

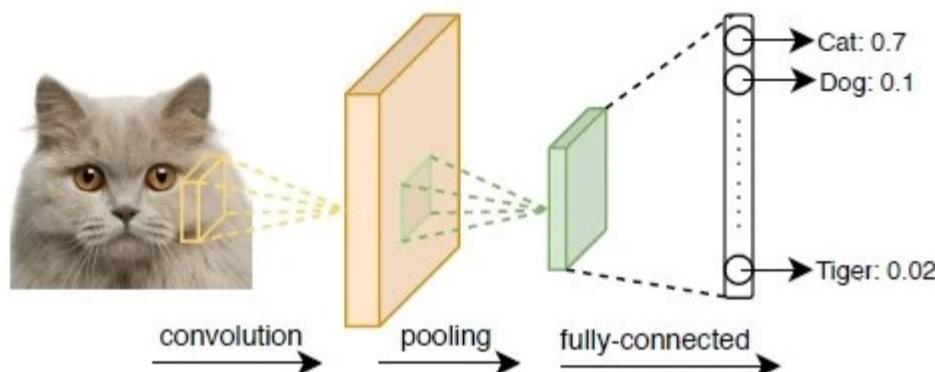
Neural Machine Translation by Jointly Learning to Align and Translate

☰ Author	Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio
☰ Field	NLP
⊖ Journal	NIPS
# Published Year	2014
☰ Speaker	오원준 이승호
☰ Summary	Attention
⊖ status	Finished!
☰ Q&A	1. key와 value 다른 모델이 뭐 있지?
🔗 link	https://arxiv.org/pdf/1409.0473.pdf

Background

기존의 문제들(one-to-one)

Convolutional Neural Network



만약 input data가 시퀀스라면?

- 데이터의 시간 순서가 중요해짐

시퀀스 데이터의 예시

- sequence of images



- sequence of words

Sequential data refers to a type of data where the order and arrangement of individual elements or items hold significance.

- sequence of numbers



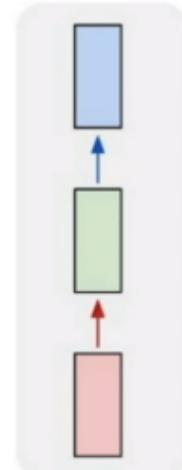
output도 시퀀스가 될 수 있음.

