DATA
SCIENCE LAB

RNN

23.02.23 / 8기 김지희

0. Intro

RNN	NLP가 무엇인지 / Text Data / Vanilla RNN / LSTM / GRU
Transformer1	Seq2Seq / Encoder & Decoder / Attention / Transformer
Transformer2	Transformer 구현

CONTENTS

1. NLP

What is Natural Language Processing?

자연어 처리

: 컴퓨터가 인간의 언어를 이해, 생성, 조작할 수 있도록 해주는 인공지능의 한 분야

EX) 번역, 챗봇, 대본 작성, 연관검색어…

1. NLP

Why NLP is difficult? 언어 = 사회적 약속

오직 인간만을 위한 system임. 너랑 나만 이해하면 상관없지 뭐

=> 컴퓨터가 이해하기 어려움

[국어 수업 때 많이 본 언어의 특징]

중의성

차를 마시러 공원에 가던 차 안에서 나는 그녀에게 차였어

선생님은 울면서 돌아오는 우리를 위로했다.

다양성

'춥다'의 유의어: 쌀쌀하다, 으슬으슬하다, 차갑다,…

1. NLP

Representation the meaning of a word

표시론적 의미론: 의미란 이 단어를 듣고 떠오르는 모든 것이야! 나무 = [🌲,🌳,🌴,…]

Q) 그럼 이걸 어떻게 컴퓨터가 이해하게 표현해?

1. Wordnet (유의어 사전 이용하기)

- 의미 : 유의어와 상위어를 가진 거대한 전자사전
- 단점 : 단어 자체의 뉘앙스 놓침 / 모든 의미 표현 불가 / 주관적 / 노동력 多 / 정확한 유사도 계산 불가

synonym sets(동의어): good = goodness, commodity

hypernyms: panda < mammal < animal

1. NLP

Representation the meaning of a word

2. One-Hot Vector

- 의미 : 단어의 개수가 곧 차원으로 해서 단어를 vector로 표현하자
- 단점
  - 500,000개 단어 표현 -> 500,000 dimension 필요 => 비효율적
  - 각 vector가 orthogonal => 단어간 similarity 표현 불가 (그럼 의미는…?)

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

1. NLP

Representation the meaning of a word

3. Distributional Semantics

벡터 자체에 단어 간 유사도를 encode함

분포 의미론: 단어의 의미는 문맥을 통해 파악하는 거야!

⇒ 주변에 오는 단어들로 단어가 정의된다

Example:

The kid said he would grow up to be superman.

The child said he would grow up to be superman.

고정된 크기의 window로 주위단어 확인

⇒ 단어 주변이 비슷하면 비슷한 단어이고 유사한 벡터를 가질 것

1. NLP

Representation the meaning of a word

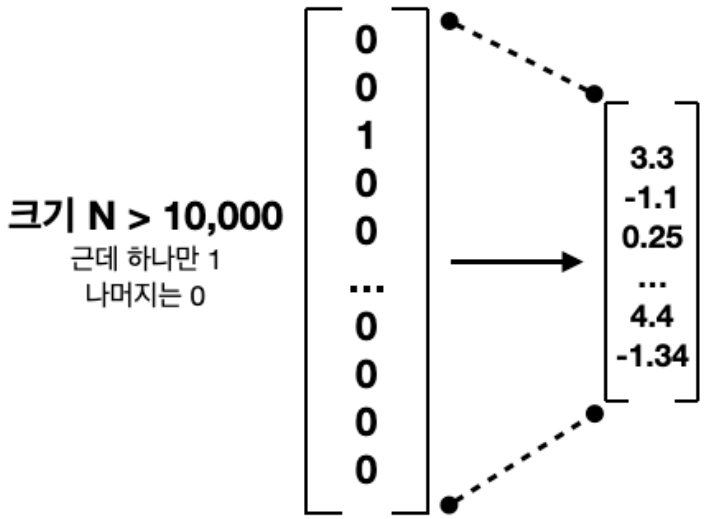
Word Embedding (Word Vector)

Embed : “끼워넣다 "

: 단어를 one-hot vector로 표현하는 게 아니라 차원을 낮춰서 정보를 담자

- 훈련 데이터로부터 학습 필요

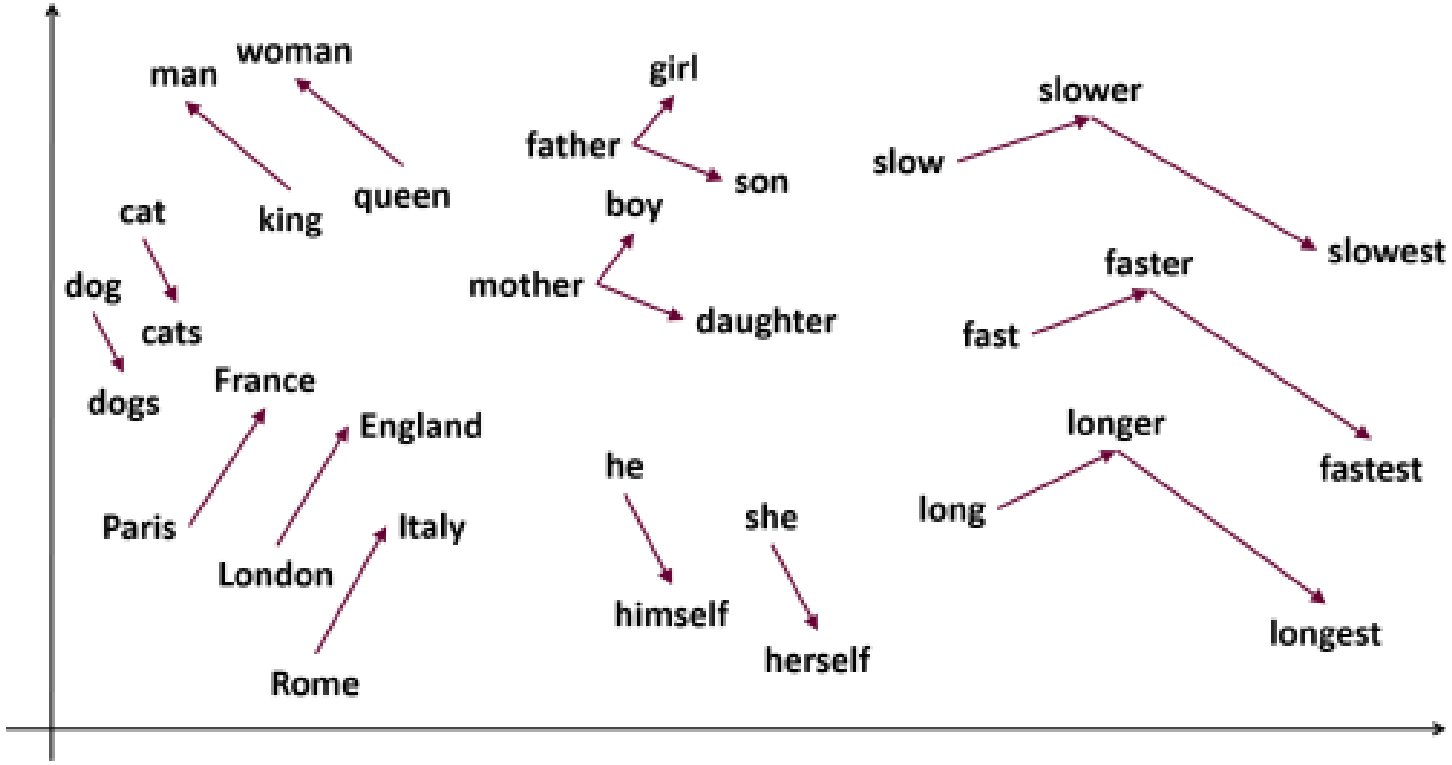
Sparse Representation



Dense Representation

1. NLP

Representation the meaning of a word



word벡터로 표현되면

King - Man + Woman = Queen

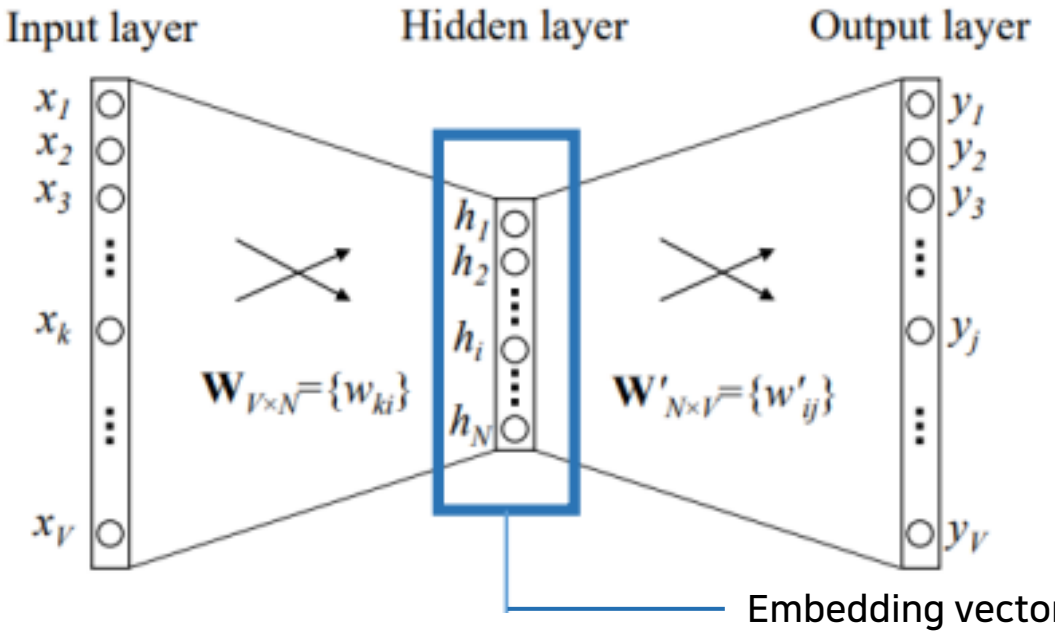
와 같은 연산도 가능!

1. NLP

Word2Vec

Word vector를 NN을 사용해 학습하는 방법

단어 벡터 간 **유의미한 유사도**를 이용해 맥락에서 특정단어 나타날 확률 계산해 단어의 의미를 **수치화**



1. NLP

Word2Vec

"The fat cat sat on the mat"

Window = 2

[CBoW]

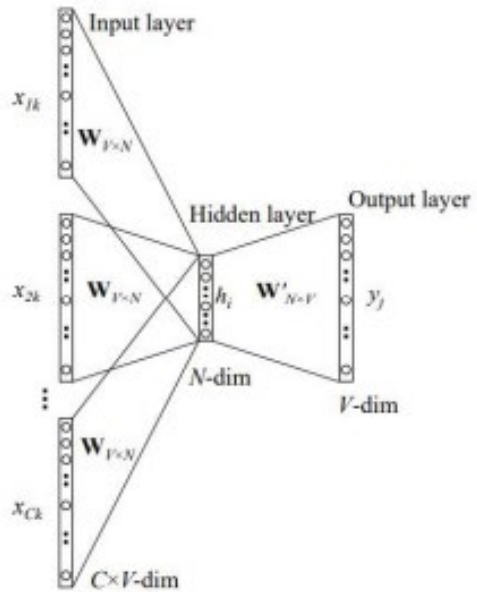


Figure 2: Continuous bag-of-word model

Fat cat ___ on the => sat 예측

[SkipGram]

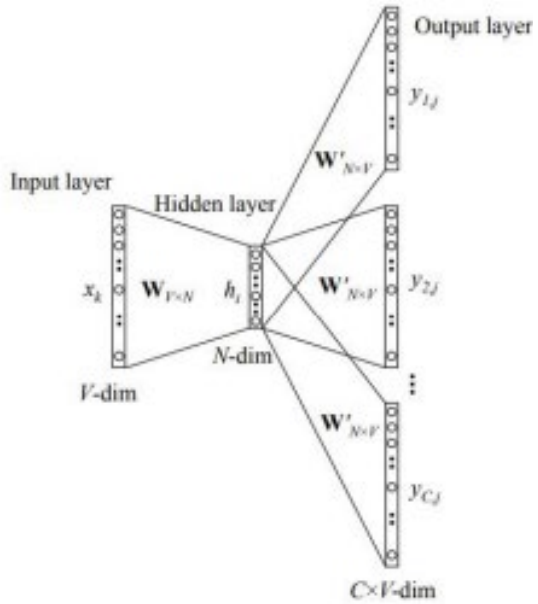


Figure 3: The skip-gram model.

