

# 시계열, LLM

구름

도시공학과 일반대학원

한양대학교

# 1. 시계열 분석

2. RNN

3. LSTM

4. 언어모델

5. Attention

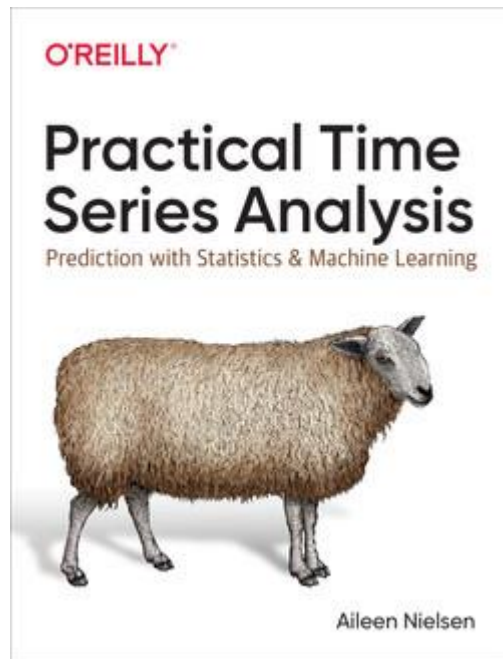
6. Transformer

7. GPT

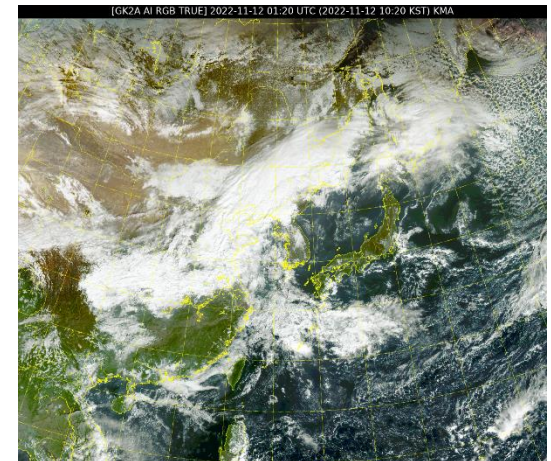
# 시계열 분석 (Time Series Analysis)

*Time series analysis* is the endeavor of extracting meaningful summary and statistical information from points arranged in chronological order.

<https://www.oreilly.com/library/view/practical-time-series/9781492041641/>



심장 박동



날씨

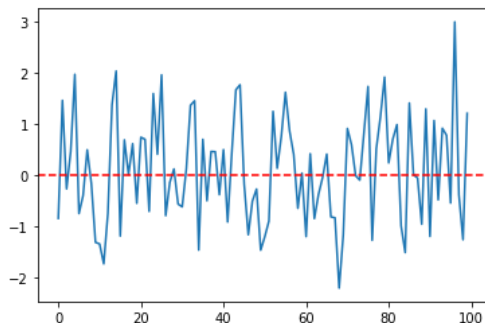


주가지수

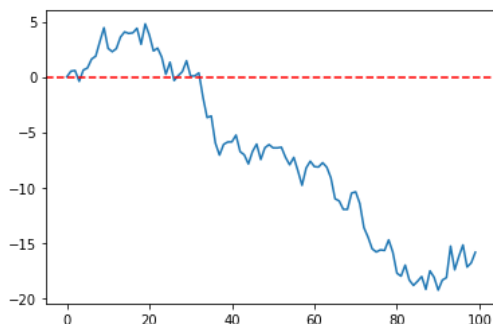


천문학

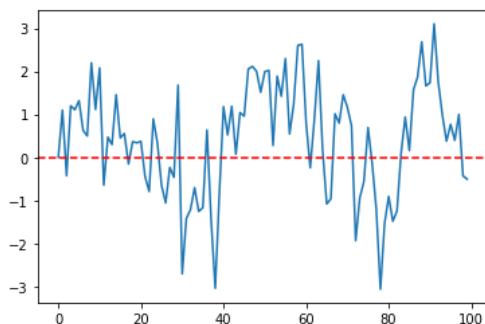
## 시계열 데이터



```
1 #백색 잡음 (White Noise Process)
2
3 data = np.random.normal(0, 1, 100)
4
5 plt.plot(data)
6 plt.axhline(0, 0, 100, color='red', linestyle='--')
7 plt.show()
```



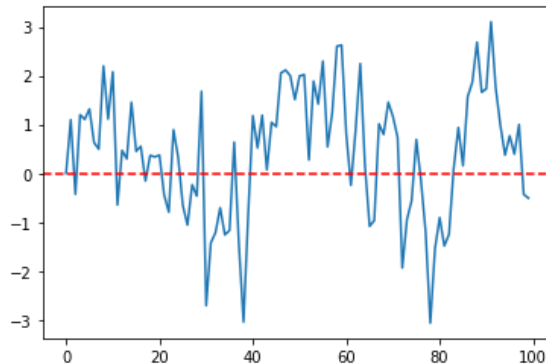
```
1 #확률보행과정 (Random Walr Process)
2
3 data = np.array([0])
4
5 for i in range(99):
6     data = np.append(data, data[-1] + np.random.normal(0, 1, 1))
7
8 plt.plot(data)
9 plt.axhline(0, 0, 100, color='red', linestyle='--')
10 plt.show()
```



```
1 #정상 확률 과정 (Stationary Proecess)
2
3 data = np.array([0])
4 A_corr = 0.5
5
6 for i in range(99):
7     data = np.append(data, data[-1] * A_corr + np.random.normal(0, 1, 1))
8
9 plt.plot(data)
10 plt.axhline(0, 0, 100, color='red', linestyle='--')
11 plt.show()
```

# AutoRegressive Integrated Moving Average

ARIMA 모형은 정상시계열 데이터에 적합한 모델로 Box-Jenkins Model을 통해 데이터 정상성 여부를 확인 후 사용



AR 모형 :  $x_t = \beta \cdot x_{t-1} + \alpha + \varepsilon_t$

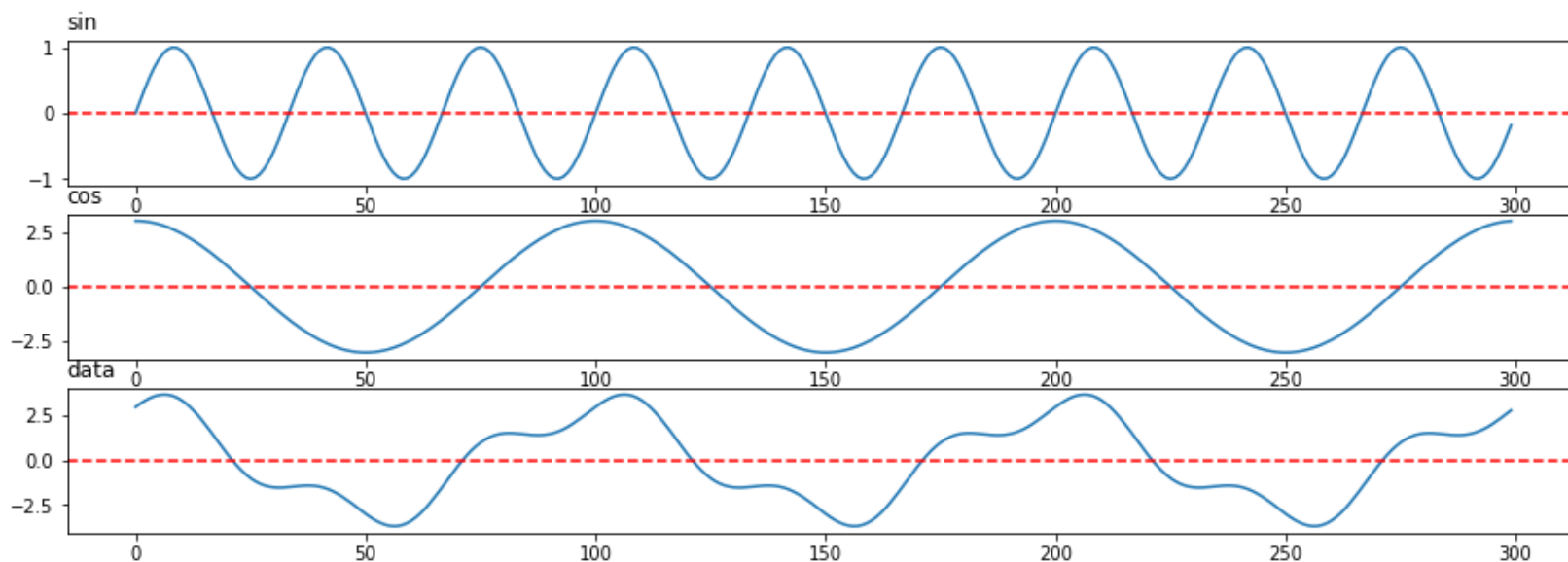
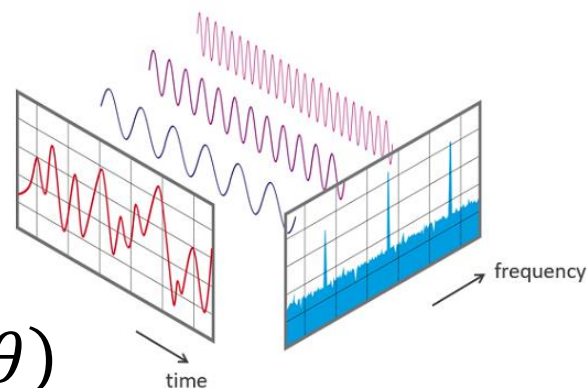
MA 모형 :  $x_t = \alpha + \varepsilon_t + \beta \cdot \varepsilon_{t-1}$

Interated :  $x_t = x_t - x_{t-1}$

# Fourier Transform

Sin, Cos 함수를 이용하여 신호를 분리  
주파수, 음성 인식 등에 활용

$$x_t = a \cdot \sin(b \cdot \theta) + c \cdot \cos(d \cdot \theta)$$

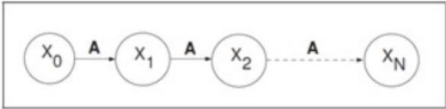


# Hidden Markov Model

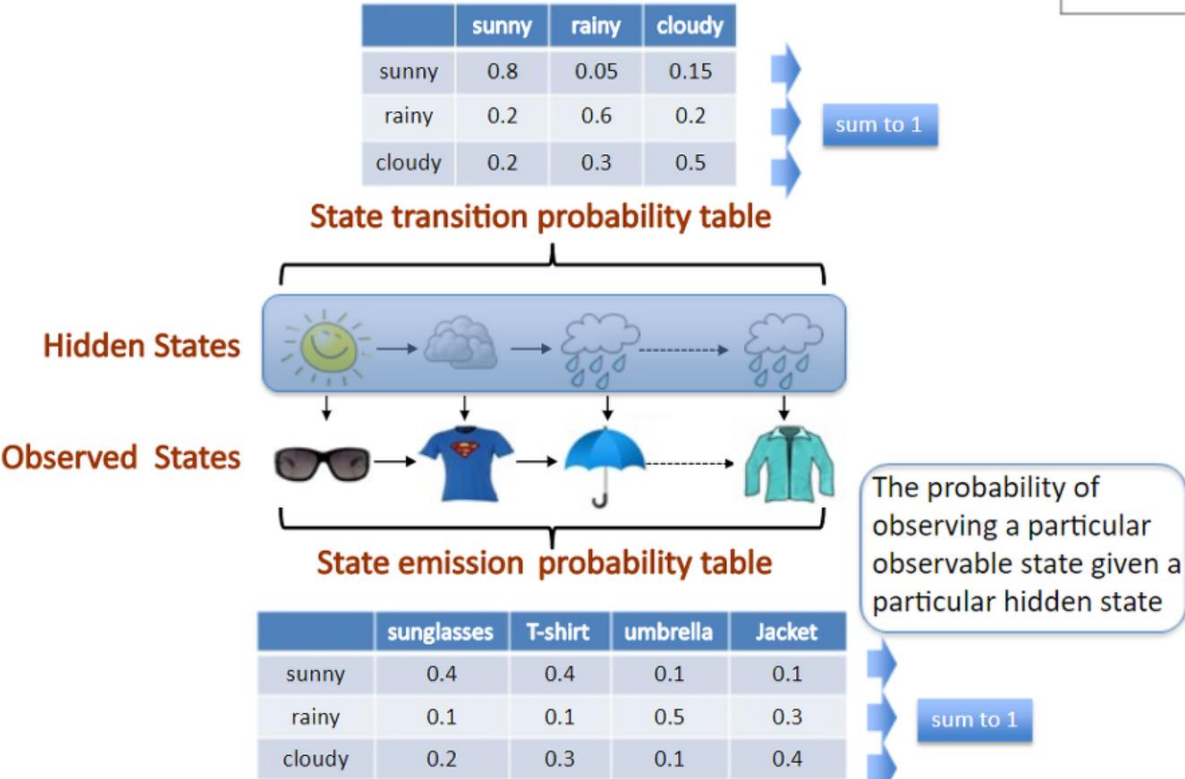
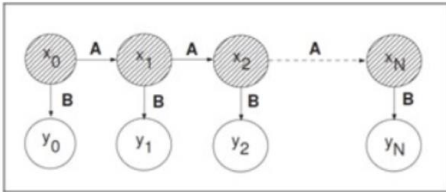
은닉층이 마코프 모델을 따른다는 가정으로 예측  
날씨, 음성인식 등에 활용

