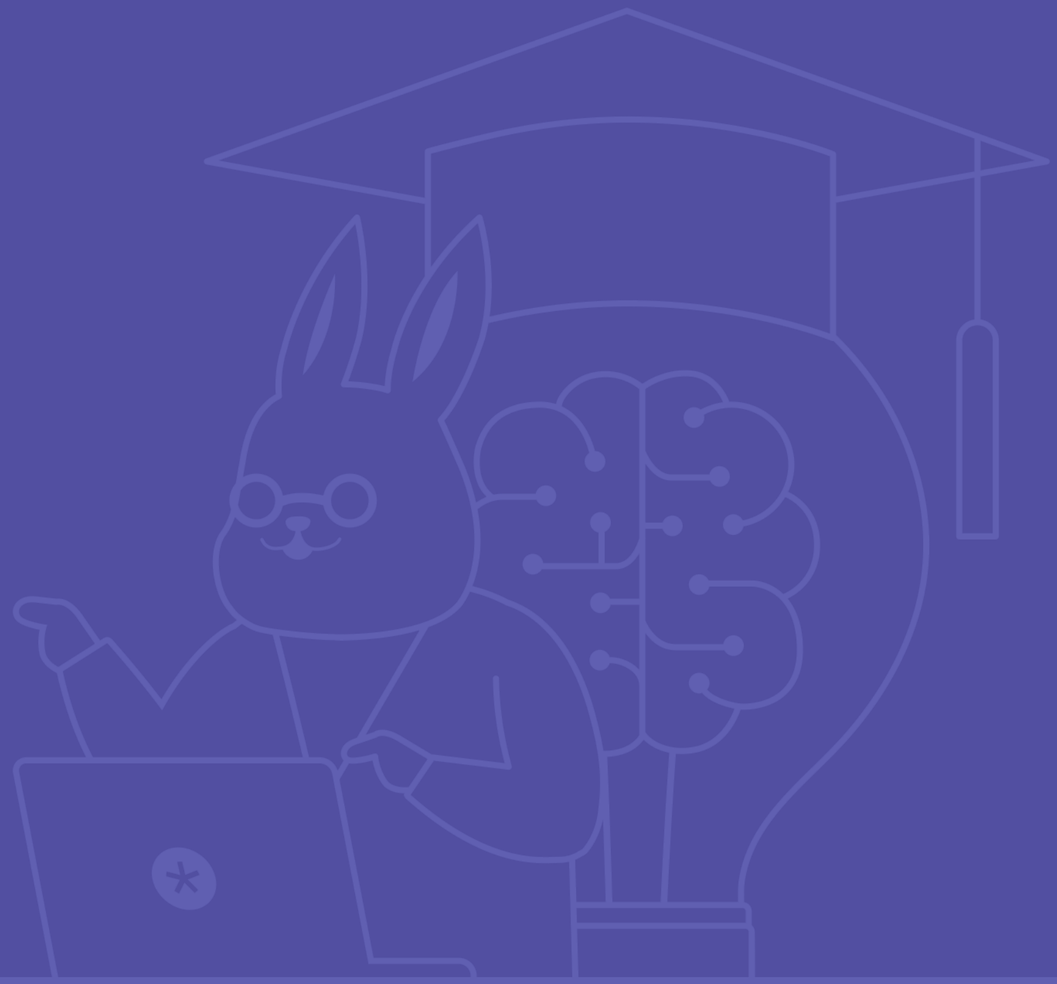
자연어 처리를 위한 딥러닝2

2장 Text Modeling

JunnG_T



/* elice */

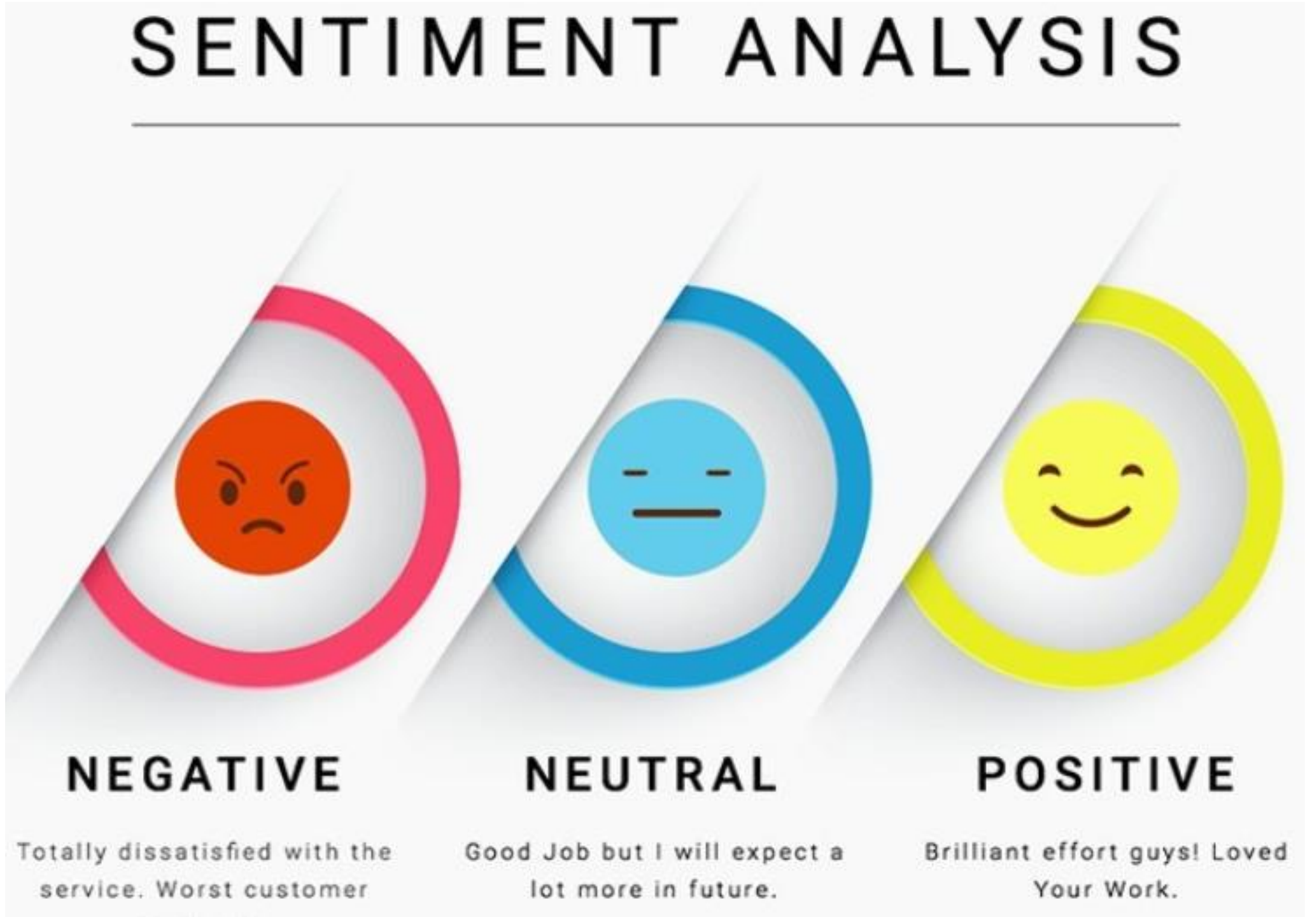
Contents

01

Document Classification

Document Classification Example

감성분석(Sentiment Analysis) : Document 긍/부정 분석 -> Classification Task



01 Document Classification

Document Classification Example

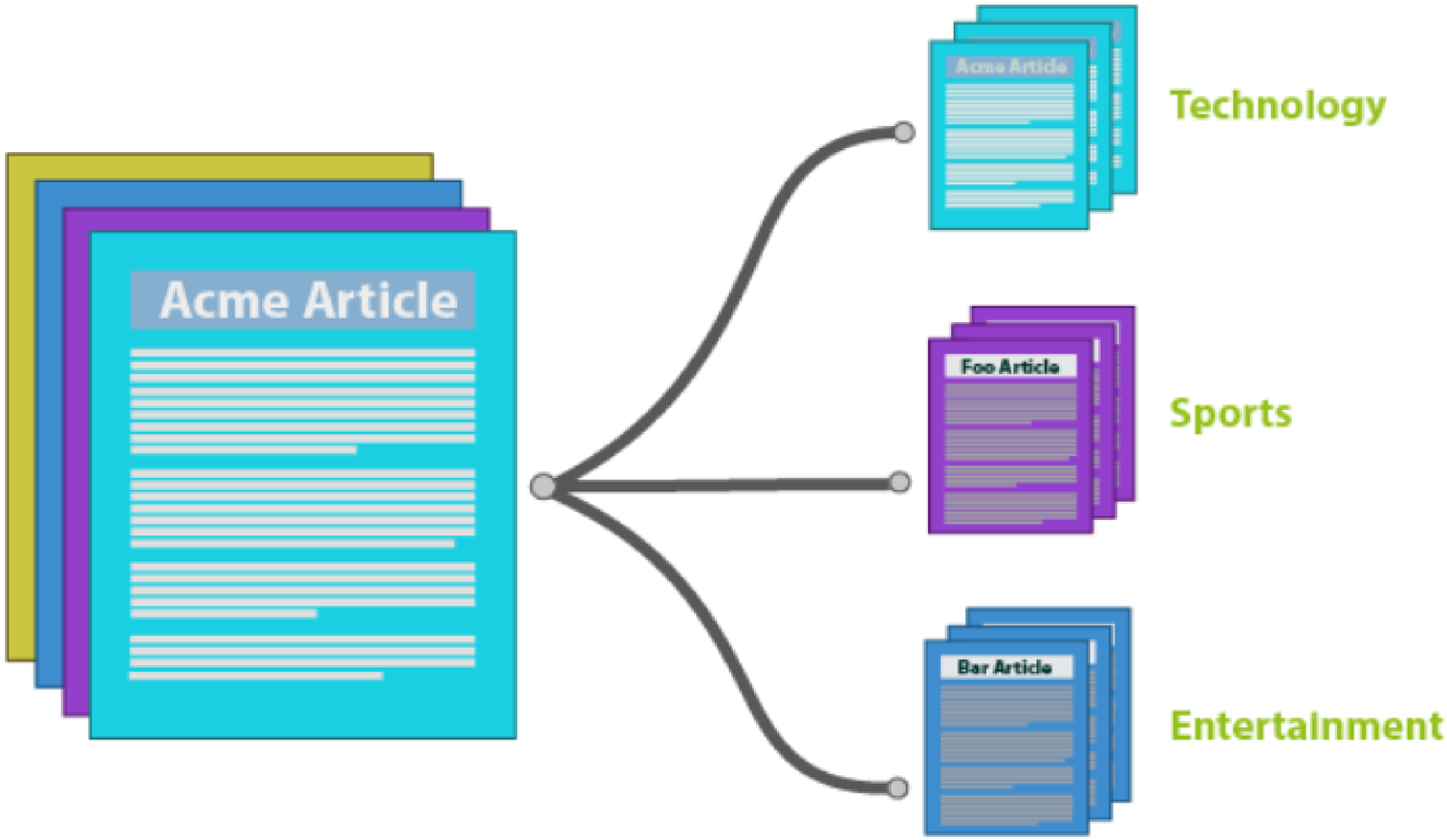
Spam Mail Detection

코멘토	사용자설정	면접에서 대답할만한 직무 관련 경험을 쌓을 수 있는 방법
티몬 1212타임	사용자설정	(광고) 국산 3중 마스크 반값 할인!
청약당첨	시스템차단	(광고)분양즉시 프리미엄3000만원인 오피스텔?
티몬 티몬블랙딜	사용자설정	(광고) 오쿡 닭가슴살 3+2 3,900원!
코멘토	사용자설정	혹시, 인턴하면서 실무보다는 자료 정리만 하셨나요?
티몬 무료배송데…	사용자설정	(광고) 단 하루의 찬스!
코멘토	사용자설정	LG 유플러스에서 MULTILAYER 멘티님께 인터뷰를 요청합니다!
따뜻한하루	사용자설정	삶은 밥의 역사다
티몬 티몬블랙딜	사용자설정	(광고) KC인증 마스크 100매 8,900원!
링커리어	사용자설정	(광고)SK, 롯데 서포터즈 외 핫한 정보
빅데이터 러닝센터	시스템차단	마케터를 위한 웹크롤링 (R for Business Insight) 교육 개설 안내 (8/13~14) [빅데이터 러닝센터] (AD)
코멘토	사용자설정	혹시, 상반기에 자소서만 정말 많이 써봤나요?
티몬 1212타임	사용자설정	(광고) CJ제일제당 BEST 간식 플렉스!
티몬 티몬블랙딜	사용자설정	(광고) 스파오 최대 90% 빅세일!
Walgreens.com	사용자설정	Extra 20% off your self-care routine. You deserve a break
링커리어	사용자설정	❤이준걸님! 8,000명에게 540만원 지급하는 인턴 어때요?
빅데이터러닝센터	시스템차단	[빅데이터 러닝센터] 3일만에 익히는 R 기초통계분석 교육 개설 안내 (8/10~12) (AD)
코멘토	사용자설정	MULTILAYER 멘티님, 시간은 없는데 인턴 경험은 하고 싶으신가요?
티몬 히트상품	사용자설정	(광고) 3차 완판 녹물제거 필터샤워기!
따뜻한하루	사용자설정	진정한 보배
티몬 티몬블랙딜	사용자설정	(광고) 반찬 걱정 끝! 수제떡갈비 특가
[CJ ONE]	사용자설정	(광고)[CJ ONE] 이준걸님, 금주 추천 이벤트를 확인하세요!