__ __ sat __ __ => cat, on 예측

1. NLP

Word2Vec

[CBoW로 작동원리 익히기]

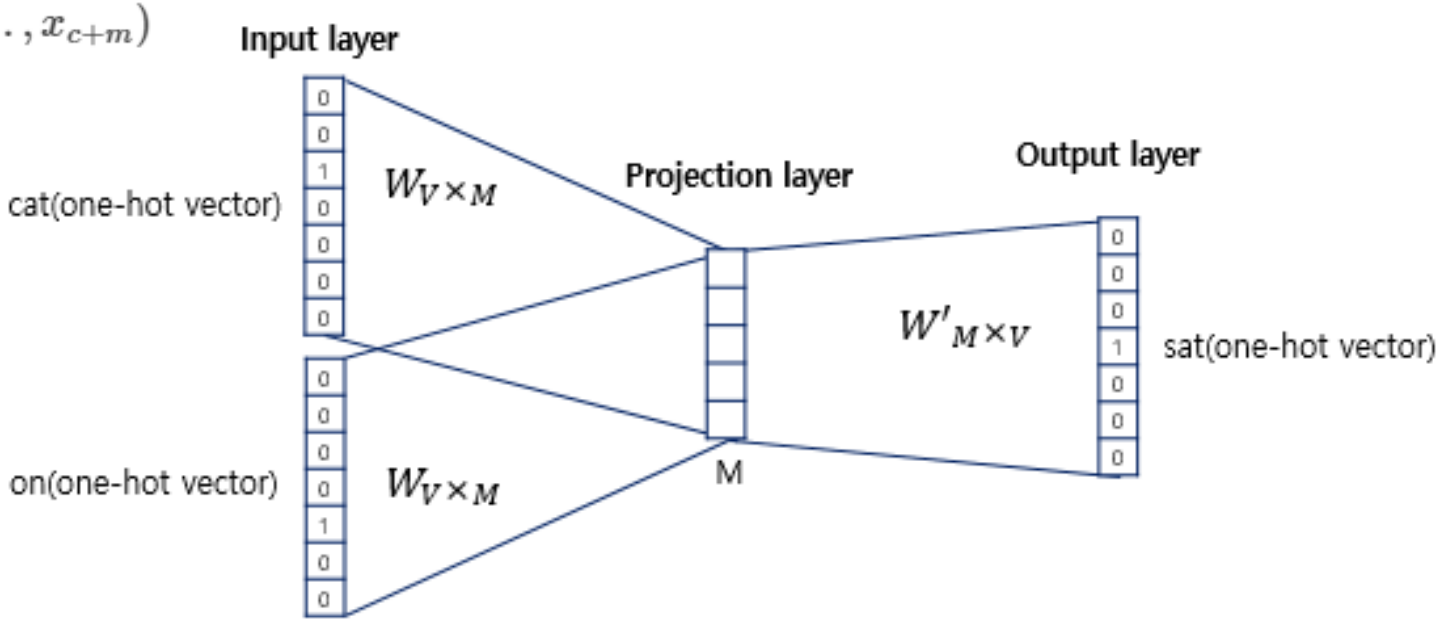
Goal : maximize $P(x_c|x_{c-m},\dots x_{c-1},x_{c+1},\dots,x_{c+m})$

1. One-Hot encoding

the : [1,0,0,0,0,0], fat : [0,1,0,0,0,0]

cat : [0,0,1,0,0,0], sat : [0,0,0,1,0,0]

on : [0,0,0,0,1,0],mat : [0,0,0,0,0,1]



1. NLP

Word2Vec

[CBoW로 작동원리 익히기]

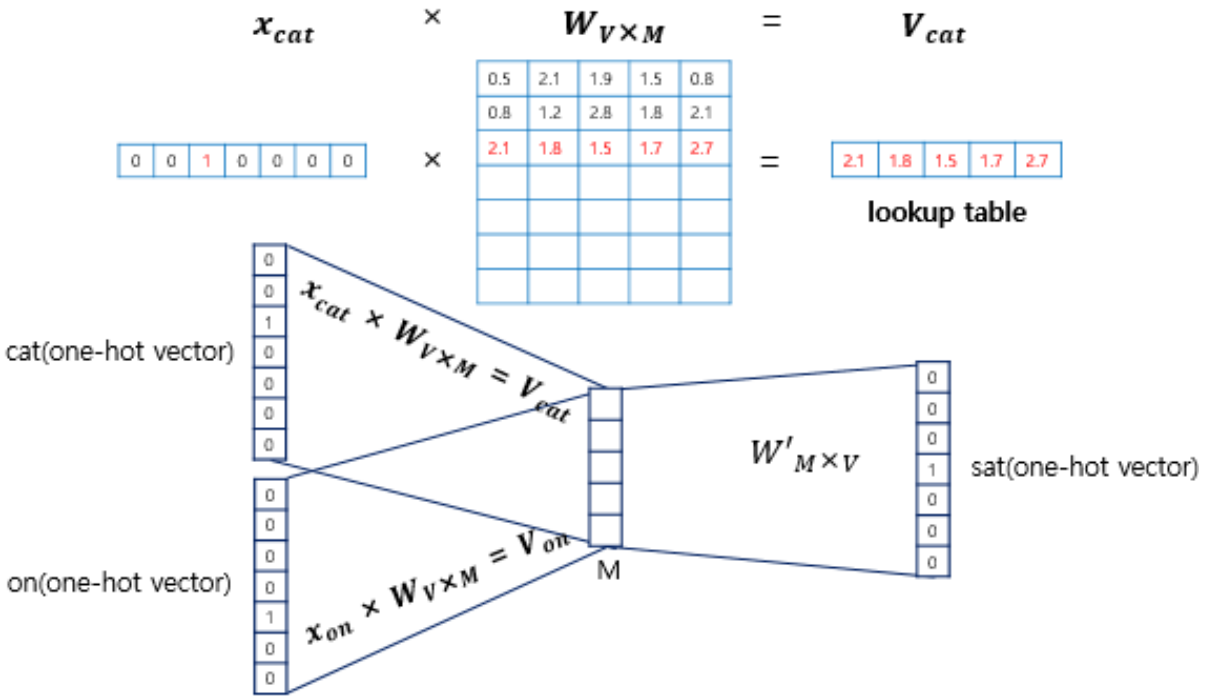
2. Input layer -> Hidden layer

One Hot Encoding 한 vector와 Embedding matrix를

곱해, N 차원의 embedding vector를 얻는다.

Cat: [0,0,1,0,0,0,0] @ embedding matrix

=> Cat의 embedding vector



1. NLP

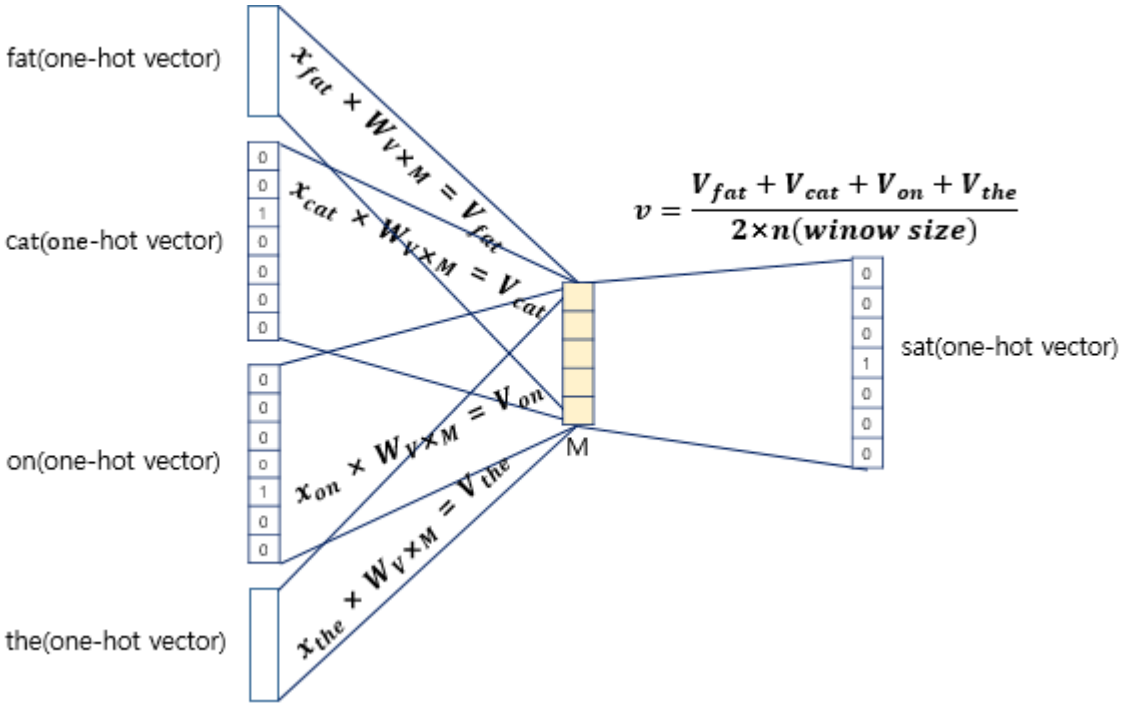
Word2Vec

[CBoW로 작동원리 익히기]

2. Input layer -> Hidden layer

모든 주변 단어의 embedded vector를 구하고 평균 내기

$$v = \frac{V_{fat} + V_{cat} + V_{on} + V_{the}}{2 \times n(winow\ size)}$$



1. NLP

Word2Vec

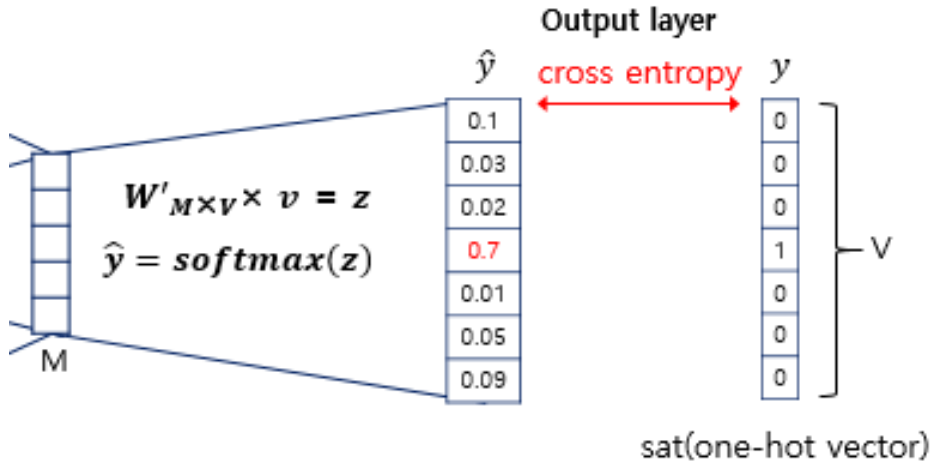
[CBoW로 작동원리 익히기]

2. Hidden layer -> Output layer

구한 평균 vector를 score matrix W'와 곱하기

[0.1, 0.03,0.02,0.7,0.01,0.05,0.09] @ Score matrix

=> 결과로 나온 vector는 input vector와 동일 차원



1. NLP

Word2Vec

[CBoW로 작동원리 익히기]

2. Hidden layer -> Output layer

Score vector를 확률로 나타내기 위해 Softmax 함수 사용

⇒ 이 결과는 center word 'sat'의 one-hot vector와 가까워져야 함

[loss function] : cross entropy

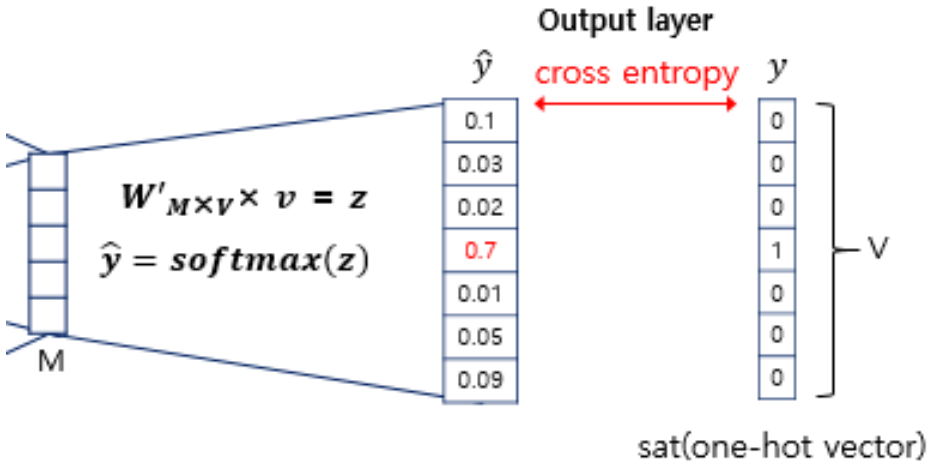
$$cost(\hat{y}, y) = -\sum_{j=1}^{V} y_j \, log(\hat{y}_j)$$

이후엔 backpropagation 진행하면서,

Embedding matrix, score matrix update 진행

[Softmax function]

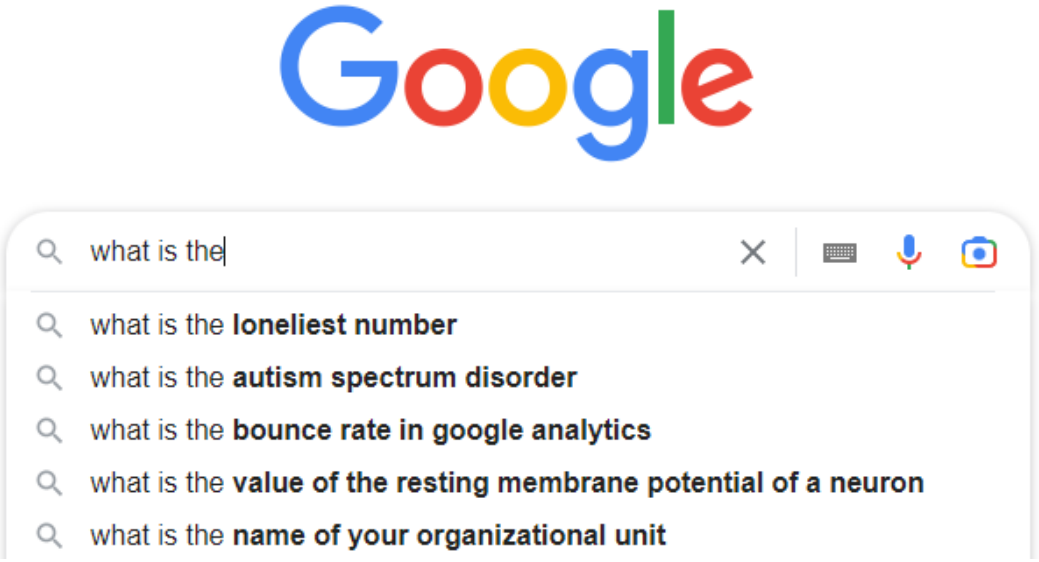
$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