problem types

one to one

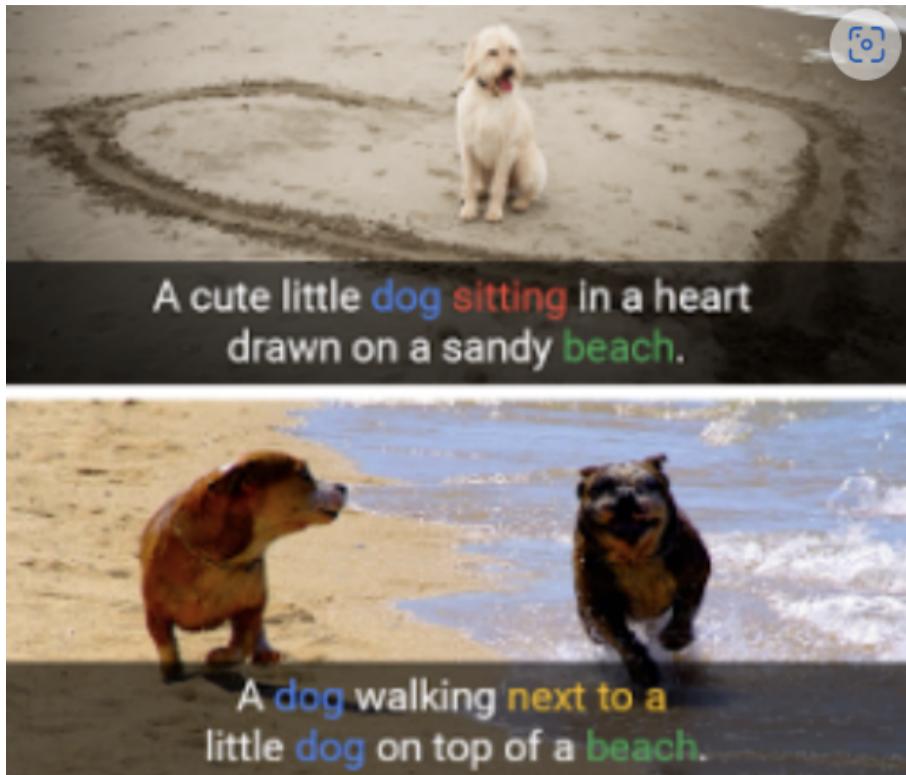
- image classification

one to one

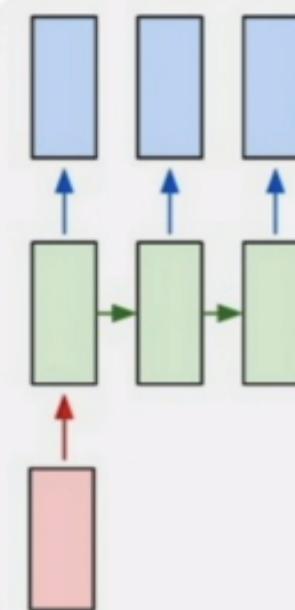


one to many

- image captioning



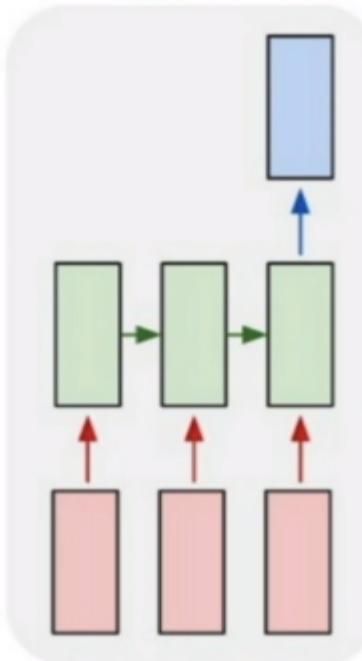
one to many



many to one

- sentiment classification
- frame classification

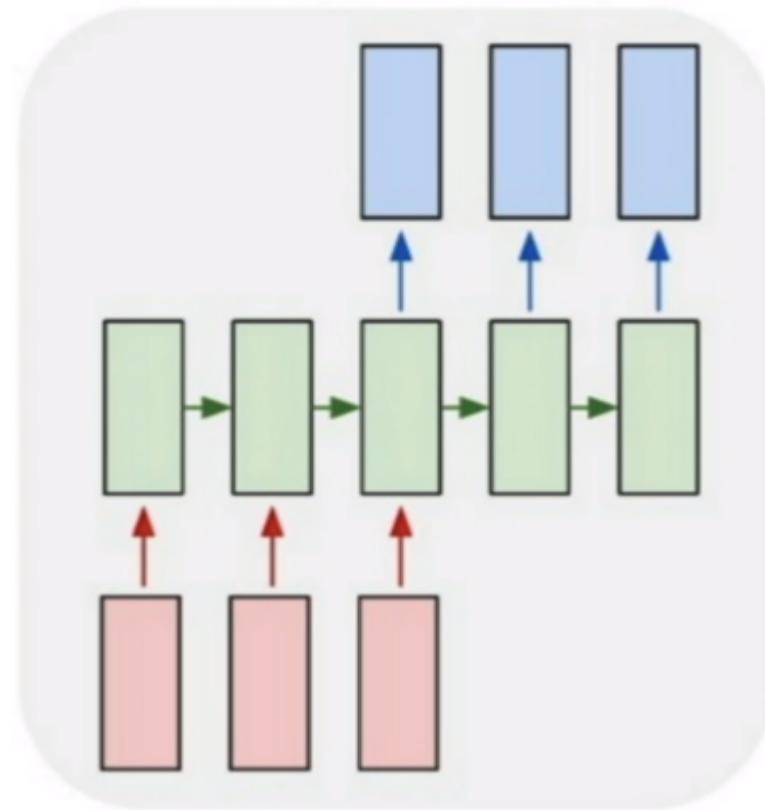
many to one



