RNN (Recurrent NN)

LG 인화원 교육
윤세영
KAIST 김재철AI대학원

KAIST

Important References

Stanford CS231n course, **CS 224n**
http://cs231n.stanford.edu/index.html
Lecture slides, Youtube video,

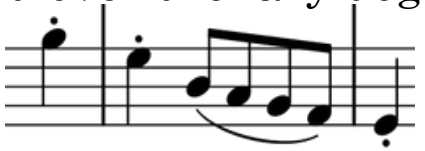
Coursera Deep Learning course by Andrew Ng
https://www.deeplearning.ai
Not free if you want to get certifications

PyTorch Deep Learning Mini Course
https://github.com/Atcold/PyTorch-Deep-Learning-Minicourse

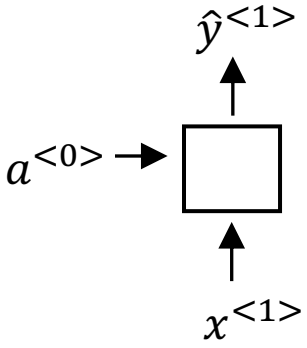
Many source codes in Github

목차

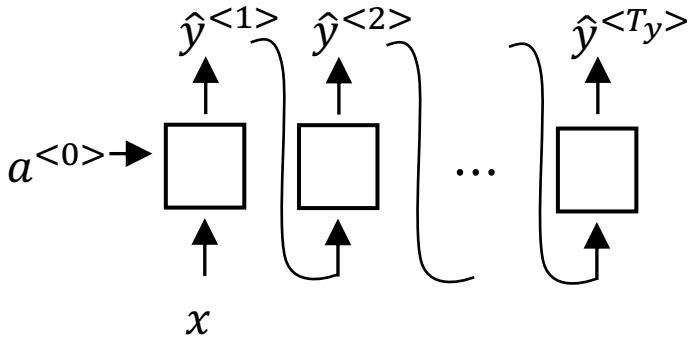
Sequential Data

Speech recognition		→	"The quick brown fox jum ped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

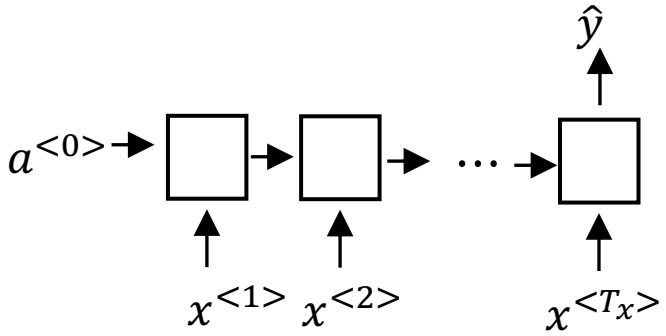
Recurrent Neural Network



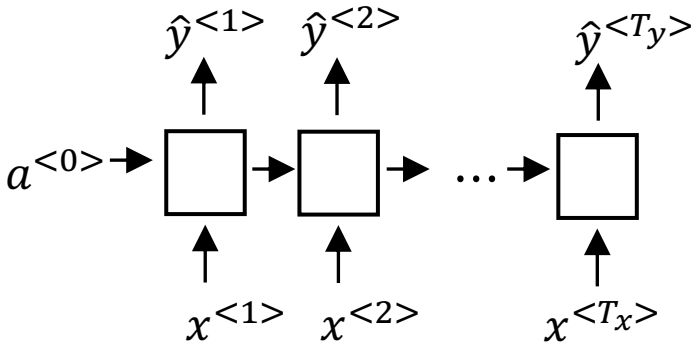
One to one



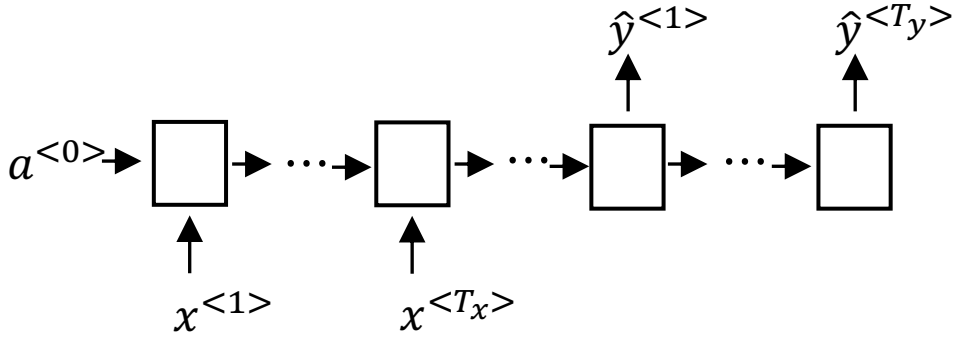
One to many



Many to one



Many to many



Many to many

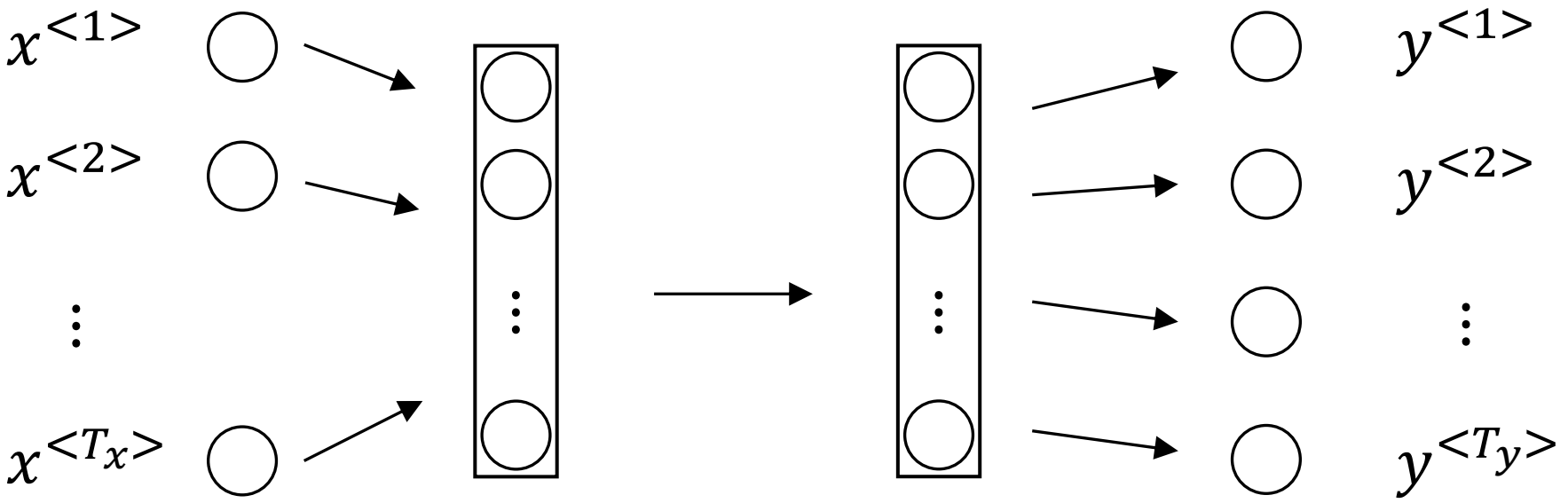
번역?

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ … $x^{<9>}$

And = 367
Invented = 4700
A = 1
New = 5976
Spell = 8376
Harry = 4075
Potter = 6830
Hermione = 4200
Gran… = 4000

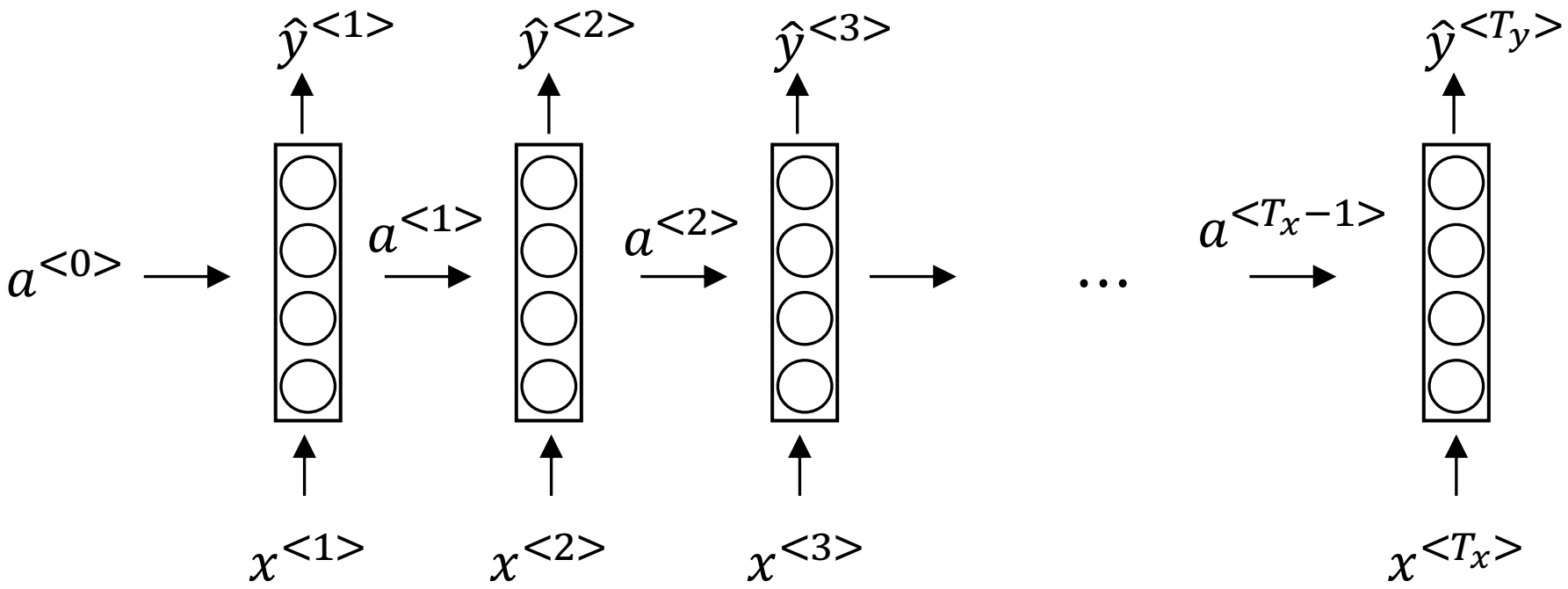
다른 종류의 NN을 사용 하는 경우 발생하는 문제?