/* elice */

Document Classification Example

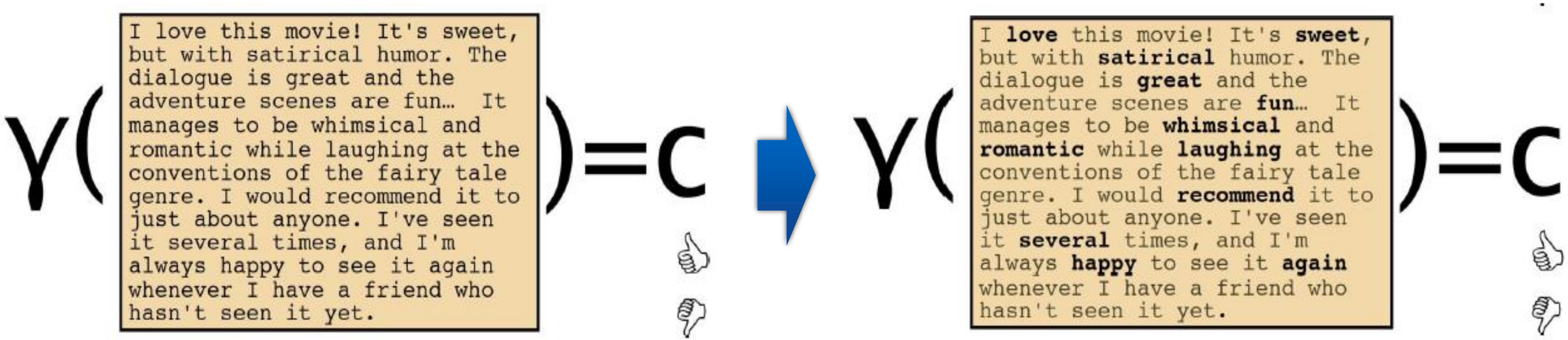
Article Topic Classification



01 Document Classification

Bag-of-Words Embedding with Classification

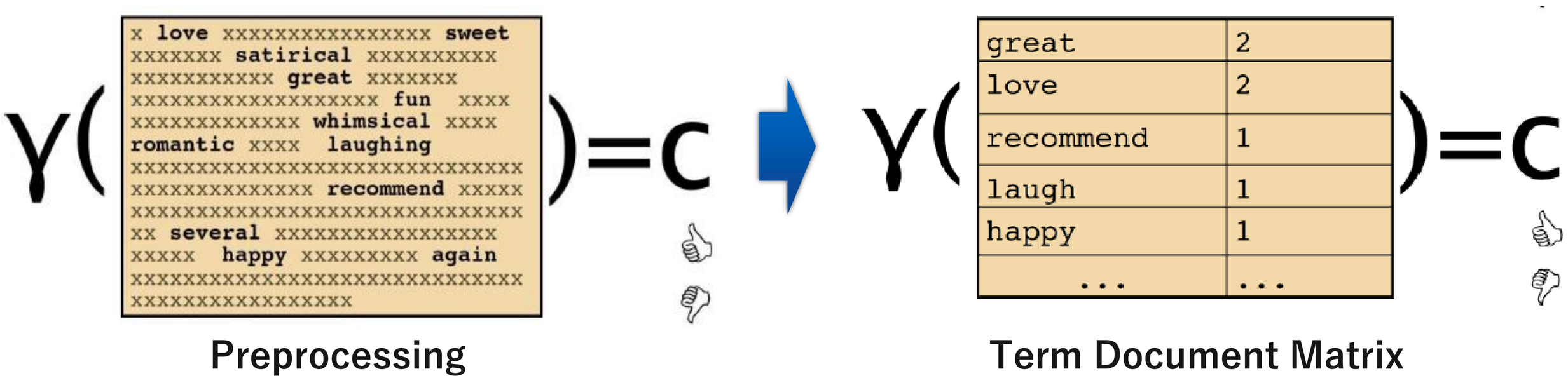
Article Topic Classification



Raw Text **Preprocessing**

Bag-of-Words Embedding with Classification

Article Topic Classification



Bayes's Rule

$$P(C_i|x_1, x_2) = \frac{P(x_1, x_2|C_i) * P(C_i)}{P(x_1, x_2)}$$

$$P(C_i|x_1, x_2) = \frac{P(x_1|C_i) * P(x_2|C_i) * P(C_i)}{P(x_1, x_2)}$$

Maximum likelihood estimates

✓ Frequency 기반

$$\hat{P}(c_j) = \frac{N.Doc(C = c_j)}{Total\ number\ of\ documents}$$

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{W \in V} count(w, c_j)}$$

c_j는 Class를 뜻함.

w_i는 전체 단어의 빈도

Maximum likelihood estimates

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{W \in V} count(w, c_j)}$$

- ✓ 영화 리뷰 긍/부정 분류 예시
- ✓ Review 1:This movie was awesome! I really enjoyed it.
- ✓ Review 2:This movie was boring and waste of time.

STEP 1 : 각 Class의 Conditional Probability를 구하자

Words	P(Word/Positive)	P(Word/Negative)
This	0.1	0.1
Movie	0.1	0.1
Was	0.1	0.1
Awesome	0.4	0.01
I	0.2	0.2
Really	0.3	0.05
enjoyed	0.5	0.05
It	0.1	0.1
Boring	0.02	0.3
And	0.1	0.1
Waste	0.02	0.35
Of	0.02	0.02
Time	0.15	0.15

/* elice */

Maximum likelihood estimates

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{W \in V} count(w, c_j)}$$

- ✓ 영화 리뷰 긍/부정 분류 예시
- ✓ Review 1:This movie was awesome! I really enjoyed it.

Words	P(Word/Positive)	P(Word/Negative)
This	0.1	0.1
Movie	0.1	0.1
Was	0.1	0.1
Awesome	0.4	0.01
I	0.2	0.2
Really	0.3	0.05
enjoyed	0.5	0.05
It	0.1	0.1
Boring	0.02	0.3
And	0.1	0.1
Waste	0.02	0.35
Of	0.02	0.02
Time	0.15	0.15

/* elice */

Maximum likelihood estimates

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

- ✓ 영화 리뷰 긍/부정 분류 예시
- ✓ Review 2:This movie was boring and waste of time.

Words	P(Word/Positive)	P(Word/Negative)
This	0.1	0.1
Movie	0.1	0.1
Was	0.1	0.1
Awesome	0.4	0.01
I	0.2	0.2
Really	0.3	0.05
enjoyed	0.5	0.05
It	0.1	0.1
Boring	0.02	0.3
And	0.1	0.1
Waste	0.02	0.35
Of	0.02	0.02
Time	0.15	0.15

/* elice */

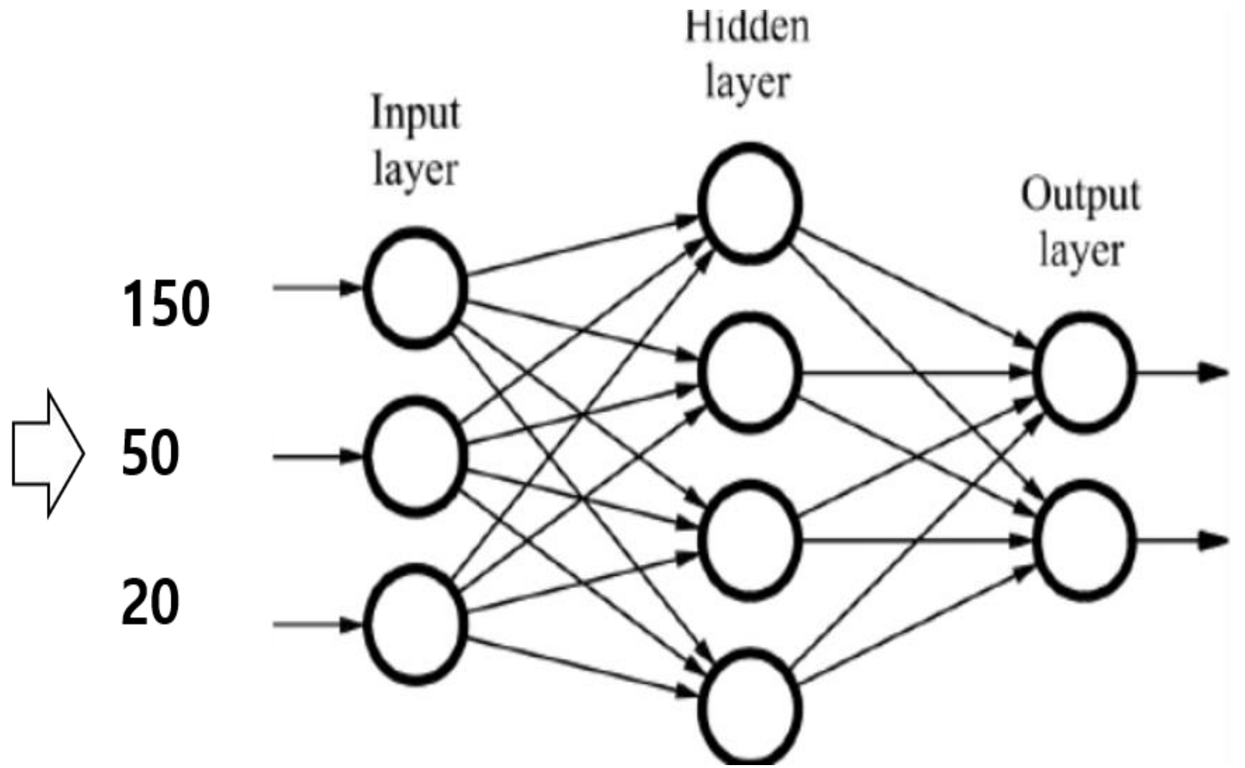
02

RNN-based

MLE with text

✓ Embedding Matrix를 통해 다음과 같이 입/출력을 받게 된다.

	키	몸무게	나이
A	150	50	20
B	160	60	22
C	170	70	24

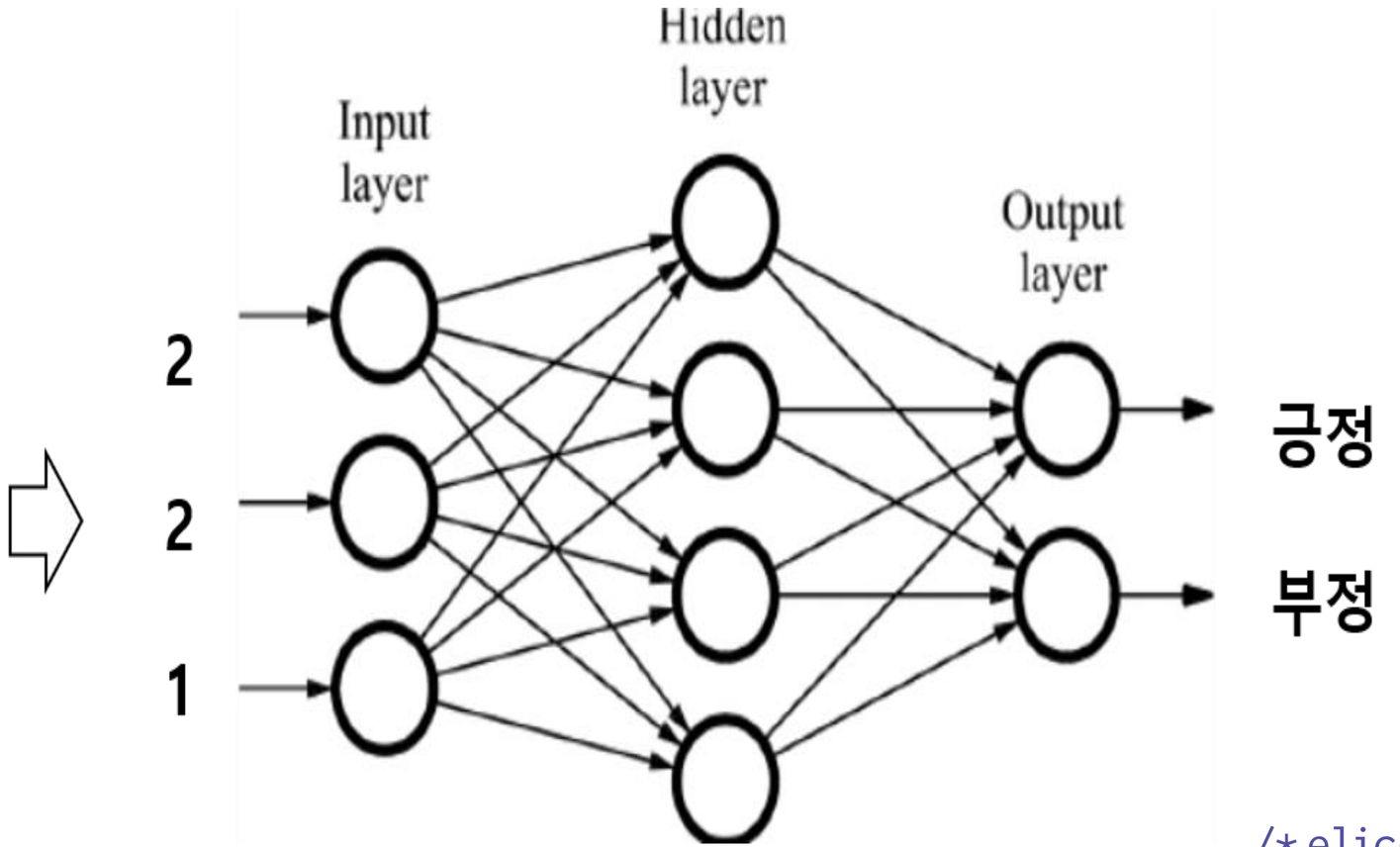


MLE with text

✓ 1. 빈도기반 문서 임베딩

*** 영화 리뷰 데이터**

	영화	최고	최악
Doc A	2	2	1
Doc B	2	0	0
Doc C	0	3	1



/* elice */

MLE with text

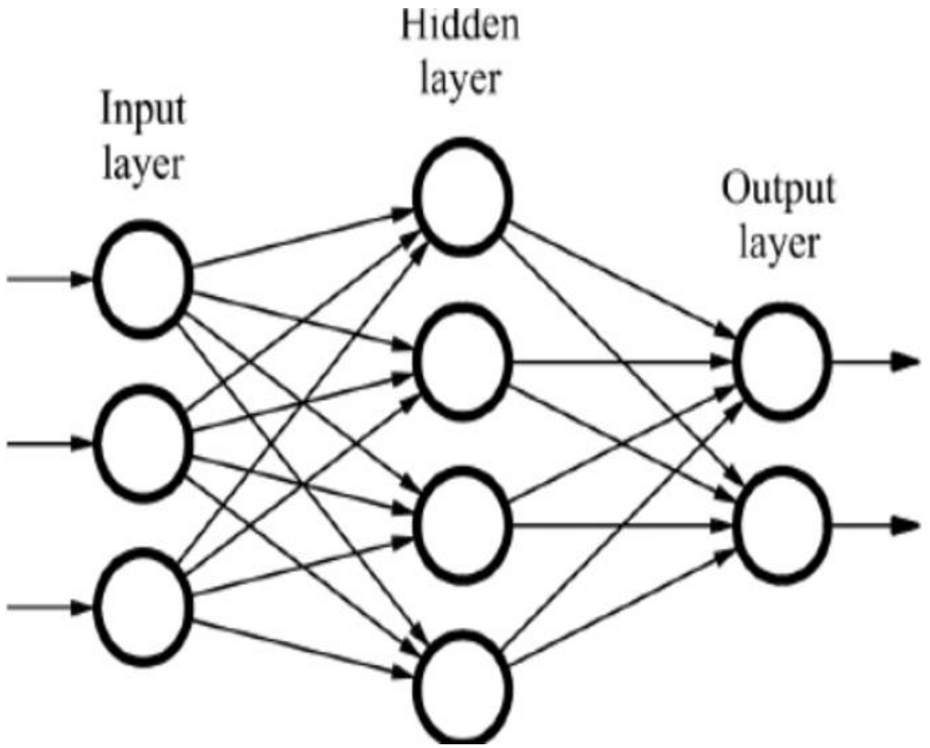
✓ 2. 원-핫 벡터 기반 문서 임베딩

문서1 : [[1 0 0 0] - 오늘
[0 0 1 0] - 밥
[0 0 0 1]] - 먹다

[1 0 0 0]

[0 0 1 0]

[0 0 0 1]



MLE with text

✓ 2. 원-핫 벡터 기반 문서 임베딩 문제점

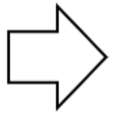
문서1 : [[1 0 0 0]
[0 0 1 0]
[0 0 0 1]]

[1 0 0 0] x_2

[0 0 1 0] x_1

[0 0 0 1] x_0

문서2 : [[0 1 0 0]
[0 0 1 0]
[1 0 0 0]
[0 0 1 0]]



[0 1 0 0] x_2

[0 0 1 0] x_1

[1 0 0 0] x_0

[0 0 1 0]

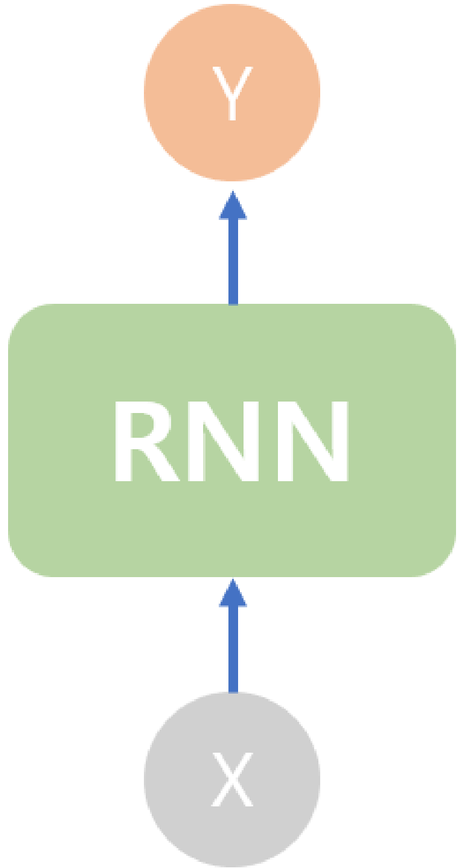
* 벡터의 행, 즉 단어 벡터의 개수가 신경망 입력 노드의 개수와 맞지 않을 수 있다.

