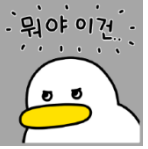
딥러닝팀

1팀

정승민
변석주
이정환
송승현
최용원

CONTENTS

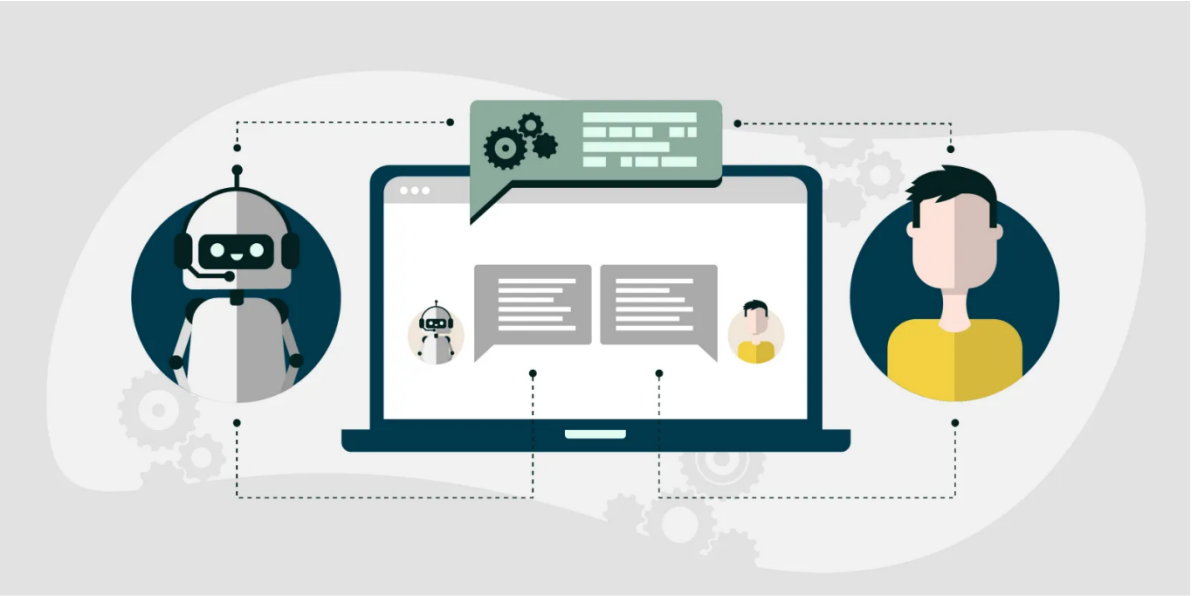


1

자연어

자연어

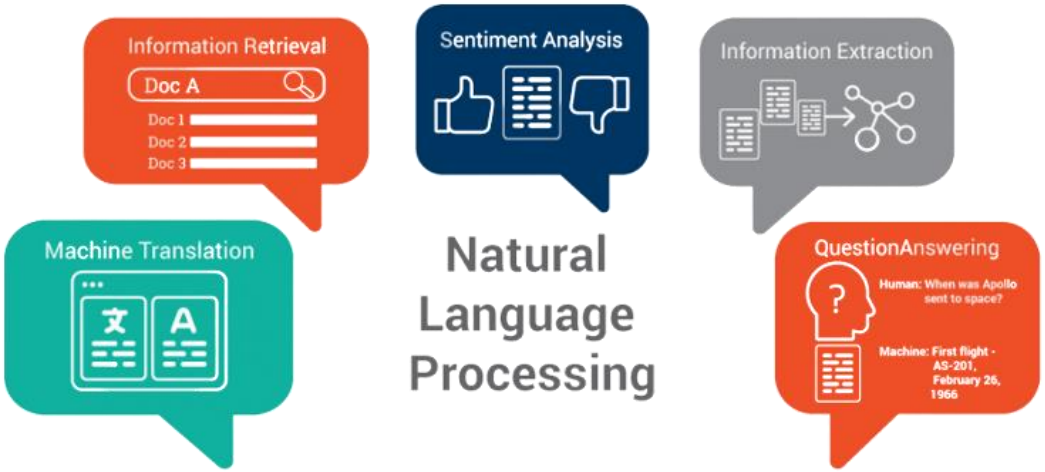
자연어
Natural Language



일상생활에서 사용하는 언어를
인공적으로 만들어진 인공어와 구분하는 개념

자연어 처리

Natural Language Processing (NLP)



자연어를 컴퓨터가 이해할 수 있는 형태로 변환하는 작업

감성 분석, 챗봇, 기계번역, 자동요약에 사용됨

자연어

자연어의 특징

순차성

자연어는 순서가 달라질 경우 의미가 손상되는 순차적인 데이터

RNN은 자연어의 순차적인 특징 반영

"마감시간이 끝나기 전에 과제를 끝냈다."

"과제가 끝나기 전에 마감시간이 끝났다."

불연속성

형태가 유사하더라도 완전히 다른 의미 가질 수 있음

경제 / 결제 , 지양 / 지향

자연어의 특징

모호성

표현의 중의성

"차를 마시러 공원에 가는 차 안에서 나는 그녀에게 차였다."

➡ '차'는 각각 의미가 다른 동음이의어

문장 내 정보의 부족 문제

"나는 철수를 안 때렸다."

➡ 정보 생략으로 문장 해석의 어려움

대응 형태의 다양성

자연어의 전처리

정제 (Cleaning)

텍스트 데이터의 의미 분석에 필요하지 않은 변수 제거

Apple | apple → 같은 의미

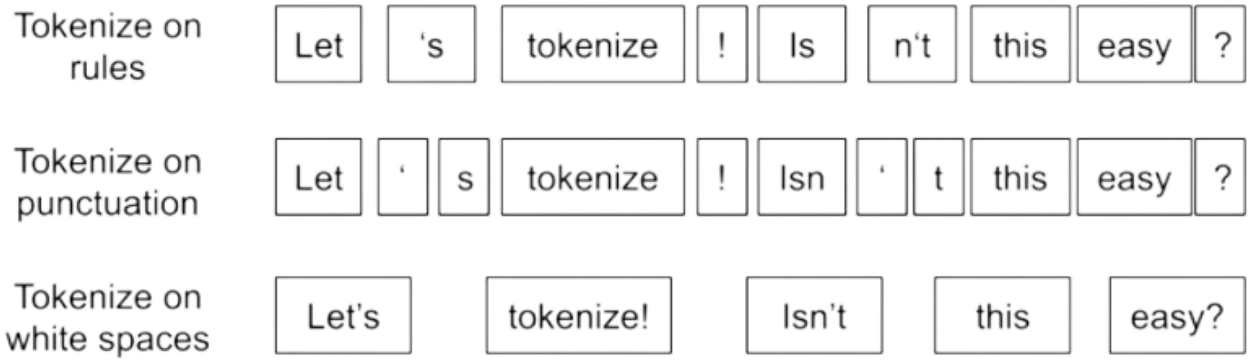
US (미국) | us (우리) → 다른 의미

코퍼스(corpus): 정제의 대상이 되는 전체 문장 데이터 집합

필요한 부분에 한해 대문자를 남기고 소문자를 제거하는 과정 진행

자연어의 전처리

토큰화 (Tokenize)



코퍼스를 의미를 가지는 최소 단위로 잘게 쪼개는 과정

토큰 (token): 의미를 가지는 최소 단위

자연어의 전처리

토큰화 (Tokenize)

```
print('단어 토큰화1 :',word_tokenize("Don't be fooled by the dark sounding name, Mr. Jone's Orphanage is as cheery as cheery goes for a pastry shop."))
```

```
단어 토큰화1 : ['Do', "n't", 'be', 'fooled', 'by', 'the', 'dark', 'sounding', 'name', ',', 'Mr.', 'Jone', "'s", 'Orphanage', 'is', 'as', 'cheery', 'as', 'cheery', 'goes', 'for', 'a', 'pastry', 'shop', '.']
```

nltk - word_tokenize()

가장 기본적인 토큰화 함수

띄어쓰기 단위와 구두점 기준으로 토큰화 진행

nltk - WordPuncTokenizer() 함수는 구두점 별도 분리

자연어의 전처리

토큰화 (Tokenize)

```
print('단어 토큰화3 :',text_to_word_sequence("Don't be fooled by the dark sounding name, Mr. Jone's Orphanage is as cheery as cheery goes for a pastry shop."))
```

```
단어 토큰화3 : ["don't", 'be', 'fooled', 'by', 'the', 'dark', 'sounding', 'name', 'mr', "jone's", 'orphanage', 'is', 'as', 'cheery', 'as', 'cheery', 'goes', 'for', 'a', 'pastry', 'shop']
```

keras – test_to_word_sequence()

대문자를 소문자로 일괄 변환

구두점 제거

자연어의 전처리

토큰화 (Tokenize)

```
print('단어 토큰화3 :',text_to_word_sequence("Don't be fooled by the dark sounding name, Mr. Jone's Orphanage is as cheery as cheery goes for a pastry shop."))
```

```
단어 토큰화3 : ["don't", 'be', 'fooled', 'by', 'the', 'dark', 'sounding', 'name', 'mr', "jone's", 'orphanage','is', 'as', 'cheery', 'as', 'cheery', 'goes', 'for', 'a', 'pastry', 'shop']
```

라이브러리와 **함수**에 따라 여러 방식의 토큰화 가능

정의와 **목적**에 따라 다양한 방식의

토큰화 방법 선택 필요

- keras - test_to_word_sequence()
  - 대문자를 소문자로 일괄 변환
  - 구두점 제거

자연어의 전처리

한국어의 토큰화

한국어의 특징

어간에 접사가 붙어 단어를 이루고, 의미와 문법적 기능이 더해지는 교착어

어간 '가-' + 접사 '-ㅆ다' = 단어 '갔다'

한국어의 토큰화

형태소 (morpheme) 단위로 토큰화 진행

Word Vector

One-hot-vector

단어 집합 (vocabulary)

이산적인 단어의 의미를 표현하기 위해

서로 다른 단어들을 중복하지 않고 모아 놓은 집합

One-hot-encoding 

One-hot-vector

N개의 단어를 각각 N개 차원의 벡터로 표현한 것

Word Vector

One-hot-encoding

One-hot-encoding

표현하고 싶은 단어의 index에는 1을, 나머지에는 0을 넣는 방법

희소표현: 벡터 또는 행렬의 값이 대부분 0으로 표현되는 방법

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

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

서로 직교 (orthogonal) 하기 때문에 유사성 찾을 수 없음

→ 검색 결과에 반영 어려움

Word Vector

One-hot-encoding

One-hot-encoding

표현하고 싶은 단어의 index에는 1을, 나머지에는 0을 넣는 방법

N개의 차원 형성으로 인해 **차원의 저주**

희소표현: 벡터 또는 행렬의 값이 대부분 0으로 표현되는 방법

유의미한 정보가 있는 공간이 적음 → **공간 낭비 문제**

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

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

서로 직교 (orthogonal) 하기 때문에 유사성 찾을 수 없음

→ 검색 결과에 반영 어려움

Word Vector

Word Embedding

밀집 표현

모든 단어의 벡터 차원을 사용자가 설정한 값으로 맞춤

→ 실수 값으로 벡터 구성

hotel = [0.7 0.3 0.1 0.5]

motel = [0.5 0.1 0.3 0.6]

차원 = 4인 밀집 벡터

Word Embedding

밀집 벡터의 각 요소로 서로 다른 특징 표현

의미를 고려하여 텍스트 구분

단, 각 요소의 의미는 알 수 없음

자연어

Word Vector

Word2Vec

"비슷한 위치에 등장하는 단어는 비슷한 의미를 가진다"

