

1. 자연어처리 (NLP) 정의

• 자연어?

: 사람이 평소에 쓰는 말. → 부드러운 언어 : 같은 의미의 여러 표현
의미의 변화/생성/사라짐 (형태)

• 자연어 처리 목표

: 사람의 말을 컴퓨터가 이해 하는 것 :

- 문장은 "문자"로 구성 되고 문장의 "의미"는 "단어"로 구성
단어는 "의미의 최소 단위" → 즉 목표: 컴퓨터가 단어의 의미를 이해 하는 것

I am JSY : 각 단어의 의미??

2. 단어의 의미를 표현 하는 방법

1. 시소러스

: 사람이 직접 단어의 의미를 정의 함.

: 비슷한 의미의 단어 끼리 그래프로 계층적으로 표현.

ex) WordNet.

• 문제점

: 사람의 개입. ← 필요
: 부드러운 대처? →

※ 말뭉치
사람의 언어가 들어간 Text Data
or
NLP에 사용하기 위해 작성하고 모은

2. 통계 기반 기법

• "단어를 벡터로 표현" ⇒ 단어의 분산 표현.

→ 아이디어 ⇒ 분포가설 : "단어의 의미는 주변 단어에서 형성된다"
→ 맥락.

You say goodbye and I say hello.
window size = 1

• 분포가설에 기초해 주변 단어(맥락)의 등장 횟수를 Count.

	You	say	goodbye	and	I	say	hello	
You	0	1	
say	1	0	1	
.	

: 동시발생 행렬.

• 벡터간 유사도

- 코사인 유사도

- 문제점

• 동시발생 행렬에서 벡터간 유사도는 줄지 못한 척도

ex) the car, 조상연은,
the 와 car 가 유사한가? (의미적으) x ←

• 해결: 점별 상호정보량

• 상호 정보량

$$- PMI(x,y) = \log_2 \frac{P(x,y)}{P(x) \cdot P(y)}$$

PMI ↑ : 관련성 ↑

PMI ↓ : 관련성 ↓

- 문제점

• 데이터의 규모 ↑

• 희소벡터 (0이 많음)

• 해결: 차원 감소 (SVD)

• 차원 축소

• 차원을 줄이되, "중요한 정보"의 손실의 최소화 하도록.

3. 추천기반 기법 (word 2 vec)

• 통계기반은 단 한번만에 단어의 분산 표현을 획득

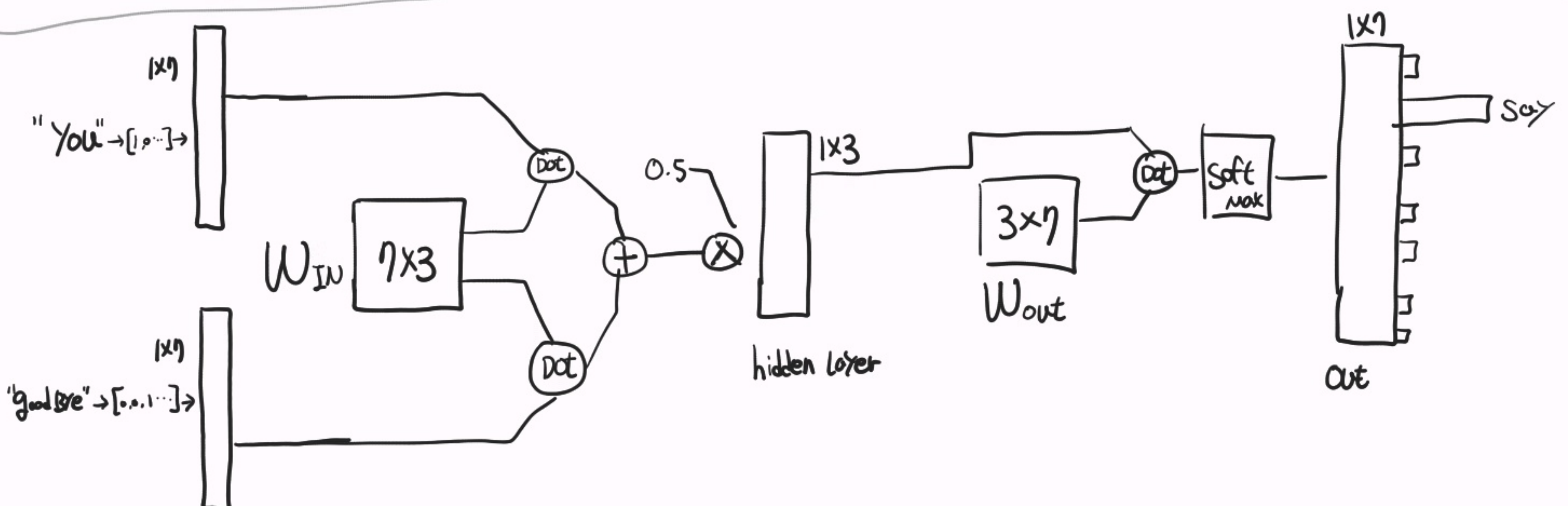
• 반면, 추천기반은 신경망을 통해 순차적으로 학습하여 가중치 갱신
(미네버리)

• 기본 아이디어

You [?] goodbye and I say hello.

• 맥락을 통해 단어를 특정해낸다.

• 입력 시퀀스를 one-hot encoding



- 학습이 잘된 모델의 가중치 (w_{in}, w_{out})가 단어의 분산 표현이다.
- 학습이 진행 될 수록 맥락을 통해 단어를 잘 추측하는 방향으로 가중치(= 분산 표현)가 개선 될 것.
- w_{in} 을 사용하는 것이 대중적인 선택.

• CBOW 와 Skip-gram



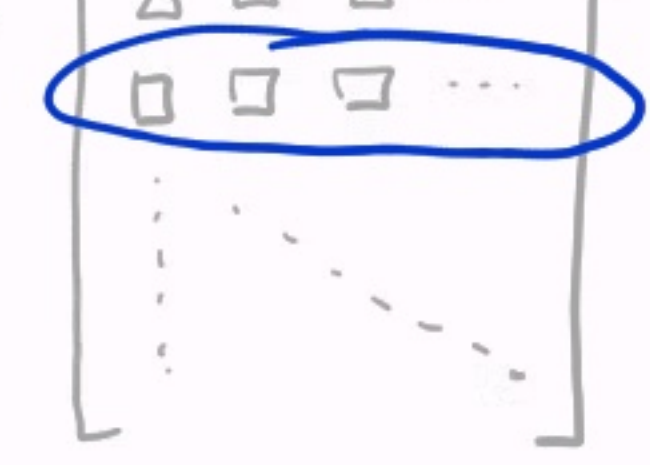
- 성능면은 skip-gram, 학습속도면은 CBOW

• word 2 vec 개선

1. Embedding

: 원형 벡터와 가중치 간 연산시 병목현상 발생

$$\text{soy} \Rightarrow [0 \ 1 \ 0 \ 0 \ 0 \ \dots] \cdot \begin{bmatrix} \triangle & \triangle & \triangle & \dots \\ \square & \square & \square & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} w$$



해결: 가중치 행렬에서 특정 행을 추출.