MLE with text

- ✓ 문서 임베딩은 여러 단어의 임베딩이 모여 하나로 표현됨
- ✓ 문서 임베딩은 단어의 조합으로써 문서 간의 유사도를 판별할 수 있음.
- ✓ 대표적으로 빈도 기반 임베딩과 원-핫 벡터 기반 임베딩이 있음.
- ✓ 빈도 기반 임베딩은 단어의 순서를 파악할 수 없다는 단점이 있음.
- ✓ 원-핫 벡터 기반 임베딩은 신경망 학습을 위한 고정된 입력 노드 개수에 맞추기 어려움.

Recurrent Neural Network

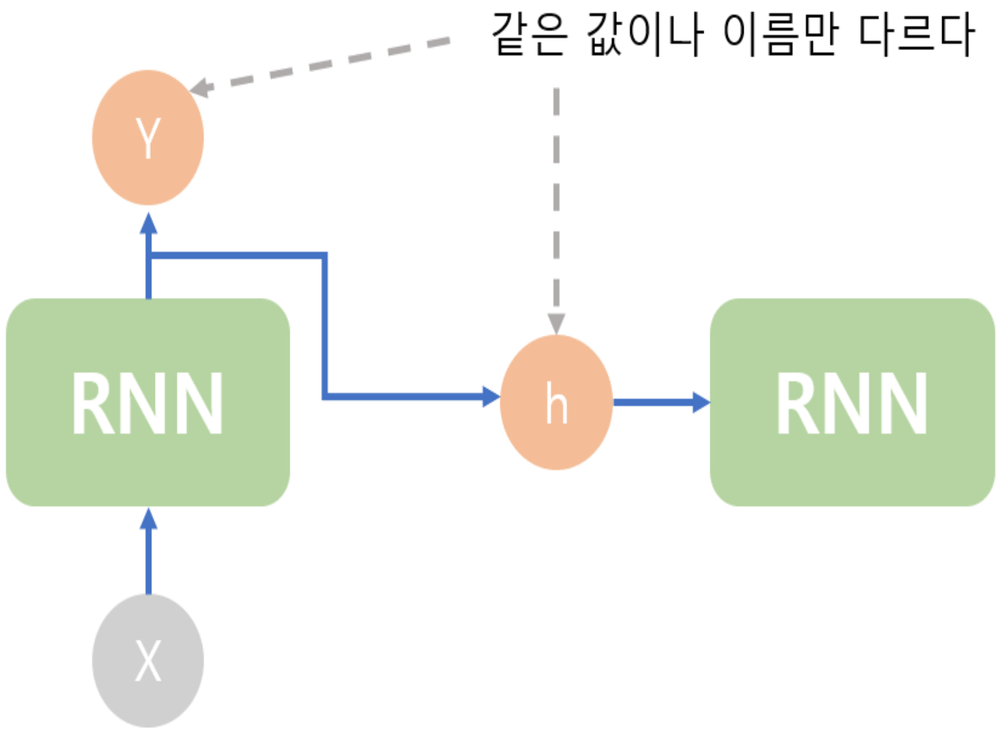
- ✓ 단순한 MLP로는 텍스트 데이터를 처리하기엔 역부족이다
- ✓ 텍스트 특성



입력 노드는 단 하나
(주로 원-핫 벡터 하나 입력)

Recurrent Neural Network

✓ RNN Input의 이해

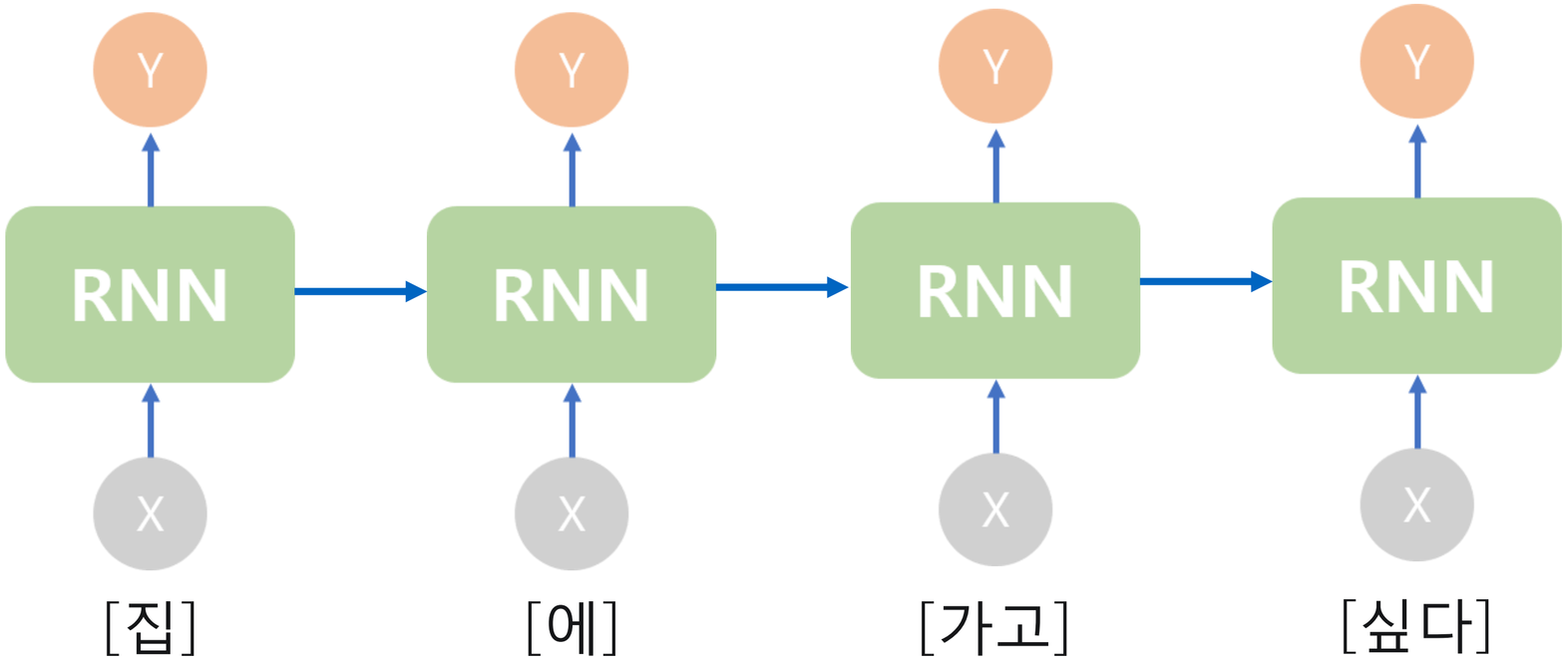


특이한 점은 출력값을 두 갈래로 나뉘어 한 쪽은 출력(Y)을,
나머지는 다음 스텝에서 RNN 자기 자신의 또 다른
입력(h)으로 사용한다
→신경망에게 '기억' 하는 기능을 부여했다

Recurrent Neural Network

- ✓ RNN의 입력을 이해하자
- ✓ Time Step : 학습할 문장을 끝까지 입력하는데 걸리는 시간

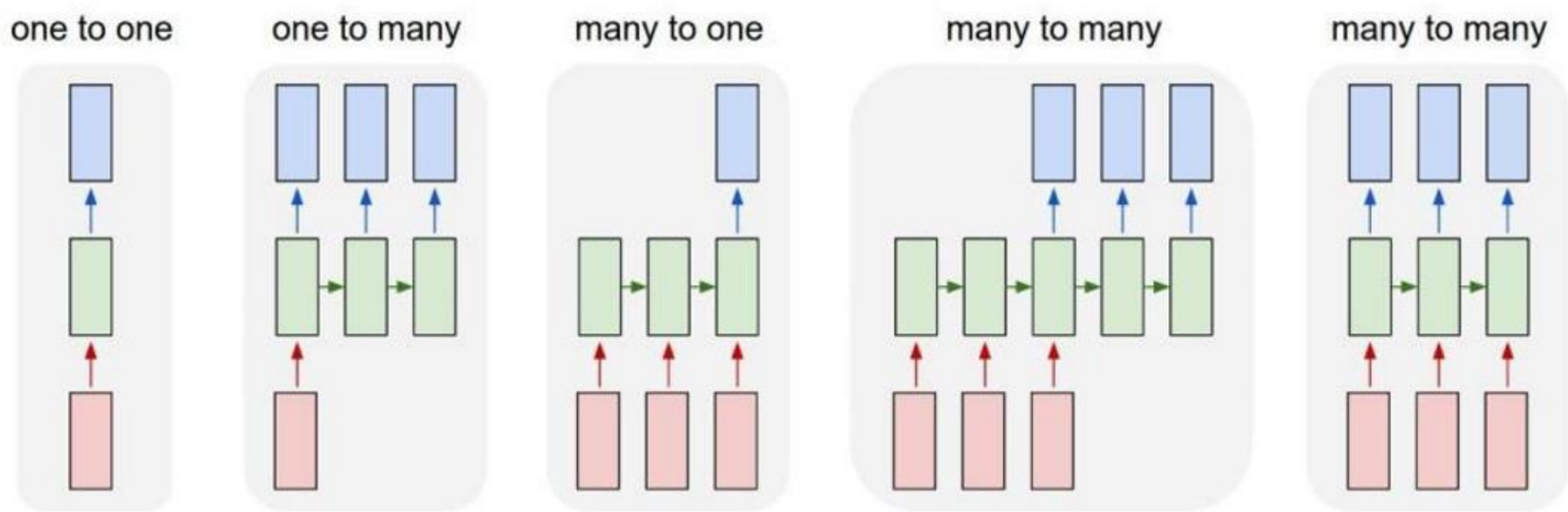
  Ex) [[집], [에], [가고], [싶다]] -> 4 Time Step



Recurrent Neural Network

✓ RNN의 출력을 이해하자
✓ 보통 출력에 따라 Task가 바뀌게 된다.

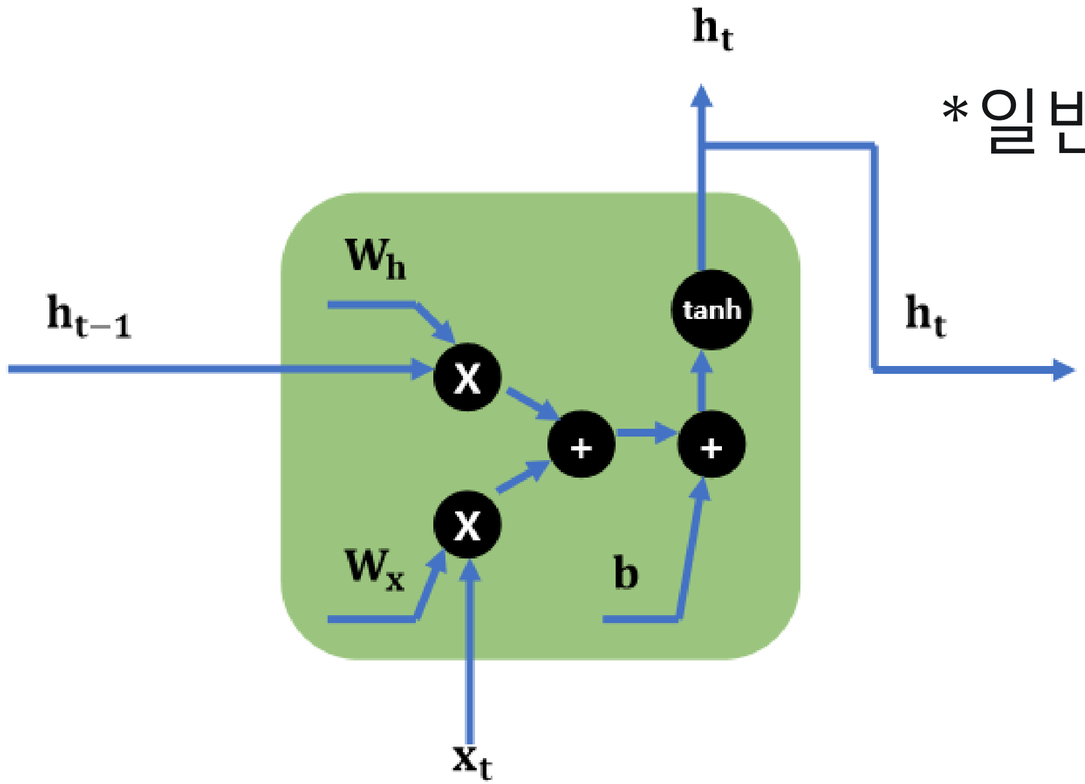
- Recurrent Neural Networks (RNN) Architecture
  - Input과 Output 개수에 따라 RNN Architecture를 구분



✓ *사용처 : Image Captioning Text Classification Machine Translation Video classification