CBow

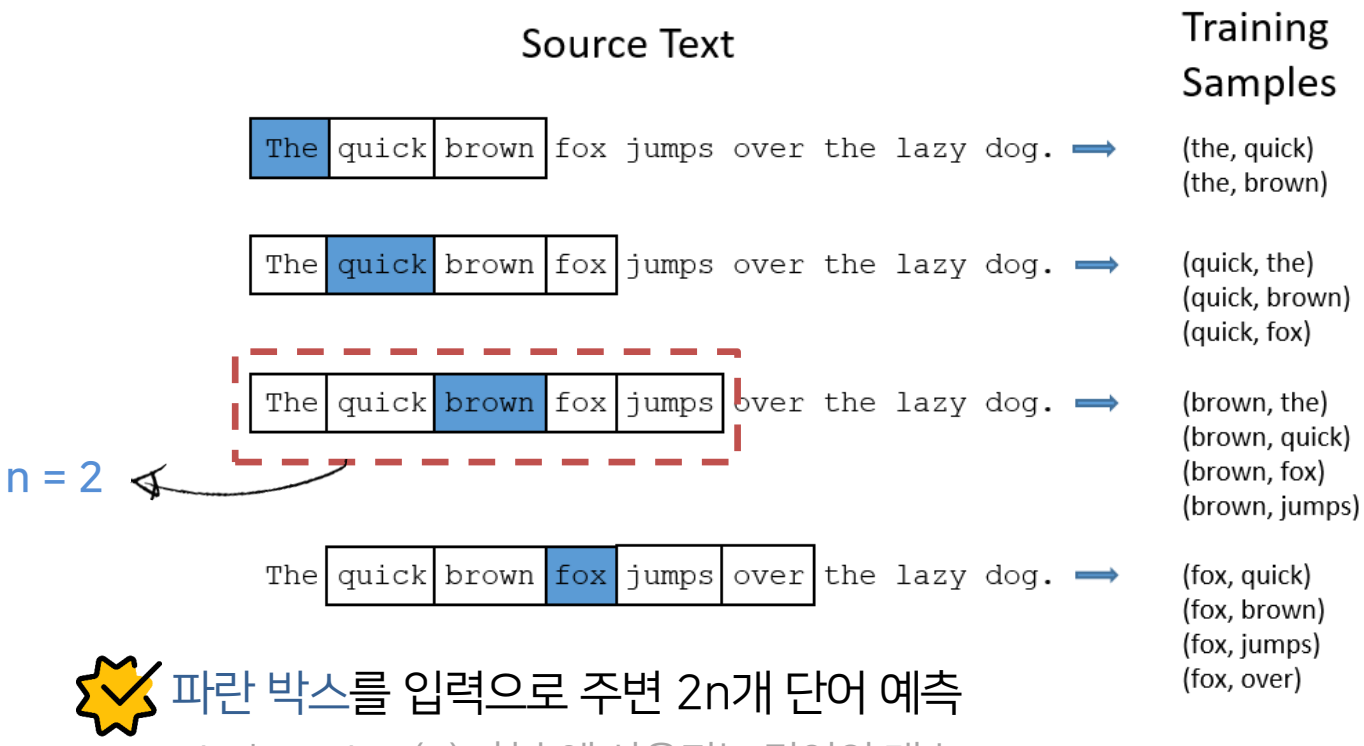
주변의 단어들을 입력으로 사용하여 중간에 있는 단어 예측하는 방법

Skip-gram *성능이 좋아 더 많이 사용*

중간에 있는 단어를 입력으로 사용하여 주변 단어들을 예측하는 방법

Word Vector

Word2Vec – Skip_gram

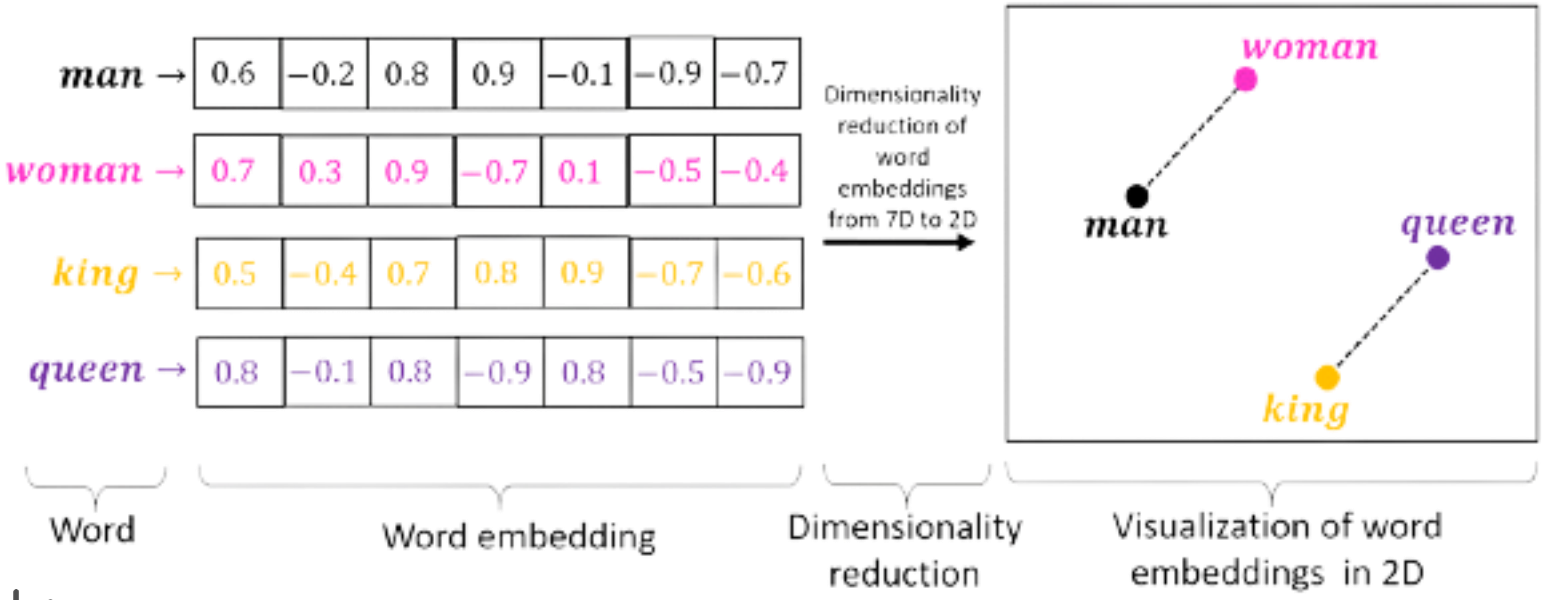


파란 박스를 입력으로 주변 2n개 단어 예측

window size (n): 학습에 사용되는 단어의 개수

Word Vector

Word Embedding



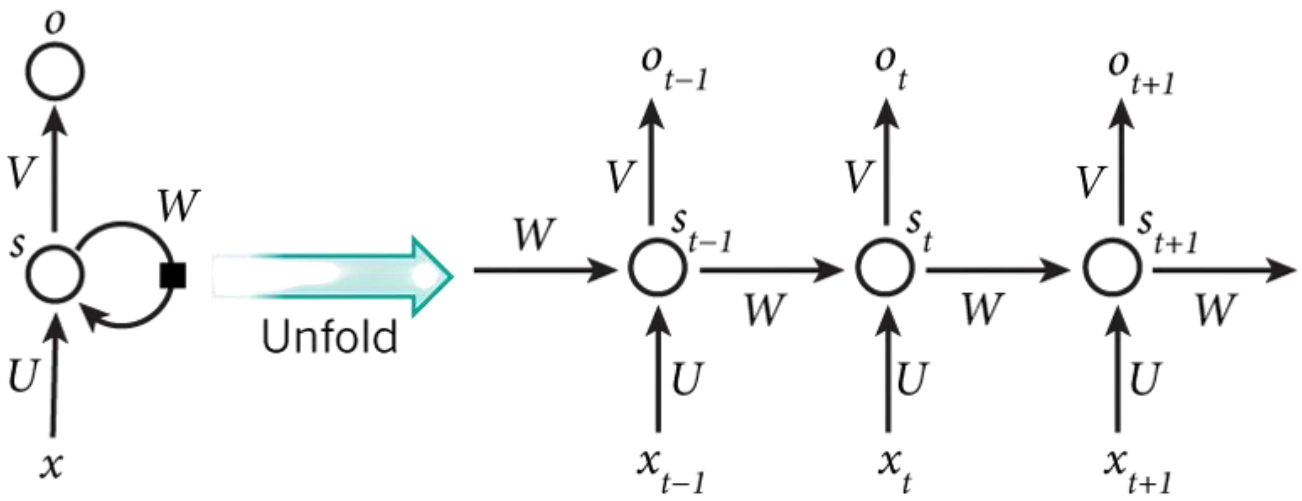
제한된 문장 안에서 여러 번 학습 진행하여 임베딩 벡터 생성

단어가 고차원의 한 점으로 표시되어 벡터 간 연산 가능

2

RNN

RNN (Recurrent Neural Network)



RNN

순차적인 자료 처리하는 신경망

매개변수 공유를 통해 가변 길이 순차열 처리 가능

입력 데이터의 길이가 다른 데이터 학습 가능

RNN (Recurrent Neural Network)

계산 그래프 펼치기

점화식 구조

$$a_{n+1} = f(a_n), \qquad n = 1, 2, \cdots$$

RNN

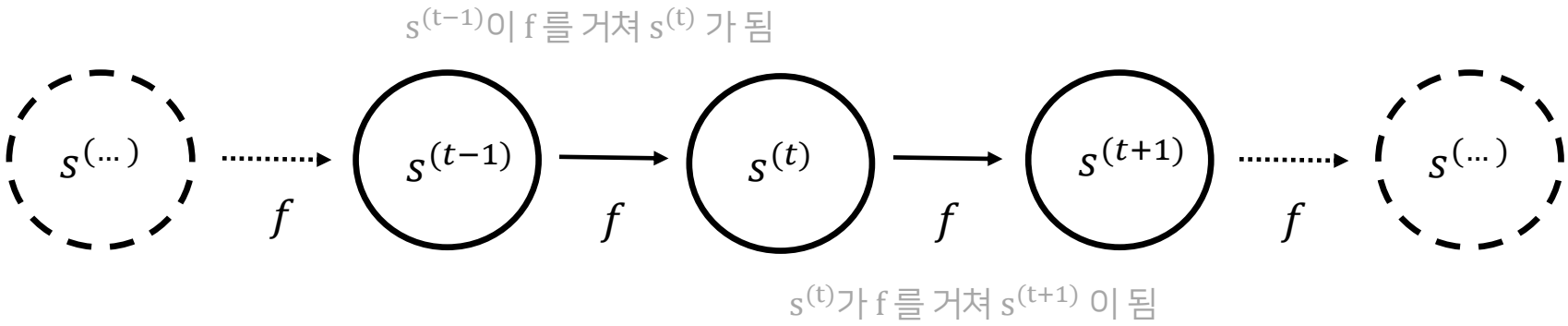
$$s^{(t)} = f\left(s^{(t-1)};\ \theta\right)$$

노드의 상태 (state)

점화식 구조로 이전 항을 통해 다음 항 구조를 파악할 수 있음

RNN (Recurrent Neural Network)

계산 그래프 펼치기



$$s^{(t+1)} = f\left(s^{(t)};\ \theta\right) = f\left(f\left(s^{(t-1)};\ \theta\right);\ \theta\right)$$

위 수식은 자기 자신의 과거 정보만을 반영

→ 현재 시점의 외부 정보 반영 필요

RNN (Recurrent Neural Network)

계산 그래프 펼치기

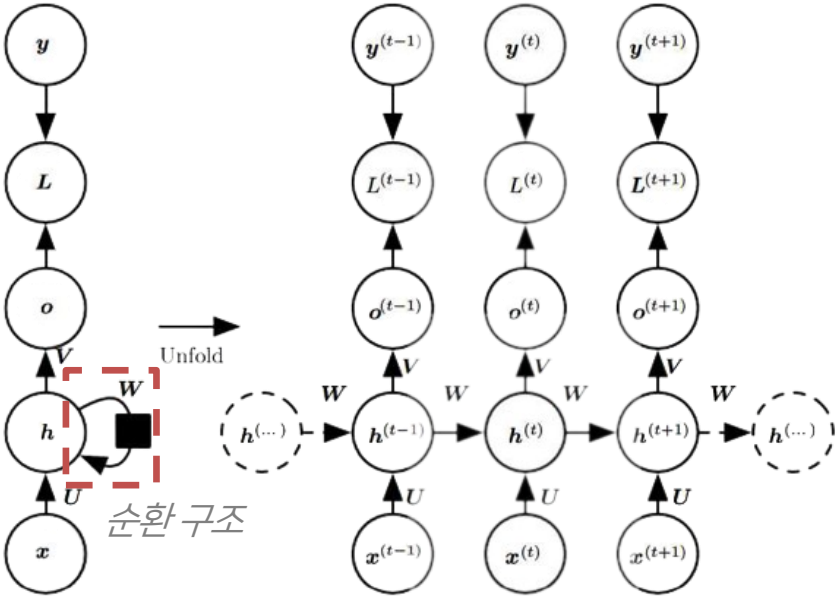
손실이 있는 과거 요약 함수

$$h^{(t)} = f\left(h^{(t-1)}, x^{(t)}; \theta\right)$$

과거시점의 정보 ($h^{(t-1)}$) + 현재 시점의 정보 ($x^{(t)}$) 사용

$h^{(t-1)}$: RNN의 hidden layer 이전 시점의 데이터를 기억하는 메모리 셀(memory cell)

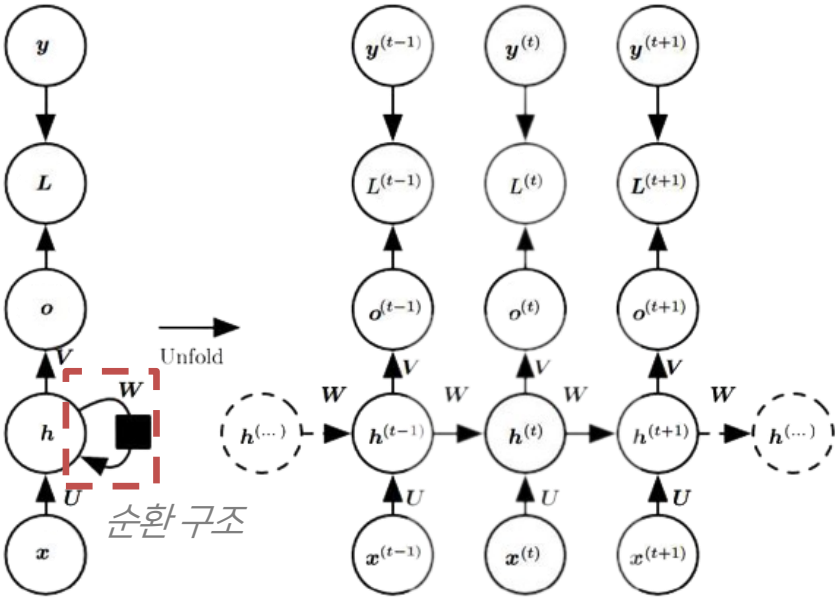
Vanilla RNN



$$h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \dots, x^{(2)}, x^{(1)})$$

$$= f(h^{(t-1)}, x^{(t)}; \theta)$$

Vanilla RNN

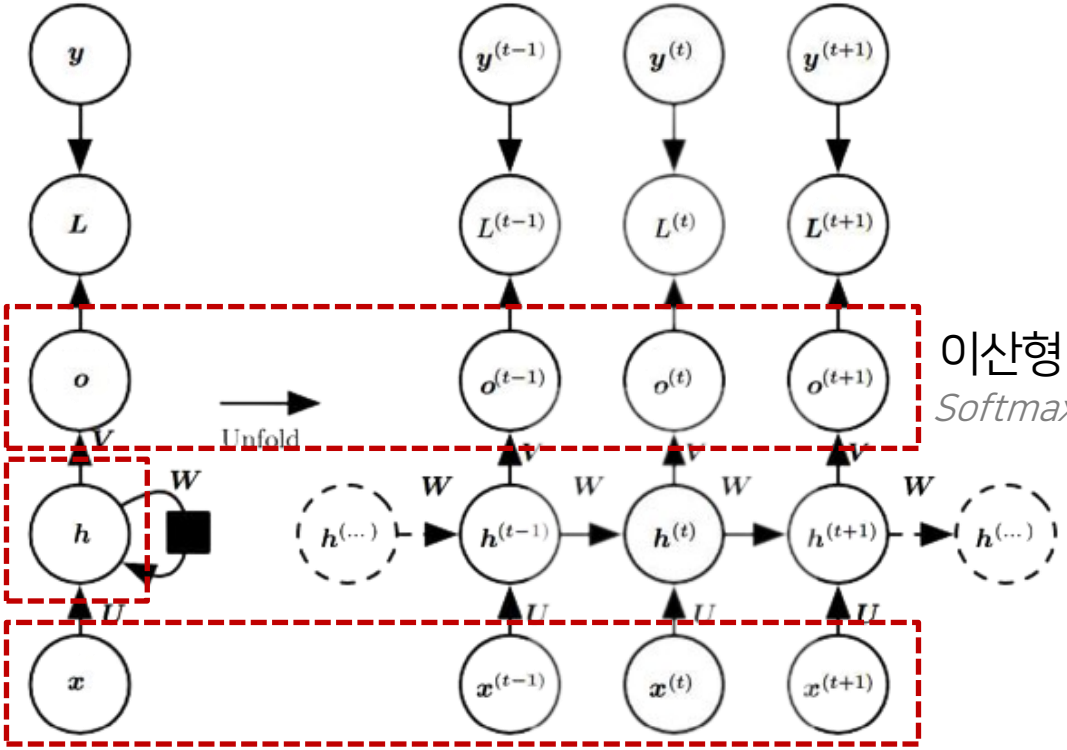


시간 순서의 데이터를 학습함에 있어 하나의 Layer 반복 사용

→ $h^{(t)}$는 1 ~ t 시점의 모든 데이터를 반영

Vanilla RNN

작동 원리



이산형 출력값

Softmax 함수 사용!