문제점

- Input의 길이가 매번 달라짐
- Output의 길이가 매번 달라짐

Recurrent Neural Network

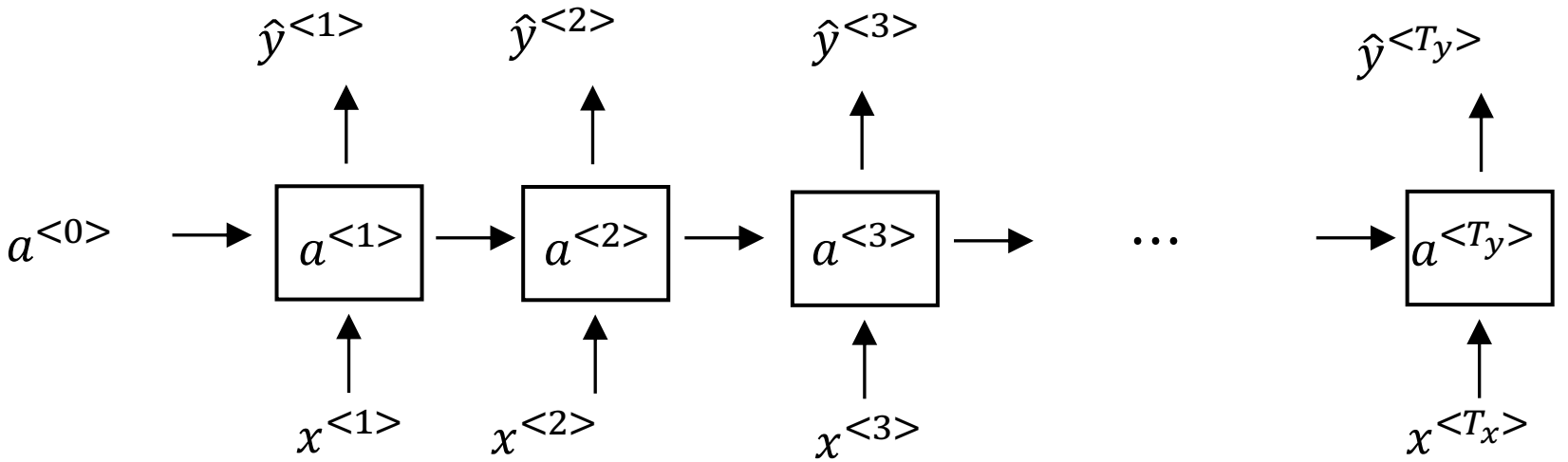


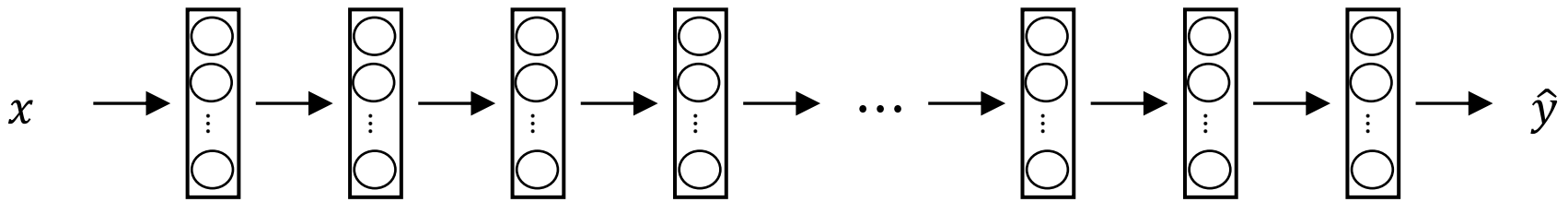
해결 방안

- 〈end〉 에 해당하는 신호를 지정하여 다양한 문장 길이 처리 가능
- Parameter 재사용으로 overfitting 해결

Vanishing Gradient

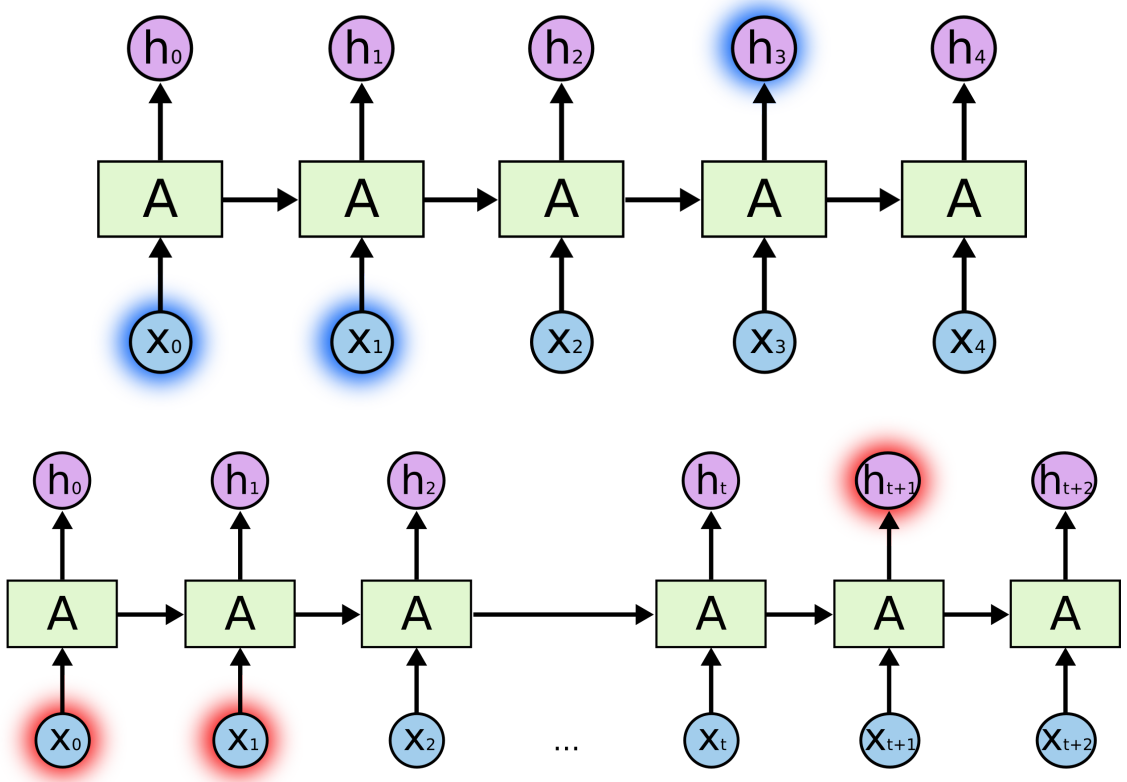
기존 deep learning 이 가지고 있는 vanishing gradient 문제가 동일하게 존재





Long-term Dependency

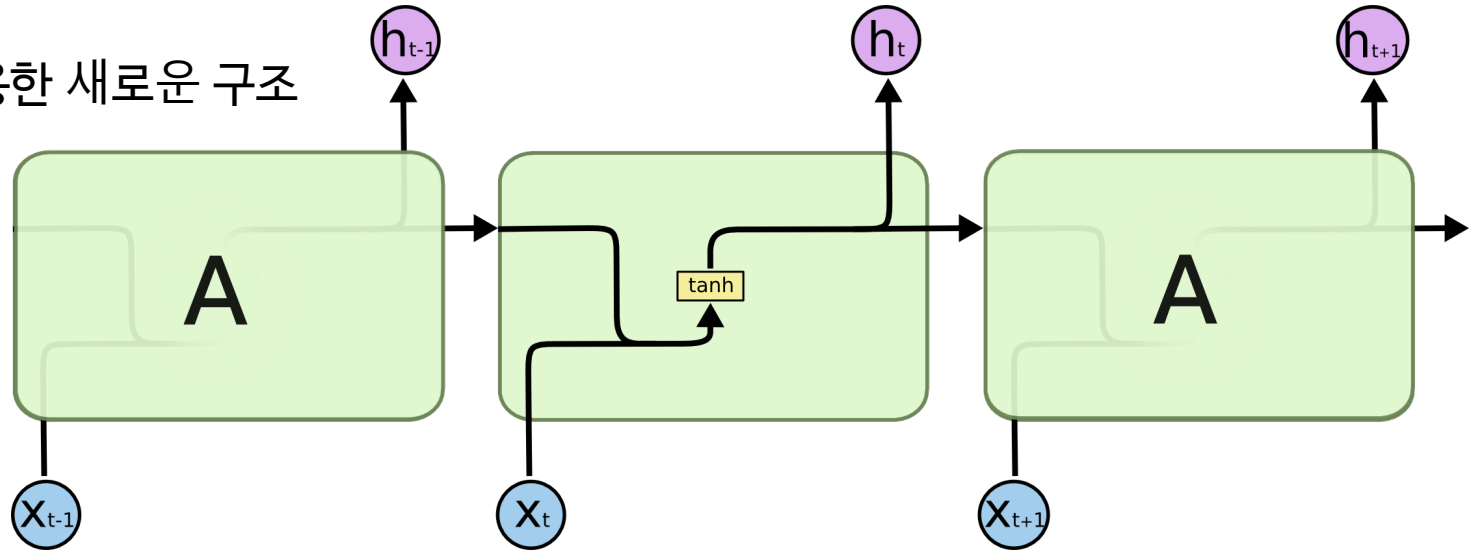
이것을 흔히 long-term dependency 문제라고 부름 (앞쪽 단어와 뒤쪽 단어의 상관관계를 배우기 어려움)



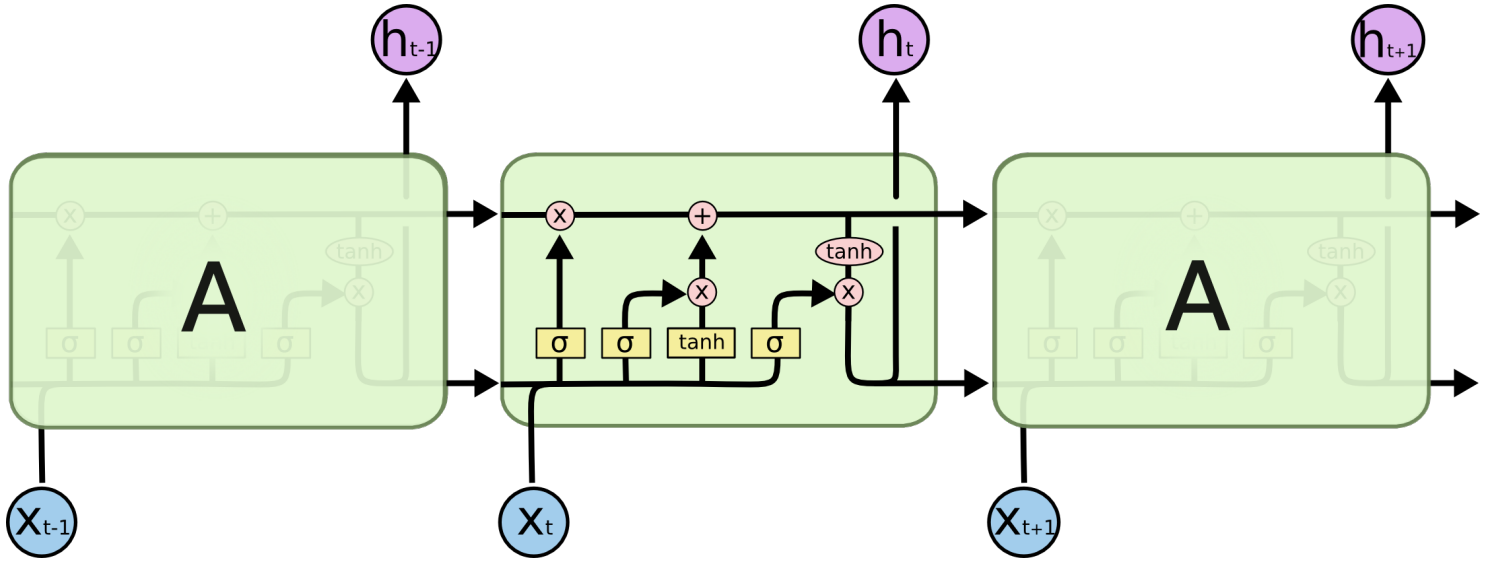
목차

LSTM (Long Short-Term Memory)

