AI Solution for AI needs customer

Mr.MIND

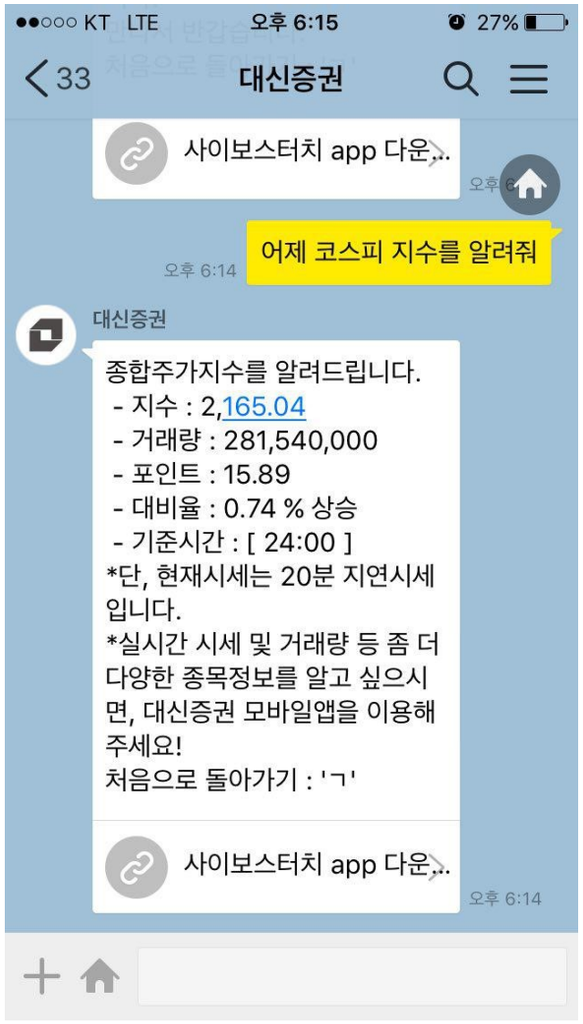
# Python과 Tensorflow를 활용한
# CHATBOT 개발

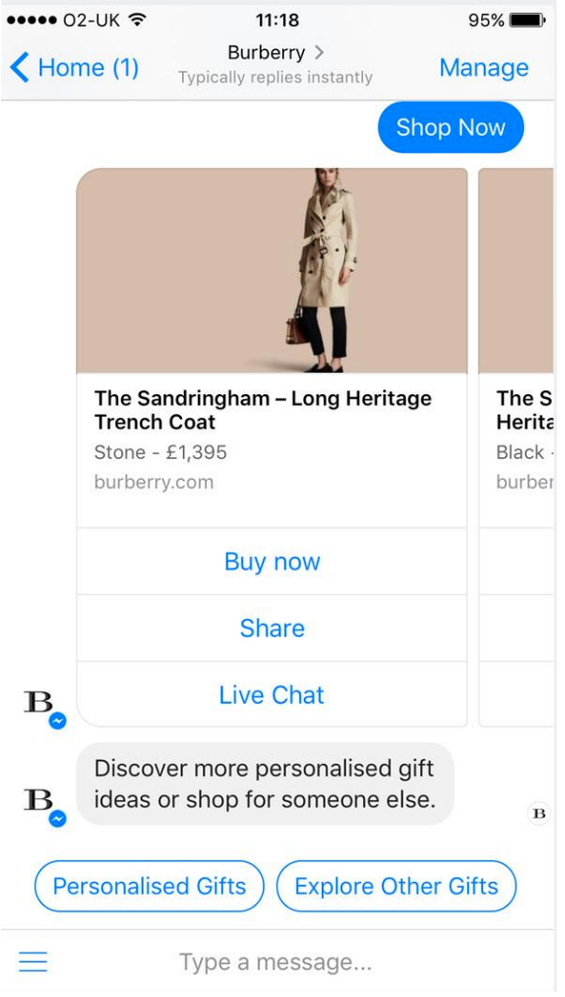
김선호 미스터마인드 CTO
| seonho.aii@gmail.com

V1.0 | Released in 2018.01
Document by **Mr.MIND AI Consulting**

Technology for network Appliance _ Understand User, Create Experience **AI Consulting Inc.**

# About CHATBOT


구글 홈

SK텔레콤 누구

네이버 웨이브

카카오 미니

# CHATBOT 기술의 발전방향

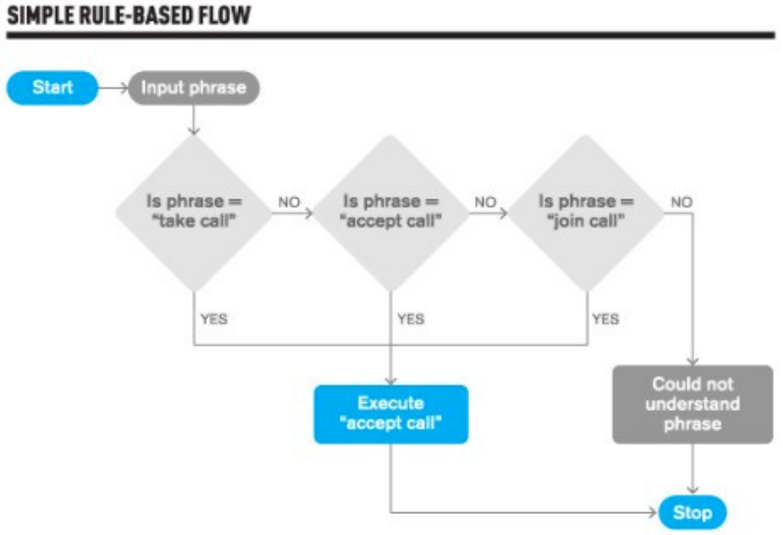
**Easy** → **Hard**

| Easy | Hard |
| --- | --- |
| 검색기반 모델<br>Retrieval-based model | 생성모델<br>Generative model |
| 전통적인 머신러닝 기법<br>Traditional ML algorithms | 딥러닝 기법<br>Deep Learning algorithms |
| 짧은 대화<br>Short Conversation | 긴 대화<br>Long Conversation |
| 제한된 도메인<br>Closed Domain | 열린 도메인<br>Open Domain |

# Retrieval-Based vs Generative Models

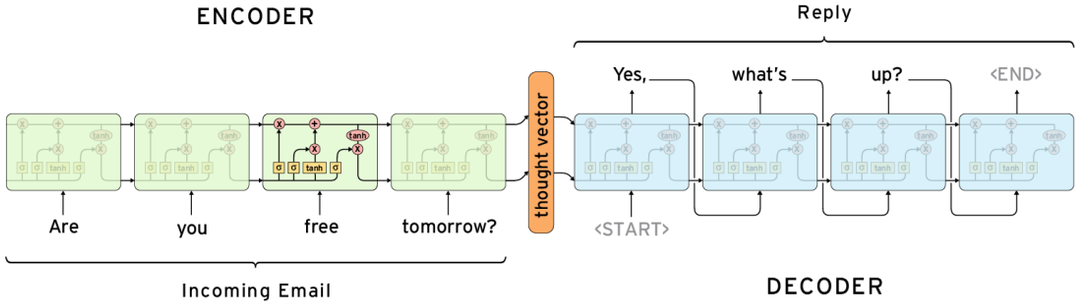
## Retrieval-based models (easier)

기 정의된 응답에 의존하여 입력 및 컨텍스트에 대한 적절한 응답을 선택함.
이 시스템은 새로운 텍스트를 생성하지 않고 단지 고정 된 세트에서 응답을 선택함.




## Generative models (harder)

사전 정의 된 응답에 의존하지 않고 모델을 통해 항상 새로운 응답을 생성함
생성 모델은 일반적으로 기계 번역 기술을 기반으로함.



# Use Deep Learning or Not

## Using traditional algorithms

현재 대부분의 챗봇들은 전통적인 머신러닝 기법들을 더 많이 사용해 왔음.
샘플데이터가 적을땐 오히려 딥러닝보다 더 좋은 성능을 보일 수 있음.

Clustering
Decision tree
LSA
HMM
Dictionary
SVM
Logistic
Bayesian
TFIDF
Random forest

## Using Deep Learning

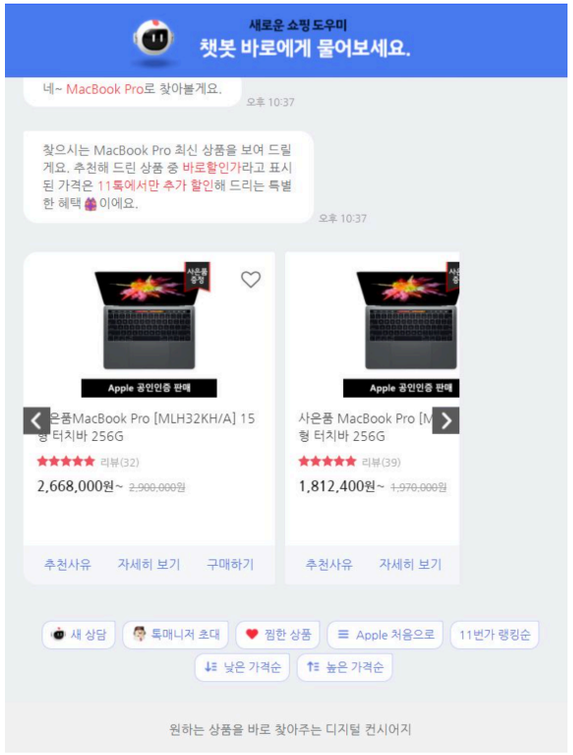
항상 성능이 좋을수 없음.
데이터가 많을수록 좋은 성능을 보일 가능성이 높아지므로 최근 여러 분야에서의 데이터의 홍수와 맞물려 좋은 성능을 보이고 있음.

DMN
LSTM
GRU
Attention
Word2Vec
CNN
RNN
DNN
CharCNN
Seq2Seq

# Short Conversation vs Long Conversation

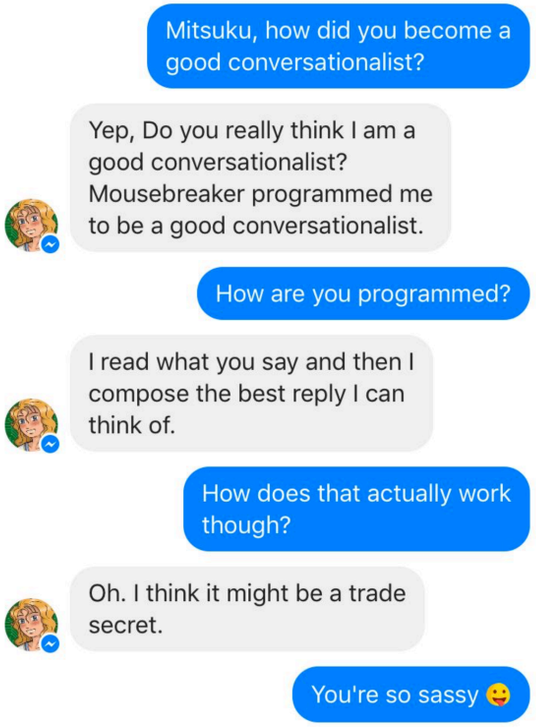
## Short Conversation

단일 입력에 대한 단일 응답으로 목표가 달성될 수 있음.




## Long conversation

여러턴에 걸쳐서 무엇을 말하는지 추적해야 함



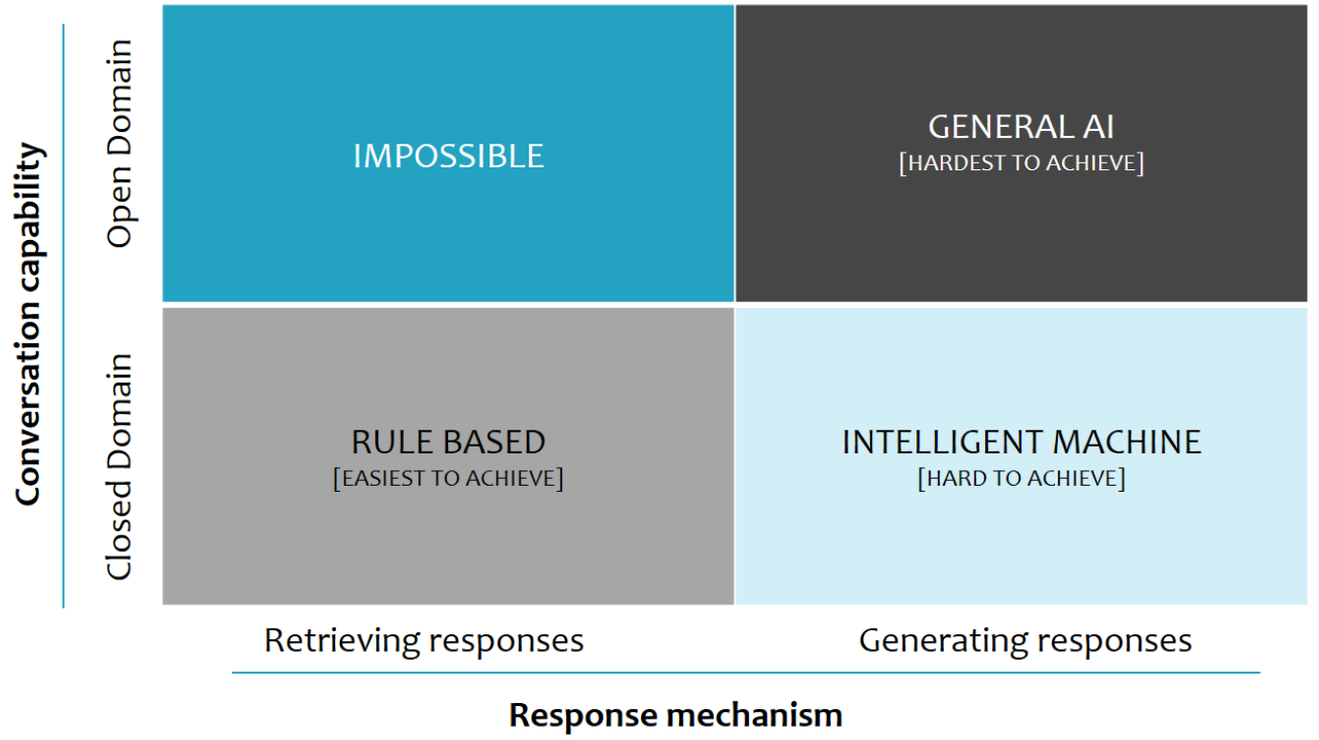
# Closed Domain vs Open Domain

## Closed Domain

특정 주제에 대해 제한된 질문을 할 수 있으며 해당 주제에 대한 대답만 가능하도록 프로그래밍 되어 있음.

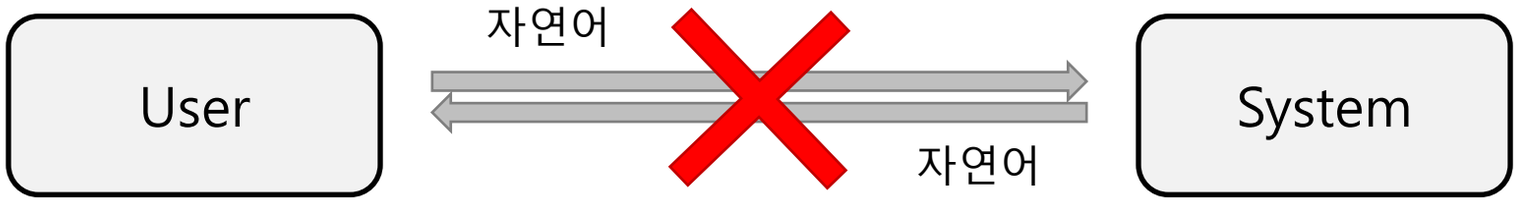
## Open Domain

어떤 주제든 상관없이 질문을 할 수 있고 그에대한 응답을 할 수 있음.

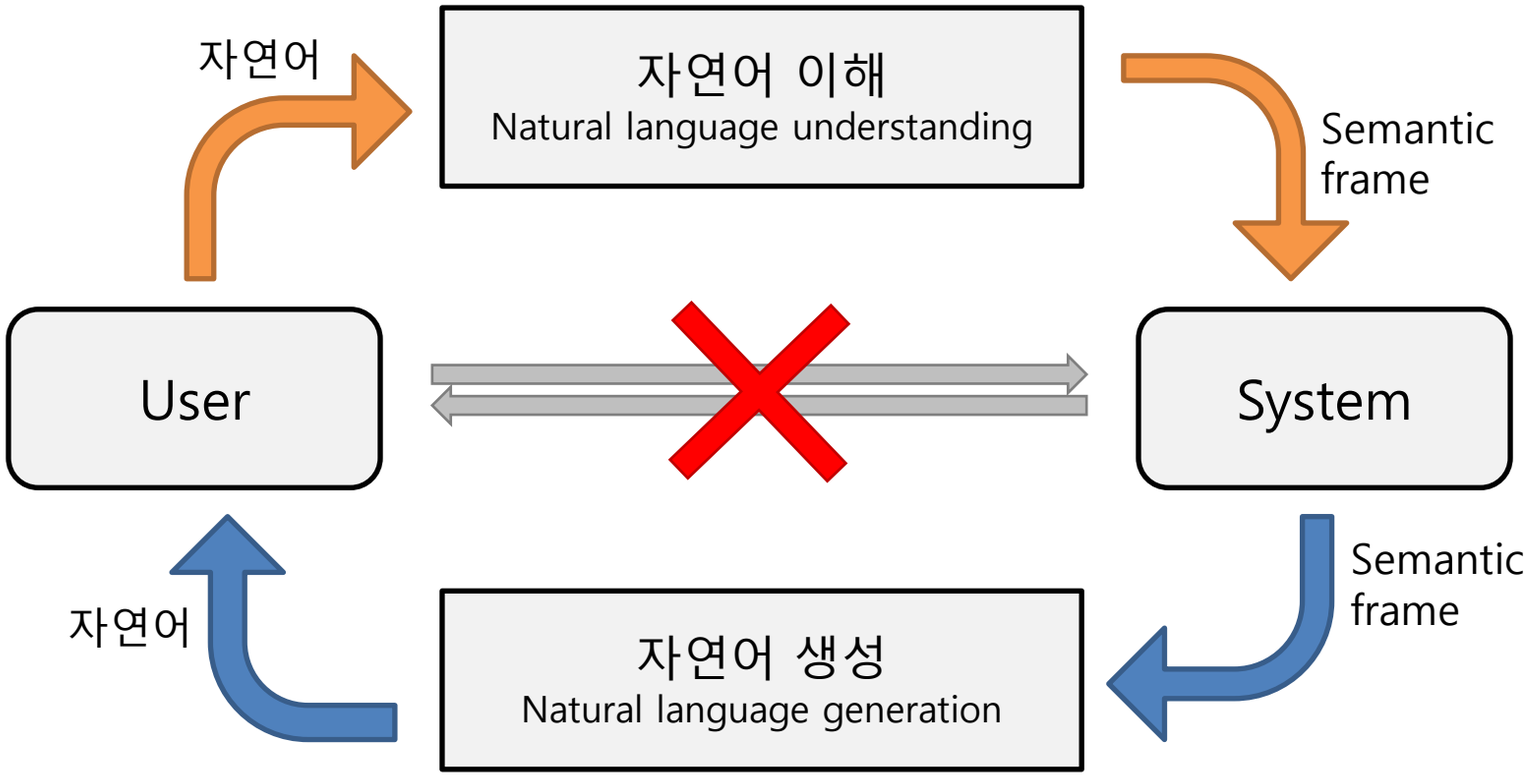


# Simple CHATBOT System

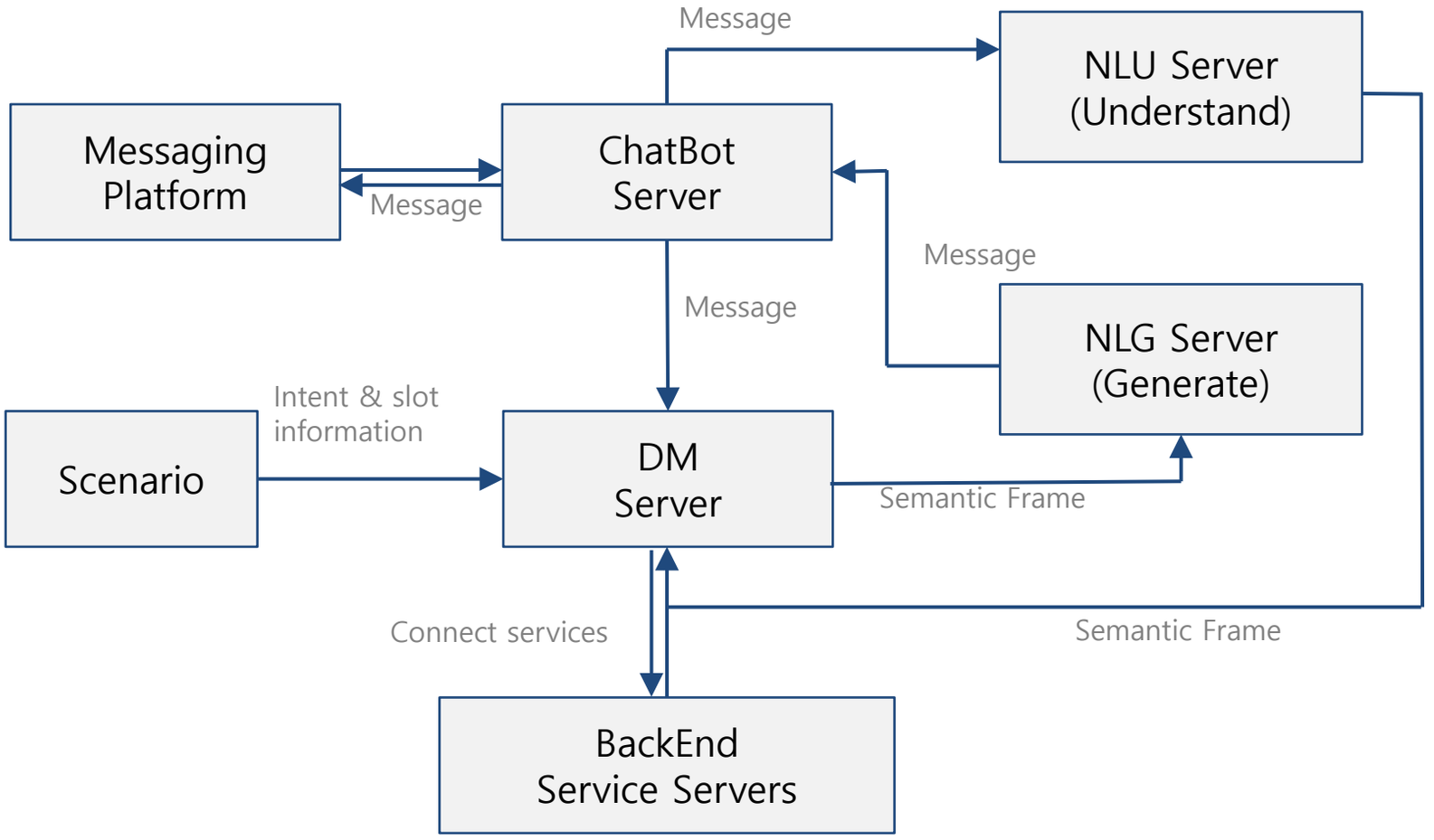
# Simple CHATBOT System



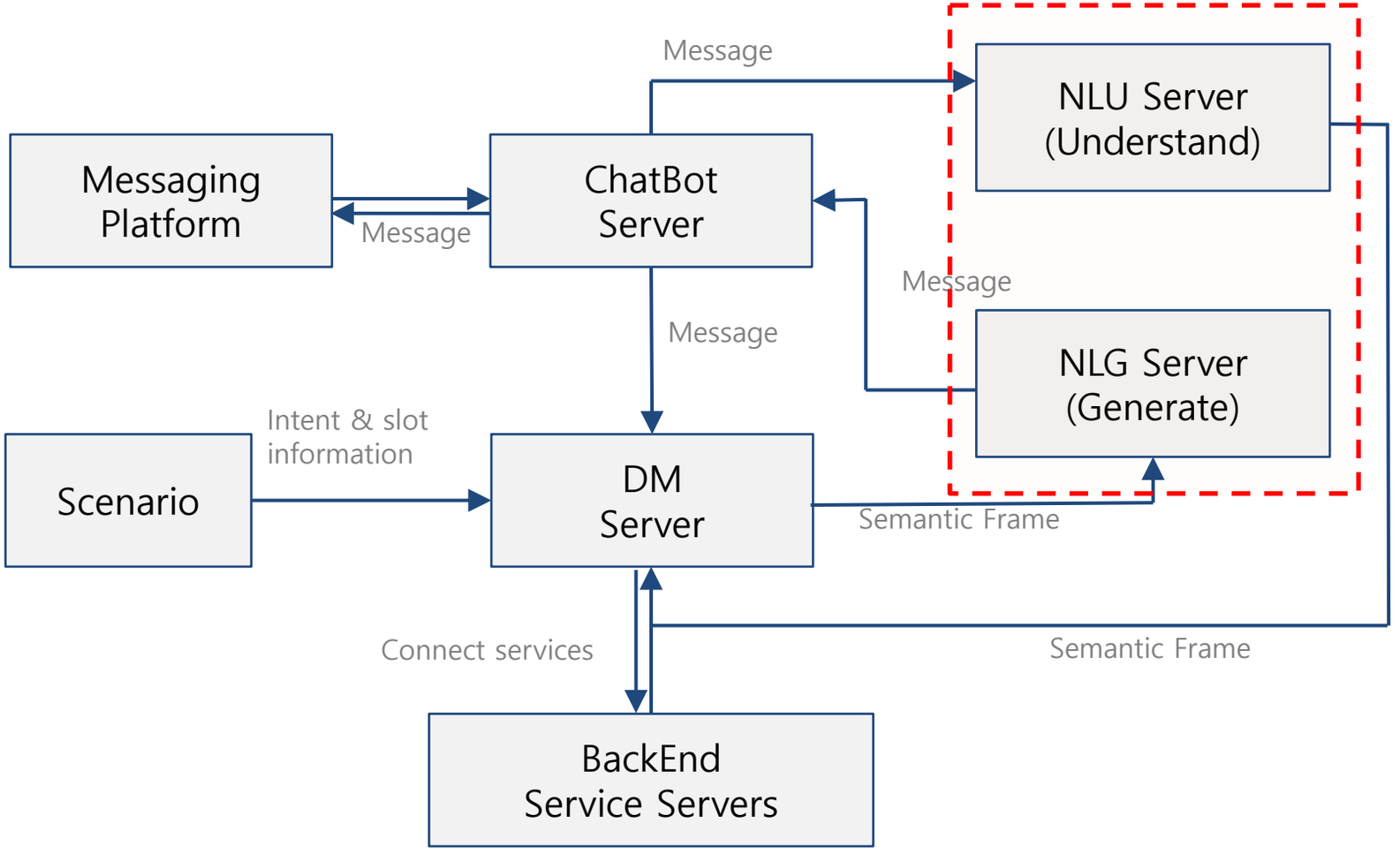
# Simple CHATBOT System



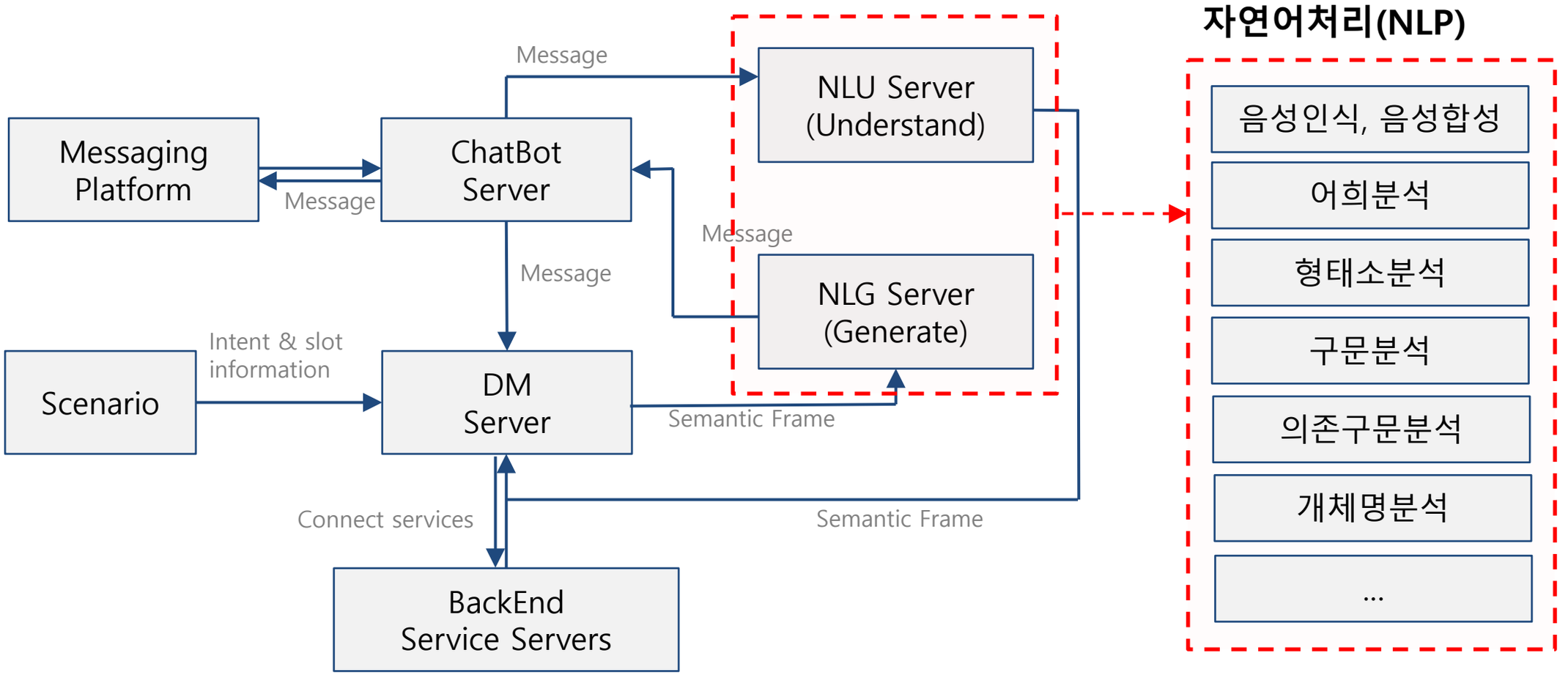
# CHATBOT System Architecture



# CHATBOT System Architecture



# CHATBOT System Architecture



# NATURAL LANGUAGE PROCESSING

# 목적

무엇을 위해 자연어 처리를 할까?

## 자연어 처리의 목적

자연어로 만들어진 모든 데이터에 대해서

### 이해하고

- 문서 이해
- 발화 이해
- 질문 이해
- …

### 응답하기

- 검색
- 추론
- 분류
- …

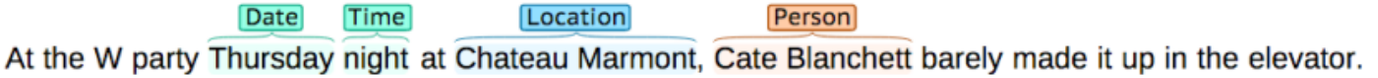
# 자연어 처리와 응용

## 응용

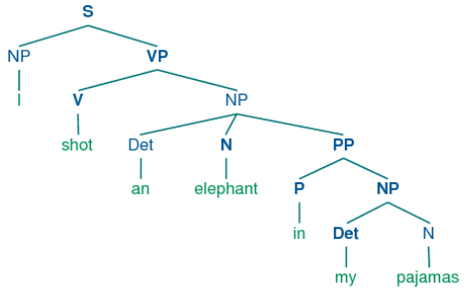
| | | | | | |
|---|---|---|---|---|---|
| 감성분석 | Q/A | 번역 | 가짜뉴스탐지 | 챗봇 | 검색 |
| 유사도분석 | 스팸탐지 | 주어복원 | 주제분류 | 관련어분석 | ... |

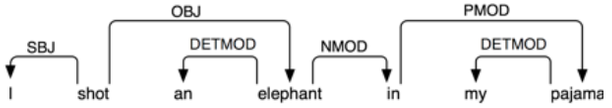
## NLP

개체명분석




구문분석
의존구문분석




형태소분석 ex) 밥/NNG+먹/VV+기/ETN 싫/VA+대/EF

정규화 ex) 채널 삼백번 틀어줘→채널 300번 틀어줘

띄어쓰기 ex) 무한도전좀보여줘→무한도전 좀 보여줘

# 문장구분(Sentence splitting), 토크나이징(Tokenizing)

**문장구분(Sentence splitting)**은 일반적으로는 개행문자(\n)나 마침표(.), 느낌표(!), 물음표(?) 등 기준으로 수행. 하지만 토픽모델링 같은 특정 알고리즘이나 방법론의 경우 문장분리를 반드시 수행하지 않아도 됨.

**토크나이징(tokenizing)**이란 문서나 문장을 분석하기 좋도록 토큰으로 나누는 작업. 영어의 경우 공백만으로도 충분히 토큰을 나눌 수 있다고 함.

```
[개행 문자가 있는 문장]
첫 번째 문장입니다.
두 번째 문장입니다.
김승우 입니다.

[개행 문자로 분리]
['첫 번째 문장입니다. ', '두 번째 문장입니다. ', '김승우 입니다.', '']

[공백으로 문자 분리]
[['첫', '번째', '문장입니다.', ''], ['두', '번째', '문장입니다.', ''], ['김승우', '입니다.'], ['']]
```

# 정규화, Text Normalization

토큰들을 좀 더 일반적인 형태로 분석해 단어수를 줄여 분석의 효율성을 높이는 작업.
영어에선 대문자를 소문자로 바꿔주는 **folding**도 이와 관련된 중요한 작업이라고 함.

| Word | stemming | lemmatization |
|---|---|---|
| Love | Lov | Love |
| Loves | Lov | Love |
| Loved | Lov | Love |
| Loving | Lov | Love |
| Innovation | Innovat | Innovation |
| Innovations | Innovat | Innovation |
| Innovate | Innovat | Innovate |
| Innovates | Innovat | Innovate |
| Innovative | Innovat | Innovative |

- **stemming**이란 단어를 축약형으로 바꿔주는 것
- **lemmatization**은 품사 정보가 보존된 형태의 기본형으로 변환

# Part-Of-Speech Tagging

**포스태깅(Part-Of-Speech Tagging)**이란 토큰의 품사정보를 할당하는 작업.
의사결정나무(Decision Trees), 은닉 마코프 모델(Hidden Markov Models), 서포트벡터머신(Support Vector Machines) 등이 활용.
조사, 어미가 발달한 한국어의 경우 정확한 분석이 매우 어려움.