$h^{(t)}$: Memory Cell

활성화 함수 : tanh

$x^{(t)}$: 입력

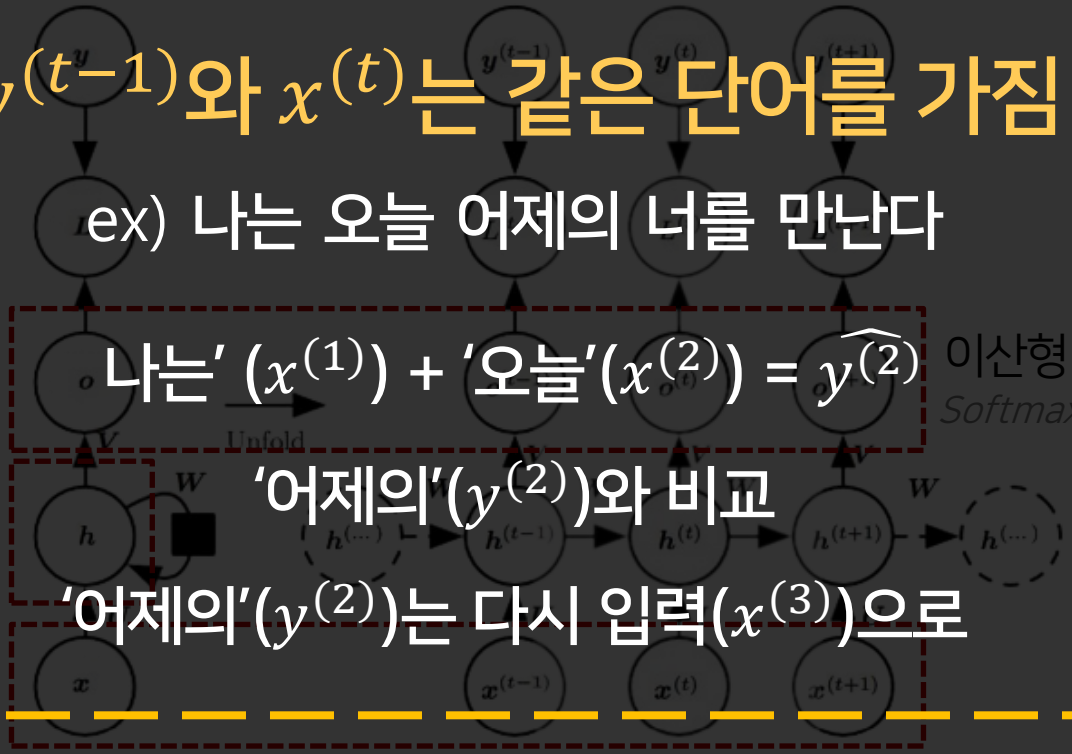
RNN

Vanilla RNN

작동 원리

$y^{(t-1)}$와 $x^{(t)}$는 같은 단어를 가짐

ex) 나는 오늘 어제의 너를 만난다



'나는'($x^{(1)}$) + '오늘'($x^{(2)}$) = $\widehat{y^{(2)}}$

이산형 출력값

Softmax 함수 사용!

'어제의'($y^{(2)}$)와 비교

'어제의'($y^{(2)}$)는 다시 입력($x^{(3)}$)으로

$h^{(t)}$: Memory Cell

활성화 함수 : tanh

$x^{(t)}$: 입력

Vanilla RNN

단계별 수식

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$$h^{(t)} = tanh(a^{(t)})$$

$$o^{(t)} = c + Vh^{(t)}$$

$$\widehat{y^{(t)}} = softmax(o^{(t)})$$

$a^{(t)}$ = 이전 단계의 입력 및 hidden state의 가중치 합

$h^{(t)}$ = $a^{(t)}$ 를 바탕으로 도출한 현재 시점의 hidden state

$o^{(t)}$ = t 시점의 출력 결과

$\widehat{y^{(t)}}$ = $o^{(t)}$ 를 softmax 함수에 통과

매개 변수 : 편향 벡터(b, c) + 가중치 행렬(W, U, V)

시점에 의존하지 않는 모습

순전파 과정에서 매개변수는 공유되면 변하지 않음!

Vanilla RNN

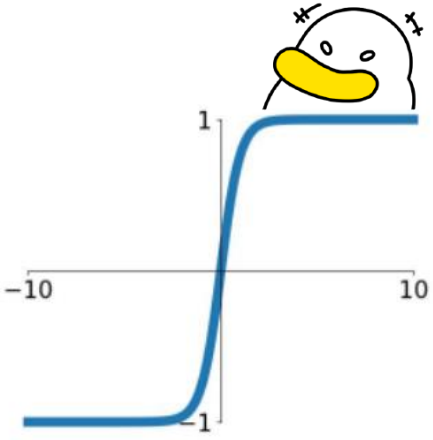
단계별 수식

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$$h^{(t)} = tanh(a^{(t)})$$

$$o^{(t)} = c + Vh^{(t)}$$

$$\widehat{y^{(t)}} = softmax(o^{(t)})$$



tanh(x)

$\tanh(x) = 2\sigma(2x) - 1.$

활성화함수 : $tanh\ (0 \leq |h^{(t)}| \leq 1)$

다음 단계로 전달될 정보를 효과적으로 표현 가능

ReLU : RNN의 순환구조로 인해 값이 발산할 수 있기 때문에 부적절

Vanilla RNN

손실 함수

$$L(\{x^{(1)}, \dots, x^{(\tau)}\}, \{y^{(1)}, \dots, y^{(\tau)}\})$$

$$= \sum_t L^{(\tau)}$$

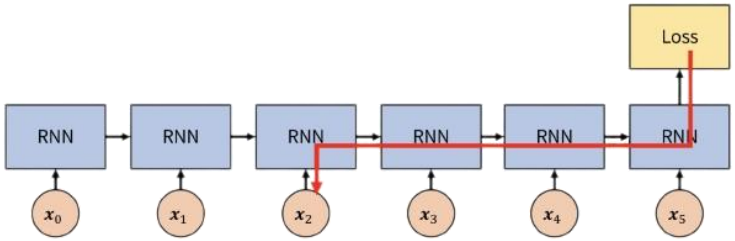
$$= \sum_t -\log(\hat{y}^{(t)}_{y^{(t)}})$$

각 시점의 손실 함수의 총합

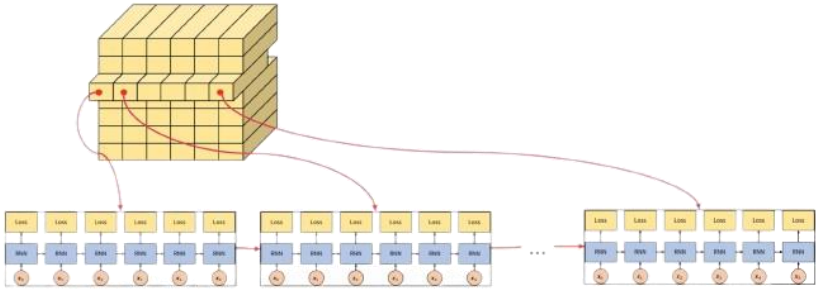
역전파 (Back Propagation)