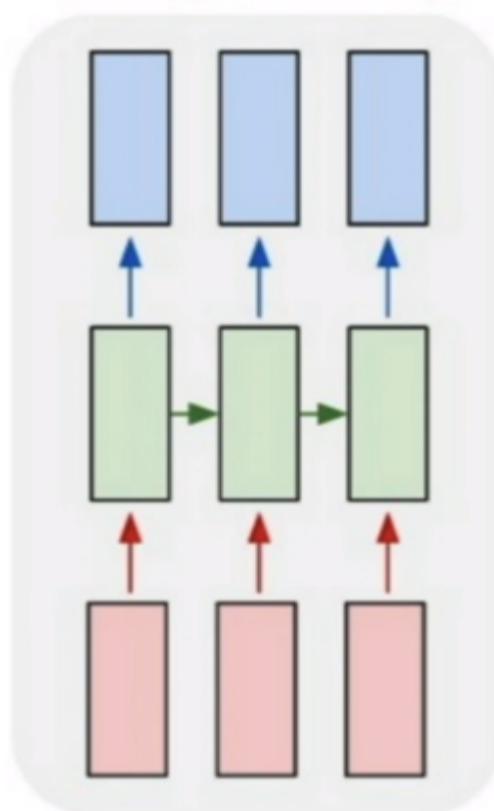
many to many

- machine translation
- video classification on frame level

many to many

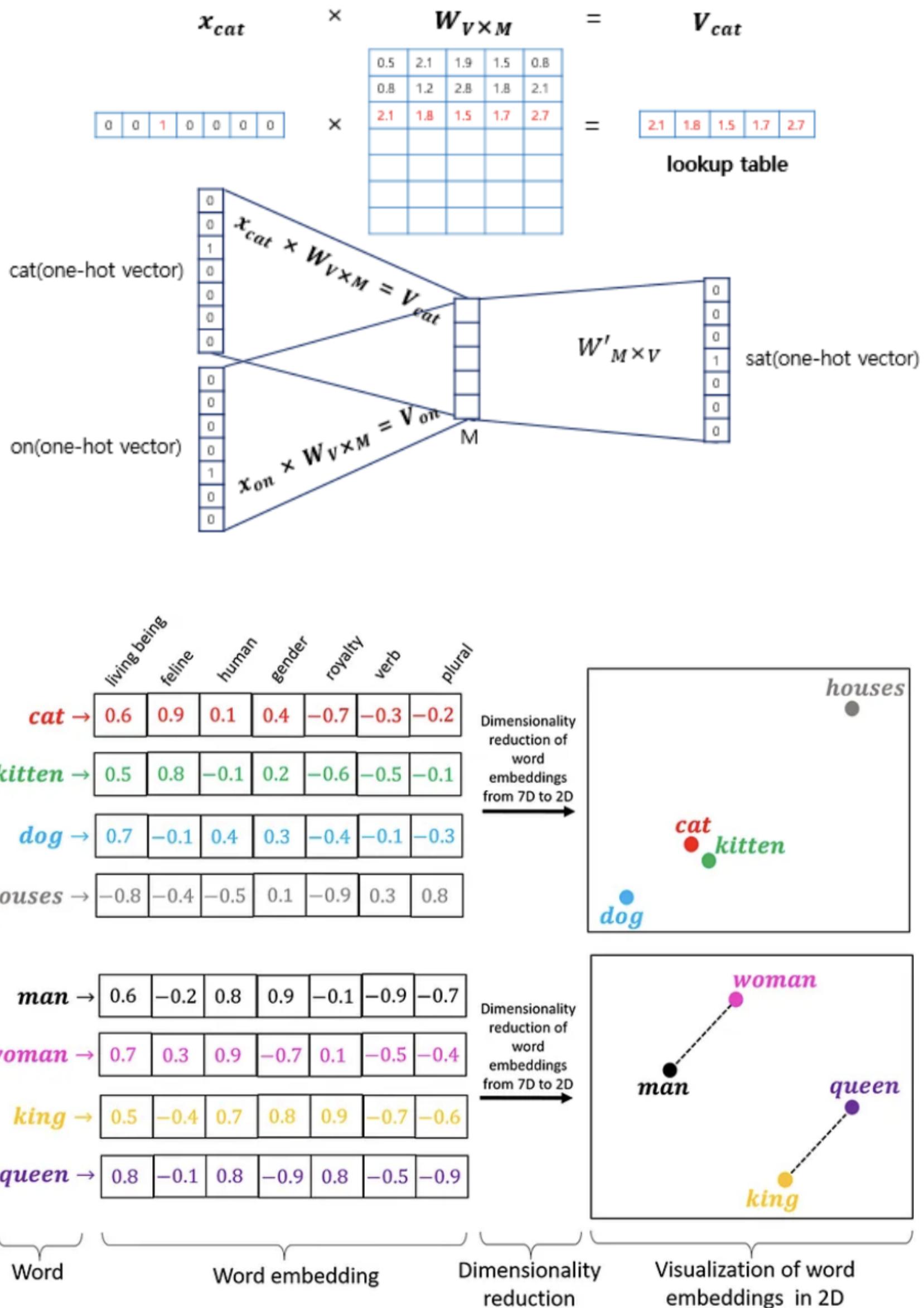


many to many

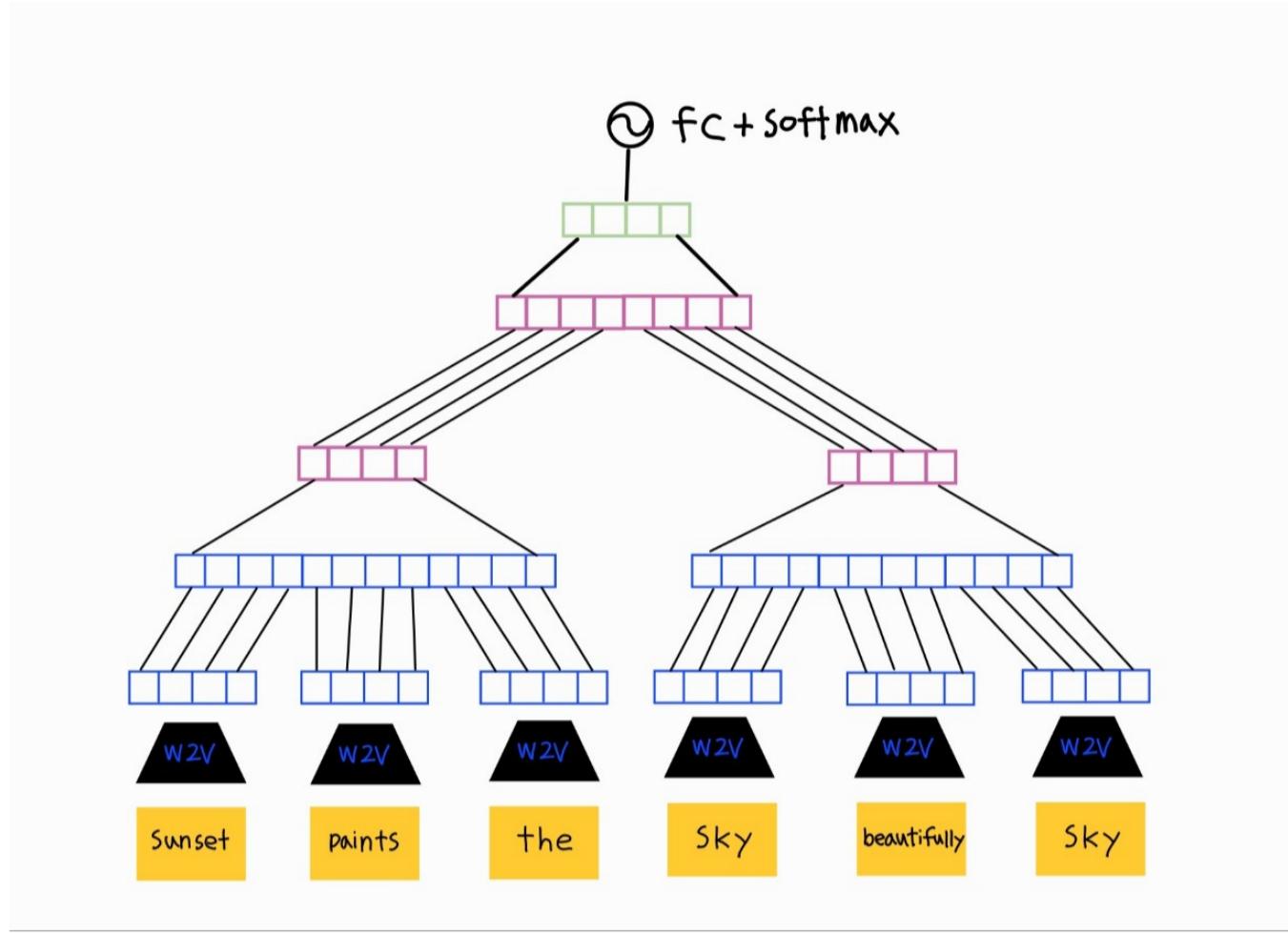


word2vec

단어를 벡터로 표현해보자

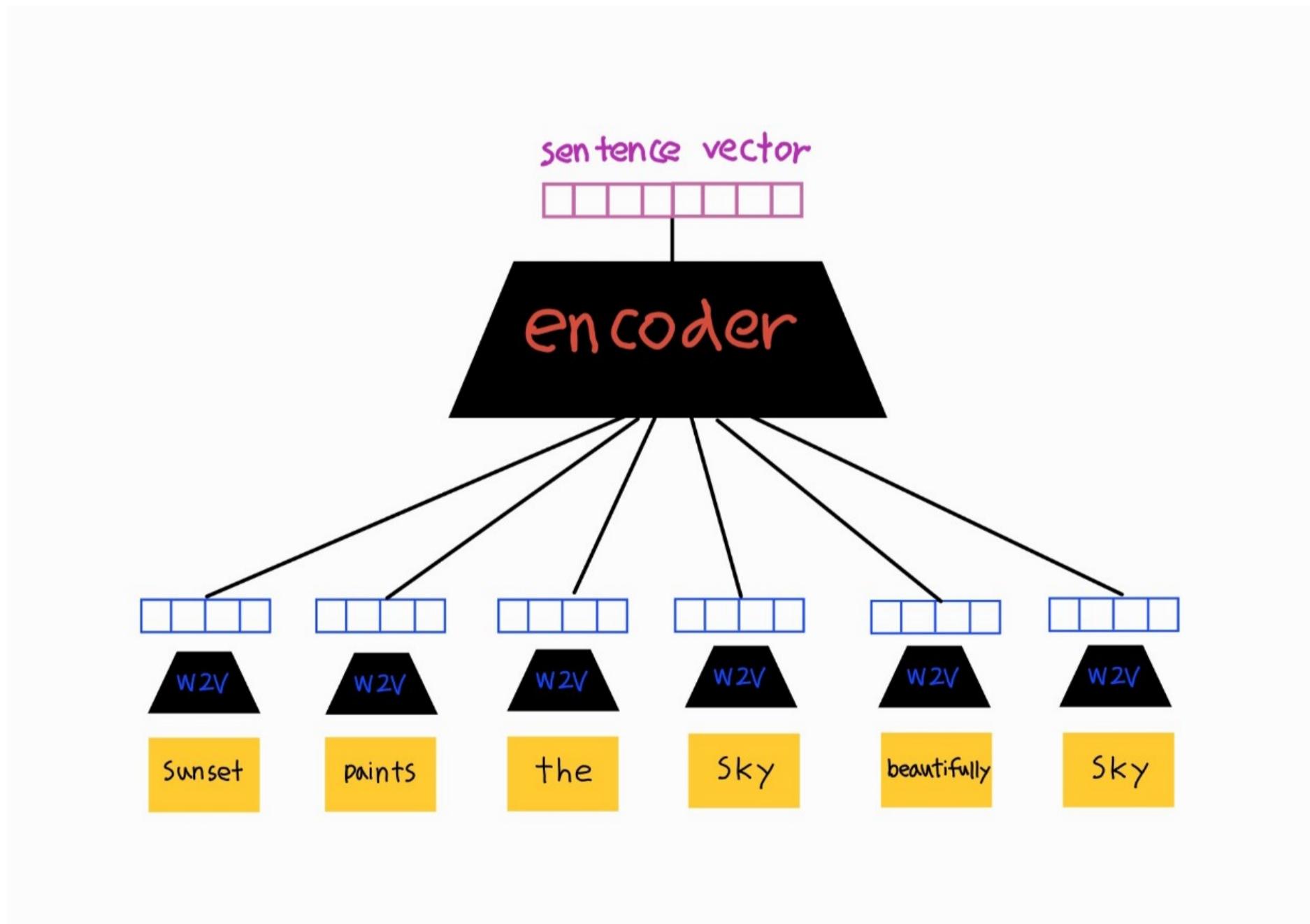


CNN으로 학습

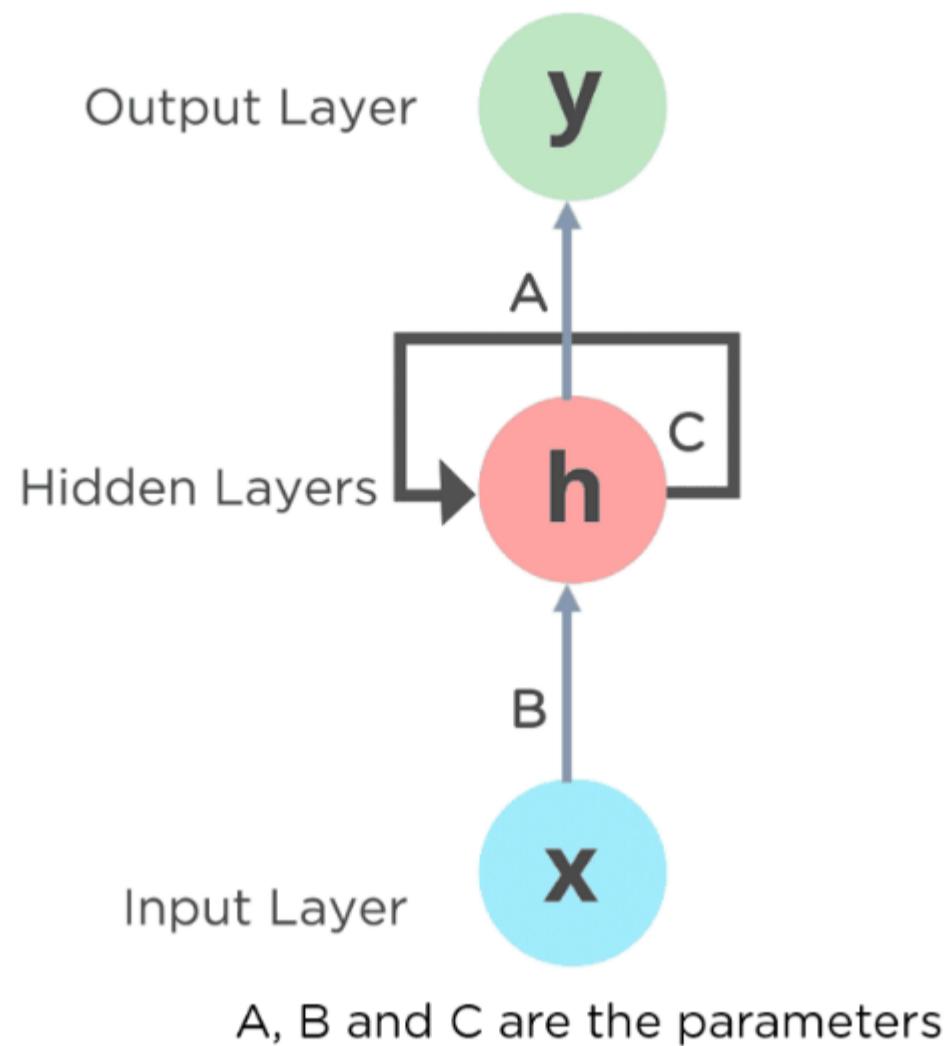


이렇게 하면 문제점?