Recurrent Neural Network

- Recurrent Neural Networks (RNN) Forward

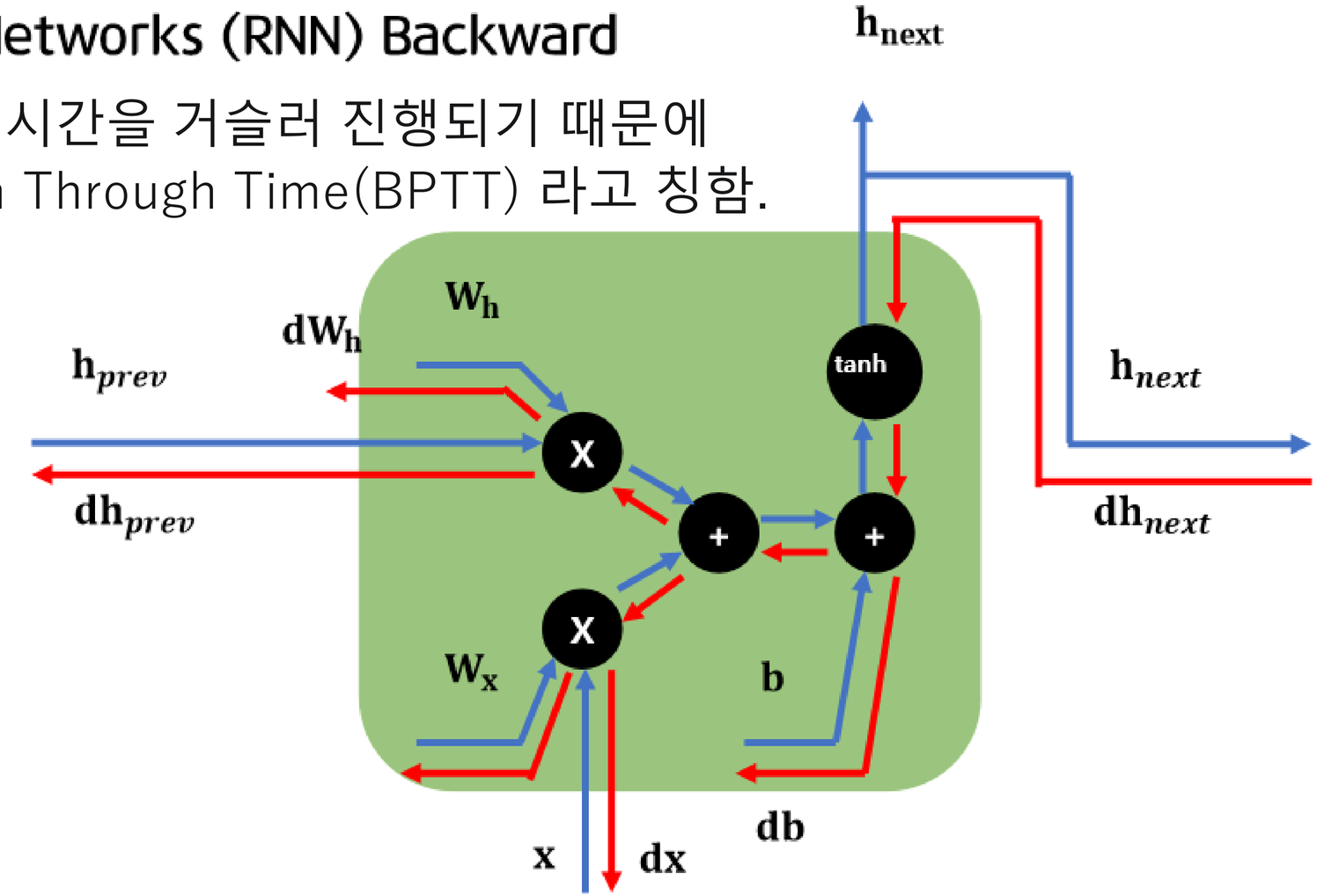


*일반적 퍼셉트론 : $f(x) = sigmoid(xW + b)$

$$\mathbf{h}_t = \tanh(\mathbf{h}_{t-1}\mathbf{W}_h + \mathbf{x}_t\mathbf{W}_x + \mathbf{b})$$

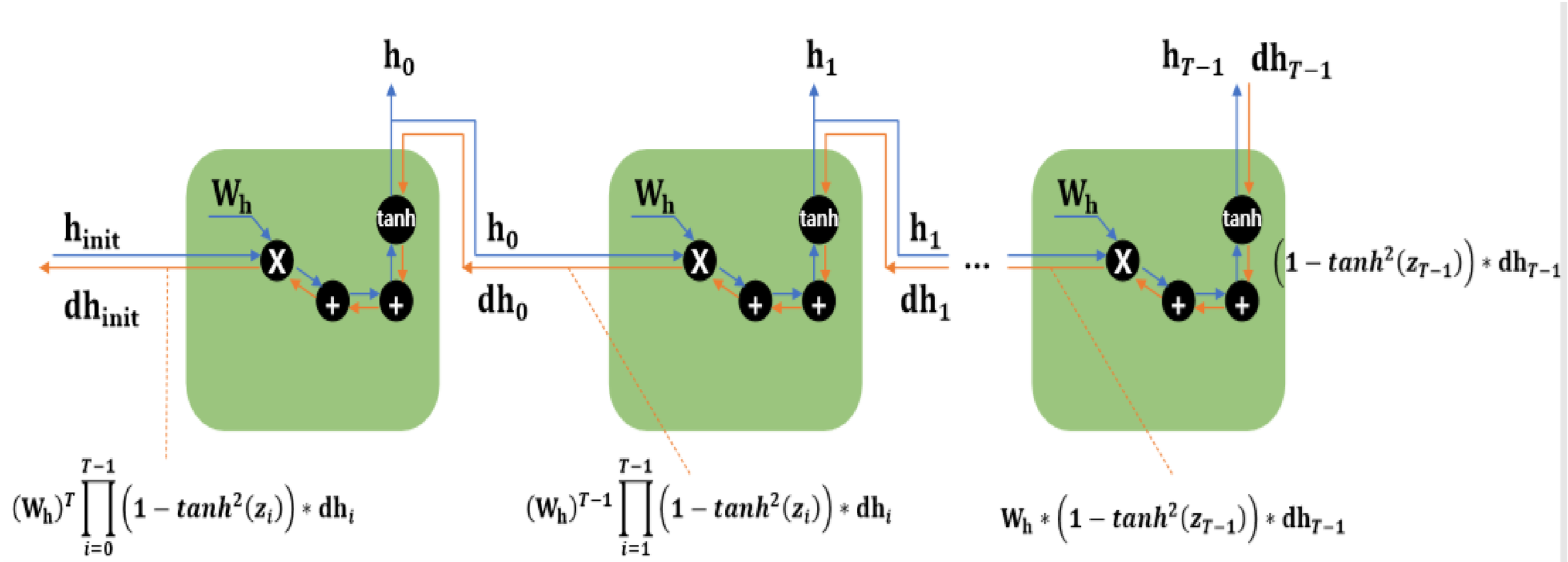
Recurrent Neural Network

- Recurrent Neural Networks (RNN) Backward
  - ✓ RNN의 역전파가 시간을 거슬러 진행되기 때문에
  - ✓ Backpropagation Through Time(BPTT) 라고 칭함.



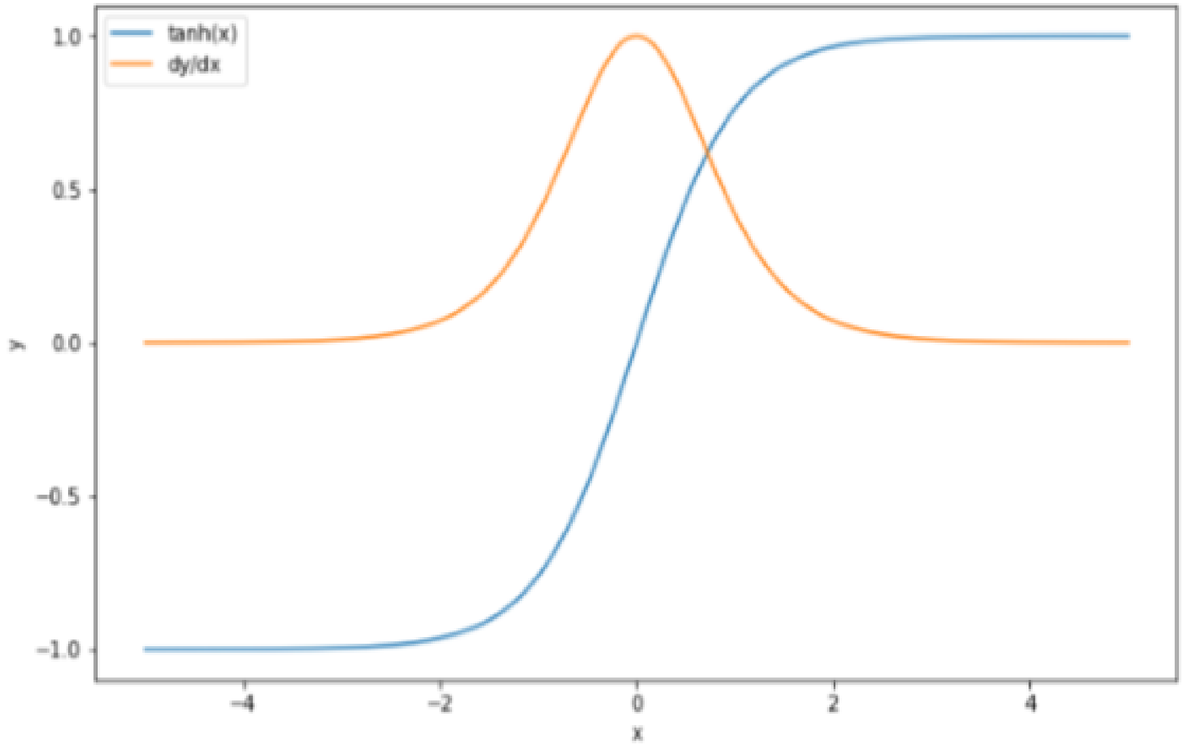
Recurrent Neural Network

- ✓ RNN이 완벽히 학습이 가능할까?
- ✓ h_{t-1}의 Gradient(역전파의 출발점)가 h_{init} (도착점)까지 만나는 tanh 활성화함수



Recurrent Neural Network

- Vanishing(or Exploding) Gradient Problem
- 원인 : tanh()와 행렬곱 : tanh의 미분
- $\mathbf{0} < \mathbf{1} - \boldsymbol{tanh}^{\mathbf{2}}(\boldsymbol{z}_{\boldsymbol{T}-\mathbf{1}}) < \mathbf{1}$
- 1보다 작은 값이 계속 곱해져
  Gradient의 크기가 줄어든다

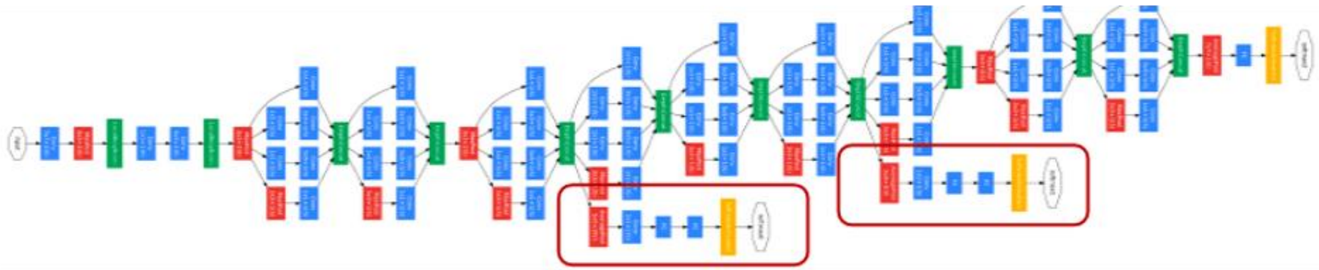


Recurrent Neural Network

✓ Vanishing Gradient Problem : 역전파가 진행되면서 Gradient가 점차 줄어 학습 능력이 저하

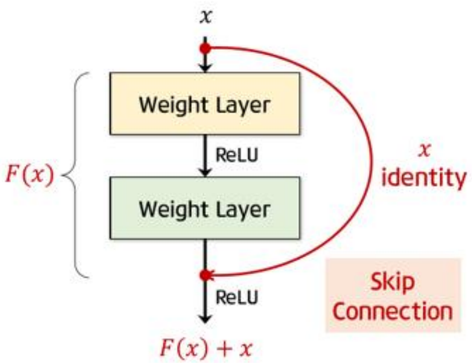
GoogLeNet (Inception v1)