→ **아버지가방에들어가신다**

| Hannanum | Kkma | Komoran | Mecab | Twitter |
|---|---|---|---|---|
| 아버지가방에 들어가 / N | 아버지 / NNG | 아버지가방에 들어가신다 / NNP | 아버지 / NNG | 아버지 / Noun |
| 이 / J | 가방 / NNG | | 가 / JKS | 가방 / Noun |
| 시ㄴ다 / E | 에 / JKM | | 방 / NNG | 에 / Josa |
| | 들어가 / VV | | 에 / JKB | 들어가신 / Verb |
| | 시 / EPH | | 들어가 / VV | 다 / Eomi |
| | ㄴ다 / EFN | | 신다 / EP+EC | |

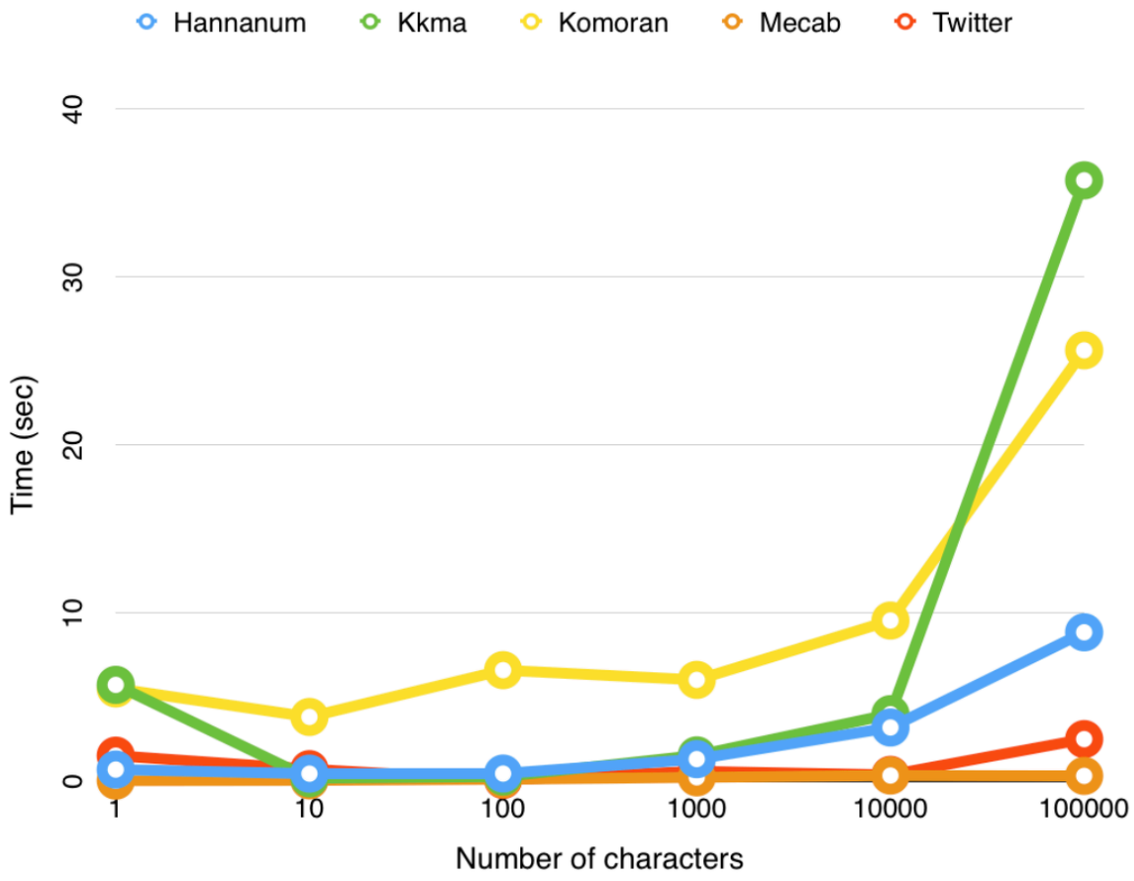
# 한국어 형태소분석기 종류

### 로딩 시간

- **Kkma**: 5.6988 *secs*
- **Komoran**: 5.4866 *secs*
- **Hannanum**: 0.6591 *secs*
- **Twitter**: 1.4870 *secs*
- **Mecab**: 0.0007 *secs*

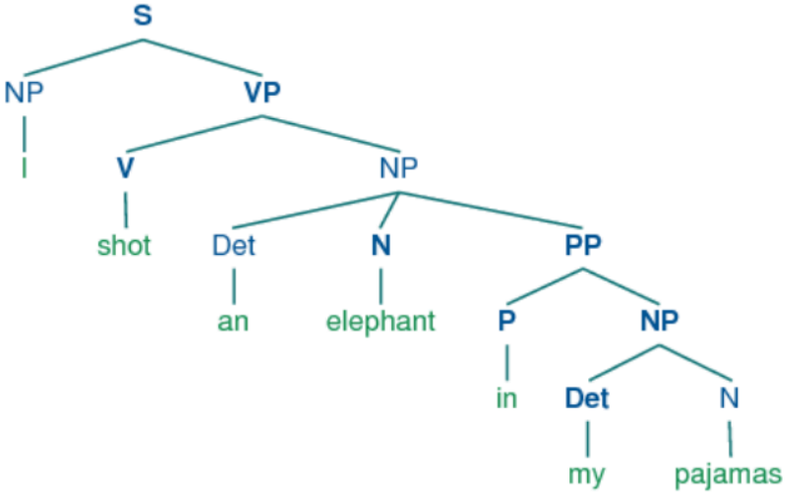
### 실행 시간

- **Kkma**: 35.7163 *secs*
- **Komoran**: 25.6008 *secs*
- **Hannanum**: 8.8251 *secs*
- **Twitter**: 2.4714 *secs*
- **Mecab**: 0.2838 *secs*



# 구문분석, Constituency & Dependency

## Constituency parsing




- 구성, 구조
- 단일 단위를 갖는 단어의 그룹 : 구(Phrase)

> 예) John loves Mary
> 동사 loves와 명사 Mary는 동사구(Verb phrase)를 구
> 명사 John과 동사구 "loves Mary"는 문장을 구성한다.

- 영어와 같이 어순이 비교적 고정적인 언어에 맞음
- 구구조문법을 통해 문장을 분석해나감

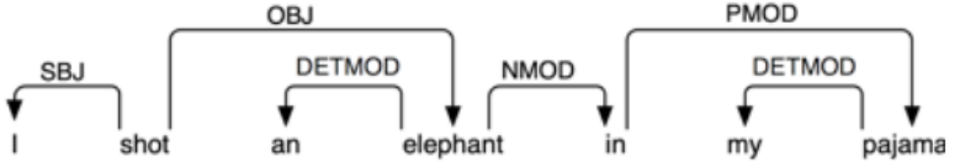
# 구문분석, Constituency & Dependency

## Dependency parsing

- 의존

> 예) John loves Mary
> "john"은 동사 loves의 주어이다. "mary"는 동사 loves의 직접목적어이다.

- 어순이 복잡한 언어에 알맞음 : scrambling language
- 한국어 같은 자유 어순을 가지거나 문장성분이 생략이 가능한 언어에서 선호됨



# 개체명분석, NER (Named Entity Recognition)

**개체명(Named Entity)**이란 문서에서 특정한 의미를 가지고 있는 단어 또는 어구를 말함
정보 검색에서 개체명은 주요 검색 대상이 되며, 질의/응답에서는 주요 질의/응답 대상이 됨
이러한 개체명을 추출하기 위해 자연어 처리 분야에서 개체명 인식 및 분류(Named Entity Recognition and Classification) 연구가 활발하 게 진행됨

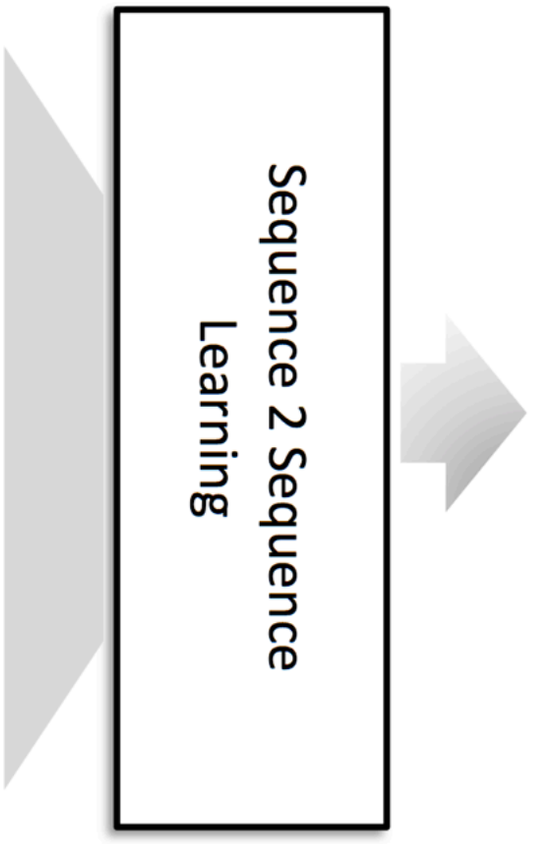
**광화문아파트**에 **피자** 배달해줘요
LC_OTHERS CV_FOOD

**문재인 대통령**이 **어젯(4일)밤 10시**부터 **30분 동안** 전화 통화를 갖고 **평창 동계올림픽**을…
PS_NAME CV_POSITION DT_DAY TI_HOUR TI_DURATION LC_CITY EV_SPORTS

# Using Deep Learning

| 단어 |
|---|
| West |
| Indian |
| all-around |
| Phil |
| Simons |
| took |
| four |
| for |
| 38 |
| on |
| Friday |

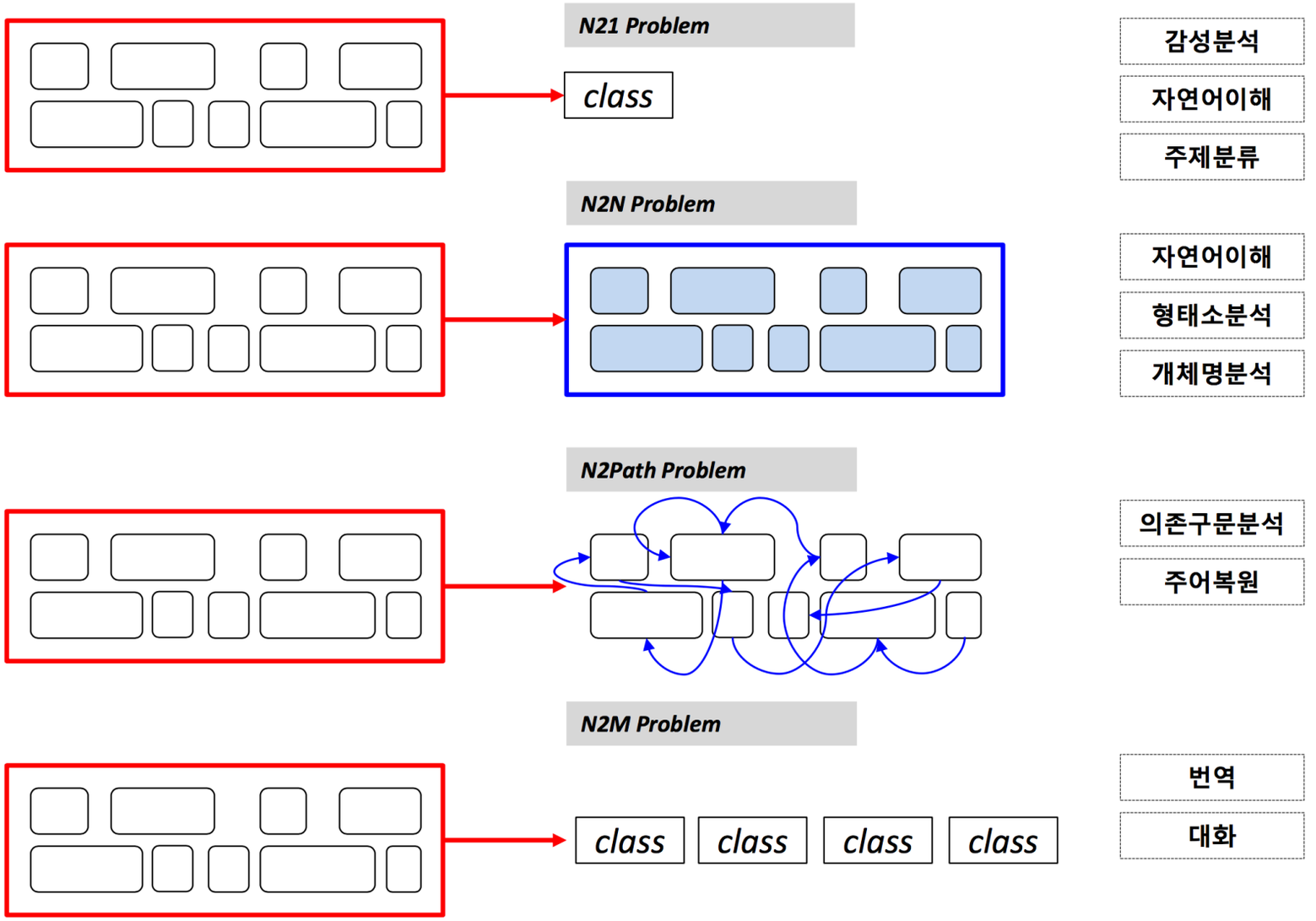
x



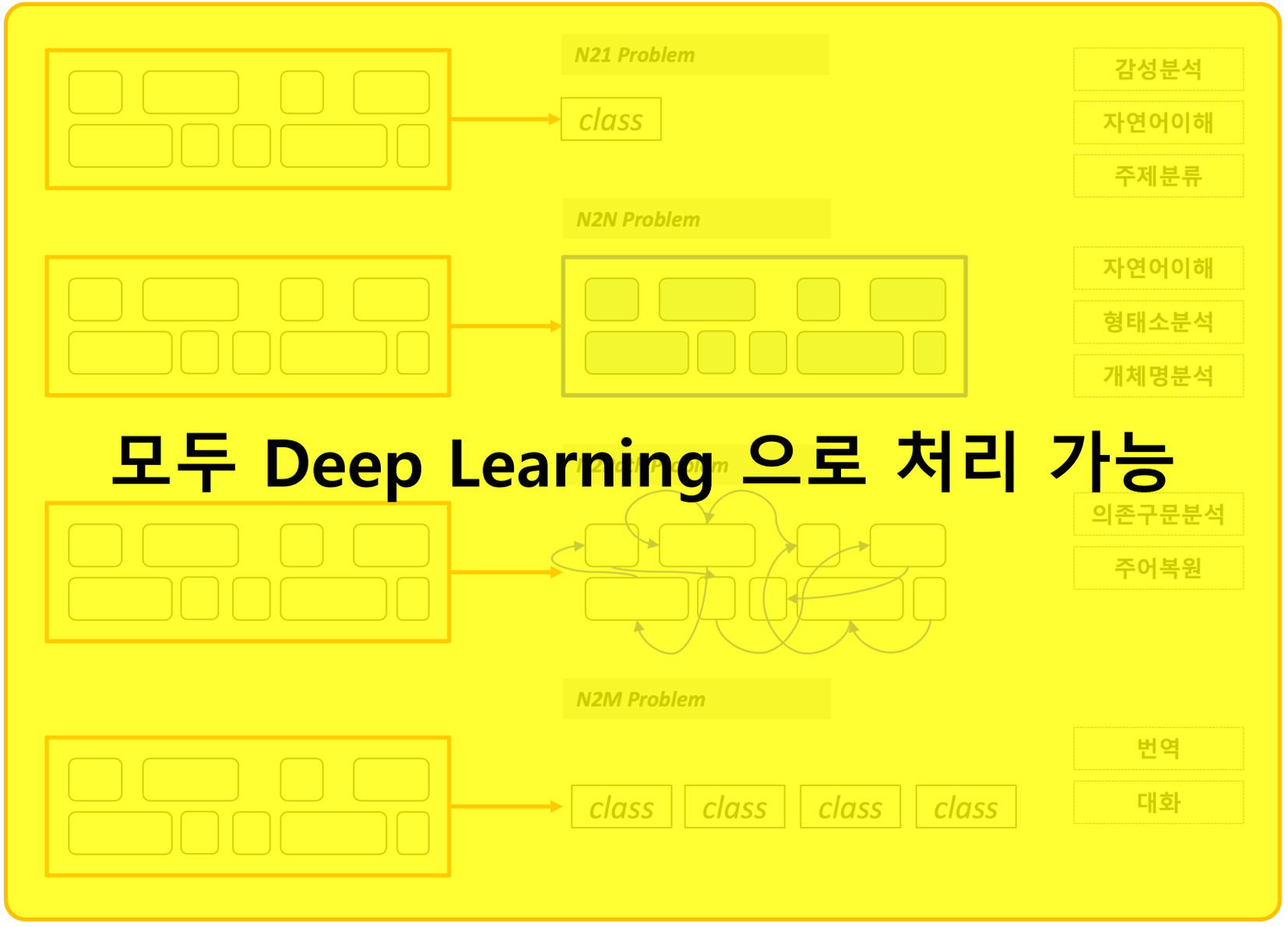

| 형태소분석 | 구문분석 | 개체명분석 |
|---|---|---|
| NNP | B-NP | B-MISC |
| NNP | I-NP | I-MISC |
| NN | I-NP | O |
| NNP | I-NP | B-PER |
| NNP | I-NP | I-PER |
| VBD | B-VP | O |
| CD | B-NP | O |
| IN | B-PP | O |
| CD | B-NP | O |
| IN | B-PP | O |
| NNP | B-NP | O |

y

# Problem Abstraction



# Problem Abstraction

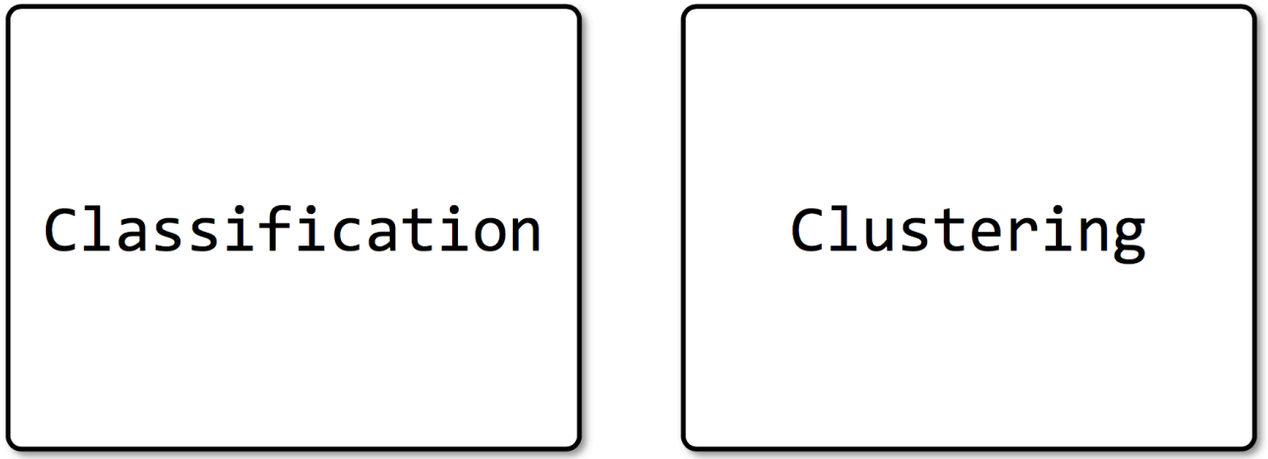


# MACHINE LEARNING REVIEW

# 어떤 형태로 풀 것인가?

Deep Learning 에서 인공지능 문제를 어떻게 풀어나가는지를 살펴봄으로써 Deep Learning 에 대한 이해도를 높임

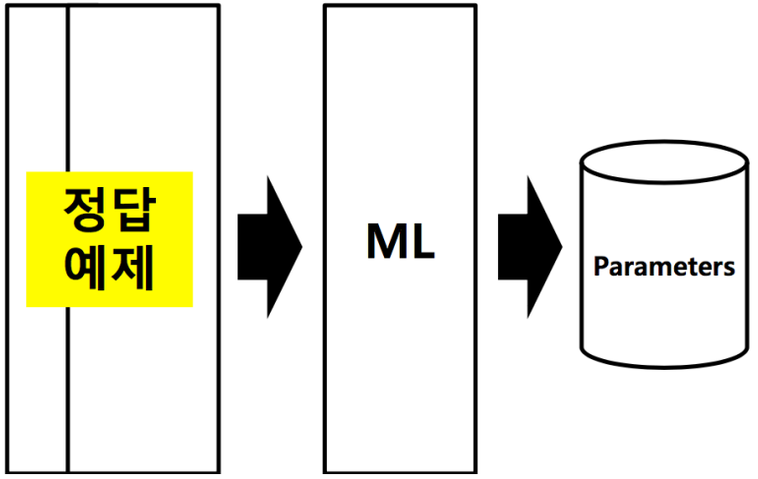
## Modern AI Problem Formulation




대부분의 문제를 **Classification** 혹은 **Clustering** 문제로 바라보고 시도함

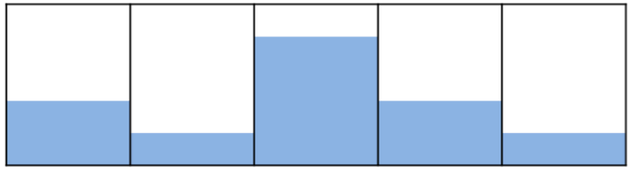
# Modern AI

## Supervised Learning




① 예제문제를풀게하고
② 정답과비교하여 잘 맞추는 방향으로 학습

## Classification



① 정해진Class에Score를부여
② 가장높은Scoring를가진 Class 선택

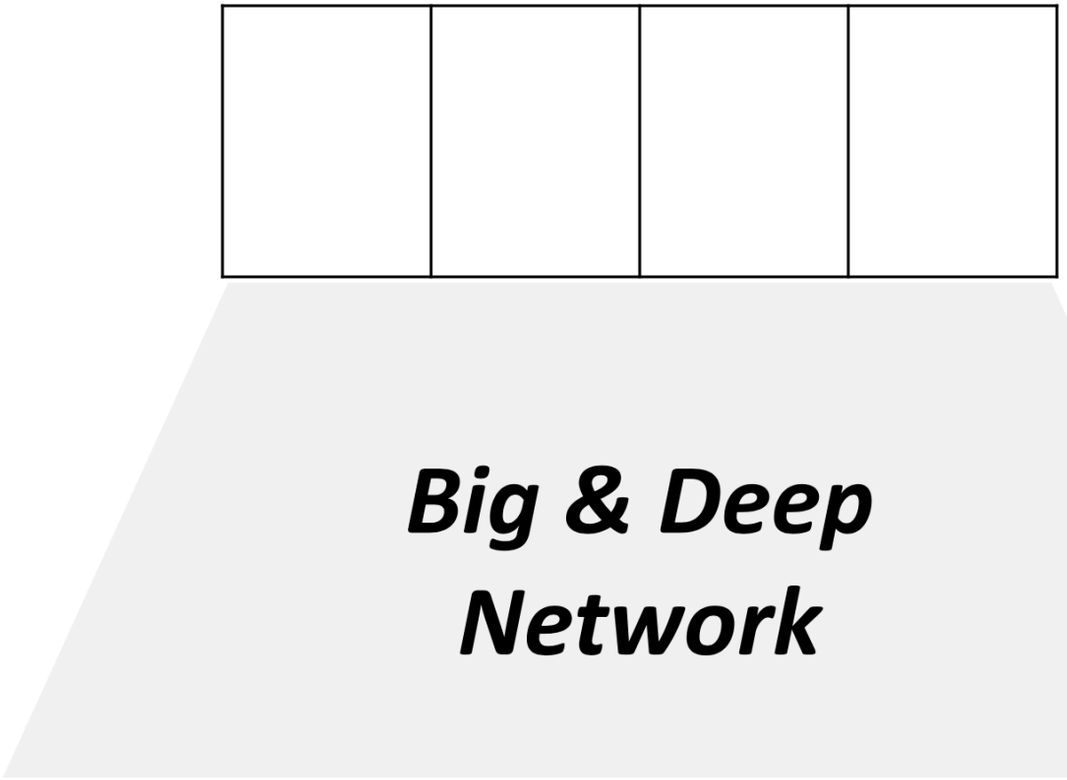
## Example

# 감성분석

- 다음에 또 가려 구요! **Positive**
- 이게 좀 비좁은 느낌을 가져다 줄지도! **Negative**
- 역시나 비싼 호텔인가 싶었어요. **Neutral**
- 무료 Wi-Fi **Objective**

**Class 설계 (4개 Class)**

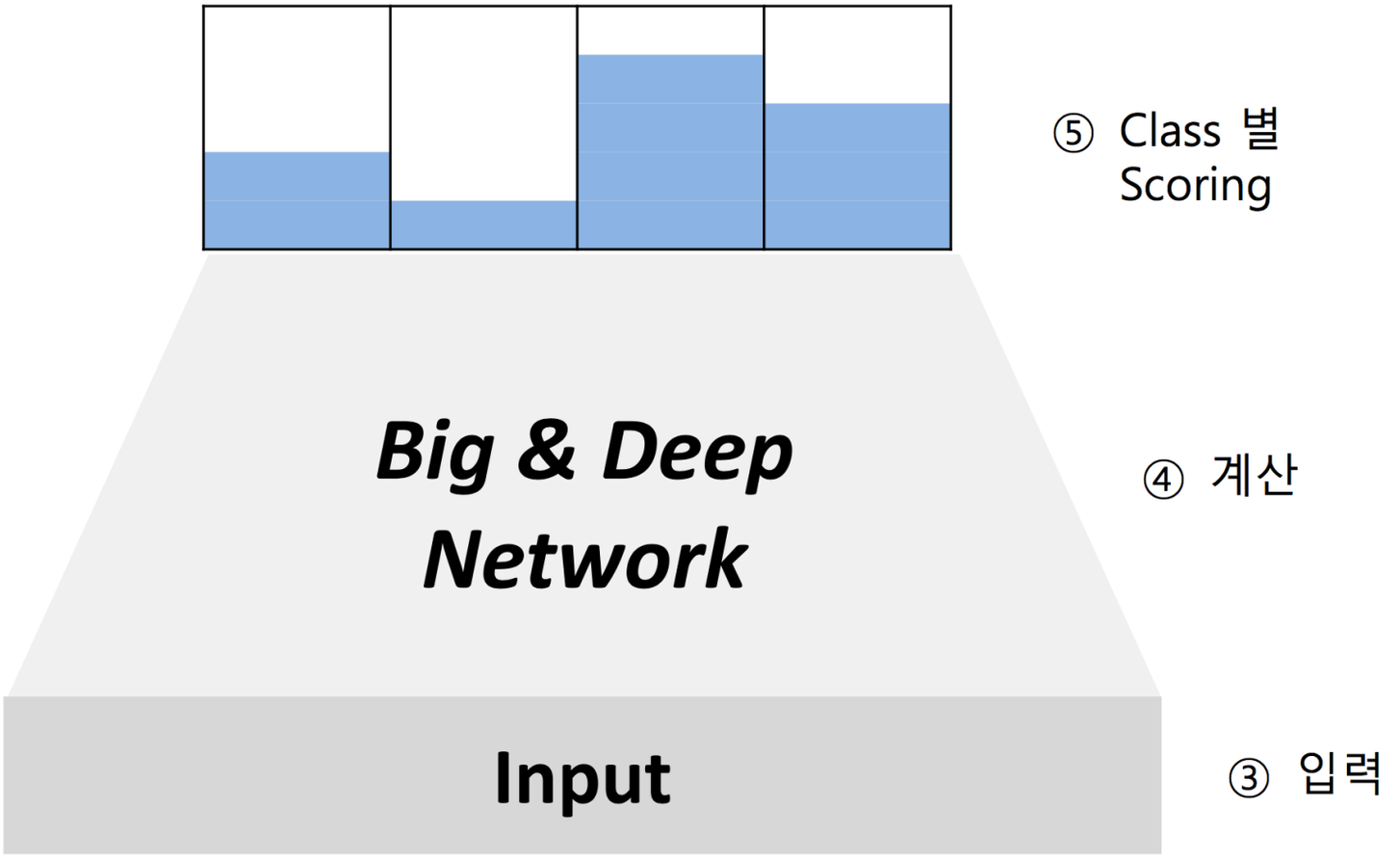
# Classification @ Deep Learning



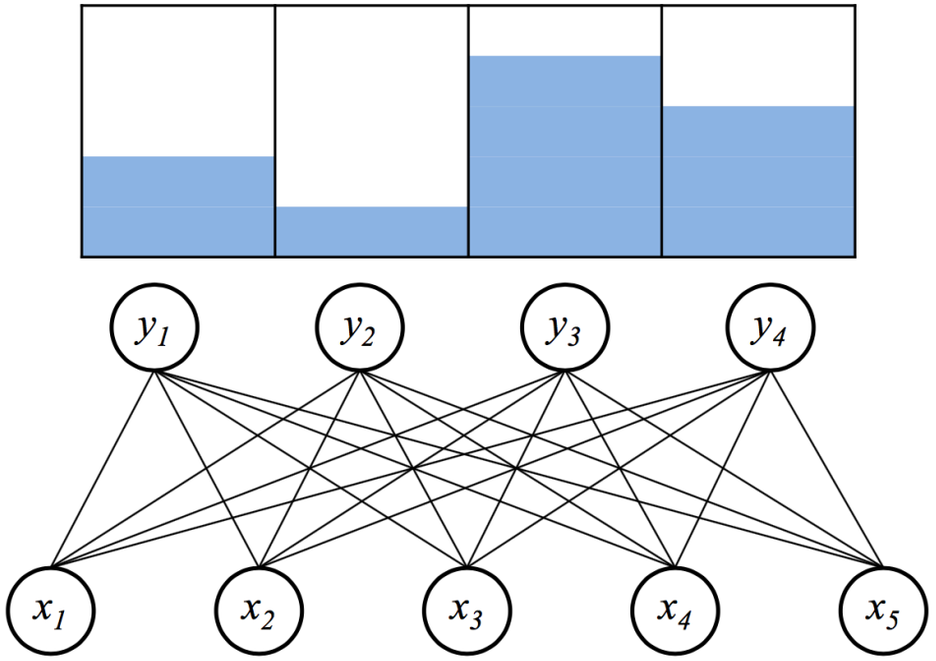

① N 개 Class 정의

② Network 구성

# Classification @ Deep Learning



# Scoring

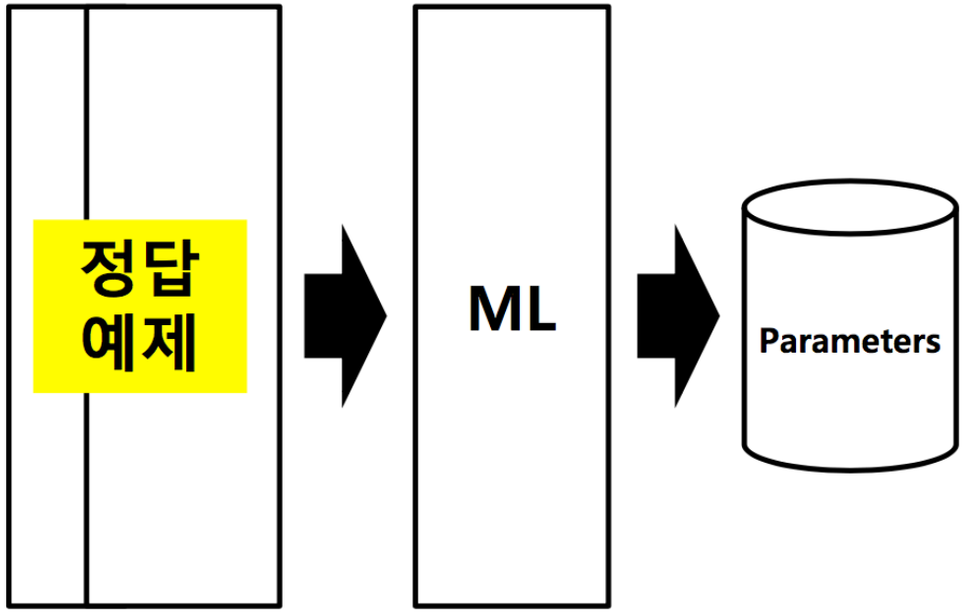



$$y_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + W_{14}x_4 + W_{14}x_4 + b_1)$$
$$y_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + W_{24}x_4 + W_{25}x_5 + b_2)$$
$$y_3 = f(W_{31}x_1 + W_{32}x_2 + W_{33}x_3 + W_{34}x_4 + W_{35}x_5 + b_3)$$
$$y_4 = f(W_{41}x_1 + W_{42}x_2 + W_{43}x_3 + W_{44}x_4 + W_{45}x_5 + b_4)$$

**Weighted Sum**

## Learning Problem

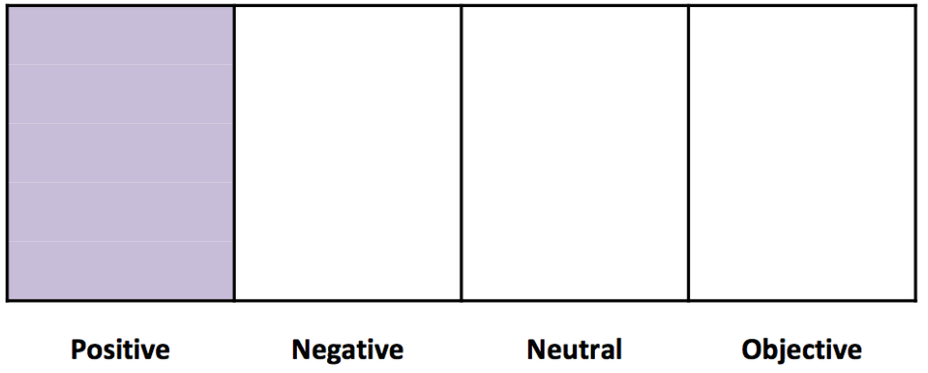
어떻게 정답을 맞추는 방향으로 학습을 시킬 수 있을까?