1. NLP

Language Modeling NLP의 가장 기본이 되는 Task

: 현재까지 주어진 단어들의 조합 뒤에 올 단어 예측하기

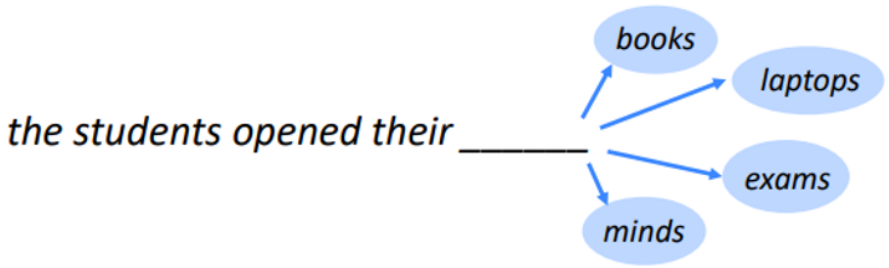


1. NLP

Language Modeling NLP의 가장 기본이 되는 Task

: 현재까지 주어진 단어들의 조합 뒤에 올 단어 예측하기

= 문장의 단어들이 주어졌을 때, 다음의 올 단어 확률 구하기



P(books | the, students, opened, their)

P(laptops | the, students, opened, their)

P(exams | the, students, opened, their)

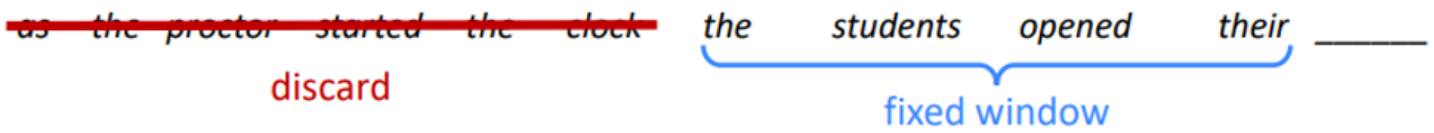
P(minds | the, students, opened, their)

1. NLP

A fixed-Window NN Model

예측할 단어 이전에 window를 고정해서 그 다음에 올 단어의 확률을 예측하는 Model

Window를 고정하는 이유:
- MLP은 input size와 output size가 고정되어 있어서



1. NLP

A fixed-Window NN Model

Input : the student opened their

Task : 문장에 이어질 단어 예측

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

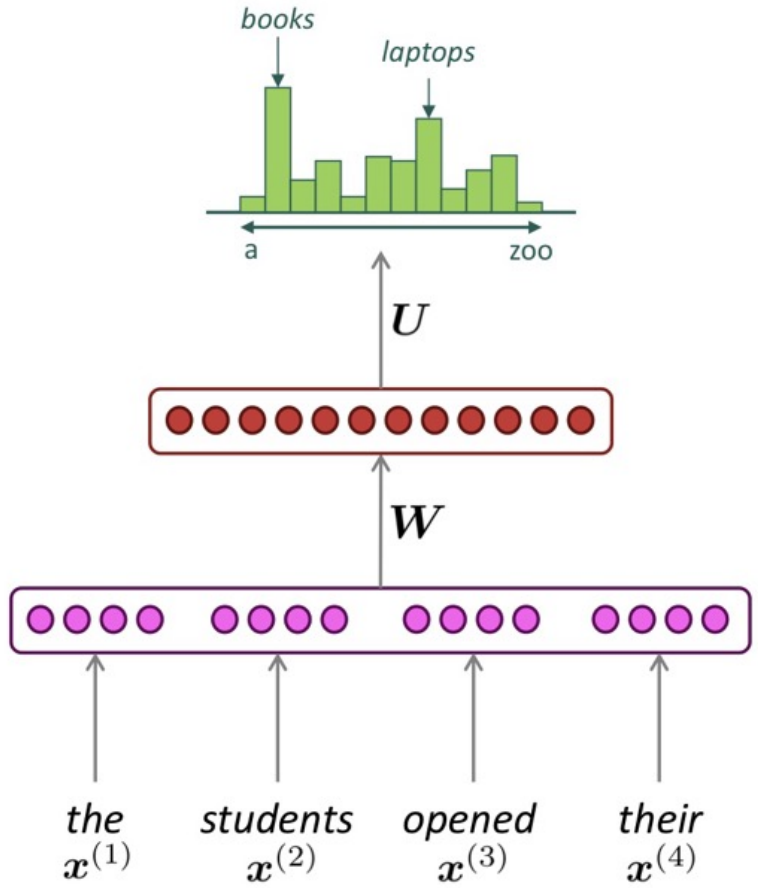
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



1. NLP

A fixed-Window NN Model

단점

- 작은 fixed window
- Window size 키우면 weight matrix가 커지는 문제
- 단어의 위치에 따라 곱해지는 weight 다름

  => 비슷한 내용에 대해서 여러 번 학습 진행해야 함

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

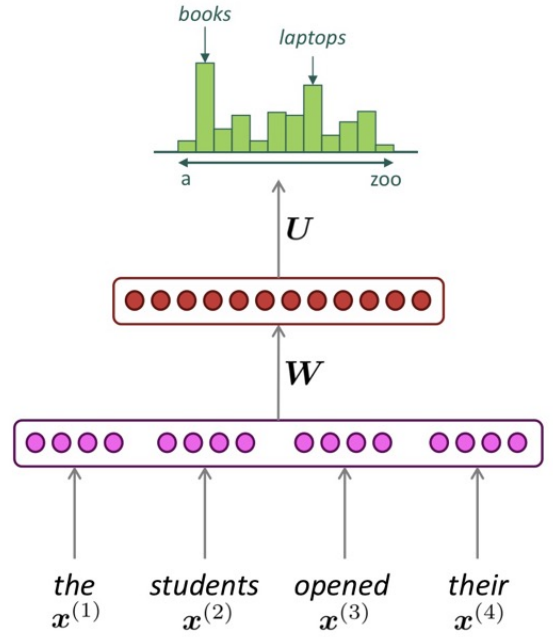
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



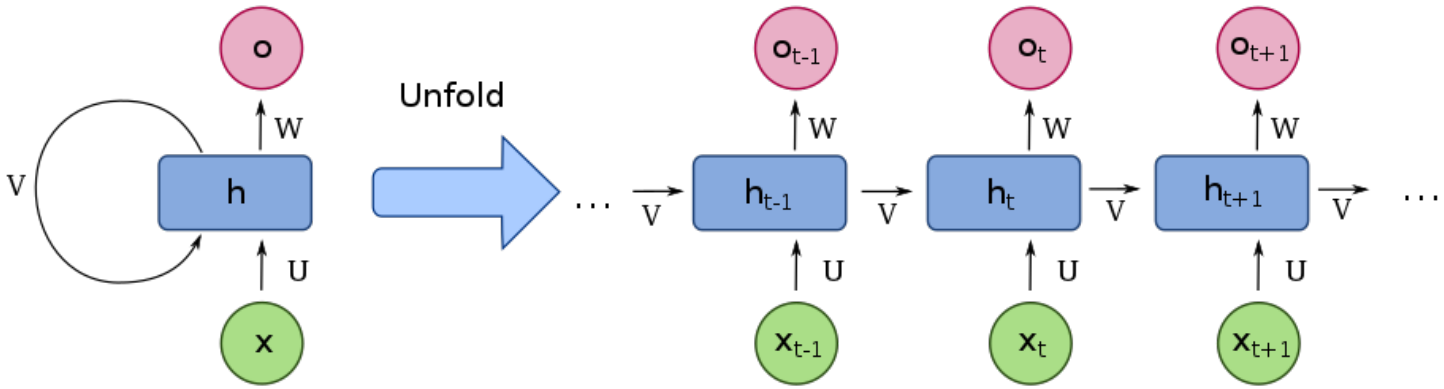
2. RNN

Recurrent Neural Networks

= 반복하다, 되풀이 되다

= 은닉층에서 나온 결과가 다시 은닉층으로 돌아가서 새로운 input과 연산을 수행함

= hidden state가 연속적으로 이어지면서 정보 저장



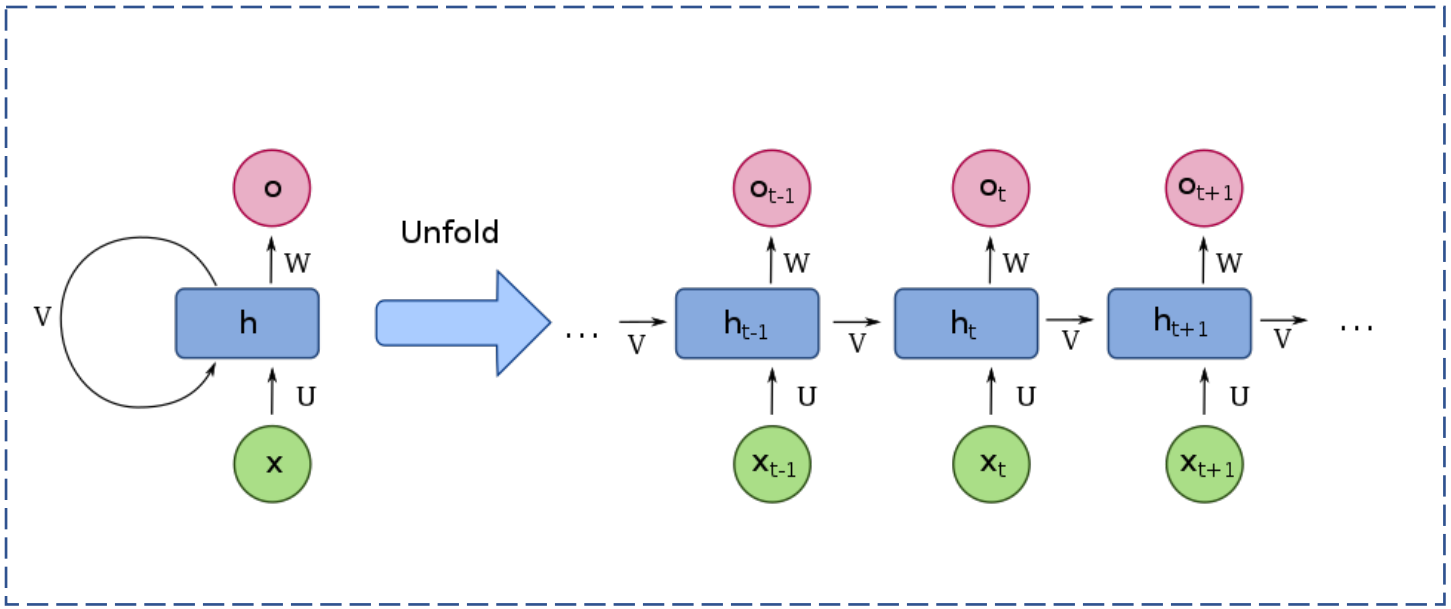
CNN에서 이미지 구역별로 같은 weight 공유하는 것처럼

RNN은 시간별로 같은 weight 공유

= 과거와 현재 같은 weight 공유

2. RNN

Structure of RNN



x_t : time step t의 input

h_t : time step t의 hidden state

(lossy summary)

$$\underbrace{\boldsymbol{h}^{(t)}}_{\text{new state}} = f_\theta(\underbrace{\boldsymbol{h}^{(t-1)}}_{\text{old state}}, \underbrace{\boldsymbol{x}^{(t)}}_{\text{input}})$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{W}_{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{hx}\boldsymbol{x}^{(t)} + \boldsymbol{b}_h)$$

o_t : time step t의 output

2. RNN

Structure of RNN

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

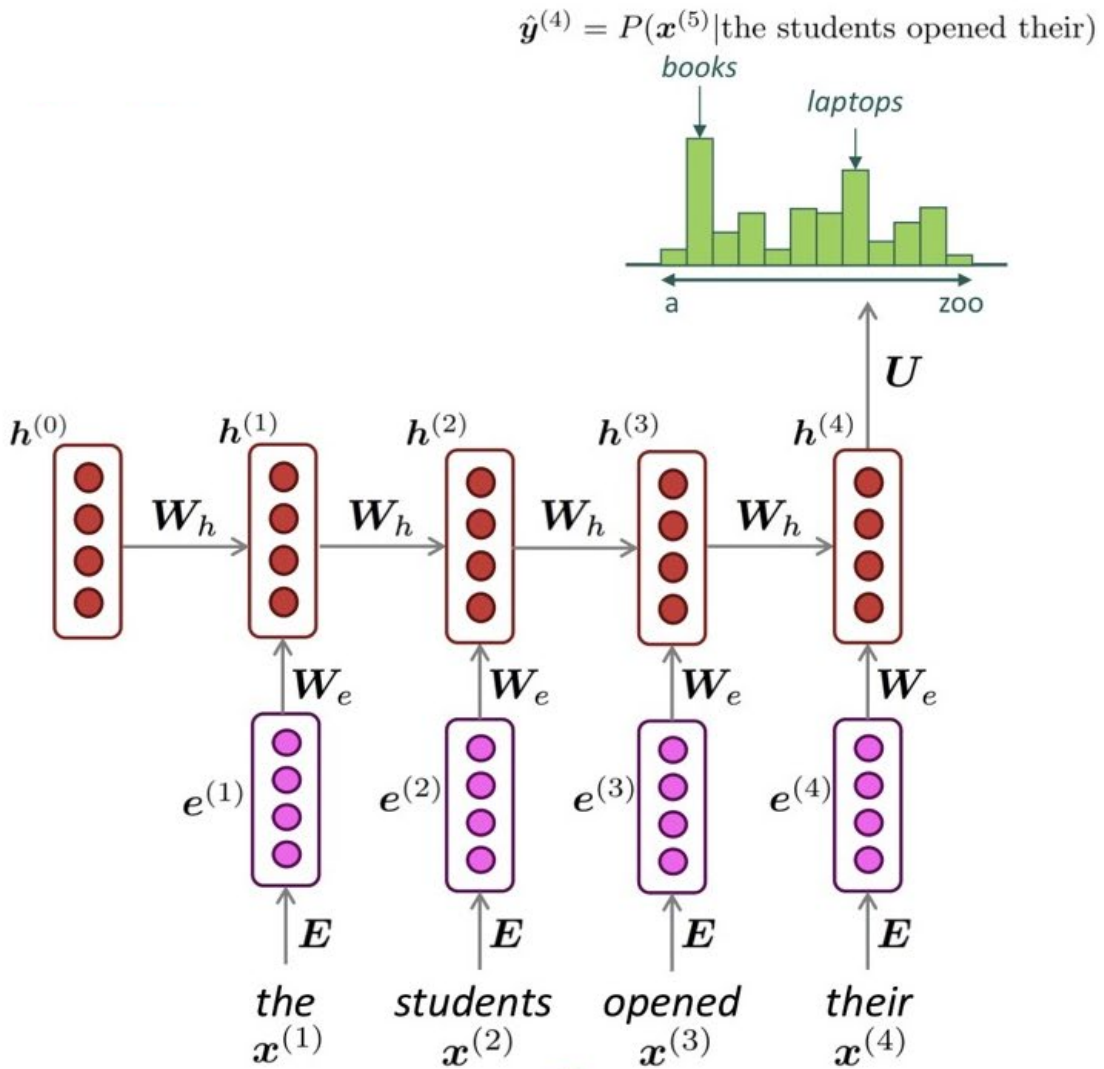
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