- Auxiliary Classifier

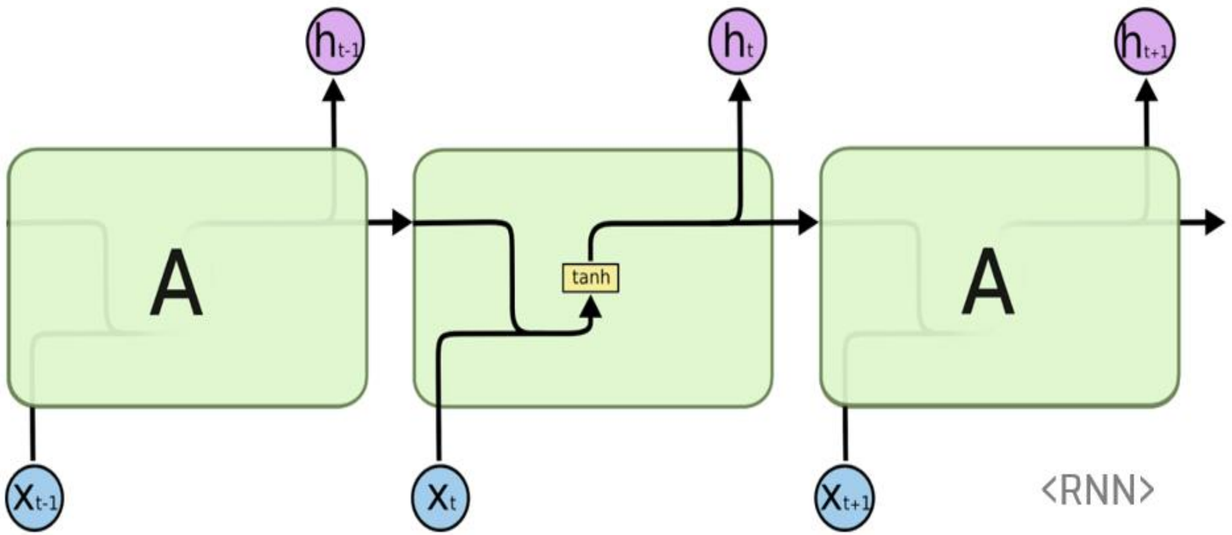


ResNet

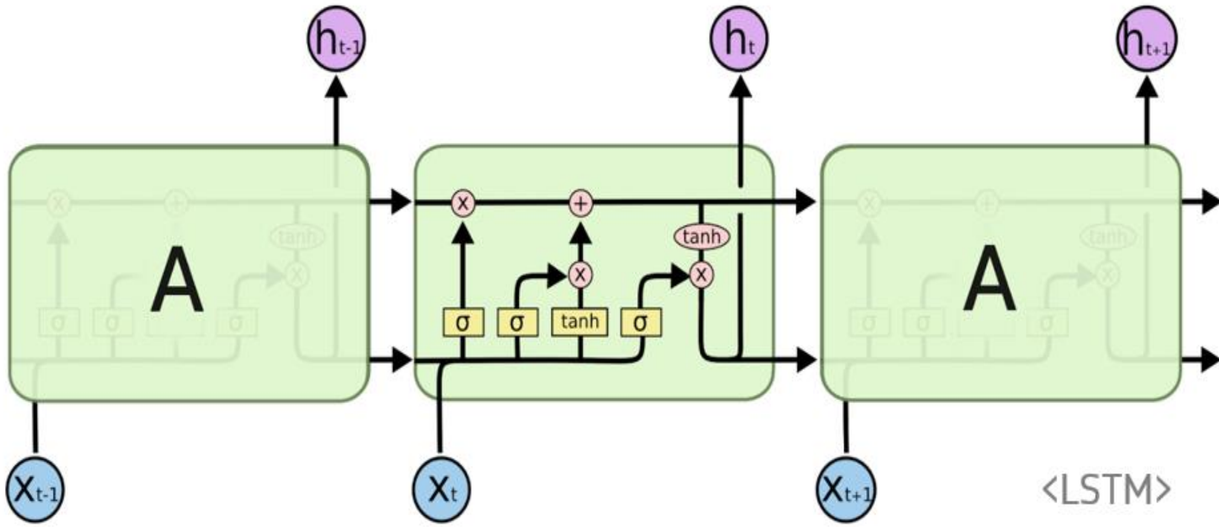
- Residual Connection



Long Short-term Memory

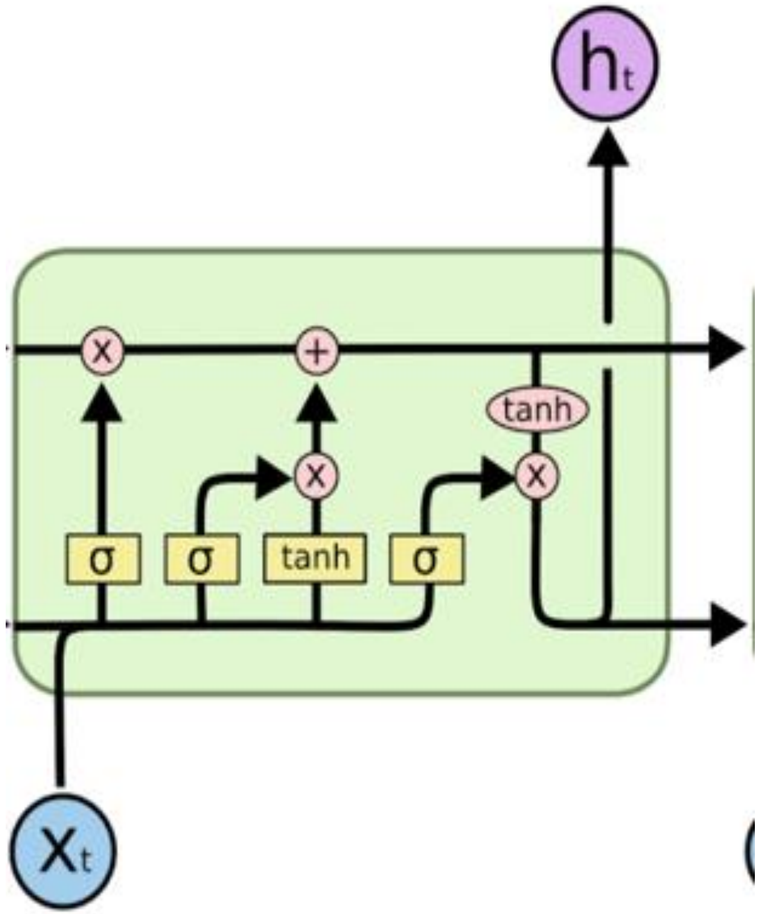


✓ 기존 RNN architecture



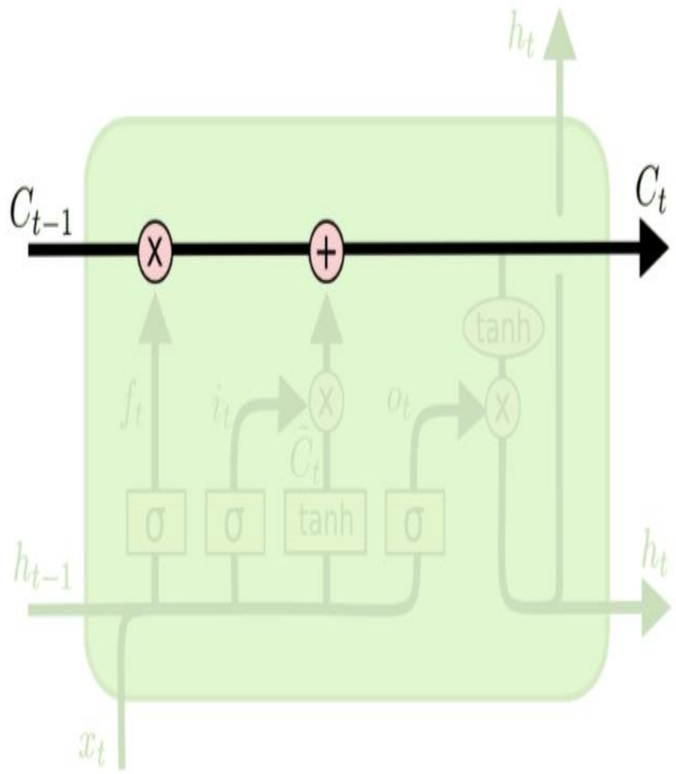
✓ LSTM architecture

Long Short-term Memory



- ✓ X (Hadamard product) : 점 단위 곱하기 연산
- ✓ ex) [1,2,3] X [1,2,3] = [1,4,9]
- ✓ Gate : 선택적으로 정보를 제어하는 관문 (σ 3개)
- ✓ σ : 시그모이드 활성화 레이어 (출력으로 0 ~ 1 출력)
- ✓ Tanh : 쌍곡 탄젠트 활성화 함수 – RNN의 그것과 역할이 같음 (출력으로 -1 ~ 1 출력)

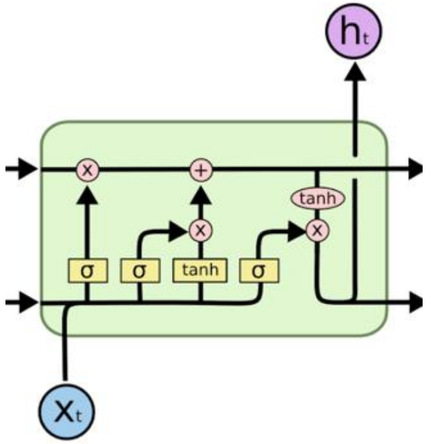
LSTM architecture : Cell State



- ✓ 하나의 컨베이어 벨트 같이 전체 연결을 관통하여 다음 단계에 전달된다
- ✓ 게이트로부터 값이 더해지거나 곱해지면서 정보를 축적한다.
- ✓ RNN에 장기 기억(Long-term Memory) 기능을 추가한 것
- ✓ → LSTM에서 Vanishing Gradient를 다루기 위한 독자적인 구조!
- ✓ * 역전파 관련 자세한 내용은 뒤에서 다시 또 다루겠습니다.

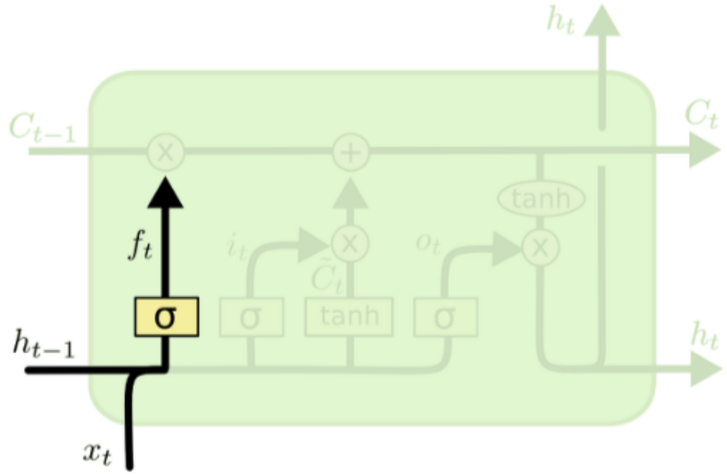
LSTM architecture

- 시점 t에서, hidden state $\mathbf{h_t}$ 와 cell state $\mathbf{c_t}$ 로 구성
  - LSTM은 Long-Term Memory인 cell $\mathbf{c_t}$로부터 정보를 **지우고**(잊고), **쓰고**, **읽을** 수 있다.(erase, write, read)

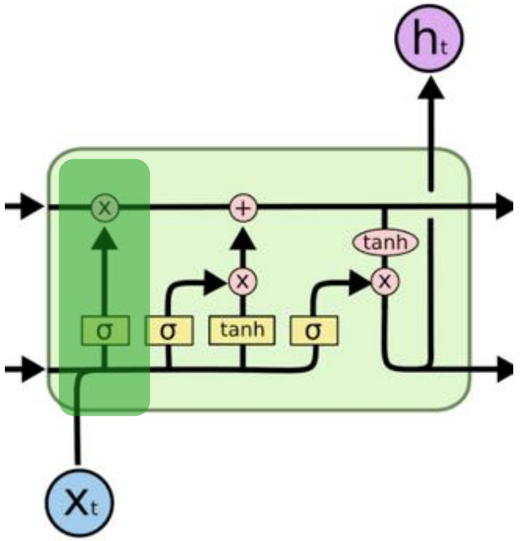


- Gate를 추가하여 어떤 정보를 지우고/쓰고/읽을지를 조절함.
  - 각 시점 t에서 gate의 원소들은 각 위치에 해당하는 원소들(정보)을 얼마나 사용할 지, 그 개방 정도를 0~1사이의 가중치로 표현함.
  - 현재의 맥락 정보를 바탕으로 gate의 개방 여부가 계산됨

Gate1. Forget gate

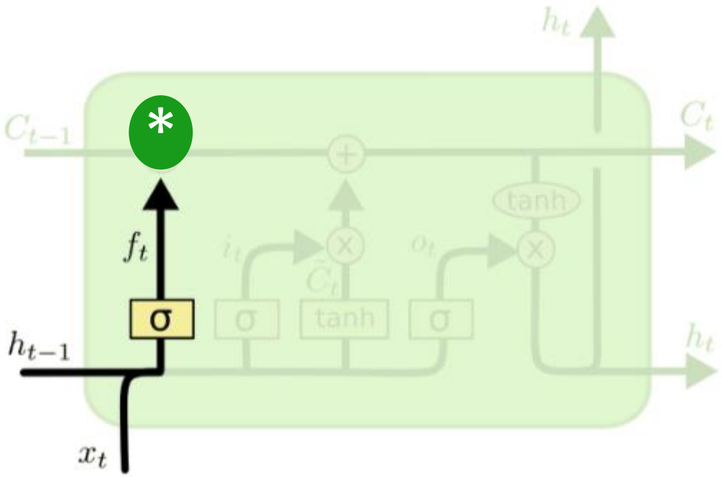


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



- ✓ 역할 : Cell state에서 어떤 정보를 잊을지 결정한다
- ✓ 입력 : 현재 입력 정보 (h_{t-1} 와 x_t)
- ✓ 출력 : sigmoid 출력(0과 1 사이의 출력값)
  → 1에 가까울 수록 “값을 유지해라”, 0에 가까울 수록 “값을 버려라“

Gate1. Forget gate



$$\mathbf{f_t} \begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix}$$

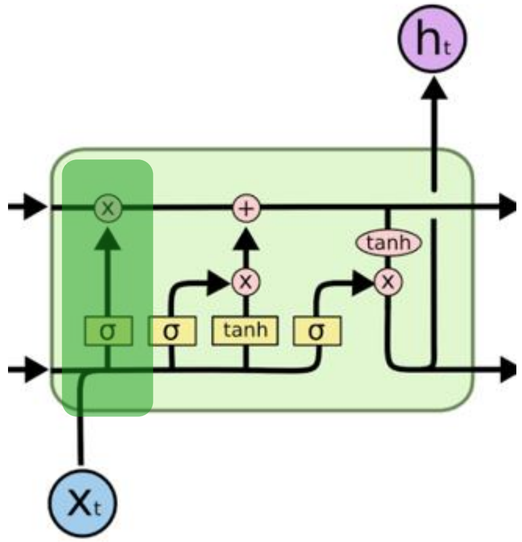
$$* \qquad *$$

$$\mathbf{c_{t-1}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\| \qquad \|$$

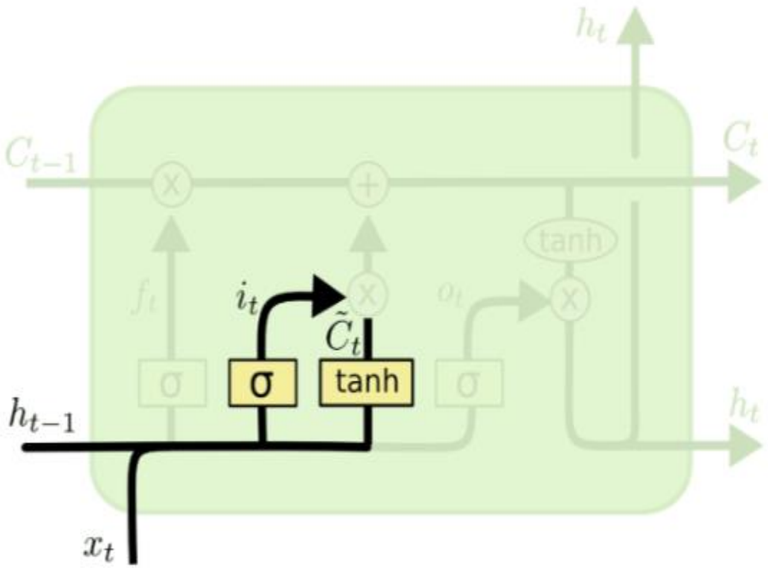
$$\mathbf{f_t} * \mathbf{c_{t-1}} \begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix}$$

망각 유지



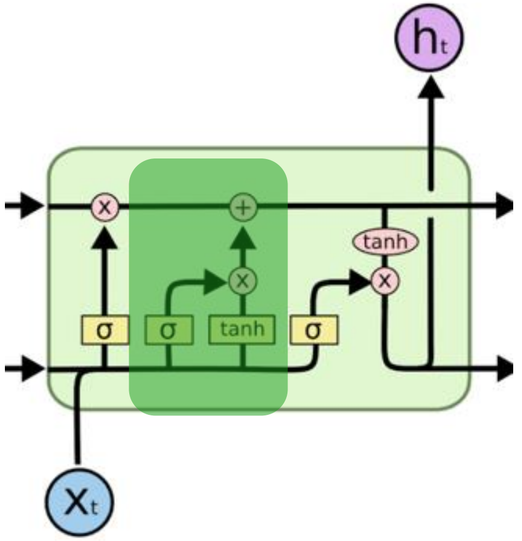
✓ 계산된 forget gate f_t를 이전 시점의 장기기억 $C_{(t-1)}$ 에 요소별로 곱(Hadamard product)하여 이전 시점의 각 정보의 망각(또는 유지)를 결정

Gate2. Input gate



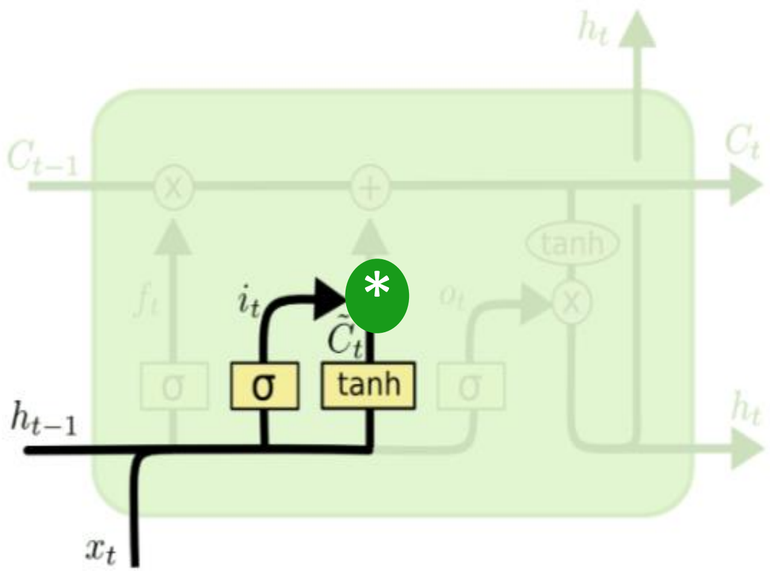
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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



- ✓ 역할 : Cell state에 어떤 정보를 더해줄지 결정한다
- ✓ 입력 : 현재 입력 정보 (h_{t-1}와 x_t)
- ✓ 출력 : tanh의 출력(-1 ~ 1) → 필요한 정보를 추출
  sigmoid의 출력(0 ~ 1) → tanh에서 추출한 정보를 취사선택