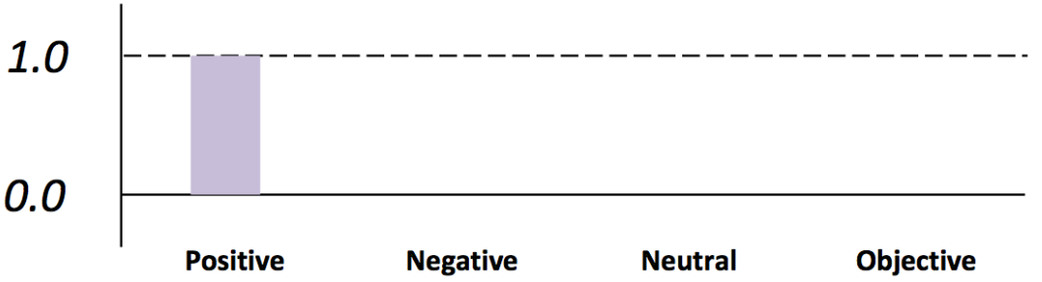

① Reference Representation
② Scoring Normalization
③ Cost Function Design
④ Parameter Update

## Reference Data Representation

정답을 어떻게 표현할 수 있을까?



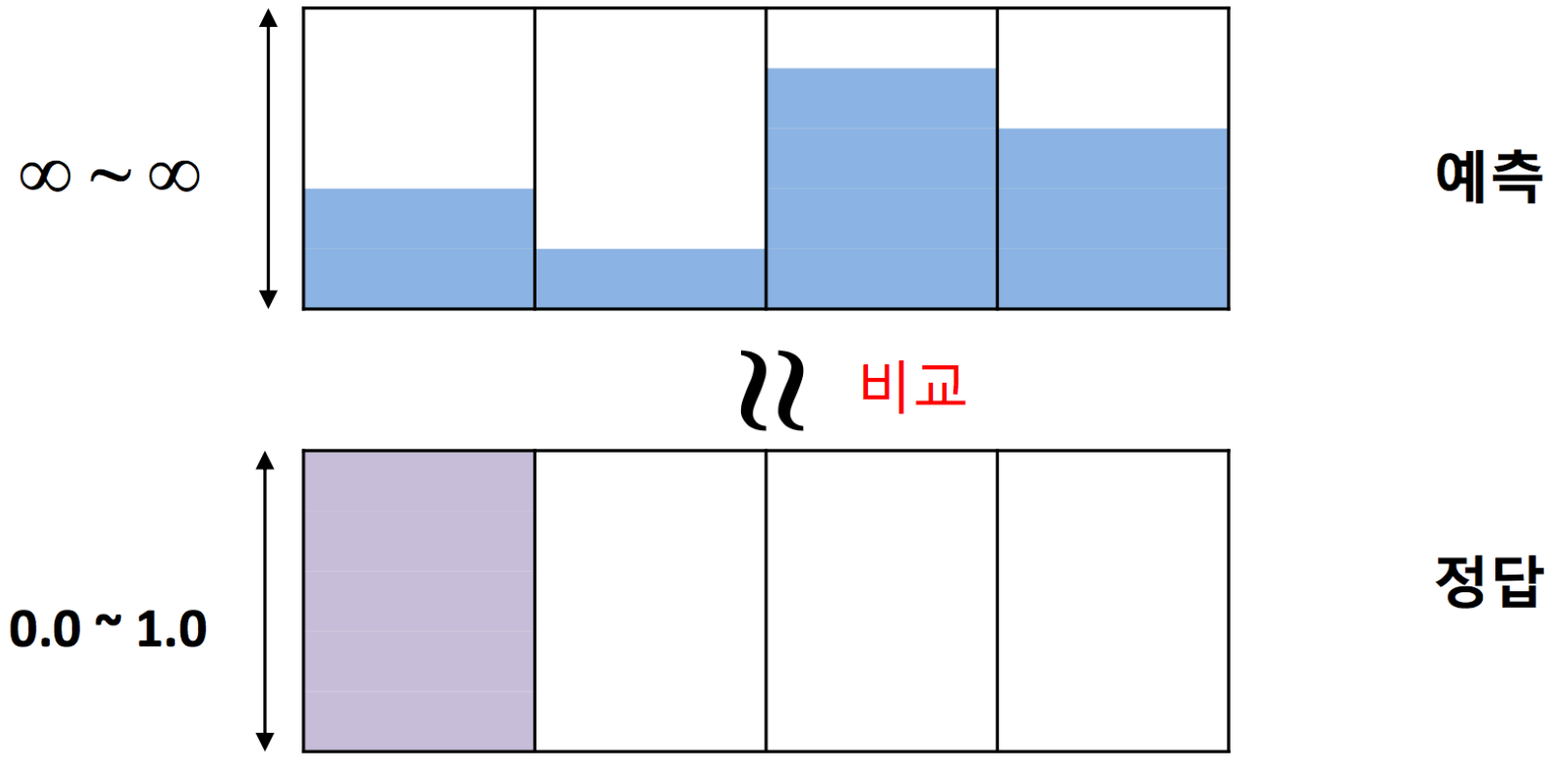

***One-Hot Representation***




***1.0 for reference class***

[1, 0, 0, 0]

## Score Normalization

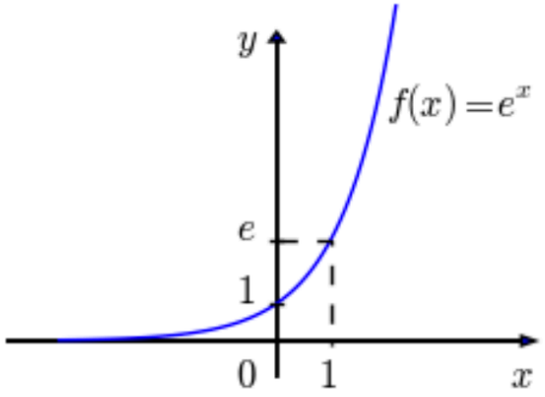



예측과 정답을 서로 비교하기 위해서 서로 같은 **Scale** 의 값이어야 한다.

# Score Normalization

**Softmax Function**

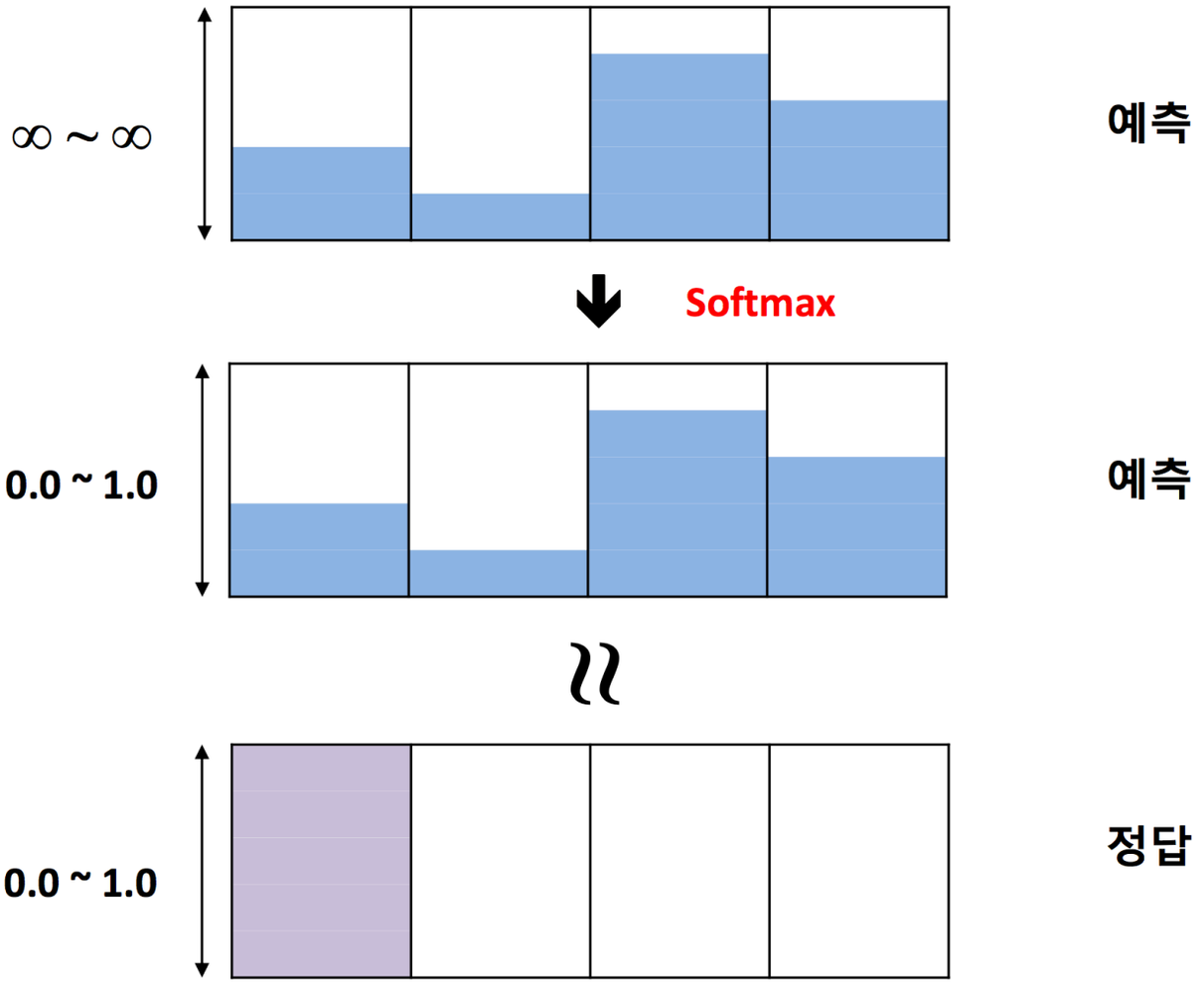
$$S_j = \frac{e^{z_j}}{\sum_{n=1}^{N} e^{z_n}} \text{ for } j = 1, \ldots, N$$



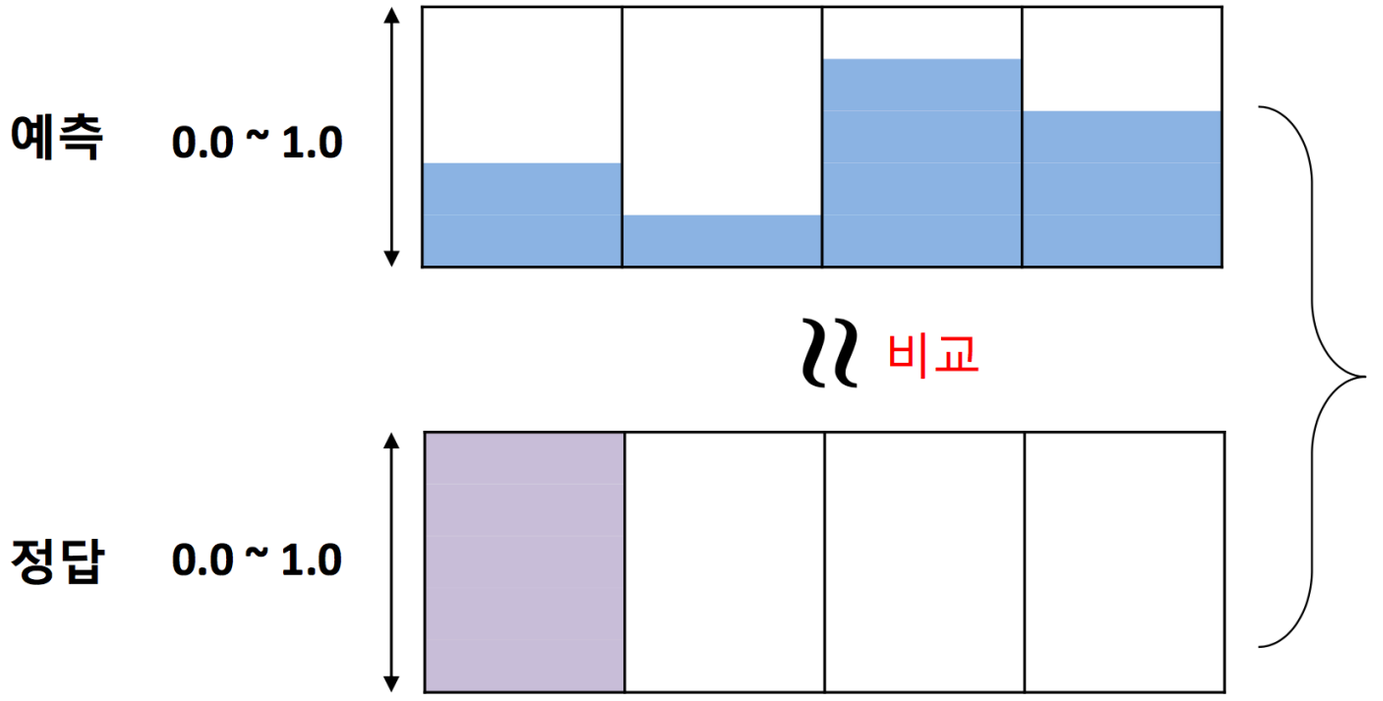

0.0 ~ 1.0 의 값으로 변환

어떤 값이든 양수 값으로 바꿔주고
큰 값일수록 더욱 크게 만들어줌

# Score Normalization by Softmax



# Cost Function

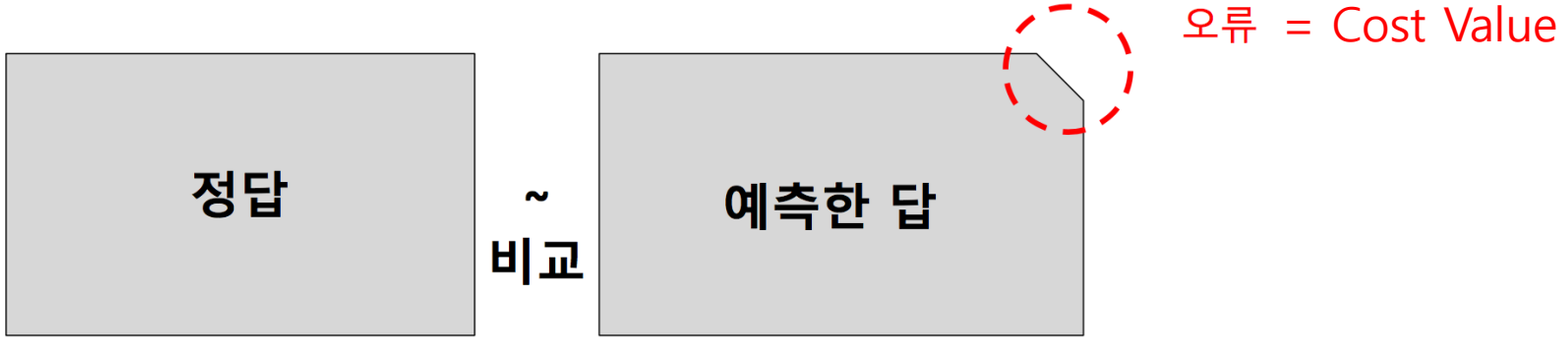



**두 점수들 간의 차이를 어떻게 수치화 할 수 있을까?**

**Cross Entropy**

## Parameter Update

오류를 줄이기 위해선 어떻게 해야하나?

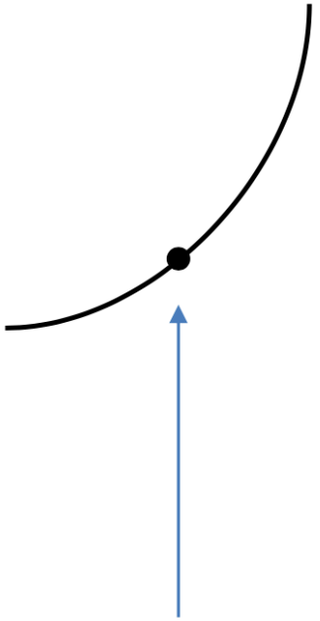



**오류가 작아지는 방향으로 parameter 를 update !!**

오류가 작아지는 방향이란 어느 쪽인가?
얼마나 고쳐야 오류를 작아지게 할 수 있을까?

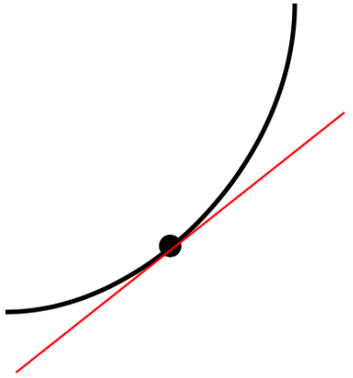
# 방향을 결정하는 방법 (1)

이 곡선이
전체의 오류를
표현한다고 하면



이 지점에서의 오류가 작아지는
방향을 결정해야 한다.

# 방향을 결정하는 방법 (2)



이 지점에서의 기울기 방향을 구해서, 기울기가 작아지는 방향으로 간다면, 오류를 작게 할 수 있을 것

**“미분”**
**“기울기 = Gradient”**

# [참고] Gradient Descent

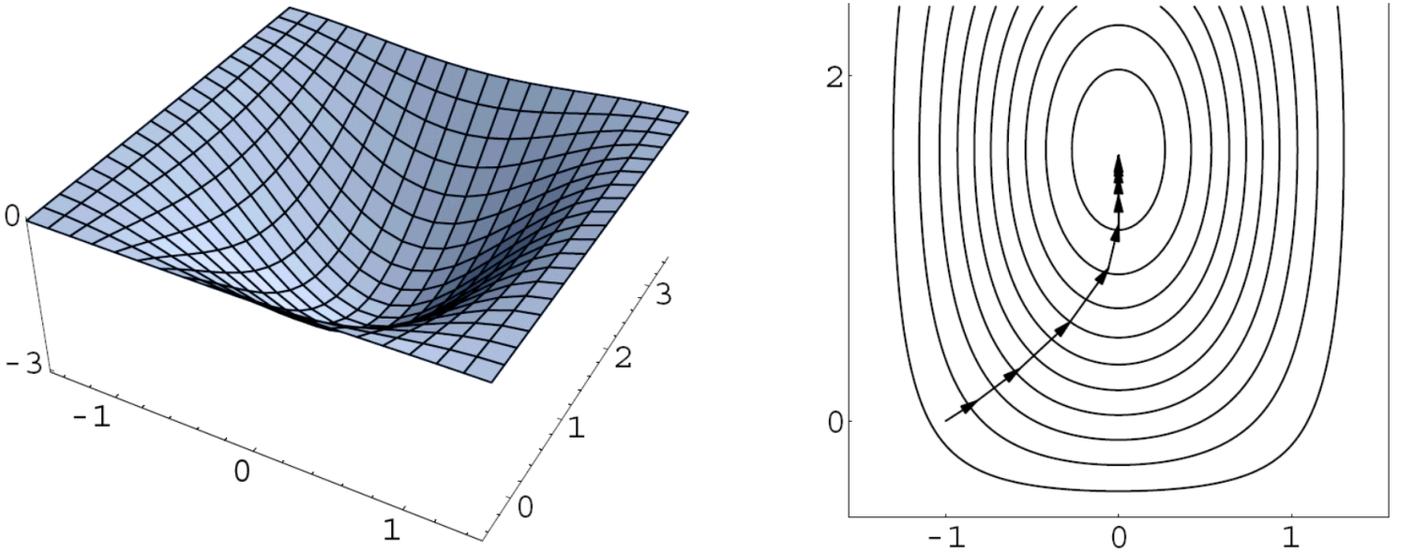
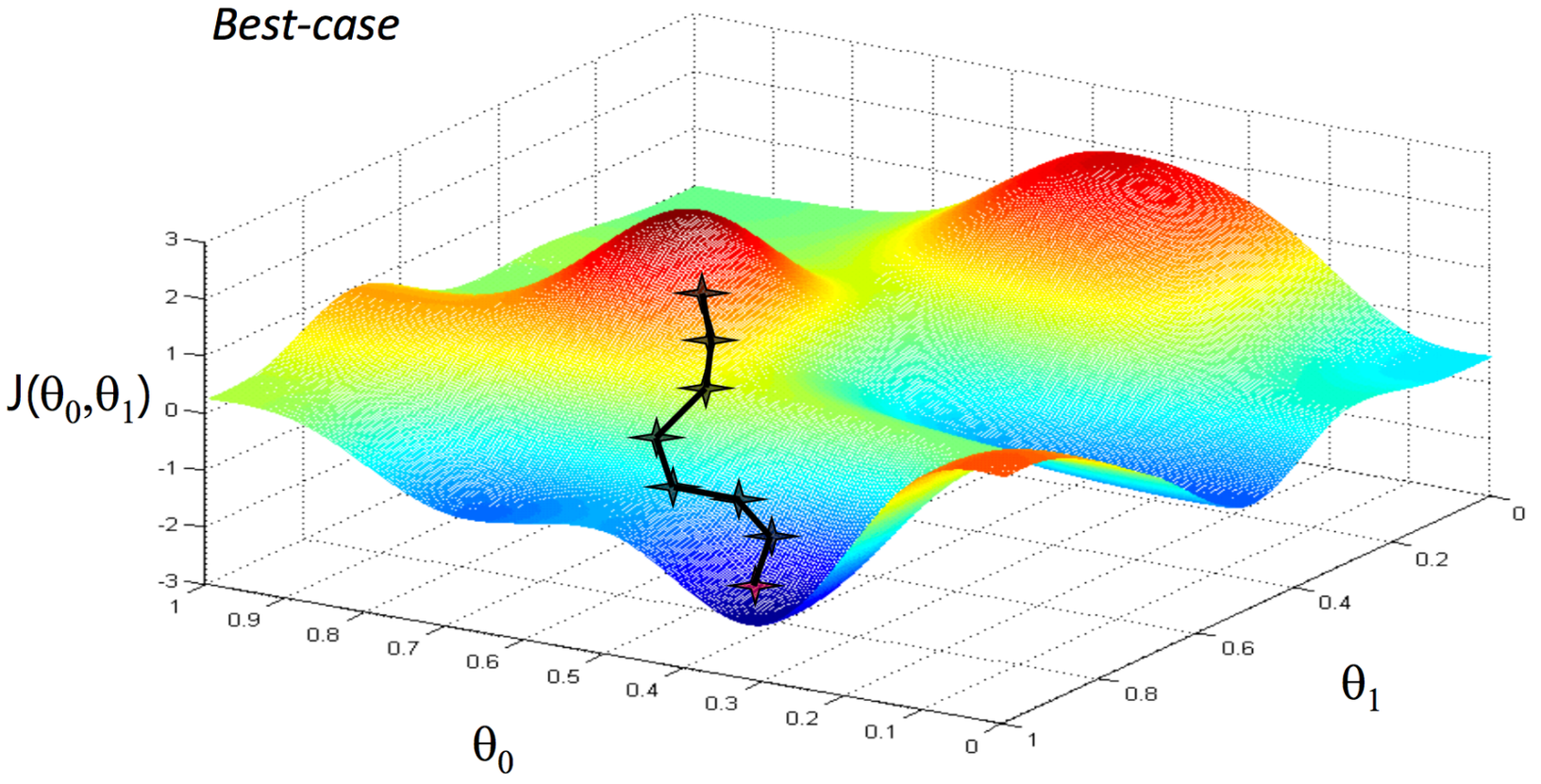


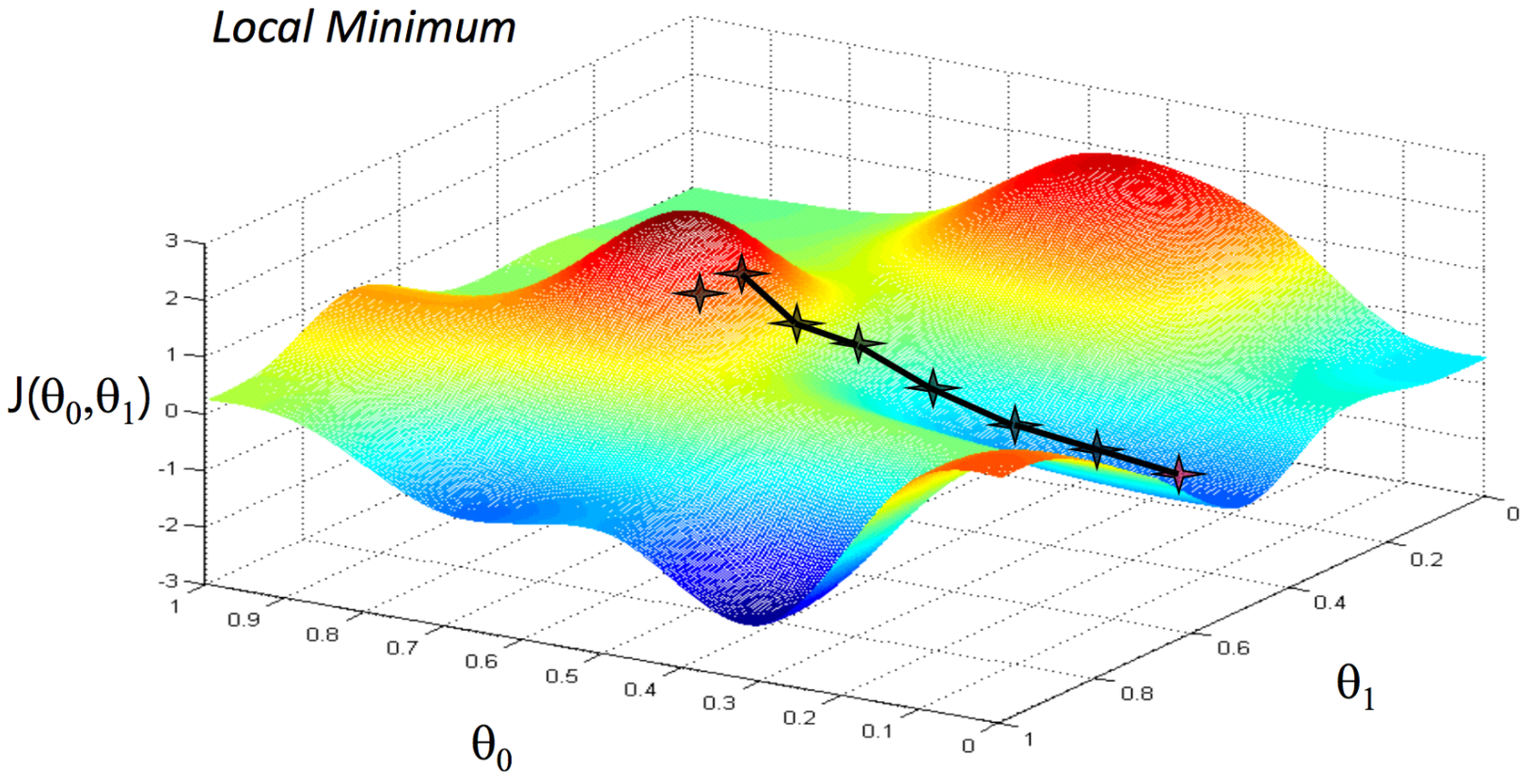

**Figure 4.3:** Visualization of the gradient descent on a two-dimensional error function. We move forward in the opposite direction of $g$, i.e. with the steepest descent towards the lowest point, with the step width being proportional to $|g|$ (the steeper the descent, the faster the steps). On the left the area is shown in 3D, on the right the steps over the contour lines are shown in 2D. Here it is obvious how a movement is made in the opposite direction of $g$ towards the minimum of the function and continuously slows down proportionally to $|g|$. **Source**: `http://webster.fhs-hagenberg.ac.at/staff/sdreisei/Teaching/WS2001-2002/PatternClassification/graddescent.pdf`

A brief introduction to neural network, David Kriesel, dkriesel.com

# [참고] Gradient Descent Illustration



# [참고] Gradient Descent Illustration



# [참고] Minima and Maxima

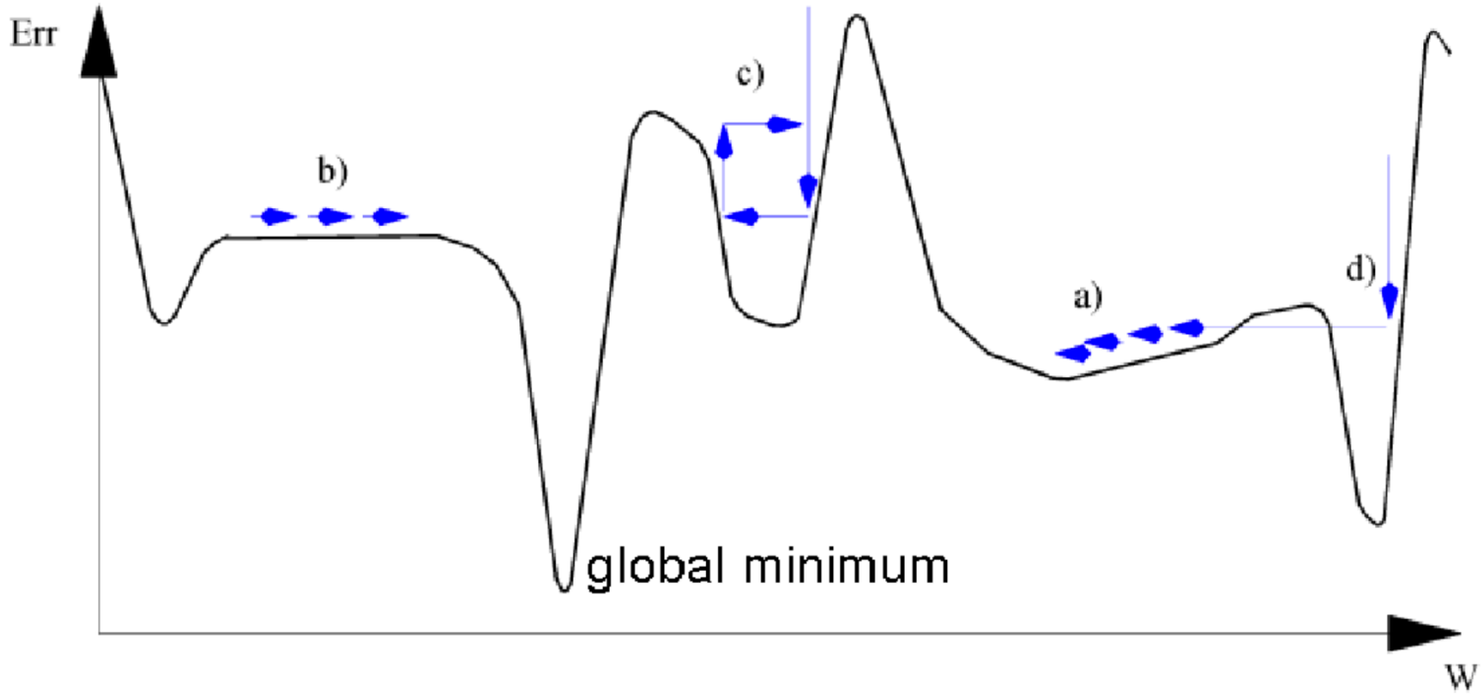



**Figure 4.4:** Possible errors during a gradient descent: a) Detecting bad minima, b) Quasi-standstill with small gradient, c) Oscillation in canyons, d) Leaving good minima.