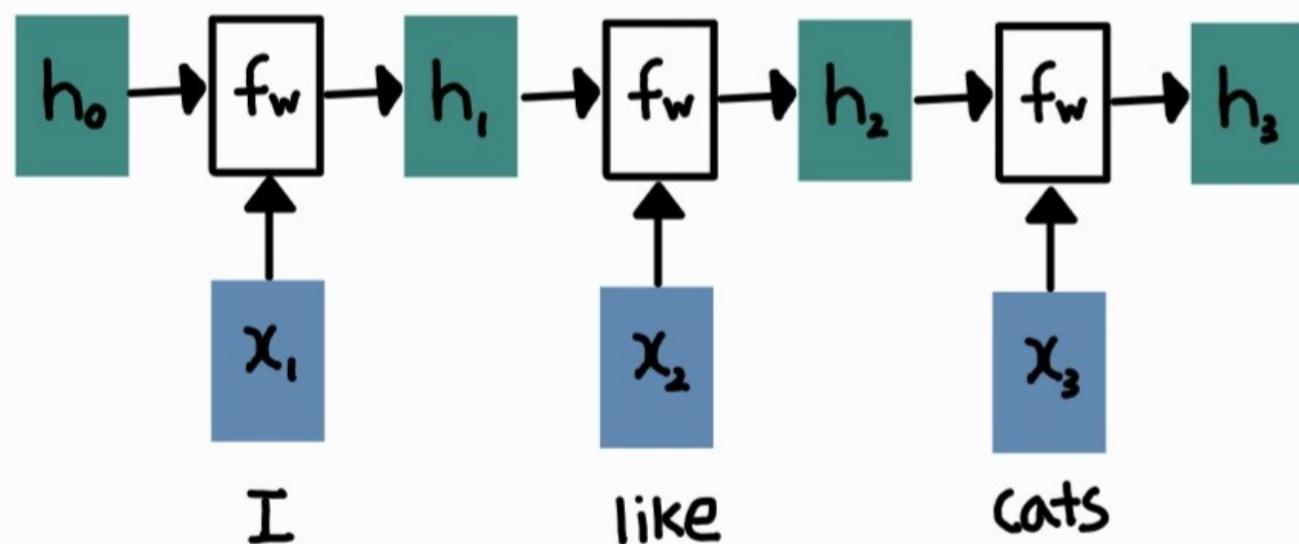
- 크기 문제
- 이미지 같은 경우는 괜찮음



RNN



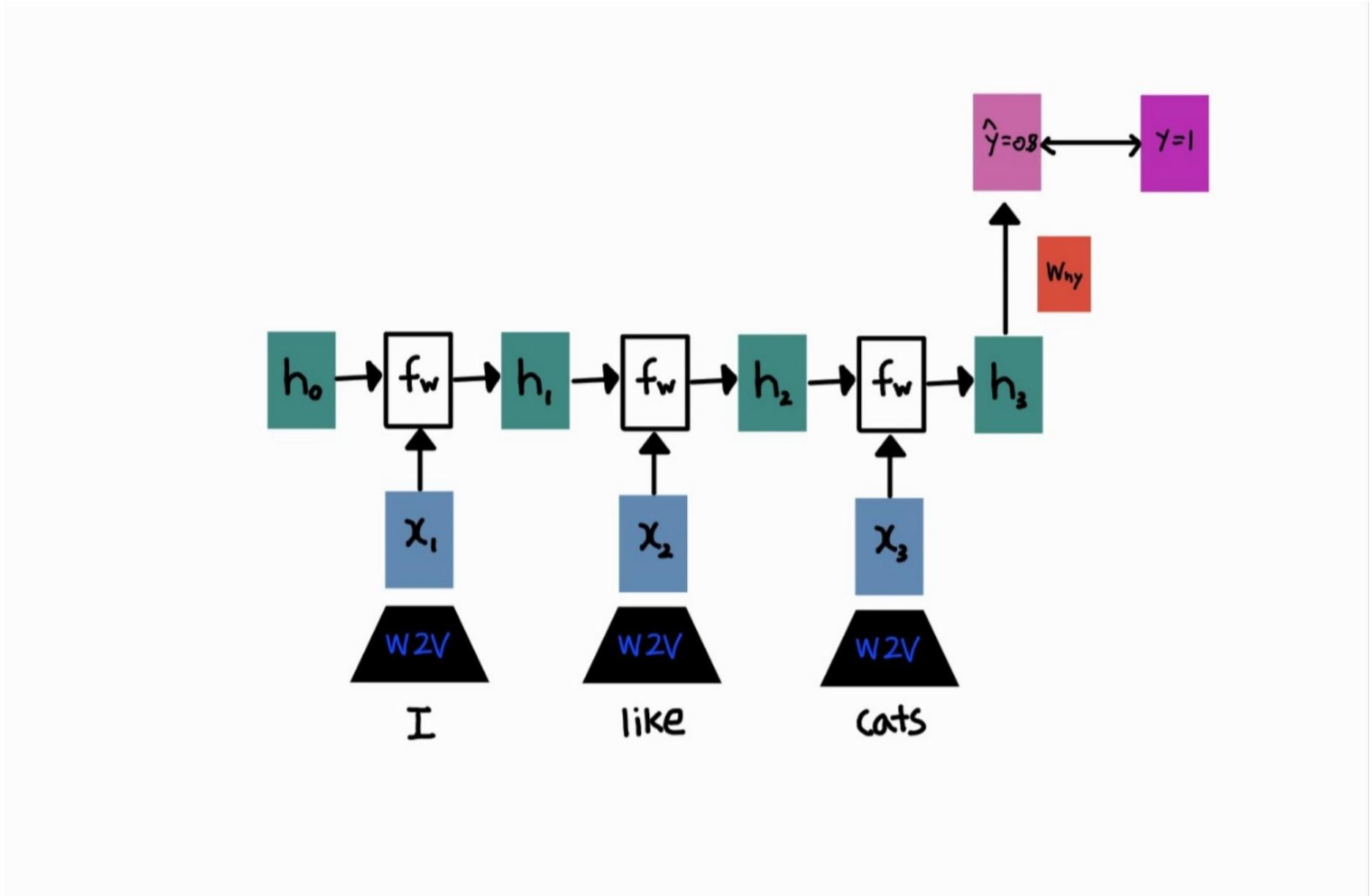
- internal state



fully connected layer

$$h_t = f_W(h_{t-1}, x_t)$$

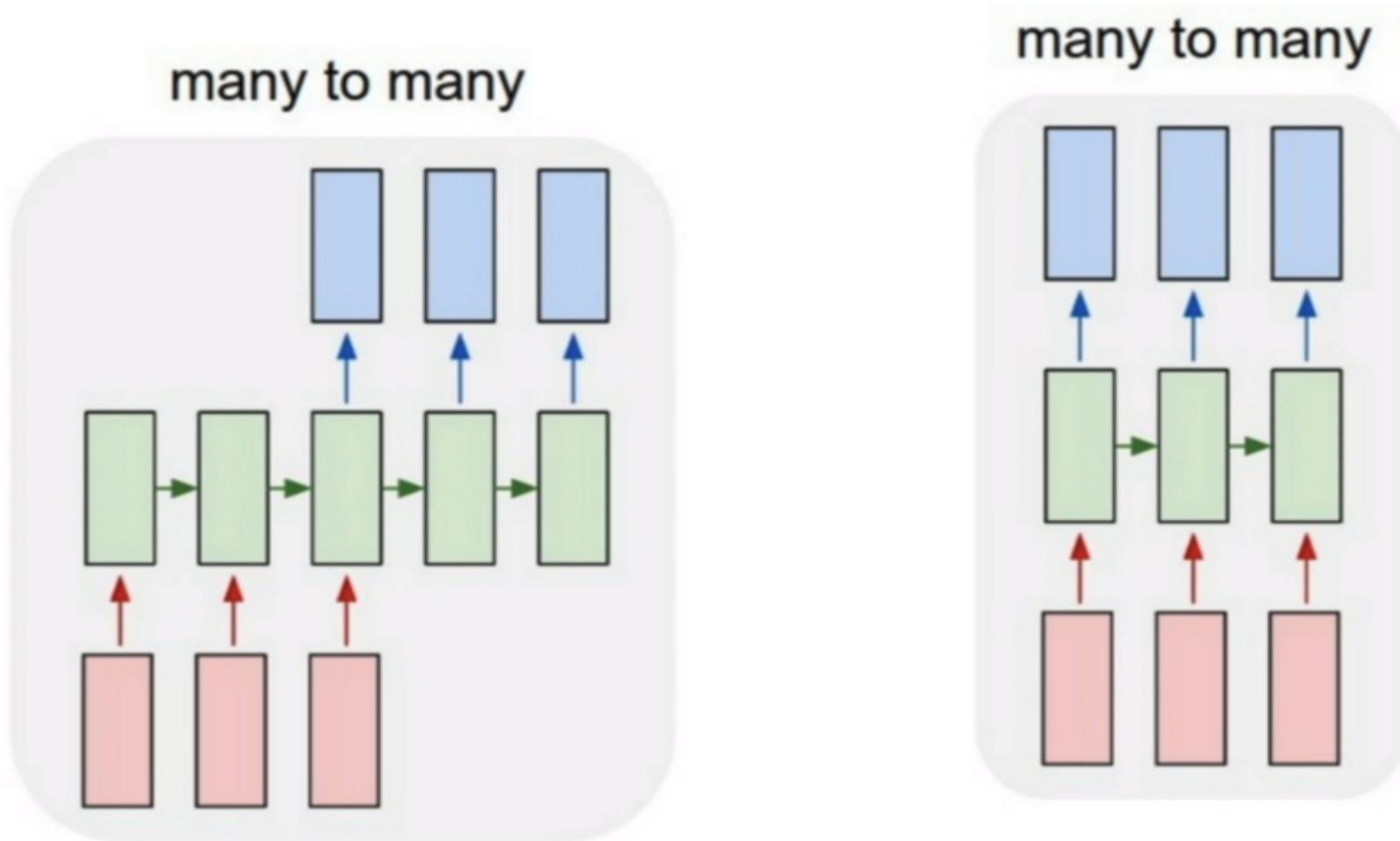
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Neural Machine Translation