Gate1. Forget gate



$$\tilde{\mathbf{c}}_t = \begin{bmatrix} -1 & 0.8 \\ -0.5 & 0.9 \end{bmatrix}$$

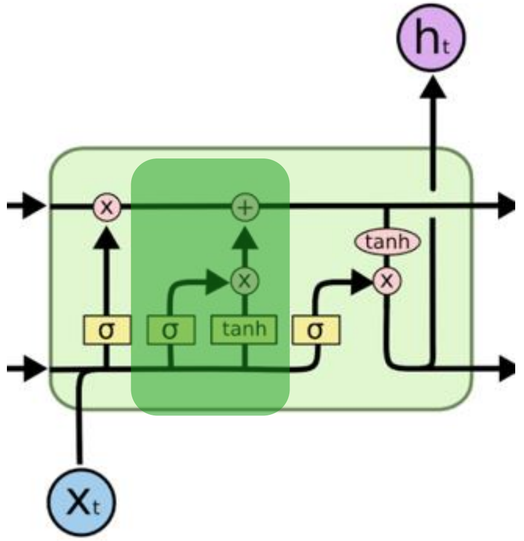
$$* \quad * \quad *$$

$$\mathbf{i}_t = \begin{bmatrix} 0.9 & 0.1 \\ 0.8 & 0 \end{bmatrix}$$

$$\| \quad \|$$

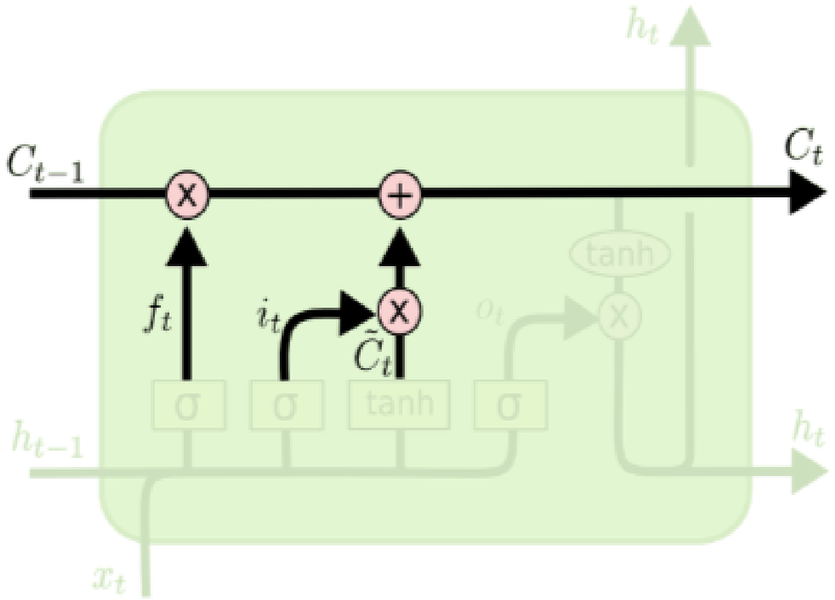
$$\tilde{\mathbf{c}}_t * \mathbf{i}_t = \begin{bmatrix} -0.9 & 0.08 \\ -0.4 & 0 \end{bmatrix}$$

유지 망각

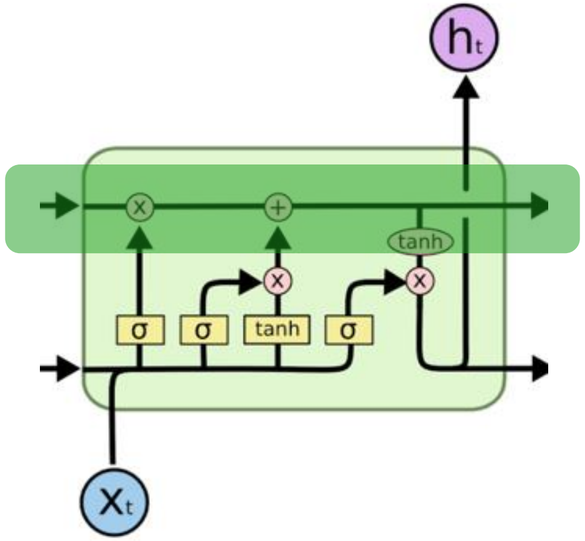


✓ •계산된 input gate i_t 를 현재의 입력 정보를 가공한 새로운 기억셀 $\widetilde{C_t}$에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 장기 기억($f_t * C_{t-1}$)에 더할 것인지를 결정

Cell State 돌아보기



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



■ 지금까지의 과정

✓ Cell state(C_{t-1})와 Forget Gate(f_t)를 pointwise(요소 별) 곱 해준다 → 망각효과

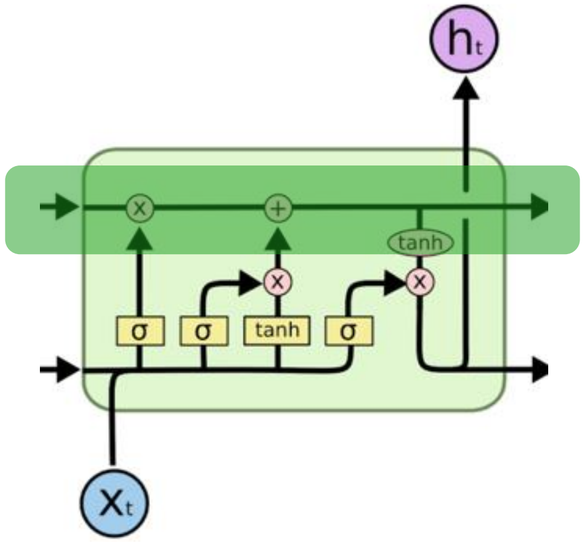
✓ Cell state(C_{t-1})와 Input Gate($i_t * \widetilde{C_{t-1}}$)를 합한다 → 정보 입력효과

✓ 망각과 정보 입력을 마친 현재의 Cell state(C_t)는 다음 Time Step으로 넘어간다

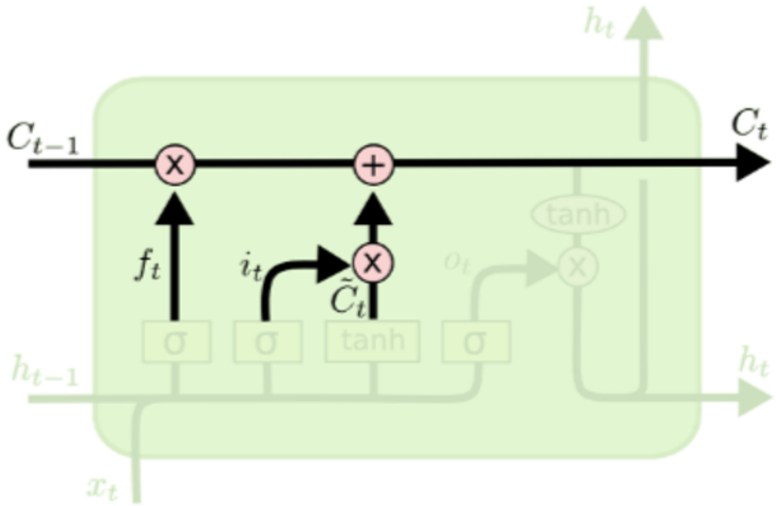
Cell State 돌아보기

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

$$\begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix} + \begin{bmatrix} -0.9 & 0.08 \\ -0.4 & 0 \end{bmatrix} = \begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}$$

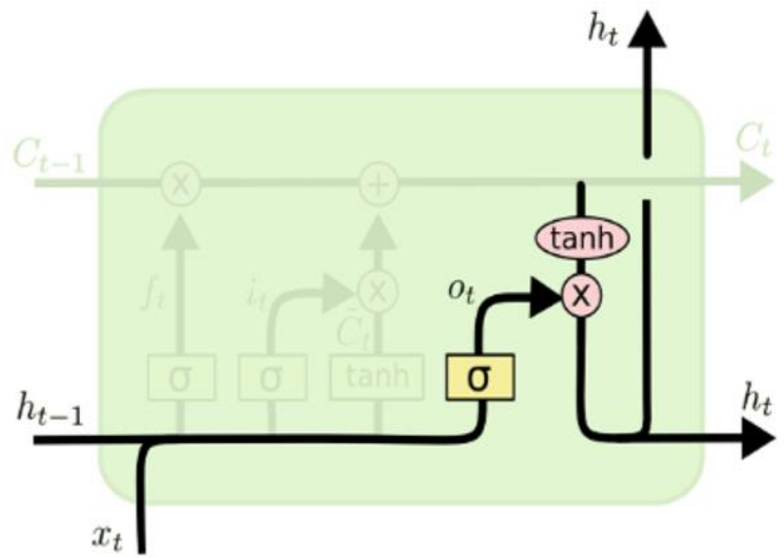


$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$



$$\begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}$$

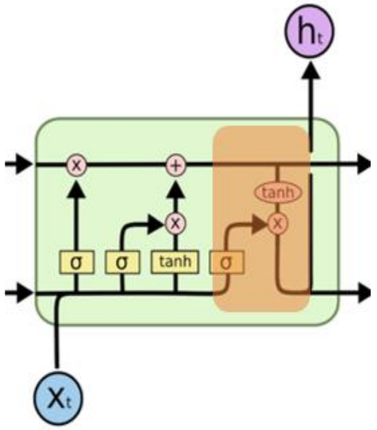
Gate3. Output gate



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

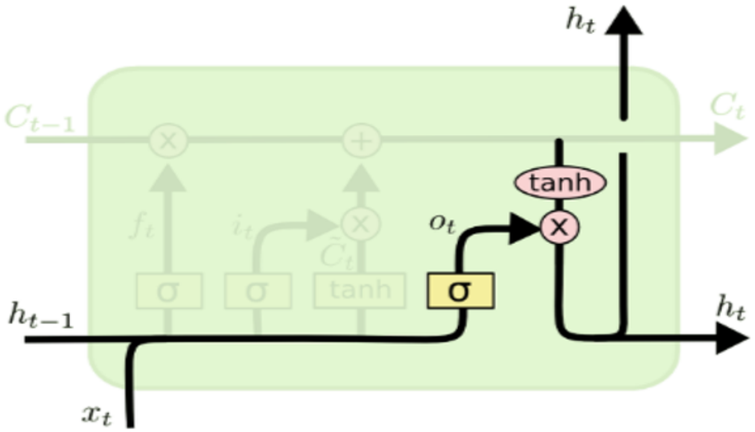
$$h_t = o_t * \tanh\left(C_t\right)$$

$$tanh\left(\begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}\right)$$



역할 : Cell state와 현재 입력으로부터 출력값 (h_t)정보를 선택
입력 : 현재 입력 정보 (h_{t-1} 와 x_t)와 망각, 입력 게이트를 거친 Cell state(C_t)
출력 : Cell state의 tanh 출력값과 현재 입력의 sigmoid 출력값의 합 (0 ~ 1)
→ 장기기억으로부터 정보를 추출하고, 현재 입력과 대비해서 중요한 것만 선별하자

Gate3. Output gate



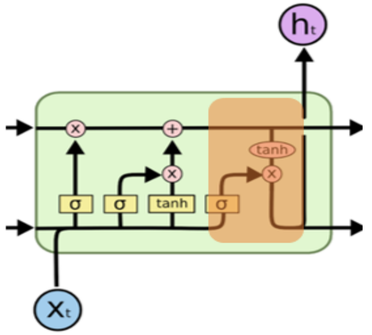
$$\mathbf{h_t^{(raw)}} \begin{bmatrix} -0.9 & 0.9 \\ -0.7 & 0.7 \end{bmatrix}$$

$$* \qquad *$$

$$\mathbf{o_t} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\| \qquad \|$$

$$\mathbf{h_t} \begin{bmatrix} 0 & 0 \\ -0.7 & 0.7 \end{bmatrix}$$



■ 계산된 output gate o_t 를 현재의 장기기억(기억셀)으로부터 꺼내온 $h_{t(raw)}$ $(\tanh(C_t))$에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 단기기억(h_t)으로 사용할지를 결정

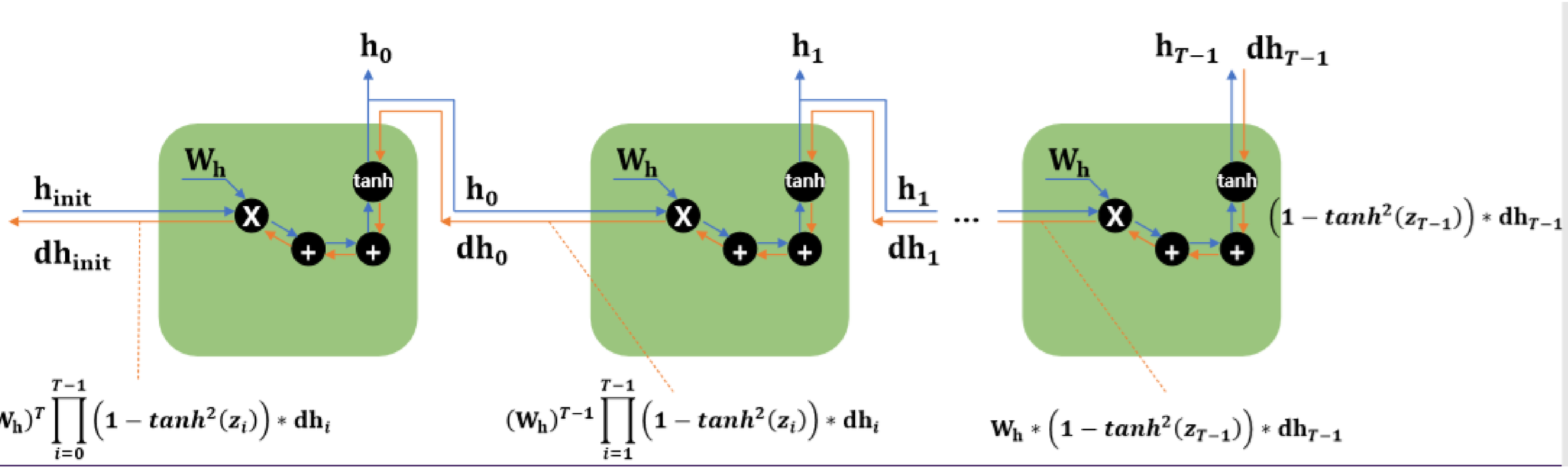
LSTM Backpropagation

RNN은 왜 문제가 생겼는지 다시 생각해볼게요

1.수 많은 활성화 함수(tanh)를 거쳐서 기울기 소실 발생 → 이미 언급되었다.

여기서 새로이 밝히는 문제점 또 하나

2. 똑같은 가중치를 반복해서 곱해주고 있다.



LSTM Backpropagation

반복곱이 왜 문제가 생기는가?

Case 1) 1보다 큰 수의 똑같은 행렬을 계속 곱하면?