2. 다중 분류에서 이진 분류

: 은닉층과 w_{out} 그리고 softmax에서 병목현상 발생

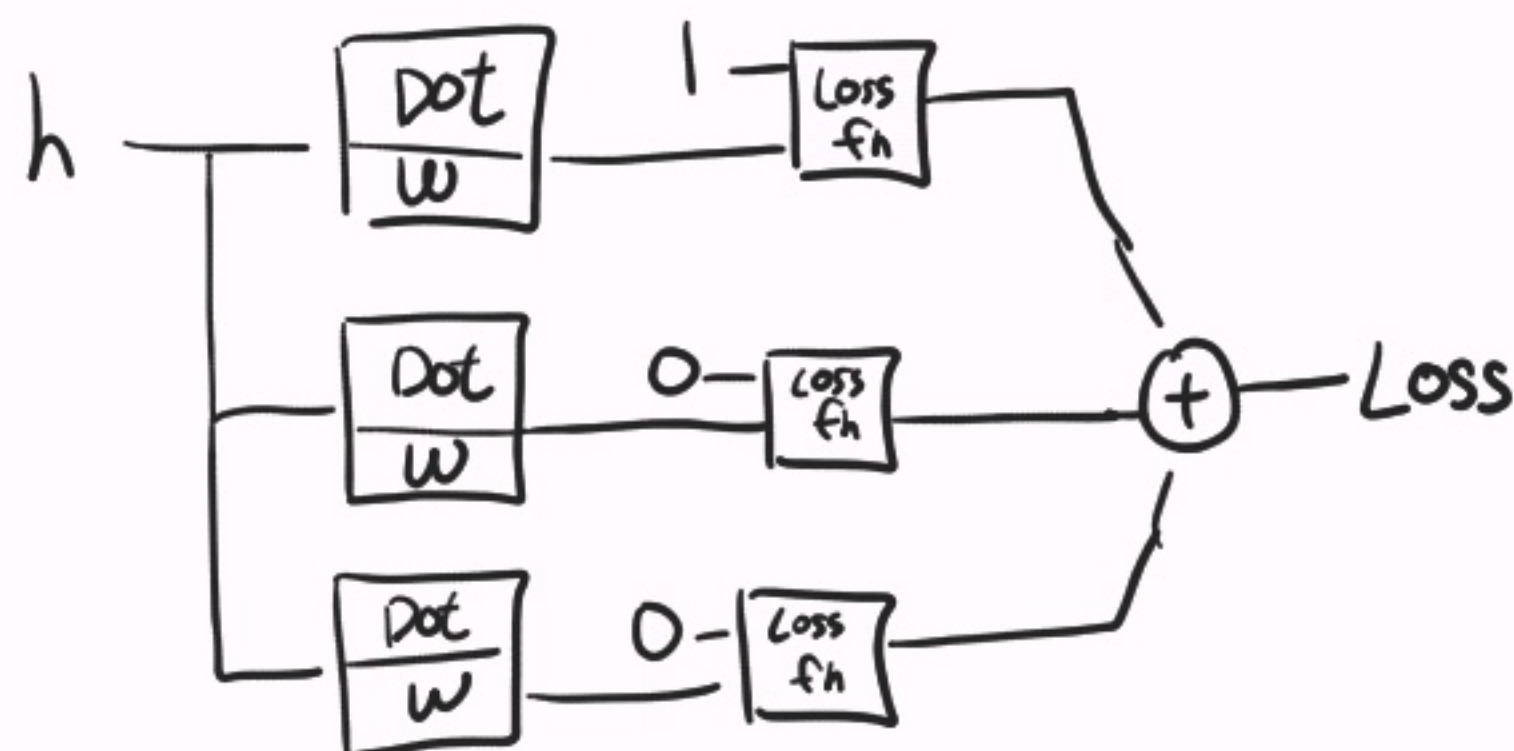
: $w_{out} = (1, 1)$ 행렬로 \rightarrow target word에 대한 점수

- 이후 Sigmoid() 2 확률 표현 : target word인가 아닌가?

: 네거티브 샘플링

- 위 경우는 target word의 점수를 높이는 경우만 해당됨.
- 타겟 word가 아닌 경우는 점수를 낮추는 경우도 고려해야 함.

— target word가인 word를 몇개 Random 샘플링해서 사용함.



4. 통계 기반 VS 추론 기반.

- 서로 관련되어 있고, 연결성이 있다(??!)

3. 순환 신경망(RNN)

- 시계열 데이터를 이해 하기 위해서는 이전의 문맥의 정보가 필요하다.
- RNN은 이런 시계열 데이터의 성질을 충분히 적용할 수있는 신경망 구조이다.

1. 언어 모델.

- 단어 나열에 확률을 부여
- 특정 Text sequence에서 그 sequence가 발생할 가능성을 확률로 평가

Text = $[w_1, \dots, w_m]$ 에서 $w_1 \sim w_m$ 이 순서대로 나타날 확률

$$P(w_1 \sim w_m) = P(A, w_m) = P(w_m | A) \cdot P(A) \rightarrow \dots \text{분해가능.}$$