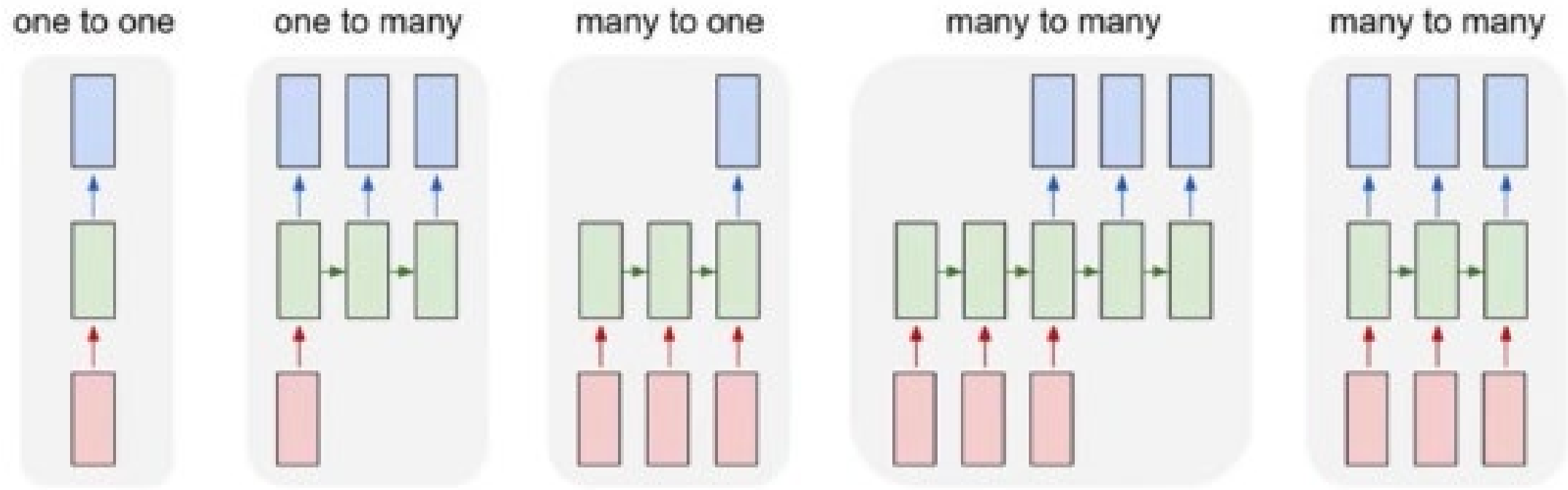


Note: this input sequence could be much longer now!