BPTT, Truncated BPTT

BPTT



Truncated BPTT

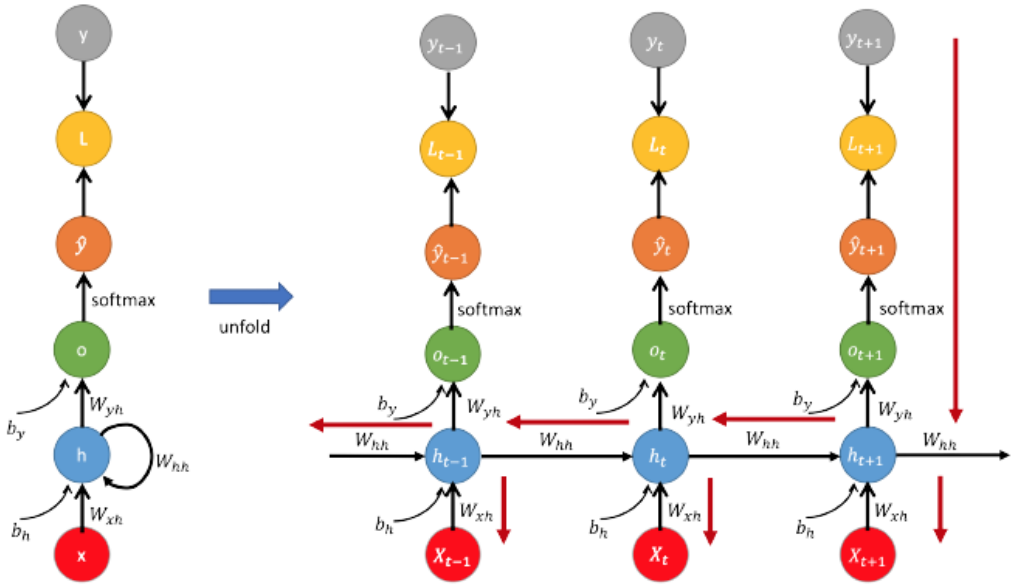


모델을 통해 해결하고자 하는 문제,

Sequential Data의 크기에 따라 다르게 정의

BPTT

back-propagation through time

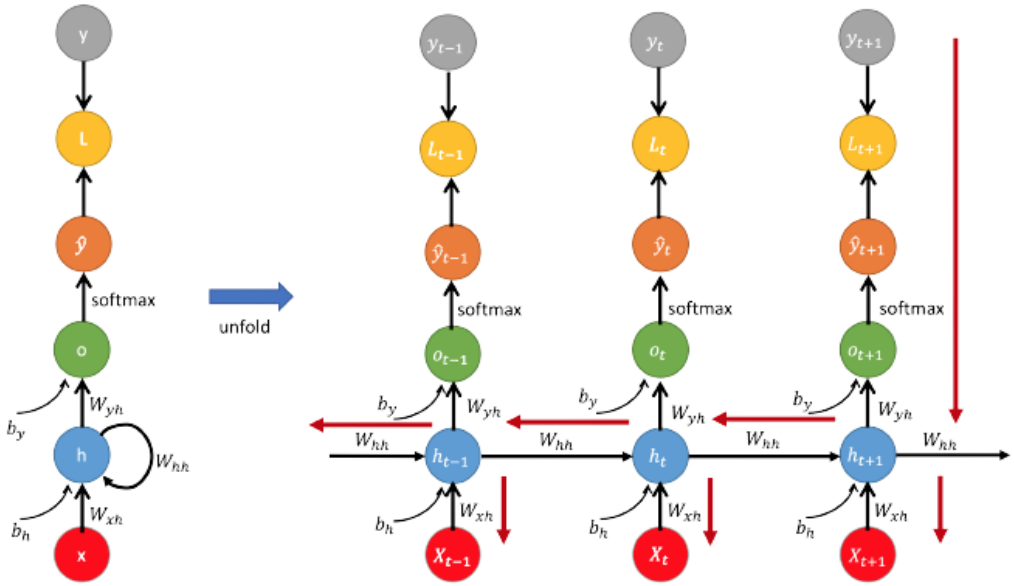


매 시점마다 출력이 반환되는 형태의 RNN에서 활용

매 시점마다 손실함수 계산 가능

BPTT

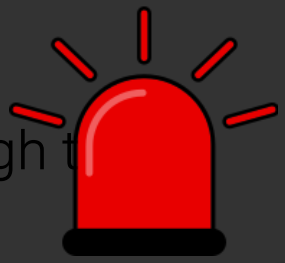
back-propagation through time

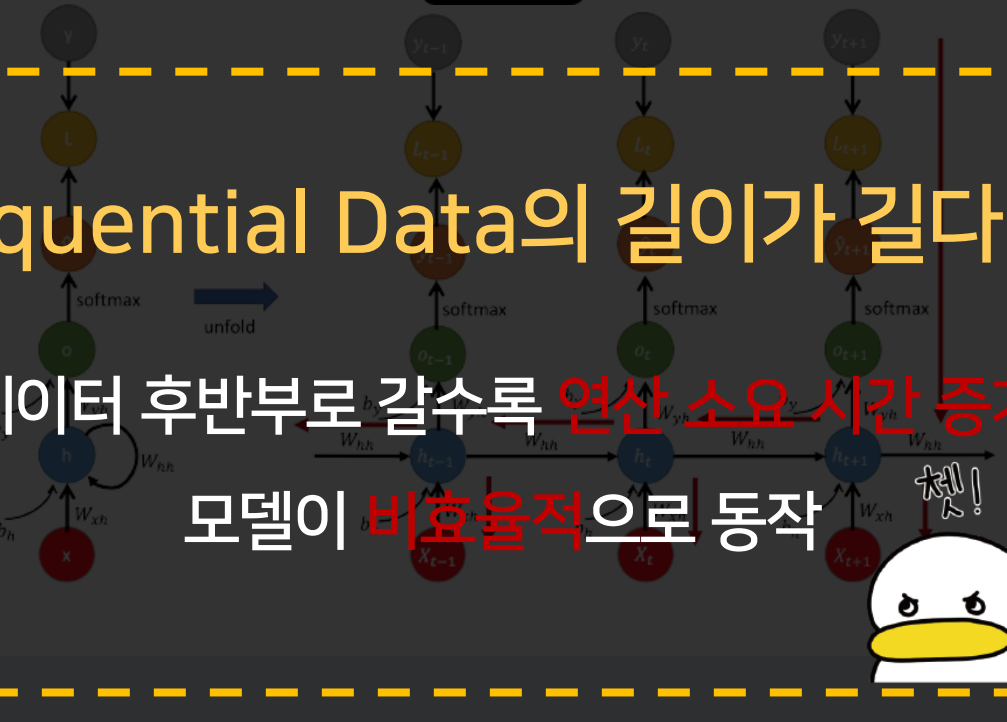


gradient를 계산해 첫 시점의 입력까지 역전파를 진행

BPTT

back-propagation through t





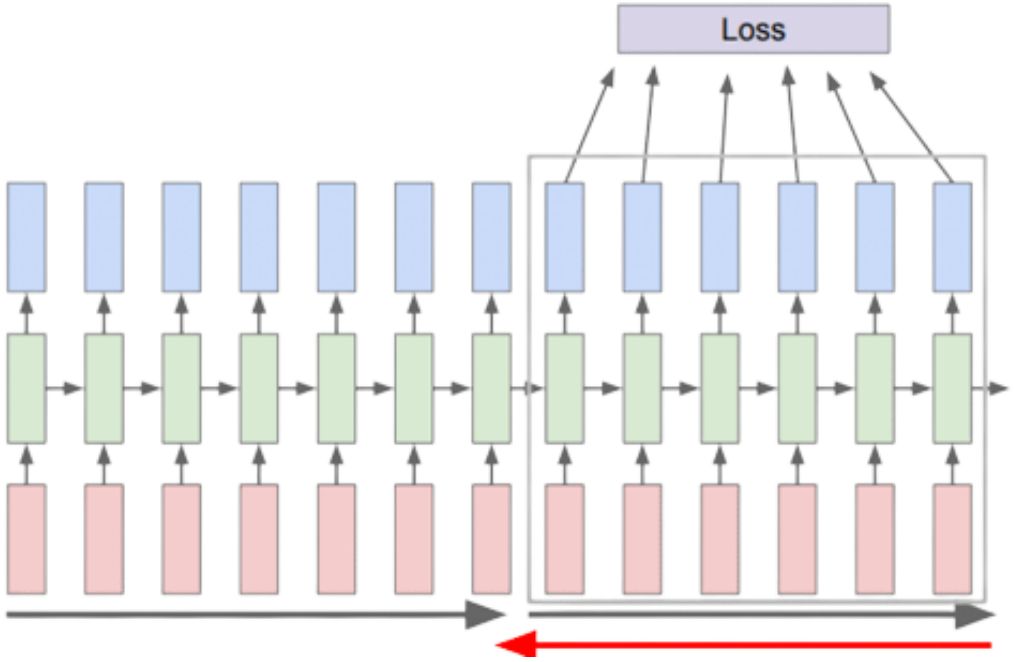
Sequential Data의 길이가 길다면?

데이터 후반부로 갈수록 연산 소요 시간 증가

모델이 비효율적으로 동작

gradient를 계산해 첫 시점의 입력까지 역전파를 진행

Truncated BPTT

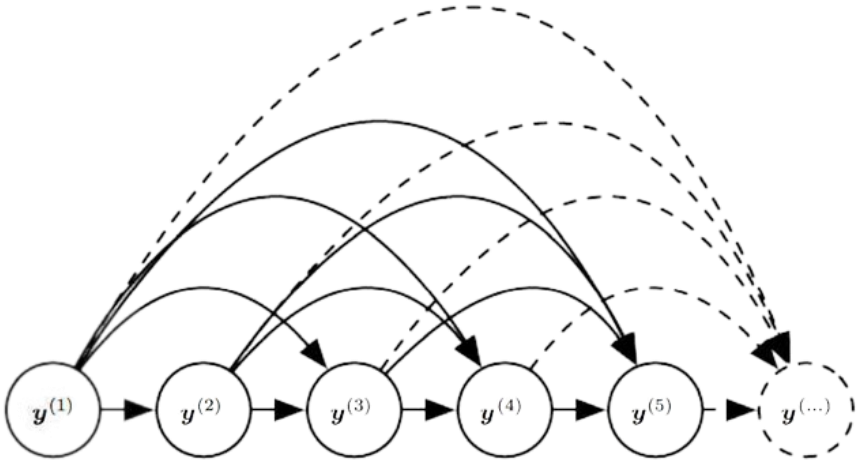


Sequential Data를 일정한 구간으로 끊어

구간마다 BPTT 진행

완전 연결 그래프 모형

Sequential 데이터에 대한 모델



$$P(Y) = P(y^{(1)}, \dots, y^{(\tau)}) = \prod_{t=1}^{\tau} P(y^{(t)} | y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)})$$

$y^{(t)}$: 이전 모든 시점의 y에 대해 영향을 받음

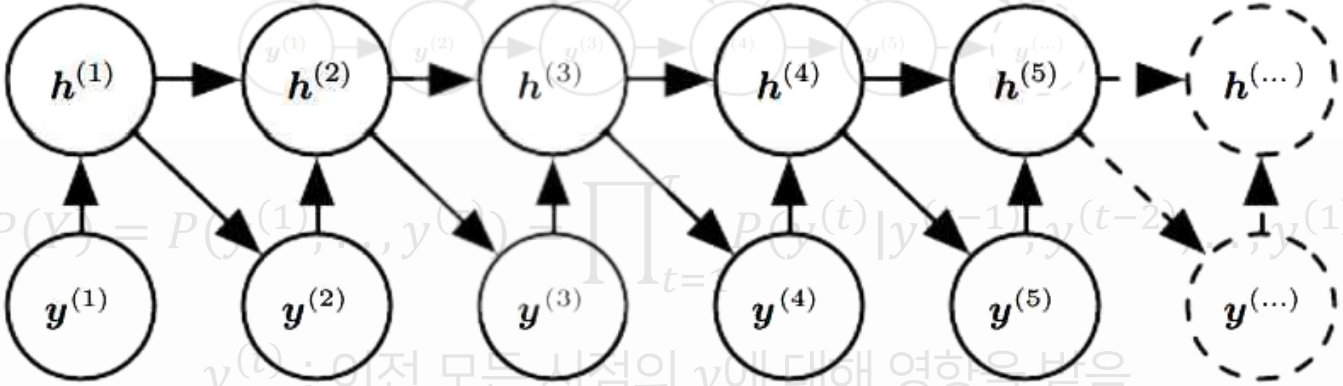
Data의 길이가 길어지면 매우 비효율적

입력의 수(τ)에 따라 매개변수가 증가하기 때문!

완전 연결 그래프 모형

Sequential 데이터에 대한 모델

Hidden state를 통한 매개변수의 공유로 해결



$P(Y) = P(y^{(1)}, y^{(2)}, \dots) = \prod_{t=} P(y^{(t)} | y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)})$

$y^{(t)}$: 이전 모든 시점의 y에 대해 영향을 받음

Data의 길이가 길어지면 매우 비효율적

입력의 수(τ)에 따라 매개변수가 증가하기 때문!

장기 의존성 문제

Long-Term Dependency Problem

단어 간의 거리가 가까워
붉은 상자의 단어를 효과적으로 예측

The clouds are in the sky

vs

I grew up in France ... when i was young ...
I speak Fluent French

단어 간의 거리가 멀어
붉은 상자의 단어 예측이 어려움

Vanilla RNN의 문제

$tanh(x)$로 인해 결과값이 점차 작아짐

Sequential 데이터의 길이에 취약

모델이 장기 기억을 제대로 처리하지 못하는 것

LSTM

Long Short-Term Memory

사람의 기억이
장기와 단기로 구분된 것에서 착안

RNN

$h^{(t)}$ (hidden state)

: 데이터 전반의 정보

포괄적 담당

LSTM

$h^{(t)}$ (hidden state)

: 단기 기억 담당

$c^{(t)}$ (cell state)

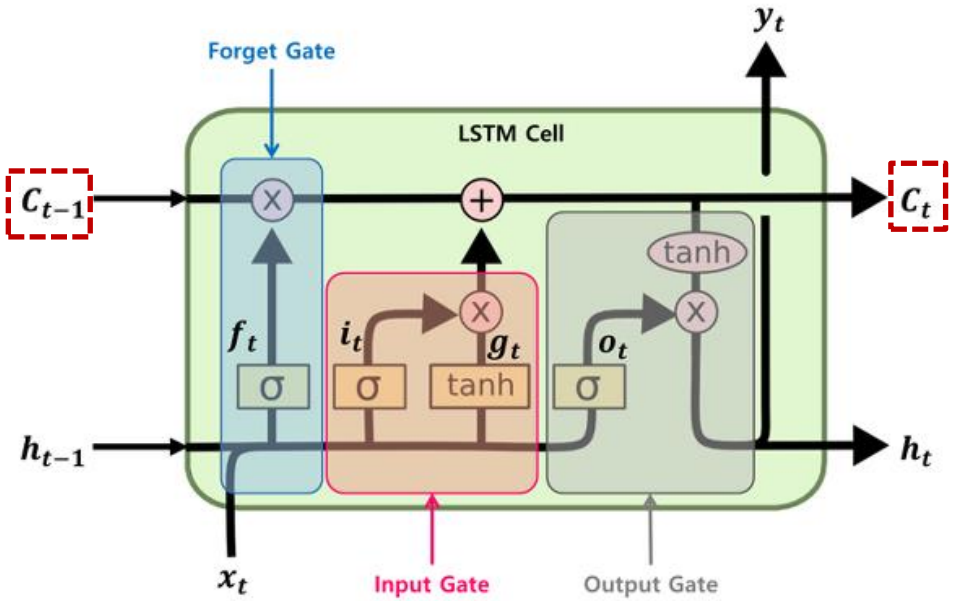
: 장기 기억 담당

데이터를 장기와 단기로 구분

기존 RNN의 **장기 의존성 문제**를 해결!

LSTM

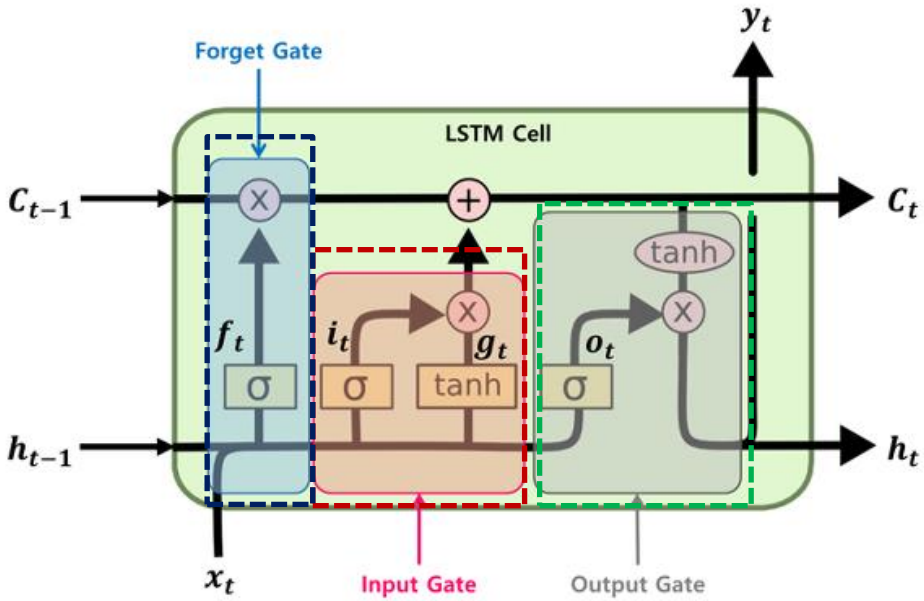
LSTM의 구조



$c^{(t)}$: 많은 연산 없이 다음 time-step으로 정보 전달

LSTM

LSTM의 구조



Gate

각각의 State에 어떤 값을 저장할지 정해주는

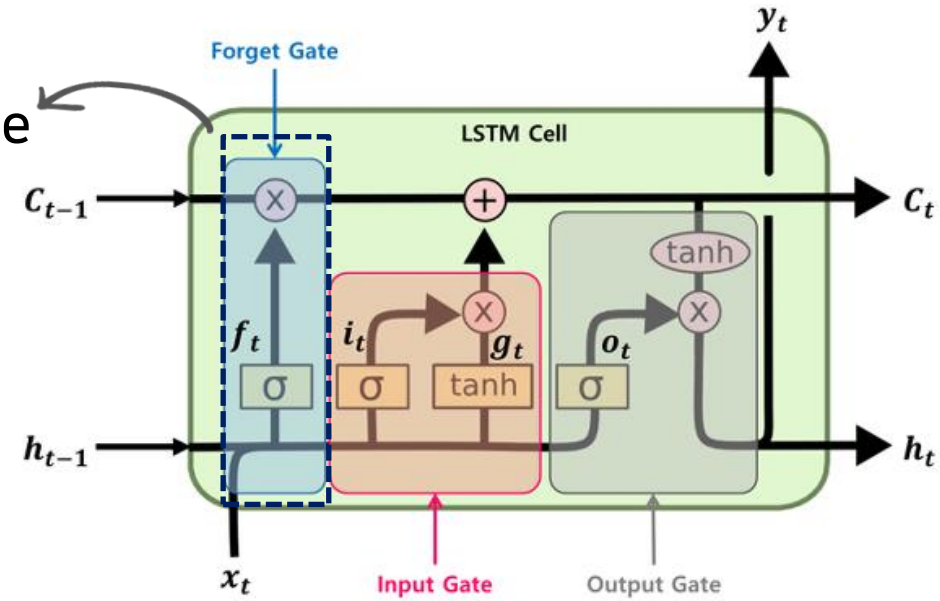
3가지 Gate 존재

Forget, Input, Output!