Switch와 Short-cut을 활용한 새로운 구조

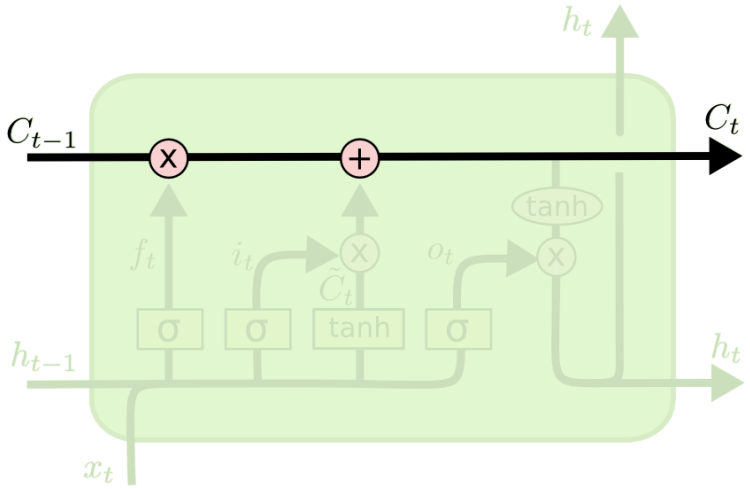


RNN



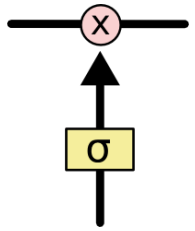
LSTM

LSTM (Long Short-Term Memory)

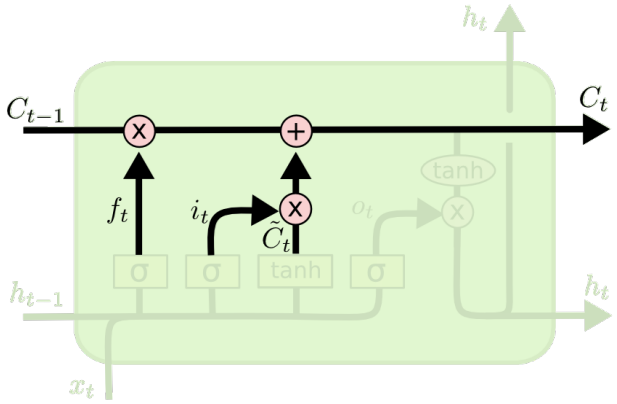


Switch와 Short-cut을 활용한 새로운 구조

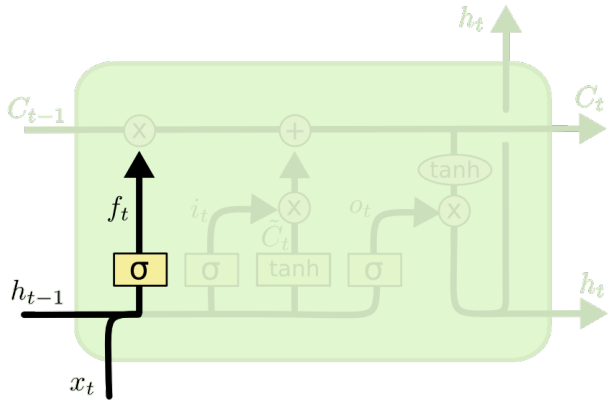
- Sigmoid 함수는 0과 1사이의 값을 결정하며 C값이 얼마나 흘러 들어오게 만들지 결정
- 아래쪽의 복잡한 과정을 건너 뛰는 신호이기 때문에 먼 과거의 정보가 흘러 들어옴



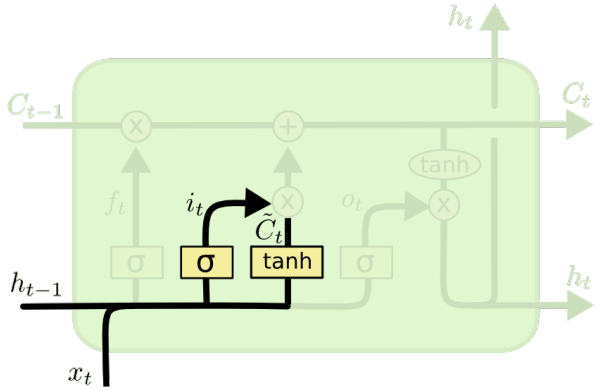
LSTM (Long Short-Term Memory)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



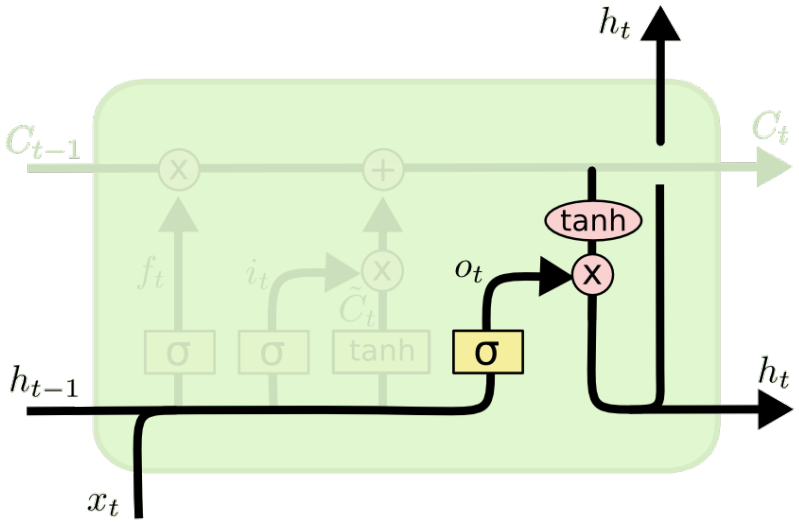
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Switch 연산

- 지난 시간의 hidden feature와 지금 시간의 입력 feature를 결합하여 switch 결정
- C로 얼마만큼의 정보를 새로 올려 놓을지 역시 다른 switch로 결정

LSTM (Long Short-Term Memory)



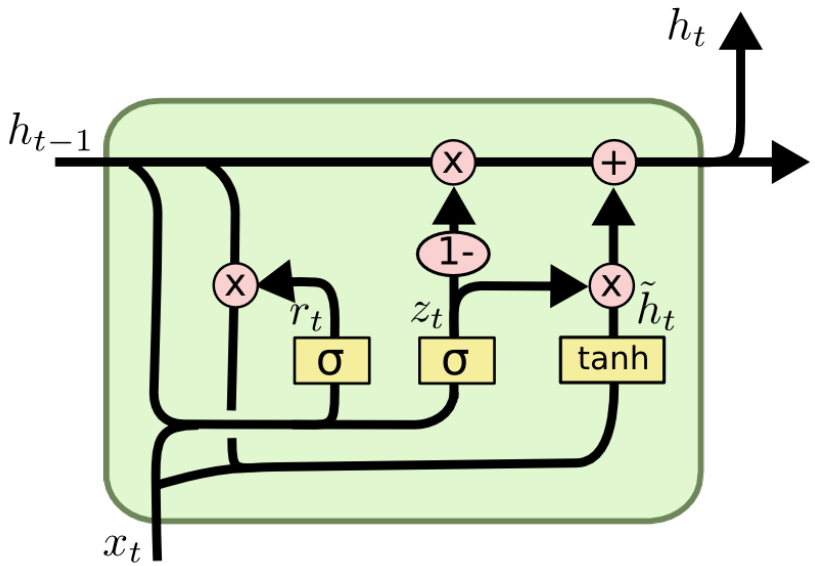
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Hidden feature 연산

- 지난 시간의 hidden feature와 지금 시간의 입력 feature를 바탕으로 새로운 switch 정의
- C의 정보를 바탕으로 hidden feature를 결정

GRU (Gated Recurrent Unit)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM과 비슷한 연산

- Switch의 위치를 살짝 조정
- 1-sigmoid를 통하여 신호를 양 갈래 중 하나로 흘러보냄

목차

여전히 존재하는 RNN의 문제점

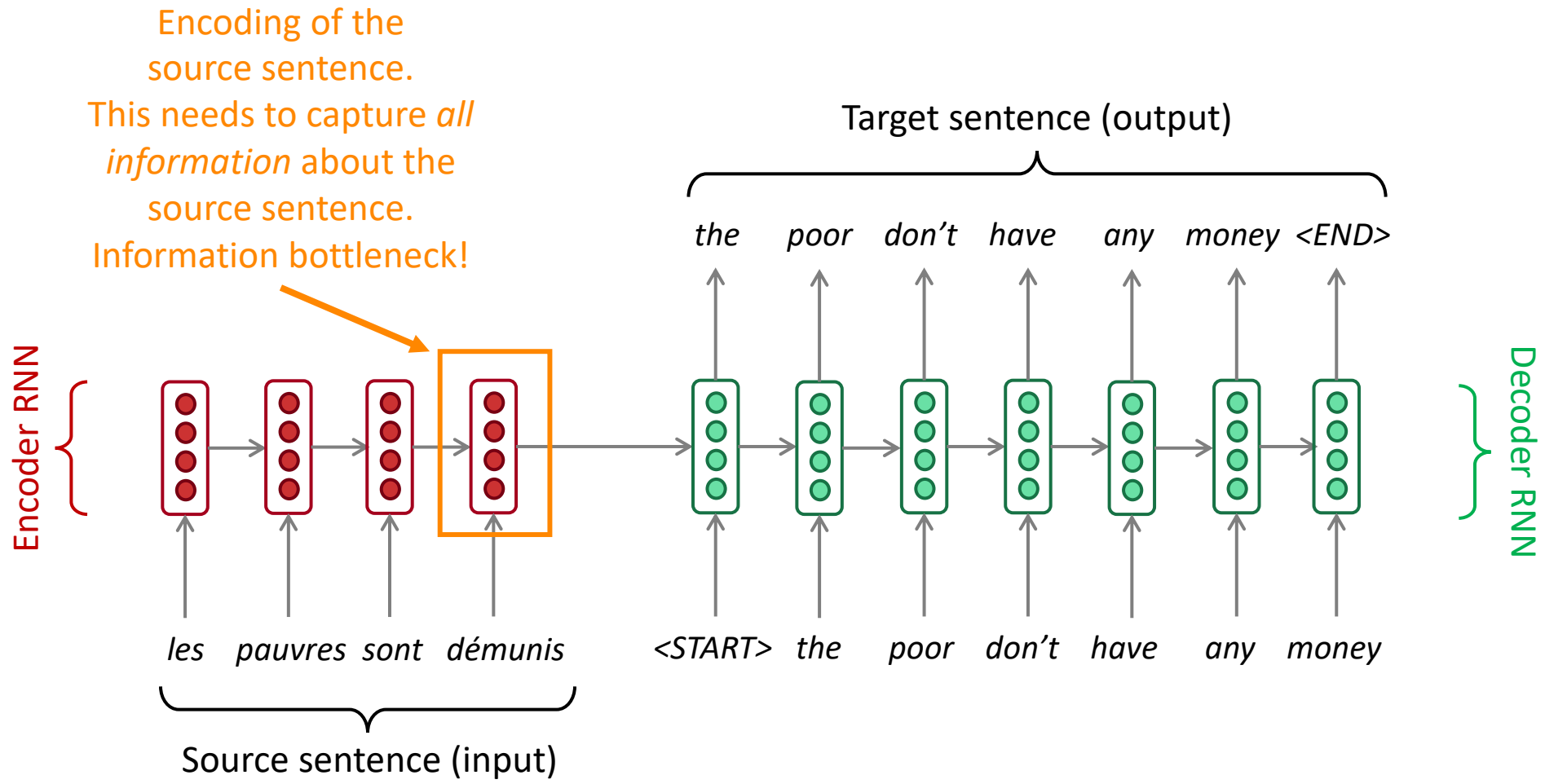
각 단어들을 어떻게 인식해야 할 것인가?