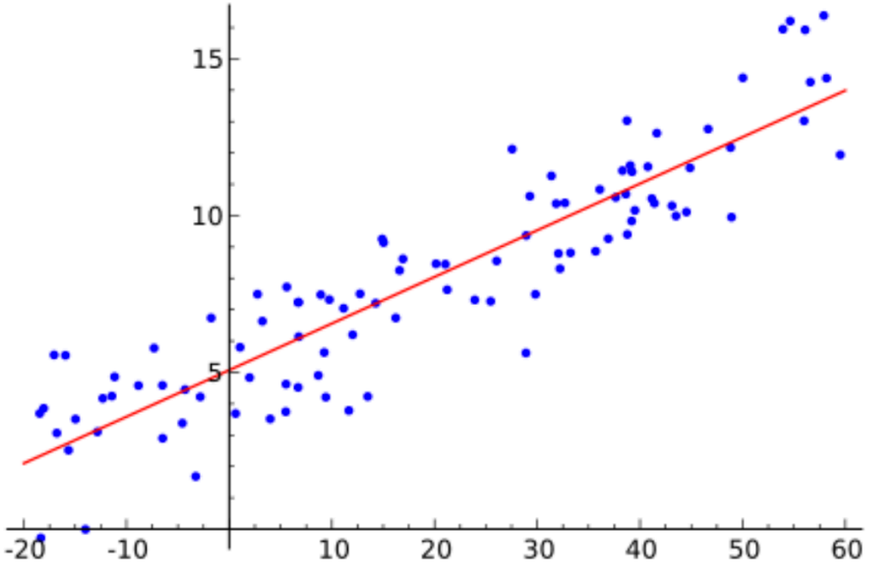
# Parameter Update (Weight Optimization)

**“Backpropagation Algorithm”**

| | |
|---|---|
| 최종 결과물을 얻고 | **Feed Forward and Prediction** |
| 그 결과물과 우리가 원하는 결과물 과의 차이점을 찾은 후 | **Cost Function** |
| 그 차이가 무엇으로 인해 생기는 지 | **Differentiation (미분)** |
| 역으로 내려가면서 추정하여 | **Back Propagation** |
| 새로운 Parameter 값을 배움 | **Weight Update** |

## Parameter Update

몇 개의 ***Example*** 을 살펴보고, ***model*** 을 ***update*** 할 것인가? **"Mini-batch"**




**Parameter Update 를 한번**
**=**
**선을 다시 한번 새로 긋는것**

↓

**모든 점(훈련 예제)에**
**잘 적용되도록**

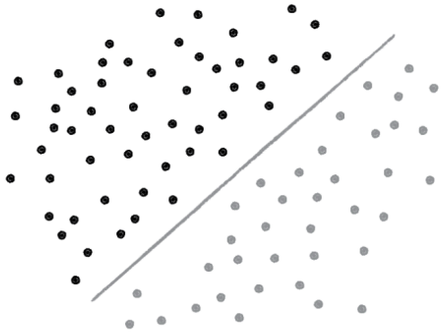
↓

$$\acute{\theta} = \theta + \alpha\Delta$$

Learning rate　　Update factor

https://en.wikipedia.org/wiki/Simple_linear_regression#/media/File:Linear_regression.svg

# Limitation of Linearity (1)



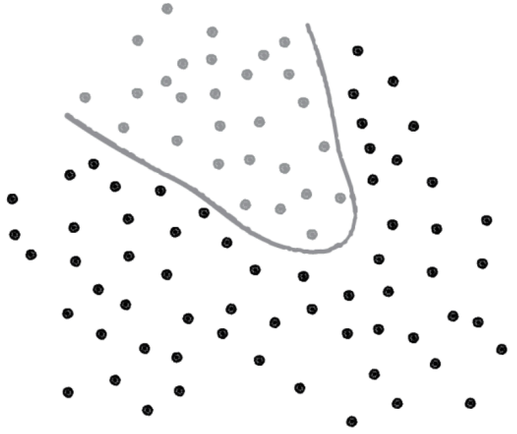
***Linearly Separable!***

AND

| | T | F |
|---|---|---|
| T | T | F |
| F | F | F |

OR

| | T | F |
|---|---|---|
| T | T | T |
| F | T | F |



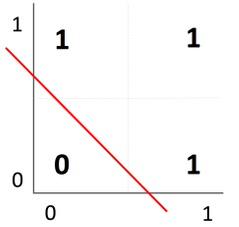
***Not Linearly Separable!***

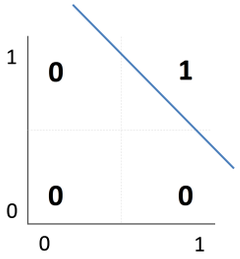
XOR

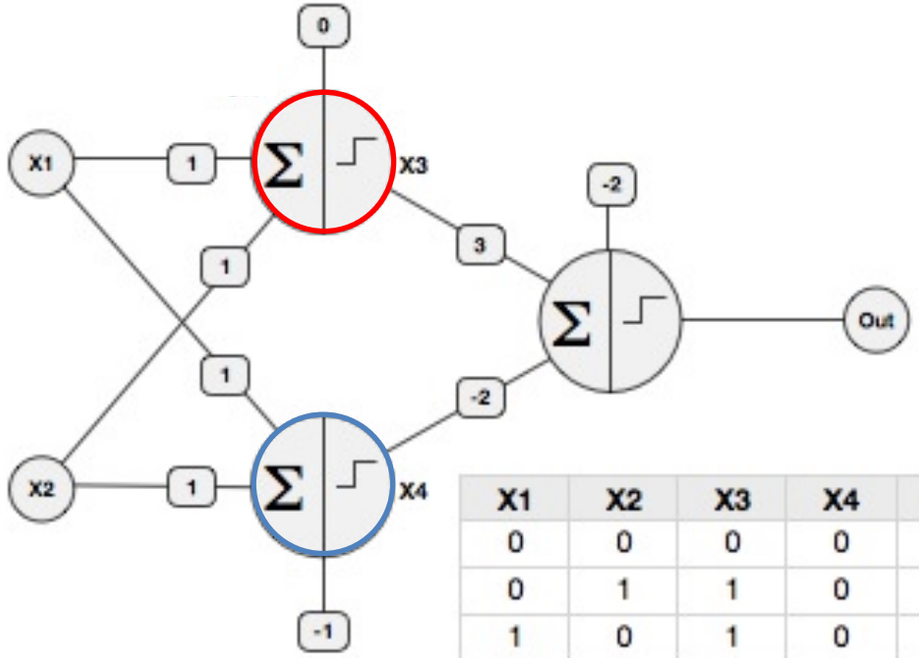
| | T | F |
|---|---|---|
| T | F | T |
| F | T | F |

우리 일상의 대부분의 문제는 **Non-Linear** 문제

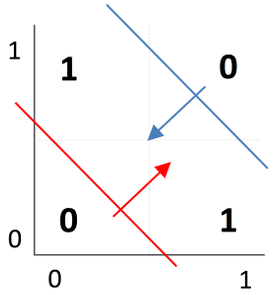
# Limitation of Linearity (2)




| X1 | X2 | X3 | X4 | Out |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

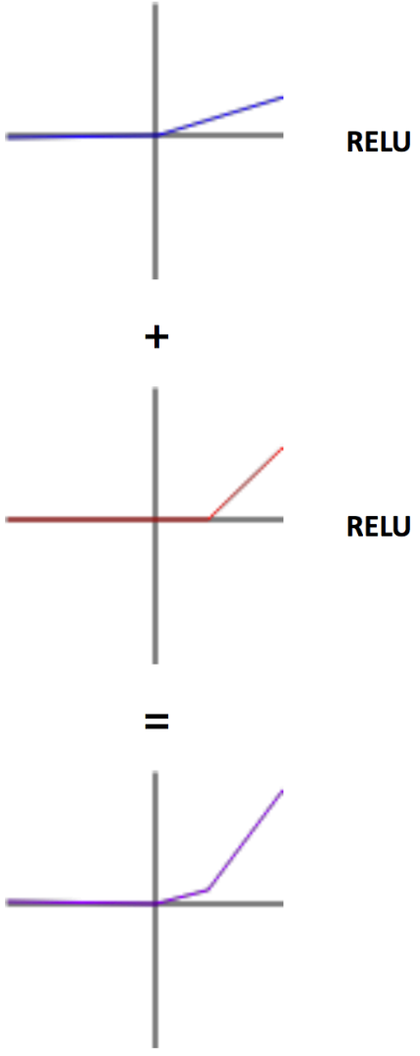



**Multiple Linear Layer** 는 **Single Linear Layer** 로 표현 가능
**Linear** 조합으로는 **Multiple Layer** 의 장점을 살릴 수 없음

# Limitation of Linearity (3)

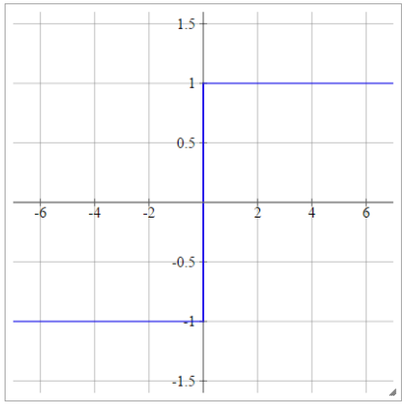
Linear Combination


Non-Linear Combination

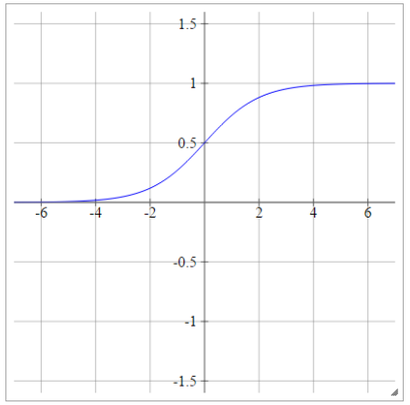


# Activation Functions

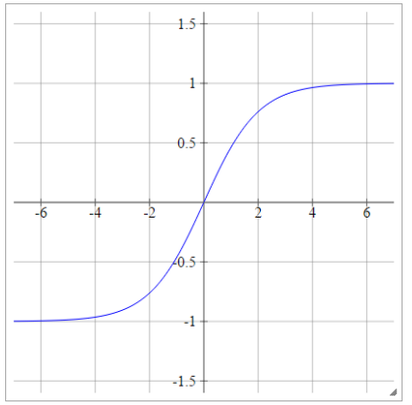
- **sign:** $f_g(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0 \end{cases}$
- **sigmoid:** $f_s(z) = \frac{1}{1+e^{-z}}, \ 0 \leq f \leq 1$
- **tanh:** $f_t(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \ -1 \leq f \leq 1$
- **ReLU:** $f_R(z) = max(0, z), \ 0 \leq f \leq \infty$



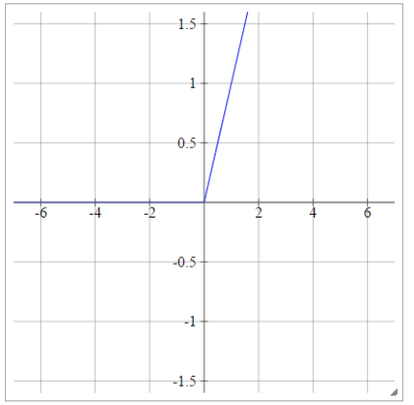

**sign**




**sigmoid**

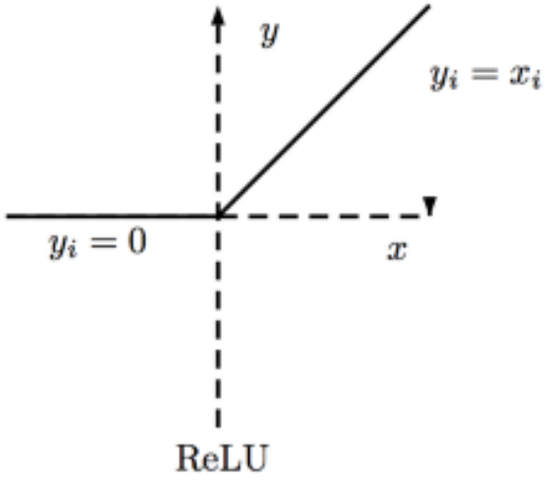



**tanh**

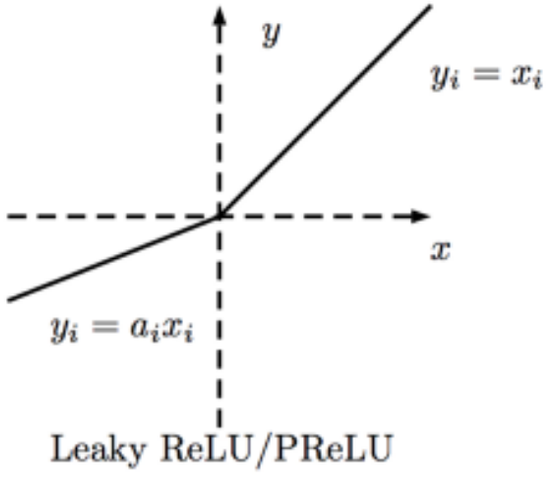



**ReLU**

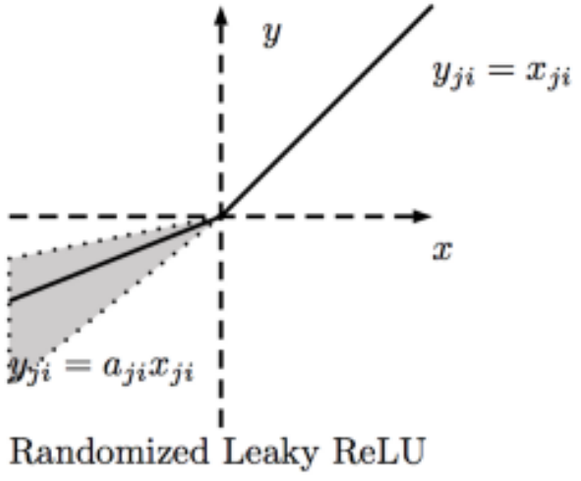
# Activation Functions



ReLU



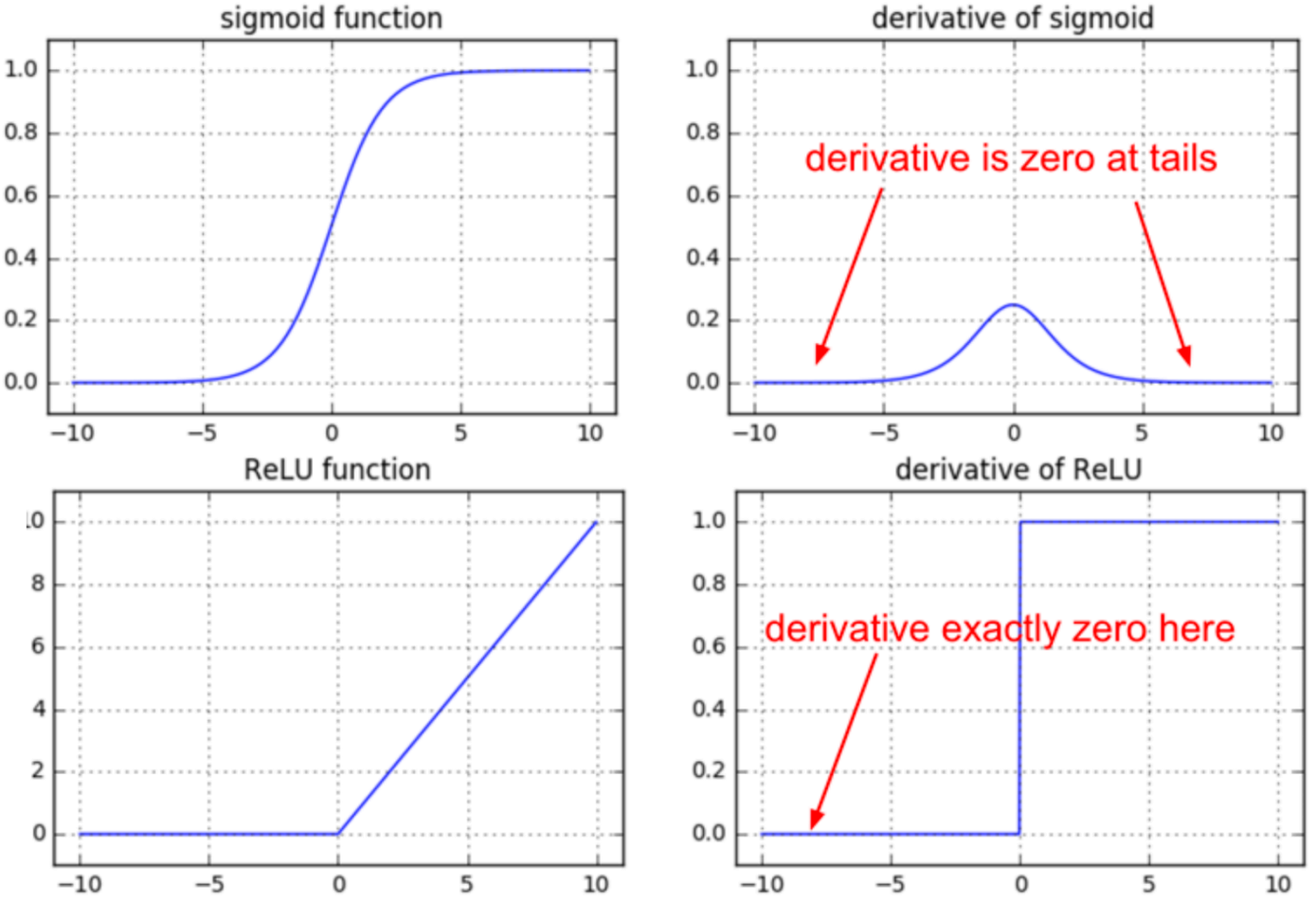
Leaky ReLU/PReLU



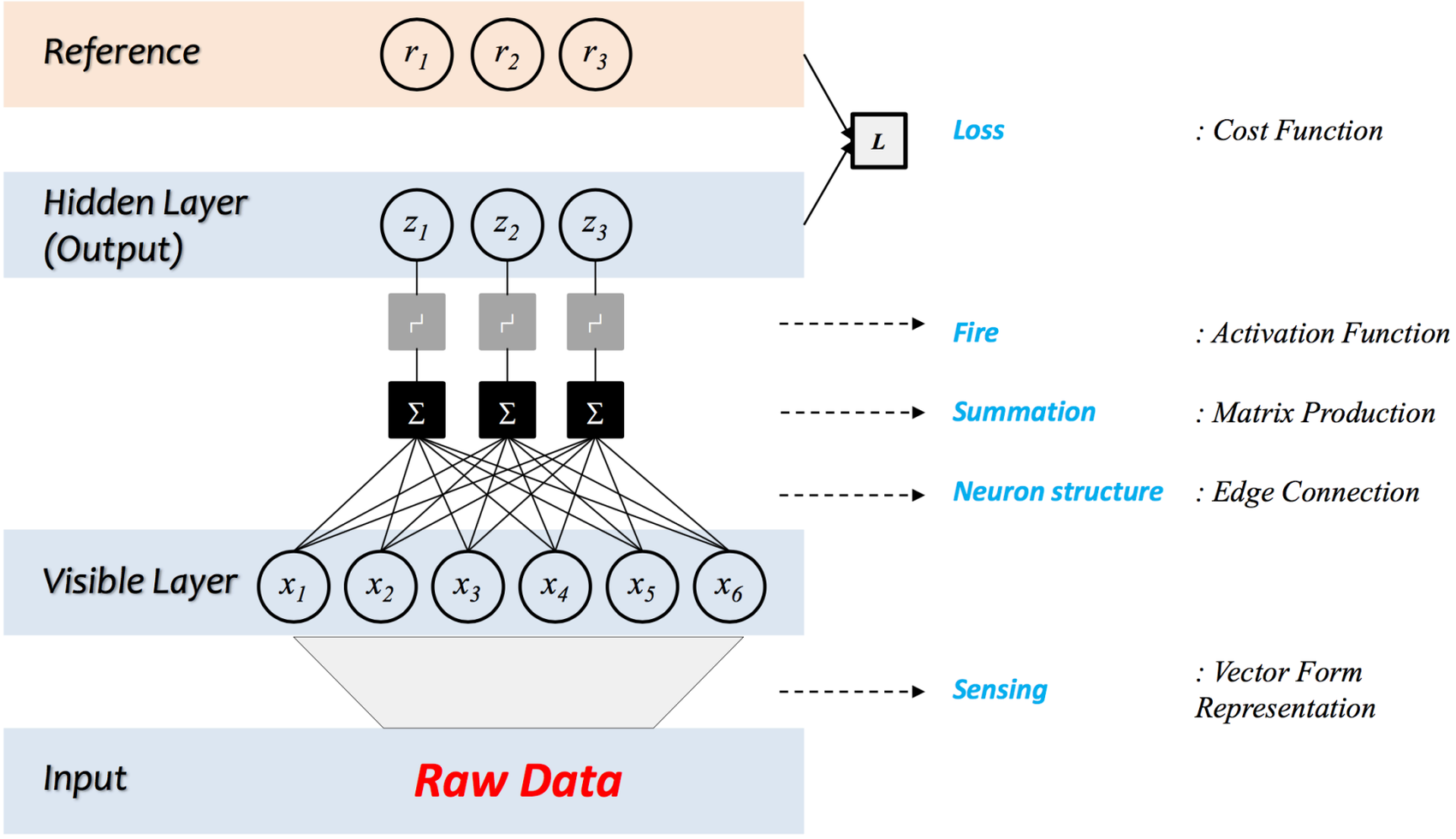
Randomized Leaky ReLU

# [참고] Back-propagation & Activation

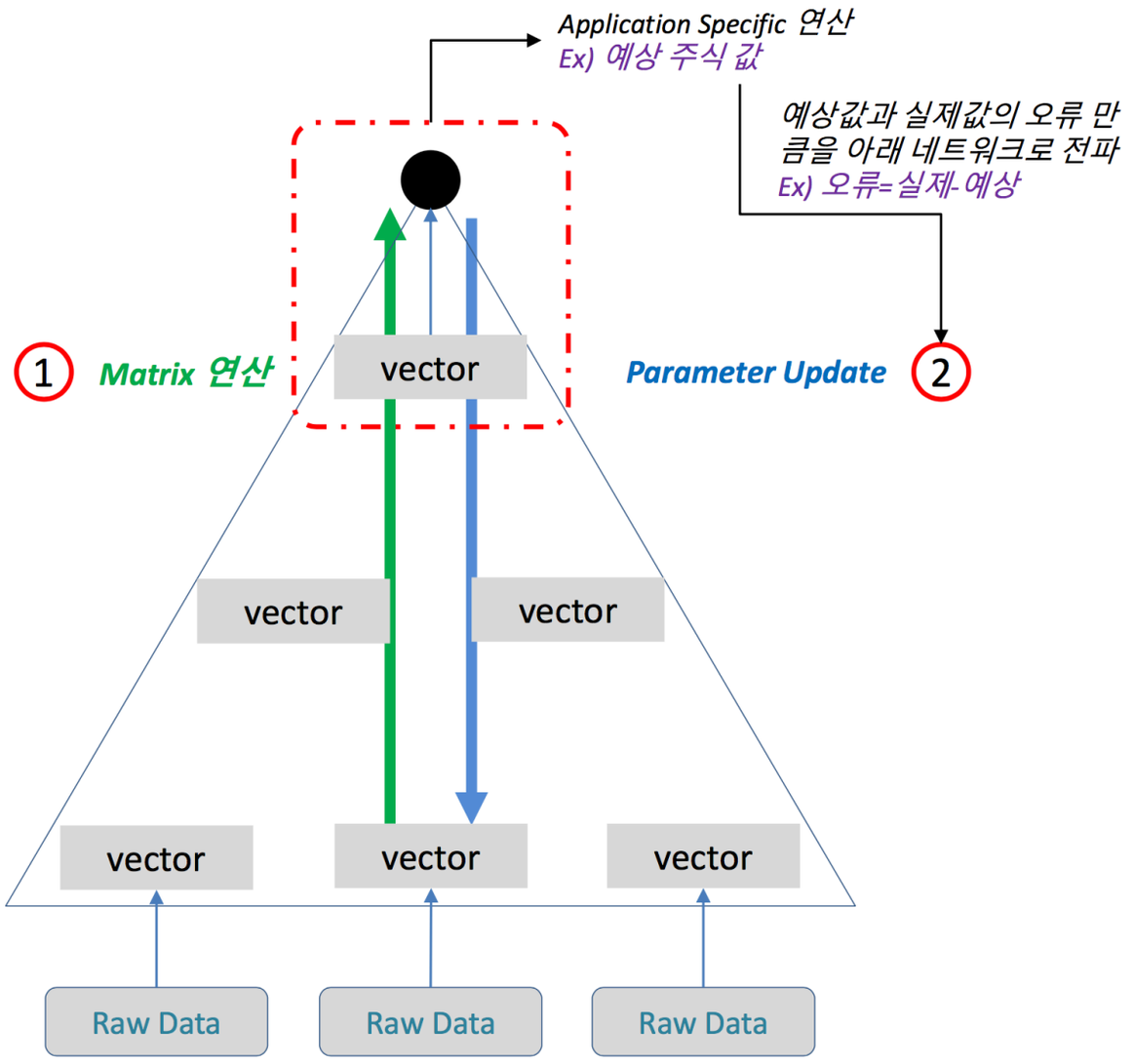
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$



# Summary : Neural Network – Core Components

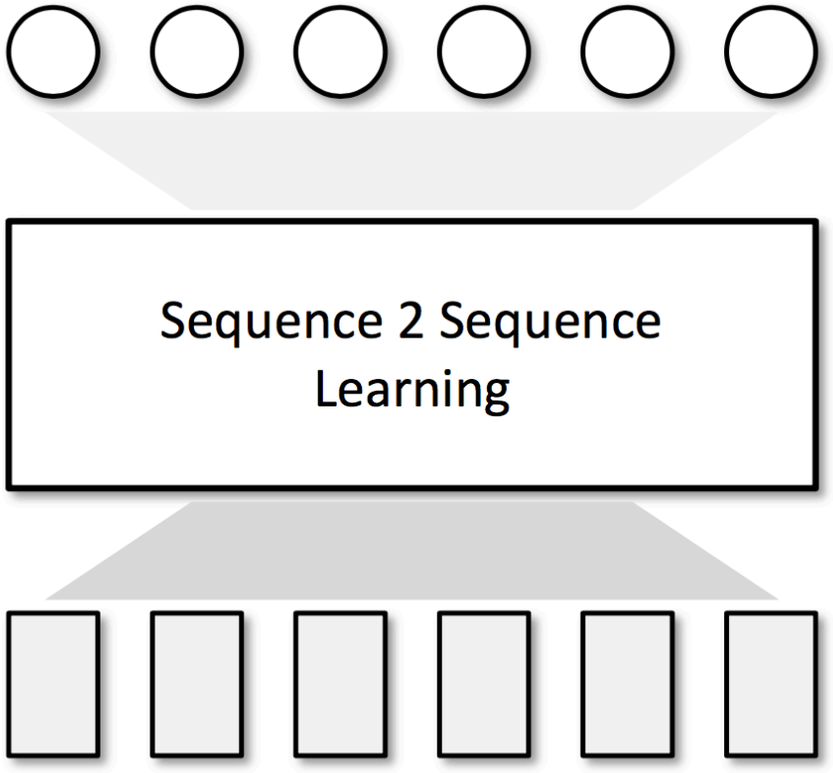


# Summary : Neural Network – Process

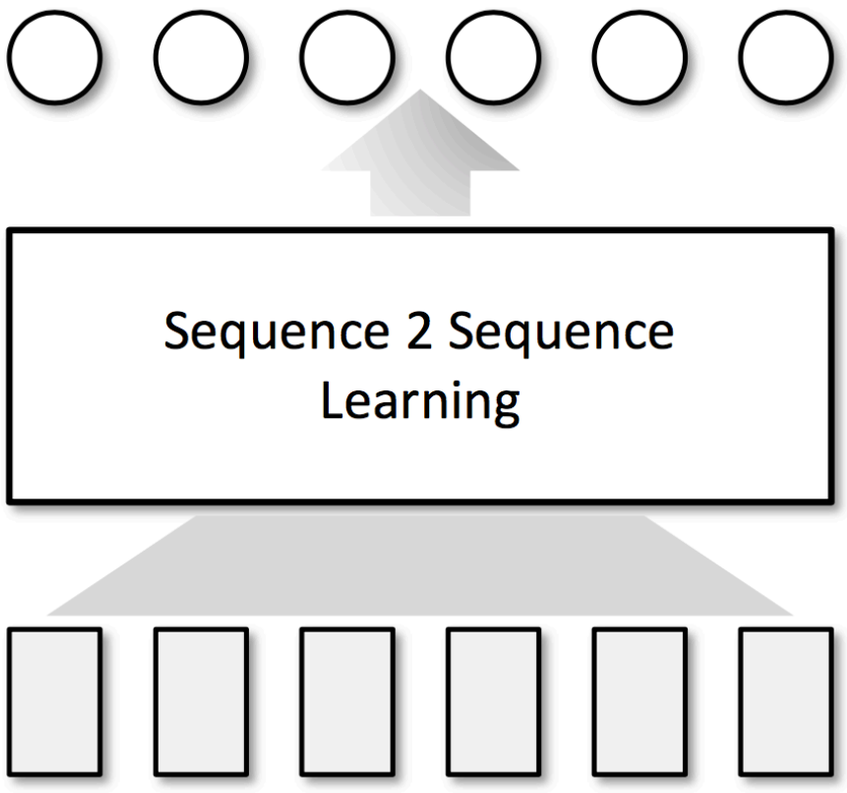


# SEQ 2 SEQ LEARNING

# Illustration



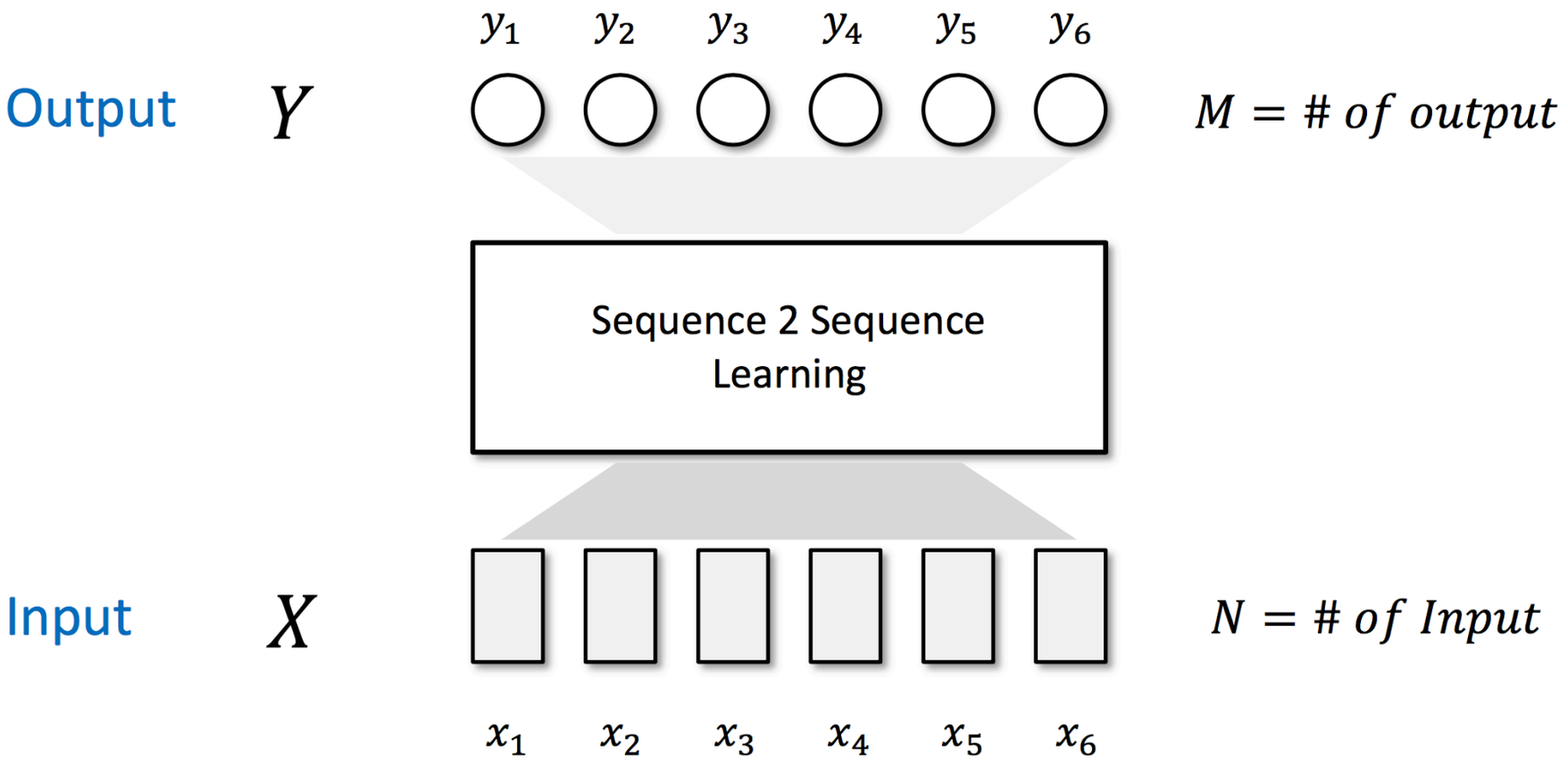
# Sequence 2 Sequence Learning @ running time



