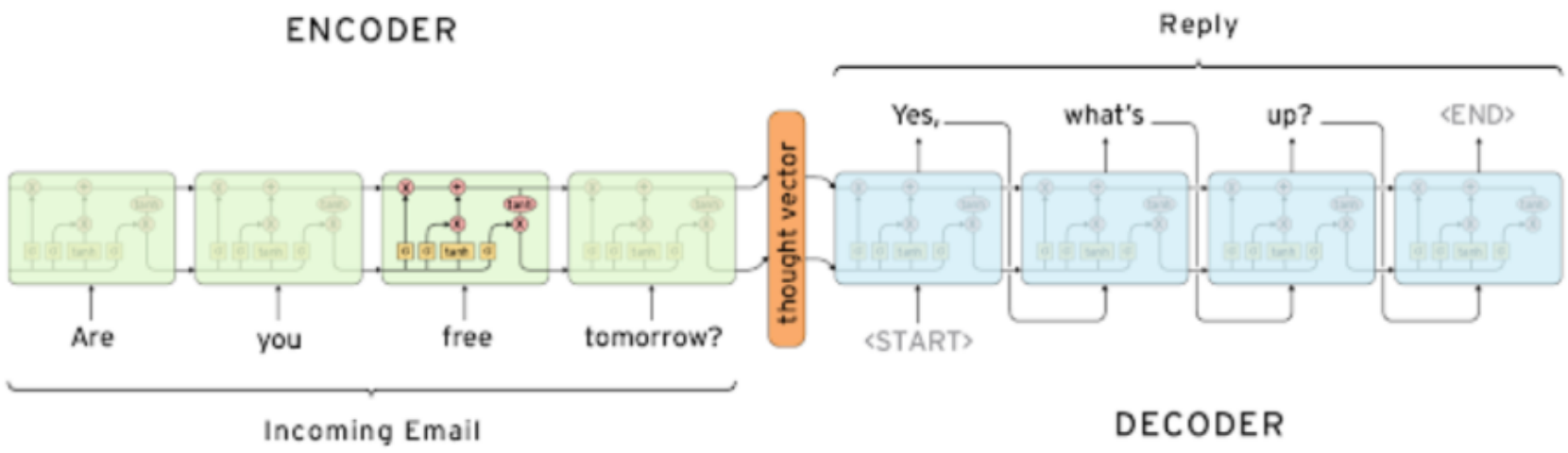
Seq2Seq (Sequence-to-Sequence) 개념



input(seq) → output(seq).

Encoder와 Decoder로 구성

params 공유 X

LSTM seq2seq example.

RNN Seq2Seq 모델 한계 (→ LSTM)

Encoder → Decoder

hidden vector 1개

⇒ Encoder 마지막 hidden state에 input seq에 대한 모든 정보를 담아서 전달

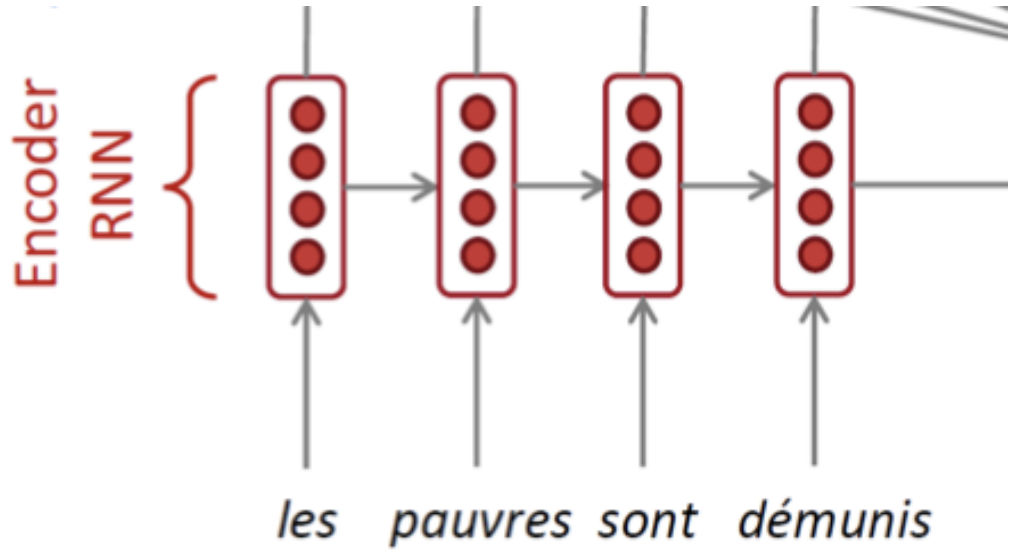
그거 길어지면 제대로 못담음 (특히 앞쪽 정보 손실)

"Bottleneck problem"

Encoder seq 정보는 많은데 이걸 하나의 vector에만 담아야해서 터지는 문제를

∴ 이거 해결하려고 Attention 씀

Encoder & Decoder (예시 - les pauvres sont démunis ⟷ the poor don't have any money)



기존이랑 비슷.