- Out-of-vocabulary words
- Domain mismatch between train and test data
- Maintaining context over longer text
- Low-resource language pairs
- Using common sense is still hard
- NMT picks up biases in training data
- ……..

Attention을 이용하여 위의 문제점을 해결 가능

번역을 위해서는 모든 정보가 필요하다!

Sequence-to-sequence: the bottleneck problem



Attention

집중! 무엇을 집중하는 의미인가?

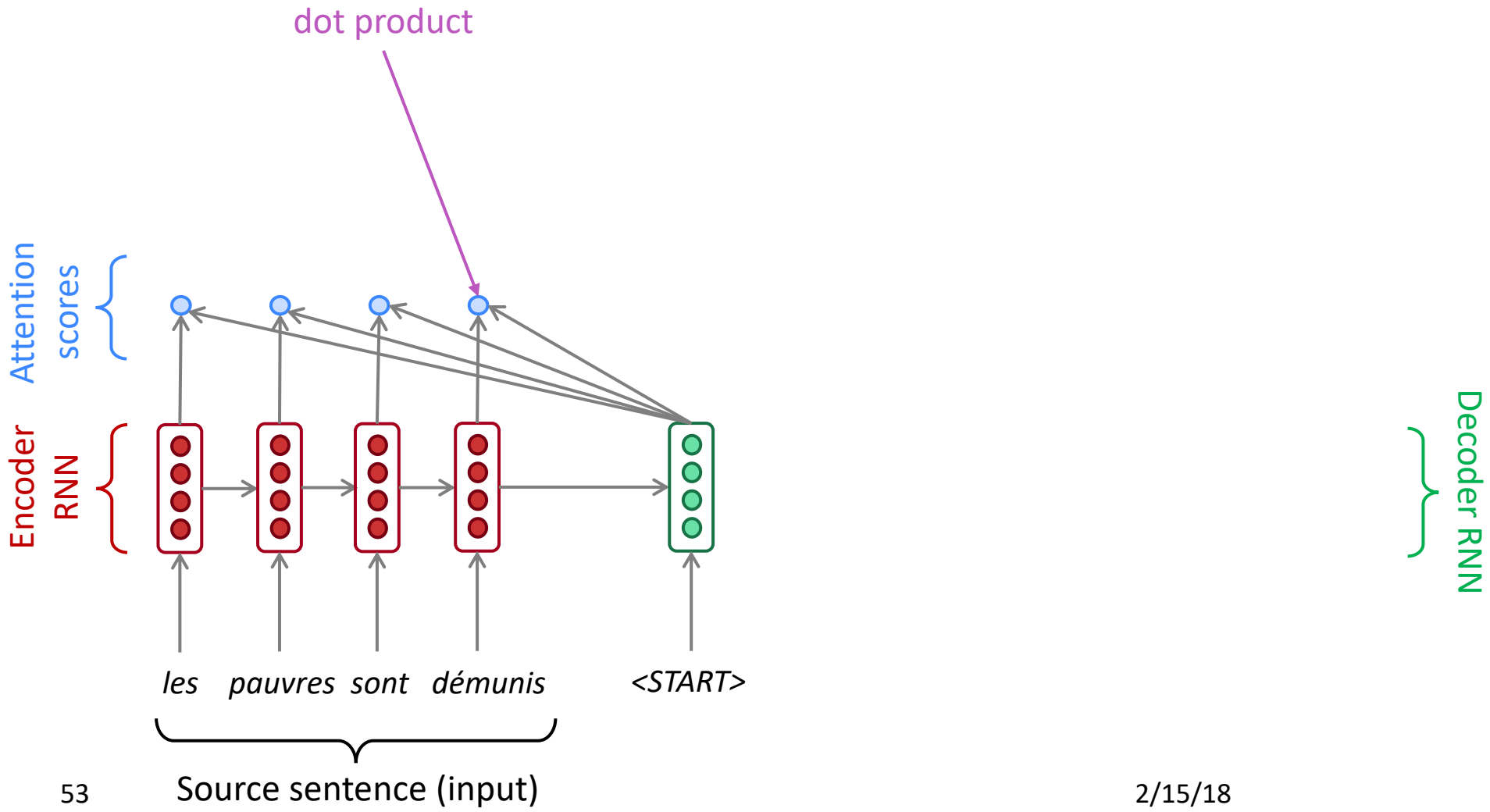
번역 문장을 만드는 과정에서 다음 단어를 만들기 위해서 원본 문장의 일부 부분이 매우 중요하게 작용

어떤 단어를 집중 할 것인가?

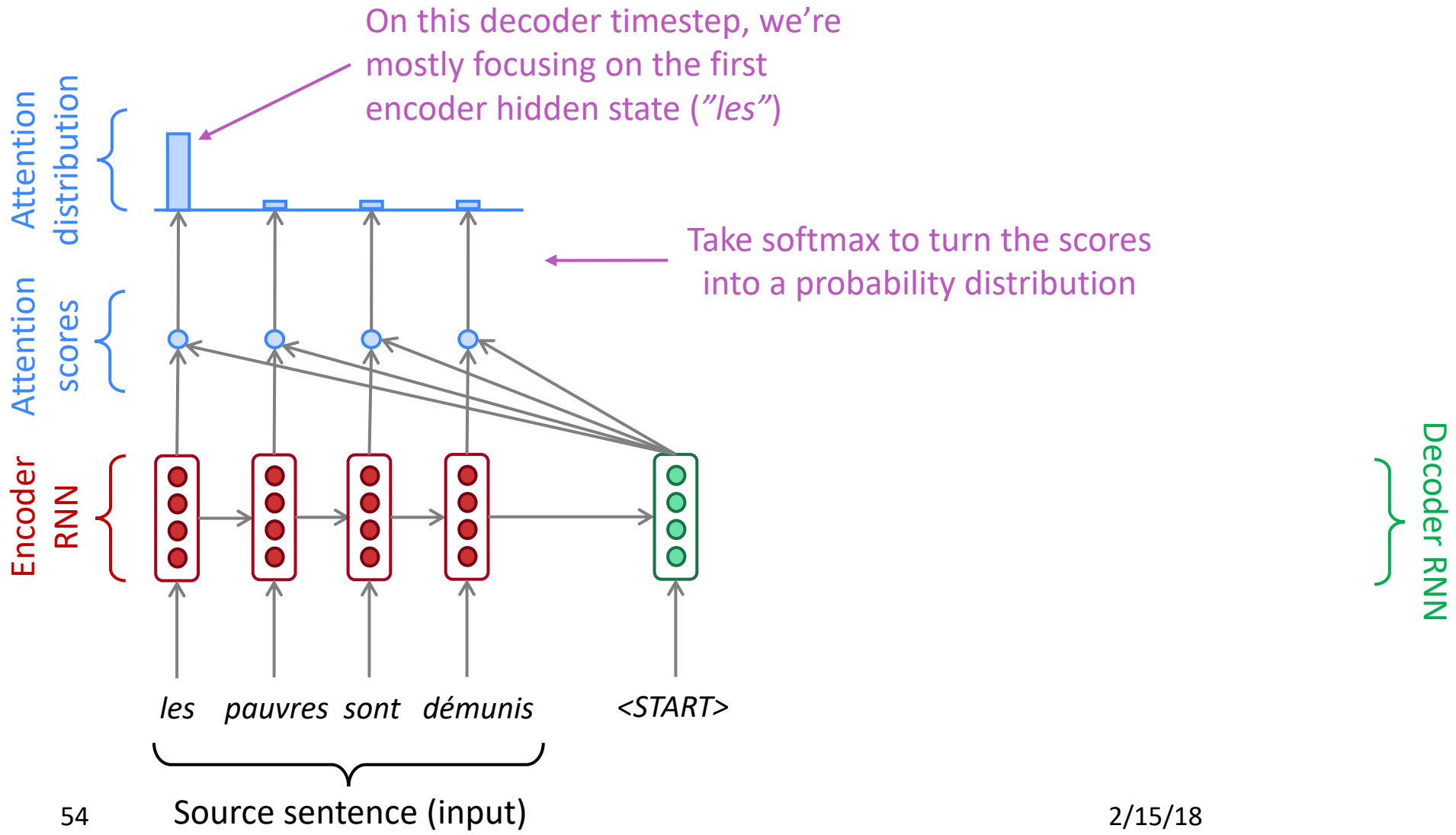
집중하는 단어를 어떻게 선정 할 수 있을까?