many to many problem

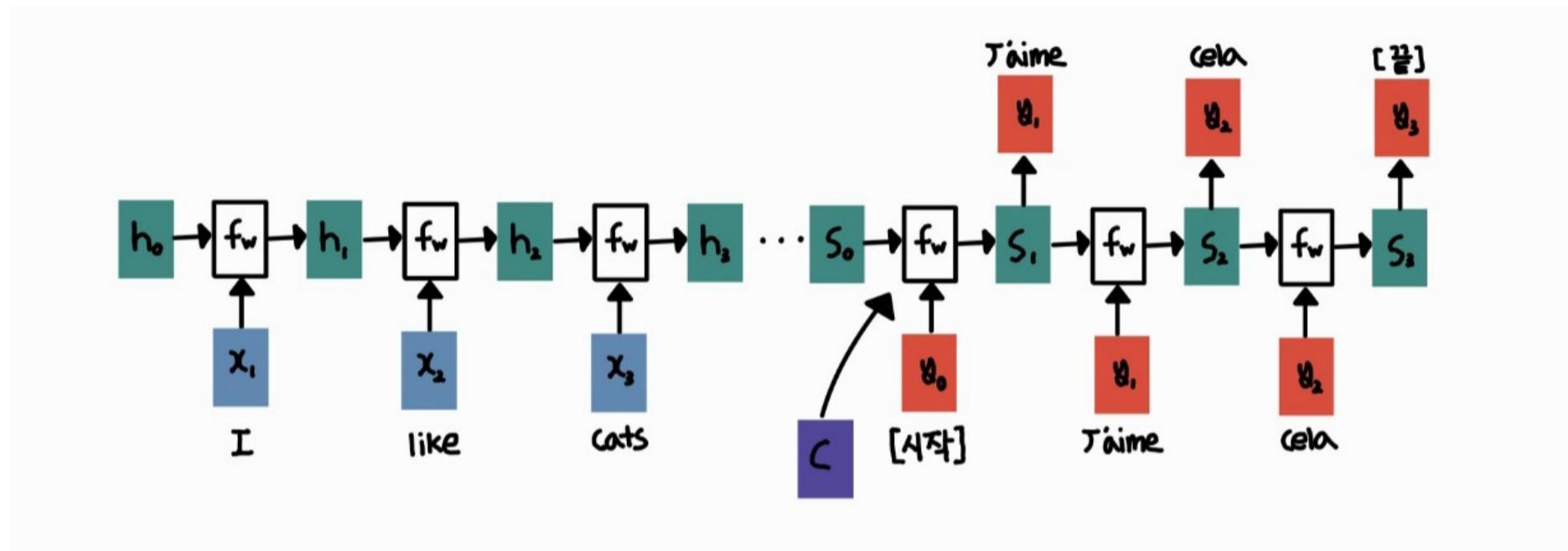
ex) 프랑스-영어 번역



encoder와 decoder 존재

encoder: 영어 문장에 대한 정보

decoder: 프랑스 문장에 대한 정보



context vector: c

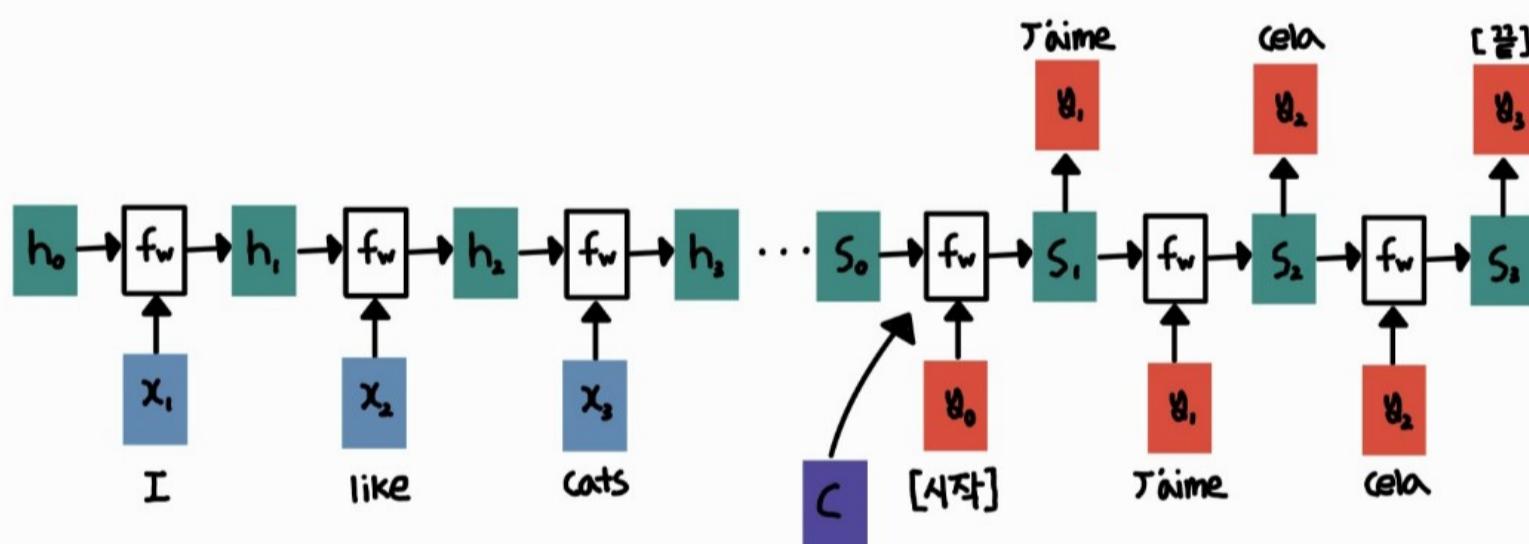
보통, $c = h_t$

▼ Problem

문장이 길어졌을 때, 문제 발생 c 에 대한 부담이 커짐.

기존의 RNN Encoder Decoder 보다 뛰어난 번역 모델을 만들 수 있을까?

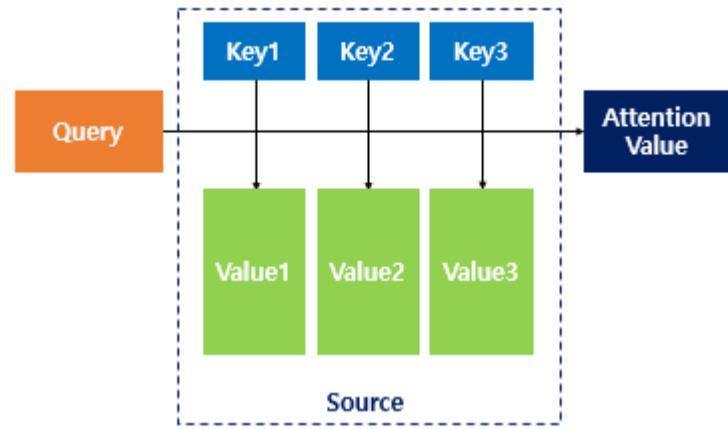
▼ Solution in paper



마지막 h_t 만 보는 것이 아니라, 전체 h_t 들을 보자. (h_1, h_2, \dots, h_t)