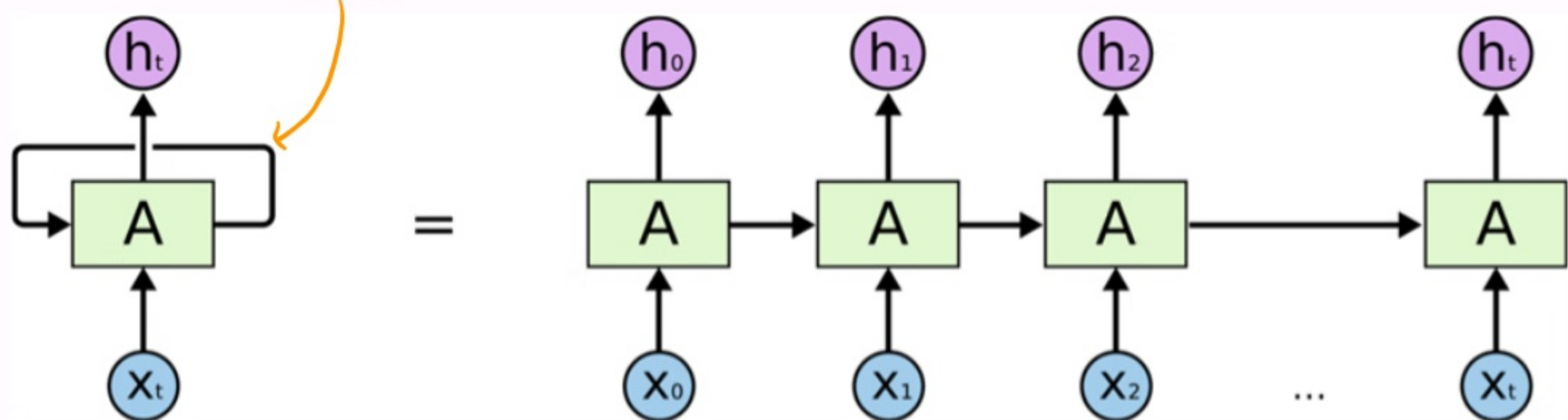
$$A = w_1 \sim w_{m-1}$$

$w_1 \ w_2 \ w_3 \ \dots \ w_{m-1} \ w_m$

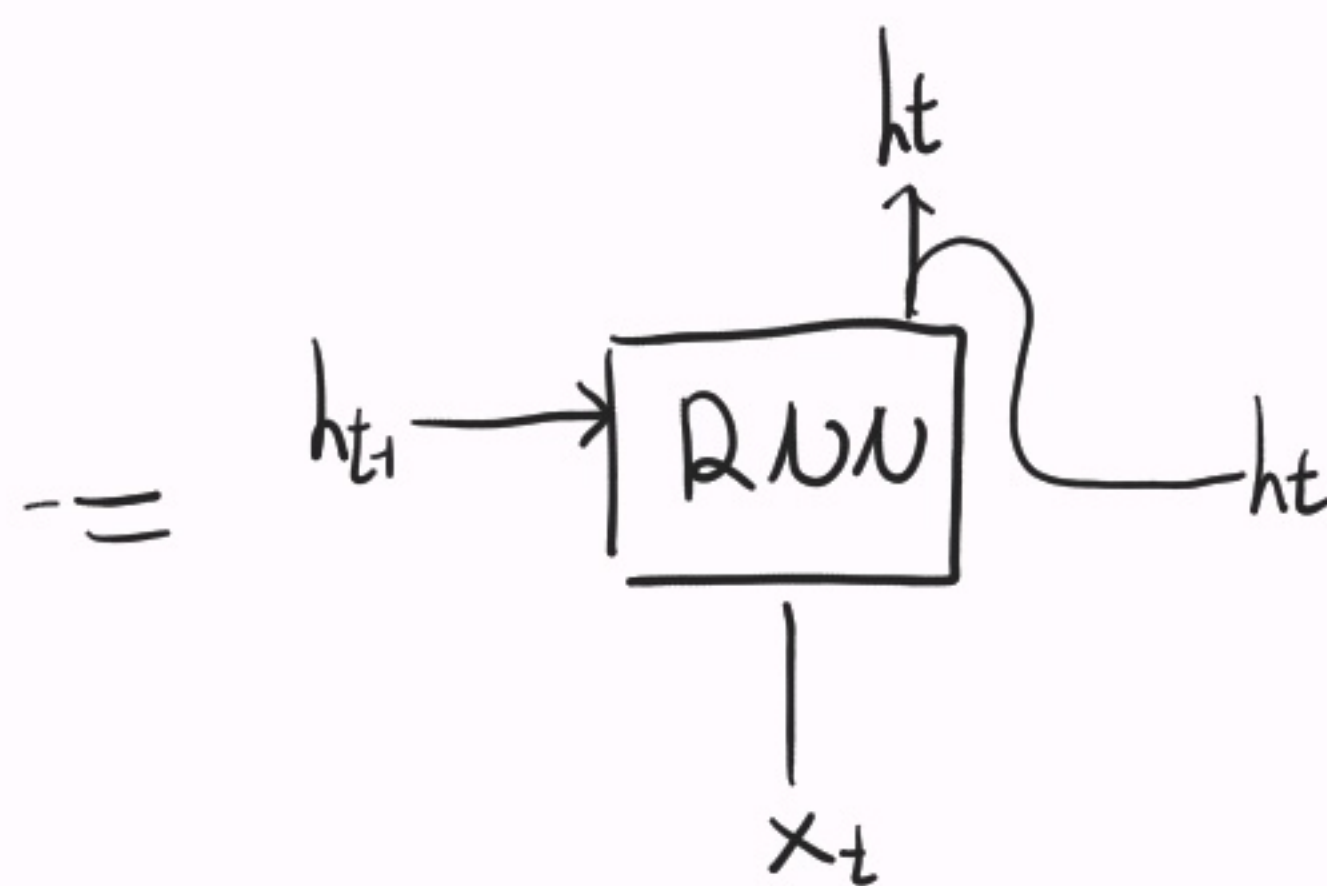
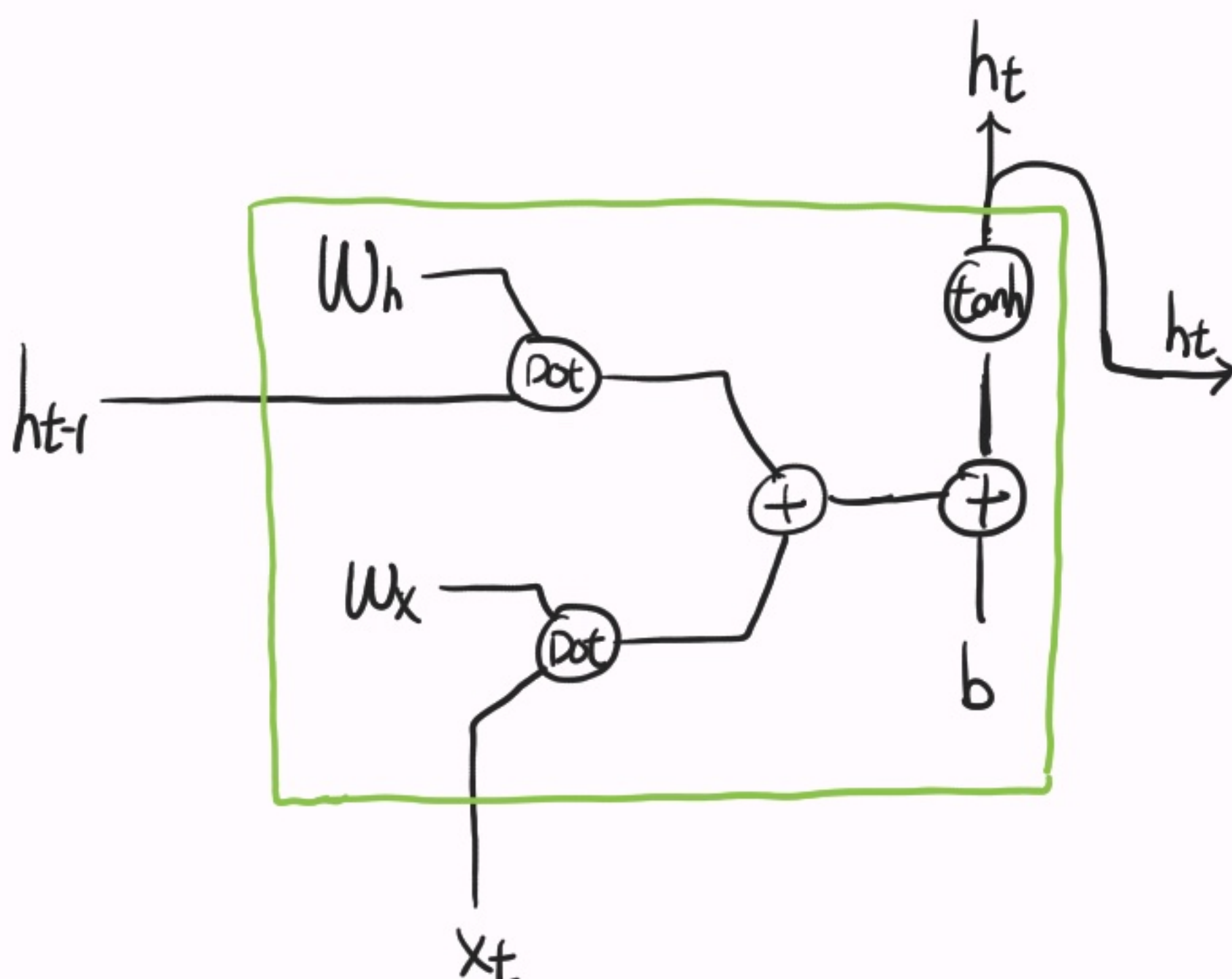
자신의 왼쪽 단어(맥락)을 조건으로 사용

2. RNN 정의

- 데이터가 순환 할 수있는 경로를 생성.