<https://bioinformaticsandme.tistory.com/53>

A Markov Model



A Hidden Markov Model



# 브라운 운동 (Brownian Motion)

1827년 식물학자 로버트 브라운이 꽃가루 입자 관찰에서 발견

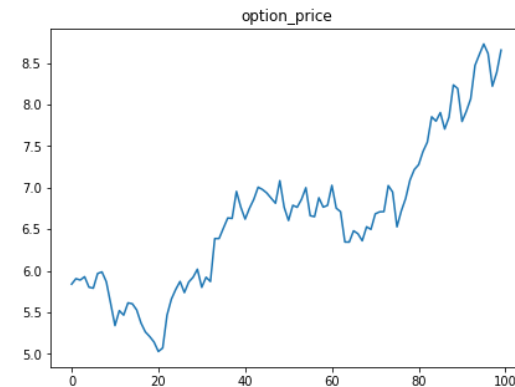
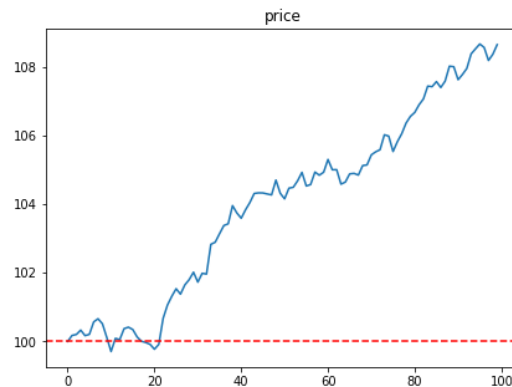
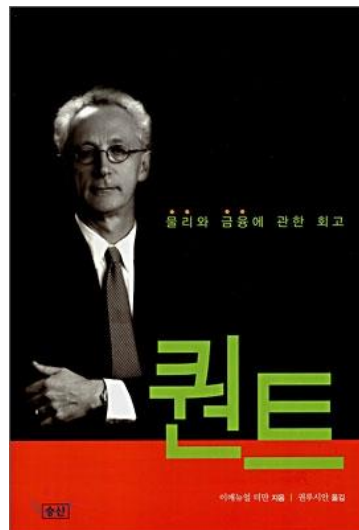
아인슈타인이 브라운 운동 방정식을 제시, 물리학자인 페렝이 증명

루이 바슐리에 1900년 투기이론으로 금융시장 가격 변동을 브라운운동으로 모형화

1950년대 중반 폴 새뮤얼슨이 기하 브라운운동을 발표하며 바슐리에 이론을 수정

1973년 피셔 블랙과 마이런 솔즈는 옵션 가격 결정이론을 발표

<http://www.yes24.com/Product/Goods/2641000>



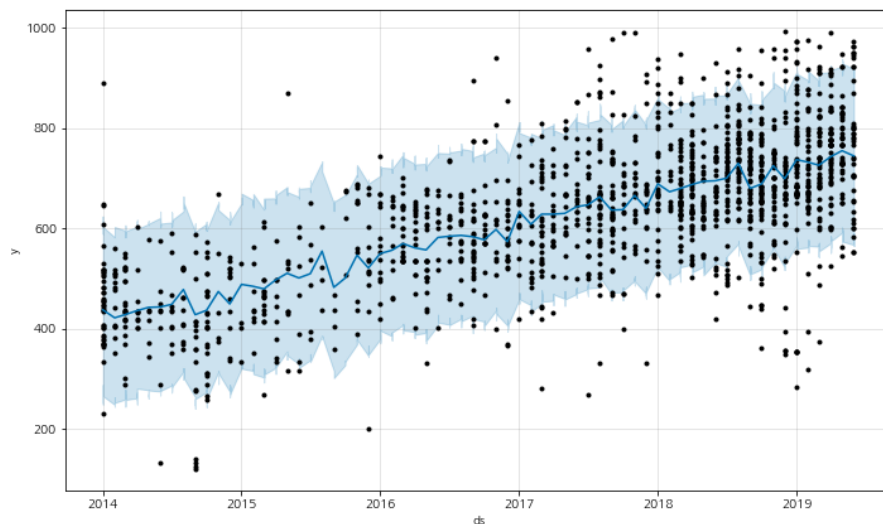


# Prophet - Facebook

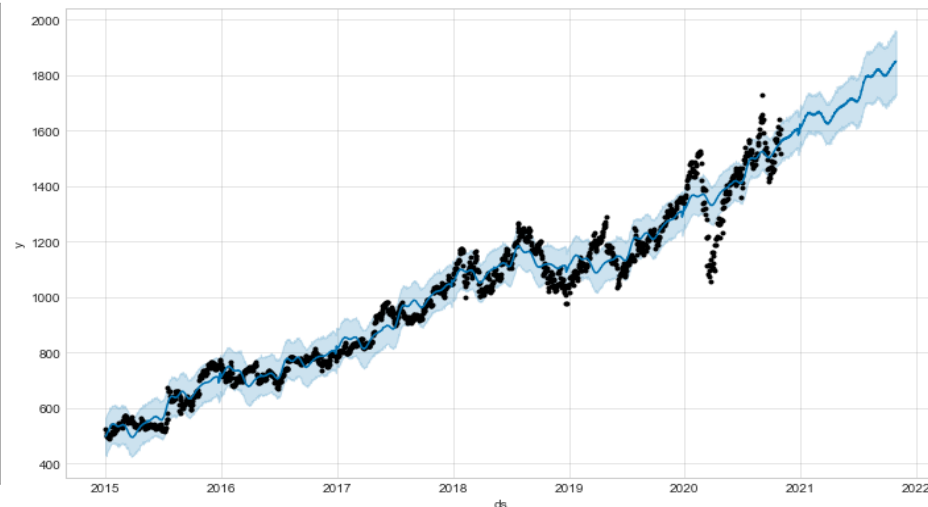


페이스북에서 비즈니스 분야에 적합하게 만든 시계열 예측 알고리즘  
 시계열 추세에 계절적 요소와 휴일 요소를 추가하여 비즈니스에 적합하게 구성  
 데이터에서 변동성 구간, 이상치 검출, 미래 예측, 예측 범위 등 다양하게 제공

<https://facebook.github.io/prophet/>



(빅밸류) 서울 클러스터내 30㎡ 이하 건축년한 0~1년 이하 실거래 단가  
 Prophet 분석 결과

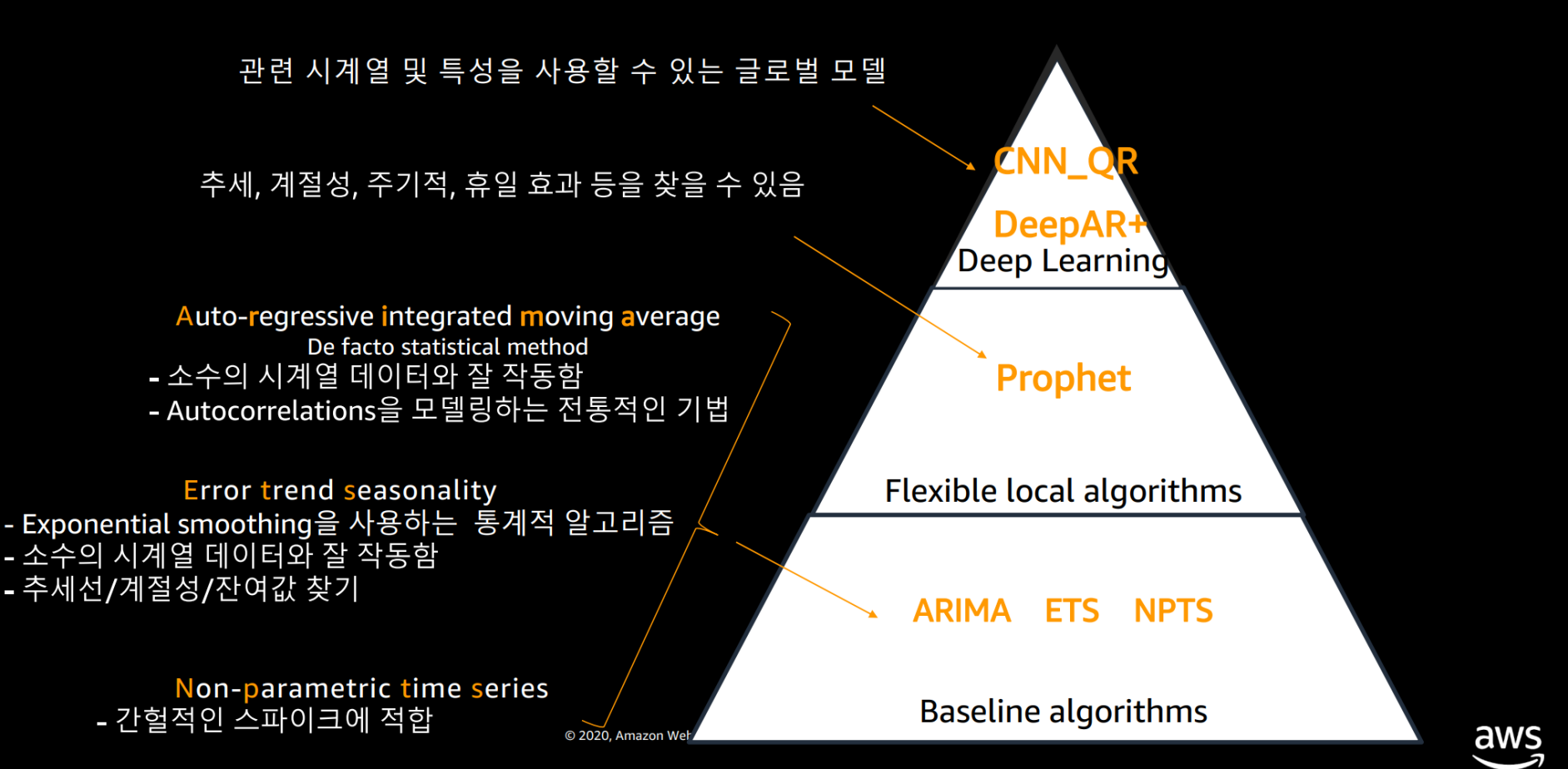


(Medium) Predicting Google's Stock Prices Using Facebook's Prophet  
<https://medium.com/mlearning-ai/predicting-googles-stock-prices-using-facebook-s-prophet-4829c83a8590>

# Deep AR+ - Amazon Forecast

아마존에서 인공지능경망을 이용해 제시한 시계열 분석 라이브러리.

[https://docs.aws.amazon.com/ko\\_kr/forecast/latest/dg/aws-forecast-recipe-deeparplus.html](https://docs.aws.amazon.com/ko_kr/forecast/latest/dg/aws-forecast-recipe-deeparplus.html)