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \text{무지하게 커진다!!!!}$$

Case 2) 1보다 작고 0보다 큰 수의 행렬을 계속 곱하면?

$$\begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} = \text{엄청나게 작아진다...}$$

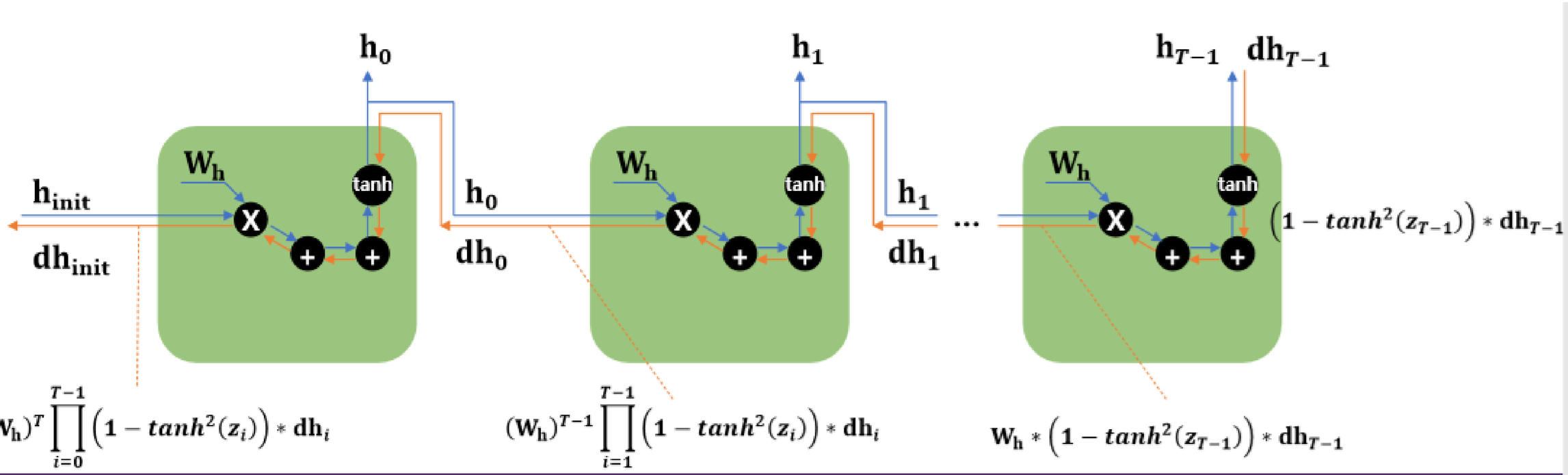
/* elice */

LSTM Backpropagation

똑같은 가중치를 반복해서 곱해주고 있다.

→ 역전파 때는 똑같은 가중치를 반복해서 나누기 때문에

무지하게 커지든 엄청나게 작아지든 둘 중 하나

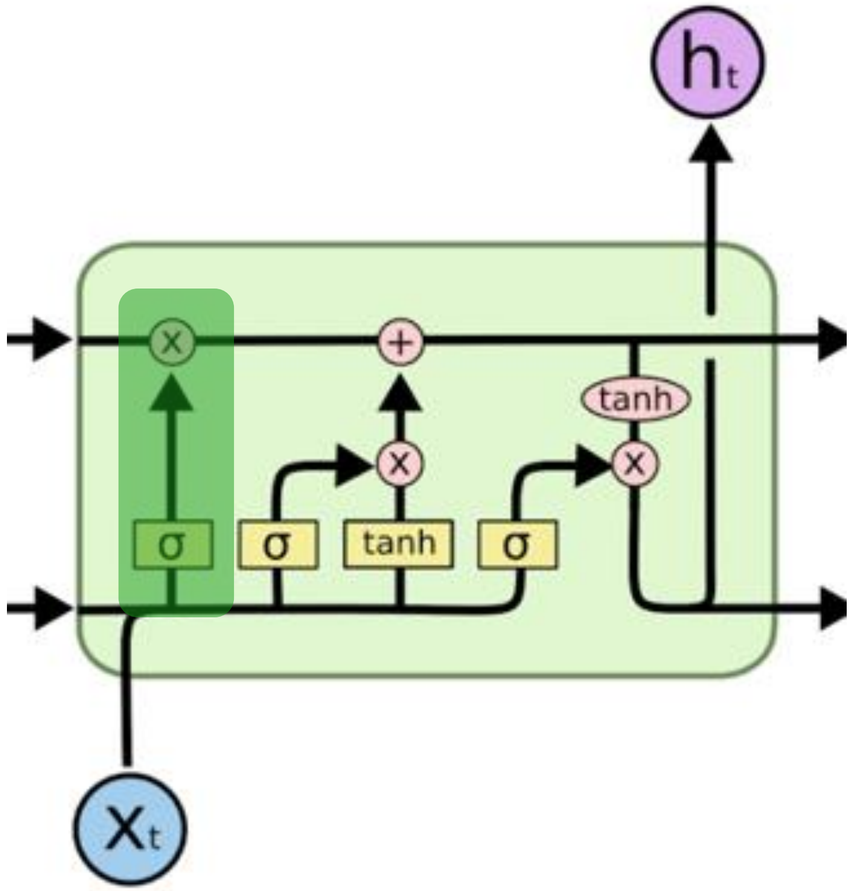


LSTM Backpropagation

그래서 LSTM은?

Q. Cell state가 곱셈에서 만나는 Forget gate(f_t) 값도
매 Time Step 마다 똑같지 않나?

A. **No!** (45번 Page에서 Forget gate 참조)



역할 : Cell state에서 어떤 정보를 잊을지 결정한다
입력 : 현재 입력 정보 (h_{t-1}와 x_t)
출력 : sigmoid 출력(0과 1 사이의 출력값)
→ 똑같은 W_t가 아닌 매번 입력에 따른 0 ~ 1사이
sigmoid 활성화 함수의 출력값이 곱해진다.
→ 적어도 똑같은 행렬을 반복적으로 곱하는 RNN보다 훨씬 낫다

LSTM Backpropagation

■ LSTM은 RNN의 기울기 소실 문제를 해결해주었습니다.

1.LSTM은 Cell state라는 장기 기억 장치를 만들어 두었습니다

2.Cell state는 특성 상 기울기 소실 문제에서 비교적 RNN보다 자유롭습니다

3.3가지 Gate(망각, 입력, 출력)로 정보의 흐름 강도를 조절하는 기능이 있습니다

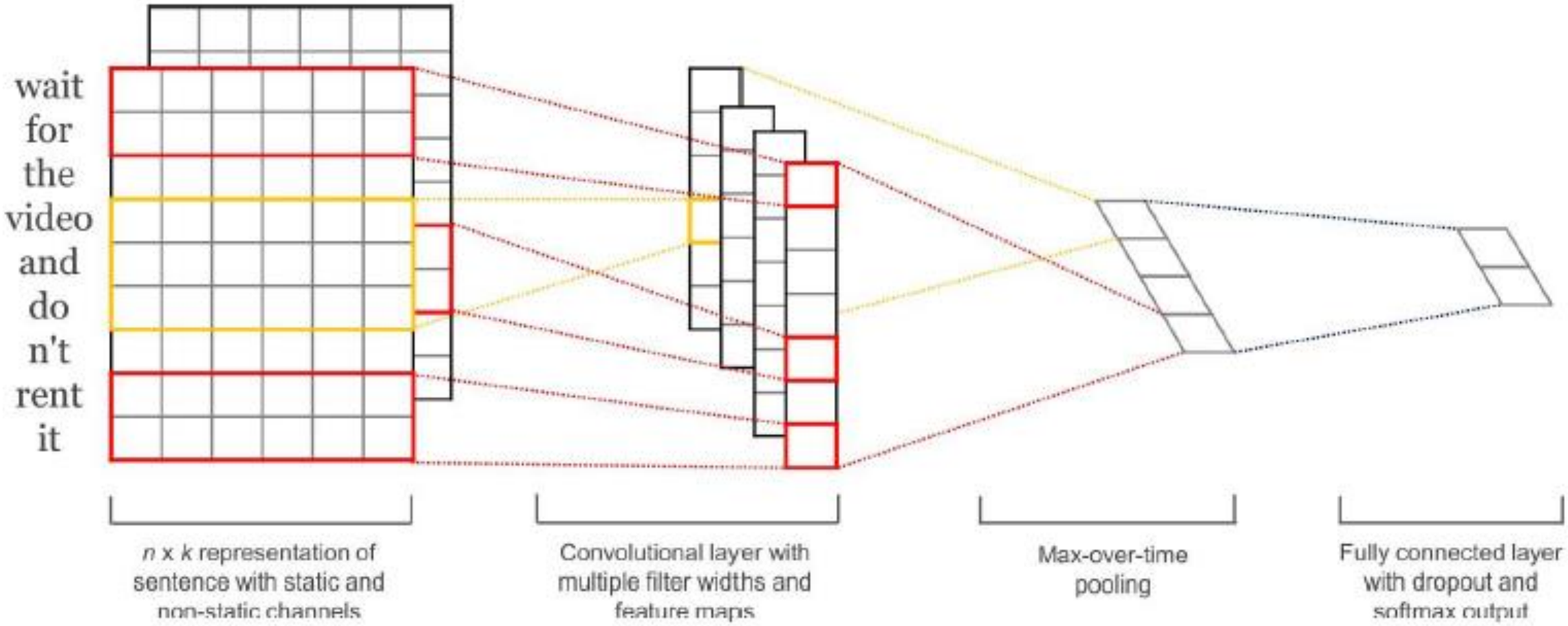
4.RNN 계열의 신경망은 여러 층을 쌓을 수도 있습니다.

5.최종 Output에 어떤 Classifier를 다느냐에 따라 다양한 Task를 수행할 수 있습니다.

03

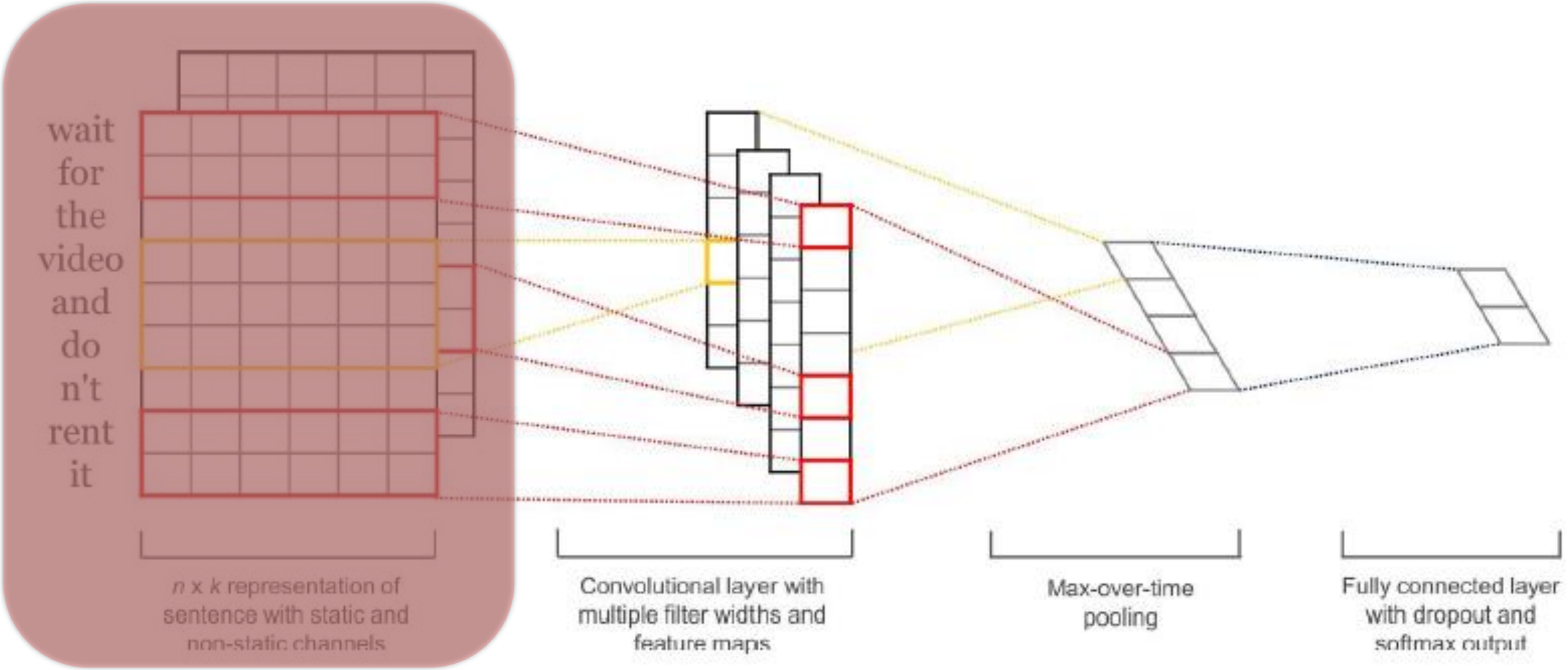
CNN-based

CNN for Sentence Classification



03 CNN-based

CNN for Sentence Classification



CNN for Sentence Classification : Input shape

✓ **Embedding Matrix = N by K**

✓ **여기서 N은 한 문장에서 투입할 Word의 개수(Max Length)**

- 문장의 word가 Max Length보다 작으면 0 padding을 하여 투입함.
- 문장의 word가 Max Length보다 크면 이후 단어들은 고려하지 않음.

✓ **여기서 K는 Embedding Dimension임.**

- 주로 Pre-trained 된 Embedding을 사용한다.
- 이러한 벡터들은 Update를 할 수도 있고 하지 않을 수도 있다. -> 성능에 영향 미침.

✓ **Input shape의 다양한 Case**

① Embedding Matrix을 Update함.

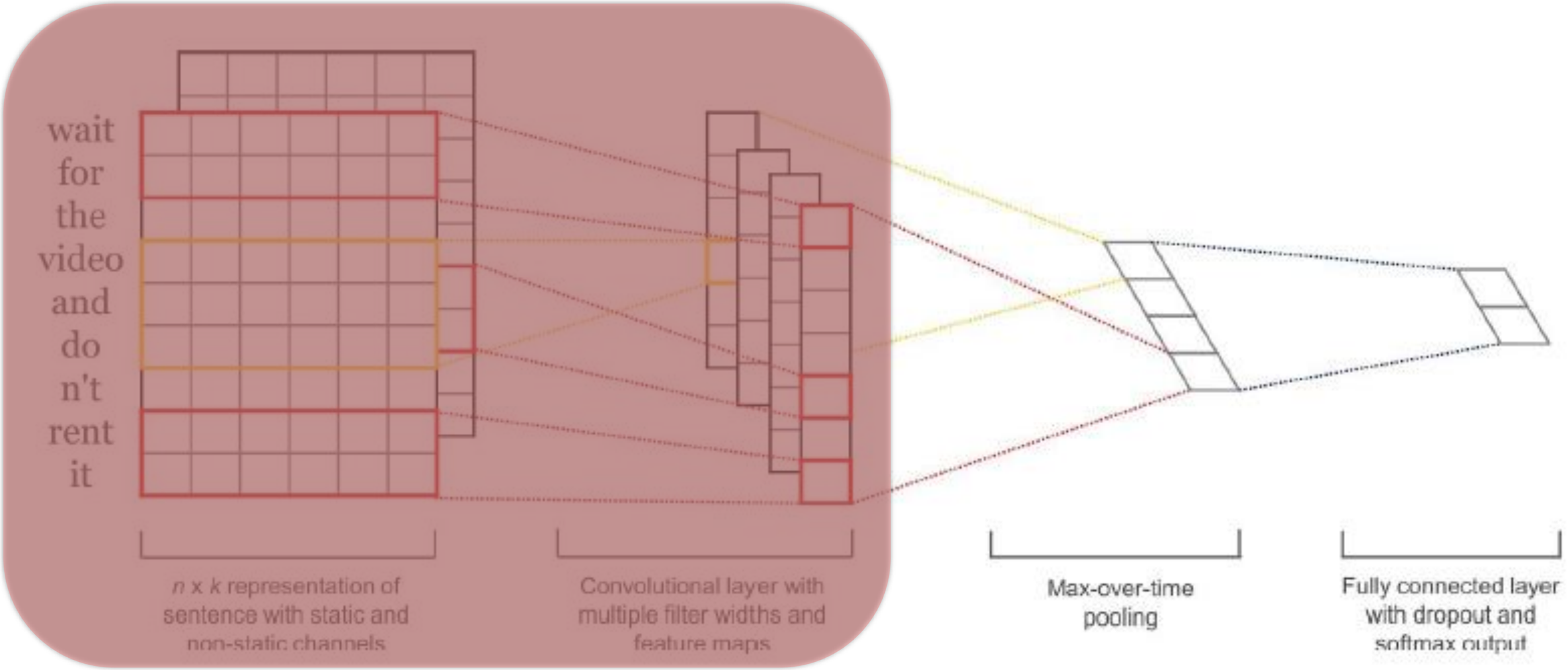
② Embedding Matrix을 Update하지 않음.