LSTM

LSTM의 구조

Forget Gate



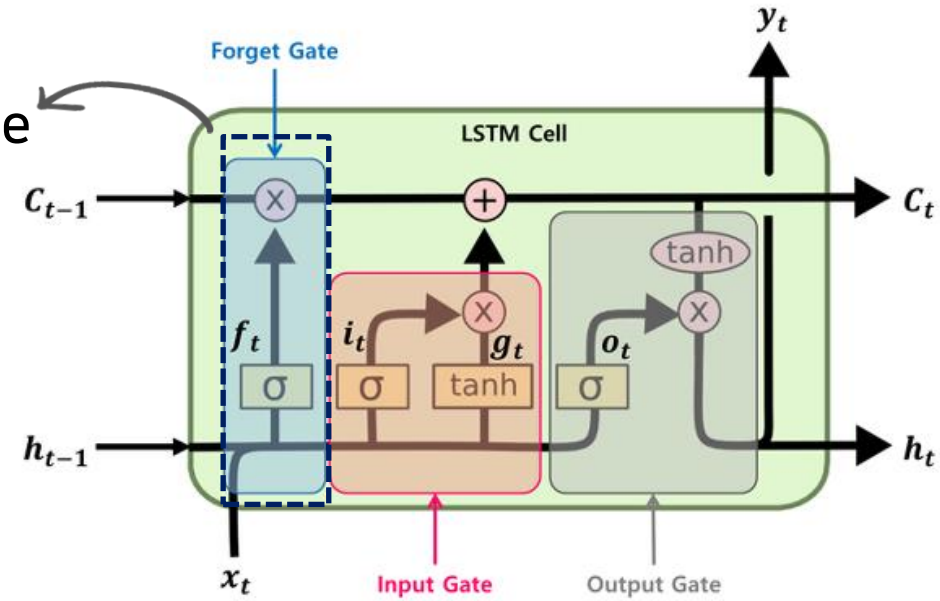
Forget Gate

이전 시점의 hidden state $h^{(t-1)}$와 현재 시점의 입력 $x^{(t)}$를 가중치 W_f와 곱해 시그모이드 통과시켜 Cell State로 전달