1. 시계열 분석

**2. RNN**

3. LSTM

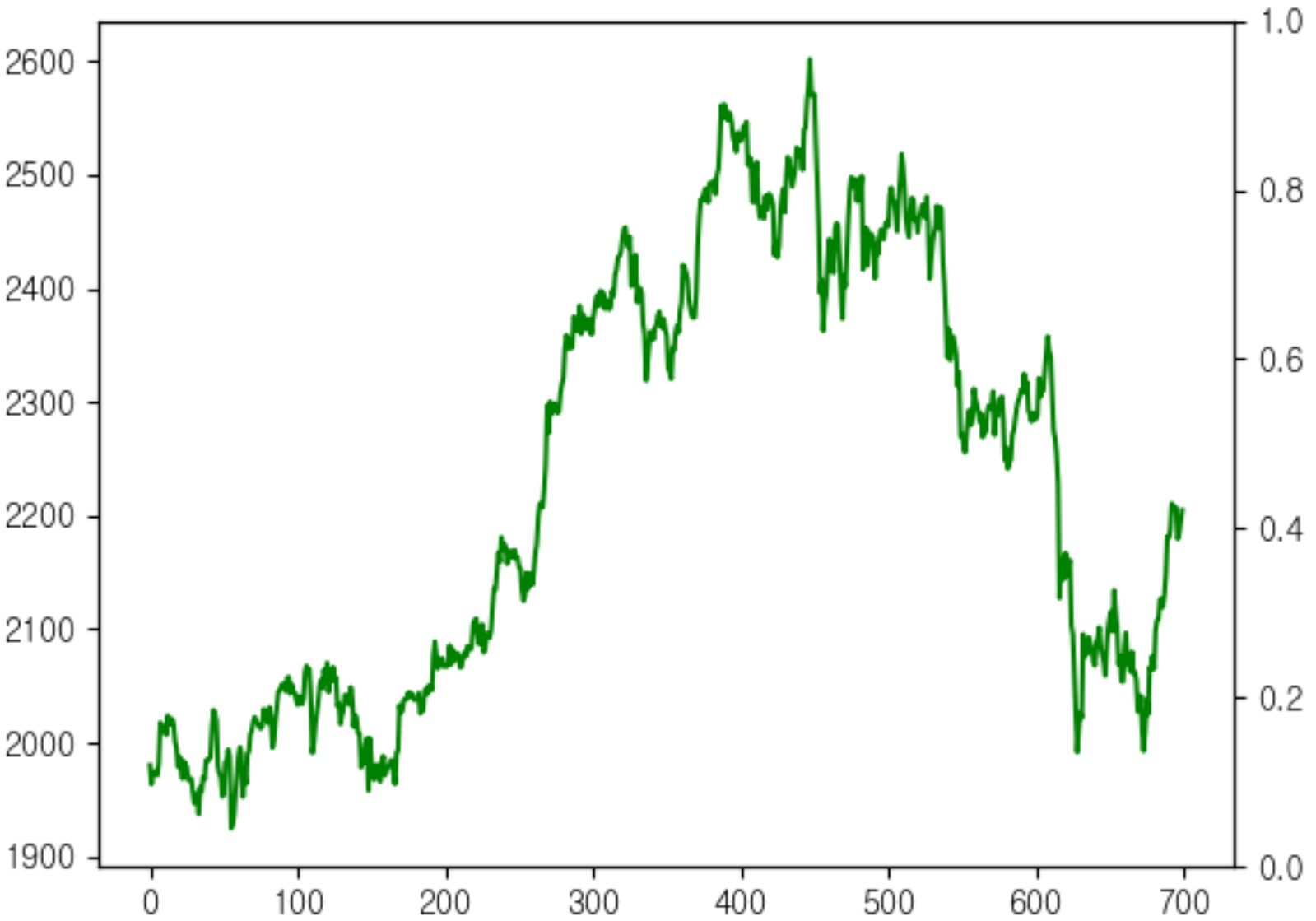
4. 언어모델

5. Attention

6. Transformer

7. GPT

KOSPI 지수 (2016-03-08~2019-01-15)

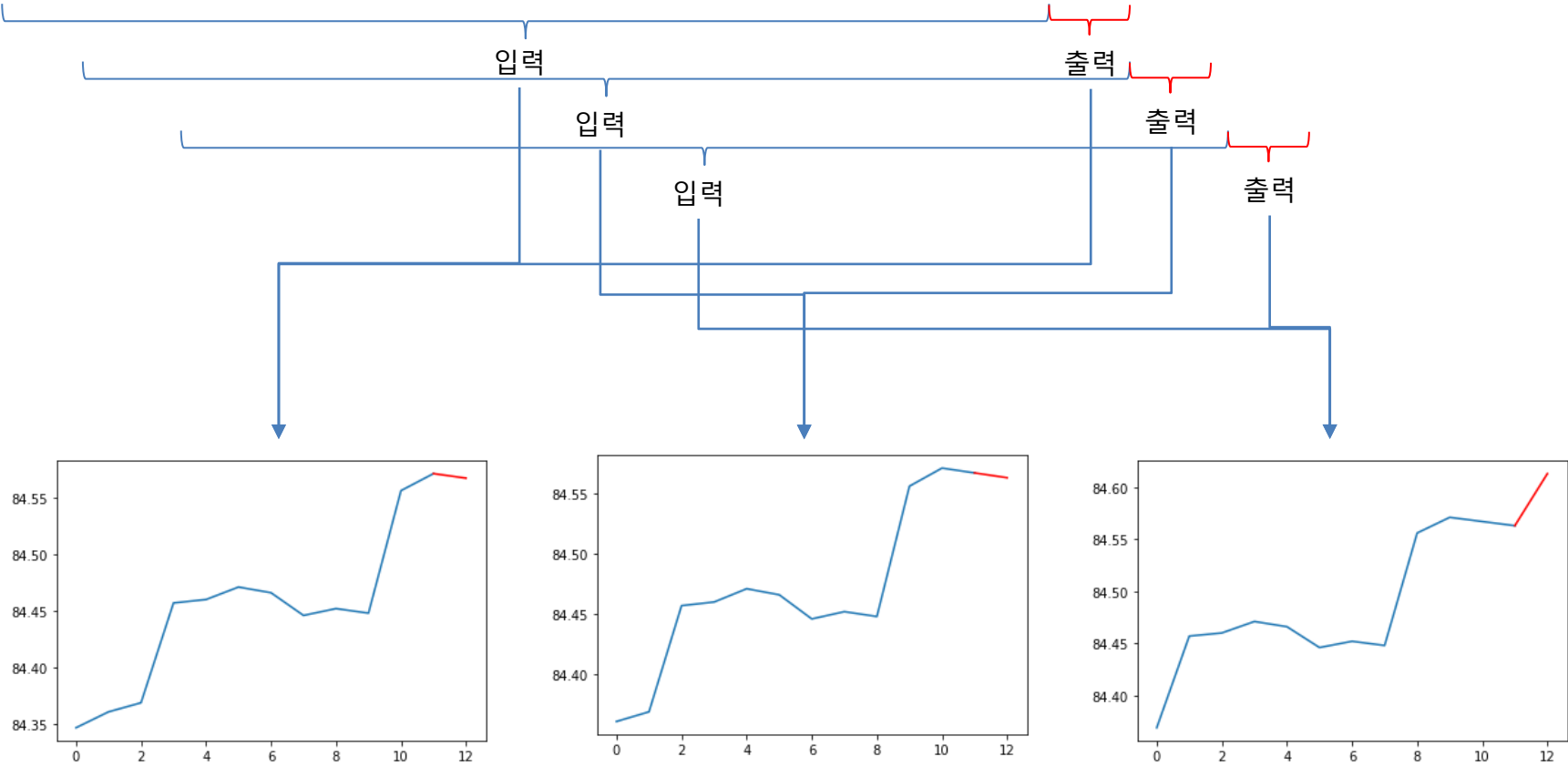


# 데이터 구조

과거 일정기간의 시계열 정보를 입력으로 넣고 그 다음 시계열 정보를 출력으로 데이터셋을 만듦

2016년 3월

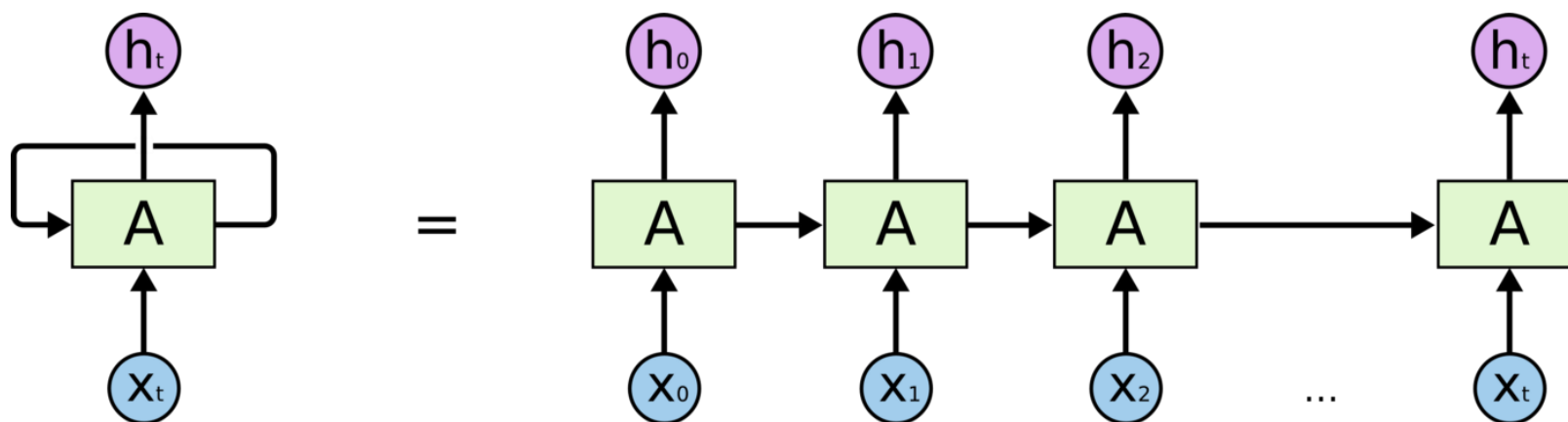
8일	9일	10일	11일	12일	13일	14일	15일	16일	17일	18일	19일	20일	21일	22일	23일	24일	25일
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----



## 순환신경망 (Recurrent Neural Network)

시계열 정보를 학습하는 신경망 모델을 의미

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



# Simple RNN

```
inputs = layers.Input(shape=(ntraining_x.shape[1], 1))
net = layers.BatchNormalization()(inputs)
net = layers.SimpleRNN(1, activation='relu')(net)
net = layers.Dense(1, activation='linear')(net)
model = Model(inputs, net)

model.compile(optimizer='adam', loss='mse')

model.summary()
```

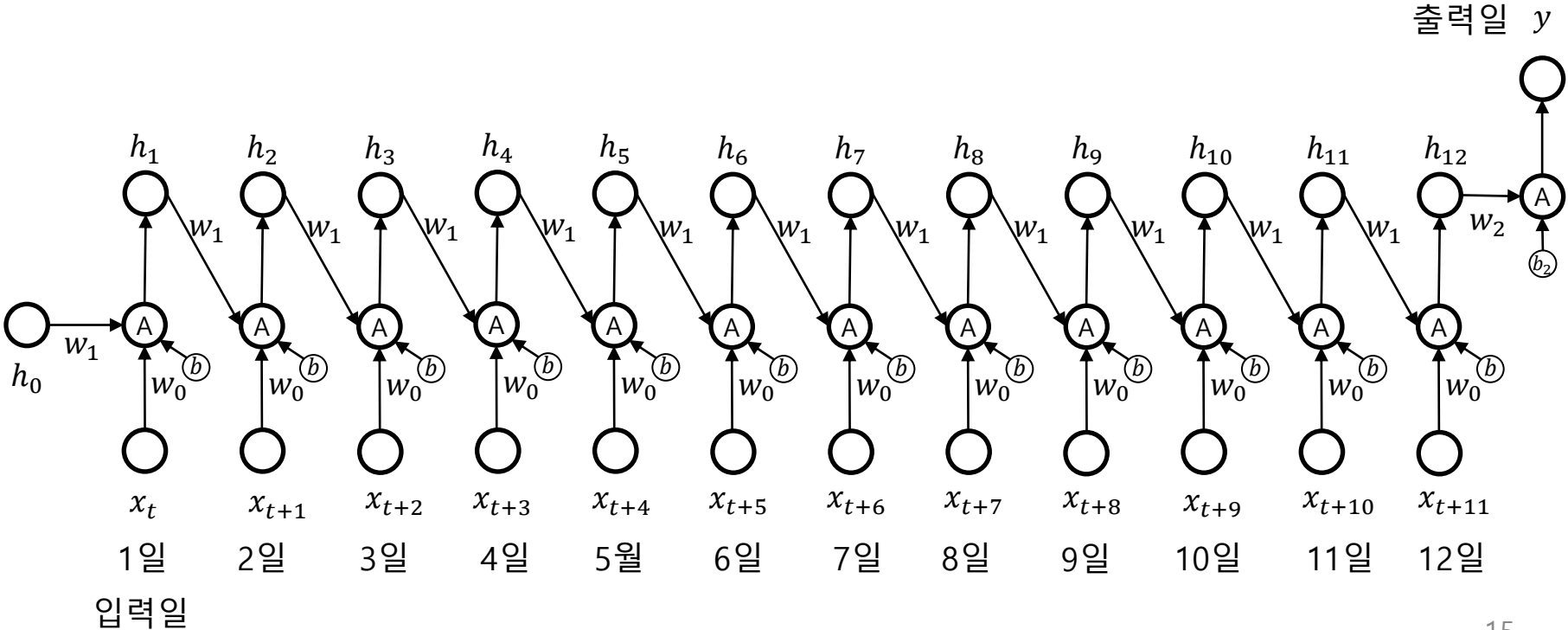
Model: "model\_19"

Layer (type)	Output Shape	Param #
input_20 (InputLayer)	[(None, 12, 1)]	0
batch_normalization_9 (Batch Normalization)	(None, 12, 1)	4
simple_rnn_19 (SimpleRNN)	(None, 1)	3
dense_29 (Dense)	(None, 1)	2