2. RNN

Types of Task Dealing with Sequential Data

순차적인 input의 길이, 순차적인 output의 길이에 따라 다음과 같이 구분됨

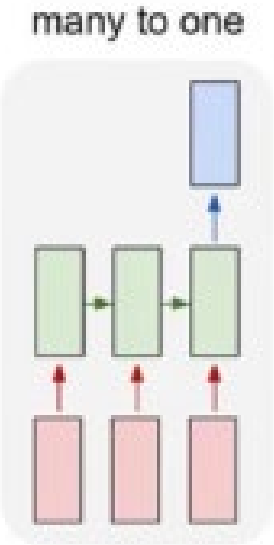


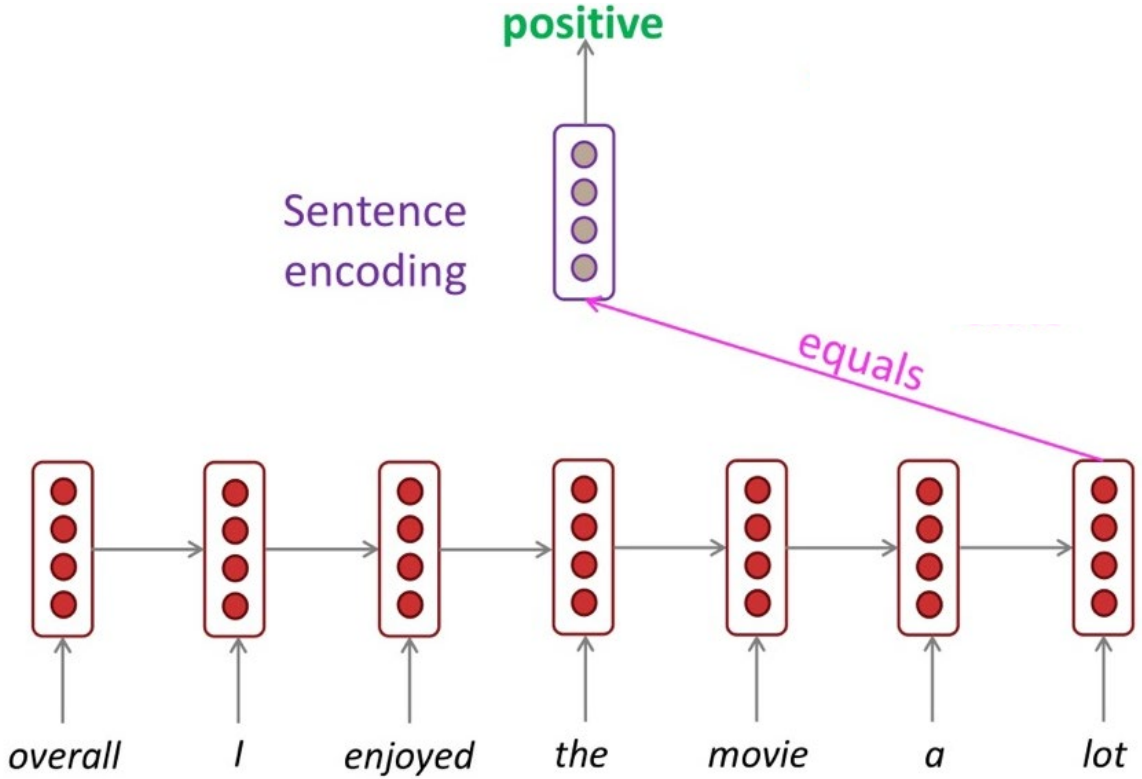
2. RNN

Many to One

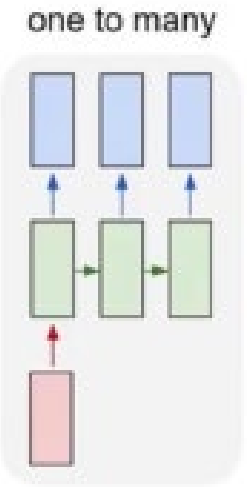
순차적인 X로 한 시점의 Y를 예측함

EX) sentiment Classification





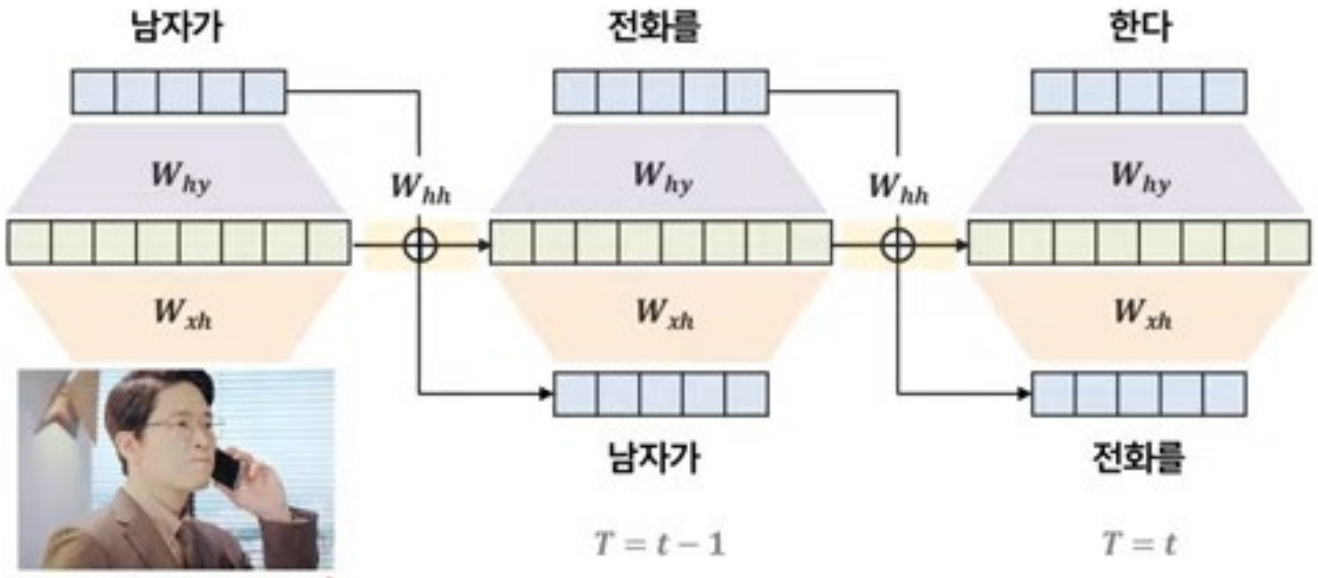
2. RNN



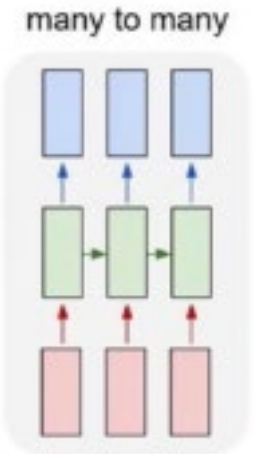
One to Many

여러 시점 X로 하나의 Y를 예측함

EX) 이미지 데이터를 입력으로 받아, 이미지에 대한 설명 글로 생성하기



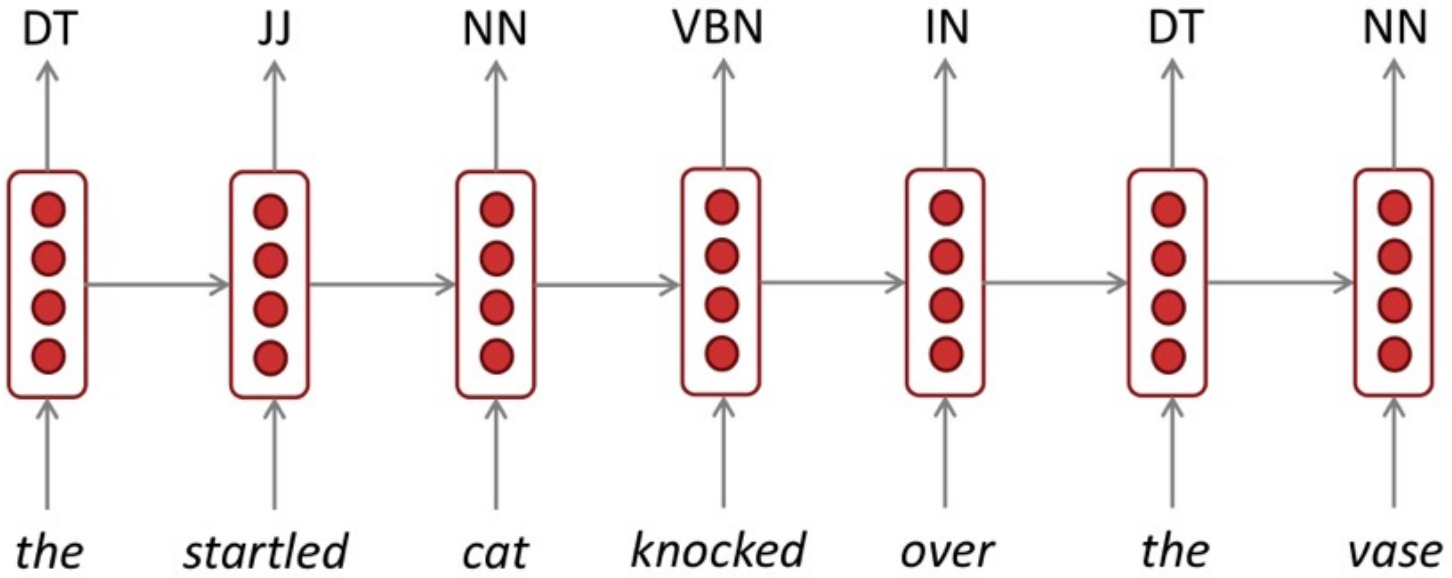
2. RNN



Many to Many

여러 시점 X로 여러 시점의 Y를 예측함

EX) 문장을 input으로 받았을 때, 각 단어의 품사 예측하는 POS Tagging

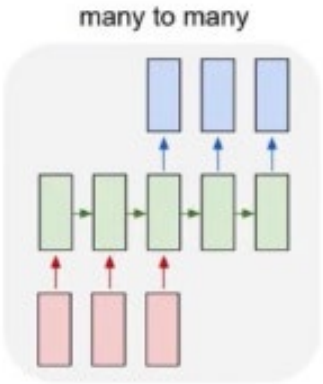


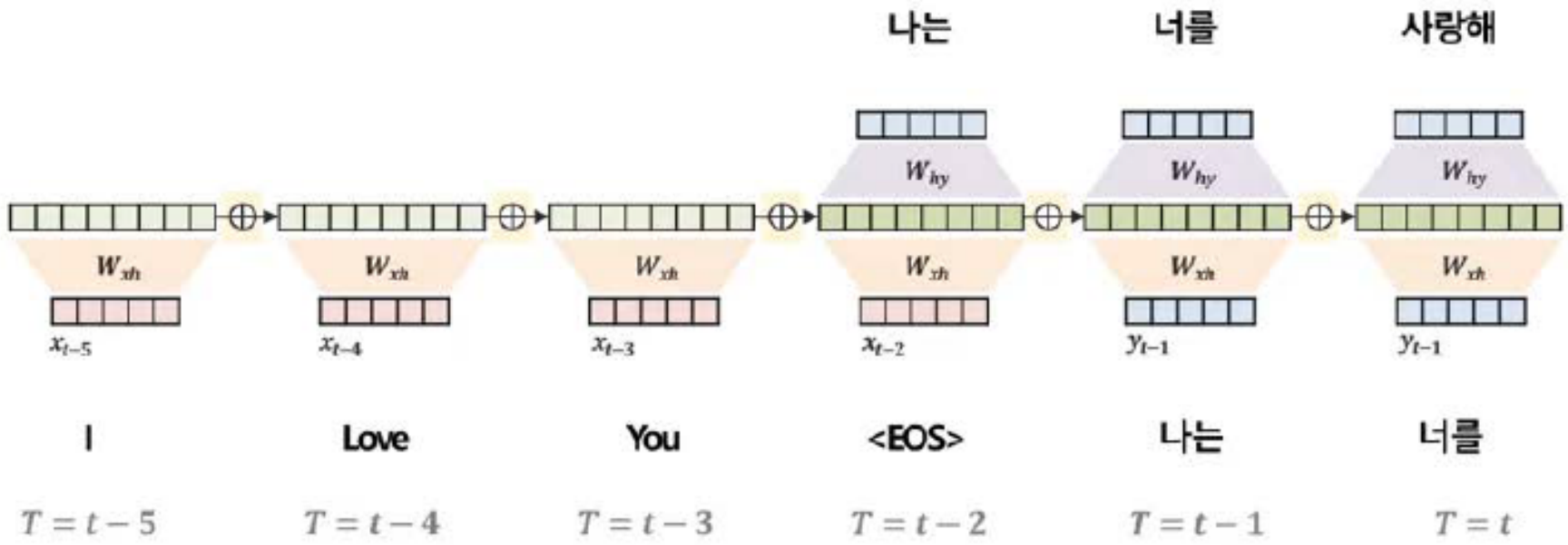
2. RNN

Many to Many

순차적인 X로 순차적인 Y를 예측함

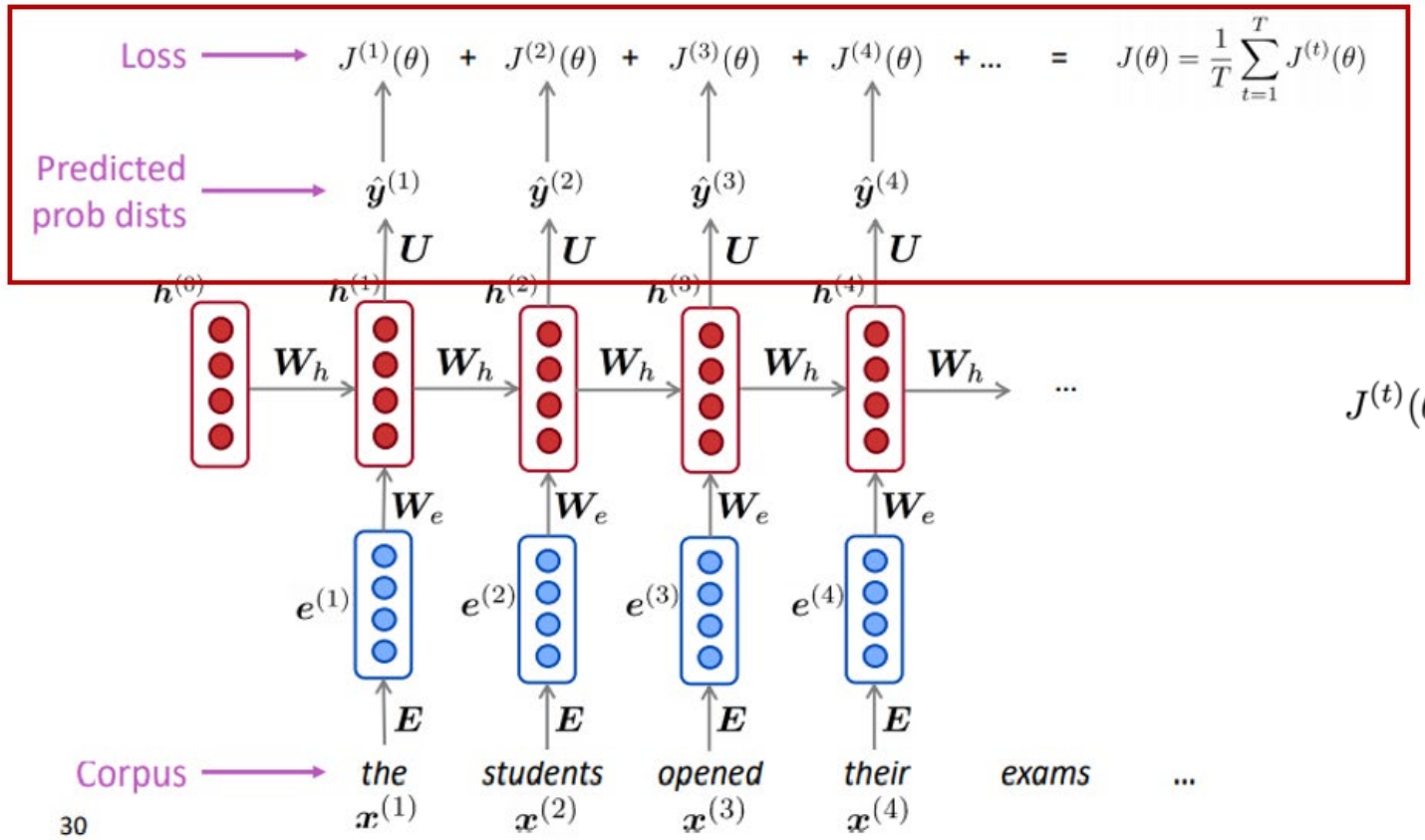
EX) 영어 문장이 주어졌을 때, 한글 문장으로 번역하기





2. RNN

Training RNN



30

다음 단어 예측하는 문제

- 모든 단계에서 예상되는 다음단어와 실제 다음 단어와의 차이를 구함 (using cross entropy)

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

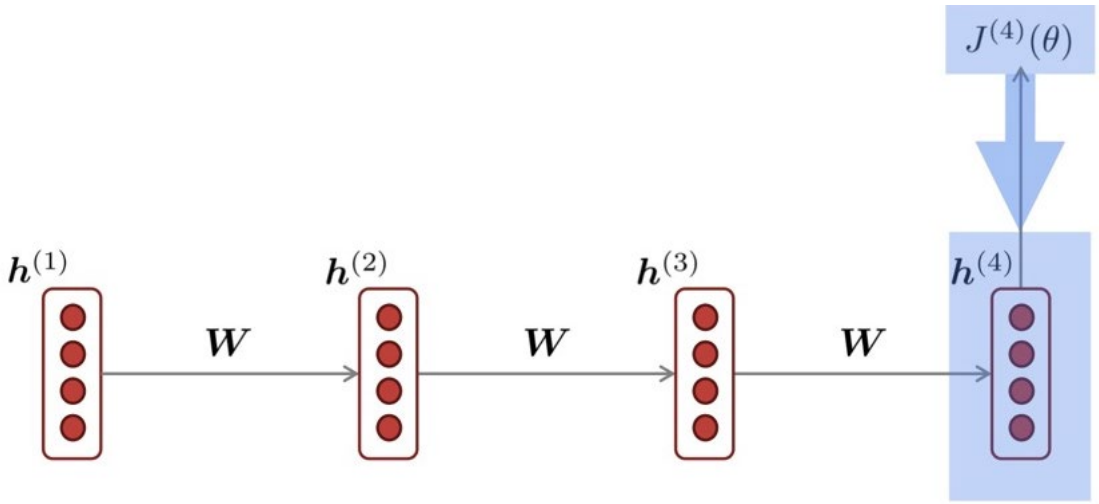
- 전체 loss = loss의 평균

2. RNN

Vanishing Gradient

다른 NN 모델처럼 구한 loss를 가지고 back propagation 진행해야 함.

But 시점마다 동일한 weight matrix가 사용되기에 chain이 길어짐



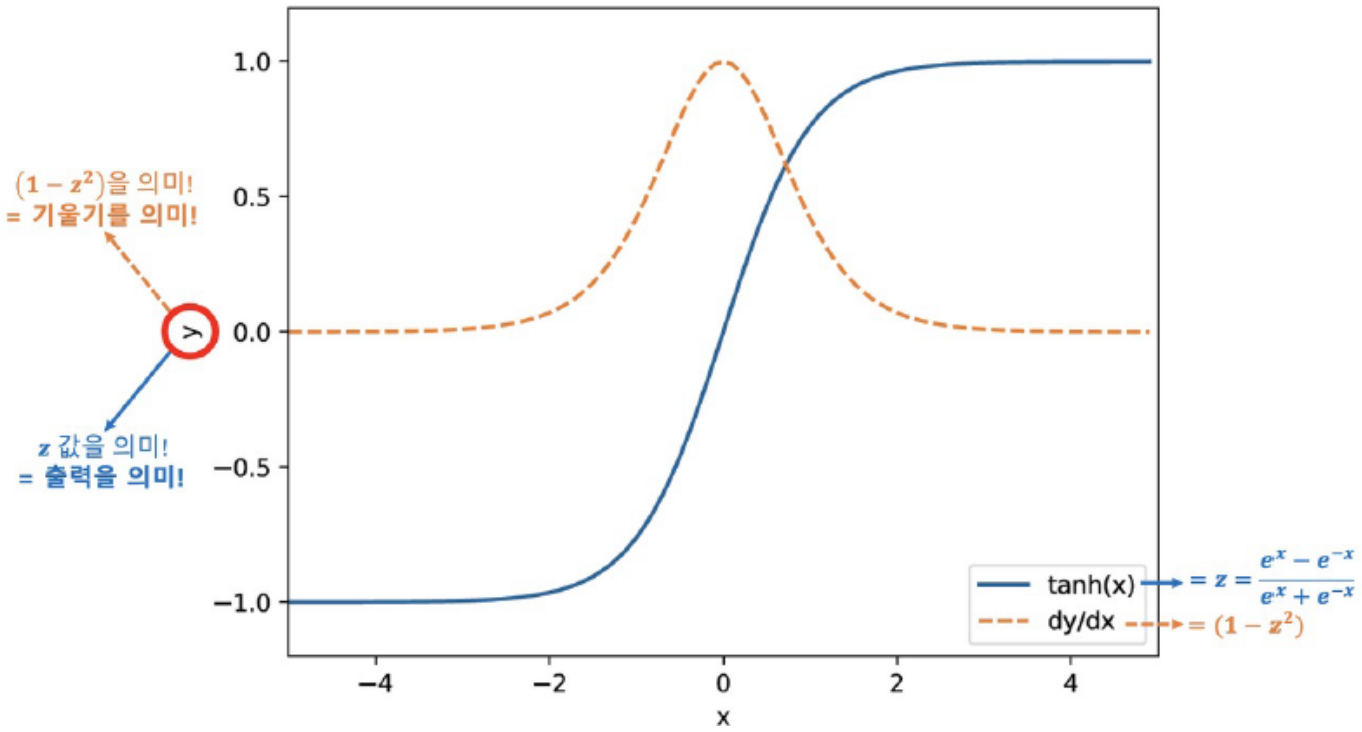
곱해지는 값들이 1보다 작으면
Gradient 빠른 속도로 0을 향해 감

=> Vanishing gradient

2. RNN

Vanishing Gradient

Activation function의 종류에 따라서도 vanishing gradient 발생 가능

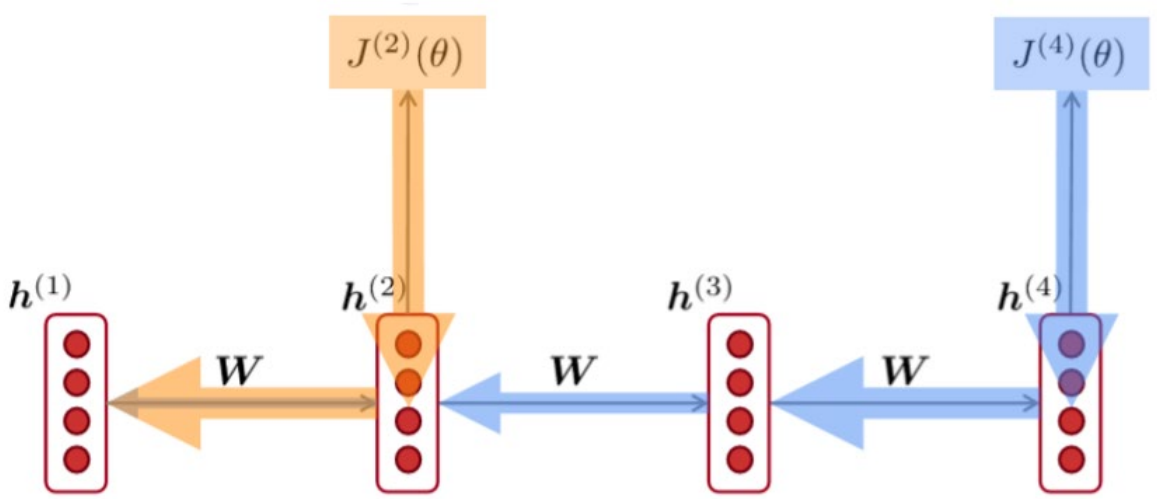


2. RNN

Vanishing Gradient

Vanishing Gradient가 문제인 이유:

멀리서 온 gradient는 소멸해버리고, 근처에서 온 gradient는 살아남아 있어서
가까운 곳에서 온 effect에 의존해 update가 진행됨



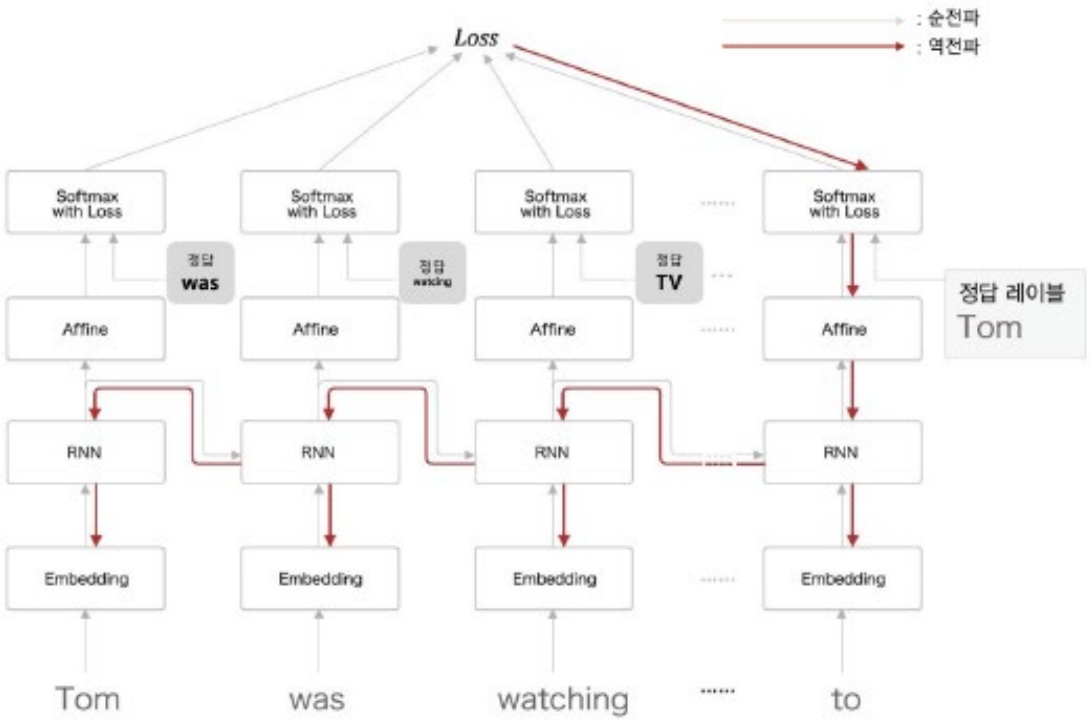
2. RNN

Vanishing Gradient

Vanishing Gradient가 문제인 이유:

그림 6-3 "?"에 들어갈 단어는?: (어느 정도의) 장기 기억이 필요한 문제의 예

Tom was watching TV in his room. Mary came into the room. Mary said hi to ?



3. LSTM

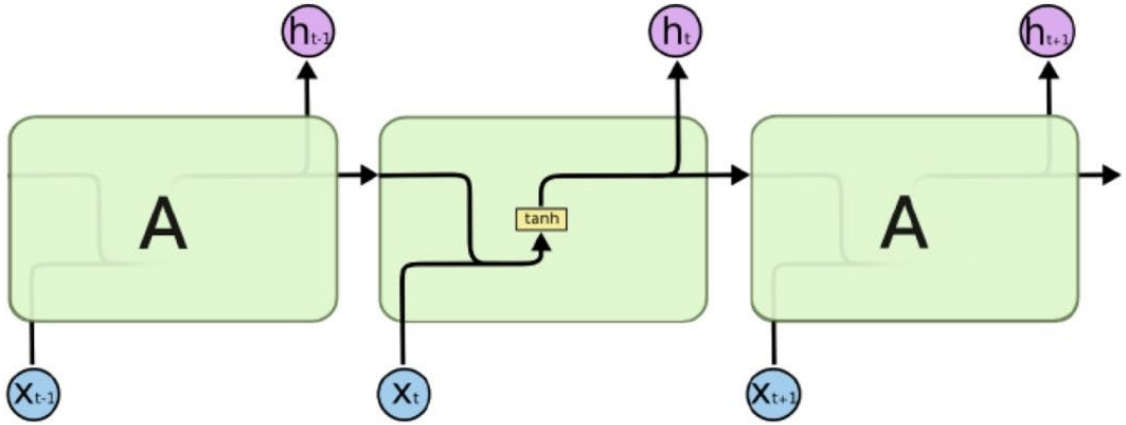
Long-Short Term Memory

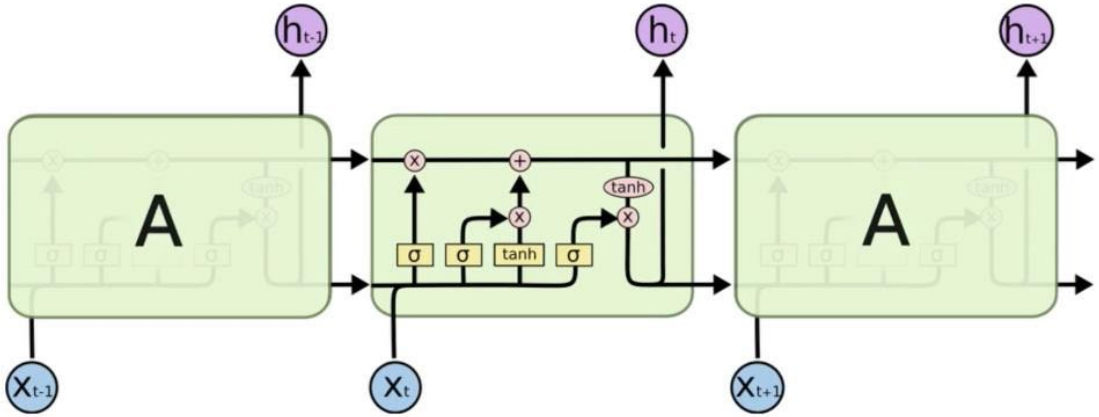
RNN의 문제점 : non linear function이 수행된 hidden state가 매시점마다 재사용됨

⇒ 기울기 소실 & 폭주 문제 발생

Idea : 장기기억용 flow를 새로 만들고 그때그때 필요한 정보는 여기서 가공해 내보내자!

:cell state





3. LSTM

Long-Short Term Memory

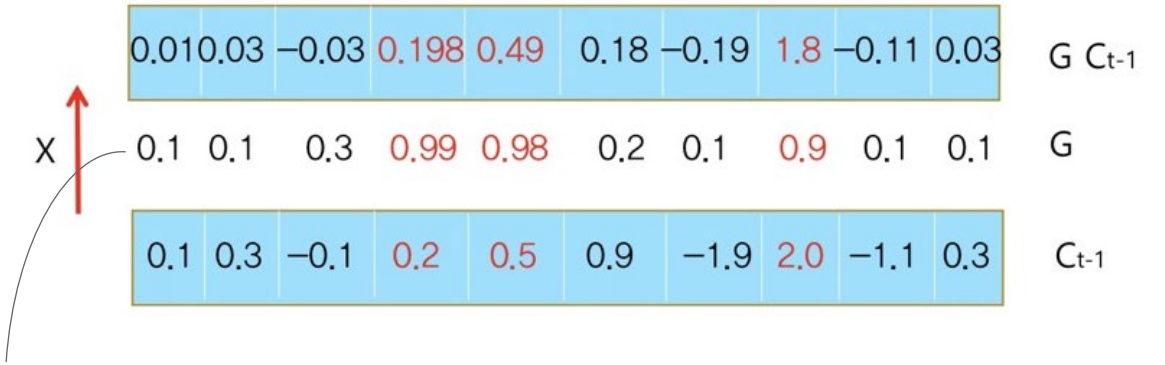
- 이전 단계의 정보 중 일부분은 잊고, 현재 정보 중 일부분을 기억해 더해서 다음 시점으로 전달
- hidden state는 cell state를 적당히 가공해서 내보내기

How? Gate 이용하기

3. LSTM

gate Elementwise coefficient multiplication

각 차원의 정보에 0~1 사이의 계수를 곱하여 정보를 통과시킬지 막을지 조절함



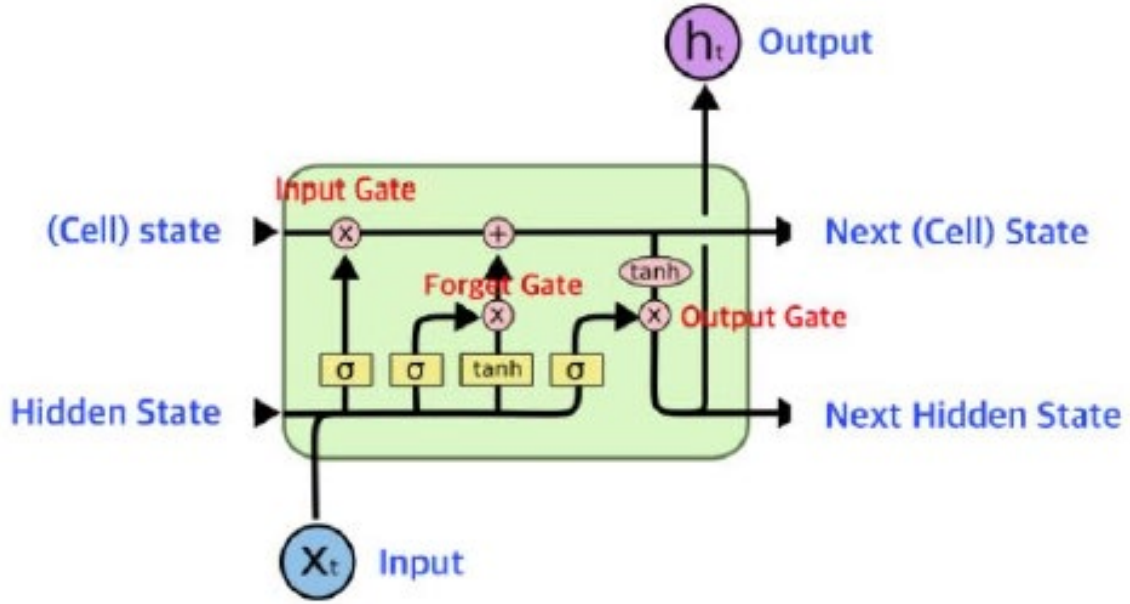
Gate의 각 계수 값만큼 정보 남겨짐

3. LSTM

LSTM의 구조

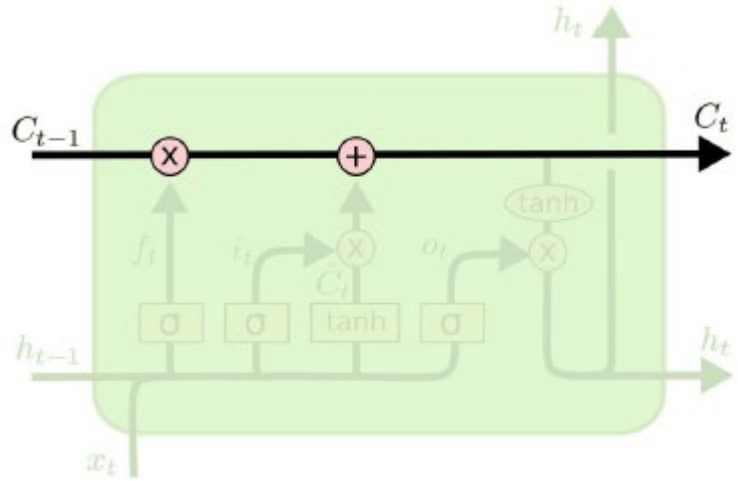
** Gate는 전부 이전시점까지의 정보(hidden state)와 현재 input을 가지고 학습됨

- Forget Gate : 과거의 기억을 남길 비율 조정 (erase)
- Input Gate : 새로운 기억을 추가하는 비율 조정 (write)
- Output Gate : cell state를 출력에 반영하는 비율 조정 (read)



3. LSTM

Cell state

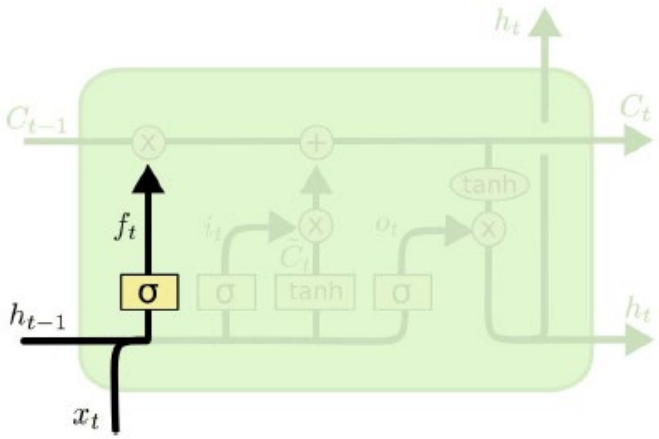


전체적으로 중요한 정보가 흘러가는 부분

- Nonlinear function이 추가로 곱해지지 X 단순 연산으로 통과됨
- 이전 정보가 큰 변화없이 다음 step으로 넘어감
- Gate에 의해 정보가 추가/제거/읽혀져 계속 흘러감

3. LSTM

Forget Gate



** gate는 이전 시점까지의 정보와 새 input을 가지고 학습됨

Cell state에서 불필요한 정보를 지움

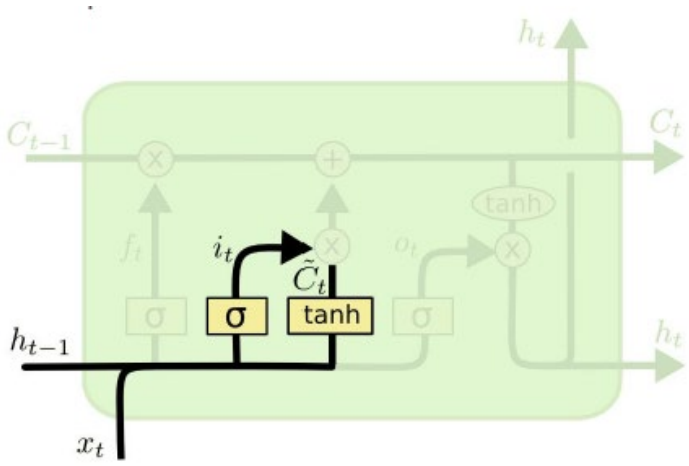
Sigmoid function을 통해 0~1 사이 값으로 표현됨

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

- f_t == 1: 모든 정보를 보존해라!
- f_t == 0: 죄다 갖다 버려라!