그리고 이 h_t 들 중에 어디에 집중할 것인가? \Rightarrow attention

Attention function

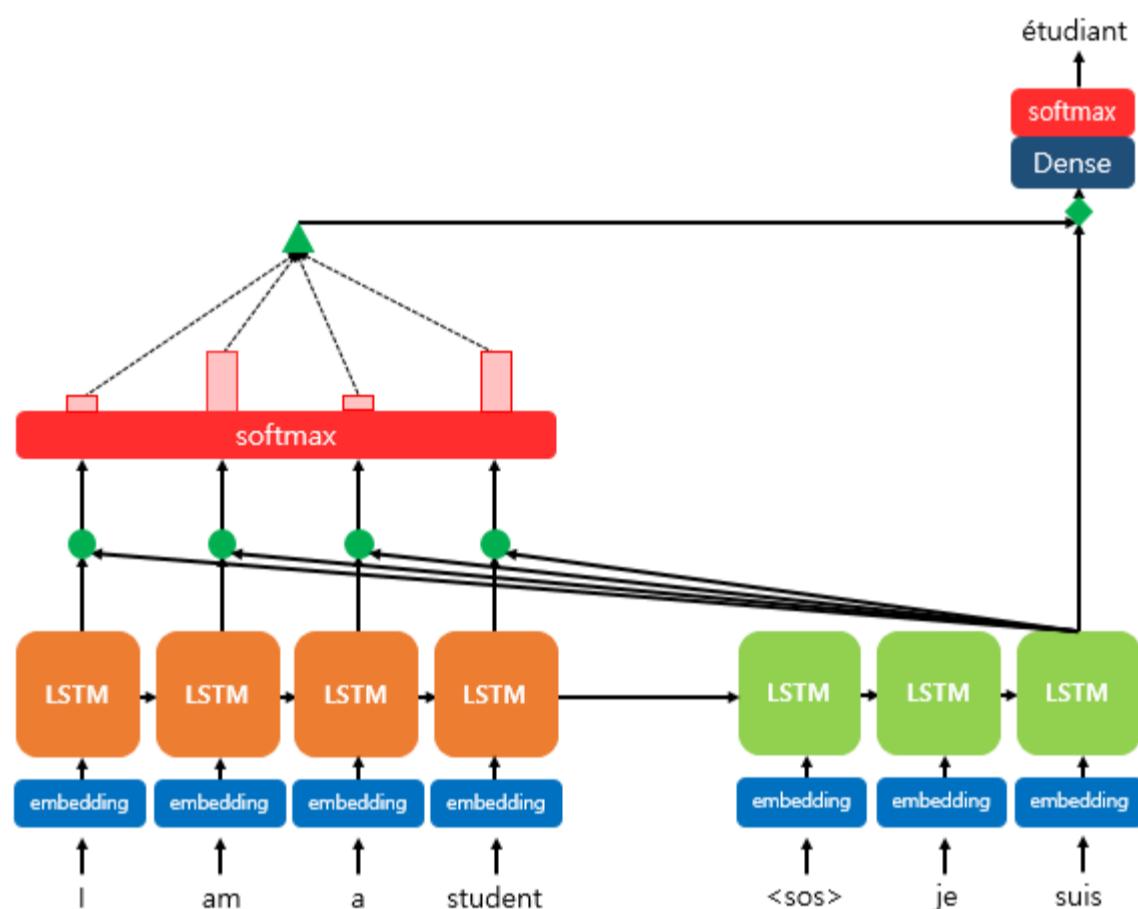


Q = Query : t 시점의 디코더 셀에서의 은닉 상태

K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

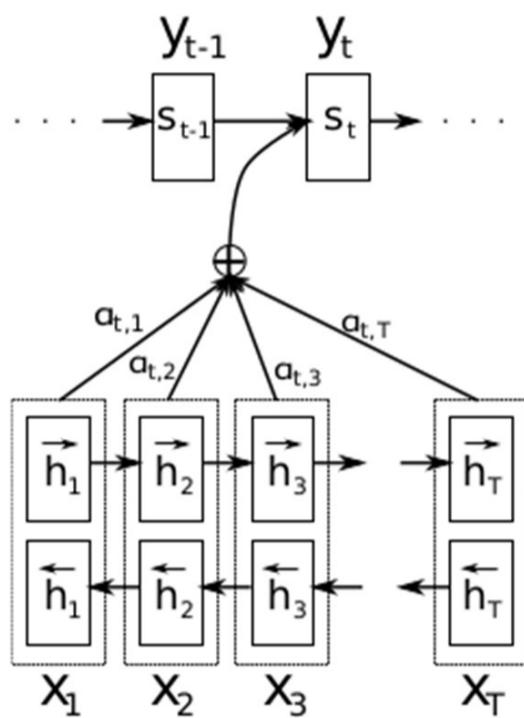
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

Attention 전체 과정



Attention 각각의 과정 설명

Attention mechanism



- 출력 단어를 예측하기 위해서 인코더의 모든 입력 단어들의 정보를 다시 한번 참고하고자 합니다
- Encoder hidden state에서는 양방향 LSTM이 사용 한번은 정방향, 한번은 반대방향으로

Attention mechanism의 과정

Dot-Product Attention

1. Attention score $e_{ij} = a(s_{i-1}, h_j)$ [스칼라]	3. Attention output $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$ [벡터]
2. Attention distribution $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$ [스칼라]	4. Decoder hidden state $s_i = f(s_{i-1}, y_{i-1}, c_i)$ [벡터]

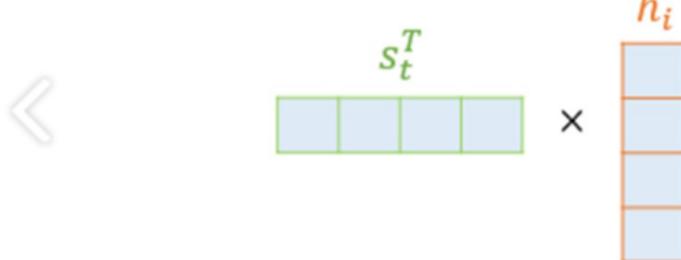
Attention mechanism : Attention score

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

[스칼라] 정렬 모델 ↑ 인코더 은닉층
디코더 은닉층
“어느 입력 시간 스텝에 집중 할지를 점수화”

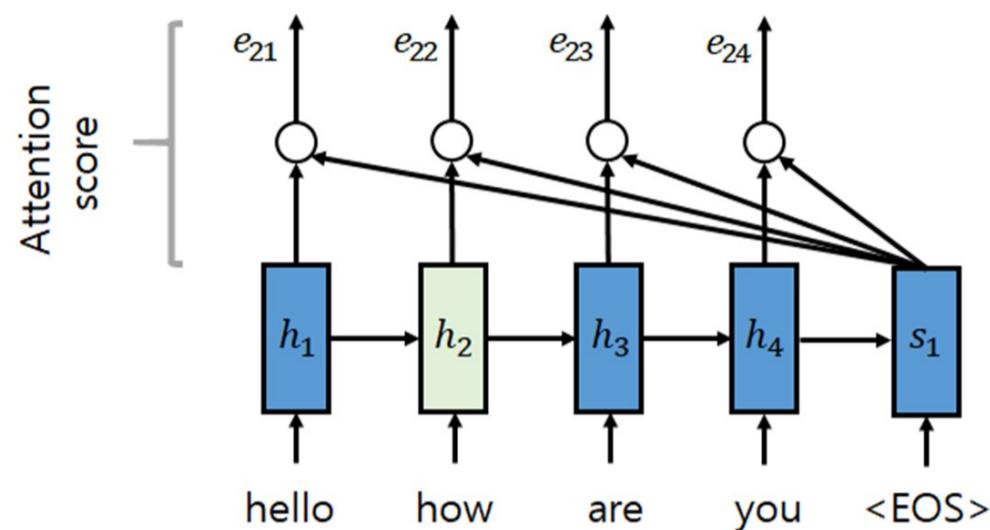
- 디코딩 할 때, 인코더의 어느 입력 시점에 집중 할 것인지를 점수화한 것
- 한 디코딩 시점으로부터 모든 입력 시점에 대해서 계산이 되고, 이 계산은 모든 디코딩 스텝에서 반복

Attention mechanism : Attention score



- 디코더의 hidden state 값(s_t)을 transpose하고 각 hidden state(h_i)와 내적(dot product)하여 스칼라 값으로
- Dot product attention인 이유

Attention mechanism : Attention score



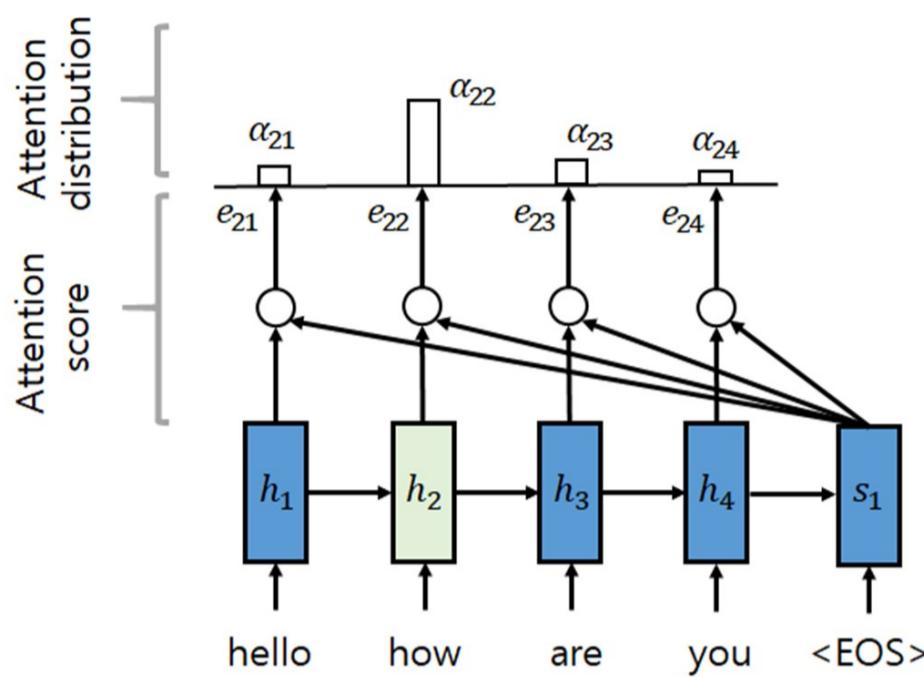
Attention mechanism : Attention distribution