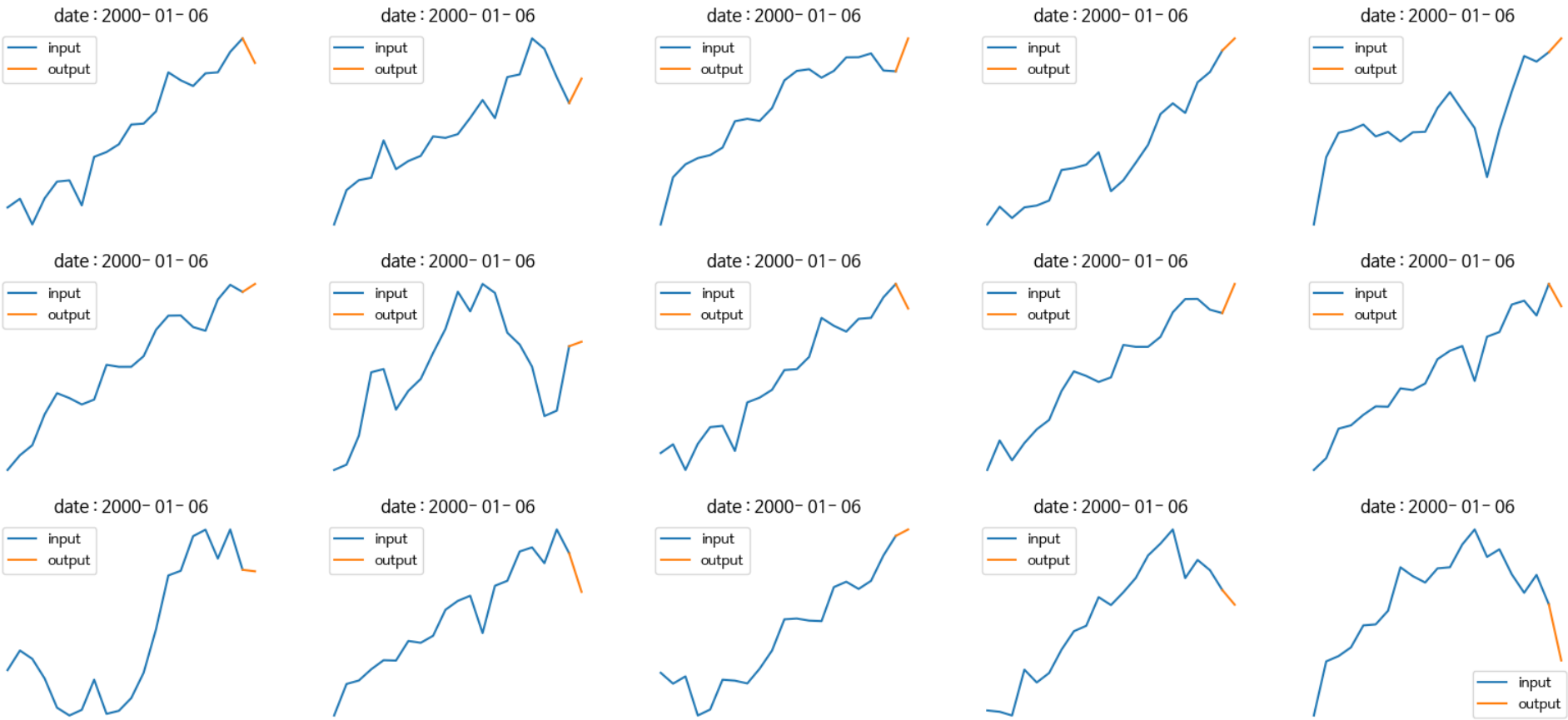
=====

Total params: 9  
Trainable params: 7  
Non-trainable params: 2

=====



# Simple RNN 결과



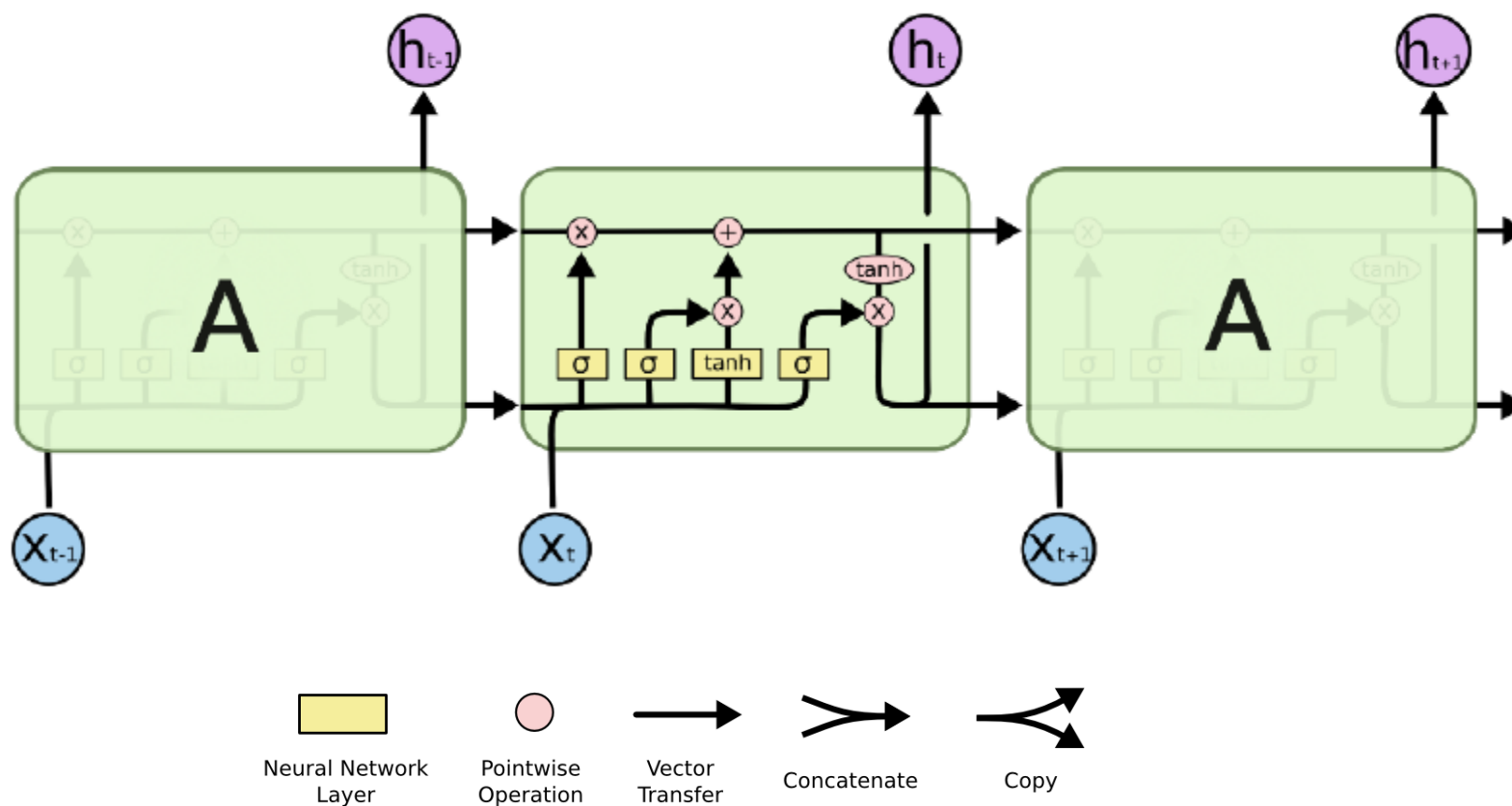


1. 시계열 분석
2. RNN
- 3. LSTM**
4. 언어모델
5. Attention
6. Transformer
7. GPT

## Long Short-Term Memory (LSTM)

RNN은 순차 학습 과정에서 초기 데이터에 대한 학습 전파가 어렵고 오래 걸리는 문제 존재  
LSTM을 통해 이러한 문제를 해결

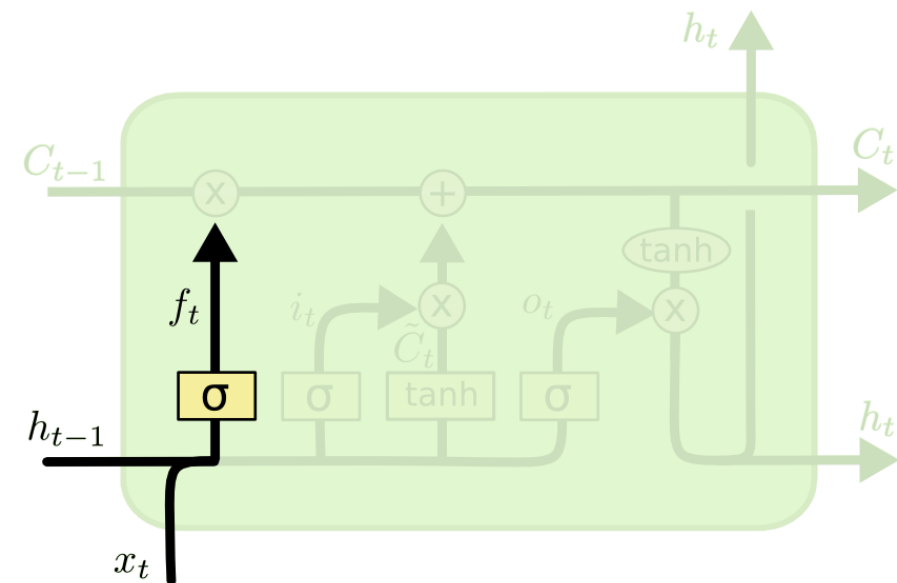
<http://www.bioinf.jku.at/publications/older/2604.pdf>



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

## Forget gate

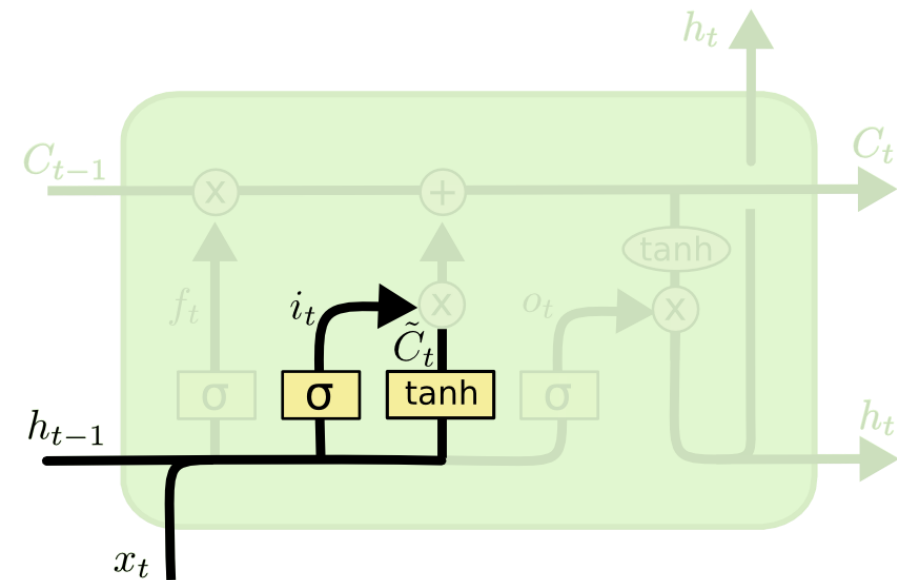
시그모이드 함수를 통해 0 혹은 1의 값을 통해 이전 cell state를 사용할지 안할지를 결정



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

## input gate

하이퍼볼릭 탄젠트를 통해 이번 Cell State를 계산하고  
시그모이드를 통해 이번 Cell state를 이전 Cell State와 합칠지를 결정

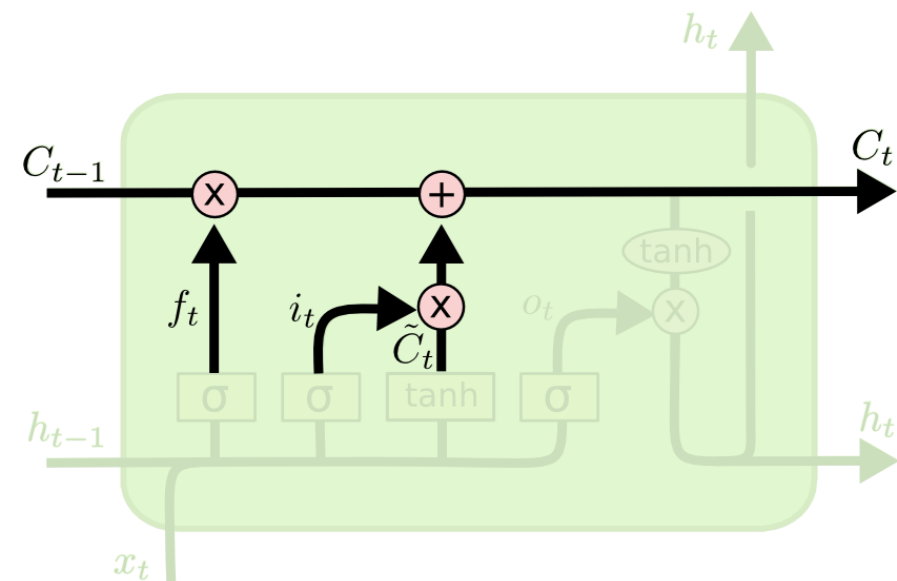


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

## Cell State

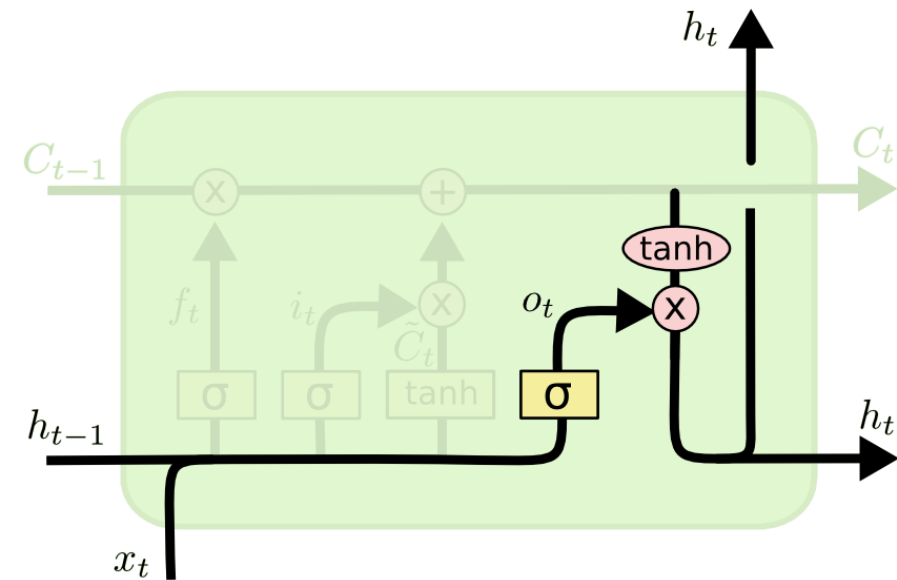
이전 Cell State와 forget gate를 곱하고  
 이번 Cell State와 input gate를 곱해서  
 두 값을 더하여 이번 Cell State를 확정한다.