3. LSTM

Input Gate



이전 cell state에 새로운 정보를 추가할 비율 조정함

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

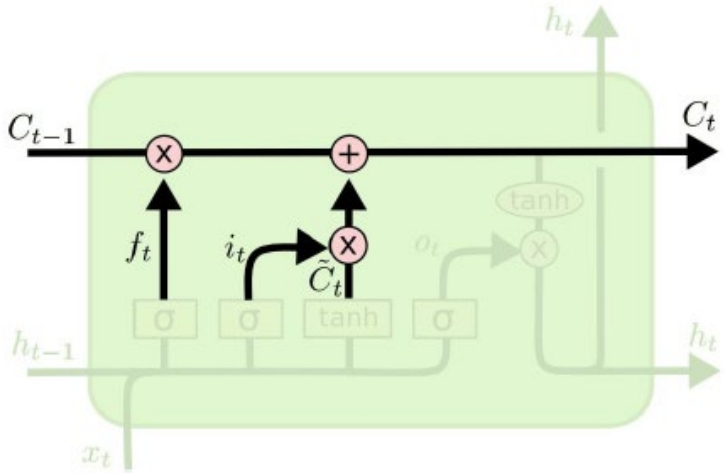
- i_t == 1: 모든 정보 다 중요해
- i_t == 0: 죄다 갖다 버려라!

Cell state에 더해질 vector 만들기

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

3. LSTM

Update



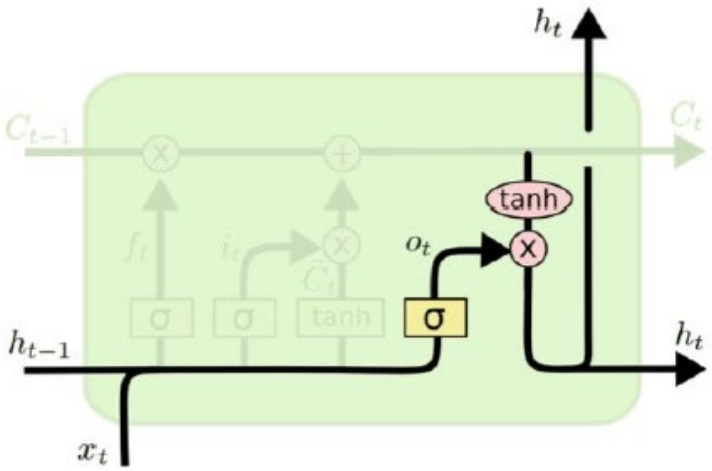
f_t 만큼 이전 cell state 잊어버리고, i_t만큼 새 정보 추가하자

둘을 합해서 cell state update

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

3. LSTM

Output Gate



각 시점마다 출력값 내보낼 때 cell state에서 출력에 반영할 비율 조정

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

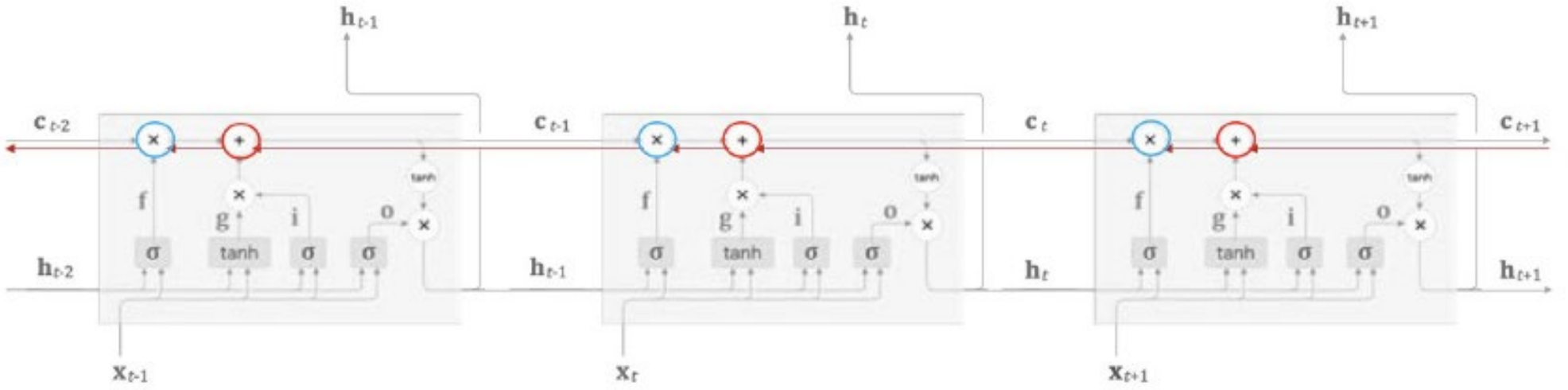
$$h_t = o_t * \tanh\left(C_t\right)$$

Tanh(cell state)는 -1 ~1 값을 가지고, 이곳에 output gate 곱해져 출력됨

& 다음 time step input으로 넘어감

3. LSTM

How to Protect Gradient Vanishing

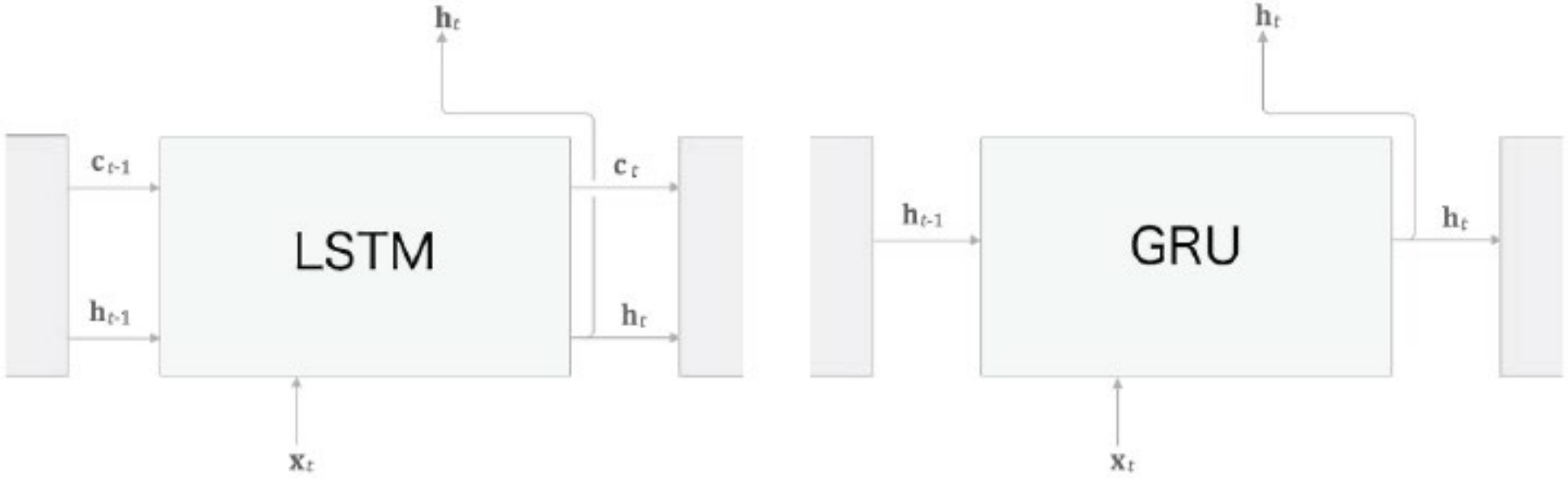


한번 cell state에 들어온 정보는 더 이상 nonlinear function 수행X

⇒ Vanilla RNN보다는 vanishing gradient 문제 줄어들 것

4. GRU

GRU



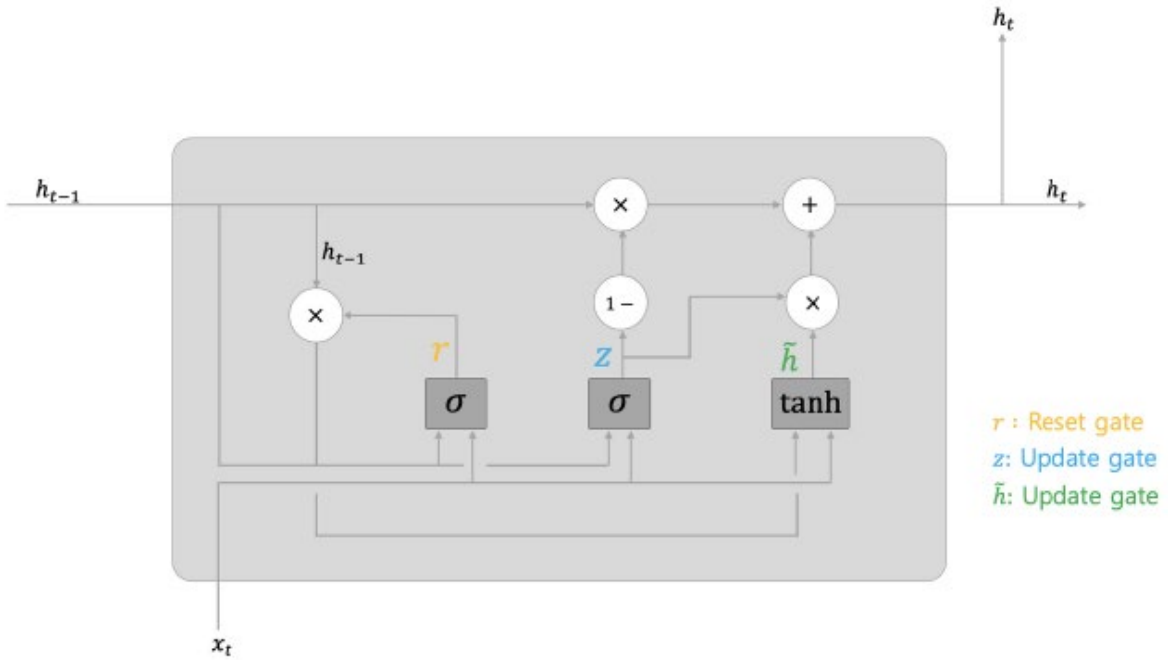
Idea: LSTM은 뭔가 중복해서 수행되는 연산이 존재함

⇒ LSTM에서 parameter 수를 줄이자

Gate가 3개에서 2개로 축소 & cell state 제거

4. GRU

GRU의 구조

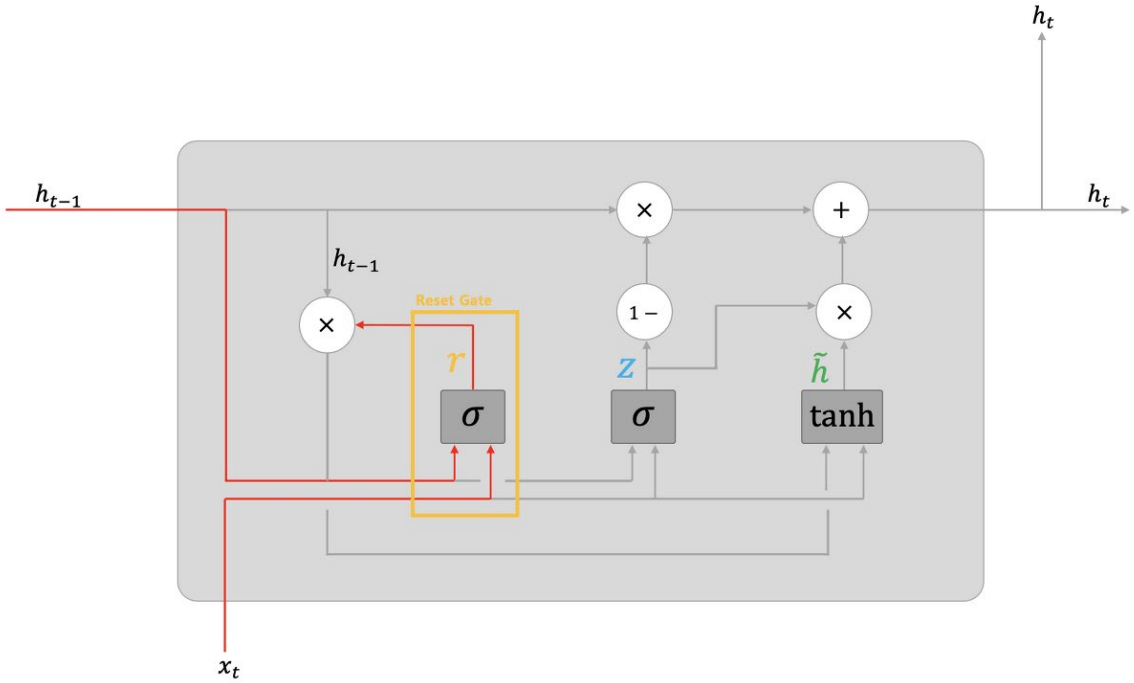


Reset Gate (= Output Gate)