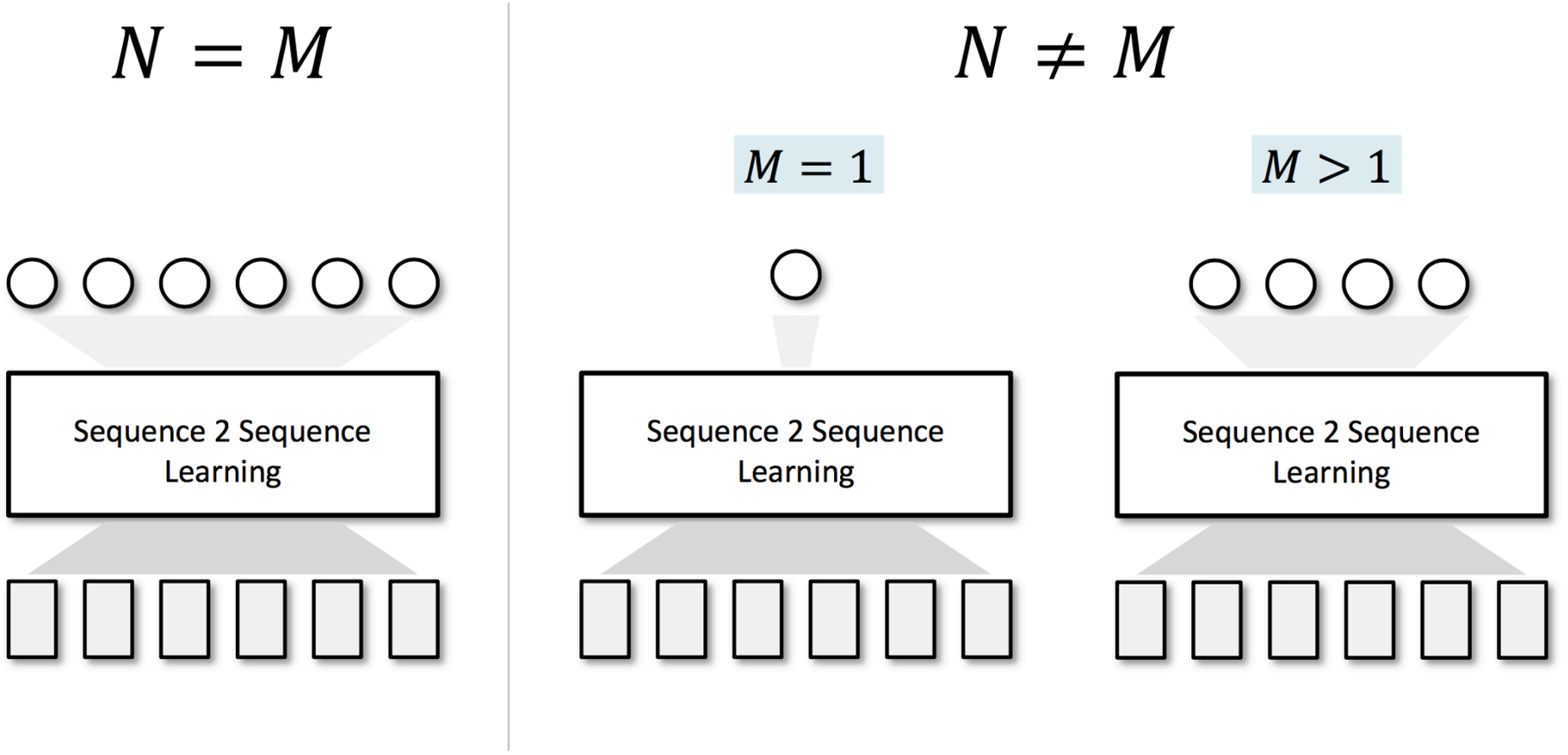

*Sequence Generation*

*Sequence Feeding*

## Definition

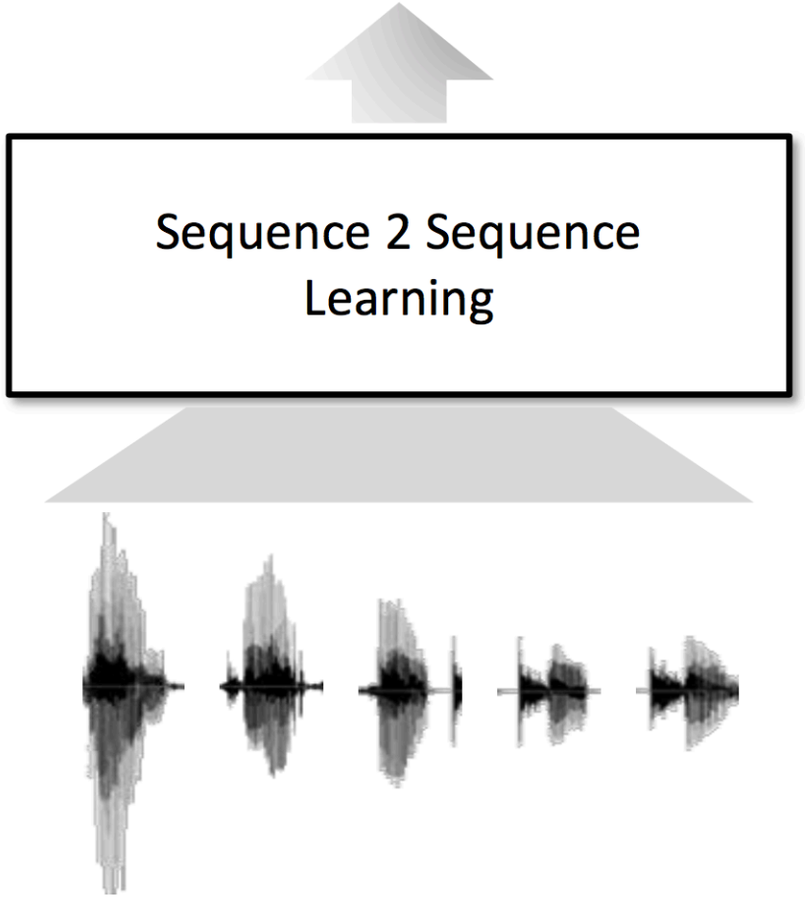


# S2S - Cases



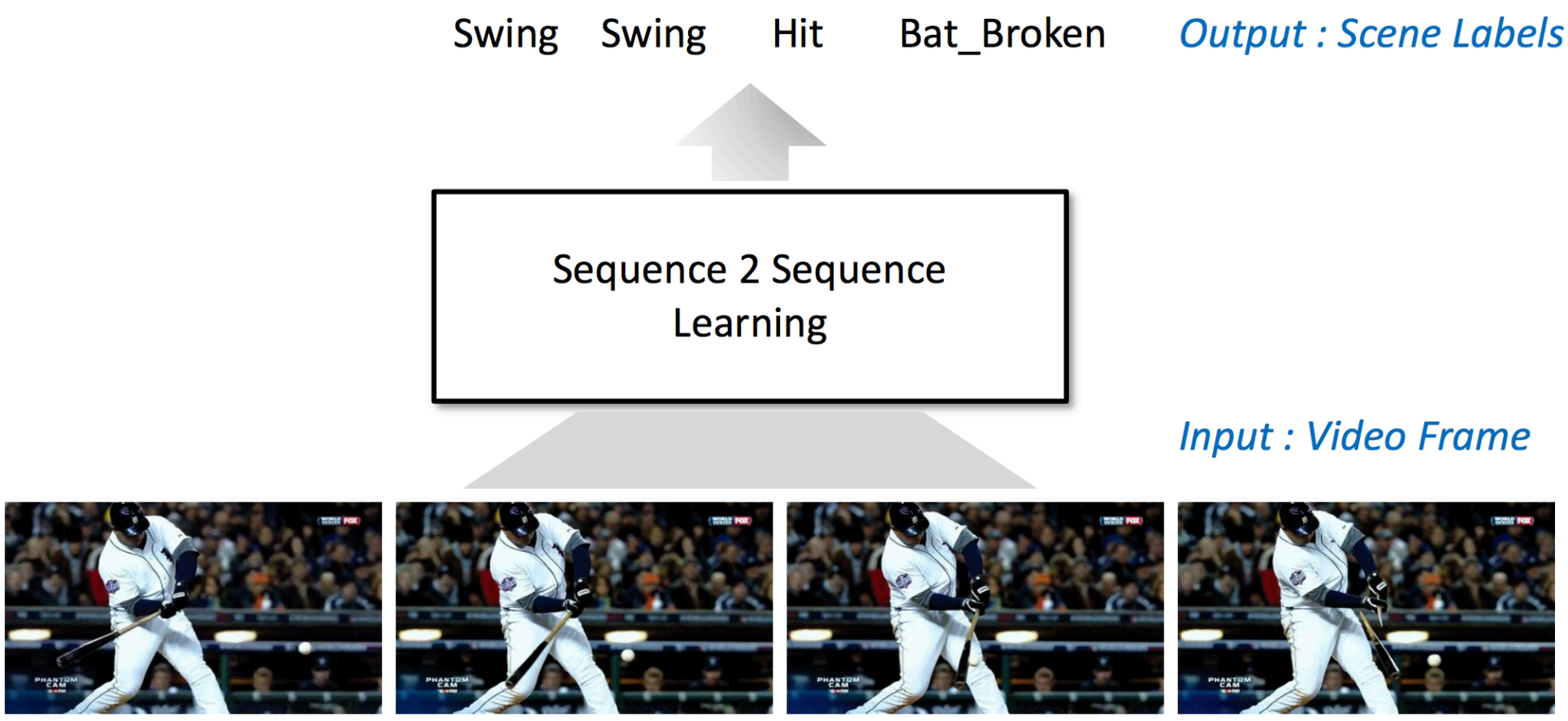
# S2S example – Speech Recognition

오늘 날씨 좀 알려 주세요 *Output : Text*

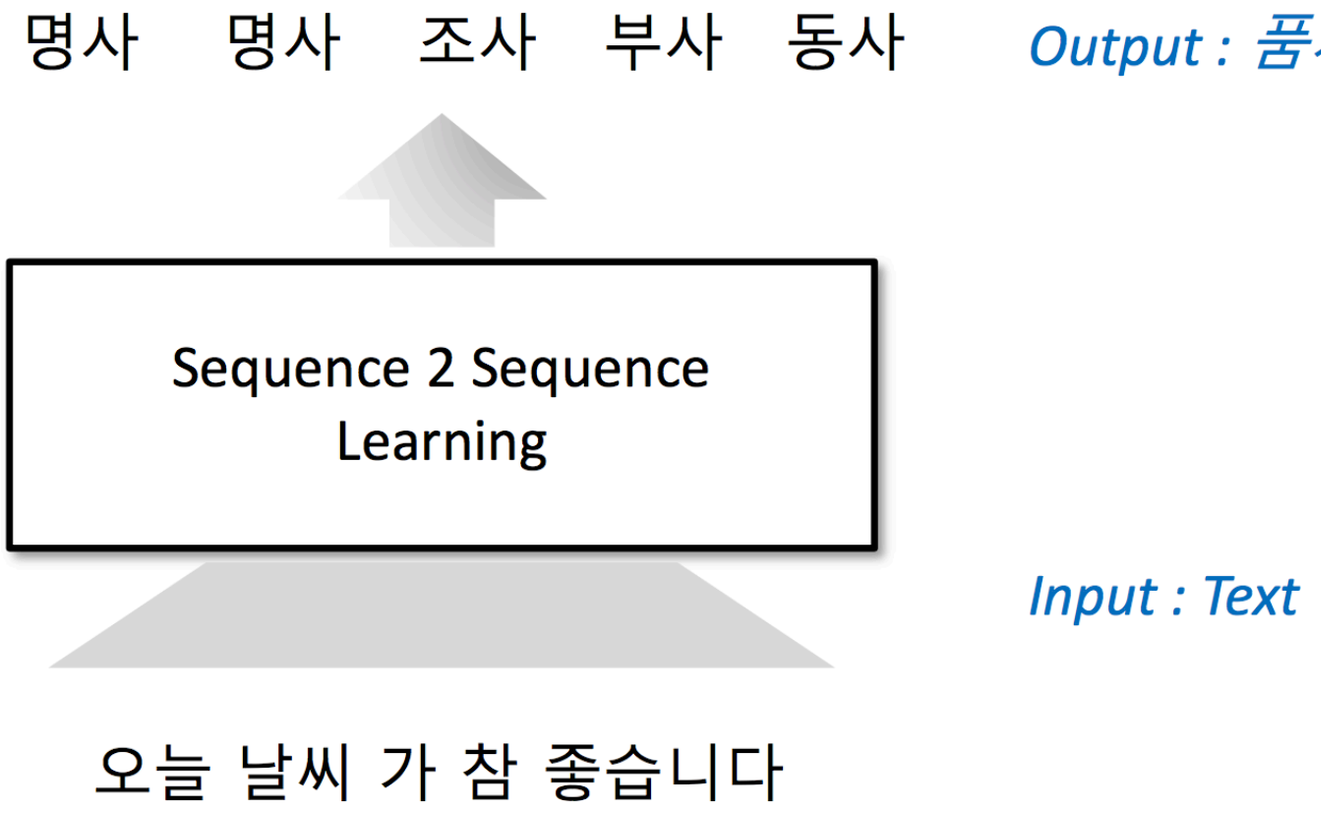



*Input : Speech Signal*

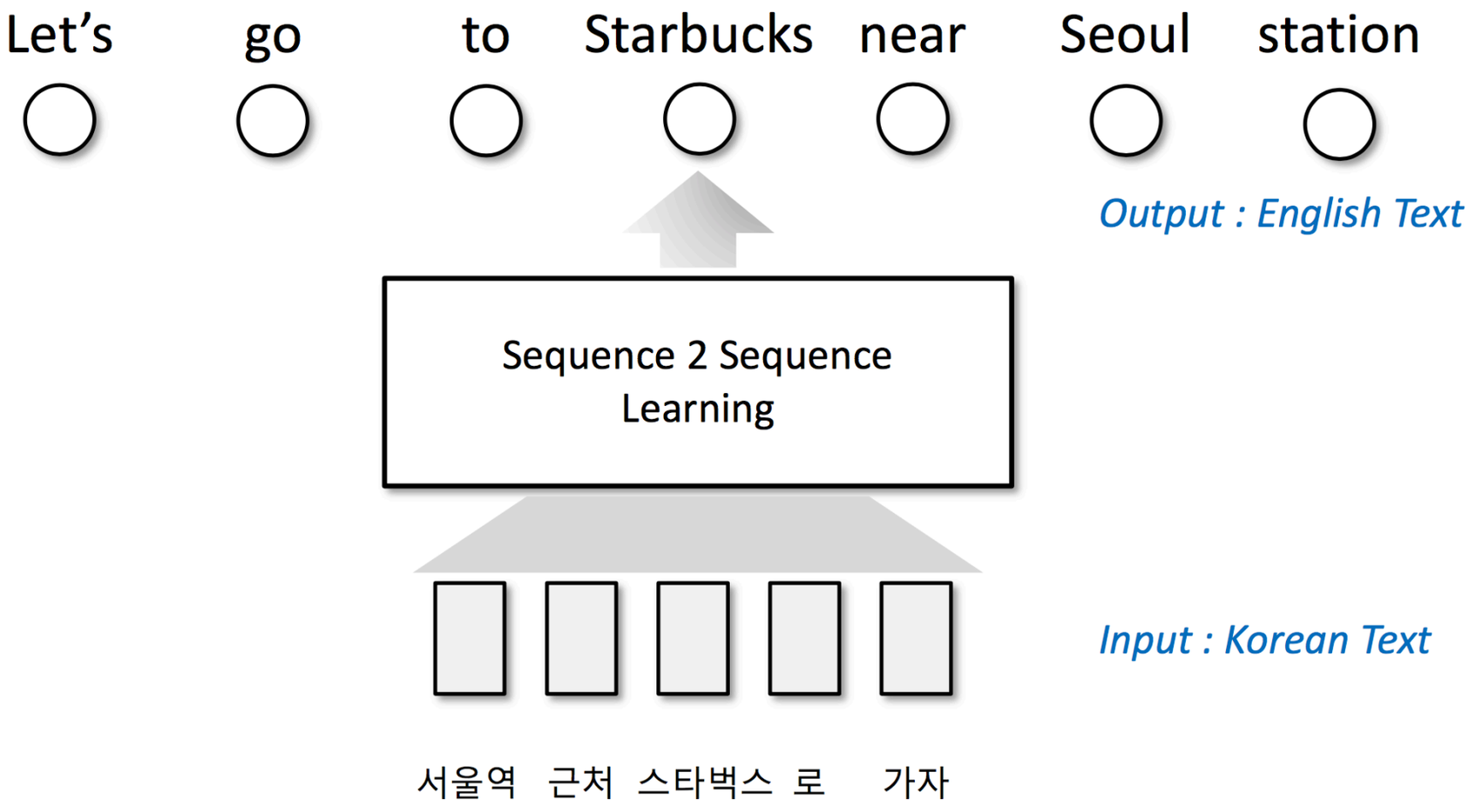
# S2S example – Movie Frame Labeling



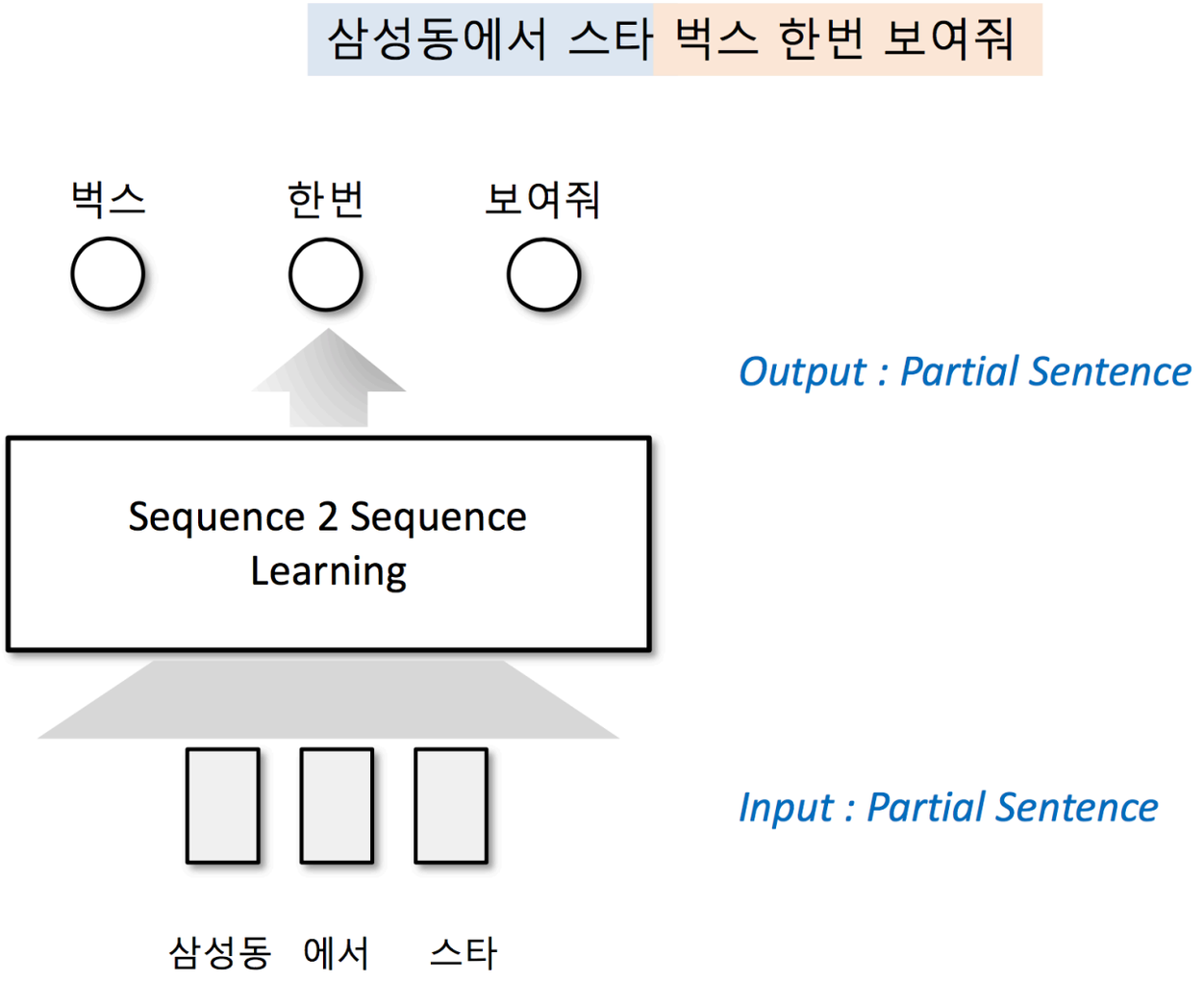
## S2S example – POS Tagging



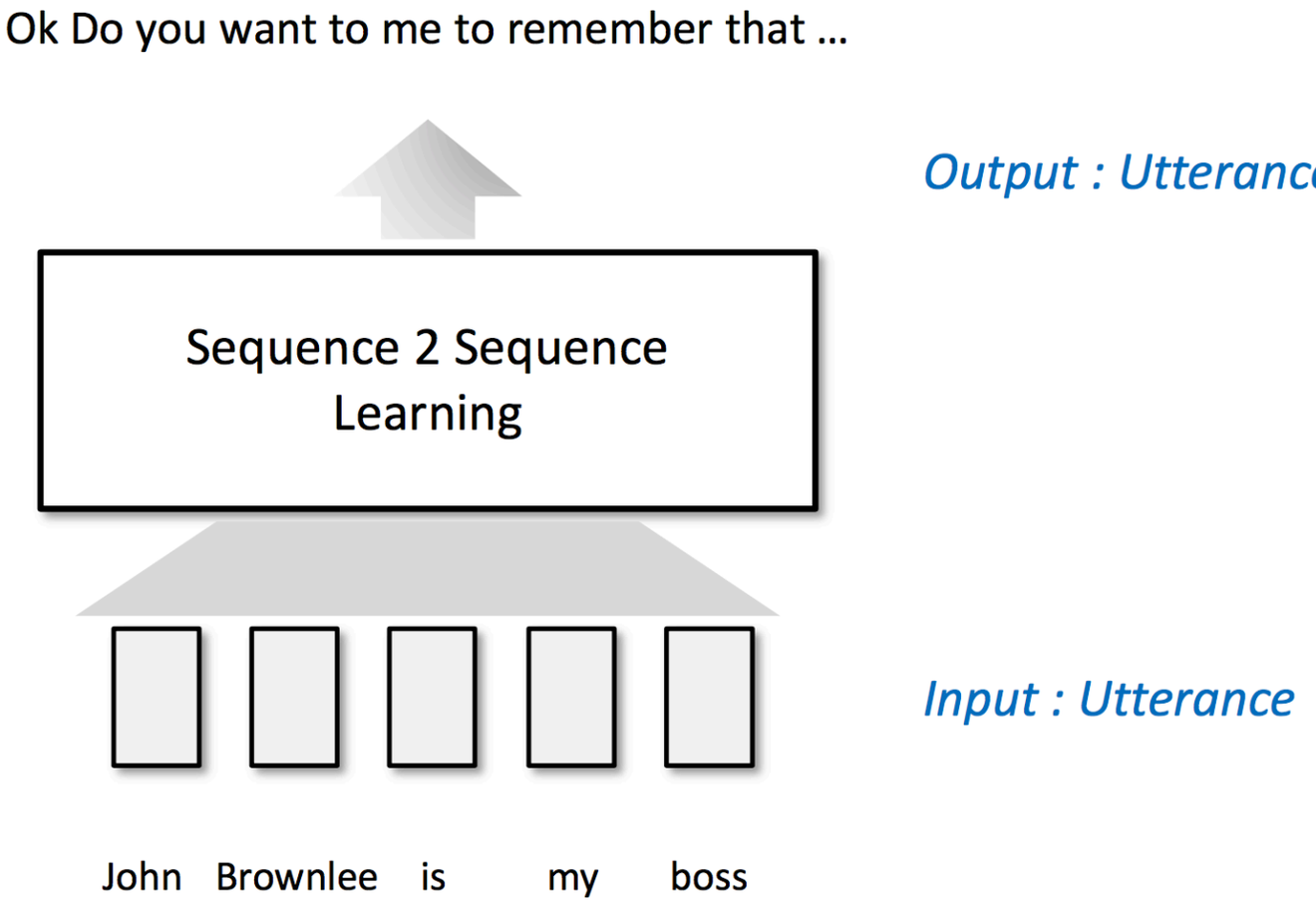
# S2S – Machine Translation



# S2S – Sentence Completion

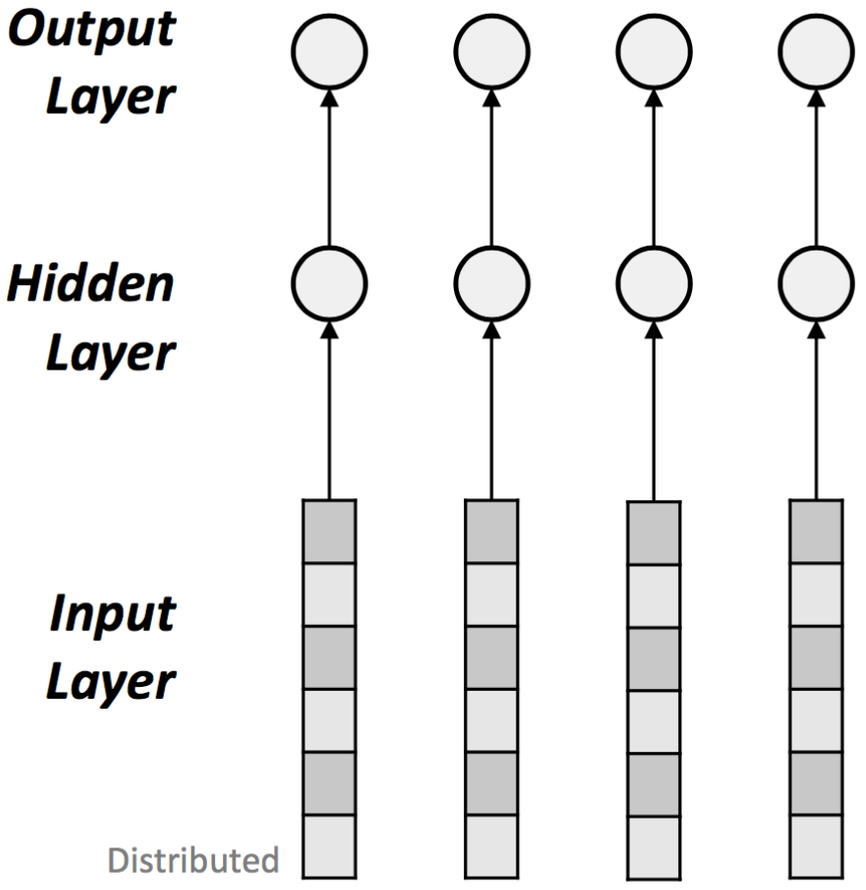


# S2S – Conversation Modeling

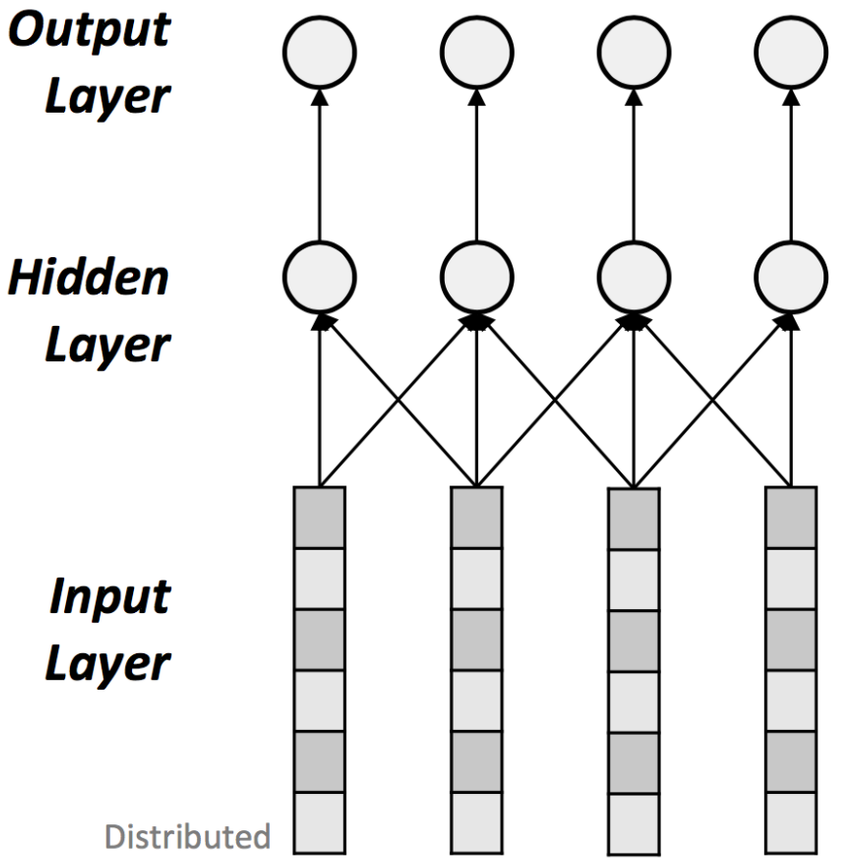


# RNN, RECURRENT NEURAL NETWORK

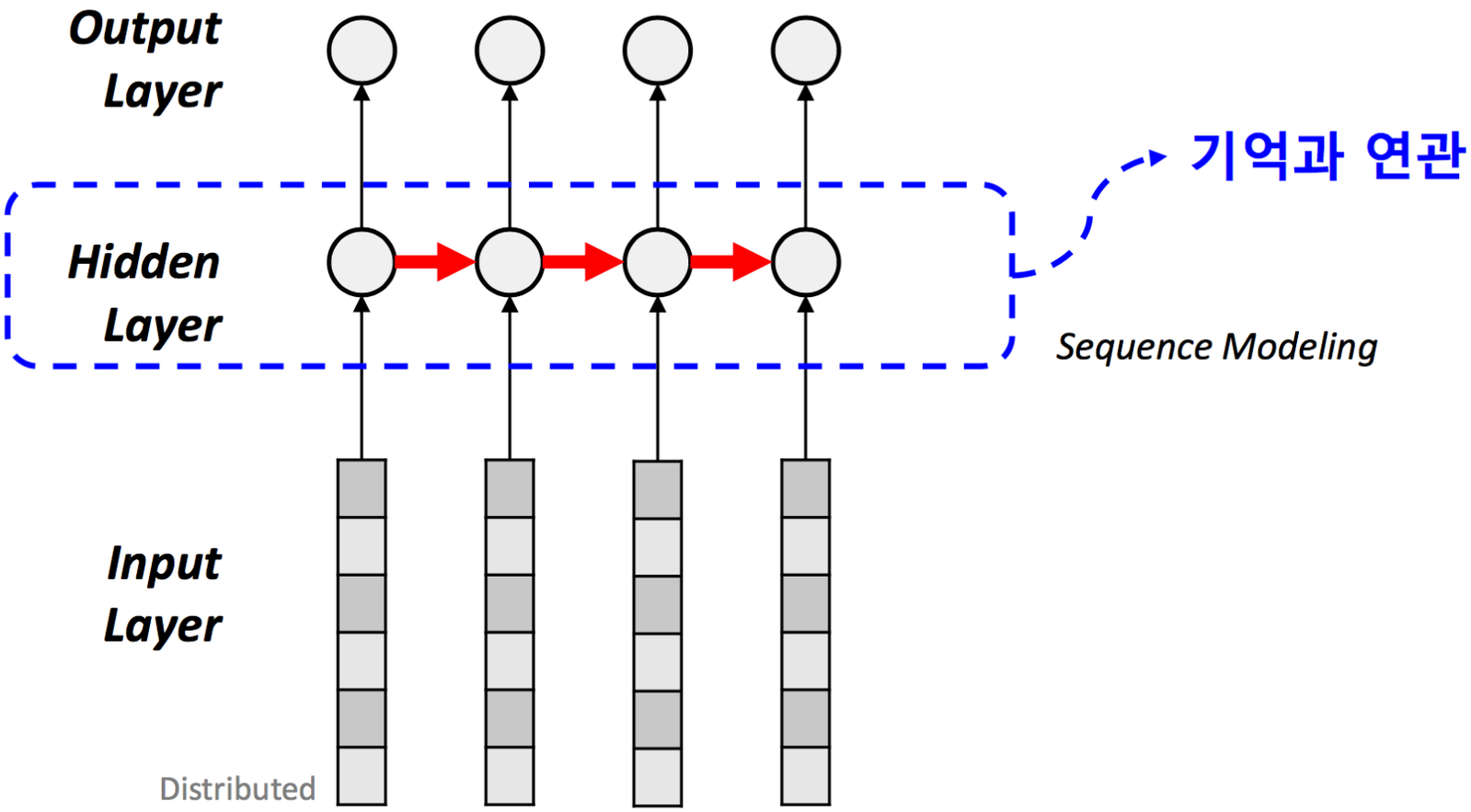
# 여러 번 (독립적으로) **Prediction**



# 여러 번 (독립적으로) **Prediction + Convolution Idea**

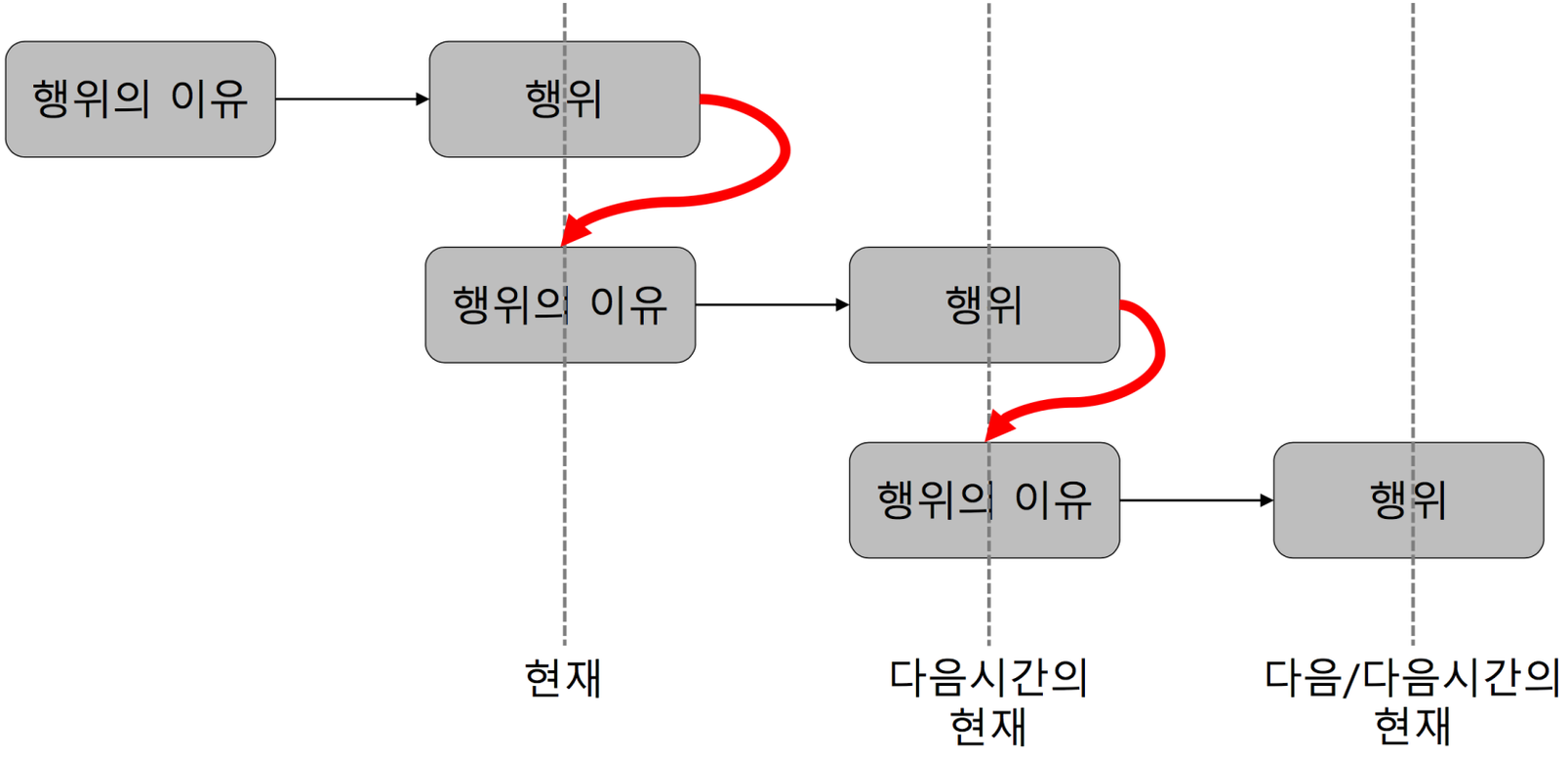


## Sequence 를 모델링

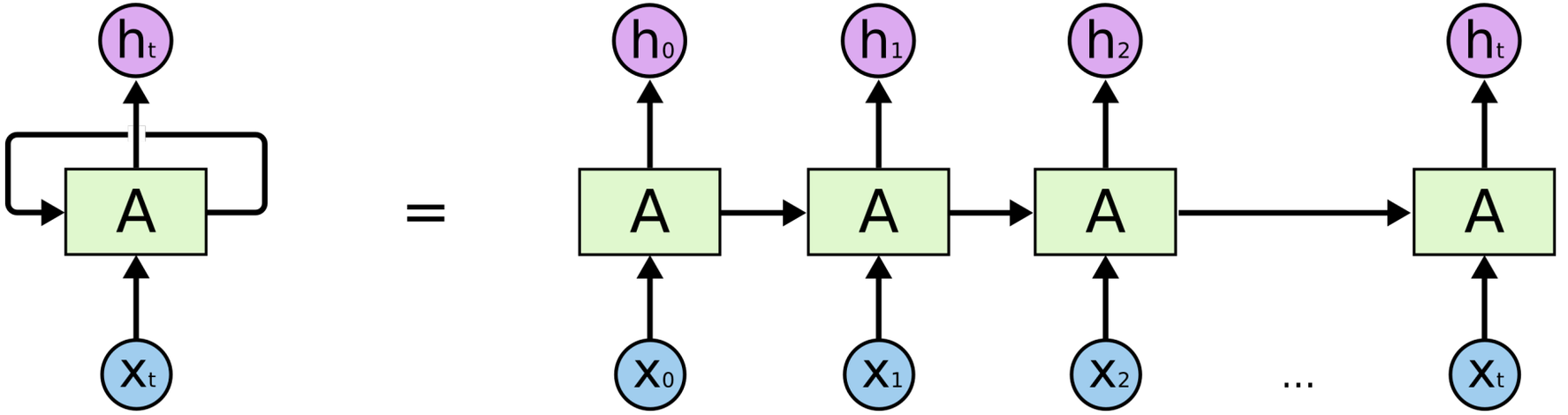


## Neural Network + Memory

**기억**: 과거의 어떤 것이 현재에 영향을 미치는 것

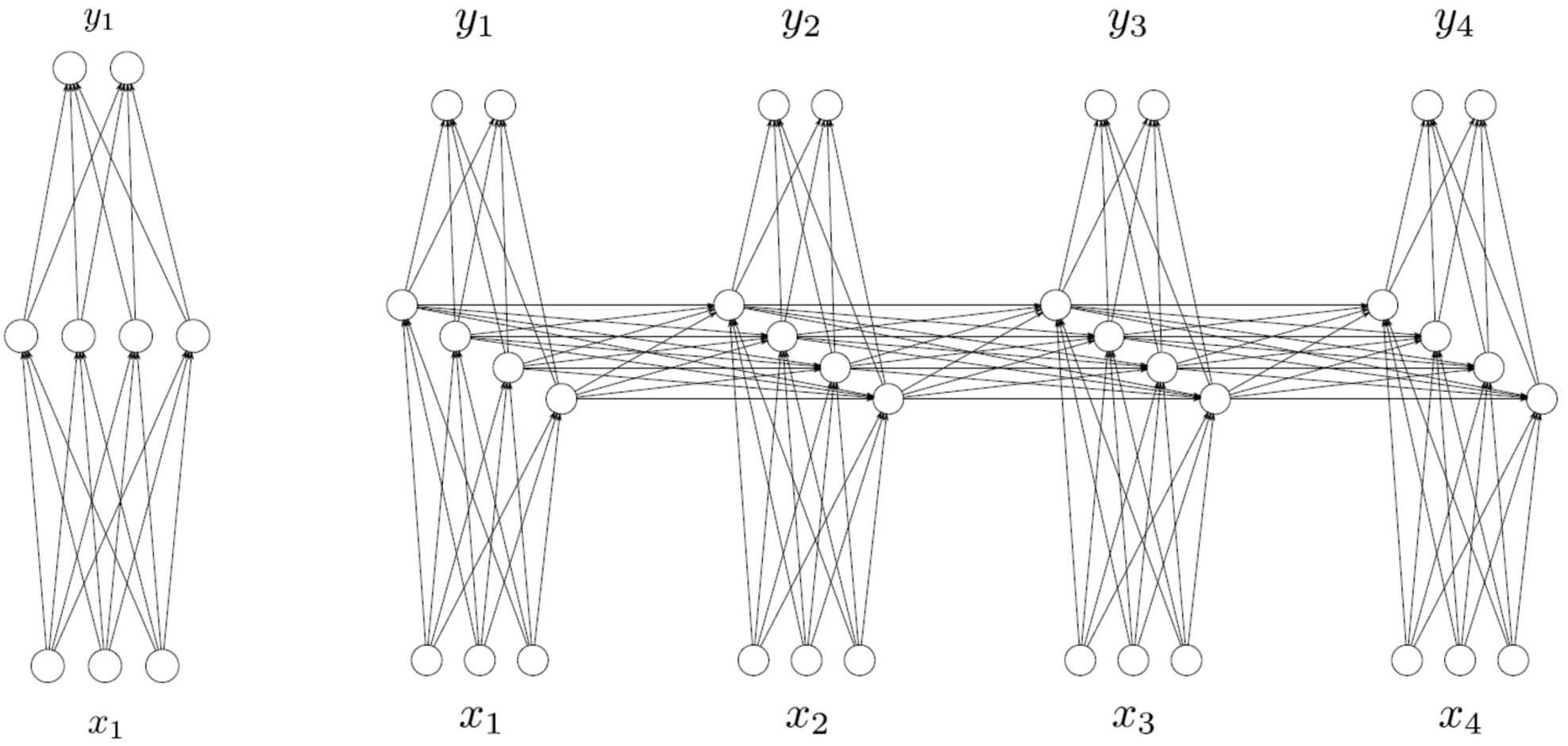


# Neural Network + Memory = Recurrent Neural Network

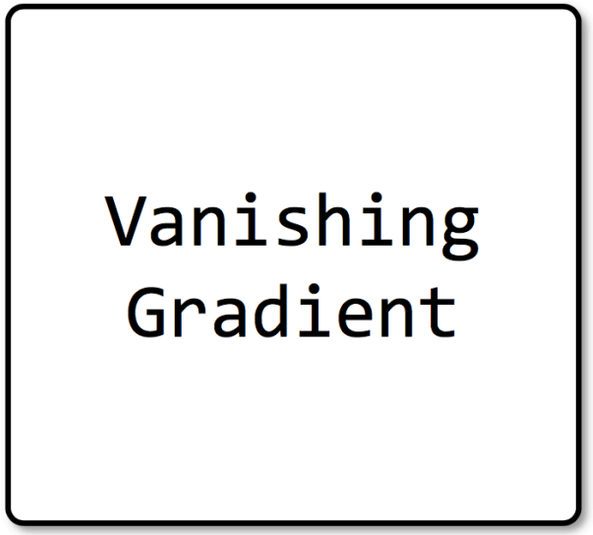



Recurrent Neural Network 모델

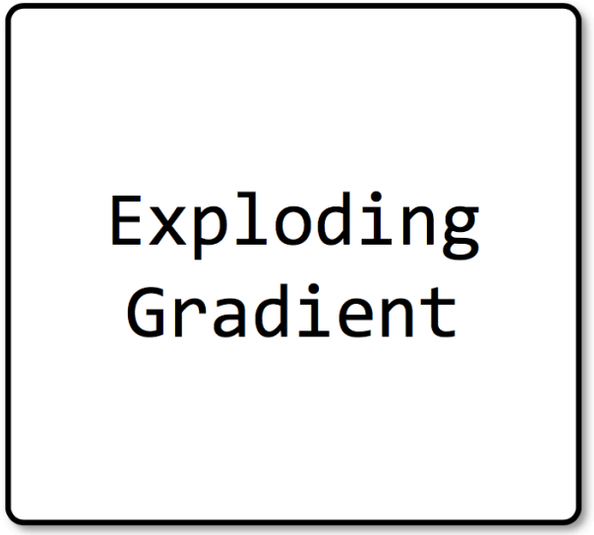
# Feed forward network vs. Recurrent Neural Network