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

## Hidden State

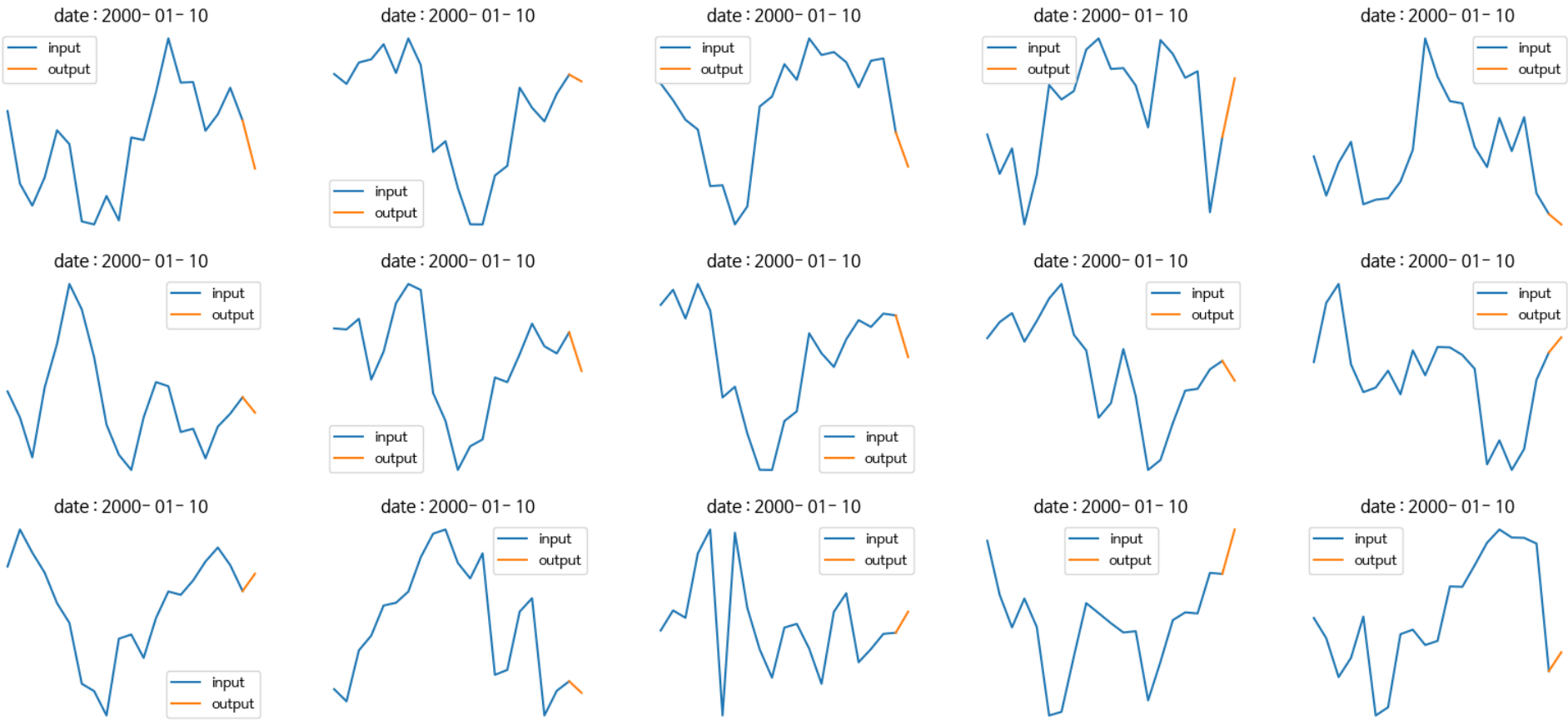
RNN과 동일하게 이전 hidden state와 입력 값을 이용해 히든 스테이트를 계산하고  
 앞에서 만든 Cell State를 하이퍼볼릭탄젠트 함수로 계산한 후 곱한다  
 새로운 hidden state가 완성



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

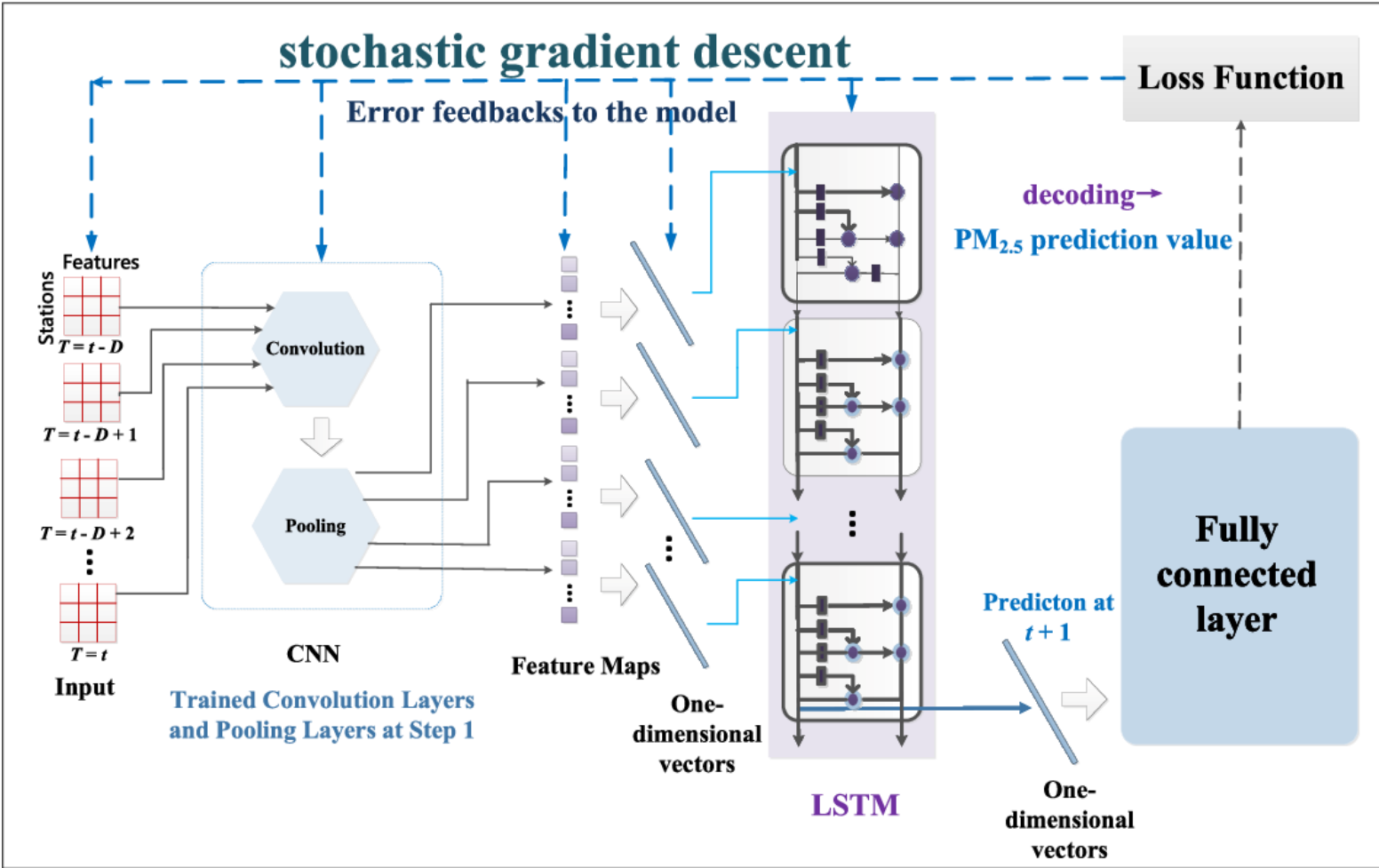
$$h_t = o_t * \tanh (C_t)$$

# LSTM 결과



# A Novel Combined Prediction Scheme Based on CNN and LSTM for Urban PM2.5 Concentration

<https://lib.hanyang.ac.kr/#/eds/detail?an=edsee.8632898&dbId=edsee>





1. 시계열 분석
2. RNN
3. LSTM
- 4. 언어모델**
5. Attention
6. Transformer
7. GPT

## 언어 모델

고려대 정순영 교수 강의안

<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/09/16/LM/>

## 정의

언어모델이란 단어 시퀀스에 대한 **확률분포(probability distribution)**를 가리킵니다. 언어모델은  $m$ 개 단어가 주어졌을 때  $m$ 개 단어 시퀀스가 나타날 확률, 즉  $P(w_1, w_2, \dots, w_m)$ 을 할당(assign)합니다. 예컨대 다음과 같습니다.

$$(1) P(\textit{Today is Wednesday}) = 0.001$$

$$(2) P(\textit{Today Wednesday is}) = 0.0000000001$$

## 확률통계 모형

<https://bab2min.tistory.com/666>

### Unigram 언어 모델

---

Unigram은 제일 단순한 언어모델입니다. 사실 언어모델이라고 하기에 민망한데요, 각각의 단어가 출현할 확률이 독립이라고 가정하고 모두 따로따로 계산하는 식입니다. 수식으로 나타내면 아래와 같습니다.

$$P_{uni}(w_1, w_2, w_3, \dots) = P(w_1)P(w_2)P(w_3) \dots$$

### n-gram 언어 모델

---

n-gram 언어 모델부터는 진짜 언어 모델이라고 부를 만합니다. Unigram과 달리 이전 단어를 고려하여 다음 단어의 확률을 계산하기 때문이지요. n-gram 모델에서는 다음 단어를 예측하기 위해 이전의  $n - 1$ 개의 단어를 활용합니다. Unigram 모델은 n-gram 모델에서  $n = 1$ 로 단순화시킨 모델인 셈이지요. n=2인 경우를 수식으로 나타내면 다음과 같습니다.

$$P_2(w_1, w_2, w_3, \dots) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots$$

## Neural Network 기반 언어 모델

**(1990) Jeffrey L. Elman,  
Finding Structure in Time**

[https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402\\_1](https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1)

**(2010) Mikolov et al,  
Recurrent neural network based language model**

[https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402\\_1](https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1)

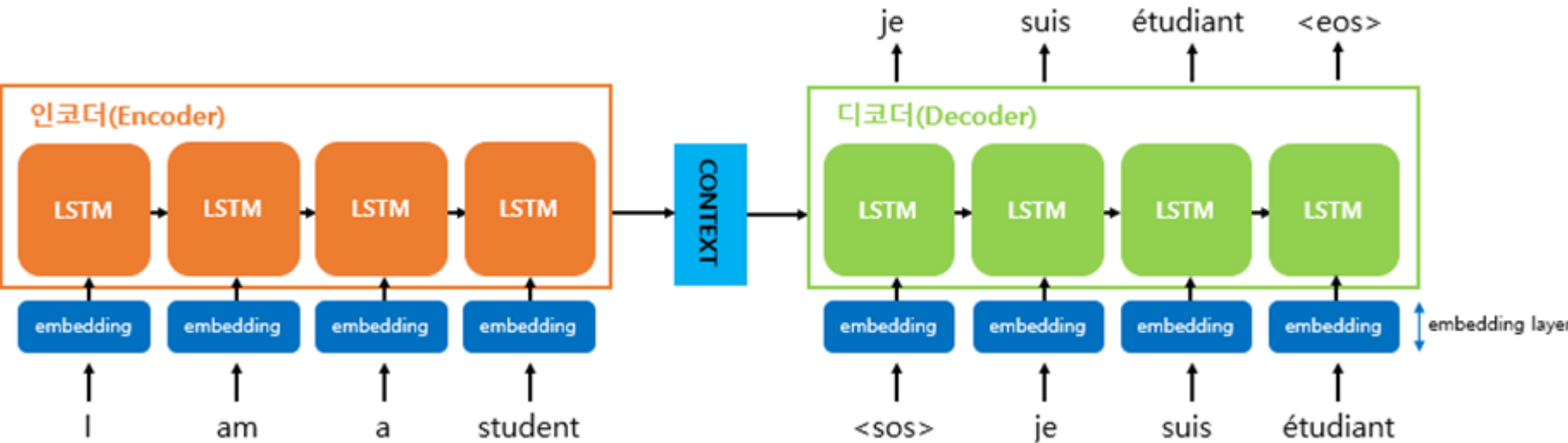
**(2014) Sutskever et al,  
Sequence to Sequence Learning with Neural Networks**