- 입력 x_t , 이전 계층의 상태 h_{t-1}
- RNN 순전파 식 $h_t = \tanh(h_{t-1} \cdot w_h + x_t \cdot w_x + b)$
- RNN에도 오차 역전파가 가능 하다. (BPTT)
 - 하지만 시계열 데이터다 보니까 역전파에서 많은 컴퓨팅 자원이 소모됨.
 - 해결: **Truncated BPTT**
 - RNN 계층을 나눠서 학습
 - 순전파의 연결은 살려둬야함.



3. RNN의 문제점.

Gradient Vanishing

1. $\tanh()$

- \tanh 의 도함수의 범위는 0~1 이다.
- 즉 RNN 계층을 여러번 거칠 수록 기울기가 작아지게 됨.

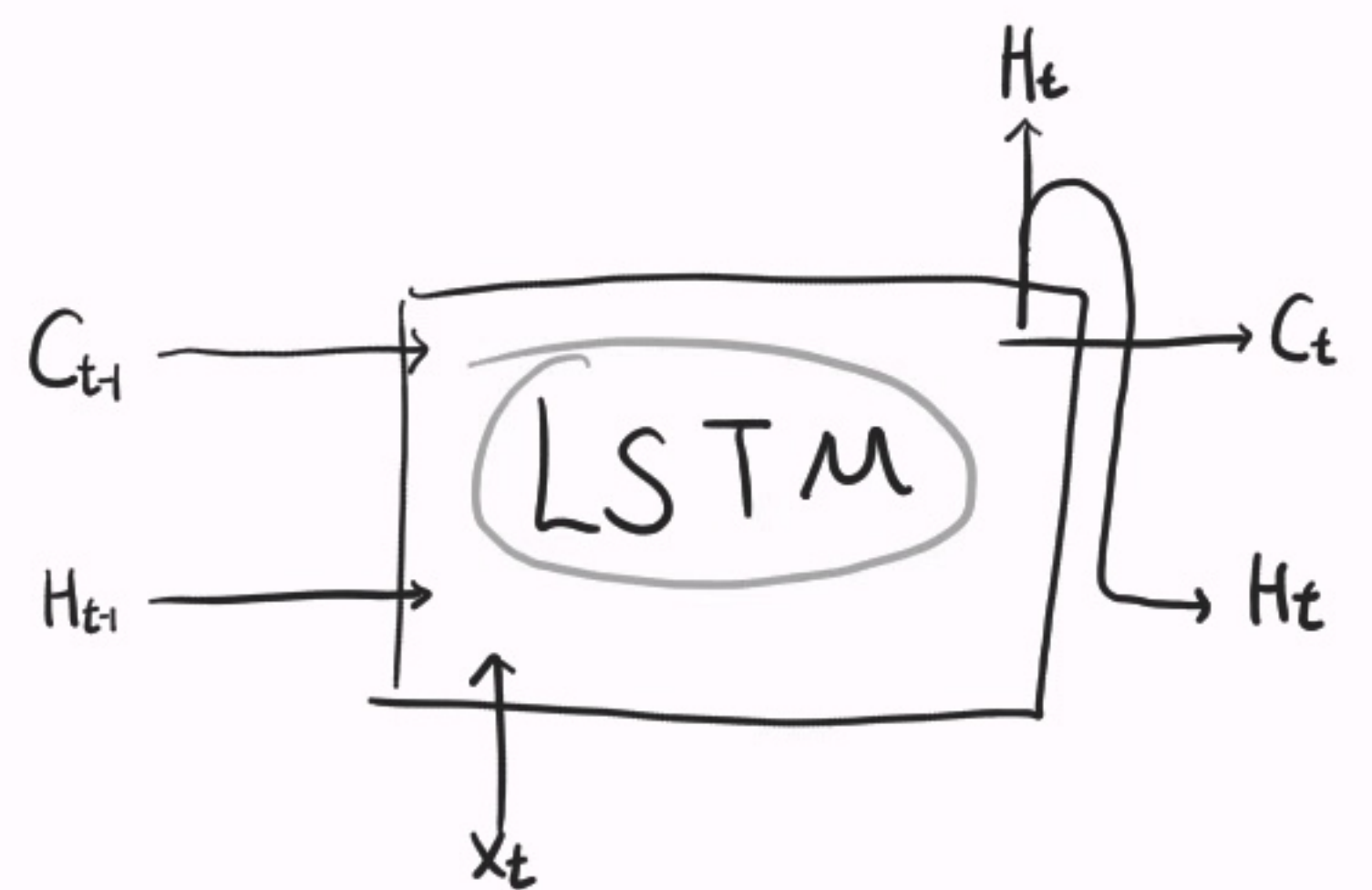
2. Dot 연산.

- W_h 는 계층을 지날때 마다 자기 자신이 곱해진다.
- 발산과 소실의 문제점.

해결: LSTM

4. LSTM.

- RNN에 "게이트"라는 컨셉이 추가됨
- 게이트를 통해서 이전 상태의 정보를 얼마큼(ex 70%) 반영할지 제어 가능
 - Sigmoid 함수로 그 반영 비율을 결정.
- 새로운 상태(?) "기억셀" C 가 존재



• H_t

- H_t 는 기억셀 C_t 를 $\tanh()$ 에 대입한 값과 → 원소별 곱
Output Gate 값(이하 0)의 어디까지나 곱으로 이루어진다.
- $H_t = 0 \odot \tanh(C_t)$

• Output Gate (o)

- h_t 에서 $\tanh(C_t)$ 가 다음 시각(t)의 은닉 상태에 → 얼마큼 내보낼 것인가.
얼마나 중요한가를 담당
- Output Gate는 입력(x_t)과 이전 상태(H_{t-1})로 계산된다.
- $0 = \text{Sigmoid}(x_t \cdot W_x + h_{t-1} \cdot W_h + b^o)$

• 기억 셀 (C_t)

- C_t 는 어떻게 계산 되는가.
- $C_t = f \odot C_{t-1} + g \odot i$ 으로 표현 되는데
 - C_{t-1} 는 이전 상태의 기억 셀이고
 - f, g, i 를 알아 보자.

- forget Gate (f)

- 현재 시각의 기억 셀(c_t)의 불필요한 정보를 잊게 하는 Gate 이다.
- 이전 상태의 기억 셀에서 얼마만큼을 잊게 할지 제어한다.
- $f = \text{Sigmoid}(x_t \cdot w_x^f + h_{t-1} \cdot w_h^f \cdot b^f)$
- f 는 c_{t-1} 에 아다마르 곱으로 곱해져 c_t 각 원소의 값을 얼마만큼 잊게 할지 결정한다.