## Recurrent Neural Network 문제점

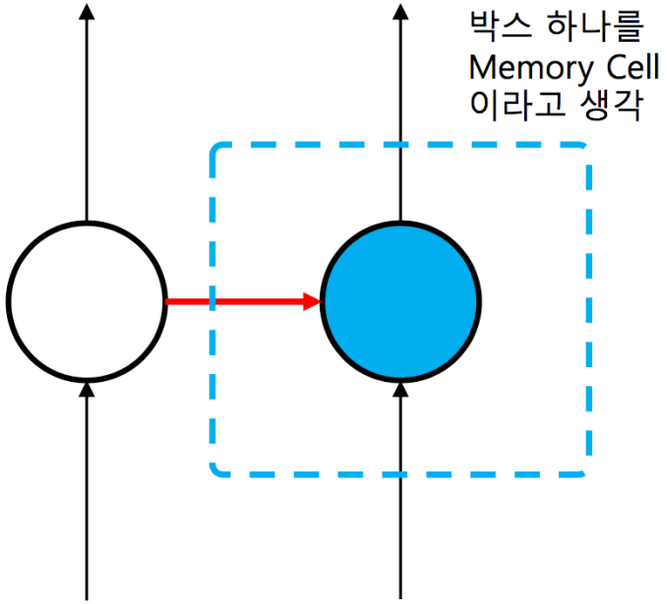


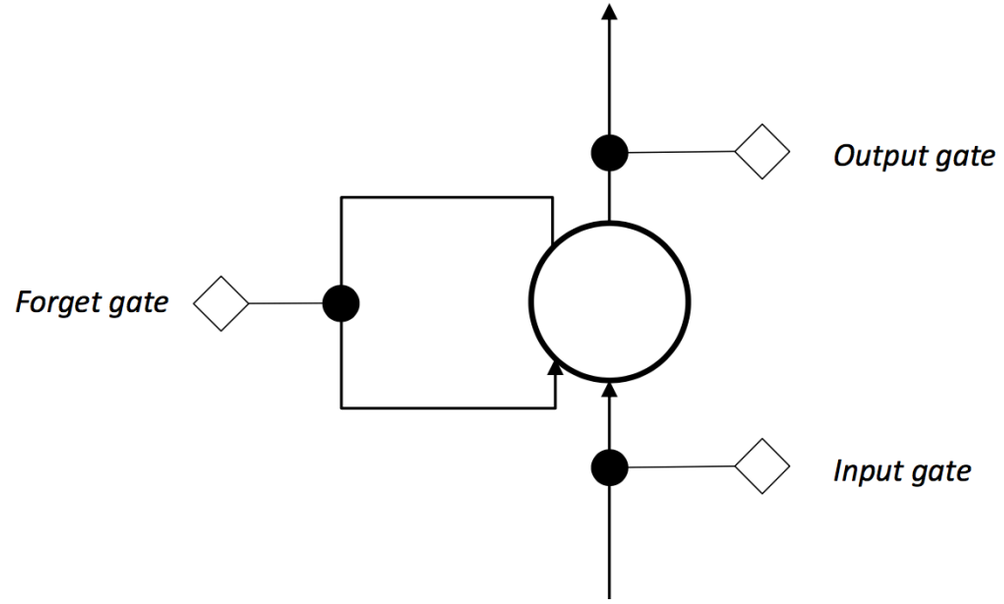

구조적으로 바로 전의 것에 영향을 받게 되어 있기 때문 에 Long-Dependancy 가 있는 입력 혹은 결과가 잘 반영되 지않는다.




Short-Dependancy 가 더 중요 함에도, 멀리 있는 gradient가 너무 높게 계산되, 가까이 있 는 gradient 가 충분히 반영되 지않는경우

# LSTM - Idea

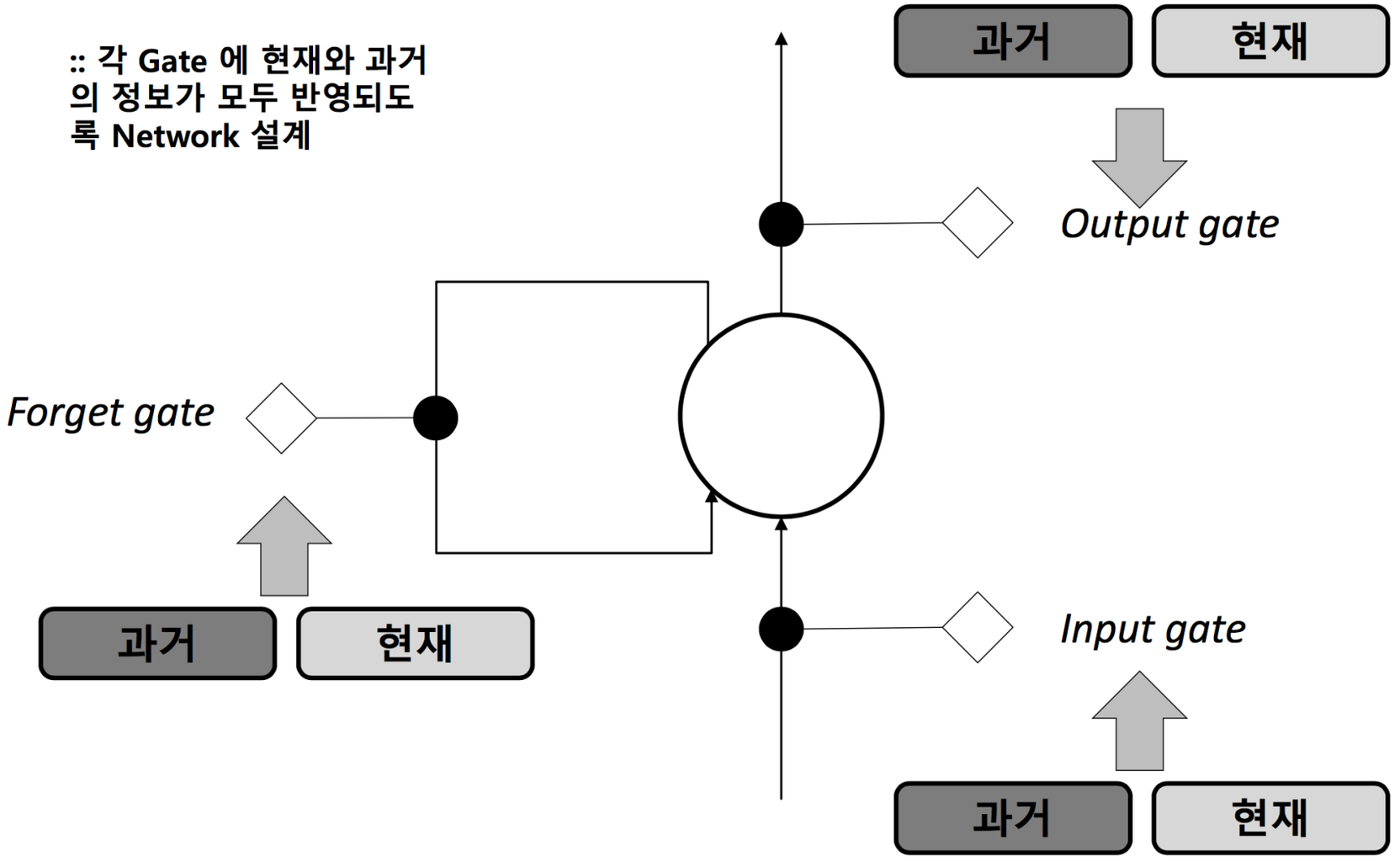



| neural | memory | 의미 |
|---|---|---|
| input | Write | 1이면 입력 x가 들어 올수 있도록 허용(open). 0이면 Block(closed) |
| output | Read | 1이면 의미있는 결과물로 최종 Output(open). 0이면 해당 연산 출력 안함(closed) |
| forget | Reset | 1이면 바로 전 time 의 memory 를 유지. 0이면 reset. Keep gate |

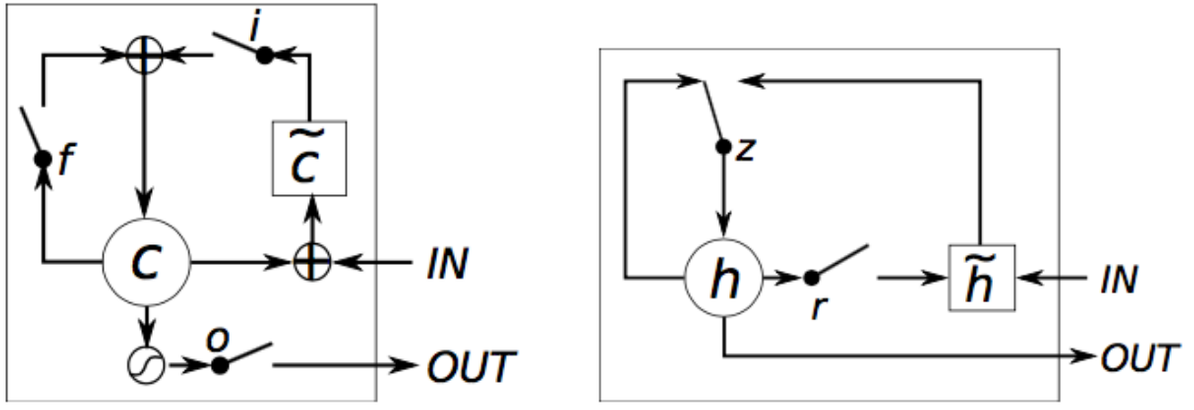
# LSTM - Implementation

:: 각 Gate 에 현재와 과거의 정보가 모두 반영되도록 Network 설계



## Gated Recurrent Unit

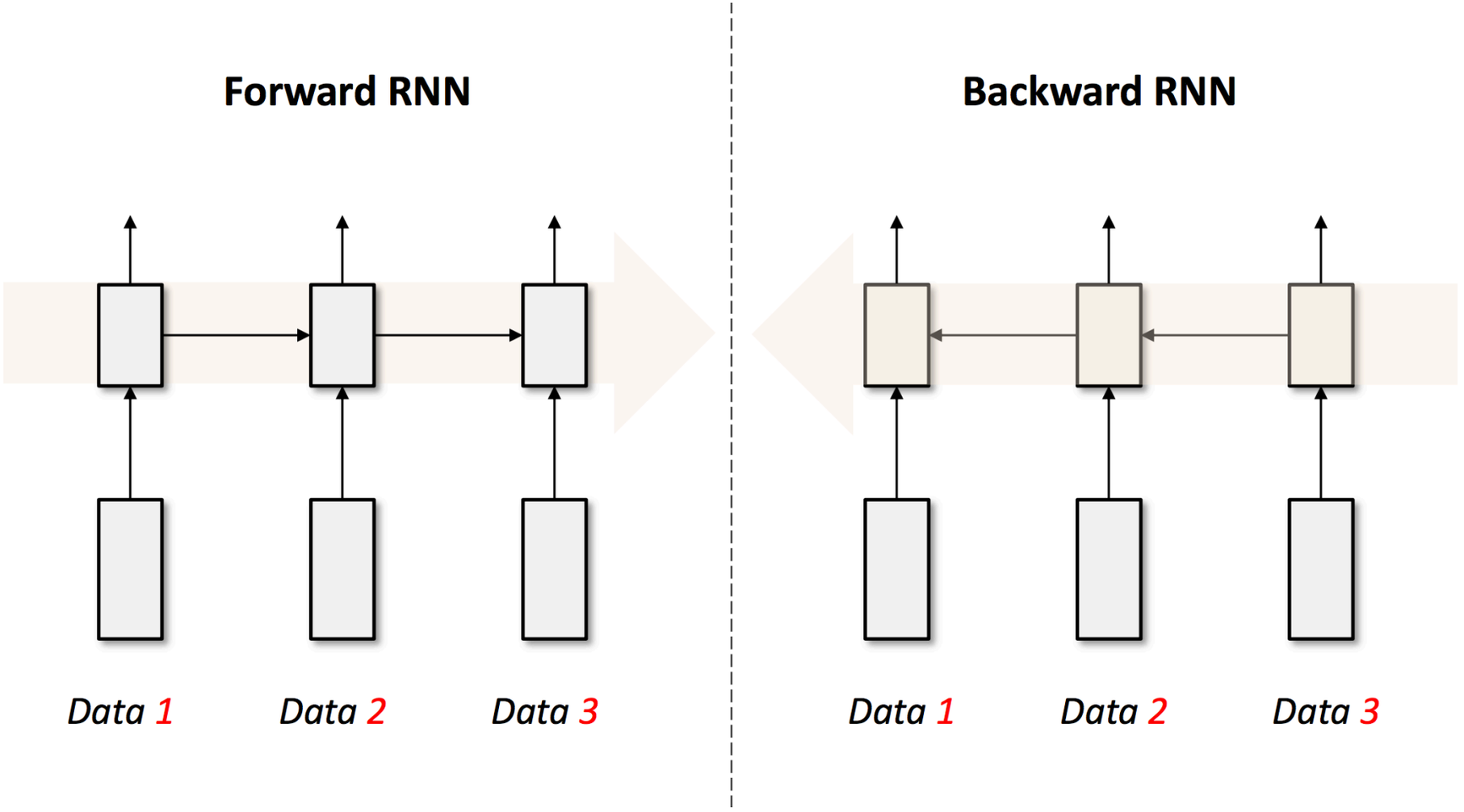
- Update Gate 를 두어서 현재 Time 의 Hidden state 를 계산할 때 update gate 의 영향을 받도록 함
- Reset Gate 를 두어서 현재 Memory 를 Reset 할지 안할지를 결정( like LSTM)
- Cho et al. 2014



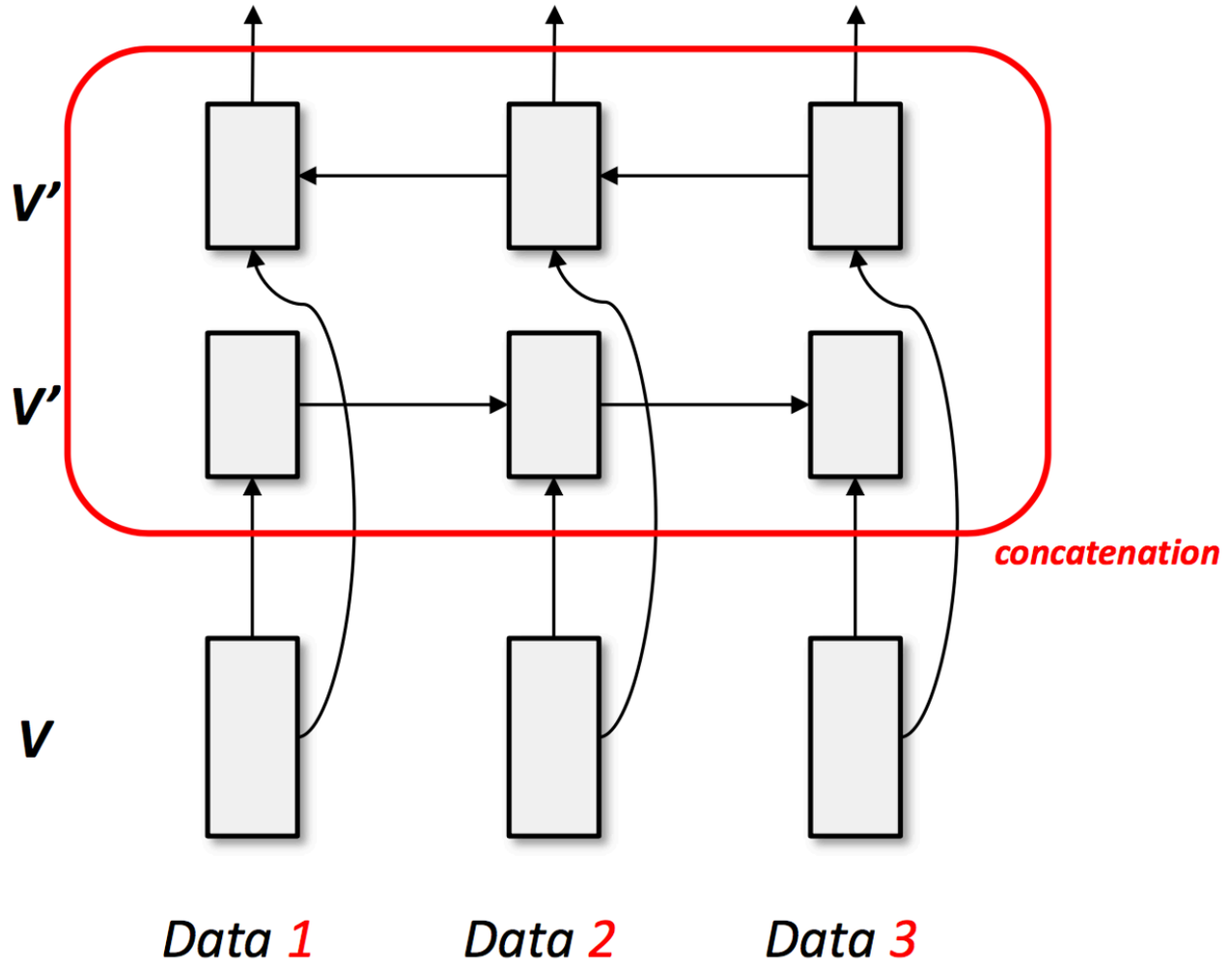

(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a) $i$, $f$ and $o$ are the input, forget and output gates, respectively. $c$ and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) $r$ and $z$ are the reset and update gates, and $h$ and $\tilde{h}$ are the activation and the candidate activation.