<https://arxiv.org/abs/1409.3215>

**(2014) Sutskever et al,  
Learning Phrase Representations using RNN Encoder–Decoder  
for Statistical Machine Translation**

<http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf>

# Seq2Seq 모델



1. 시계열 분석
2. RNN
3. LSTM
4. 언어모델
- 5. Attention**
6. Transformer
7. GPT

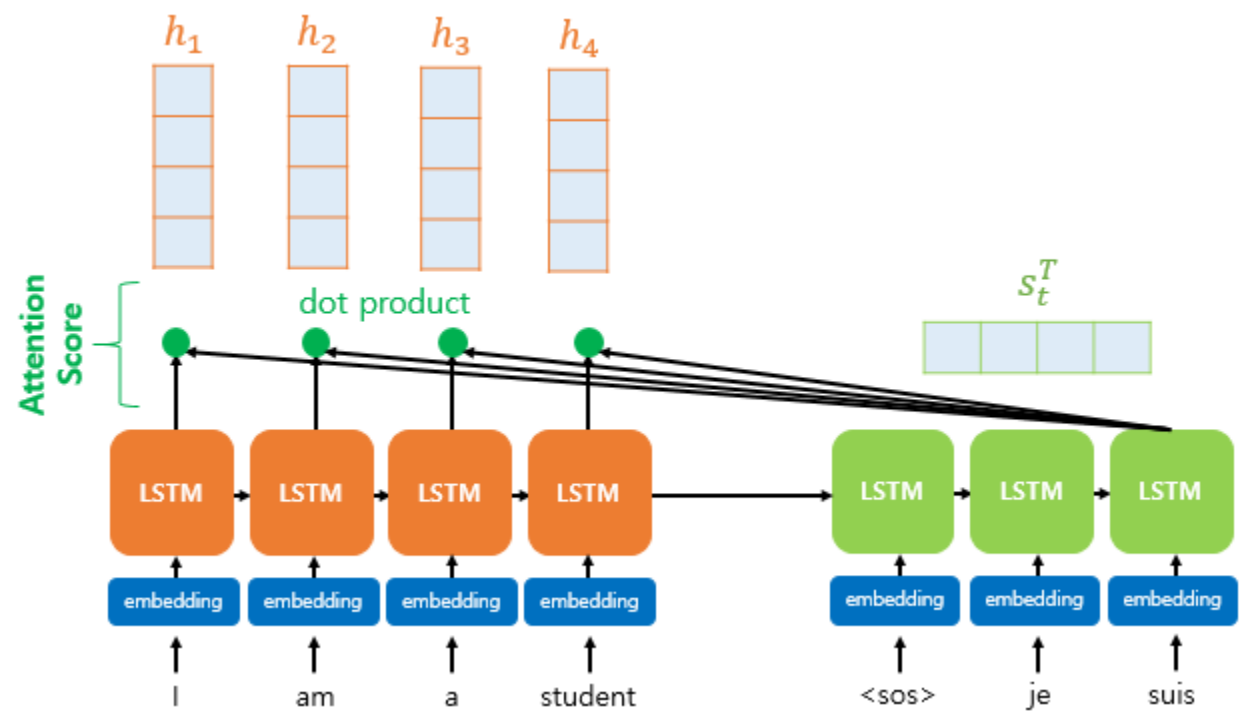
# Attention Mechanism 설명 참고 : <https://wikidocs.net/22893>



Luong et al., (2015) Effective Approaches to Attention-based Neural Machine Translation  
[https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402\\_1](https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1)

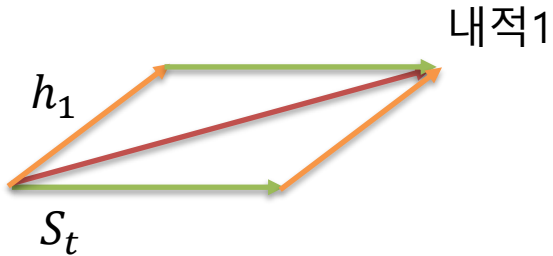
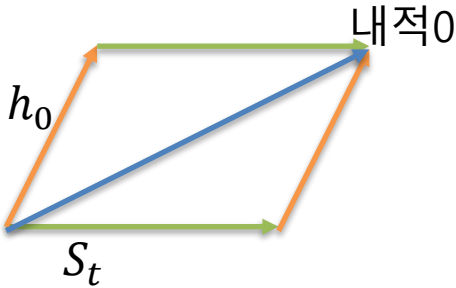
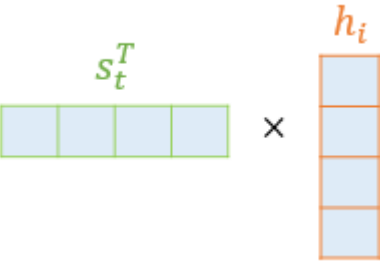
## Attention Score 계산

Decoder 마지막 은닉상태와 Encoder의 모든 입력단계의 은닉상태들과 내적을 계산한다.



# Attention Mechanism

## Attention Score 계산 의미

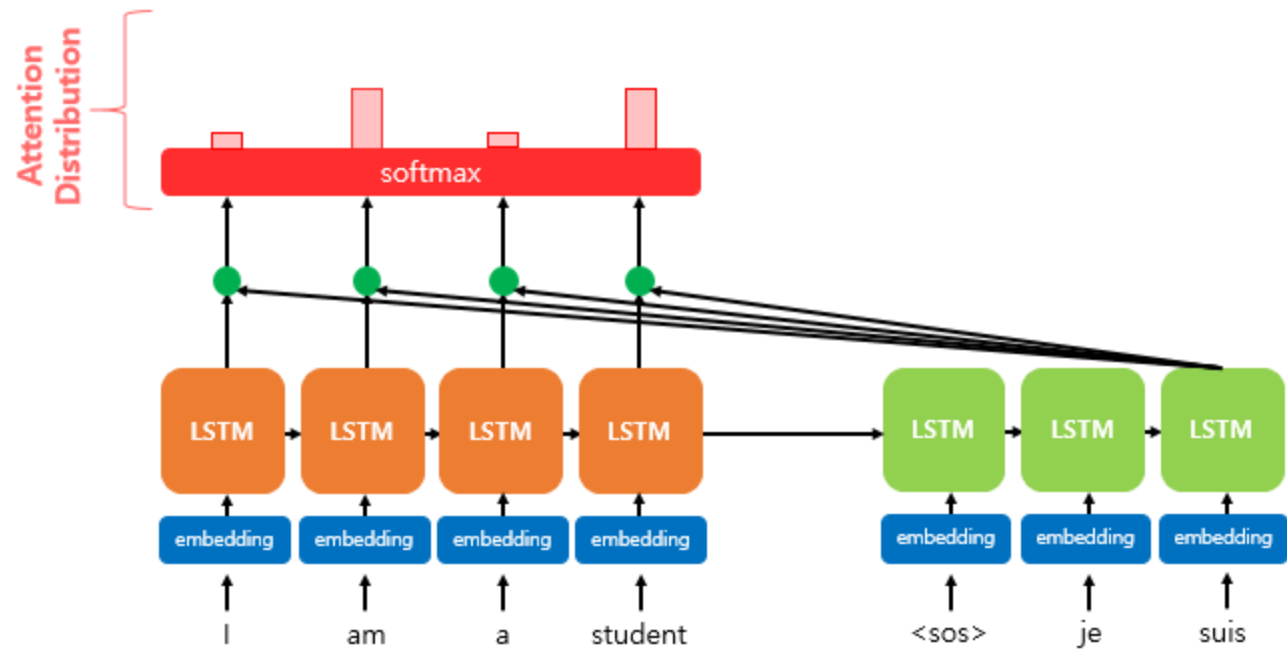




# Attention Mechanism

## Attention Distribution

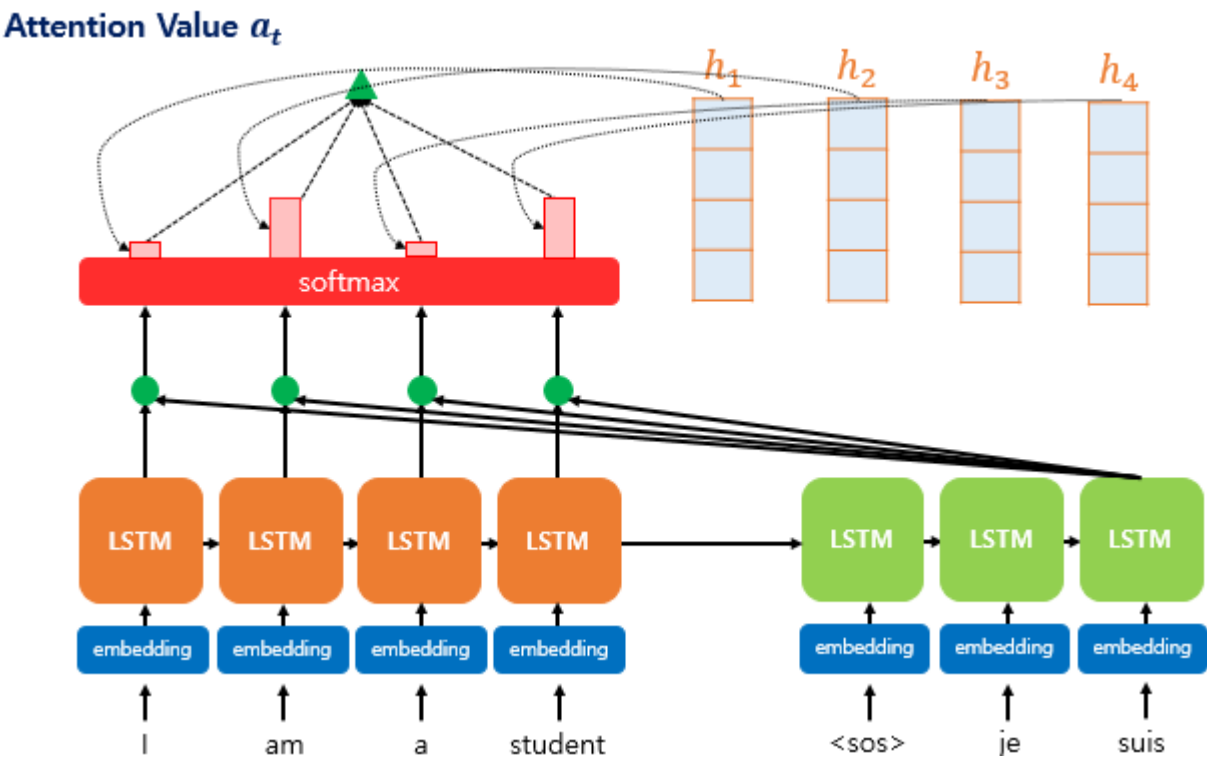
각각의 Encoder Input 은닉층에 대한 Attention Score를 기반으로 Softmax를 통해 합이 1이 되는 분포 비율 계산