Word embedding vector가 차례대로 입력으로 들어감.

각 time step마다 다음으로 hidden state 전달

(마지막 hidden state는 Decoder의 시작 hidden state로 전달)

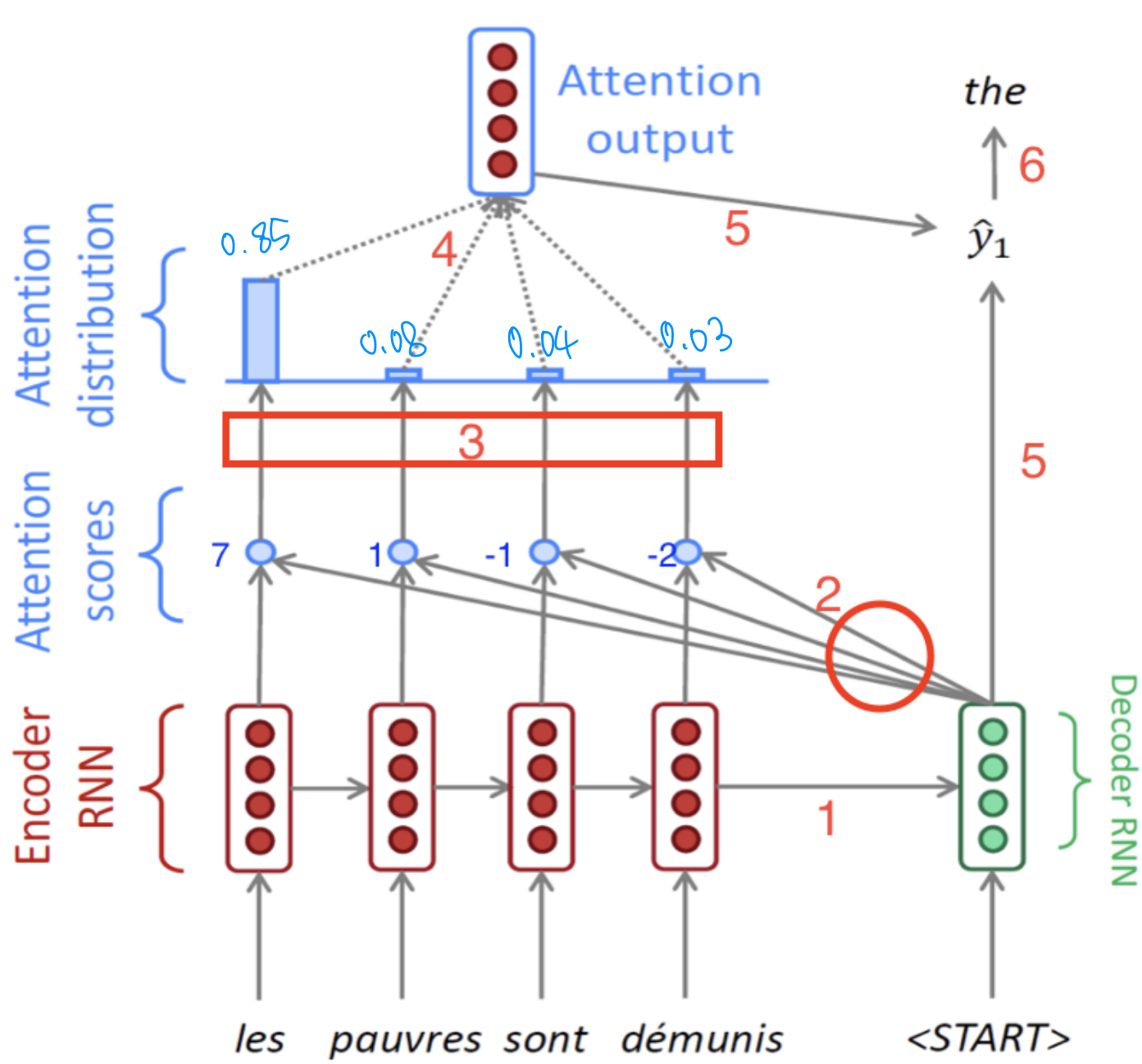
↳ 이 벡터(h_0^d)

+ [<START> token embedding vector

⇓

hidden state vector(h_1^d)

아래 그림의 1번에 해당



기존에는 이걸 그냥 다음 timestep으로 넘겼는데, Attention은 뭐가 더 있음

<2번 과정>

h_i^d가 Encoder seq의 각 단어들의 hidden state vector와 dot product를 수행함 (그림에서 1, 1, -1, -2)

↳ dot product 결과는 cosine similarity에 비례함

⇒ dot product 값이 클수록 두 벡터는 연관성이 큼!