LSTM

LSTM의 구조

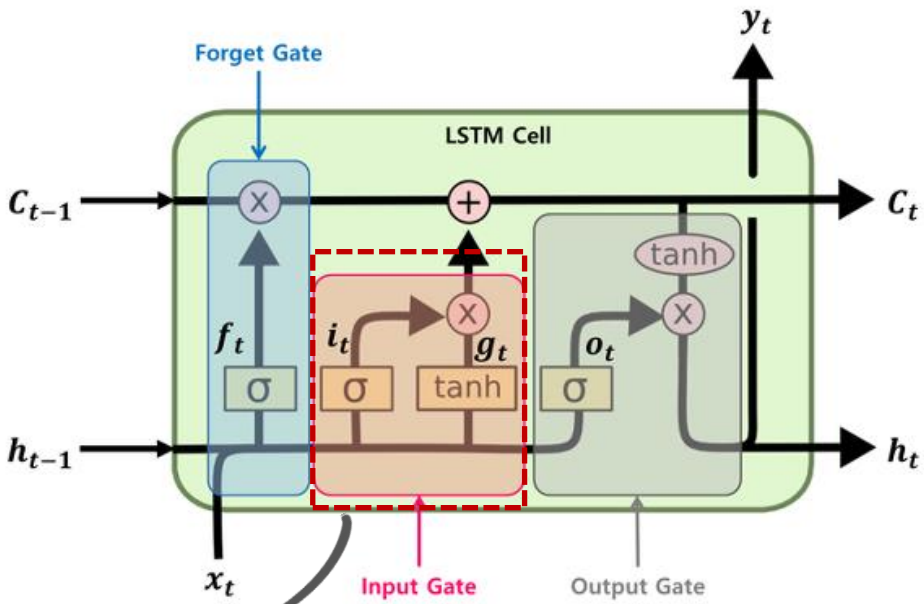
Forget Gate



과거의 정보를 얼마나 잊을지 결정하는 역할

LSTM

LSTM의 구조



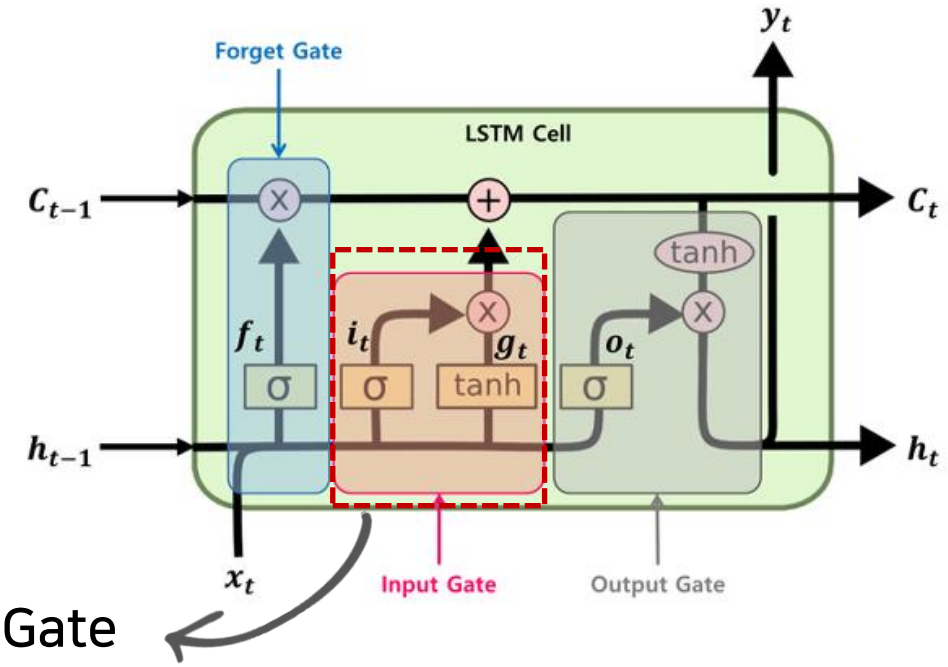
Input Gate

이전 시점의 $h^{(t-1)}$와 현재 시점의 입력 $x^{(t)}$를

가중치 W_i와 곱해 시그모이드 통과시켜 $i^{(t)}$로 변환

LSTM

LSTM의 구조

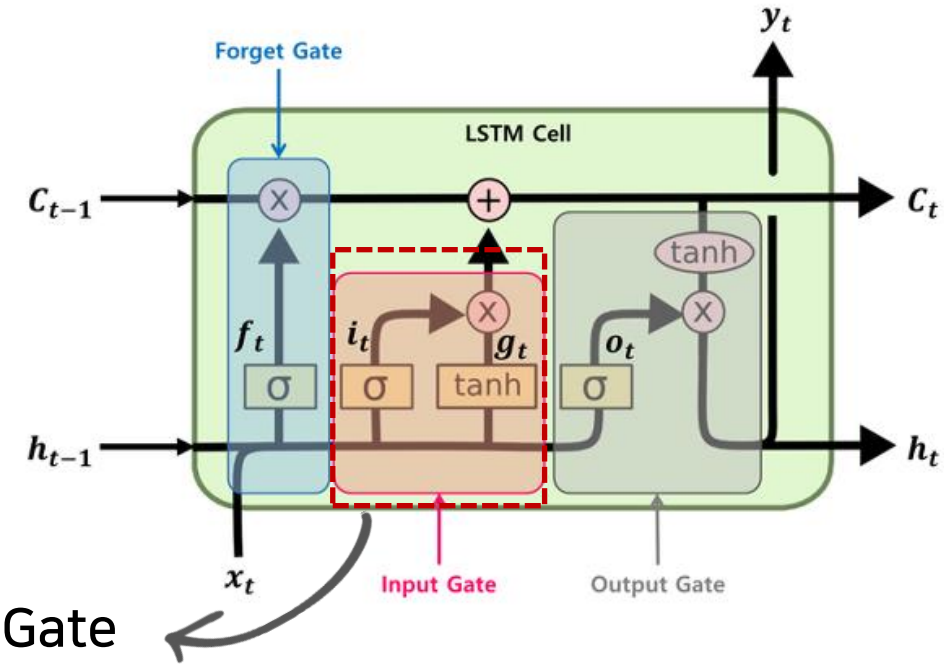


Input Gate

현재의 정보를 얼마나 기억할지 결정하는 역할

LSTM

LSTM의 구조



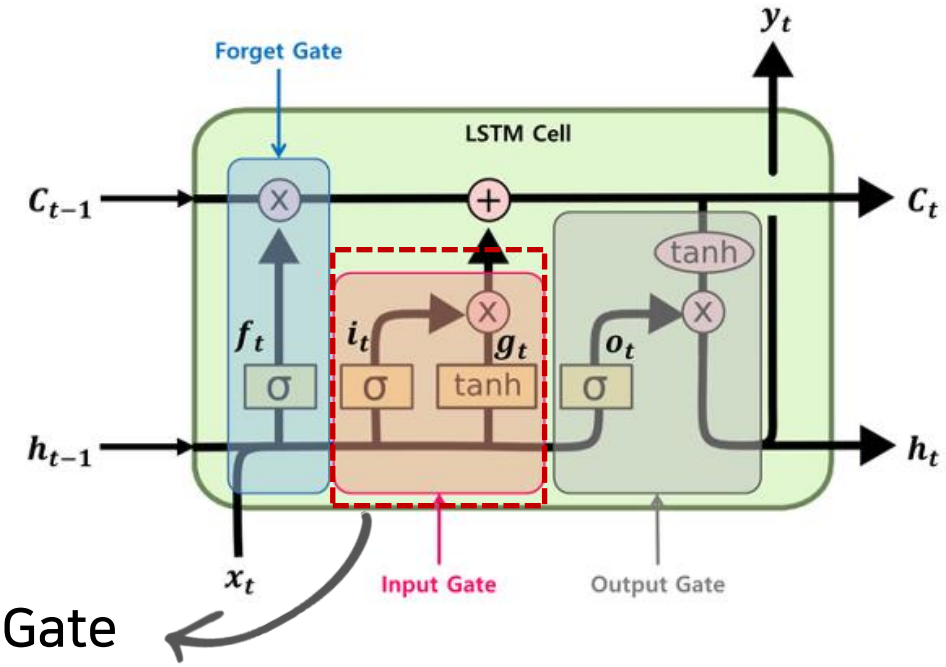
Input Gate

이전 시점의 $h^{(t-1)}$와 현재 시점의 입력 $x^{(t)}$를

가중치 W_g와 곱해 $tanh$ 통과시켜 $g^{(t)}$로 변환

LSTM

LSTM의 구조

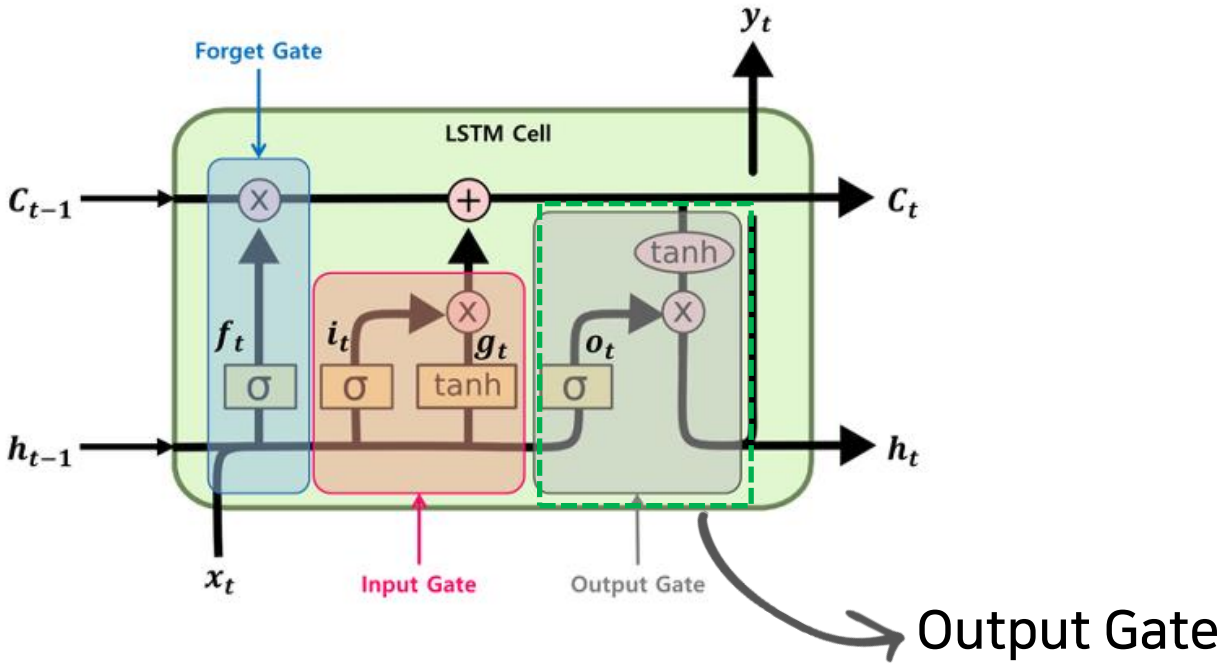


Input Gate

현재의 정보 중 어떤 정보를 Cell State에 전달할지 결정

LSTM

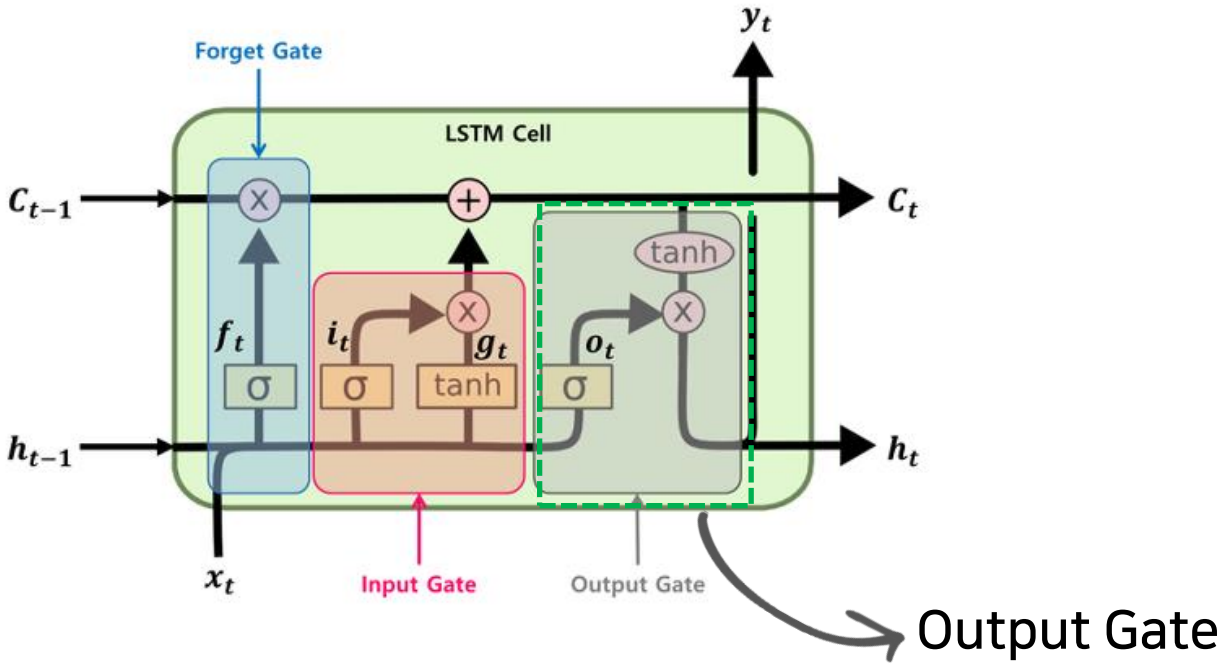
LSTM의 구조



현재 시점의 Cell State와 이전 시점의 정보들을 바탕으로
현재 시점의 hidden state 결정

LSTM

LSTM의 구조



현재까지의 정보들 중

어떤 정보들을 얼마나 활용할 것인지 결정하는 역할

LSTM

Gate의 수식과 의미

$$f_t = \sigma(W_f \times [h^{(t-1)}, x^{(t)}] + b_f)$$

$$o_t = \sigma\left(W_o \times [h^{(t-1)}, x^{(t)}] + b_o\right)$$

$$i_t = \sigma(W_i \times [h^{(t-1)}, x^{(t)}] + b_i)$$

Forget Gate: 과거의 정보를 얼마나 잊을지 결정

Input Gate: 현재의 정보를 얼마나 기억할지 결정

Output Gate: 현재까지의 정보들 중 어떤 정보들을 얼마나 활용할 것인지 결정

LSTM

Gate의 수식과 의미

$$f_t = \sigma(W_f \times h^{(t-1)}, x^{(t)} + b_f)$$

$$o_t = \sigma(W_o \times h^{(t-1)}, x^{(t)} + b_o)$$

얼마나의 결정은

0~1 사이의 출력값을 갖는

시그모이드 함수를 통해 반영

(일종의 반영 비율)

Forget Gate: 과거의 정보를 얼마나 잊을지 결정

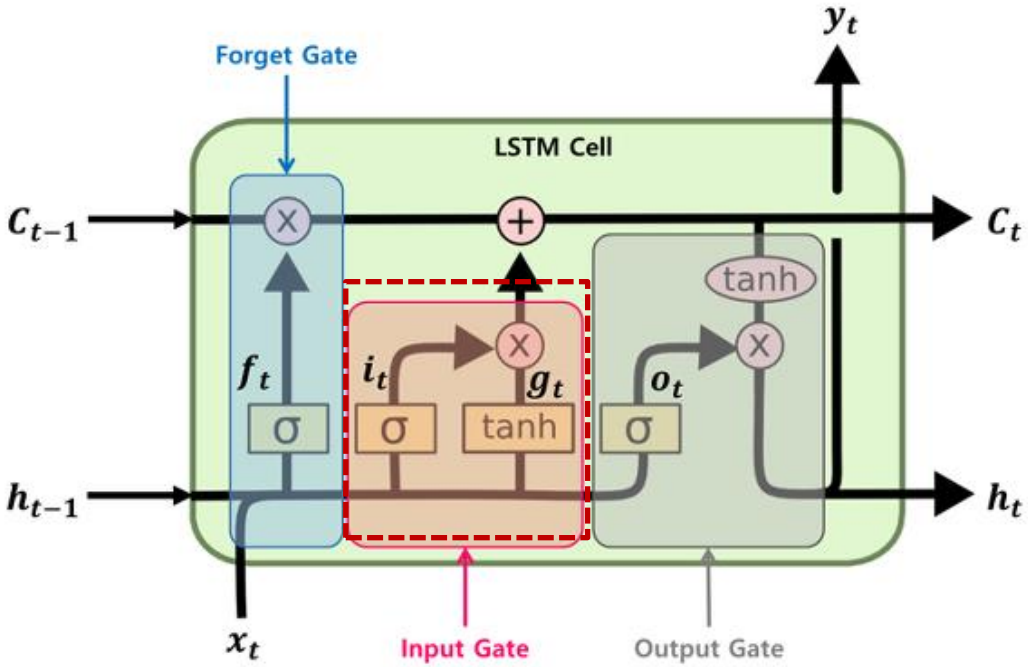
Input Gate: 현재의 정보를 얼마나 기억할지 결정

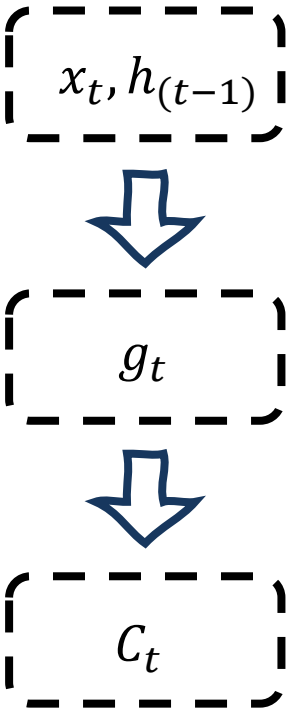
Output Gate: 현재까지의 정보들 중

어떤 정보들을 얼마나 활용할 것인지 결정

LSTM

작동 과정





i_t : 현재의 정보를 얼마나 기억할지 결정

g_t : 현재의 정보 중 어떤 정보를 Cell State에 전달할 것인지

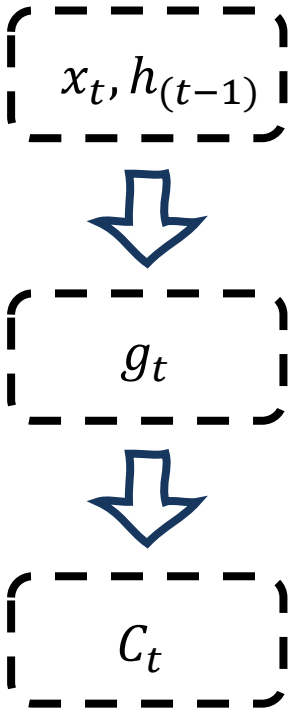
LSTM

작동 과정

$$g^t = tanh(W_h \times [h_{t-1}, x_t] + b_i)$$

'어떤' 정보를 표현하기 위해서

$tanh$ 함수를 사용



i_t : 현재의 정보를 얼마나 기억할지 결정

g_t : 현재의 정보 중 어떤 정보를 Cell State에 전달할 것인지

LSTM

작동 과정

이전 시점의 Hidden State와 현 시점의 입력을 바탕으로 현 시점 Cell State 정의

C_t 계산

$$C_t = f_t \times C_{t-1} + i_t \times g_t$$

Forget Gate와 Input Gate의 출력을 바탕으로

현 시점 Cell State의 값 결정

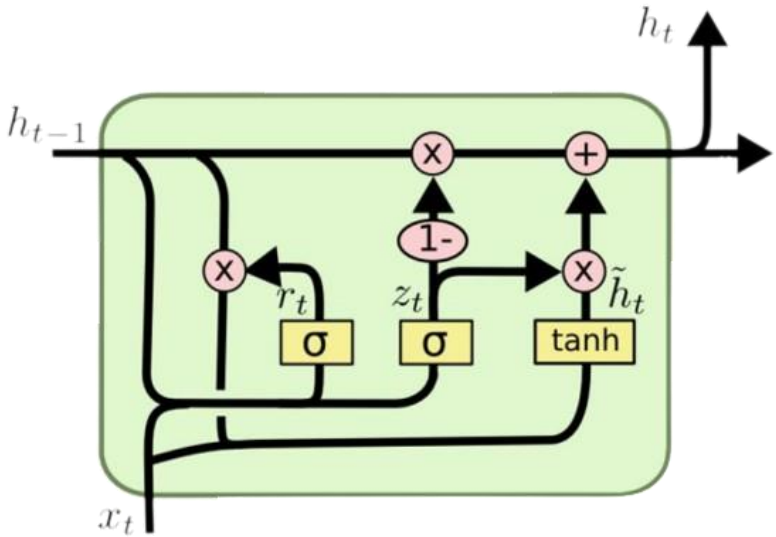
h_t 계산

$$h_t = o_t \times \tanh(C_t)$$

정의된 Cell State(C_t)와 이전 시점의 정보(o_t)들을 활용하여

현 시점의 Output이자 현 시점의 Hidden State를 결정

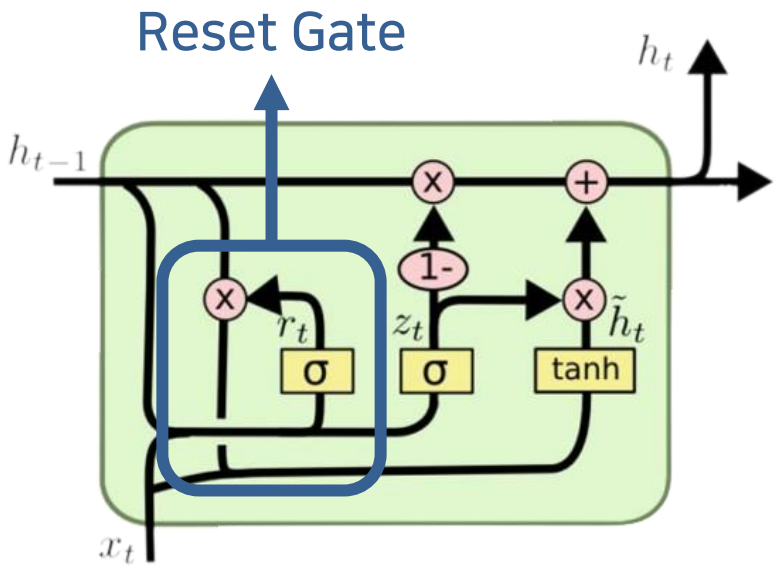
GRU (Gated Reccurent Unit)



LSTM의 구조를 간소화한 모델

Cell State가 없어지고 Gate의 구조가 변경됨

GRU의 구조



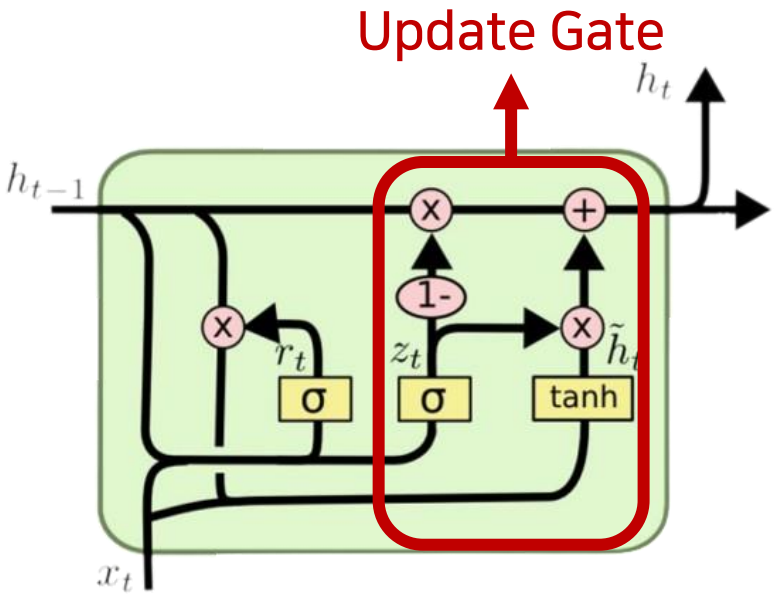
두벡터의 concatenation

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

r_t: 리셋 게이트

이전 시점의 정보를 얼마나 유지할지 결정

GRU의 구조



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Forget Gate와 Input Gate를 합친 개념

z_t: 업데이트 게이트

이전 시점과 현재 시점의 Hidden State의 반영 비율 결정

GRU의 구조

LSTM에 비해 파라미터가 적음

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

연산 속도가 더 빠름!