# Attention Mechanism

## Context Vector

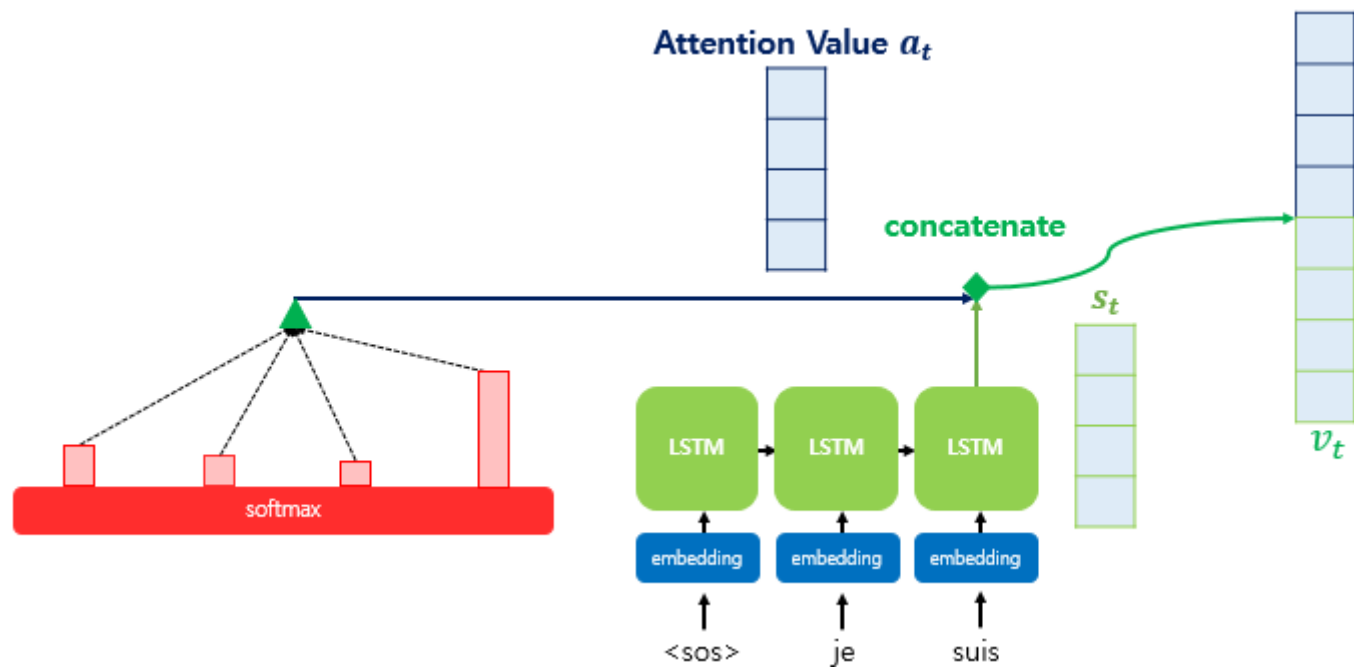
Input 은닉 상태를 Softmax 비율 만큼 가중합하여 최종 Attention Value를 계산  
해당 벡터에는 Input 데이터 중 현재 출력 상태와 가장 관계가 높은 입력 정보들의 은닉상태가 제공  
Attention 모델에서 이를 Context Vector로 표현



# Attention Mechanism

## Output token Predict

출력층 은닉상태와 attention score를 합쳐서 다음 출력 단어(토큰)을 추론

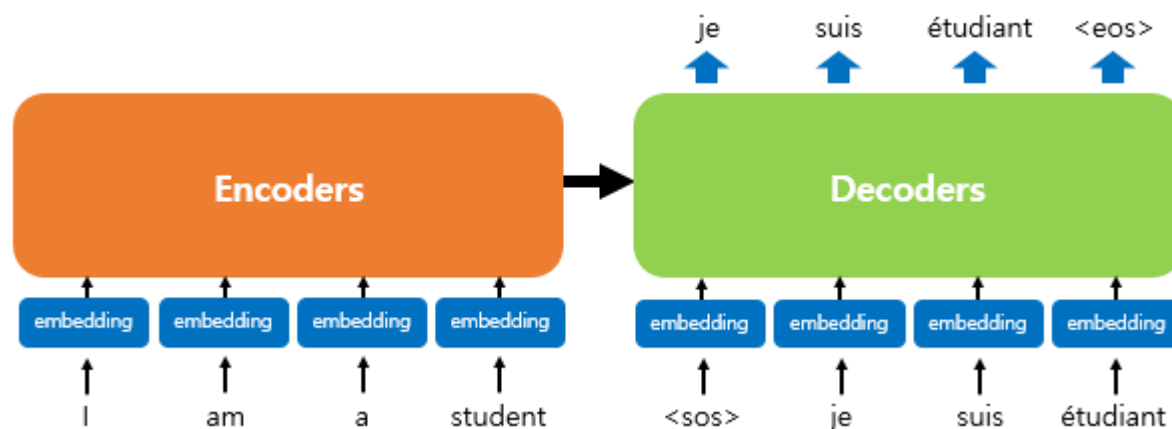
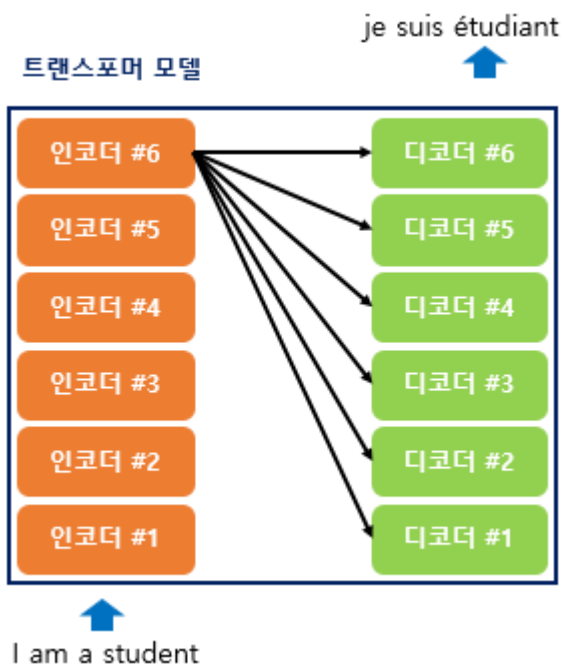


1. 시계열 분석
2. RNN
3. LSTM
4. 언어모델
5. Attention
- 6. Transformer**
7. GPT

# Attention is all you need (NIPS 2017) = Transformer

논문 : <https://arxiv.org/abs/1706.03762>

소스 : <https://github.com/ukairia777/tensorflow-transformer>

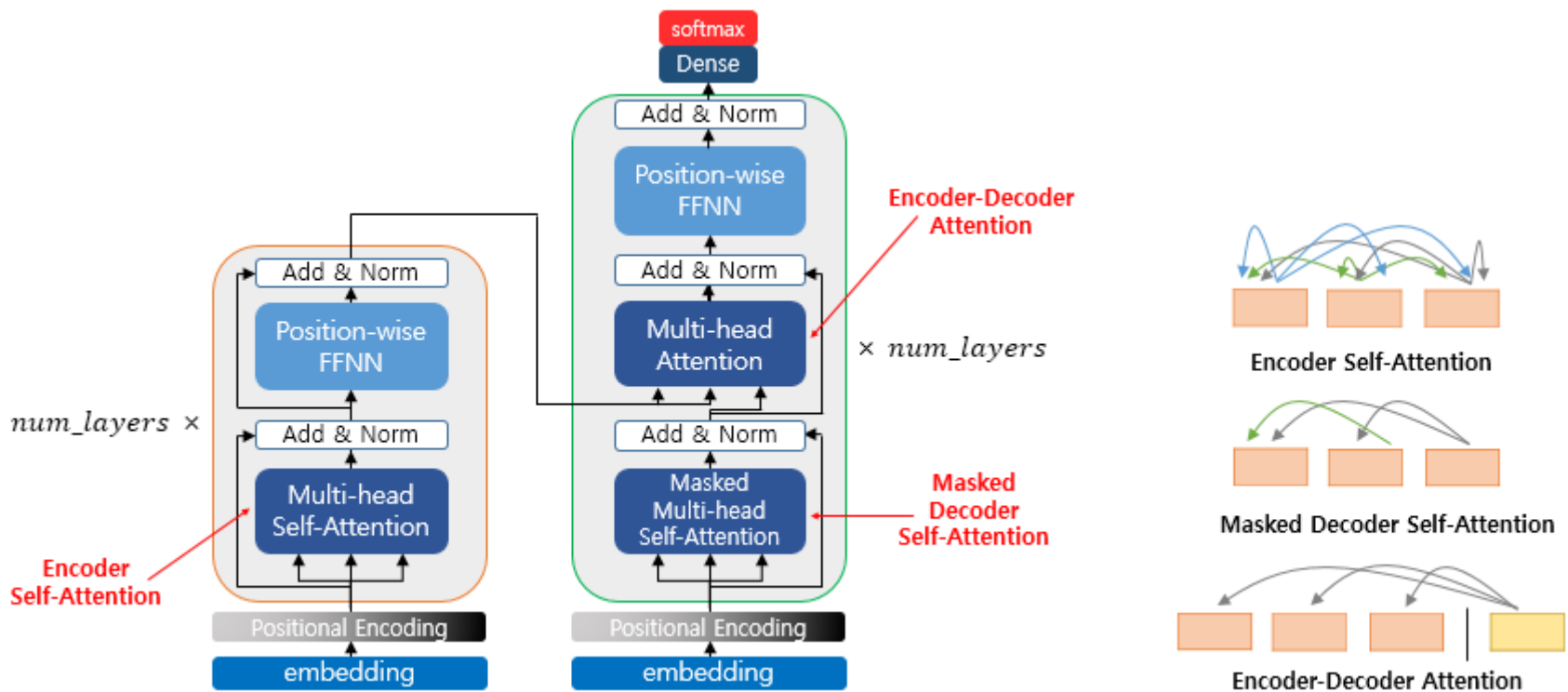




# Transformer Model

## Encoder Decoder Model 세부

인코더 모델과 디코더 모델이 각각 Attention 모델을 가지고 있으며 3가지 타입의 Attention 모델을 제시





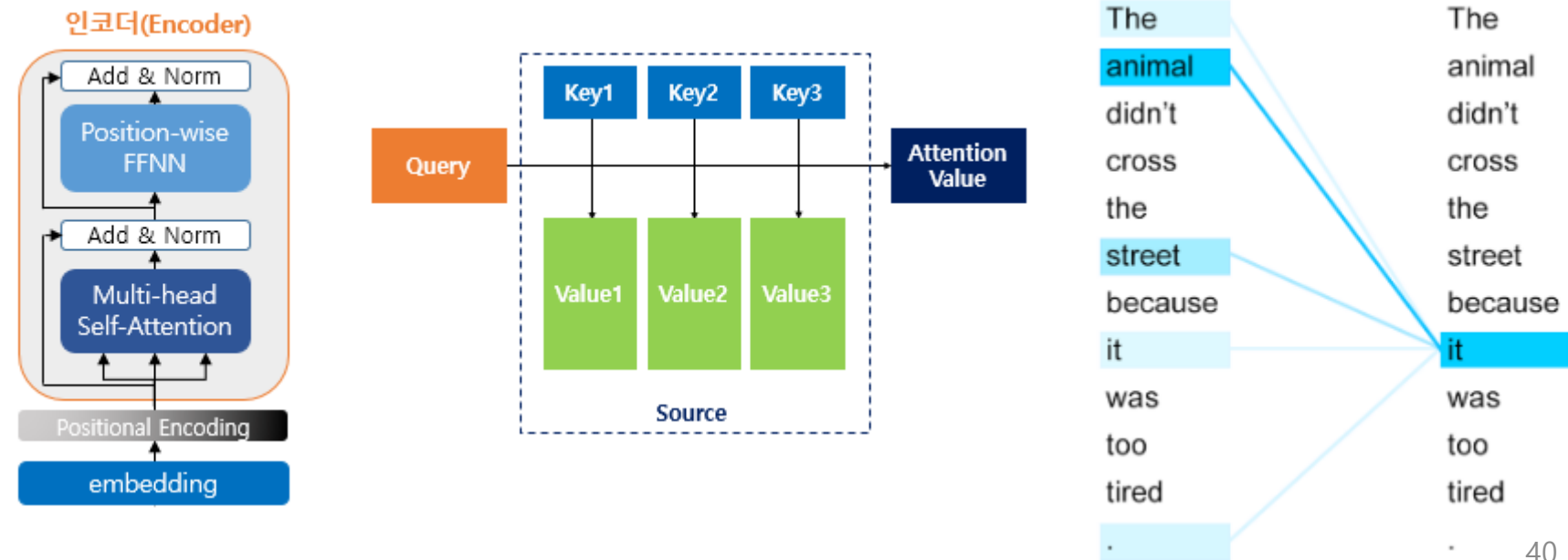
# Transformer Model

## Encoder : Multi-head Self-Attention

입력된 문장들 안에서 단어들 간의 Attention Value를 계산

Query : 타겟 단어 (인풋 문장의 단어별 벡터가 하나씩 들어간다.)