집중을 정의하는 함수를 만들고 해당 함수를 활용하여 attention network 형성

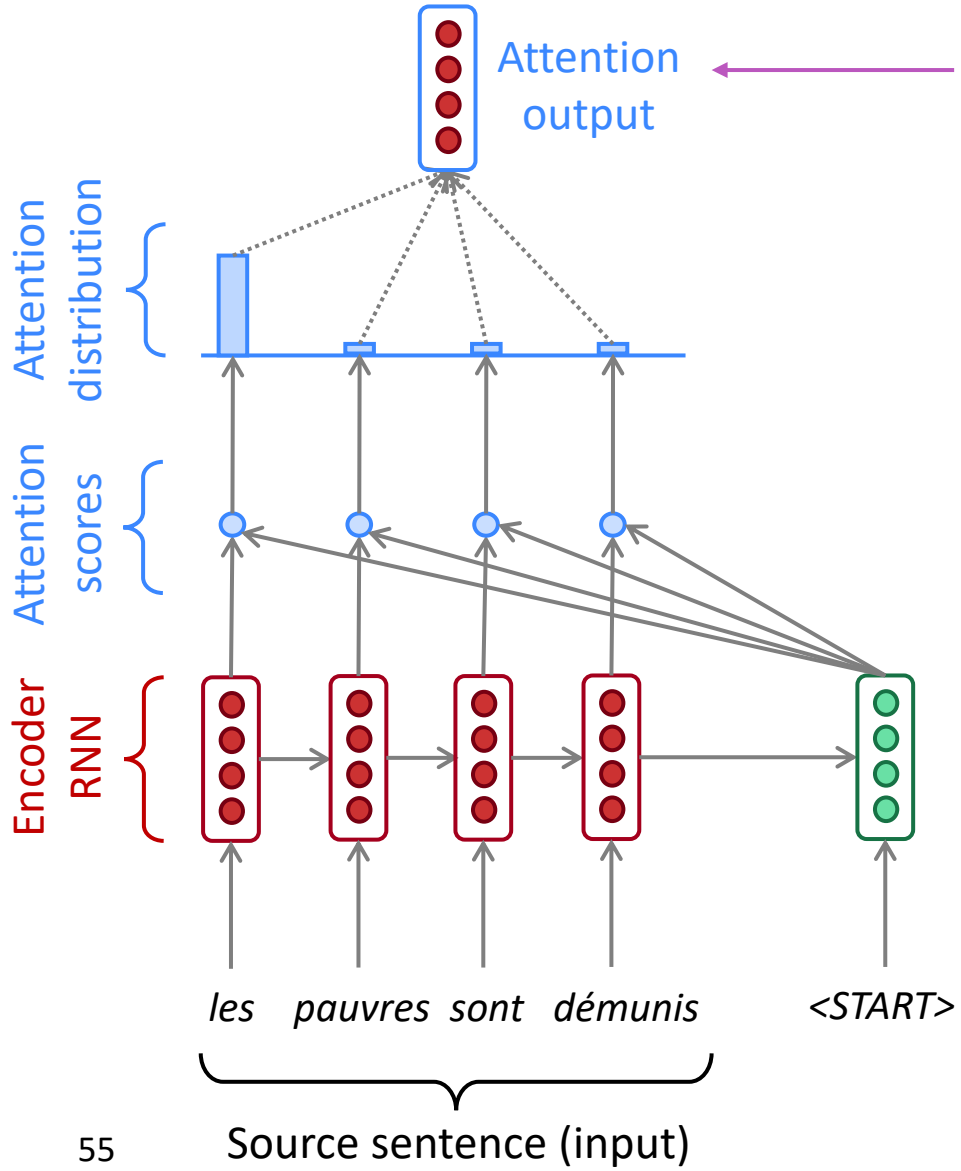
Sequence-to-sequence with attention



Sequence-to-sequence with attention



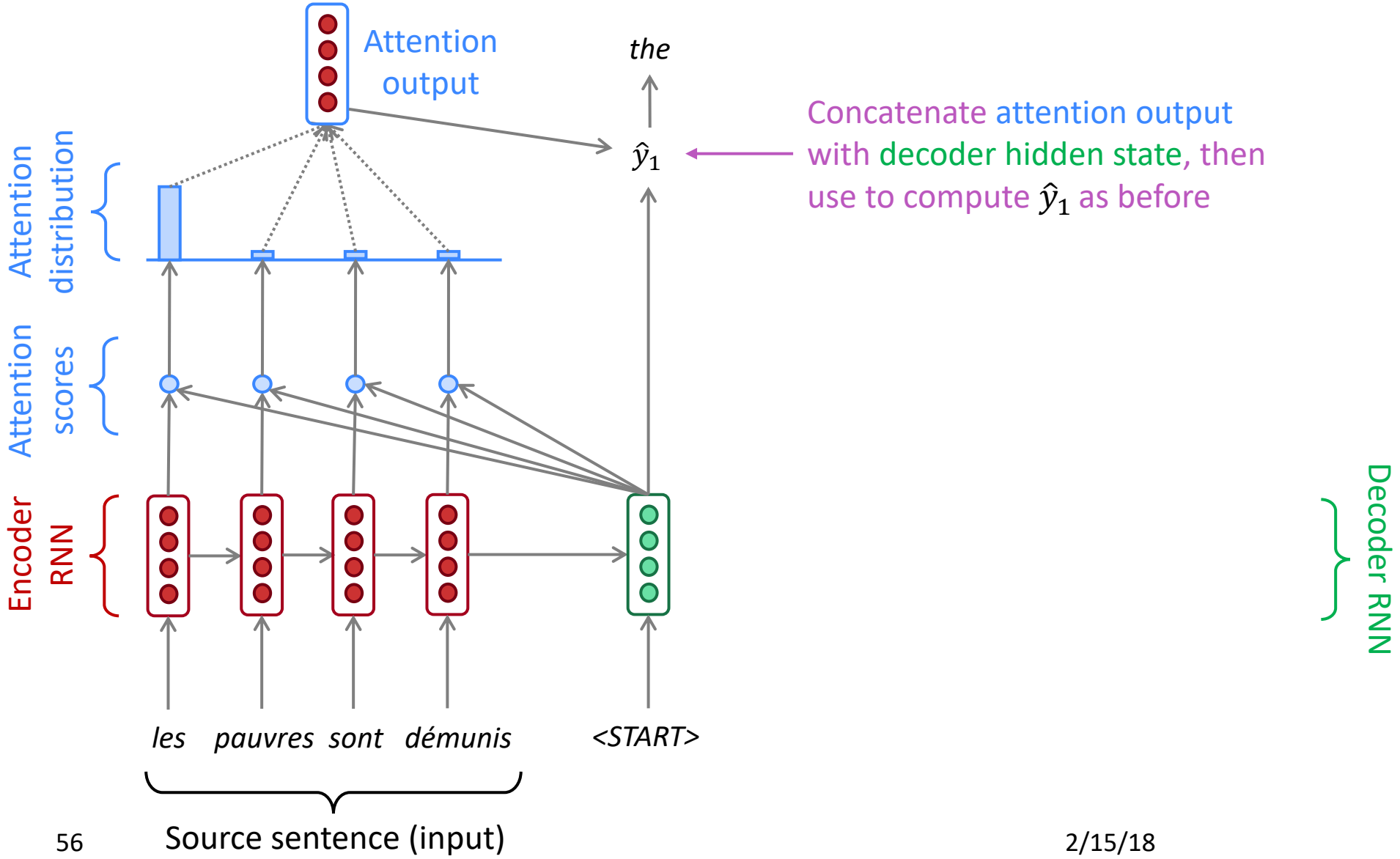
Sequence-to-sequence with attention



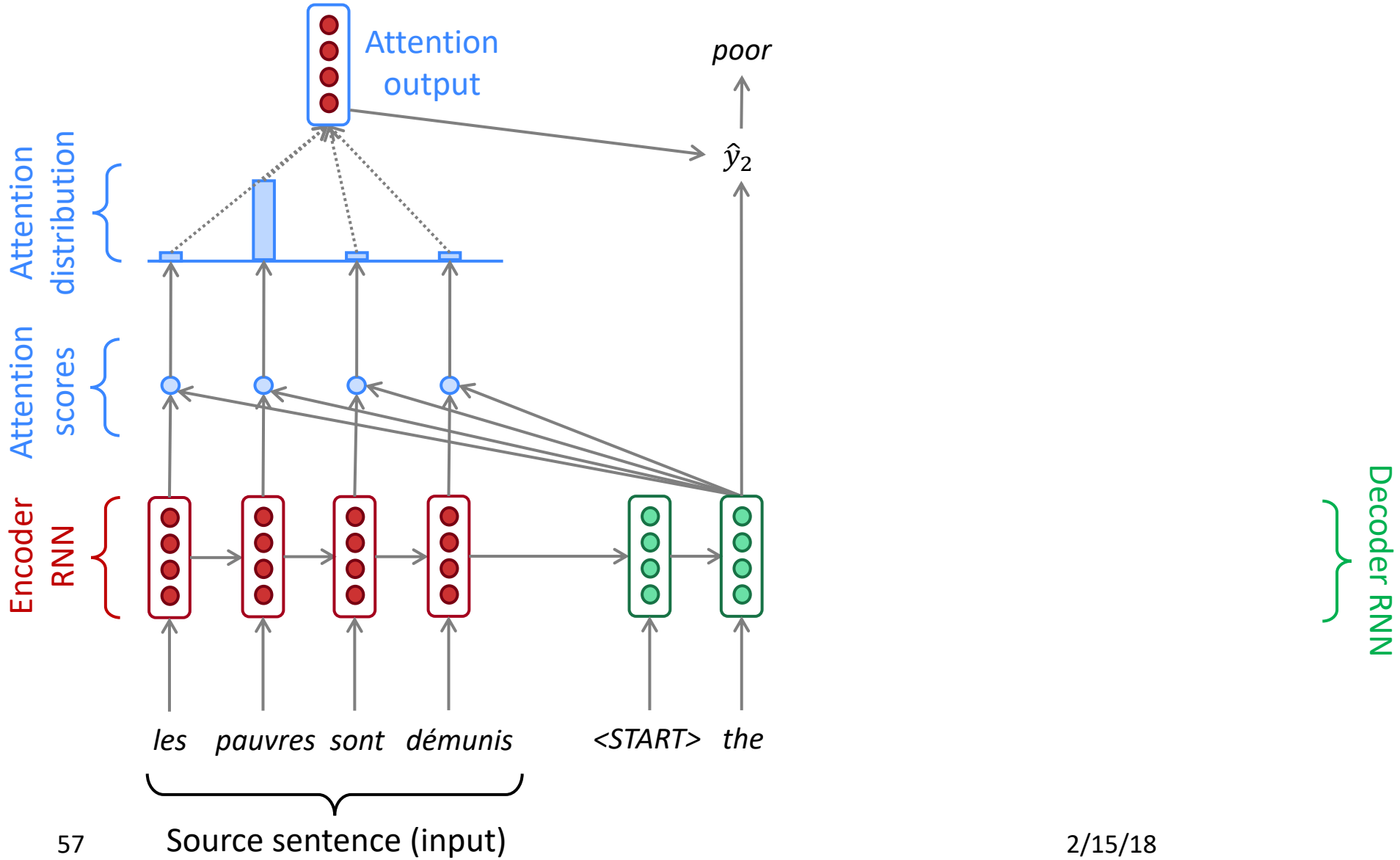
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the hidden states that received high attention.