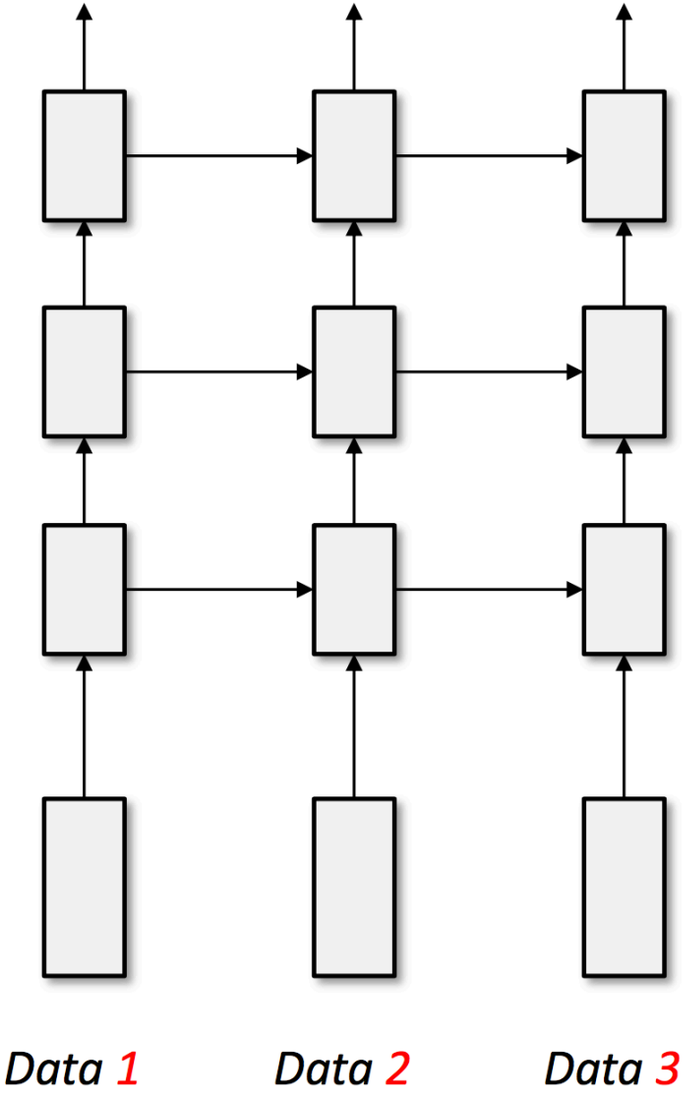
# Forward / Backward RNN



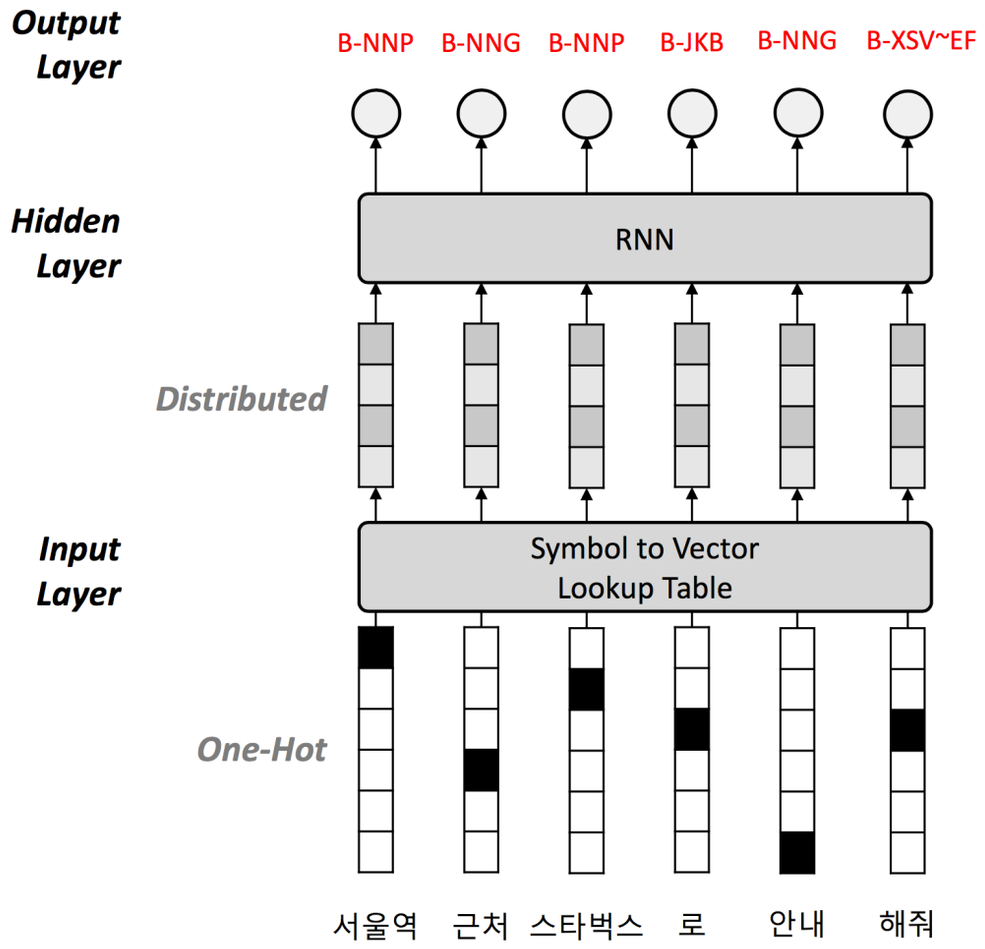
# Bidirectional RNN



# Stacking RNN

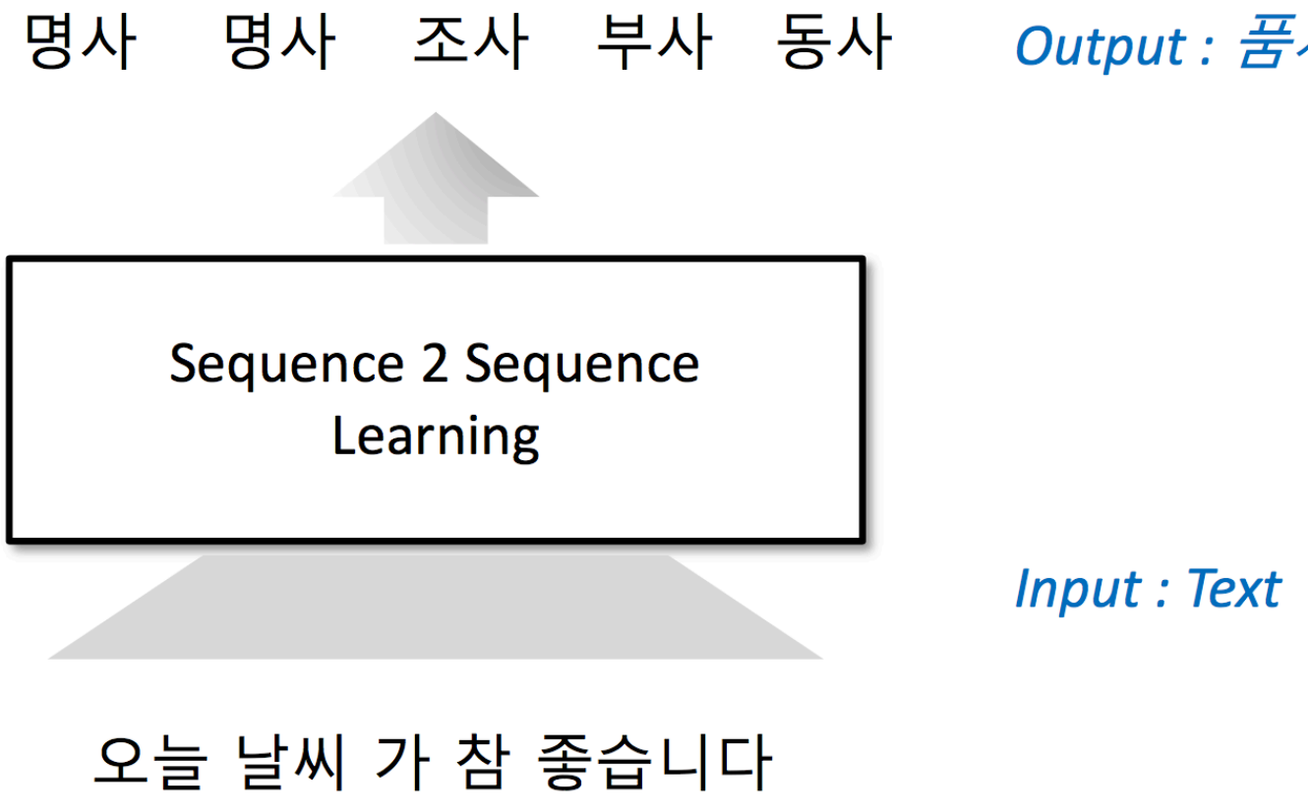


# Sequence Modeling for POS Tagging



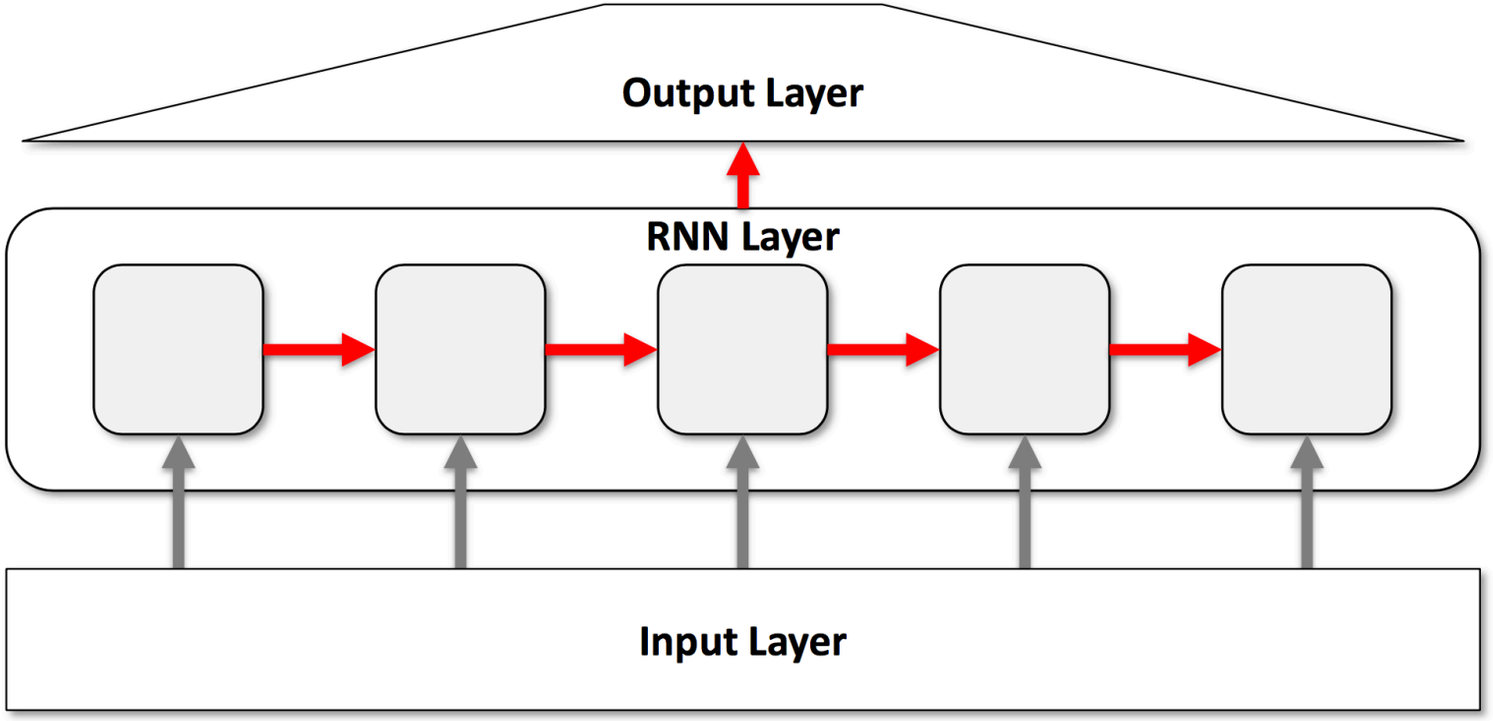

*N to N*

## Seq2Seq example – POS Tagging



# ENCODING APPROACH

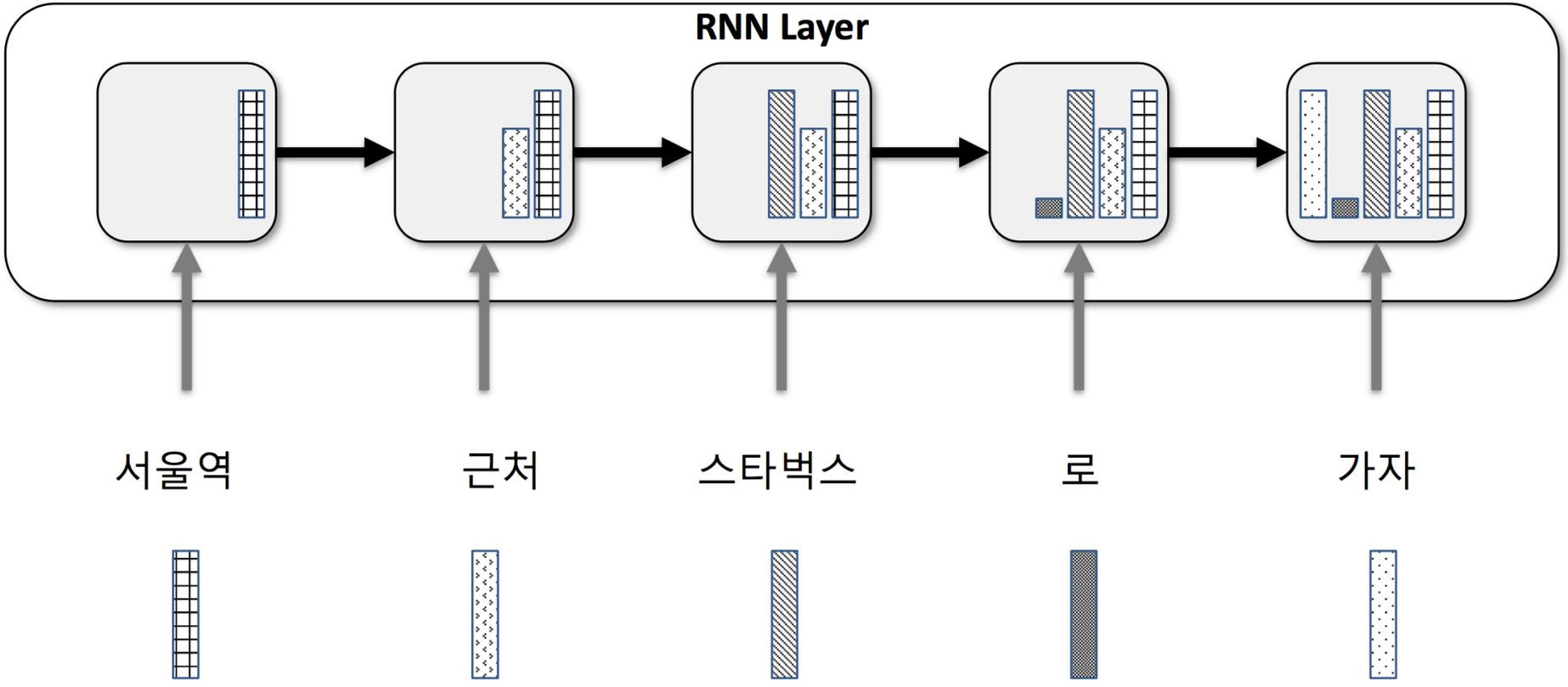
# Recurrent Neural Network - Review




Output 이 잘나오도록 Parameter 조정됨
즉, 최종 Output 이 잘나오도록 하는 정보를 RNN Layer 에 기억하게 됨

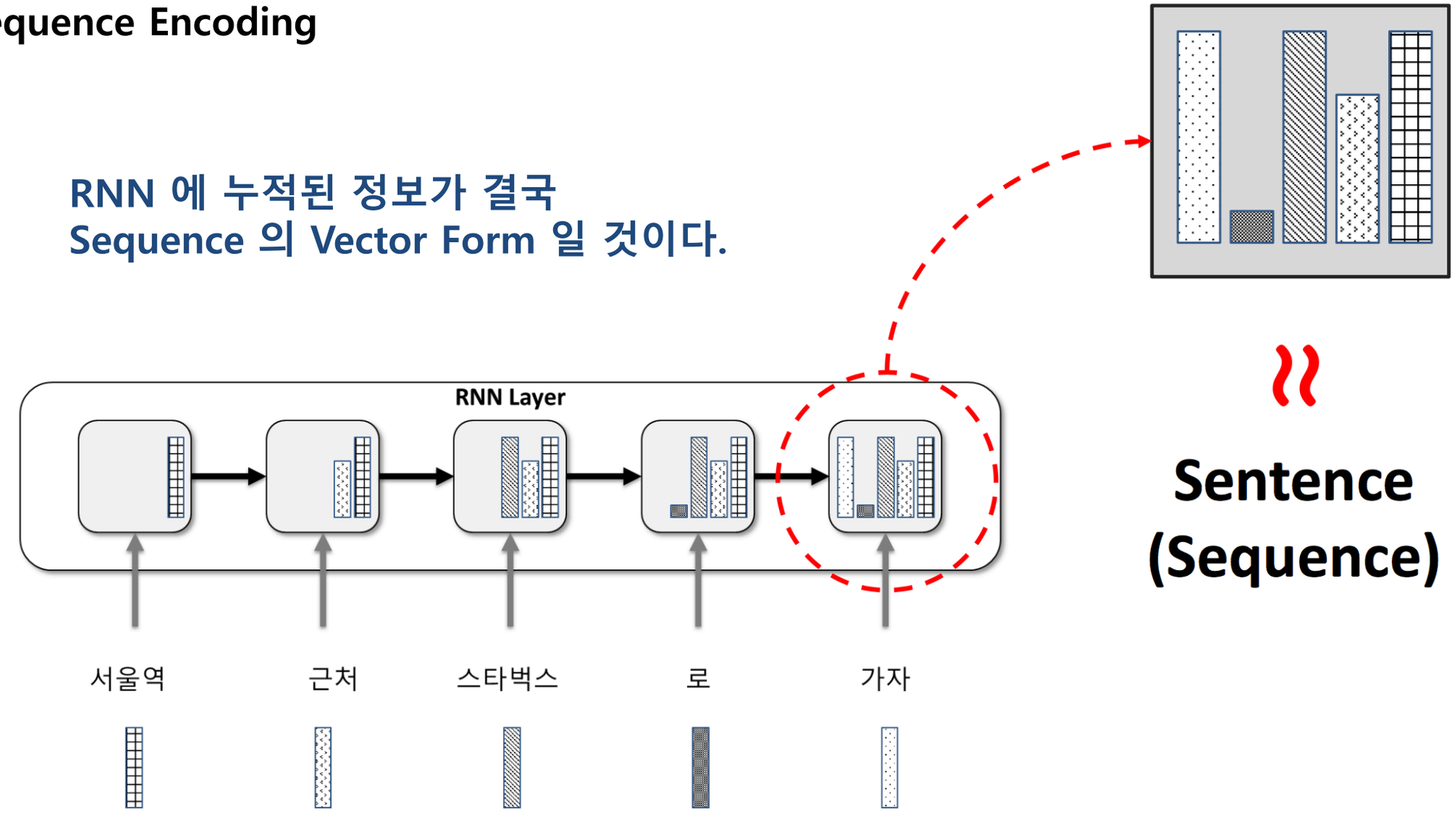
## Recurrent Neural Network - Review

Output 이 잘나오도록 하는 정보를 RNN Layer 에 기억하게 됨



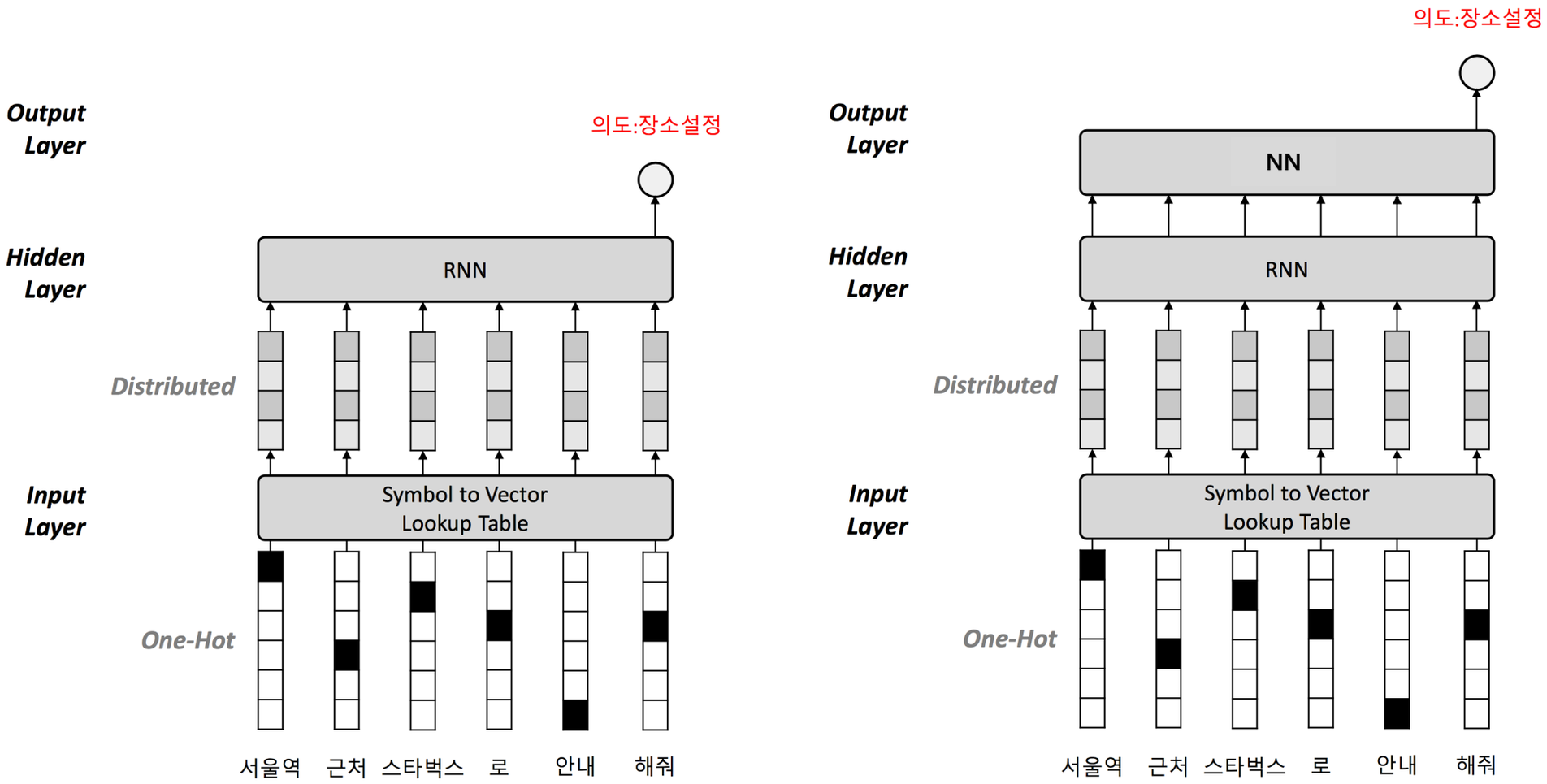
## Sequence Encoding

**RNN 에 누적된 정보가 결국**
**Sequence 의 Vector Form 일 것이다.**

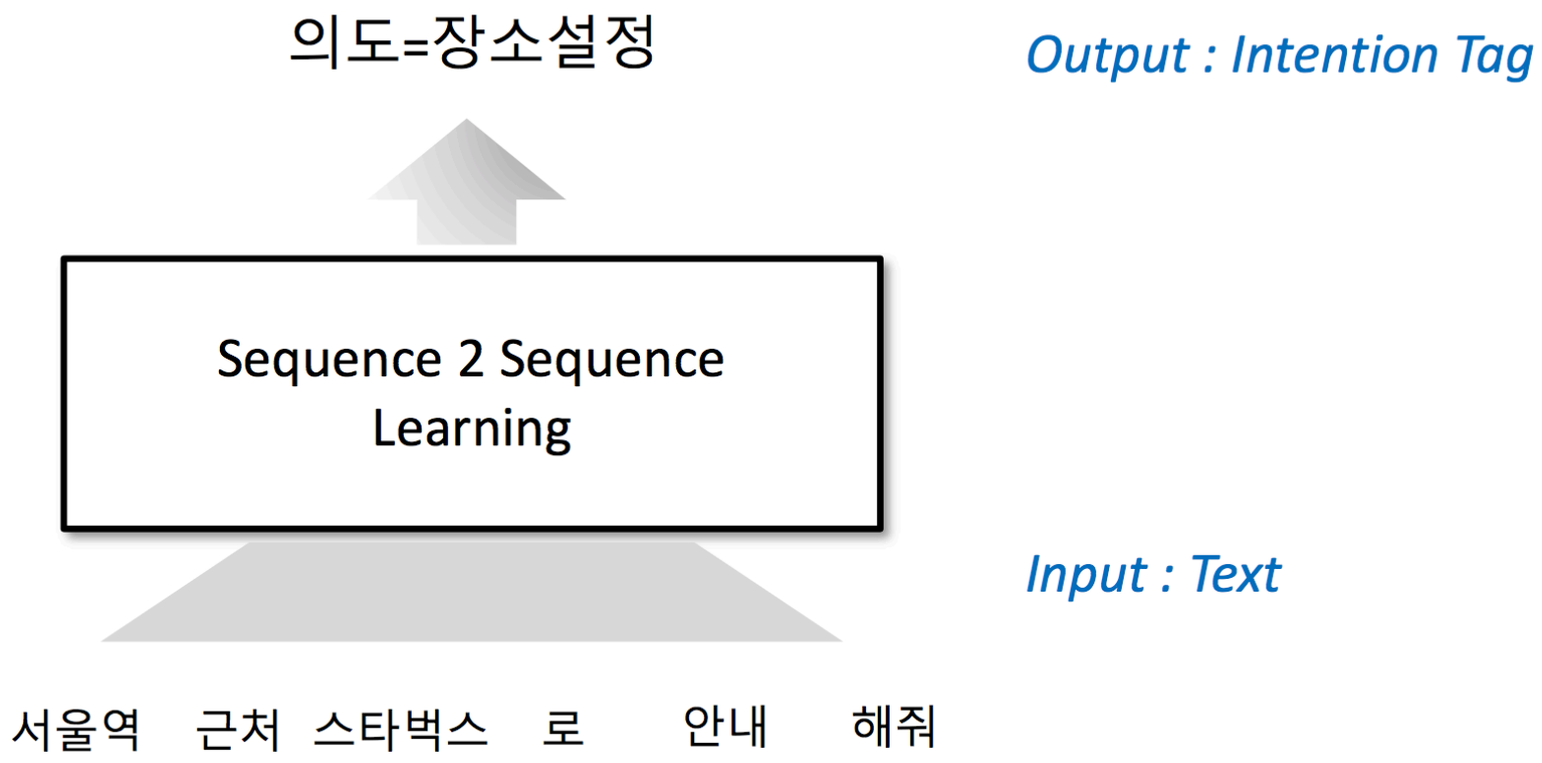


# Sequence Modeling for Intention Analysis

*N to 1*



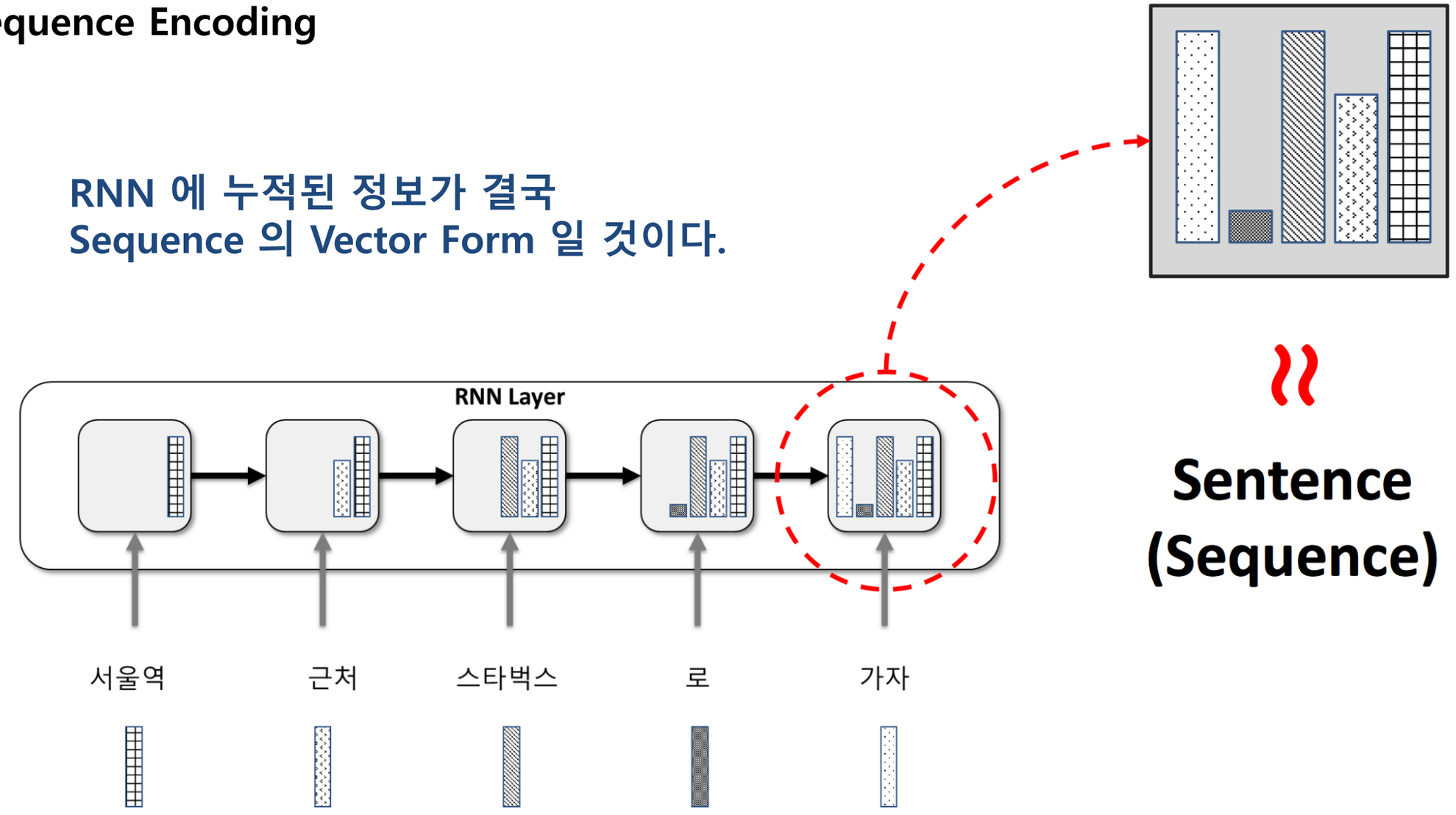
# S2S example – Intention Analysis



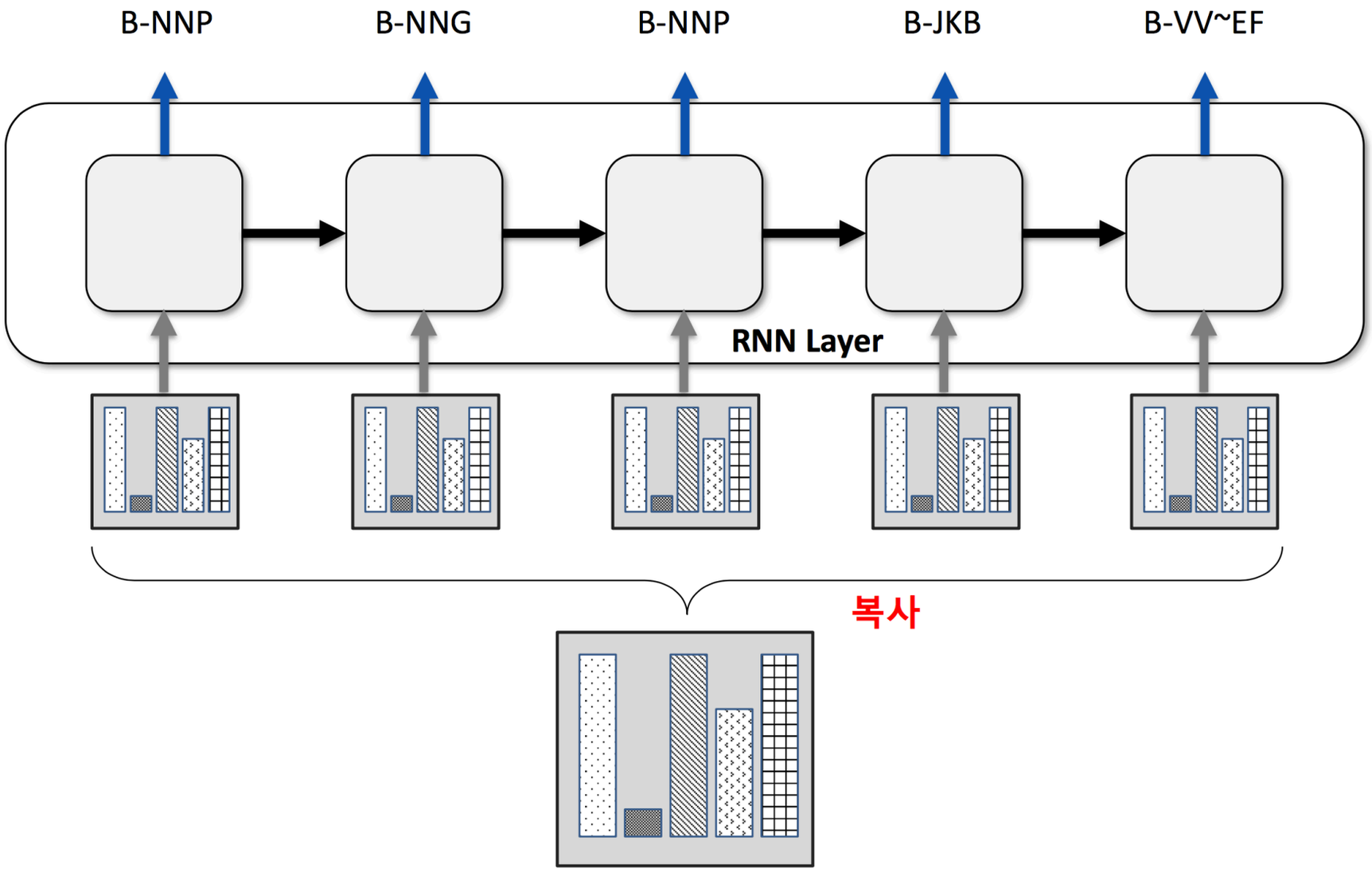
# ENCODING-DECODING APPROACH

## Sequence Encoding

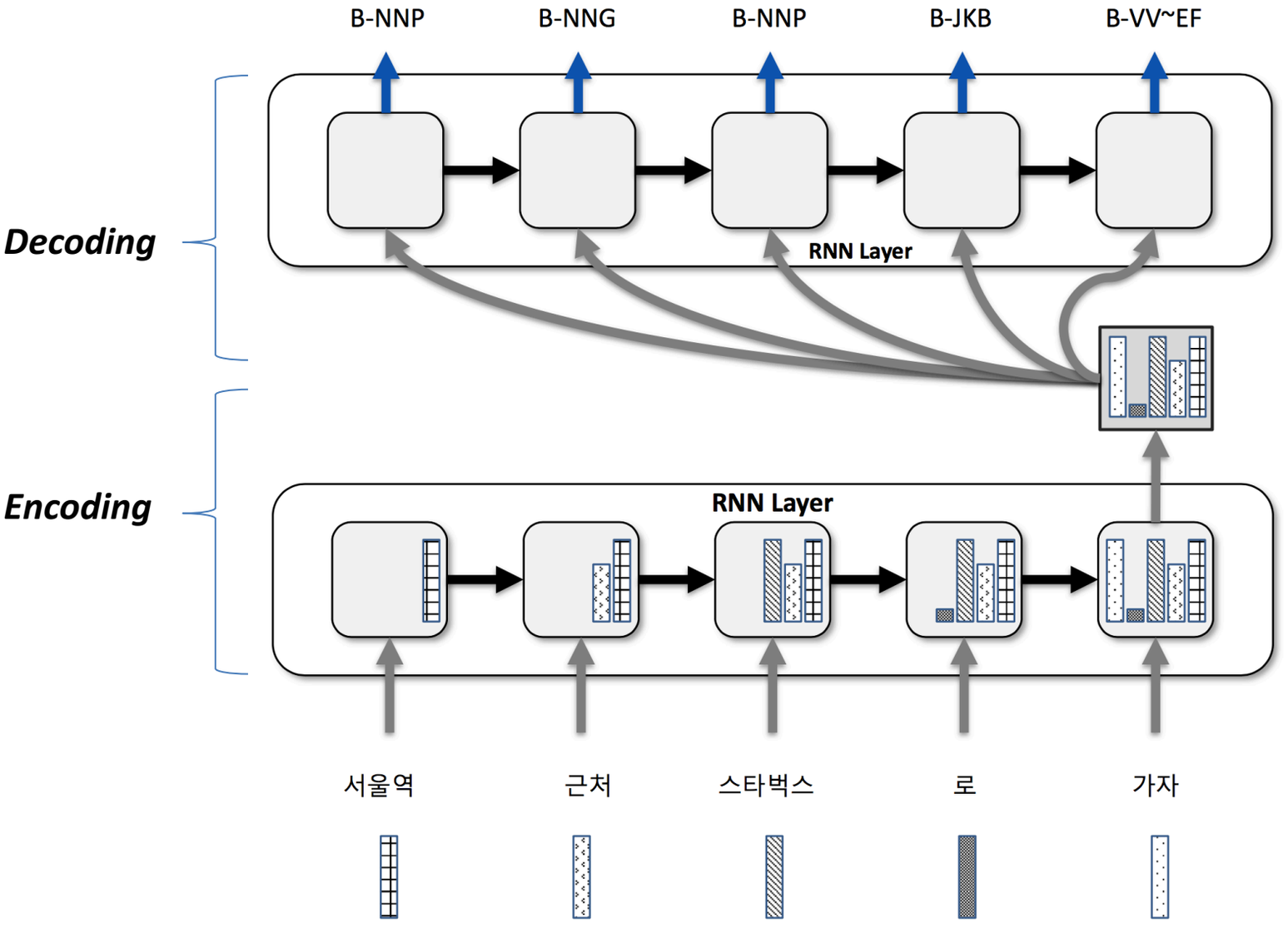
**RNN 에 누적된 정보가 결국**
**Sequence 의 Vector Form 일 것이다.**



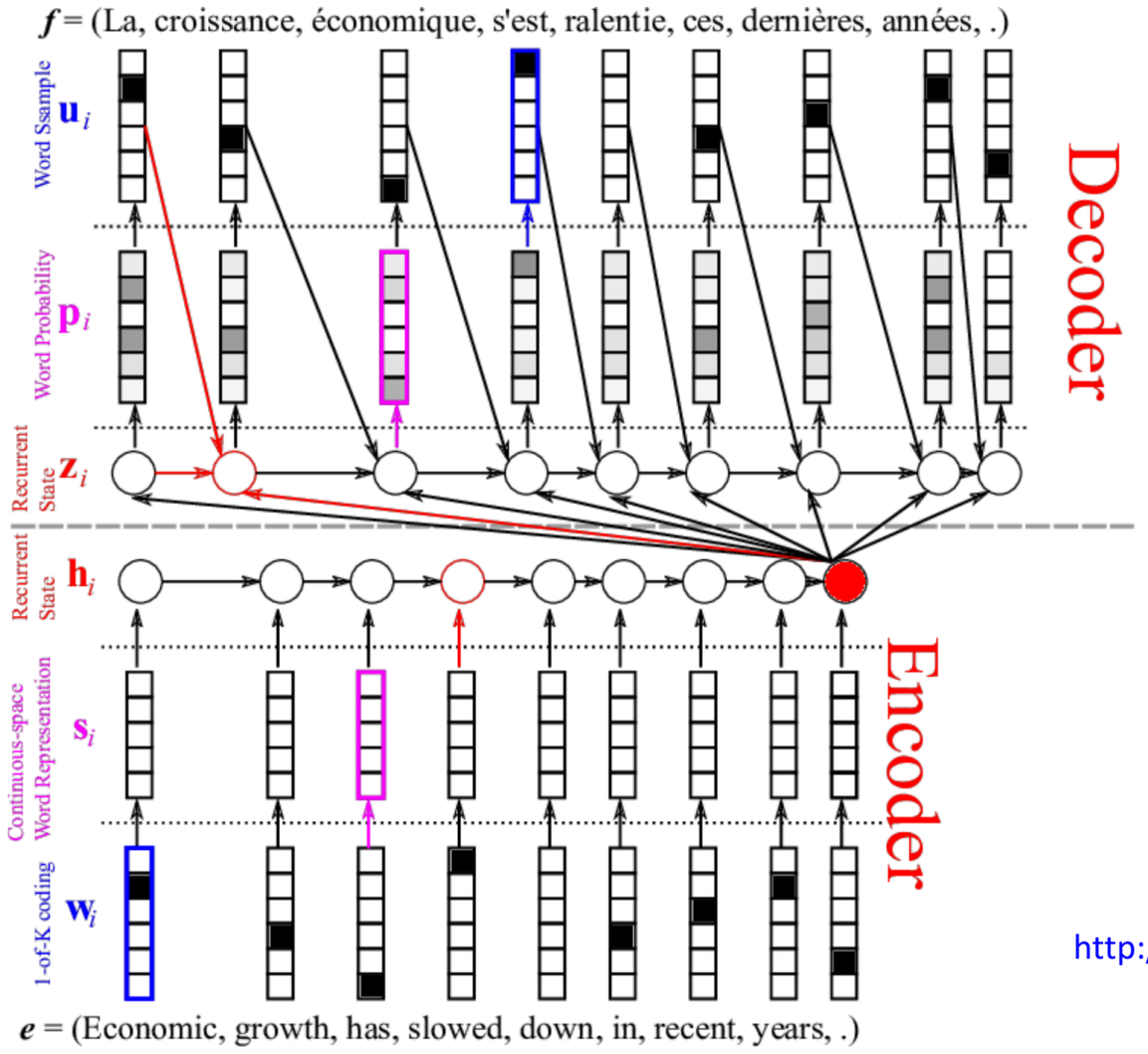
# Sequence Decoding



# Sequence Encoding-Decoding Approach



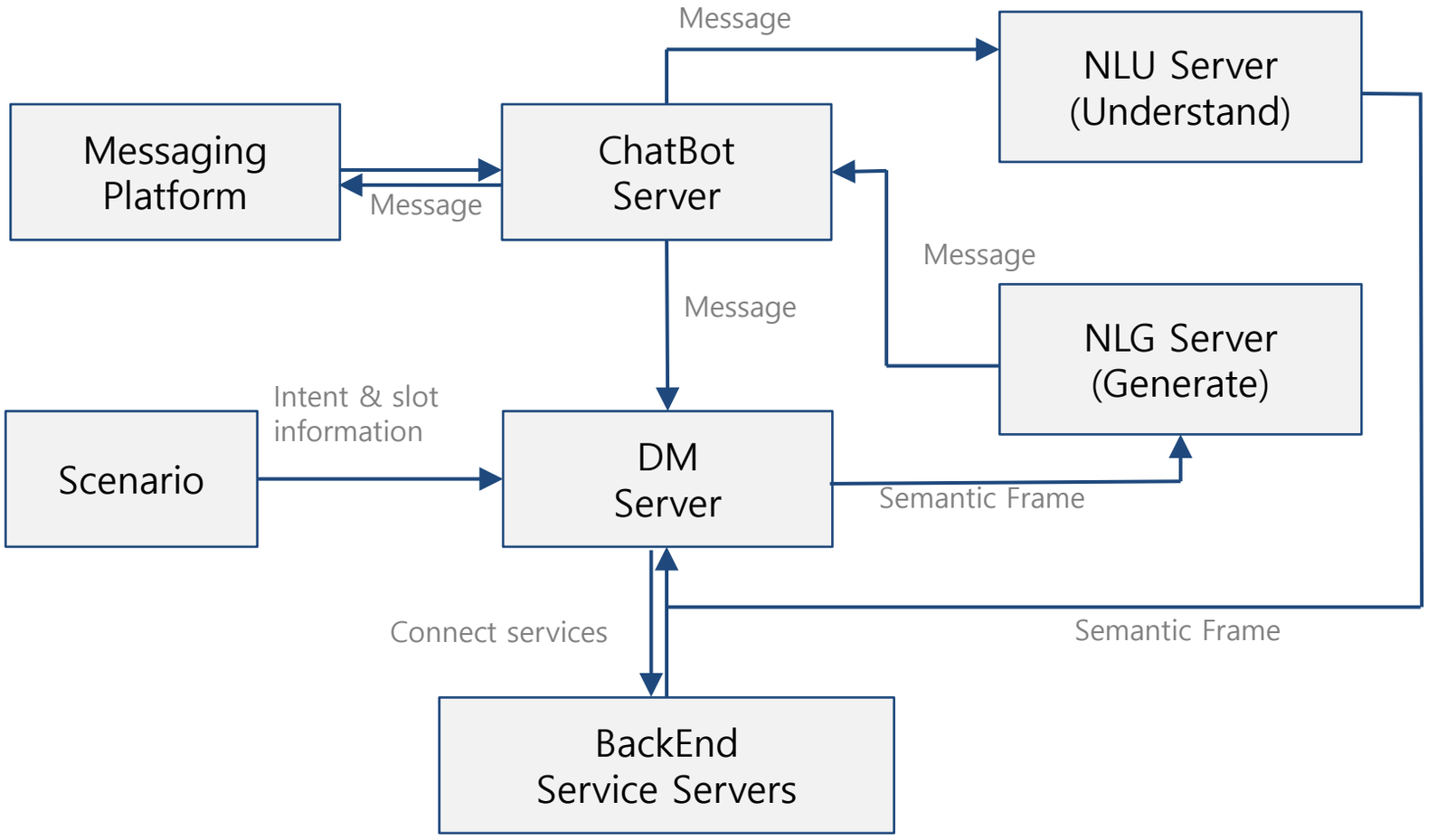
# RNN Encoder-Decoder for Machine Translation




Cho et al. (2014)
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-2/

# MAKE CHATBOT

# CHATBOT System Architecture



# 나만의 CHATBOT을 만들어보자 !!

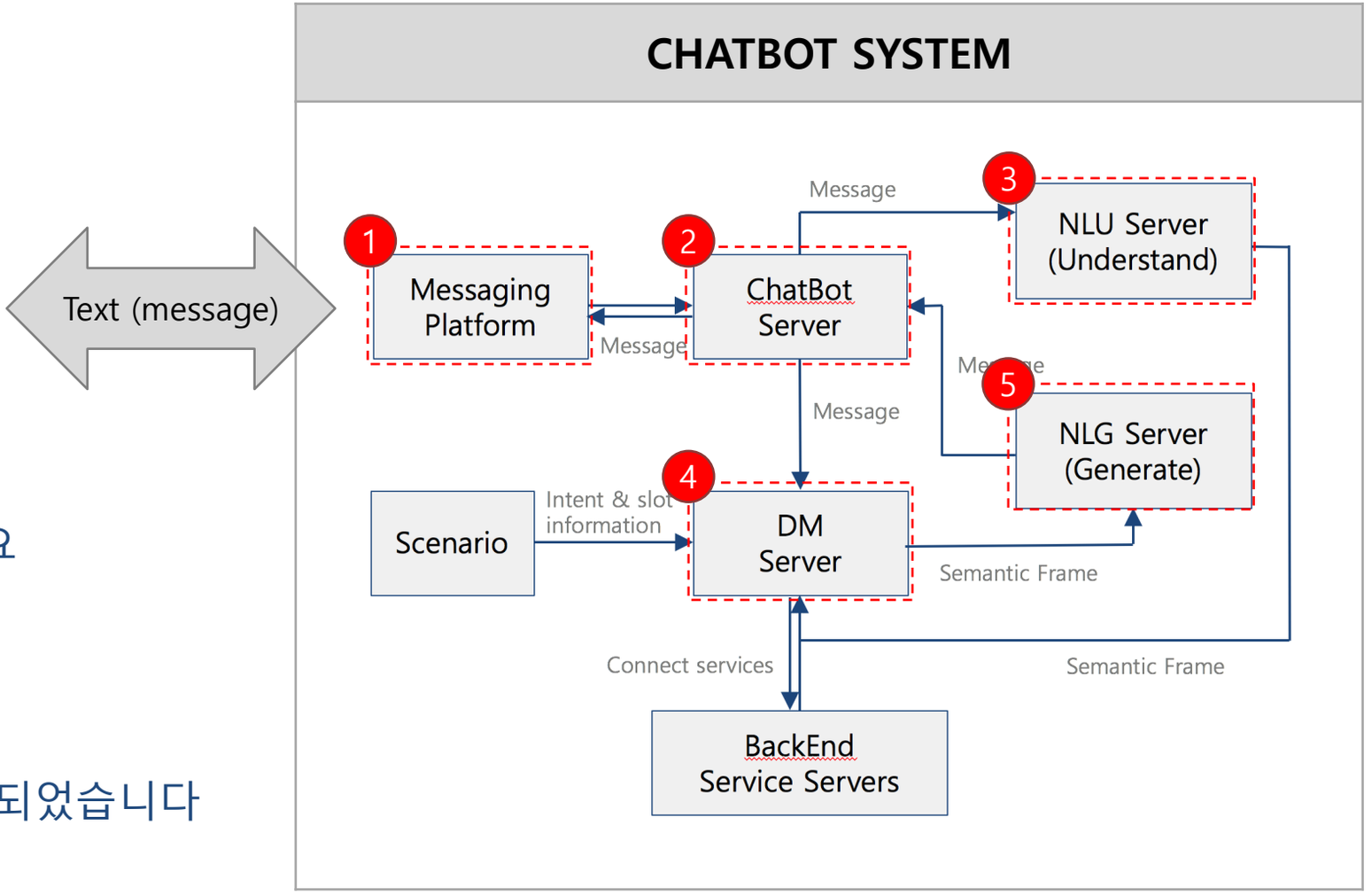
피자 주문 챗봇을 어떻게 만들지?