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

[스칼라]
"입력 시간 스텝에 대한 가중치" $= \text{softmax}(e_{ij})$

- Attention score를 가지고 softmax 함수를
 통과시켜 확률화(분수화)하여 값을 만듦
- 계산된 각 0~1사이의 값들이 바로 입력 시점에
 대한 가중치, 즉 "시간의 가중치"가 되는 것

Attention mechanism : Attention distribution



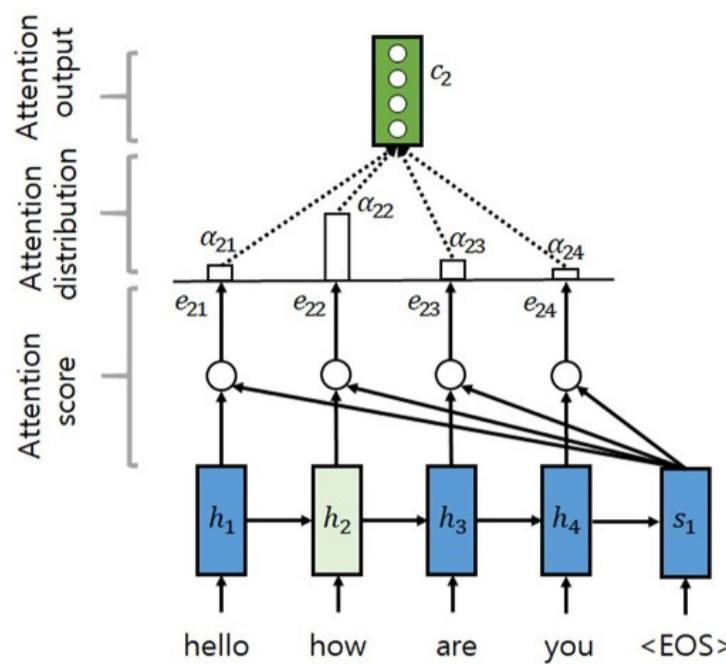
Attention mechanism : Attention output

$$[текущий] c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

"시간의 가중치"
"입력 시퀀스의 가중 합"

- 인코더의 hidden state와 attention 가중치 값을 곱하고, 최종적으로 모두 더한다.
- 가중 합을 계산하여 최종적으로 하나의 벡터로
- 이 벡터는 매 디코딩 시점마다 다르며, 따라서 기존에 fixed-length vector의 문제점을 해결할 수 있도록 구성

Attention mechanism : Attention output



Attention mechanism : Decoder hidden state

"가변 context 벡터를 고려한 hidden state"

$$s_i = f(s_{i-1}, y_{i-1}, \underline{c}_i)$$

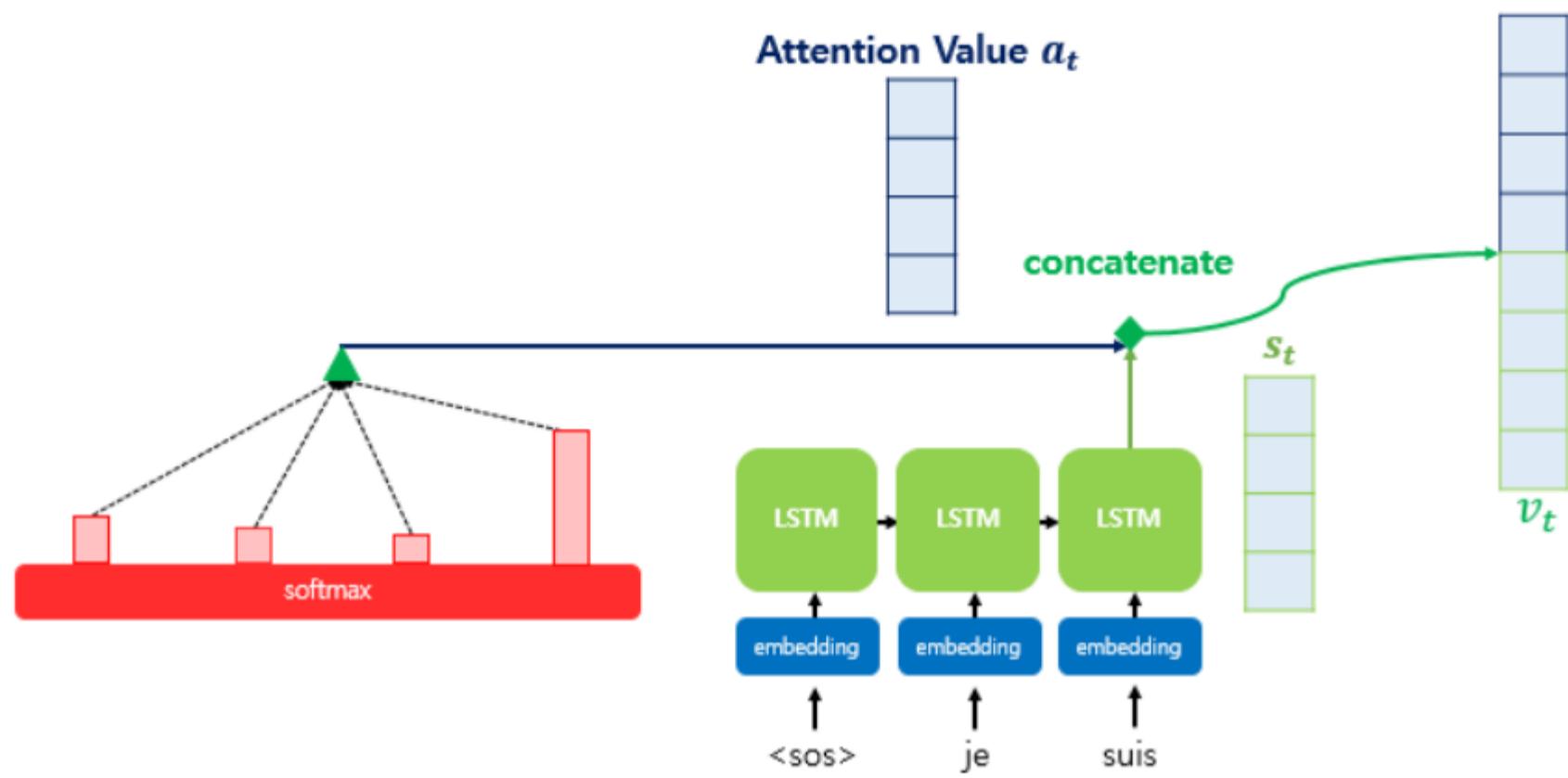
[벡터]

*비교 : Decoder without attention

$$s_i = f(s_{i-1}, y_{i-1}, \underline{c})$$

- 계산해낸 Attention output 벡터와 이전 디코더의 hidden state, 출력을 이용하여 최종적으로 다음 decoder hidden state를 출력
- fixed-length 벡터가 매 시간 인덱스마다 다르게 반영되는 것을 확인