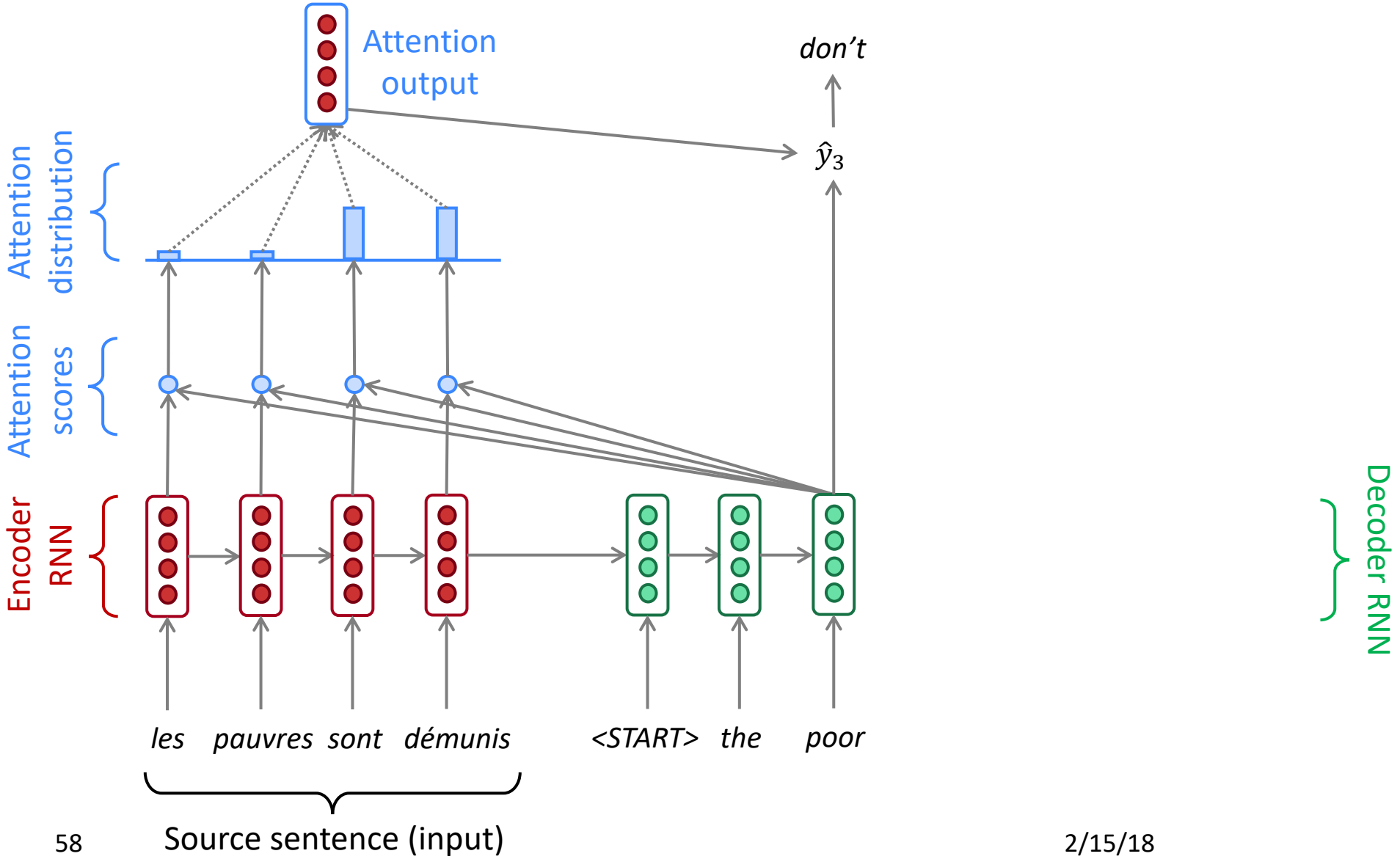
Sequence-to-sequence with attention



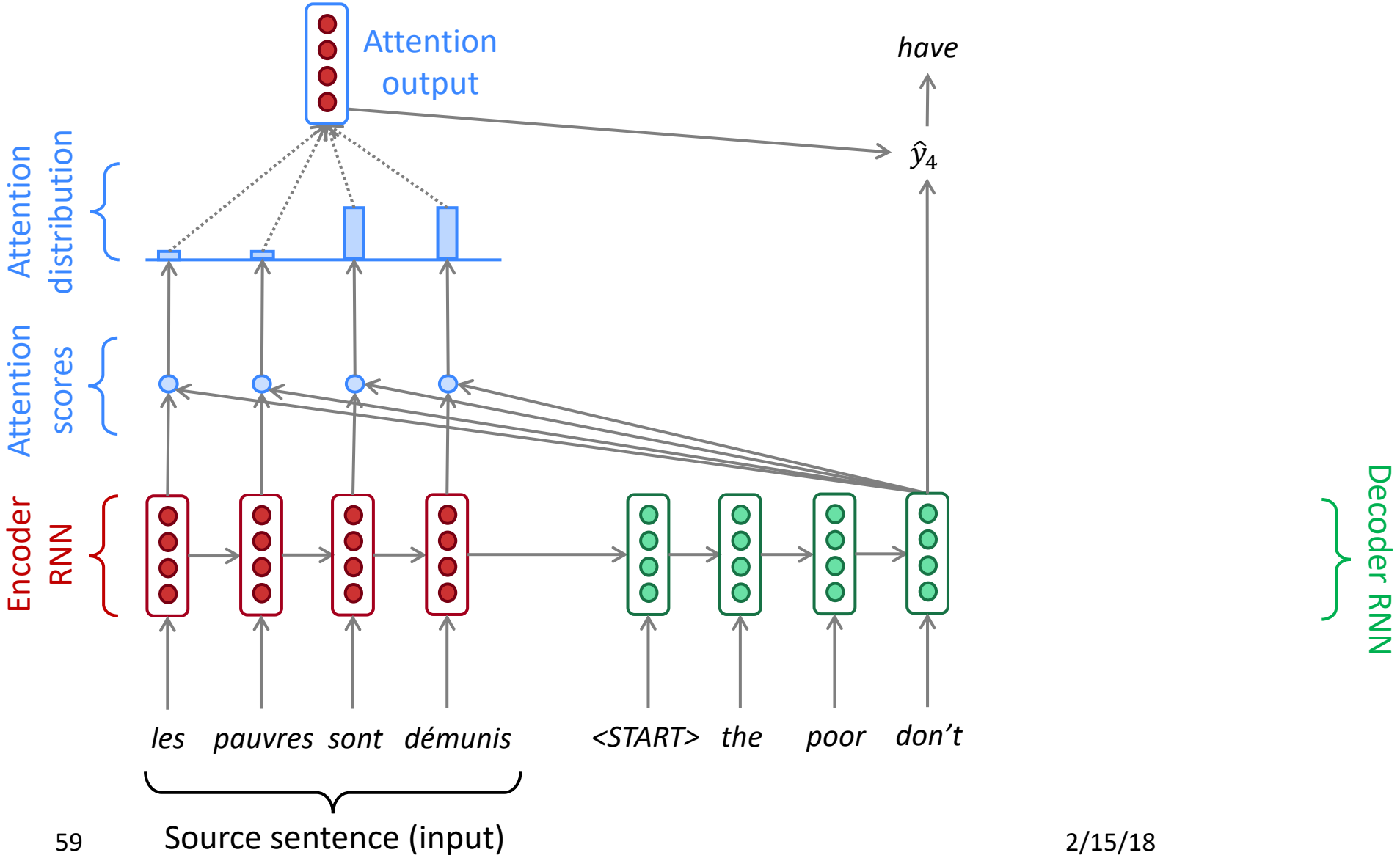
Sequence-to-sequence with attention



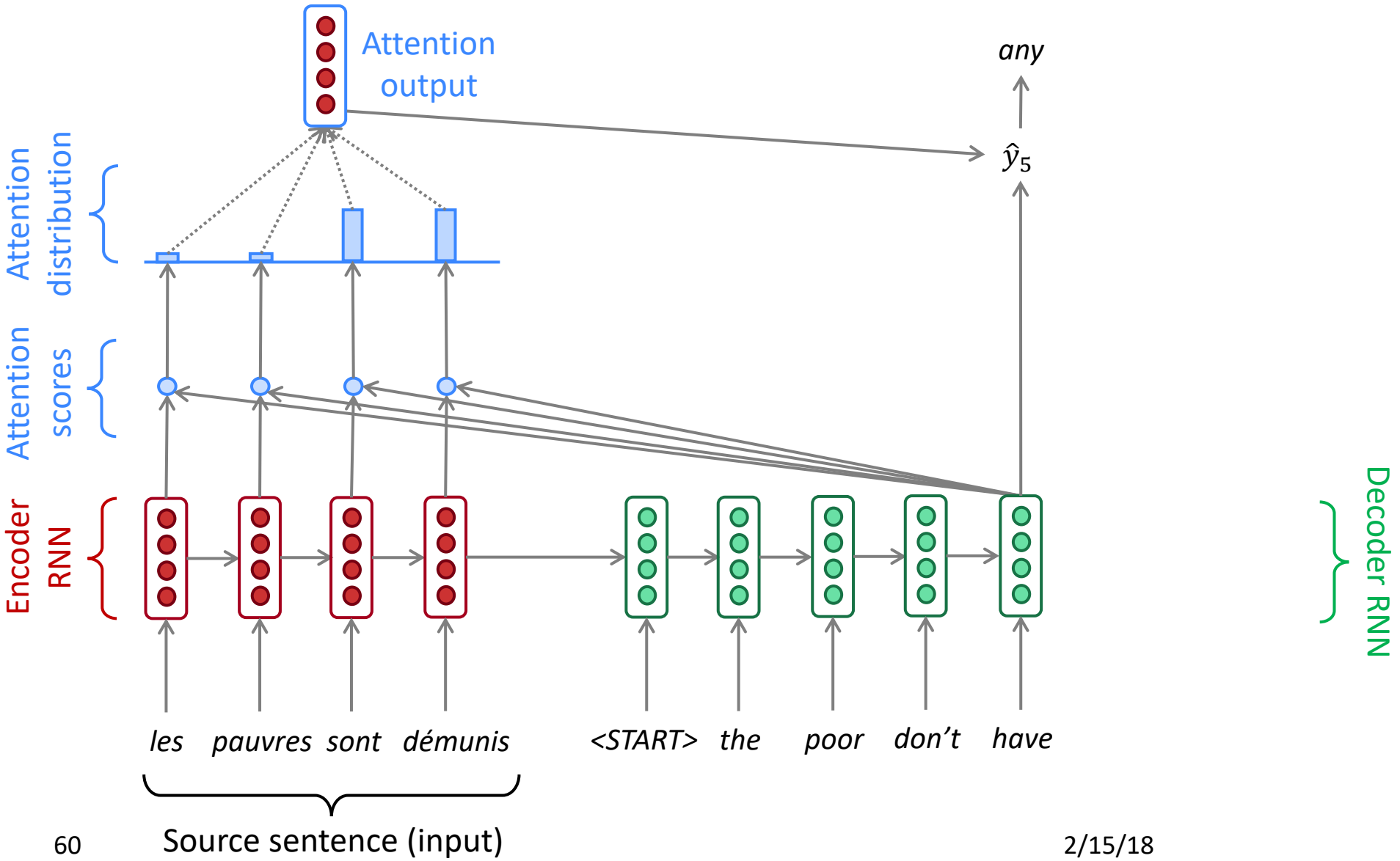
Sequence-to-sequence with attention



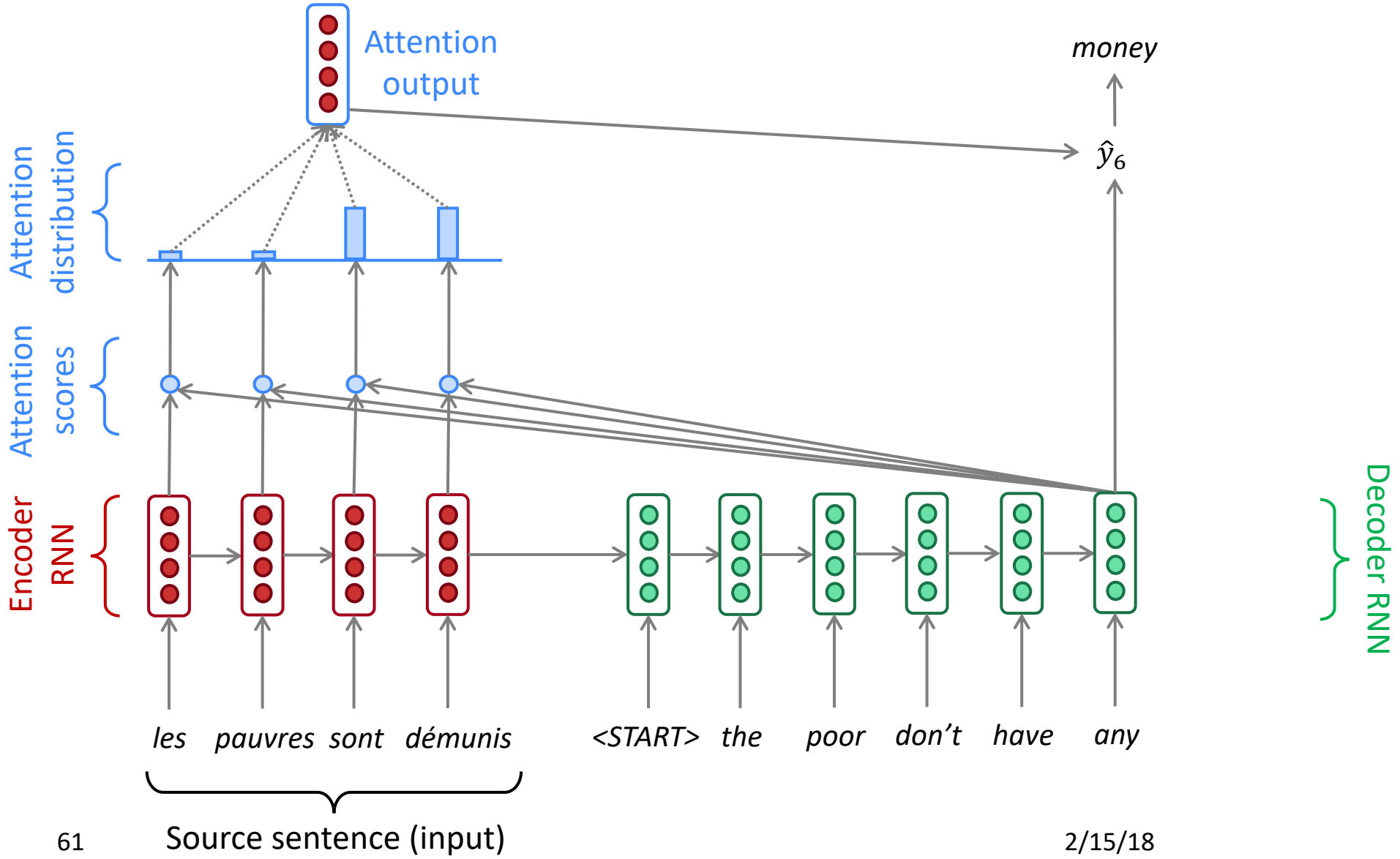
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Query, Key, Value

Attention의 3요소

- Query : 어떤 정보를 찾고 싶은지 요청을 날리면