- 추가되는 가격 정보 (9)

- forget Gate에서 잊고있는 현재 시점에서 정보를 제거해 준다.
- $g = \tanh(x_t \cdot w_x^g + h_{t-1} \cdot w_h^g + b^g)$
- g 는 Gate가 아니고 C_t 이 현재 상태의 기억정보를 의미한다 ($\because \tanh$)

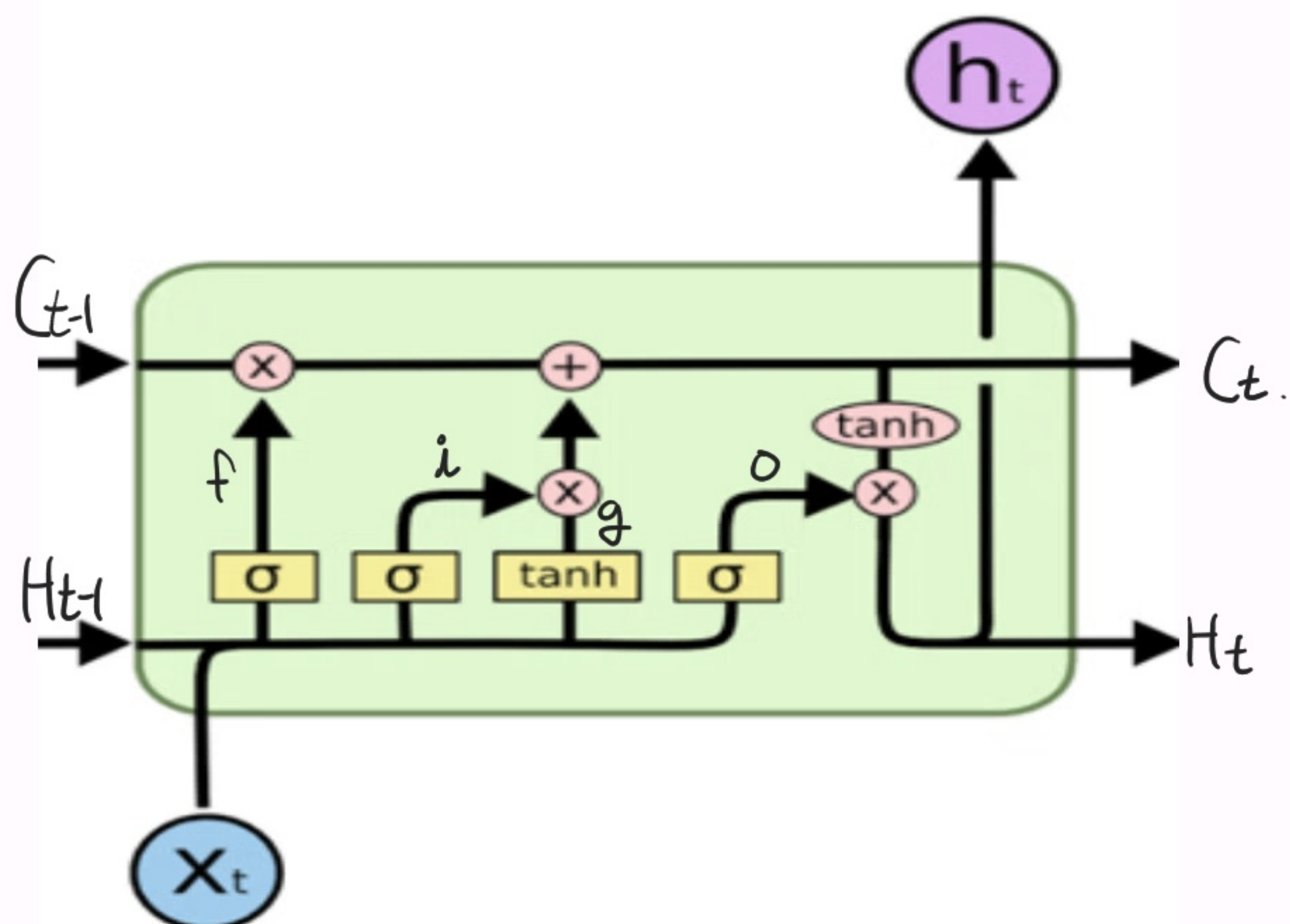
- Input Gate (i)

- 현재 기층 셀에 현재 정보 추가하는 gate에 공해지는 Gate
- 새로 추가 될 정보의 가치가 얼마큼인지 판단 한다.
- $i = \text{Sigmoid}(X_t \cdot W_x + H_{t-1} \cdot W_h + b^i)$
- 즉, Input 게이트에 의해 가중된 정보가 G_t 에 추가 되는 것
- i 또한 gate에 아다마르 곱으로 gate의 각 원소의 반영 비율로써 공해짐.

- 정리.

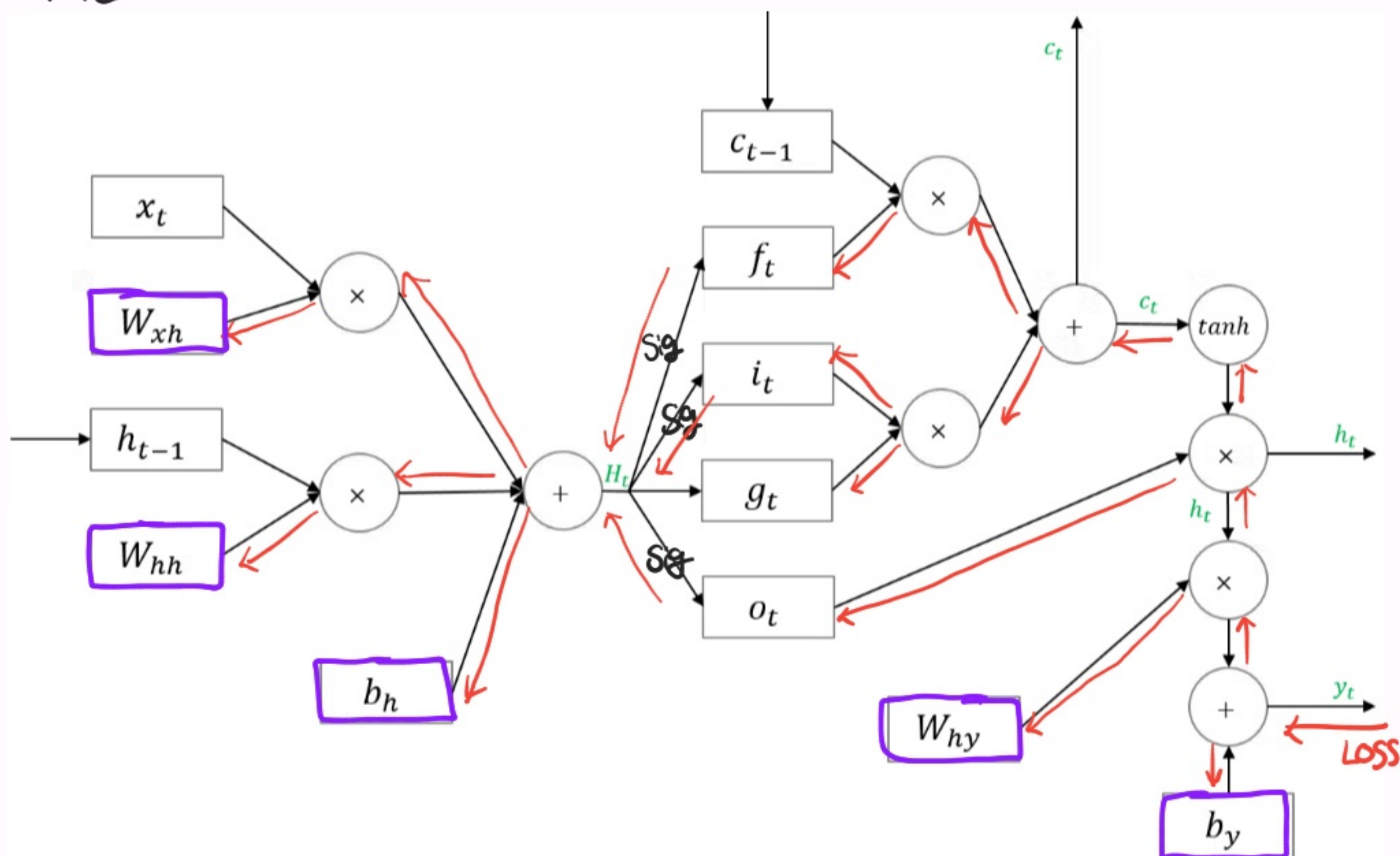
- $h_t = \tanh(c_t) \odot o$
- $c_t = f \odot c_{t-1} + g \odot i$
- $o = \text{Sigmoid}(x_t \cdot w_x^o + h_{t-1} \cdot w_h^o + b^o)$
- $f = \text{Sigmoid}(x_t \cdot w_x^f + h_{t-1} \cdot w_h^f + b^f)$
- $g = \tanh(x_t \cdot w_x^g + h_{t-1} \cdot w_h^g + b^g)$
- $i = \text{Sigmoid}(x_t \cdot w_x^i + h_{t-1} \cdot w_h^i + b^i)$