성능면에서 큰 차이 없음

Forget Gate와 Input Gate를 합친 개념

z_t: 업데이트 게이트

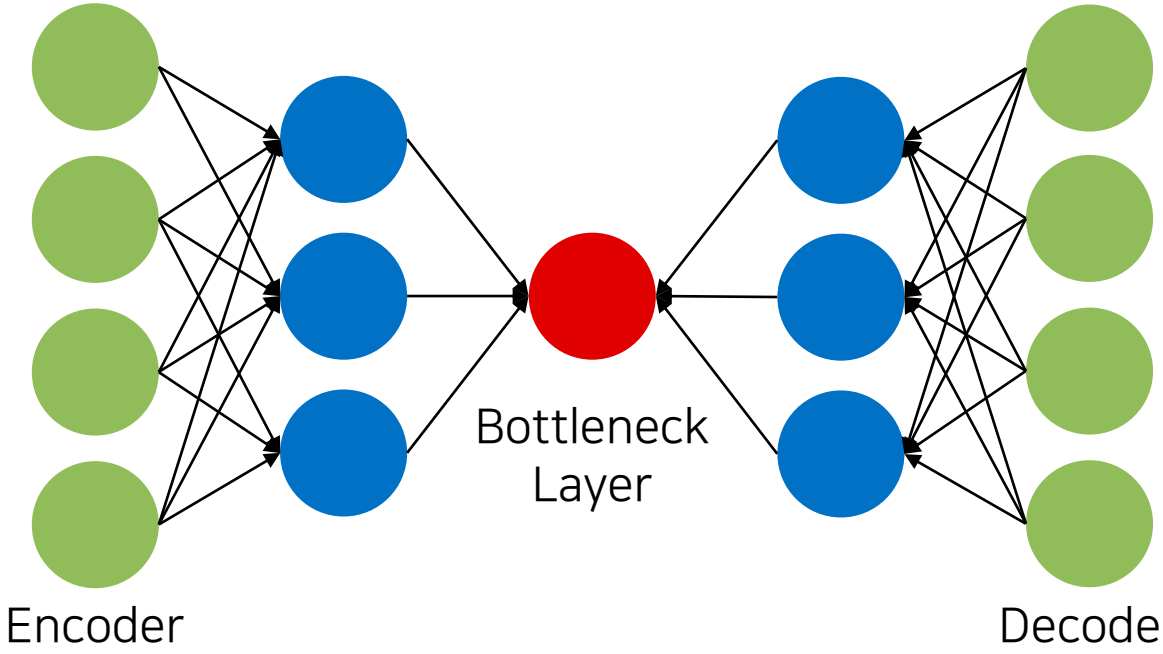
이전 시점과 현재 시점의 Hidden State의 반영 비율 결정

3

RNN 모델의 응용

RNN모델의 응용

Encoder와 Decoder



Encoder

인풋데이터를
압축하는 역할

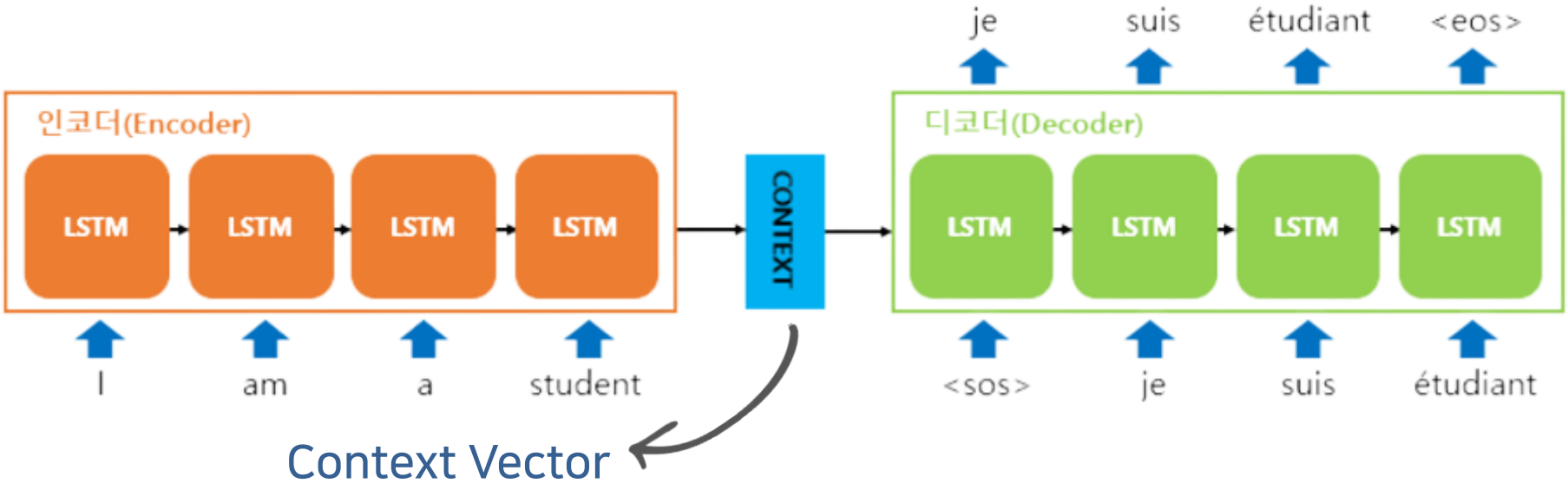
Decoder

압축된 데이터를 받아
의미있는 정보로 변환하는 역할

RNN모델의 응용

Seq2Seq

Encoder/Decoder에 RNN 적용



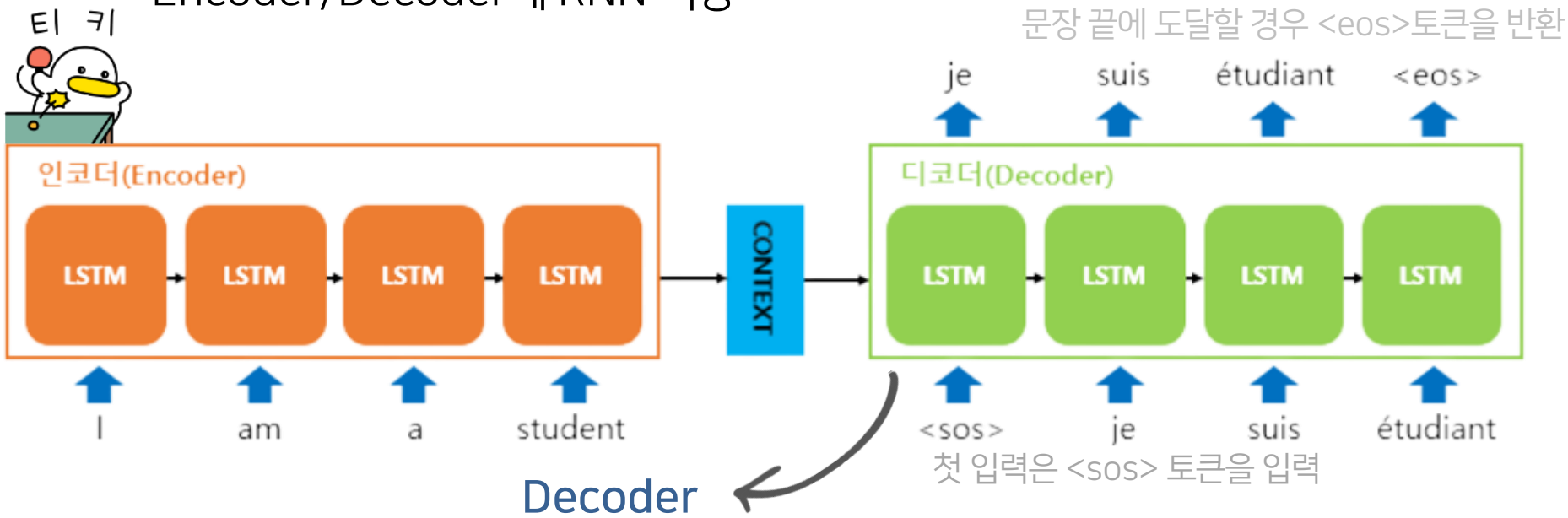
LSTM 마지막 셀의 Hidden State

입력 문장의 문맥 정보를 담고 있음

RNN모델의 응용

Seq2Seq

Encoder/Decoder에 RNN 적용



Context Vector를 입력 받아 매 시점마다 예측 단어 출력

이전 시점의 출력을 입력으로 사용

Seq2Seq

Decoder가 예측을 잘못하게 될 경우

전체의 학습 과정이 저해됨

교사 강요
(Teacher Forcing)

Decoder의 입력에 실제 정답 사용

이전 시점의 출력을 입력으로 사용

RNN모델의 응용

Seq2Seq의 문제점

Encoder에서 문장의 의미를
하나의 압축된 벡터를 사용하므로 두 가지 문제 발생

병목현상(Bottleneck Problem)