- Key: 각 위치가 가지는 key값들과 비교하여 관련 있는 정보를 파악하고
- Value: 그들이 가지는 Value를 취합하여 사용한다

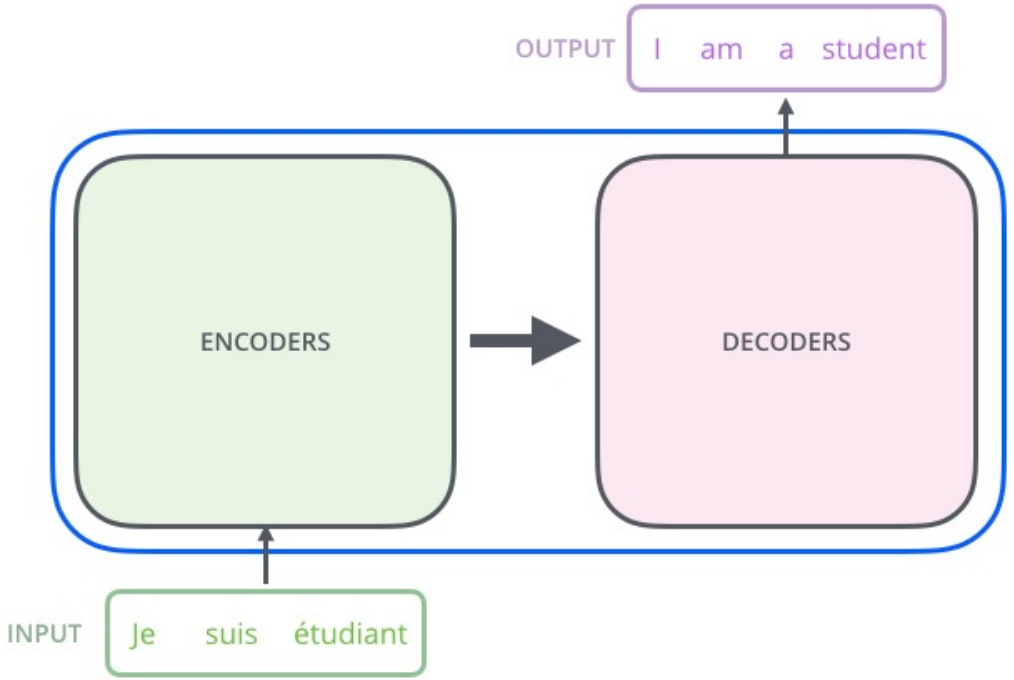
Query, Key, Value를 update 하는 방법

- 다양한 방법이 존재하며 이에 따라서 여러 구조로 나뉜다.
- 최근에는 Query Key의 inner product기반의 attention을 많이 사용하며 neural layer를 통과 시키며 변형을 줄 수 있다
- Multi-head attention은 query, key, value를 여러개를 각각이 정의하고 다양한 관점에서 attention을 실행하는 방법이다

Transformer

RNN도 필요 없고 Attention만 있으면 다 구현 가능하다!