사용하는 가중치 ($W_x^{[0,f,g,i]}, W_h^{[0,f,g,i]}, b^{[0,f,g,i]}$) 12개



• LSTM 역전파.

- 순전파 식도 있고 특별한 어려움 없이 계산 가능
- 계산 그래프



• 추가적인 성능 개선.

1. 다중화.

- T 방향 말고, LSTM 계층을 쌓으면 성능 향상을 기대할 수 있다.
- 다음 계층 LSTM의 입력은 이전 계층의 H_t^{L-1} 이다.
- 몇층을 쌓을지는 적절하게 결정

2. Drop-out.

- 과적합을 방지
- 특정한 w에 대해서 값이 커지는 것을 방지
- 이 또한 T방향으로 쌓지 말고 깊이 방향으로 쌓아야 한다.
- 변형 드롭아웃 : 시간 방향으로도 drop-out을 적용

3. 가중치 공유

- 특정 계층에서 사용하는 가중치를 다른 계층에서도 사용
- 매개 변수의 수가 크게 줄어들고 정확도도 향상! → 과적합 예방

여기서 궁금한거

1. word2vec 의 개선사항중 병목 현상을 막기 위해서 embedding을 썼는데,

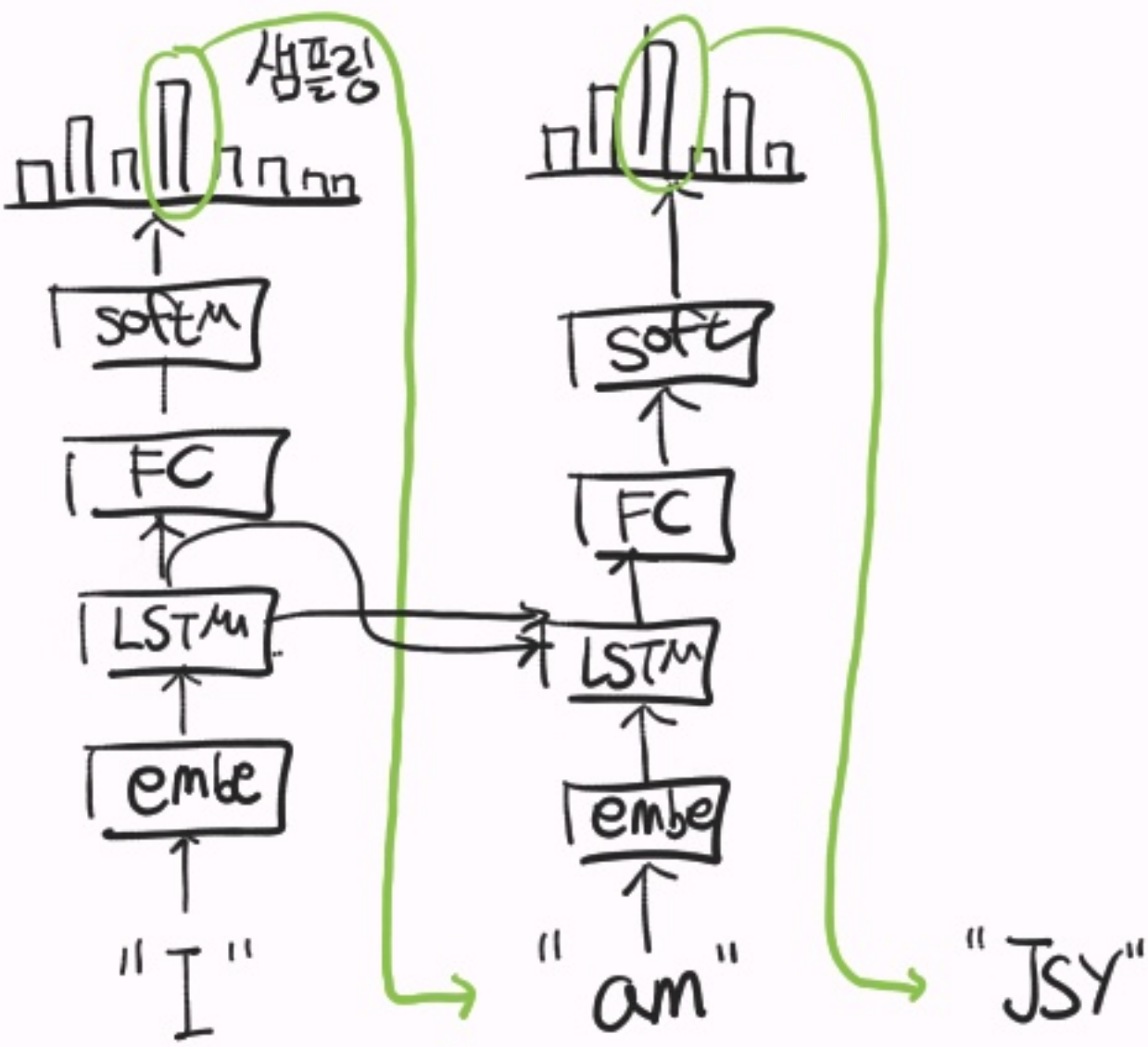
LSTM에서 embedding layer는 입력 단어의 ID를 받아 분산 표현으로 바꿔 준다고 했다.

둘이 같은 의미인가?

2. embedding은 왜 가중치가 있는가...

