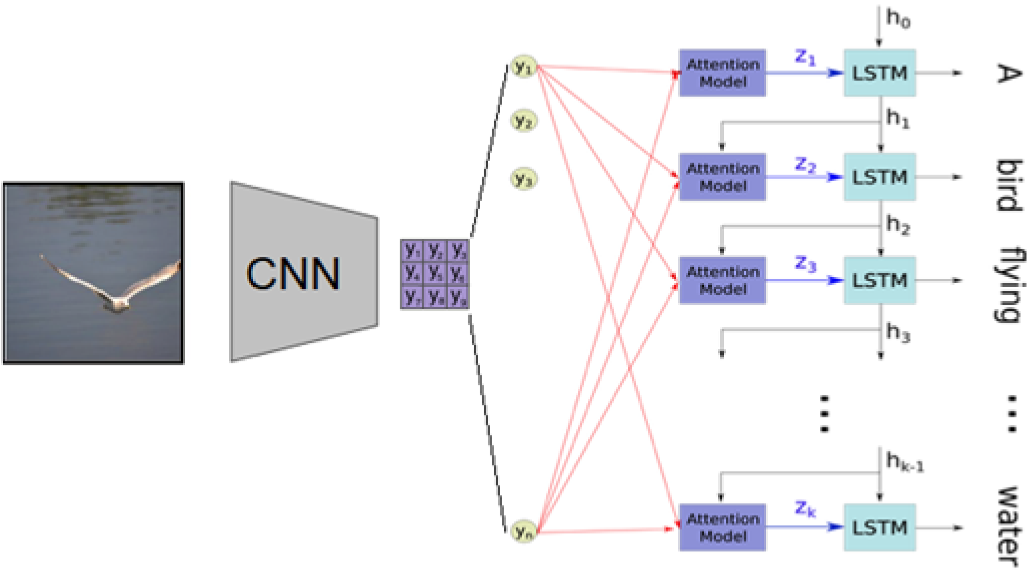
Example

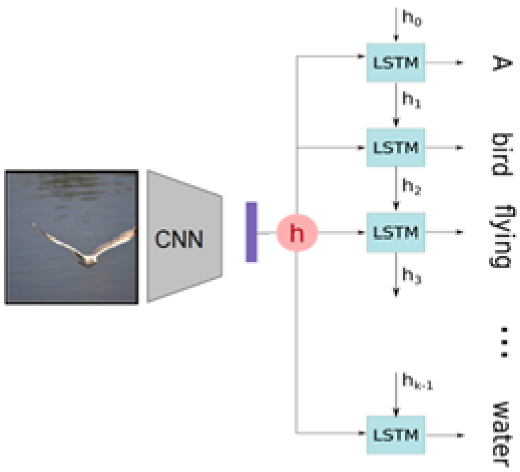


Encoder-decoder model



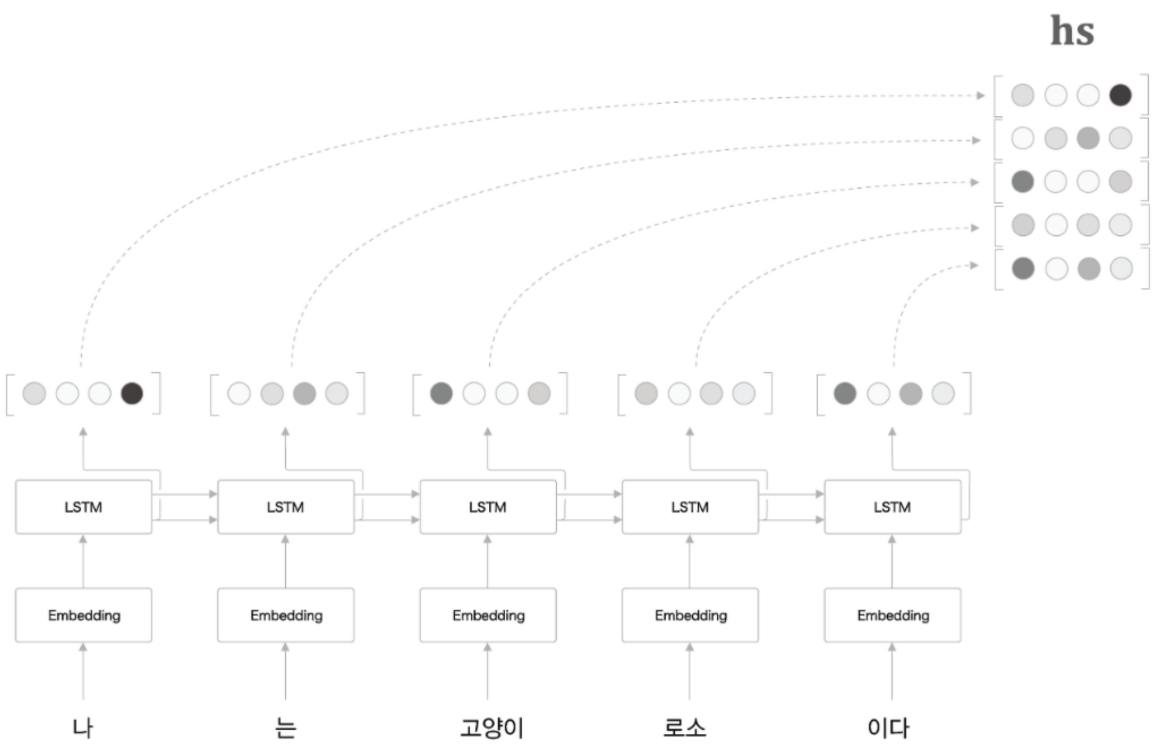
Attention based model

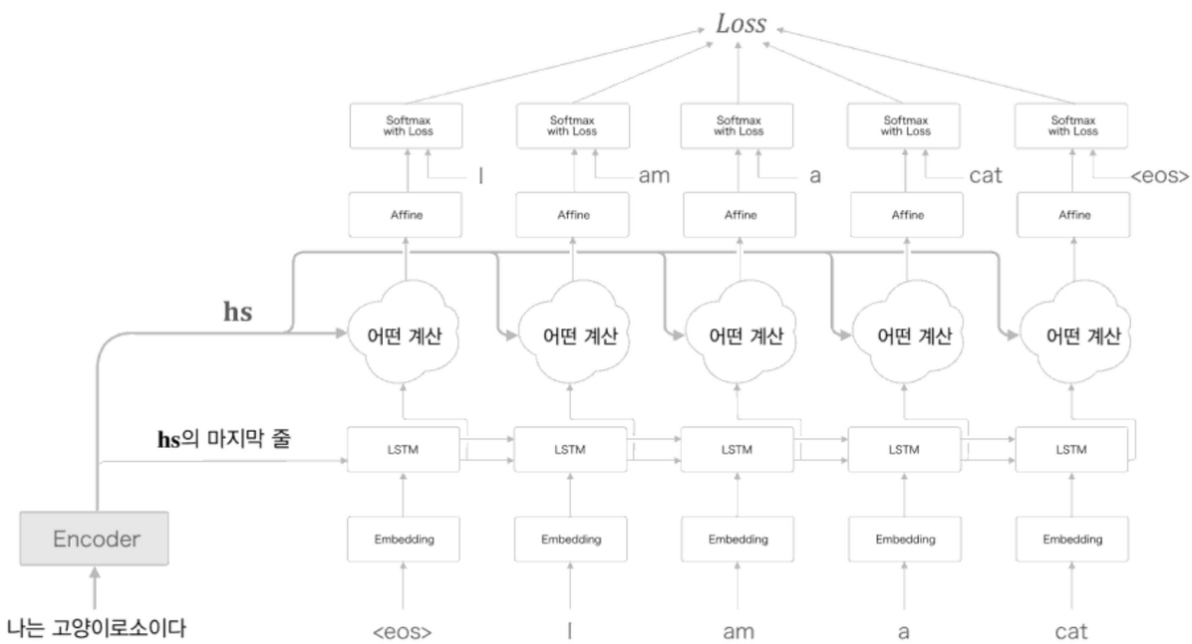




Attention is Great!

- Attention significantly improves NMT(Neural Machine Translation) performance.
  - It's very useful to allow decoder to focus on certaion parts of the source.
- Attention solves the bottleneck problem.
  - Attention allows decoder to look directly at sources:bypass bottleneck.
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states.
- Attention provides some interpretability


hs
LSTM
Embedding
나
는
고양이
로소
이다


Loss
Softmax with Loss
I
am
a
cat
<eos>
Affine
hs
어떤 계산
hs의 마지막 줄
LSTM
Embedding
Encoder
나는 고양이로소이다
<eos>
I
am
a
cat