- encoder와 decoder를 모두 attention을 기반으로 구현



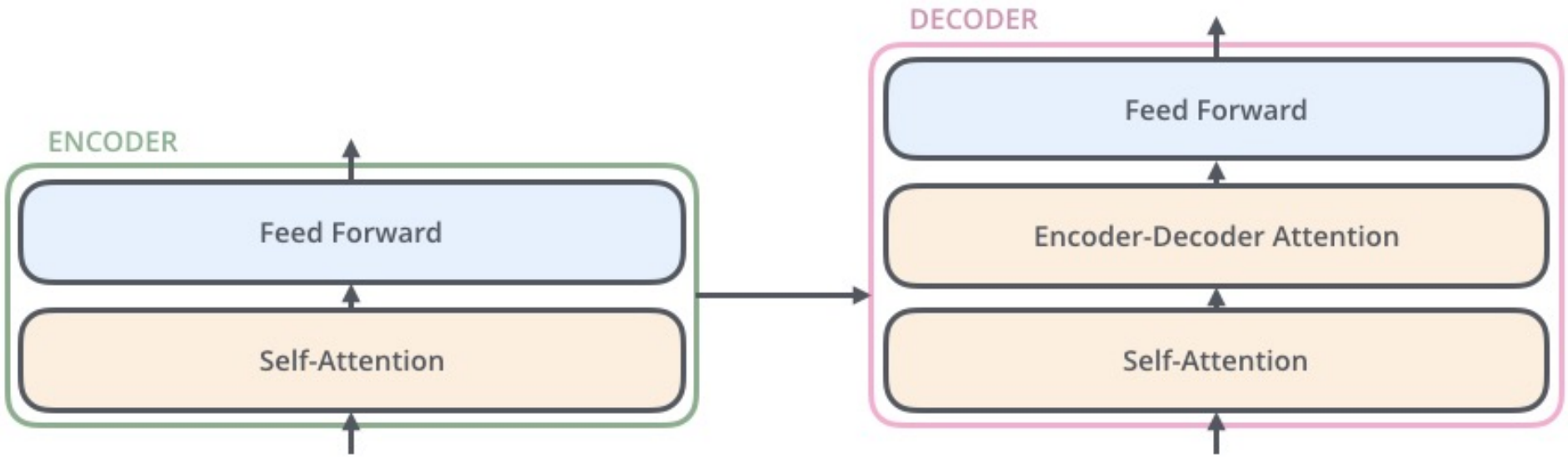
Self-Attention

입력 문장의 정확한 이해

- 같은 단어도 다른 의미로 사용되는 경우가 많이 존재
- Self-attention을 통하여 정확한 의미 파악

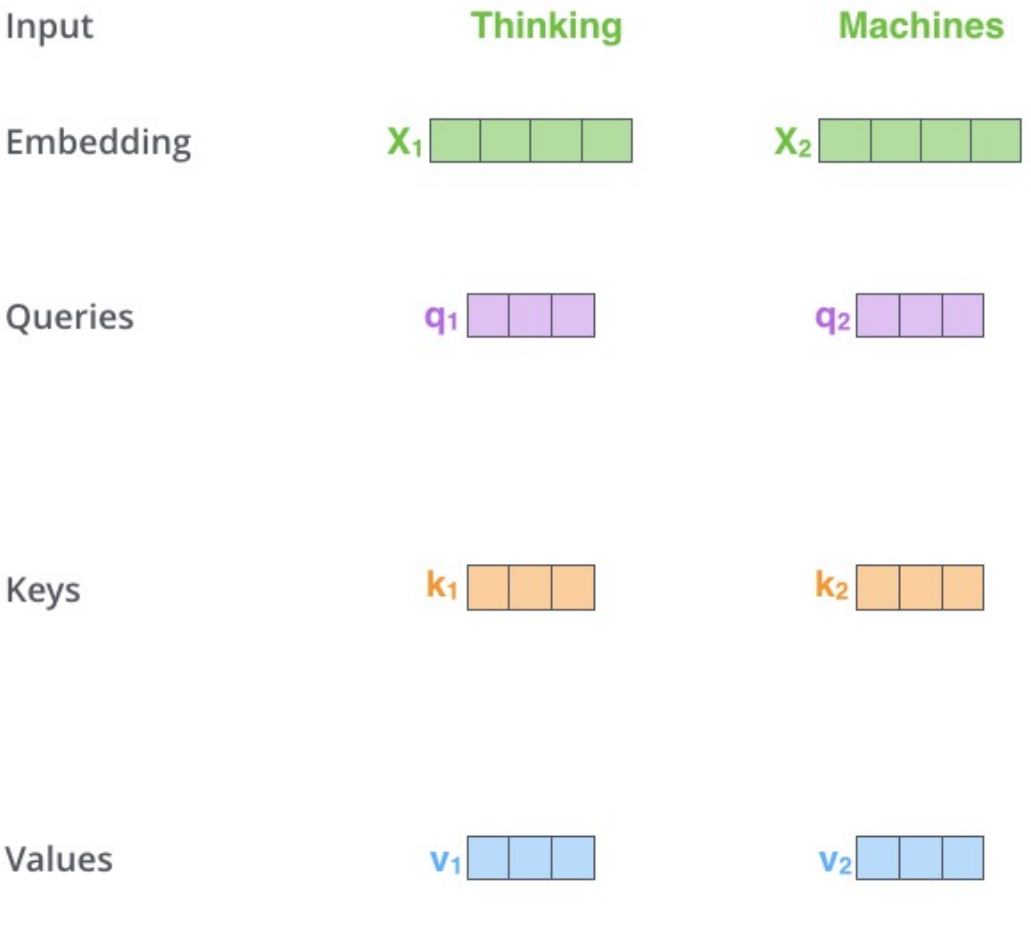
Decoder

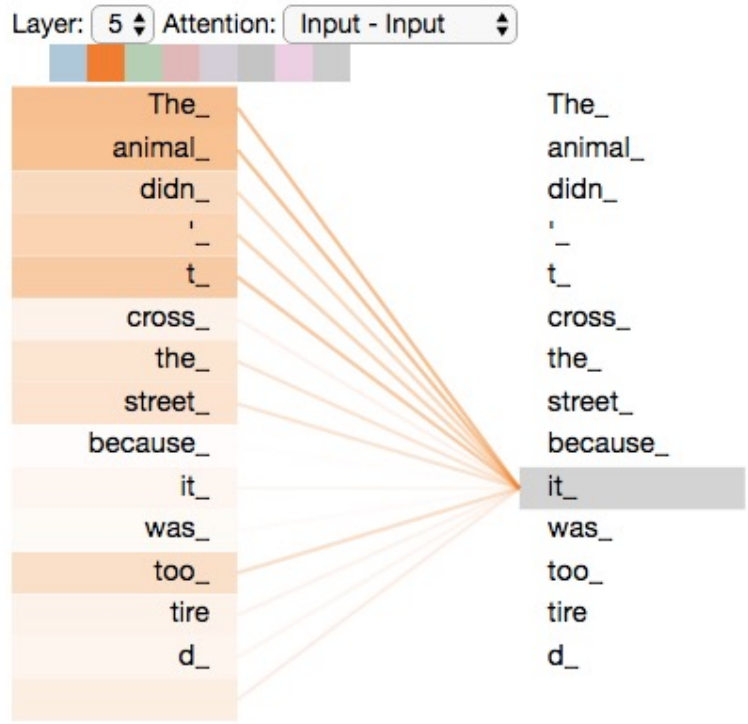
- 해석 문장에서도 self-attention을 통한 자연스러운 문장 형성



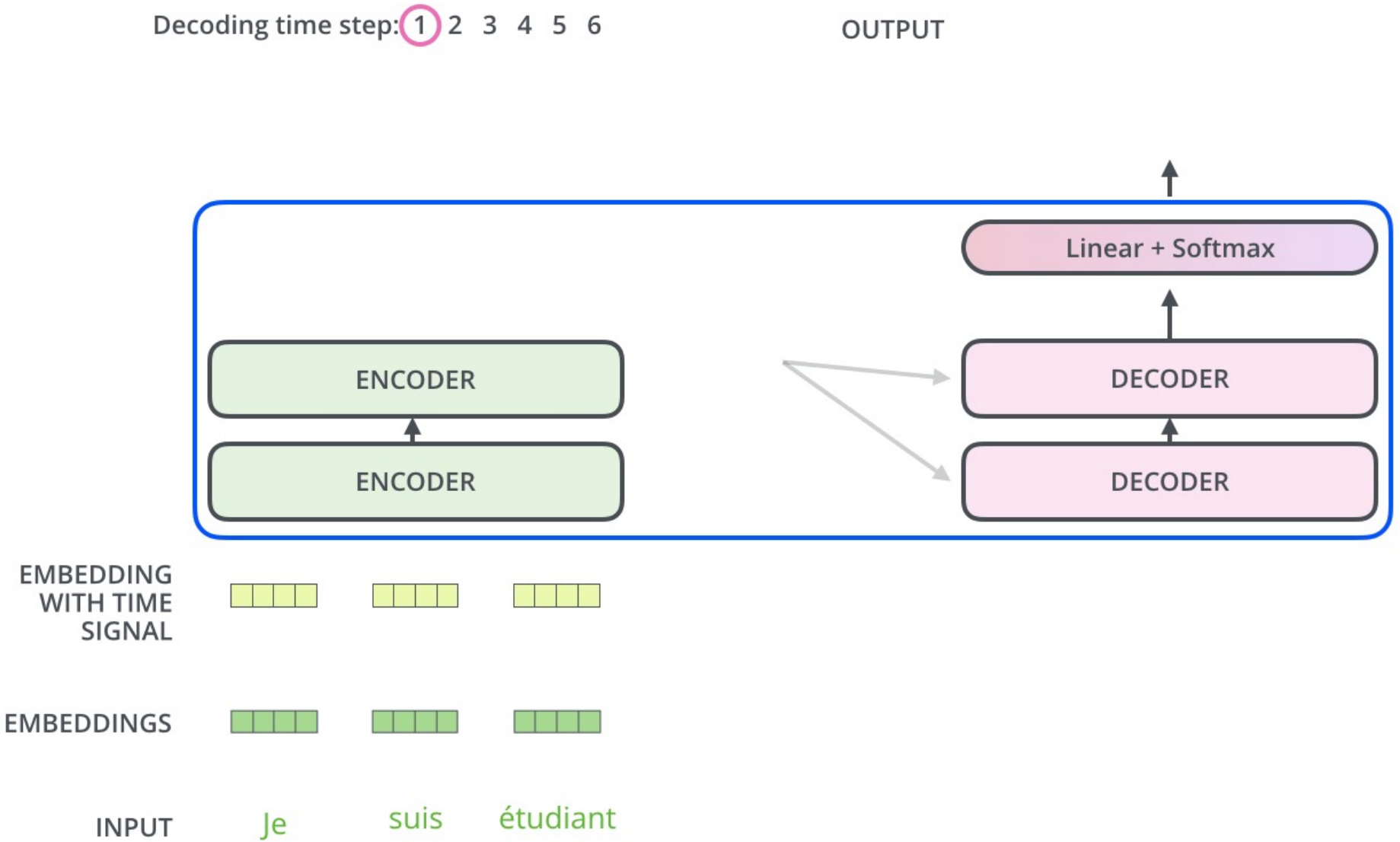
Self-Attention

동작 방법

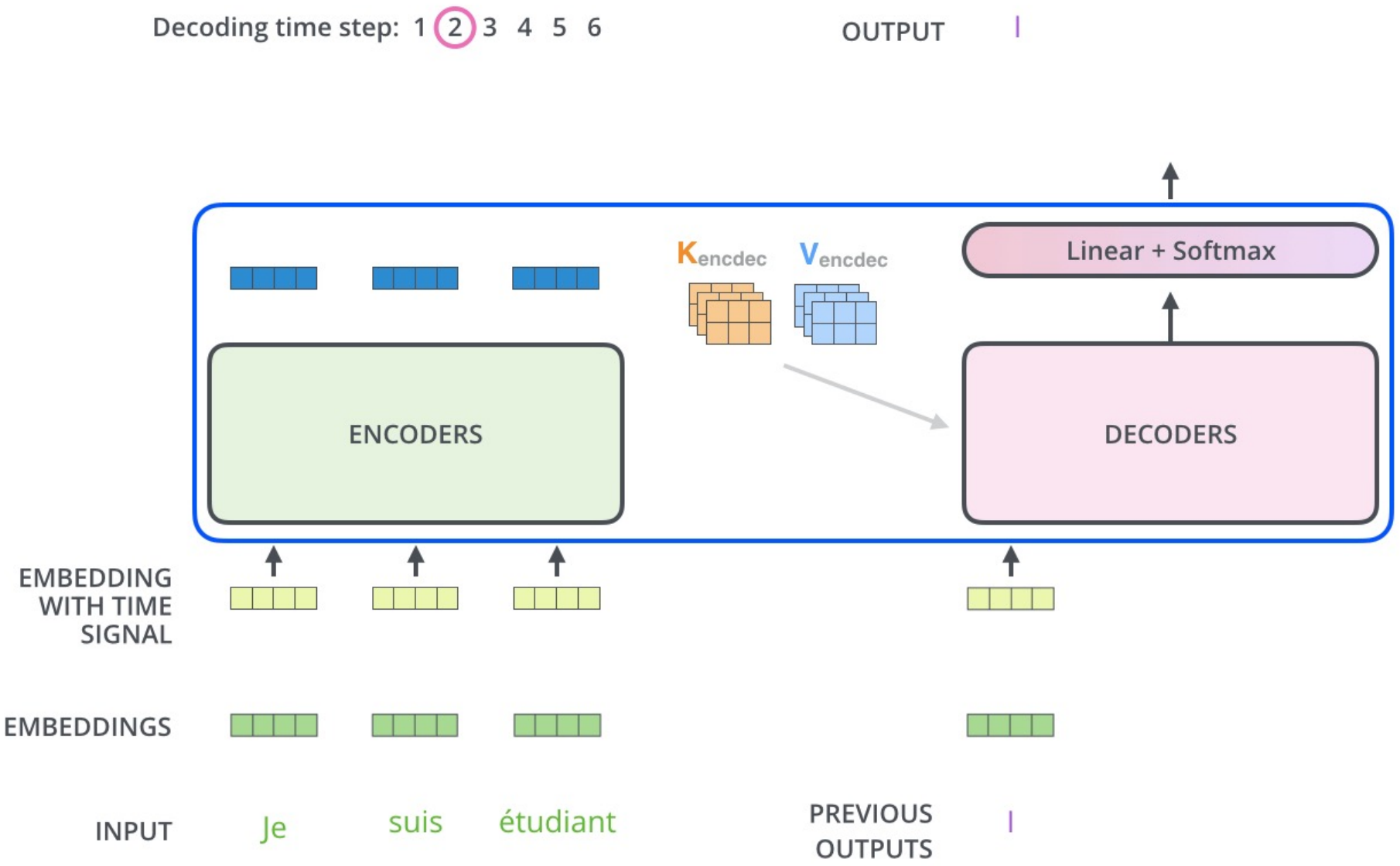




Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
Kencdec
Vencdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

감사합니다!