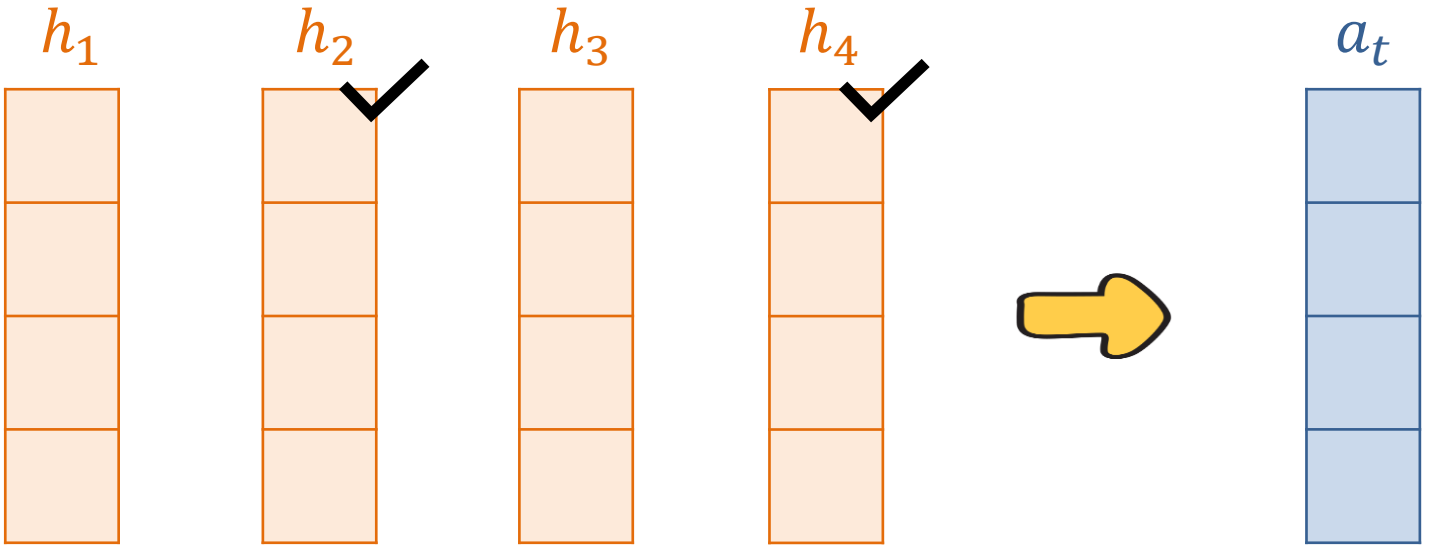
전체 입력 데이터를
고정된 길이의 벡터로 압축하여
정보의 손실 발생

Gradient Vanishing Problem

RNN에서 Cell이 늘어날수록
기울기 소실 문제 심화

Attention

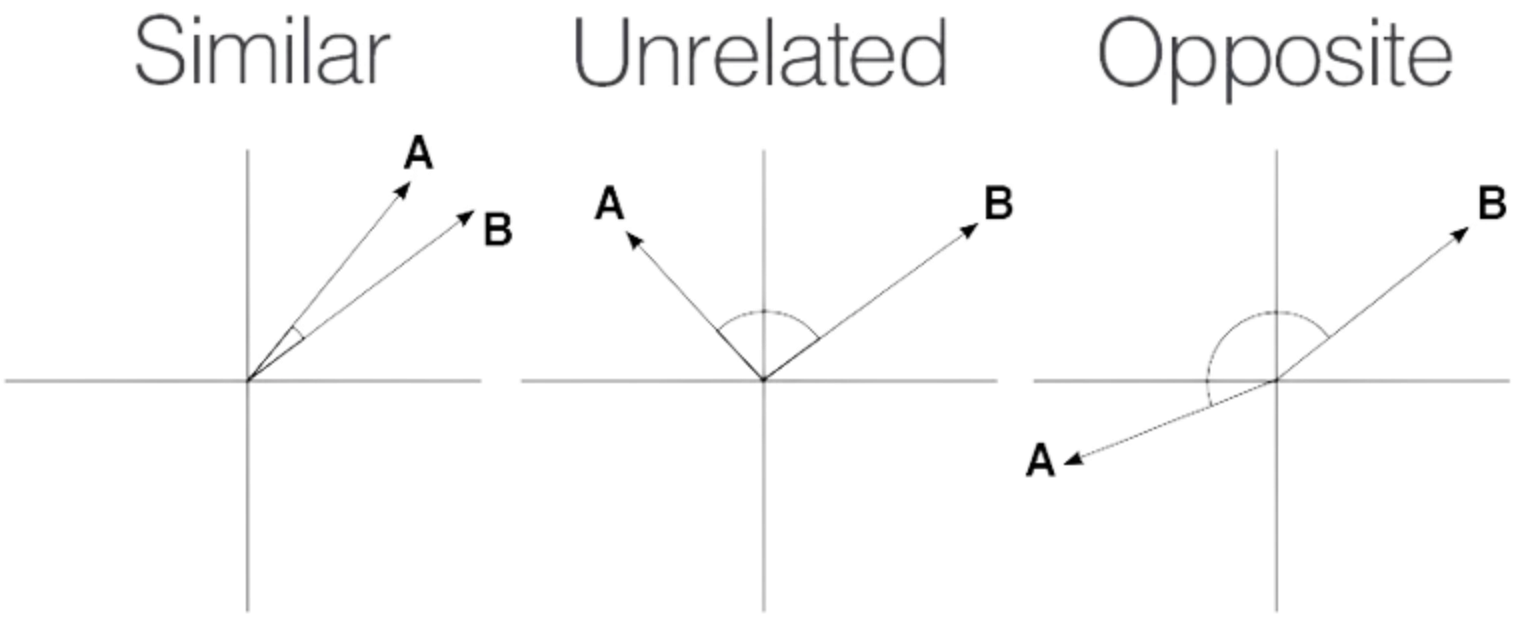
Dotproduct Attention



어떤 Hidden State에 집중해야 하는가를 반영

Attention

Dotproduct Attention



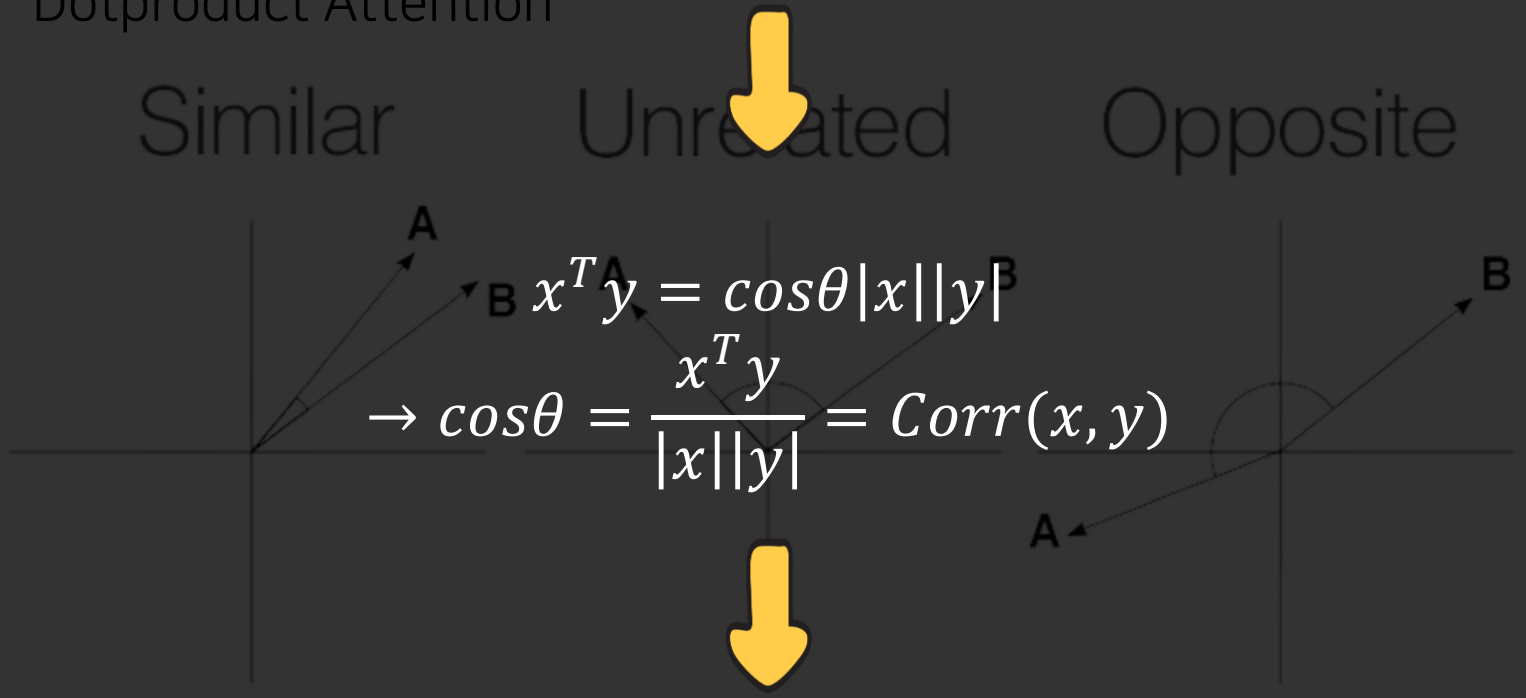
코사인 유사도를 통해 집중의 정도 측정

즉, 두 벡터가 얼마나 유사한가를 계산

Attention

Dotproduct Attention

$$x \cdot y = x^T y = cos\theta |x||y|$$



$$x^T y = cos\theta |x||y|$$

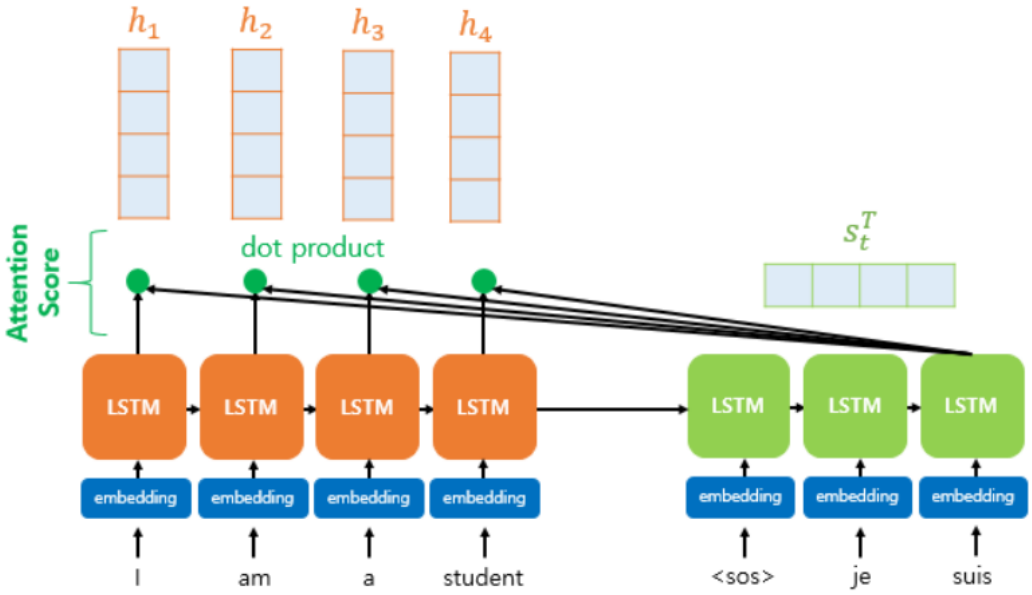
$$\rightarrow cos\theta = \frac{x^T y}{|x||y|} = Corr(x, y)$$

두 백터 사이각의 코사인은

상관관계와 같다

Attention의 진행과정

Attention Score 계산



h_i: Encoder의 각 시점의 Hidden State

s_t: Decoder의 Hidden State

$$score(s_t, h_i) = s_t^T h_i$$

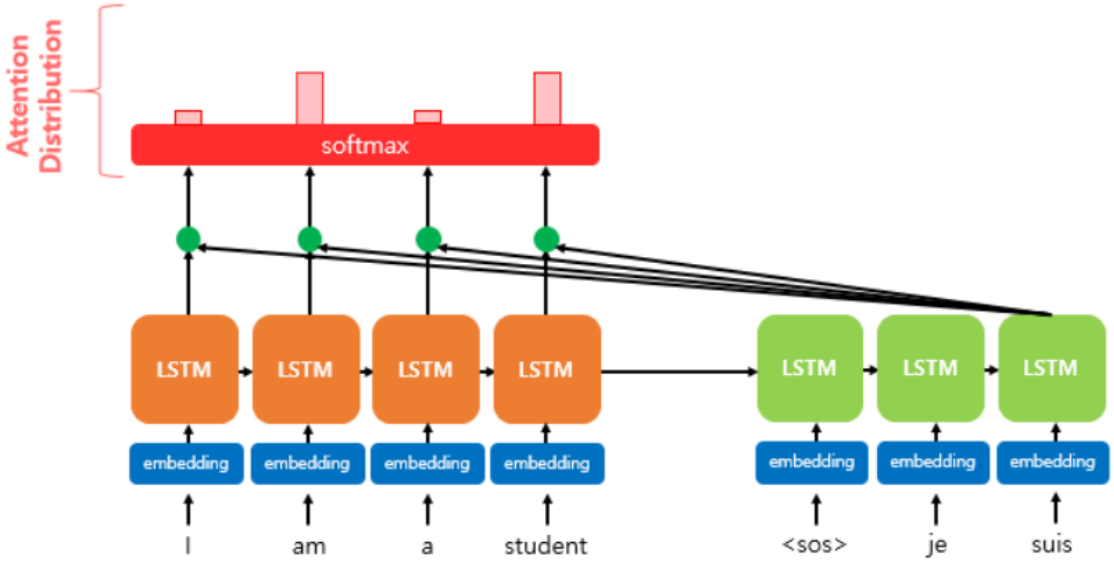
$$e_t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_n]$$

Decoder 매 시점 마다의 Hidden State와

Encoder의 Hidden State간의 내적값 계산

Attention의 진행과정

Attention Score 계산



e_t: Attention Score

$$e_t = [s_t^T h_1, s_t^T h_2, \ldots, s_t^T h_n]$$

$$a_t = softmax(e^t)$$

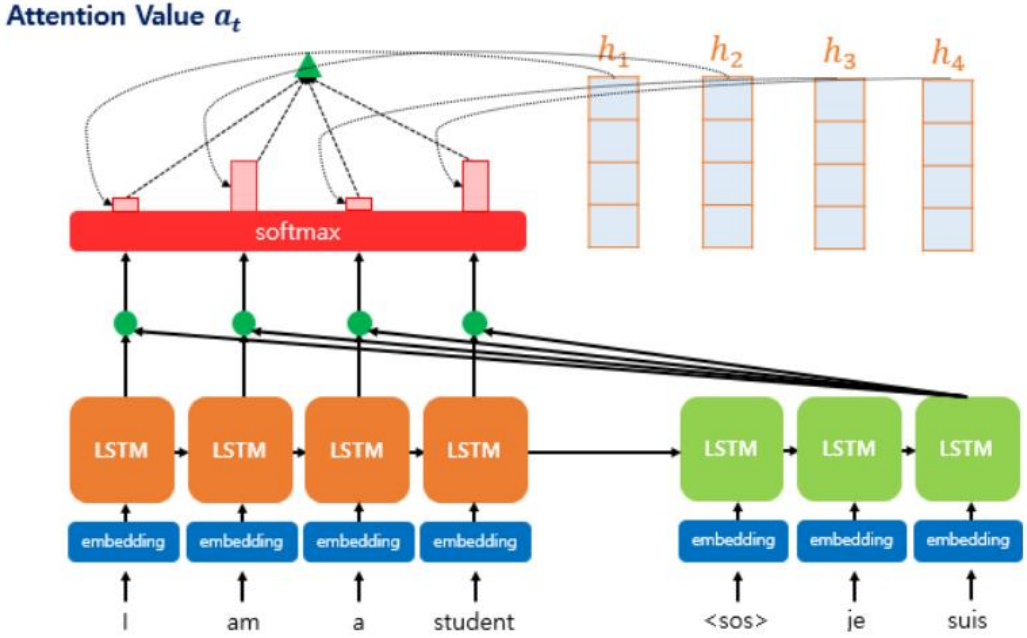
Attention Score를 Softmax함수에 통과시켜

정규화된 확률을 구함

RNN모델의 응용

Attention의 진행과정

Attention Score 계산



e_t: Attention Score

$$a_t = softmax(e_t)$$

h_i: Encoder의 각 시점의 Hidden State

$$\alpha_t = a_t \cdot h = \sum_{i=1}^{n} a_{ti} h_i$$

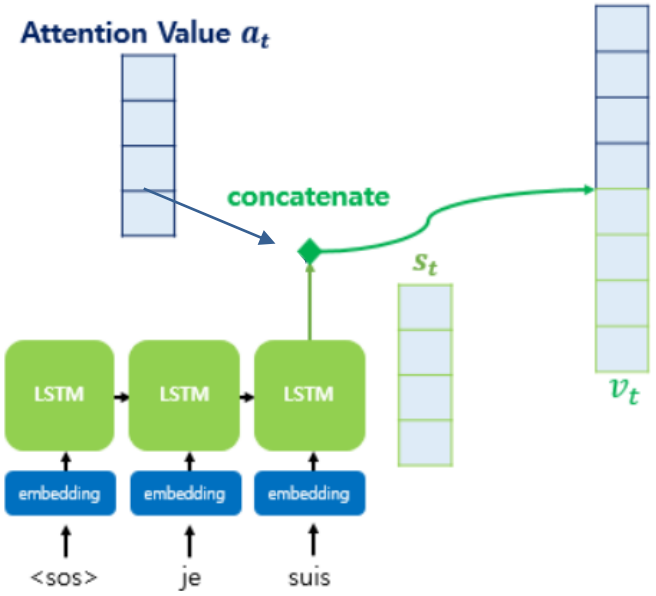
$$where\ h = [h_1\ h_2\ \cdots\ h_n]$$

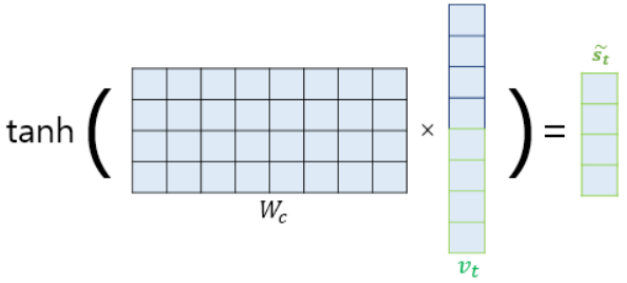
Attention Distribution을 가중치로

각 Hidden State와 가중합하여 Attention Value 계산

Attention의 진행과정

Concatenation





$$v_t = [\alpha_t, s_t]$$

$$\widetilde{s_t} = \tanh(W_c \cdot v_t + b_c)$$

$$o_t = softmax(W_o \cdot \widetilde{s_t} + b_o)$$

Attention Value와 Decoder의 Hidden State를 연결

이 벡터를 통해 정답 예측

4 마무리

주제분석
가보자고~

THANK YOU