5. Seq2Seq

• 문장 생성 과정



RNN으로도 문장 생성을 할 수 있지만, 이색하다.
⇒ 더 좋은 모델? ⇒ Seq2Seq

• Seq2Seq 정의

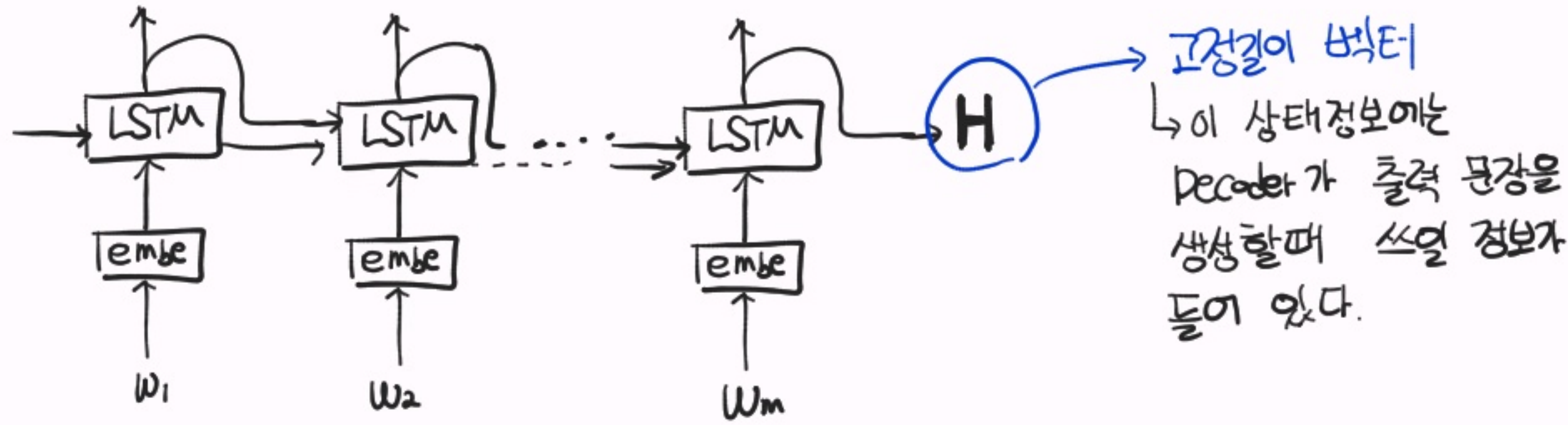
- 2개의 RNN 모델을 사용함.
- 원리

: ENCoder - DeCoder

- ENCoder는 입력 데이터를 인코딩하고 Decoder에게 인코딩한 정보를 전달.
- DeCoder는 인코딩 정보를 바탕으로 출력 시퀀스를 생성.
- 인코딩 정보에는 필요한 정보가 조밀하게 응축되어 있음 (Context vector)