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

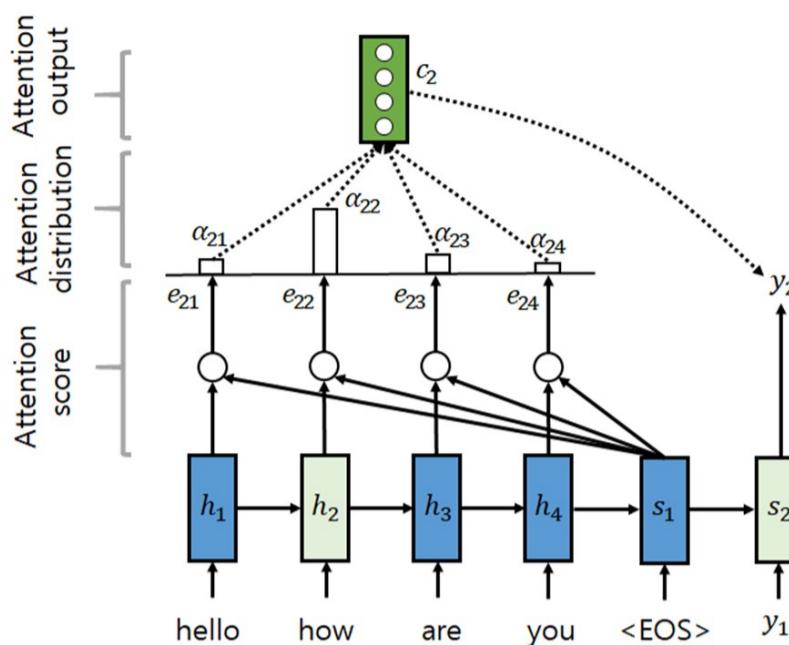


5) 출력층 연산의 입력이 되는 \tilde{s}_t 를 계산합니다.

$$\tanh \left(W_c \times v_t \right) = \tilde{s}_t$$

The diagram shows the calculation of the output layer's input \tilde{s}_t . It consists of a matrix multiplication (\times) between a weight matrix W_c (represented as a grid of blue squares) and the concatenated vector v_t (green vertical bar), followed by a tanh activation function (indicated by the tanh symbol).

Attention mechanism : Decoder hidden state



6) \tilde{s}_t 를 출력층의 입력으로 사용합니다.

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

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

▼ Implementation

<https://americanoisice.tistory.com/58>

Pseudo code

```
#Prepare dataset
src_text, target_text
src_text.tokenize()
target_text.tokenize()
target_text.push_front("<sos>")
target_text.push_back("<eos>")

src_text.vector()
target_text.vector()

# Attention
## attention score

src_hidden_state = LSTM(input=src_text, hidden_state=true)
for i, tar in enumerate(target_text):
    tar_hidden_state = LSTM(input=tar)
    attention_score = src_hidden_state.dot_product(tar_hidden_state)
## attention distribution
    attention_dist = softmax(attention_score)

## attention output(attention value)
    attention_output = (attention_dist * src_hidden_state).sum(axis=0)

# Output
```

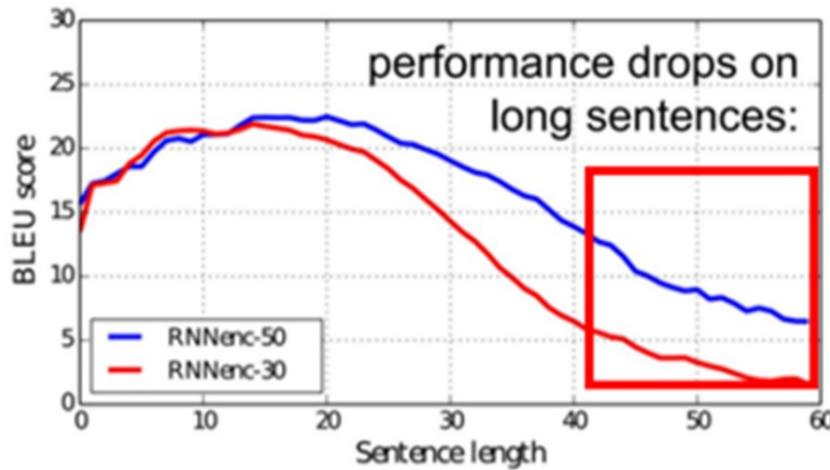
```

## concat attention value + hidden state
output = attention_output.concat(tar_hidden_state)
output = output * output_weight
## softmax
# output = softmax(output) # for inference
## learning
loss = cross_entropy(output, target_text[i + 1])
loss.backward()

```

▼ Experiment & Result

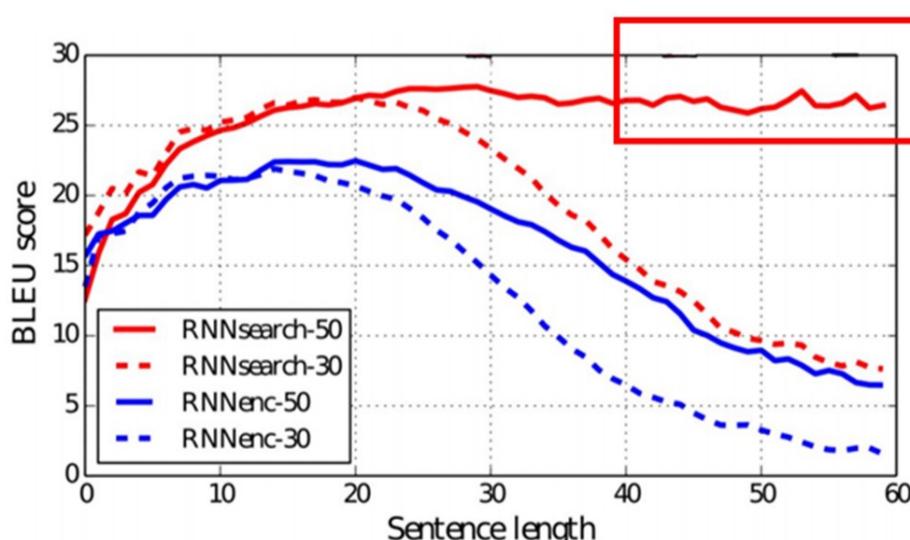
RESULT



<기존 seq2seq가 적용된 RNN model>

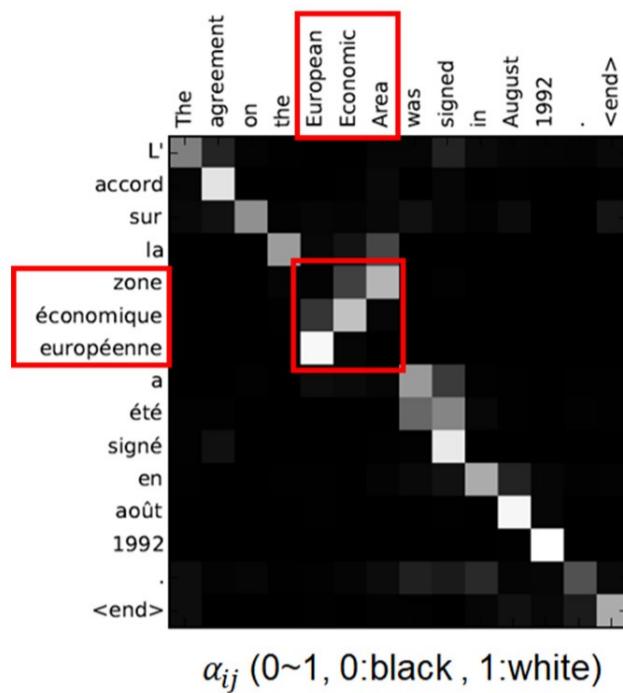
- 훈련 데이터셋을 구성하는 문장이 길면 길수록 문장 길이에 영향을 덜 받음
- 기존의 seq2seq는 test sentence의 길이가 길어질수록 심각한 성능의 저하가 발생

RESULT



- RNNenc-30/RNNenc-50
: 30개/50개 이하의 단어로 구성된 문장들로 학습시킨 기존의 seq2seq 모델
- RNNsearch-30/RNNsearch-50
: 30개/50개 이하의 단어로 구성된 문장들로 학습시킨 proposed model
- 특히 RNNsearch-50은 문장의 길이가 길어져도 성능의 저하가 거의 없는 것을 관찰할 수 있다. (문장 길이에 robust)

RESULT



<영어 → 프랑스어의 번역 결과의 Attention score를 matrix 형태로 나타낸 것>

- 검은 색은 Attention score가 0임을 의미하고, 흰색은 1을 의미한다.
- Matrix를 보면 대개 단어와 단어들이 1:1로 매칭되는 경향(monotonic alignment)을 확인할 수 있다.
- 단순히 순서대로 1:1 매칭된 것이 아님을 알 수 있다.
- Alignment model을 통해 입력 시점 중 중요한 부분에 집중하고 있다는 것을 정량적으로 확인 할 수 있다.

RESULT

English to French

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

- RNNsearch-30 성능 > RNNencdec-50 성능
- Frequent한 단어만 존재하는 No UNK에서,
영-프의 parallel corpus만 사용한 proposed model(RNNsearch-50)의 성능 >
parallel corpus + monolingual corpus까지 사용한 기존의 phrase-based model(Moses)