③ Random하게 Embedding Matrix를 생성하여 Update함

④ 여러 Embedding Matrix를 Channel로 쌓는다.

03 CNN-based

CNN for Sentence Classification



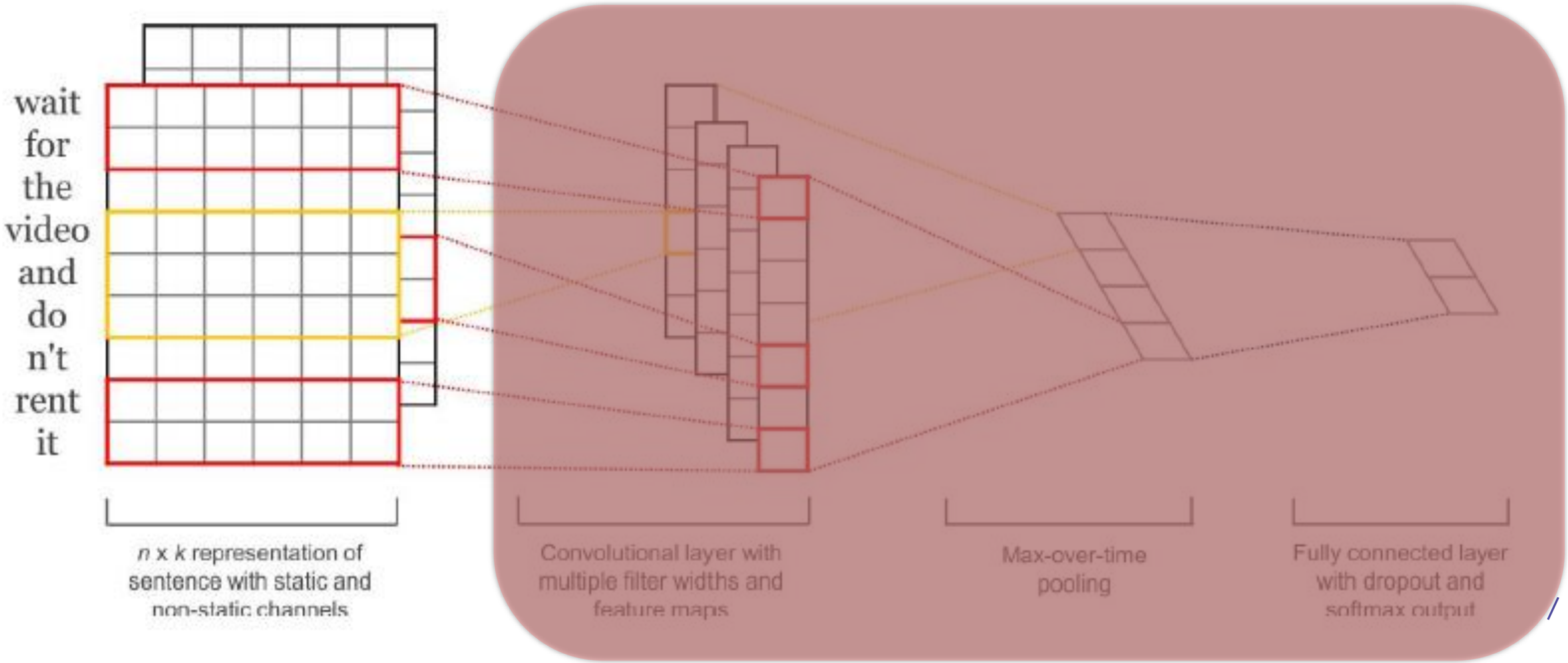
CNN for Sentence Classification : Convolution

✓ Convolution

- 다양한 사이즈의 Convolution을 사용함 -> 다양한 Feature 생성
- 주로 이미지에서는 row-column이 같은 filter를 사용 하지만 본 모델은 한 Word를 정확하게 고려하기 위해 col을 모두 포함할 수 있도록 conv를 진행함.
- Stride는 주로 1준다.
- filter의 row가 커질 수록 한번에 고려하는 단어의 개수가 많아진다.

03 CNN-based

CNN for Sentence Classification



CNN for Sentence Classification : Pooing & FC

✓ **Max Pooling**

➢ 가장 중요한 Feature를 추출하는 과정.

✓ **Fully connected layer**

➢ Output 노드를 본 모델은 2개를 사용(긍/부정) 그러나 task에 따라 다르게 정의할 수 있다. (Regression도 가능)

✓ **Learning Hyper-parameter**

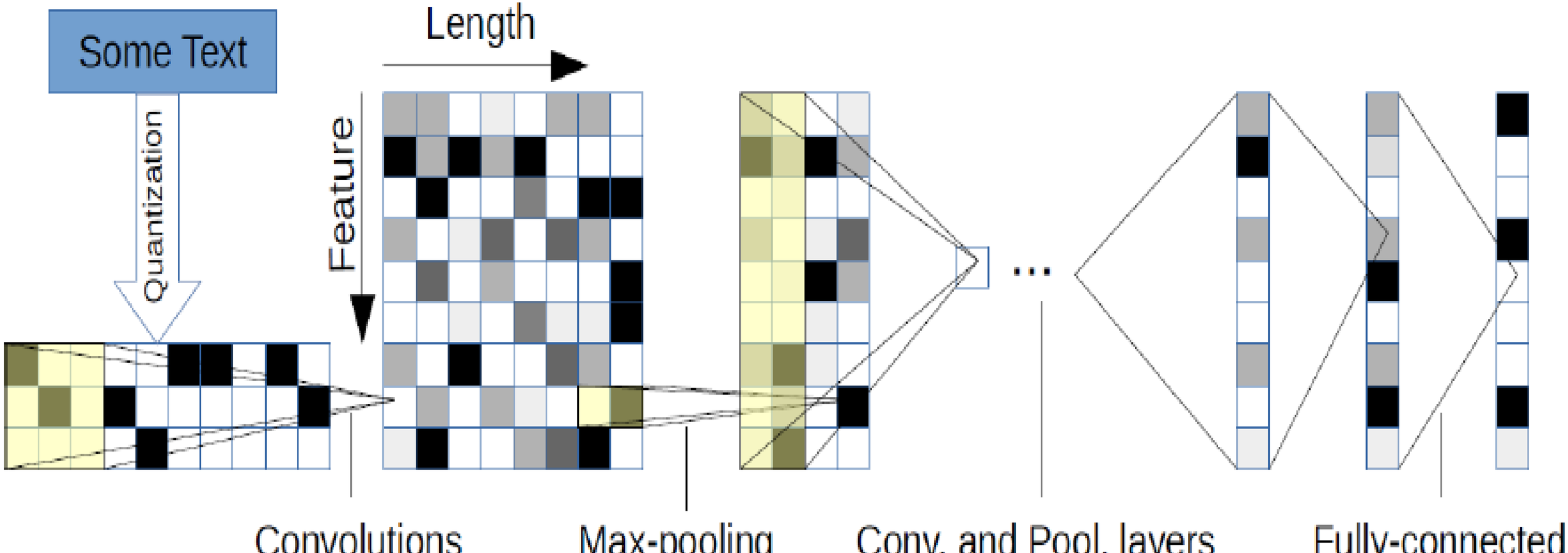
① Filter 사이즈는 3, 4, 5를 사용하였고 각 Filter마다 100개 씩 사용
② 마지막 FC Layer는 Dropout을 0.5로 적용하였음.
③ L2 regularization을 사용
④ Mini batch size는 50으로 구성

CNN for Sentence Classification : result performance

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Character-level CNN(Zhang et al, 2015)

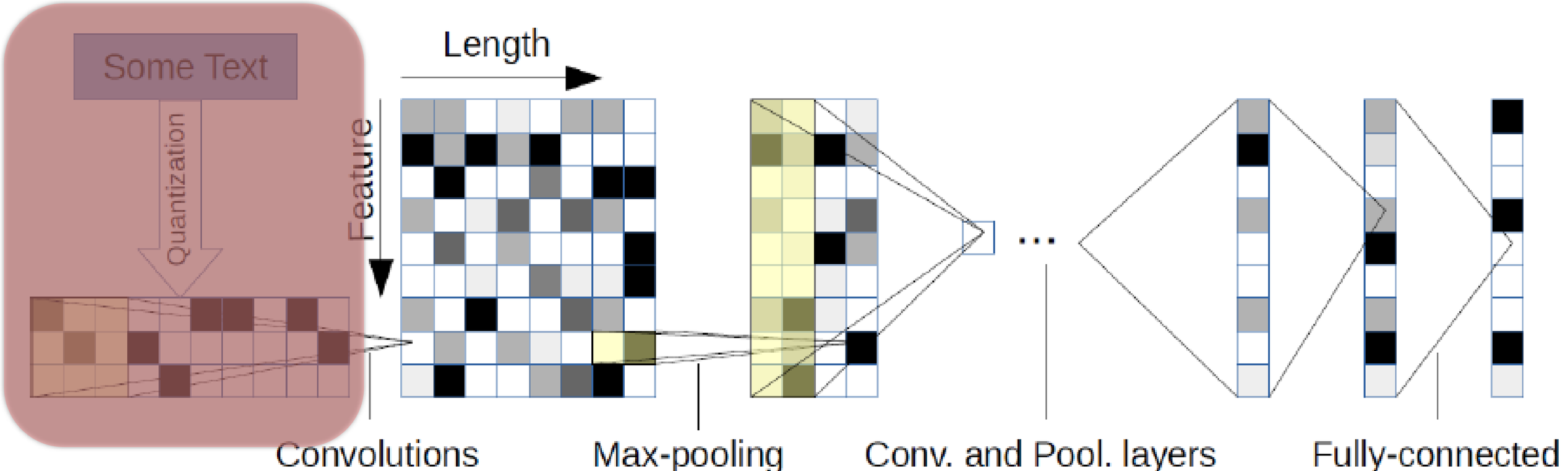
✓ 70개의 Character 사용 : 26개 알파벳, 33개의 특수 문자(공백포함)



Character-level CNN(Zhang et al, 2015)

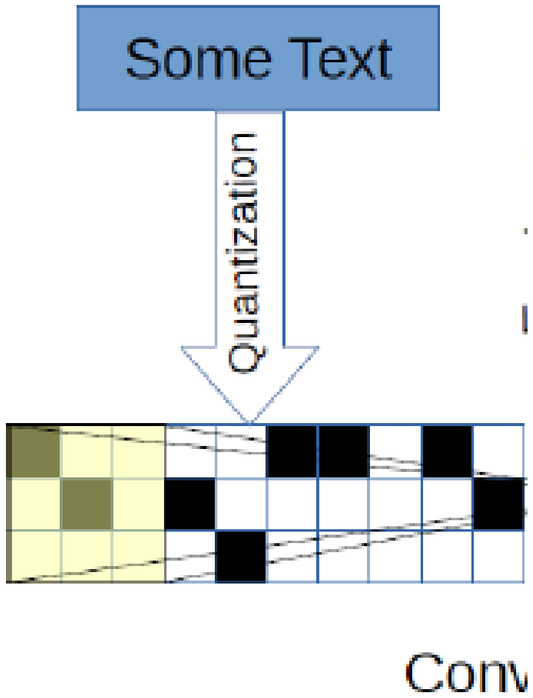
✓ 70개의 Character 사용 : 26개 알파벳, 33개의 특수 문자(공백포함)

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:'''/\|_@#$%^&*~`'+-=<>()[]{}
```



Character-level CNN(Zhang et al, 2015) : input shape

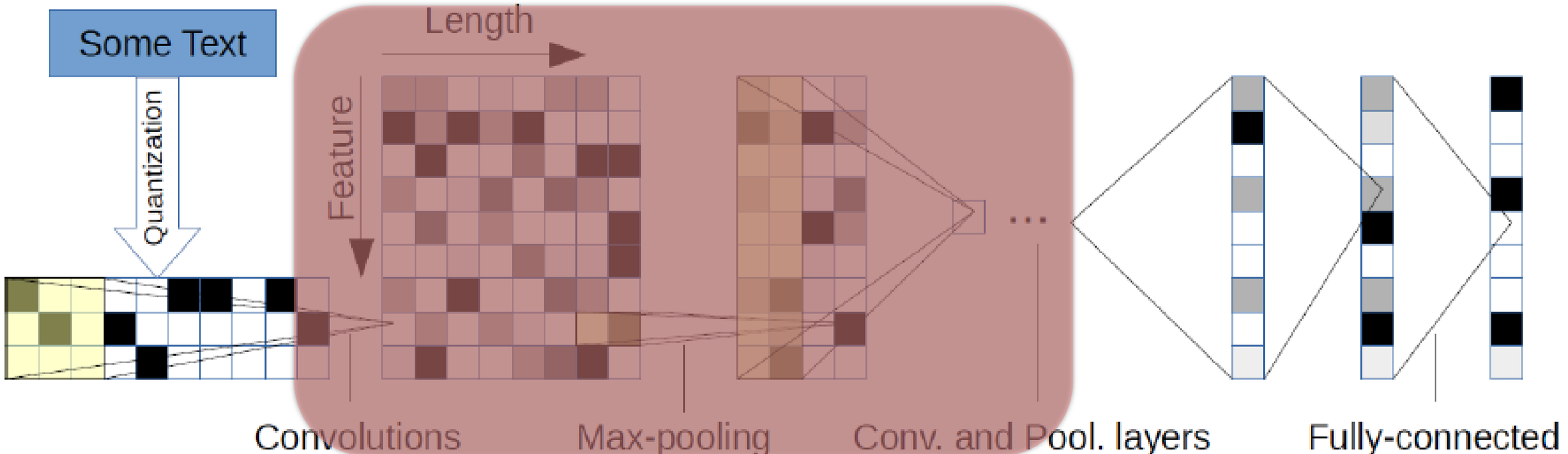
✓ 각 열은 70개의 character를 one-hot encoding vector로 표현한 것.



Character-level CNN(Zhang et al, 2015)

✓ 70개의 Character 사용 : 26개 알파벳, 33개의 특수 문자(공백포함)

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:'''/\|_@#$%^&*~`'+-=<>()[]{}
```



Character-level CNN(Zhang et al, 2015) :Convolution & padding

✓ Convolution : kernel size는 다음과 같음

- 1번째 Convolution Layer : 70 by 7
- 2번째 Convolution Layer : 1024 by 7
- 3번째 Convolution Layer : 1024 by 3

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

✓ Max Pooling

- Max Pooling size는 1 by 3이고 stride는 3으로 주었다. (Text CNN과 어떤 차이?)

Character-level CNN(Zhang et al, 2015) : Fully-connected

✓ FC layer

- Task에 따라 Output node를 바꿀 수 있다.
- Dropout 0.5를 적용하였다.

- Fully connected layers

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

✓ Data augmentation

- 그러나 성능이 매우 좋지 않았다.

Contact

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io