Update Gate = input gate + forget gate

4. GRU

Reset Gate



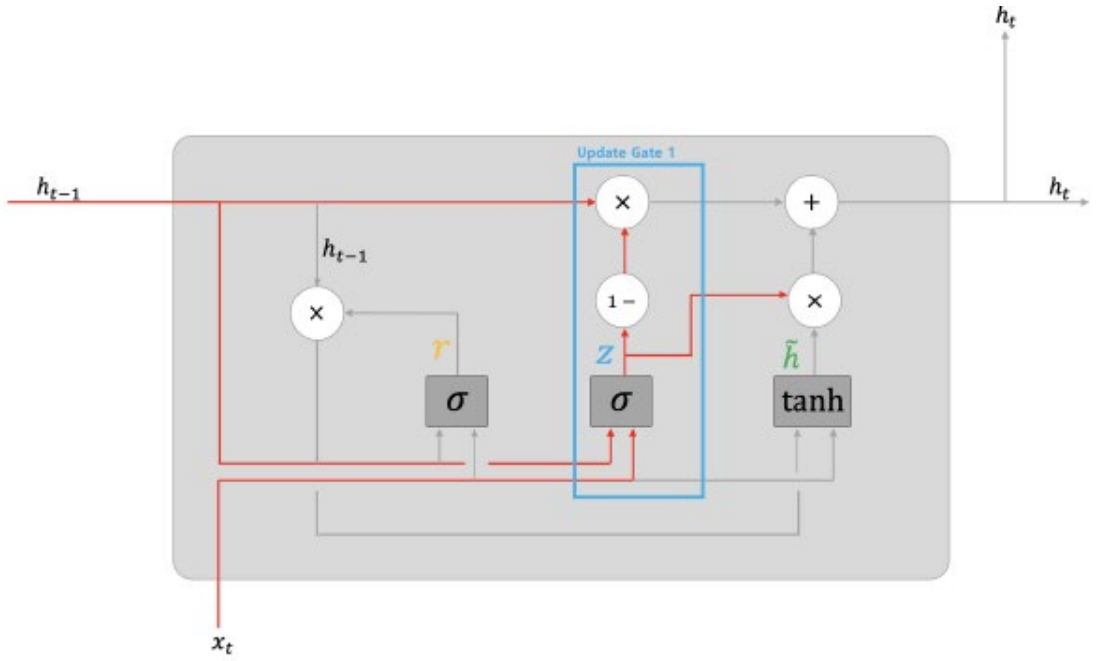
과거의 정보를 얼마나 남길 지

r=0 : 과거 정보 모두 버리기

$$r = \sigma(x_t W_x^{(r)} + h_{t-1} W_h^{(r)} + b^{(r)})$$

4. GRU

Update Gate



과거의 정보를 얼마나 남길 지

r=0 : 과거 정보 모두 버리기

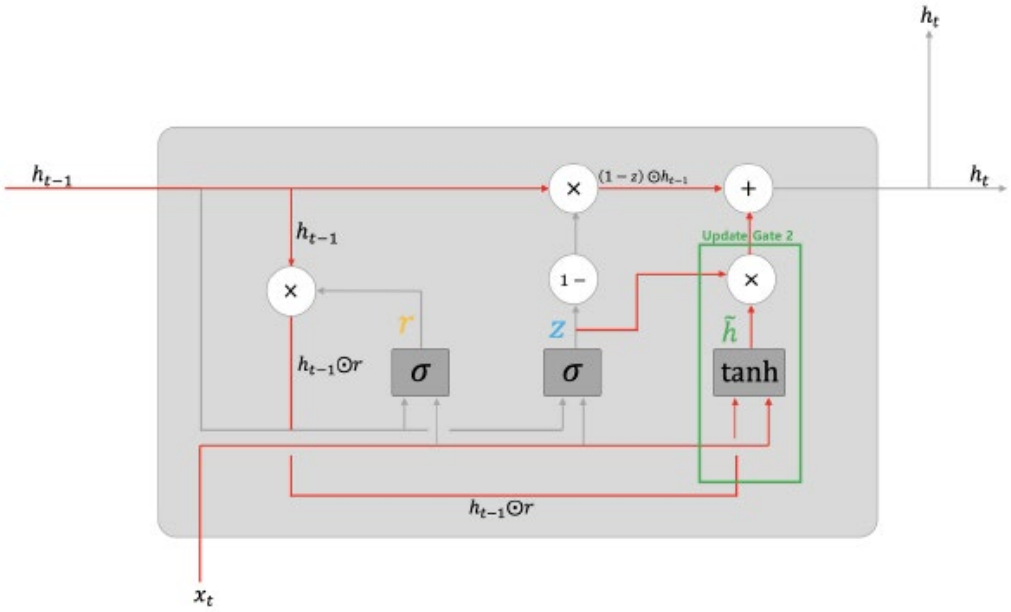
$$z = \sigma(x_t W_x^{(r)} + h_{t-1} W_h^{(r)} + b^{(r)})$$

$$(1 - z) \odot h_{t-1}$$

1-z : 과거 정보를 얼마나 남길지(forget gate)

4. GRU

Update Gate

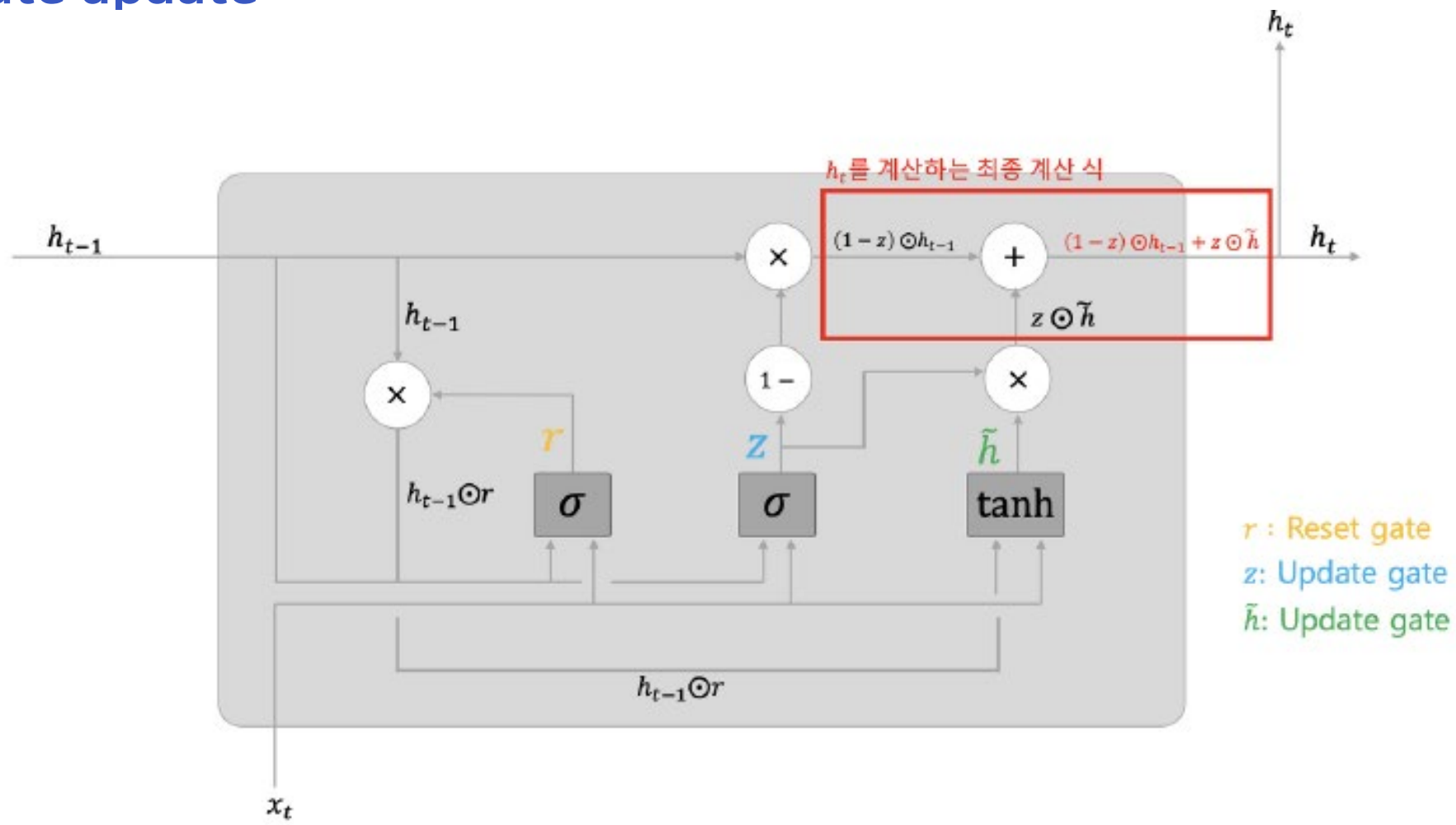


새로운 cell state 만들기

$$\tilde{h} = tanh(x_t W_x + (h_{t-1} \odot r) W_h + b)$$

4. GRU

Hidden state update



4. GRU

LSTM vs GRU

- 두 모델 모두 vanilla RNN을 보완해서 나온 모델

- GRU의 경우, LSTM보다 계산 더 빠르고 파라미터 수도 적음
- LSTM은 데이터가 많거나, 데이터가 long dependency 잇는 경우 적합
  (둘의 성능은 별 차이가 없다)

5. Summary

Summary

Why NLP?

방대한 Text data를 인간이 전부 학습하기엔 비효율적

Text Data를 computer에게 이해시키는 방법

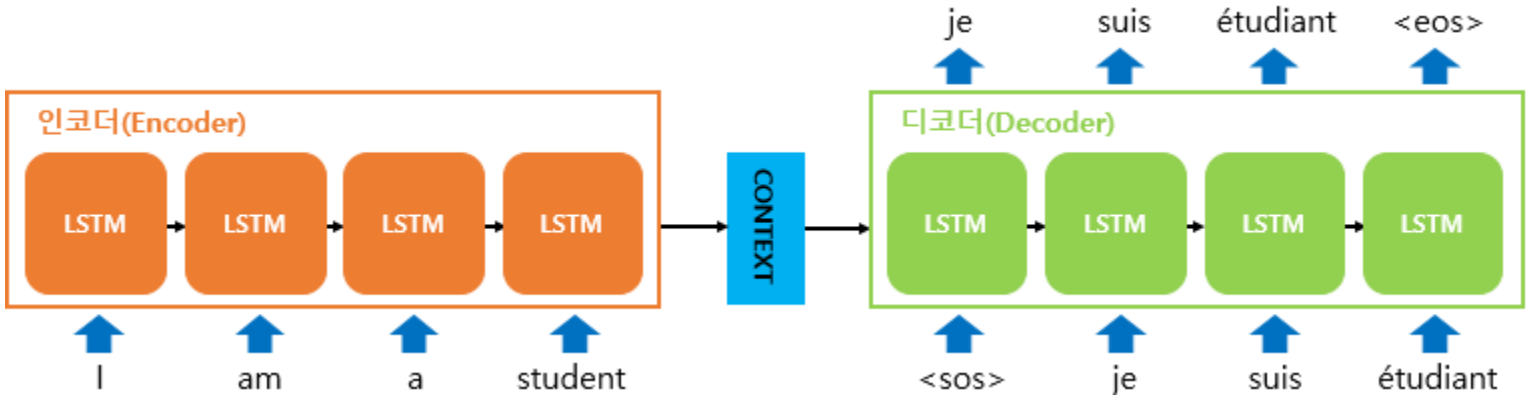
- Wordnet
- One-hot vector
- Word embedding (how? Word2vec (CBoW / SkipGram)

Text Data 다루는 model

- Fixed window NN
- RNN (기울기 소실) -> LSTM(장기기억 state 사용) & GRU (gate 간소화)

5. Summary

Next Session



긴 sequence에서 초기에 있던 정보 (I)를 다시 사용해야 할 때,

RNN, LSTM, GRU로는 맨 앞부분 정보가 중요한 걸 알려줄 수 없음

HOW?

⇒ ATTENTION!! (To be continued…)

5. Summary

Reference

7기 김채은님 세션 자료

22-2 DSL NLP 심화스터디

Stanford CS224N: NLP with Deep Learning

투빅스 텍스트 세미나 : https://velog.io/@tobigs-text1314/CS224n-Lecture-7-Vanishing-Gradients-And-Fancy-RNNs

Jiho-ml 뉴스레터 : https://jiho-ml.com/weekly-nlp-10/

딥러닝을 이용한 자연어 처리 입문 wikidocs https://wikidocs.net/book/2155

딥러닝 홀로서기 : https://www.youtube.com/watch?v=cs3tSnAsyRs &

https://www.youtube.com/watch?v=tlyzfIYvMWE&t=5129s

핵심 머신러닝 : https://www.youtube.com/watch?v=pjw7yfpk2jc

DATA SCIENCE LAB

발표자 김지희 010-6423-1824
E-mail: jihee_sta@yonsei.ac.kr