피자를 주문하려면 피자 종류도 여러가지고, 사이즈도 다양하고,
장소와 날짜, 사이드메뉴도등 다양한데 어떻게 **CHATBOT**으로 만들 수 있을까?

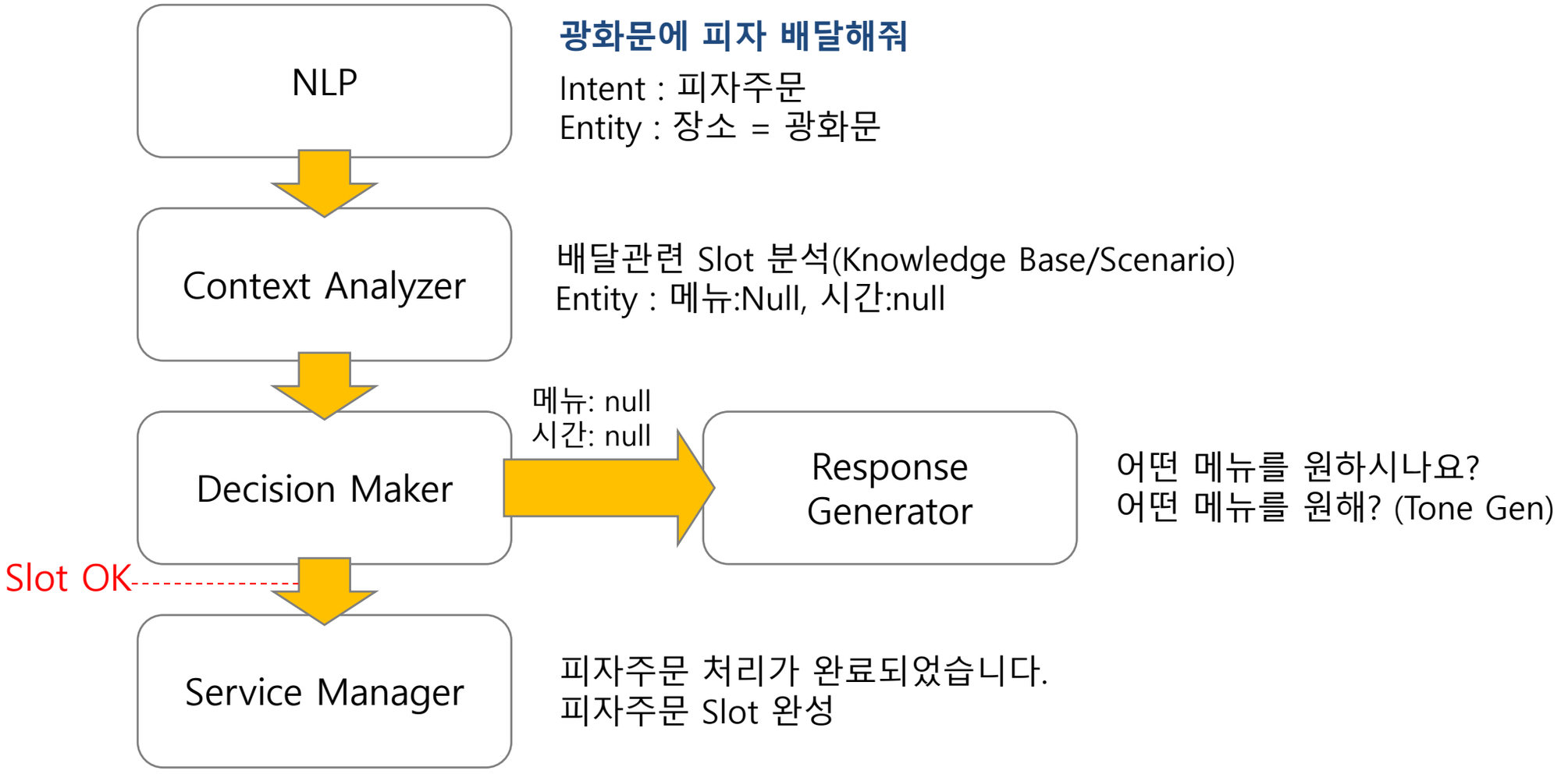
- 피자주문과 관련된 스토리가 구성되야함
- 딥러닝과 적당한 로직으로 피자 주문 Bot을 만들어보자 !!

# CHATBOT System

사용자: 피자 배달해줘
봇 : 사이즈를 선택해주세요
사용자 : 라지
봇 : 장소를 입력해주세요
사용자 : 광화문
봇 : 피자주문 처리가 완료되었습니다



# CHATBOT Interface Flow



# Story slot의 구성 (Frame-based DM)

피자 주문하고 싶어 → **피자 주문 의도 파악**

| Pizza Slot | |
|---|---|
| Size | |
| Type | |
| Side menu | |

**피자 Bot의 스토리 구성**

1) 어떤 사이즈를 원하시나요?
2) 어떤 종류를 원하시나요?
3) 사이드 메뉴는 필요하신가요?

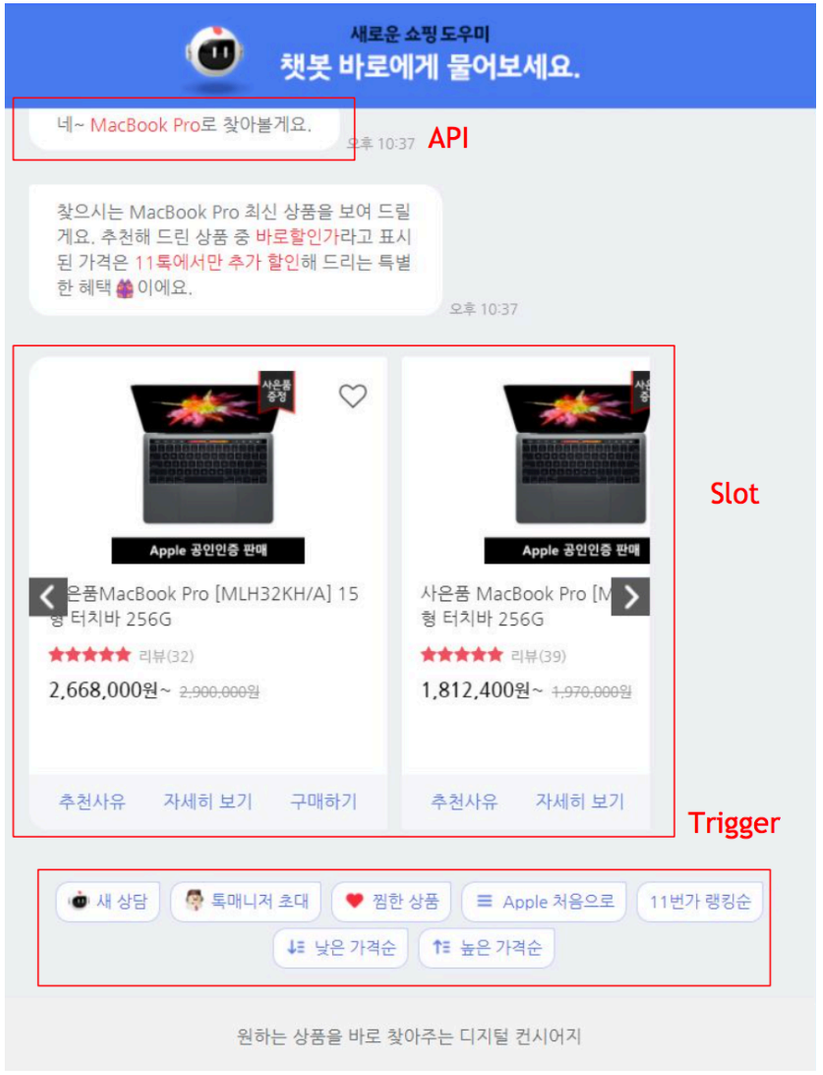
페파로니 피자로 라지 사이즈에 콜라추가해주세요
→ Entity 추출

| Pizza Slot | |
|---|---|
| Size | Large |
| Type | Pepperoni |
| Side menu | cola |

**NER처리 및 Slot 구성**

**서비스 연결 (Slot API Call)**

# Story slot의 구성 (Frame-based DM)




1. 맥북 프로 검색해줘
2. 전처리 → 맥북 프로 NER
3. 맥북프로 → 대표 Entity처리 → MacBook Pro API Call
4. 검색결과 출력
5. 상세 서비스 조회를 위한 Slot 출력
6. 새상담 원할 경우 새상담 클릭

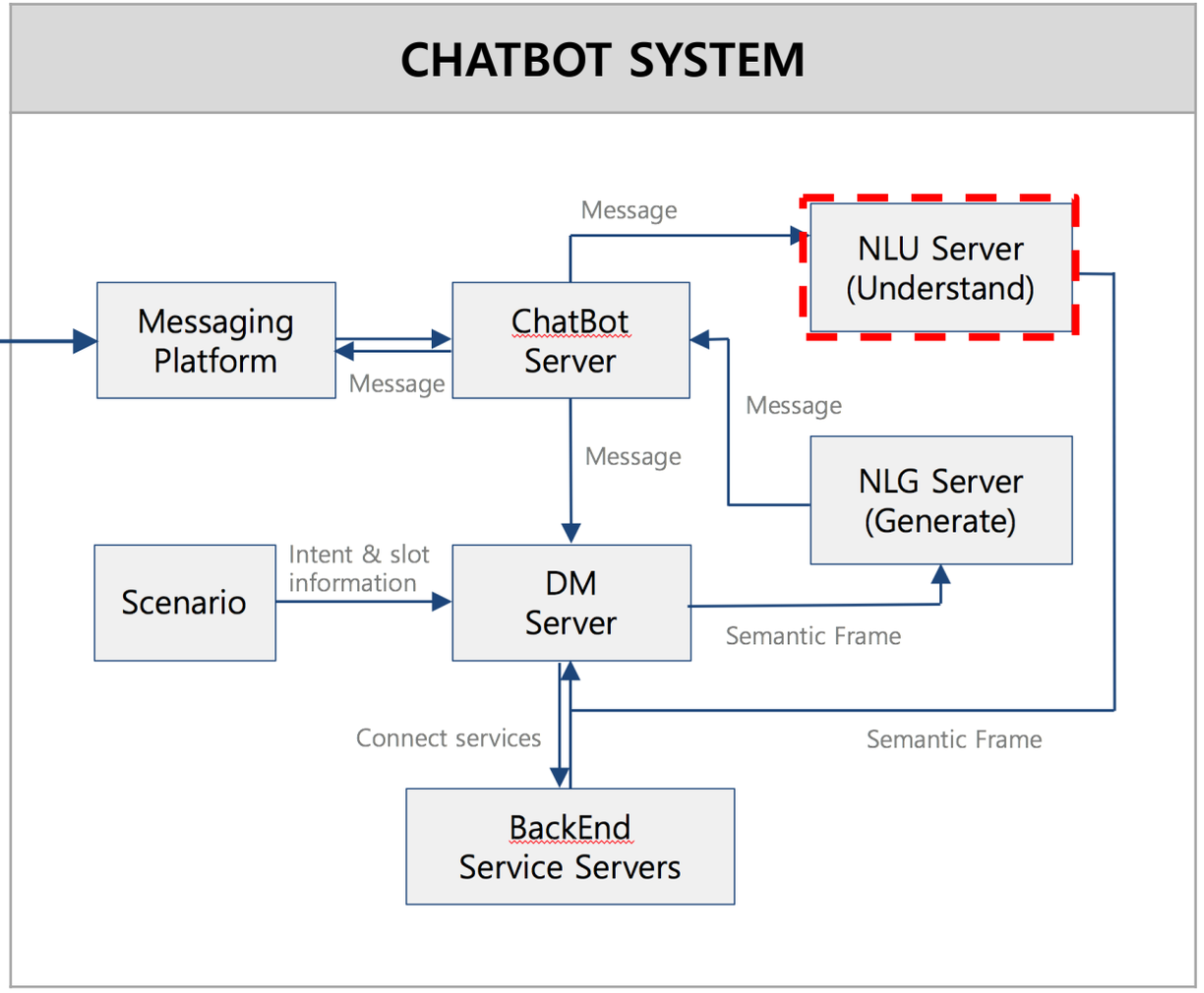
Slot를 선택할 수 있게 화면에 출력함으로써 챗봇의 정확도를 대폭 향상시킬 수 있음 (해당 Frame안에서만 선택할 수 있기에...)

ex) "삼성 노트북" 쳐보면 Slot별 선택

# CHATBOT System

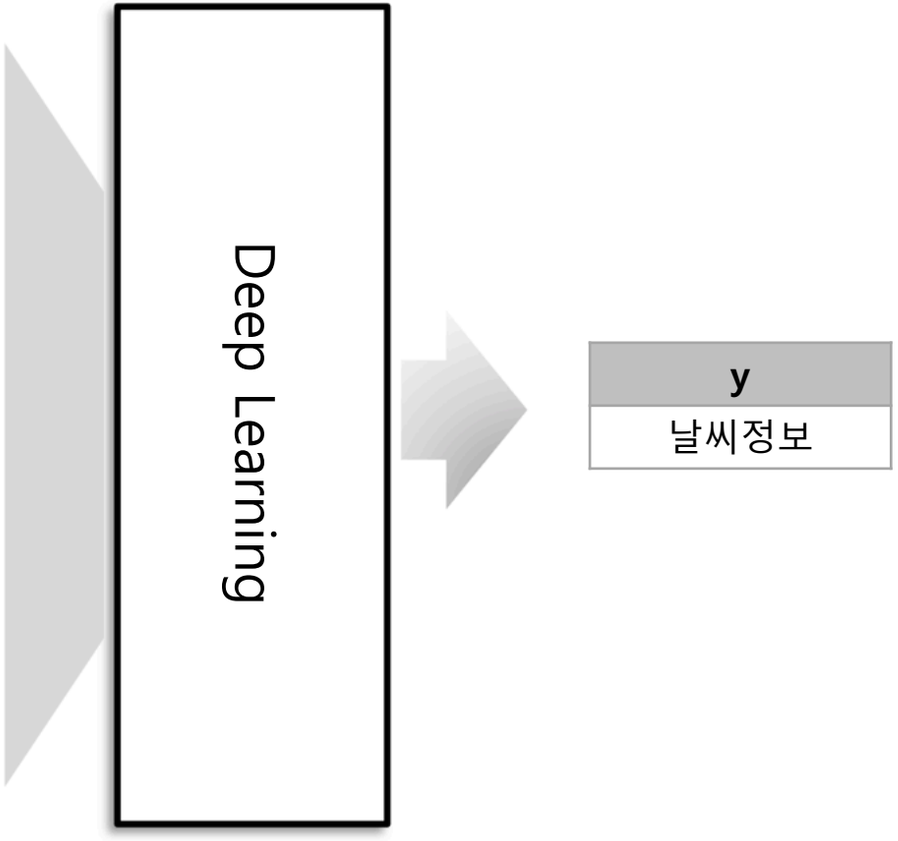
NLU를 어떻게 한다는거지?

- 데이터 수집
- 전처리, Vector 변환
- **Intent(의도) 분석**
- **Entity(개체) 인식**



# 의도(Intent) 분석

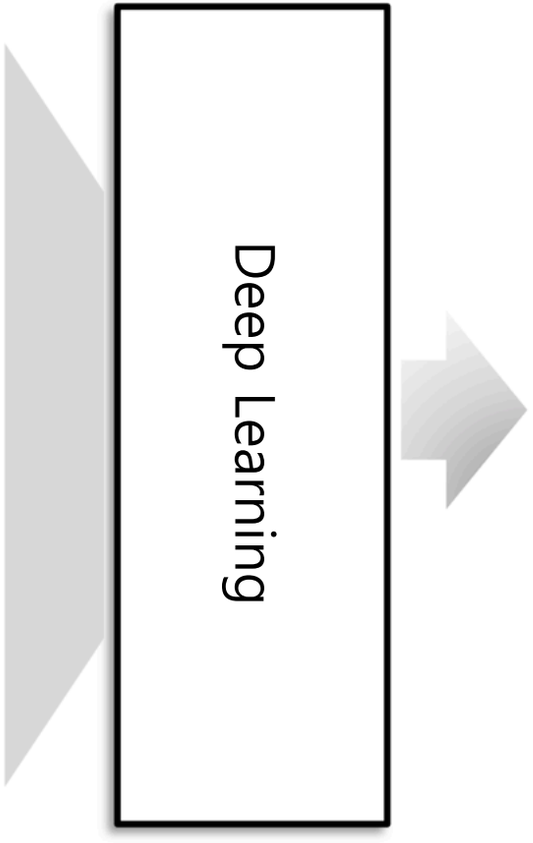
| X | W |
|---|---|
| 내 | 1 |
| 일 | 1 |
| | 1 |
| 날 | 1 |
| 씨 | 1 |
| | 1 |
| 알 | 1 |
| 려 | 1 |
| 줘 | 1 |
| Φ | 0 |
| Φ | 0 |




| y |
|---|
| 날씨정보 |

# 개체(Entity) 인식, NER (Named Entity Recognition)

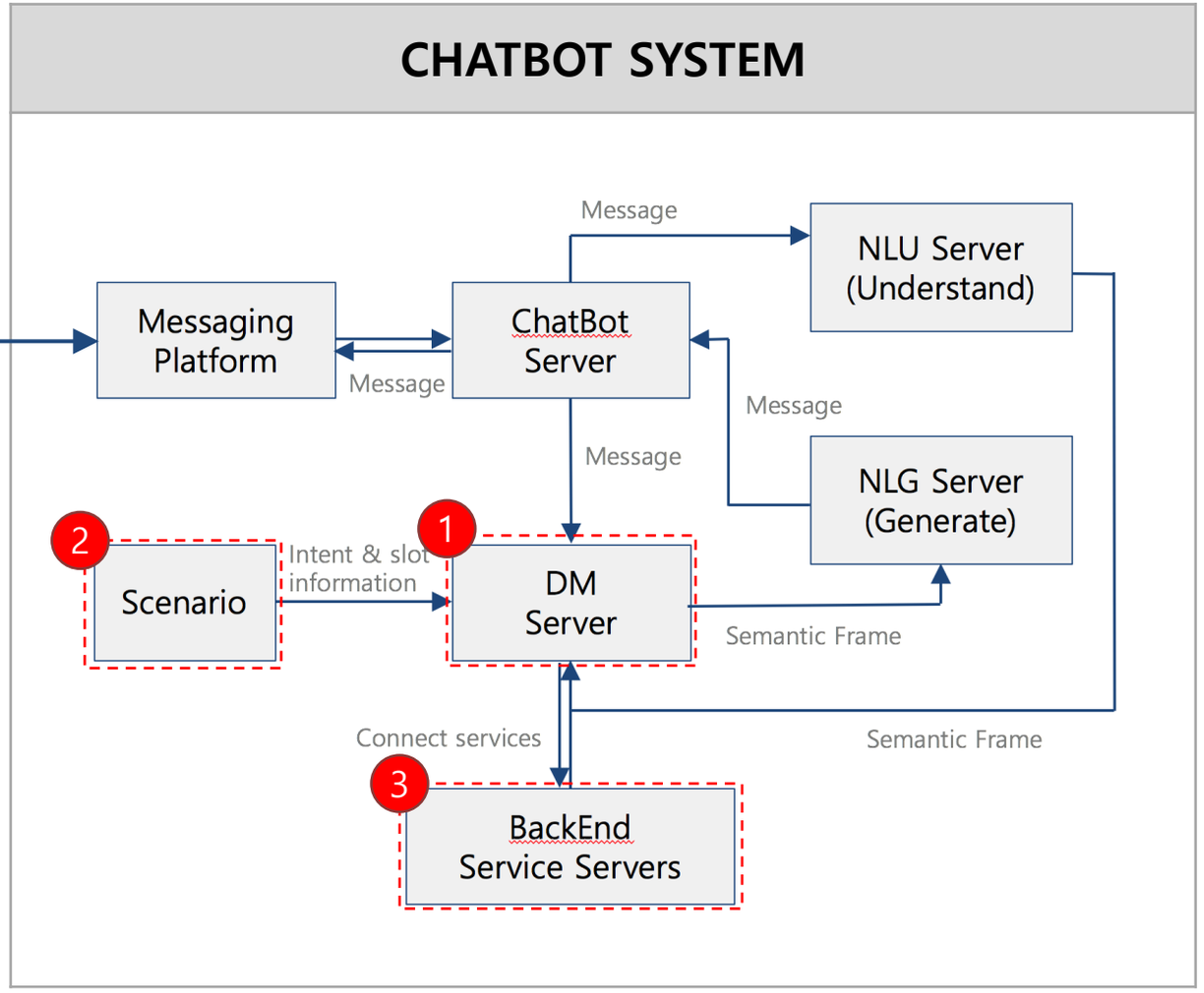
| X | W |
|---|---|
| 내 | 1 |
| 일 | 1 |
| 우 | 1 |
| 리 | 1 |
| 는 | 1 |
| 서 | 1 |
| 울 | 1 |
| 에 | 1 |
| 가 | 1 |
| 요 | 1 |
| Φ | 0 |




| y |
|---|
| B-DT |
| I-DT |
| O |
| O |
| O |
| B-LC |
| I-LC |
| O |
| O |
| O |
| O |

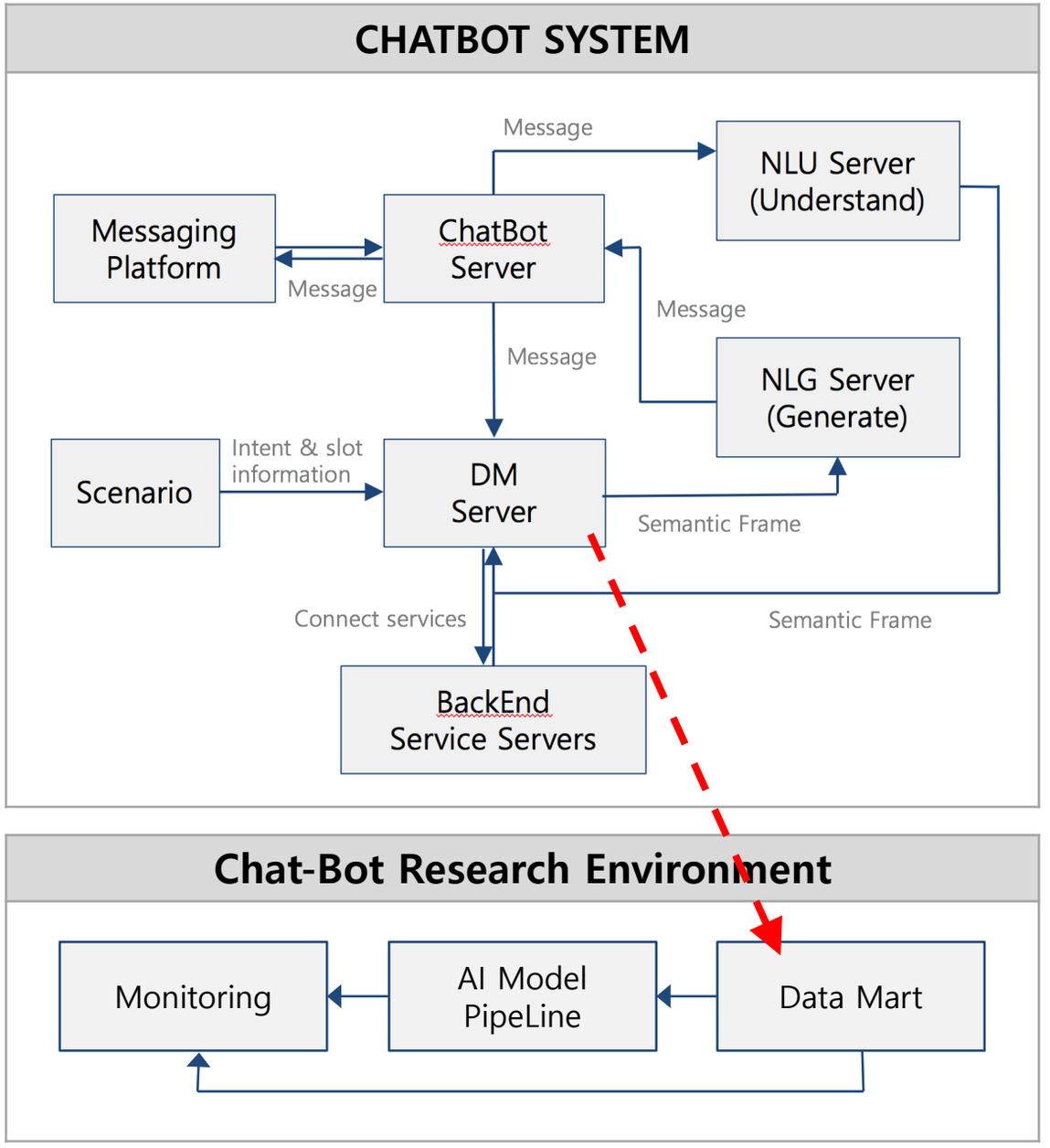
# CHATBOT System

의도도 파악했고 Entity도
알아냈으니 서비스도 만들어보자



# 실무에서 발생하는 문제와 해결 Tips

# Data를 어떻게 얻는가?

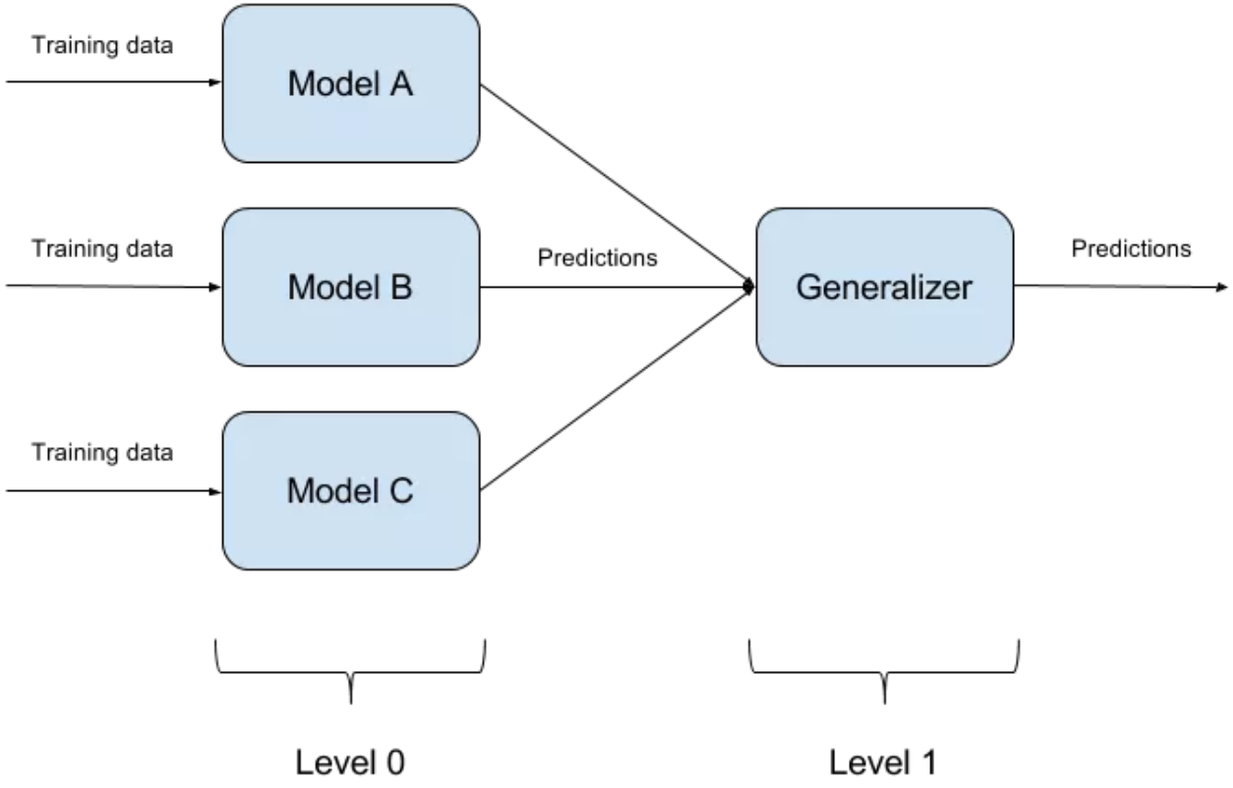



서비스(베타서비스)를 진행하며 사용자들의 데이터를 수집

모델을 주기적으로 또는 데이터가 쌓이는 양에따라 업데이트시킴

# Ensemble and Voting

모델의 정합성을 올리기 위해 복수개의 모델과 로직으로 보완 (Scoring / Voting)
의도를 찾는 경우 여러모델을 비교하여 가장 근접한 값을 찾는다

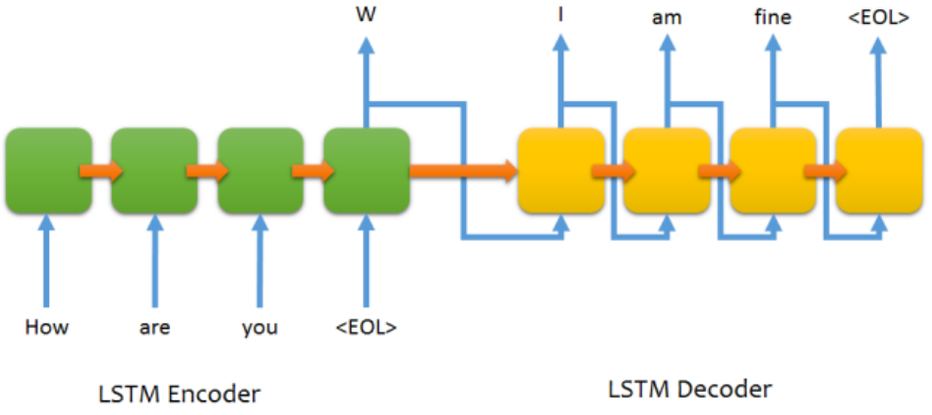


# Trigger 처리

## Rule-based 로 처리

직원 : XXX 사원에게 사랑한다고 톡 보내줘
챗봇 : 너무 쉽게 사랑하지 마세요.
직원 : 니가 먼데 내 사랑을 논해
챗봇 : 학습중이라 아직 잘 모르는게 많아요.
직원 : ㅋㅋㅋㅋ
챗봇 : ㅋㅋㅋ

## Seq2Seq Q&A 모델




[안녕, 사랑, ㅋㅋㅋ] 등에 Trigger를 적용하고
이에 확보된 Data를 Seq2Seq모델에 학습시켜 NLP전처리 모델로 사용

# Tone Generator

**말투를 다르게 만듬** (타겟층, 시장에 따라.. )

주문이 완료되었습니다 (일반)
주문이 완료되었단다 (공손)
주문이 완료되었어요 (존대)
주문이 완료되었다니깐 (짜증)

**Seq2Seq Model활용**
Encoder에 명사등 구성 Decoder에 명사+조사 구성

# LET’S MAKE CHATBOT !!

# 감사합니다