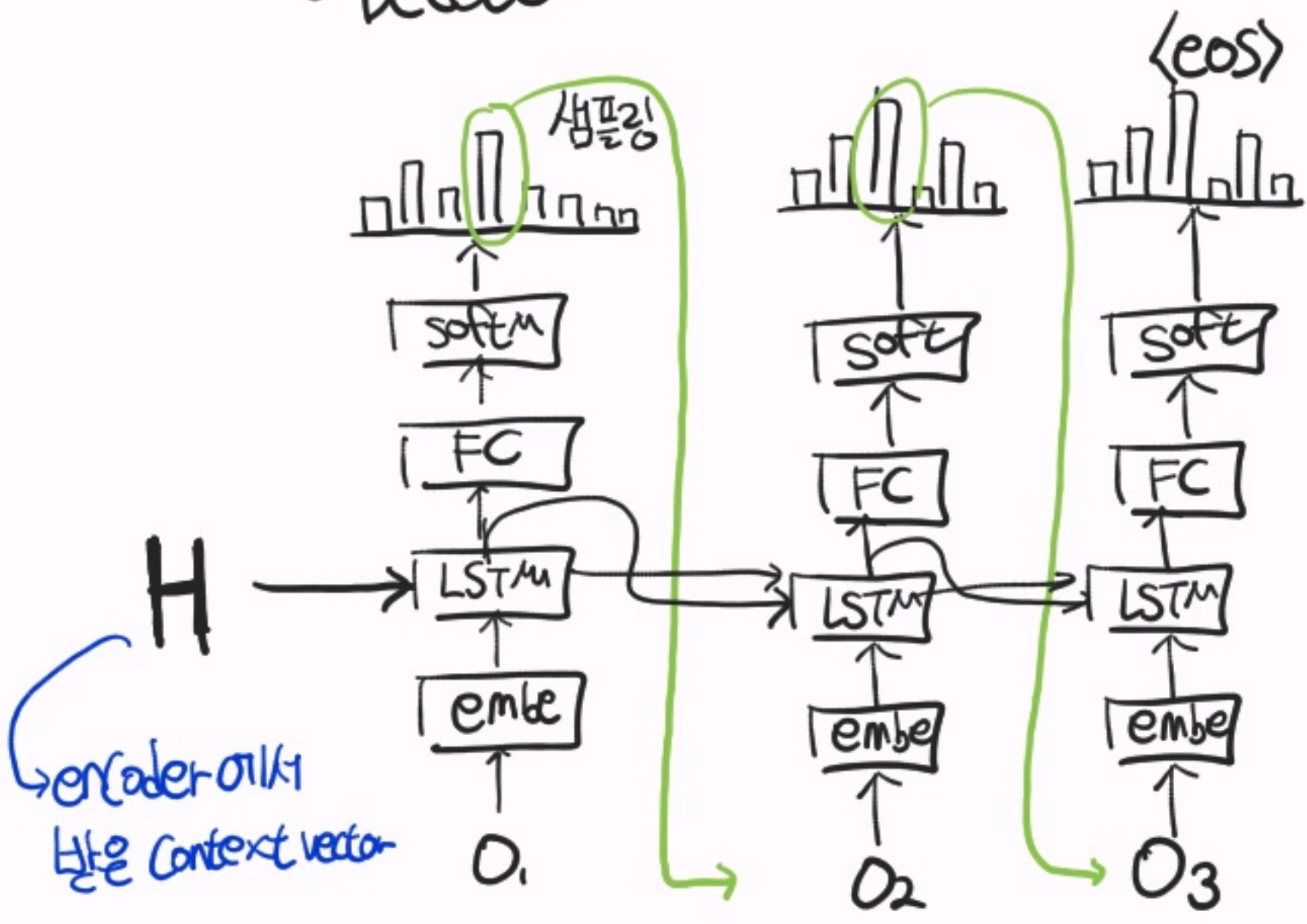
• 구조

• ENCoder



• 결국, 인코더의 임무는 "임의의 길이인 sequence를 고정 길이의 벡터로 표현"하는 것

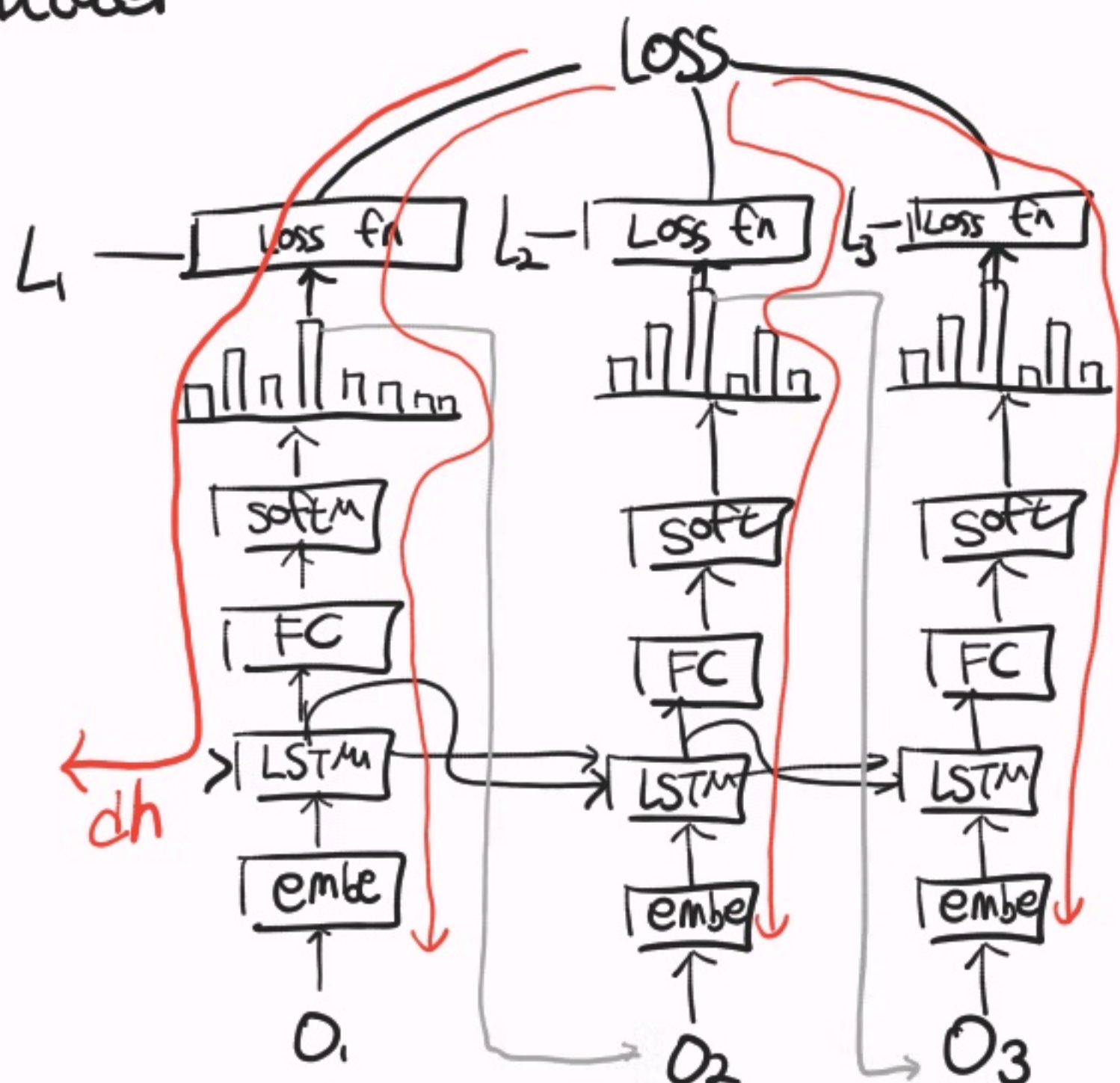
• DeCoder



- 기존 LSTM 모델과 동일
- 단 첫 LSTM은 H를 encoder에서 전달한 Context Vector가 됨.

Seq2Seq 역전파.

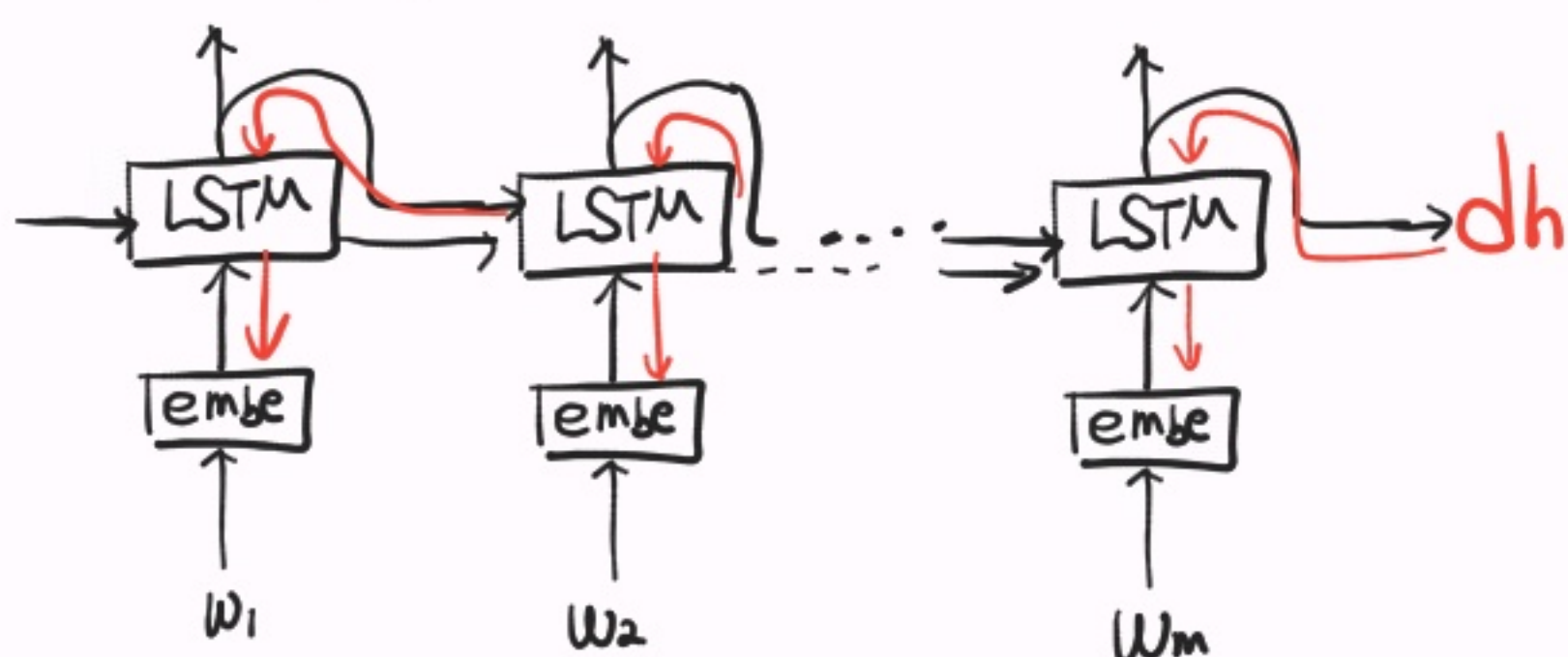
Decoder



- 순전파를 통해 출력 sequence를 생성하고 Loss값을 계산해 역으로 가중치들의 기울기를 구한다.
- 물론 encoder도 Context vector를 전달 받은 경로를 H에 대한 기울기 ds가 주어진다.

Encoder

Encoder



- 디코더에게 받은 H의 기울기를 받아 역전파를 진행한다.

Seq2Seq 개선

1. 입력 데이터 반전.

ex) 57+5 \Rightarrow 5+75
I am JSY \Rightarrow YSJ ma I

- 입력 데이터를 반전시킬 경우 학습 진행이 빨라지고 결과적으로 정확도가 높아짐.
- 이유로는 기울기 전파가 원활해지기 때문?
- 나는 이게 직관적으로 이해가 안감...ㅠ

2. 엿 보기 (Peeky)

- 인코더에서 출력한 Context 벡터 H는 Decoder가 필요로 하는 중요한 정보가 담겨 있다.
- 이 정보를 Decoder의 첫 번째 LSTM만 받는 것이 아니라
- 모든 계층의 LSTM의 입력으로 전달해 주고 뿐만 아니라 FC layer의 입력으로도 넣어준다.

↳ 집단 지성의 비유 할 수 있겠다.