* cosine similarity

$$sim(A,B) = \frac{A \cdot B}{\|A\| \|B\|} \quad for\ A, B \in V_n$$

* softmax function

<3번 과정>

위에서 구한 dot product 값들을 softmax 함수에 집어넣어서 확률값으로 바꿔줌 (그림에서 0.85, 0.08, 0.04, 0.03 / sum=1)

<4번 과정>

위에서 구한 확률값들은 Encoder의 각 hidden state vector의 가중치로 사용됨. (필요정보 취사선택)

Attention output vector = $\sum w_i h_i^e$
(context vector)

가중치 / encoder hidden state vector

<5번 과정>

context vector와 h_i^d를 concatenate하여 최종 output vector 구하기

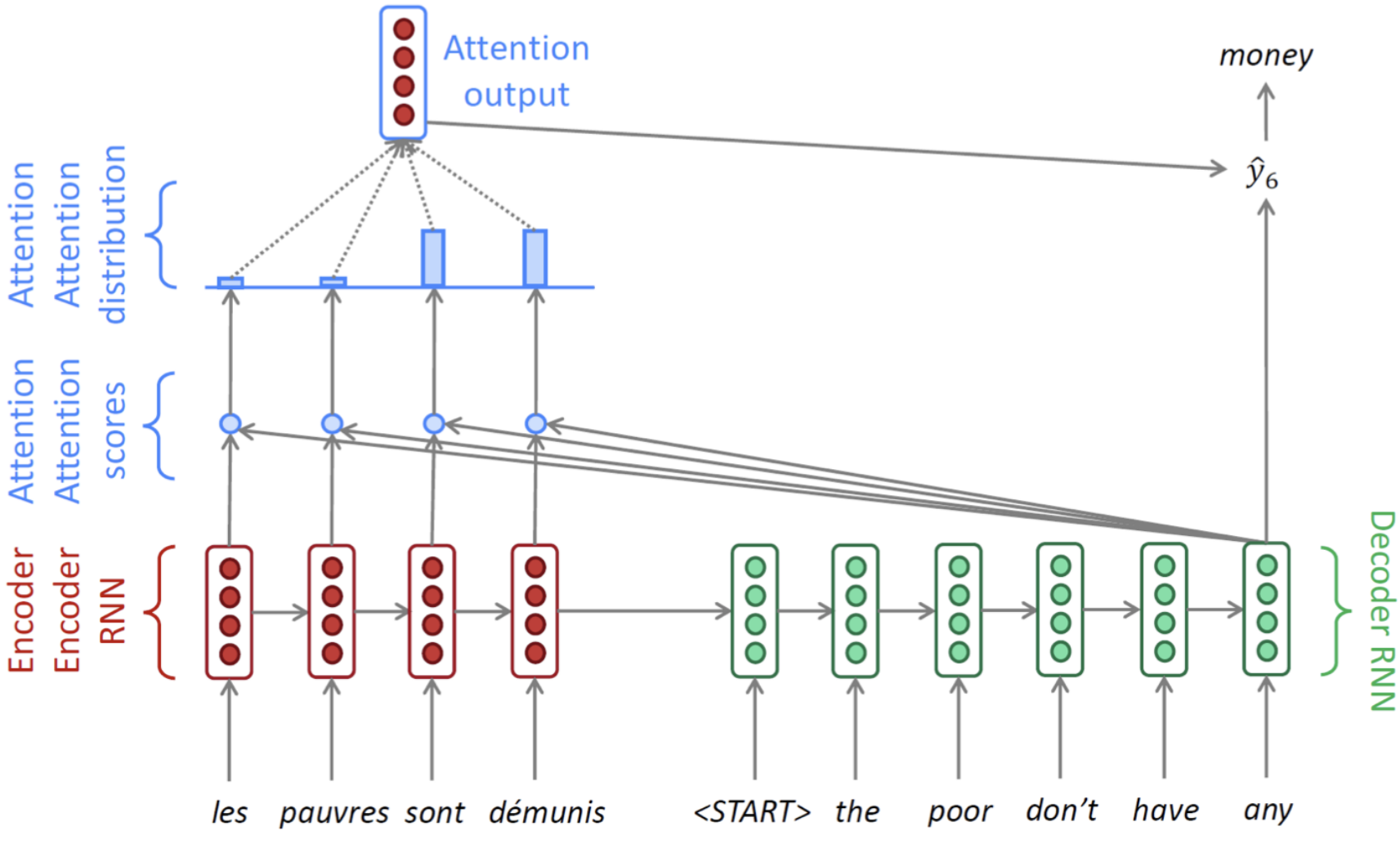
<6번 과정>

현재 time step의 다음 단어를 출력. (y_i)

↳ Q. 기존 seq2seq이랑 뭐가 다름?

A. 기존에는 현재 time step의 output vector만을 사용했는데 Attention에서는 context vector와 output vector 모두 이용해서 예측 개선

(최종결과 그림)


Attention output
money
$\hat{y}_6$
Attention distribution
Attention
Attention scores
Attention
Encoder RNN
Encoder
Decoder RNN
les
pauvres
sont
démunis
<START>
the
poor
don't
have
any