Key :





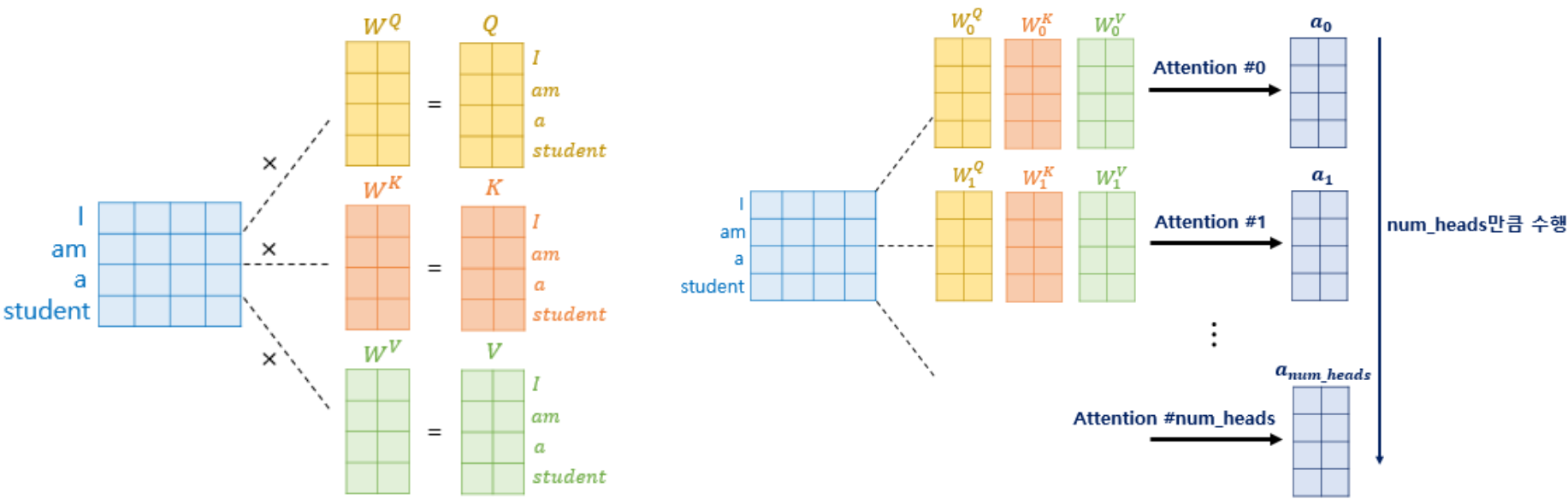
# Transformer Model

## Encoder : Multi-head Self-Attention

사용되는 은닉상태의 차원을 축소하여 여러 개의 어텐션 벨류를 추출하는 방식

$W_q, W_k, W_v$  의 3가지 가중치를 이용하여 은닉상태를 다양하게 변환하여 분석 (이는 학습 대상)

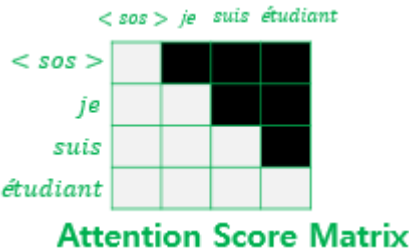
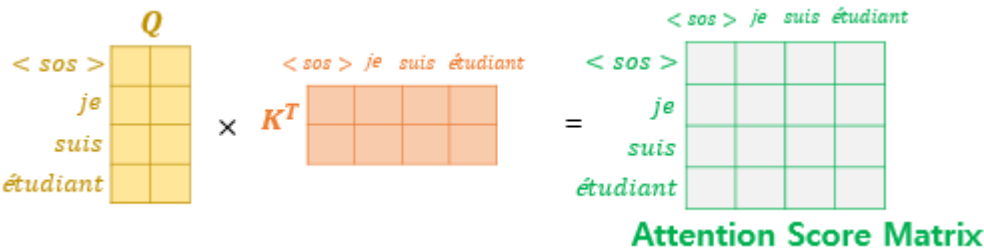
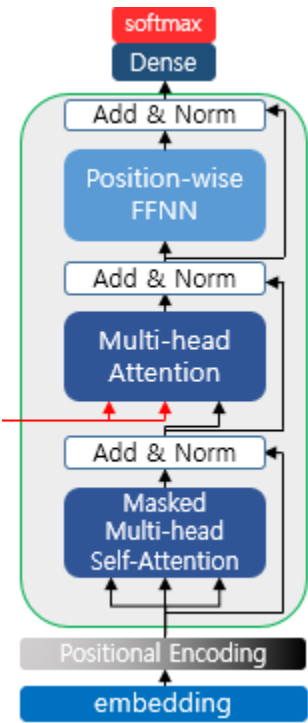
$Num\_heads$  수만큼 어텐션 벨류를 얻어내어 이들을 연결하여(concat) 다음 인코더 혹은 디코더에 전달



# Transformer Model

## Decoder : Masked Multi-head Self-Attention

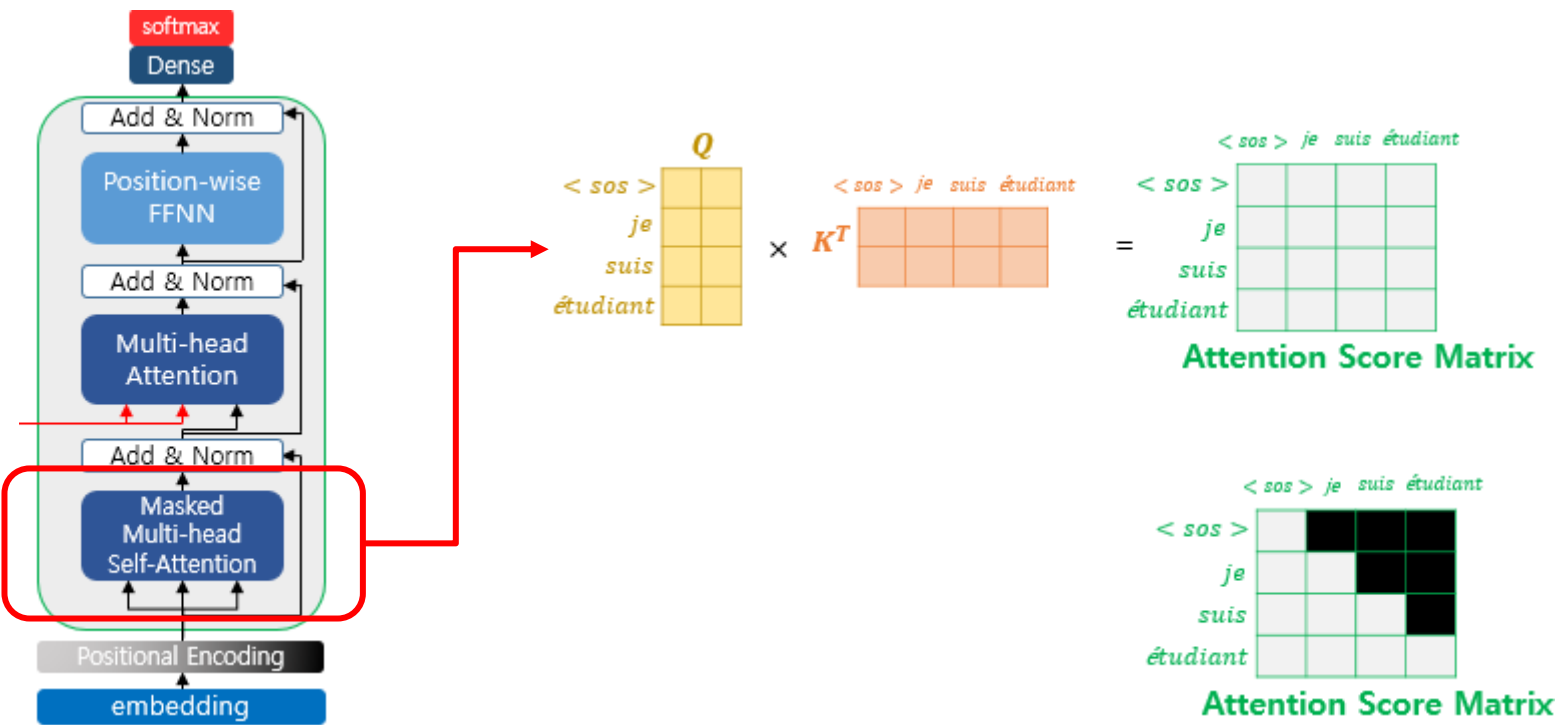
셀프 어텐션과 동일하지만 자신보다 이후에 나오는 단어에 대한 어텐션 스코어는 마스킹하여 활용하지 않음



# Transformer Model

## Decoder : Masked Multi-head Self-Attention

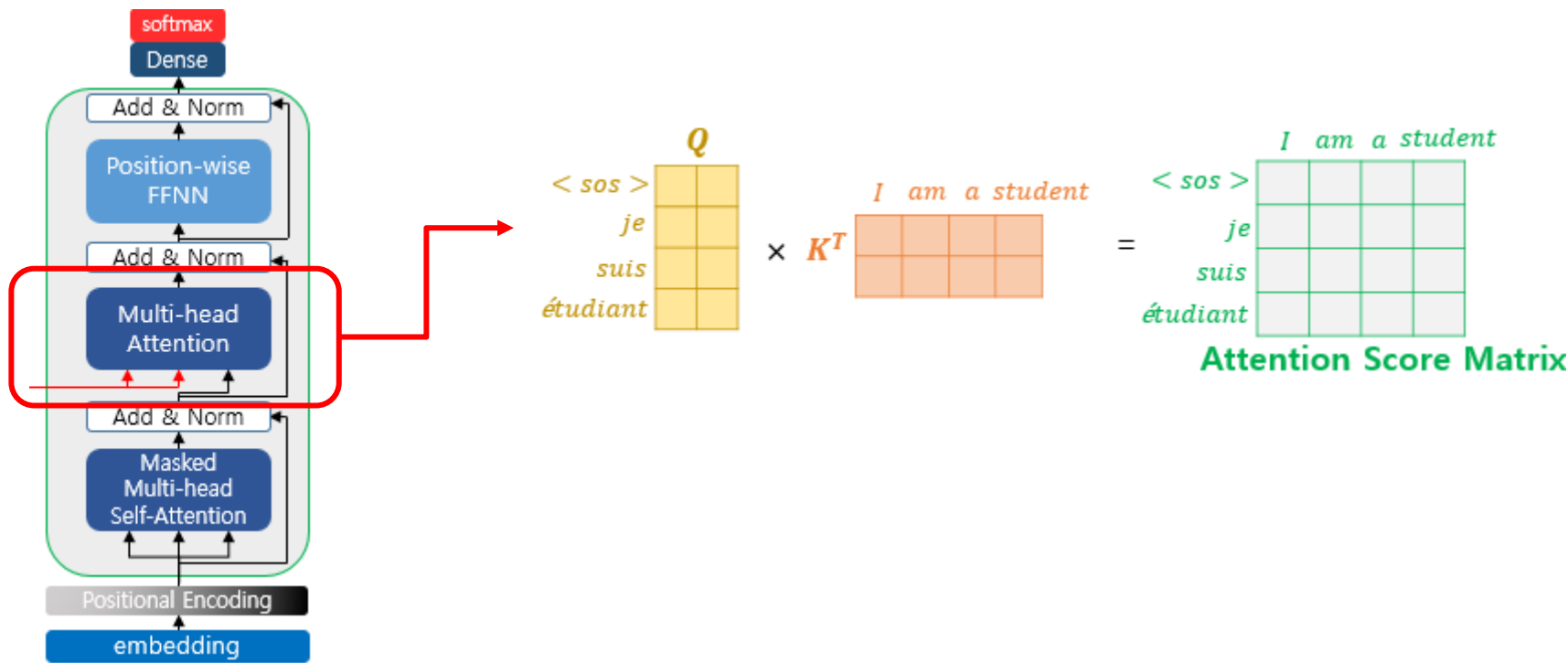
셀프 어텐션과 동일하지만 자신보다 이후에 나오는 단어에 대한 어텐션 스코어는 마스킹하여 활용하지 않음



# Transformer Model

## Decoder : Encoder-Decoder Attention

이전 Attention Mechanism과 같이 디코더 벡터를 Query로 인코더 벡터를 Key로 활용하여 Score를 계산



1. 시계열 분석
2. RNN
3. LSTM
4. 언어모델
5. Attention
6. Transformer
- 7. GPT**

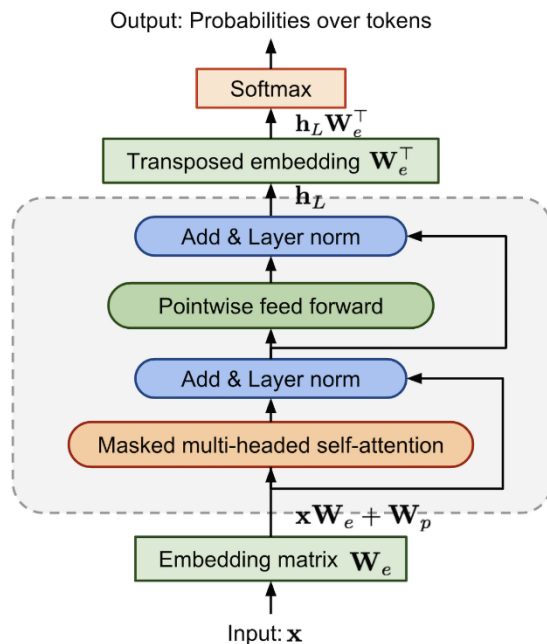
# GPT1 (Generative Pre-Training of a Language Model)

## Improving Language Understanding by Generative Pre-Training

<https://openai.com/research/language-unsupervised>

<https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf>

1. 대량의 학습 데이터 셋을 구하기 힘들
2. 비지도 학습과 지도 학습을 섞어서 학습하는 방법을 제시



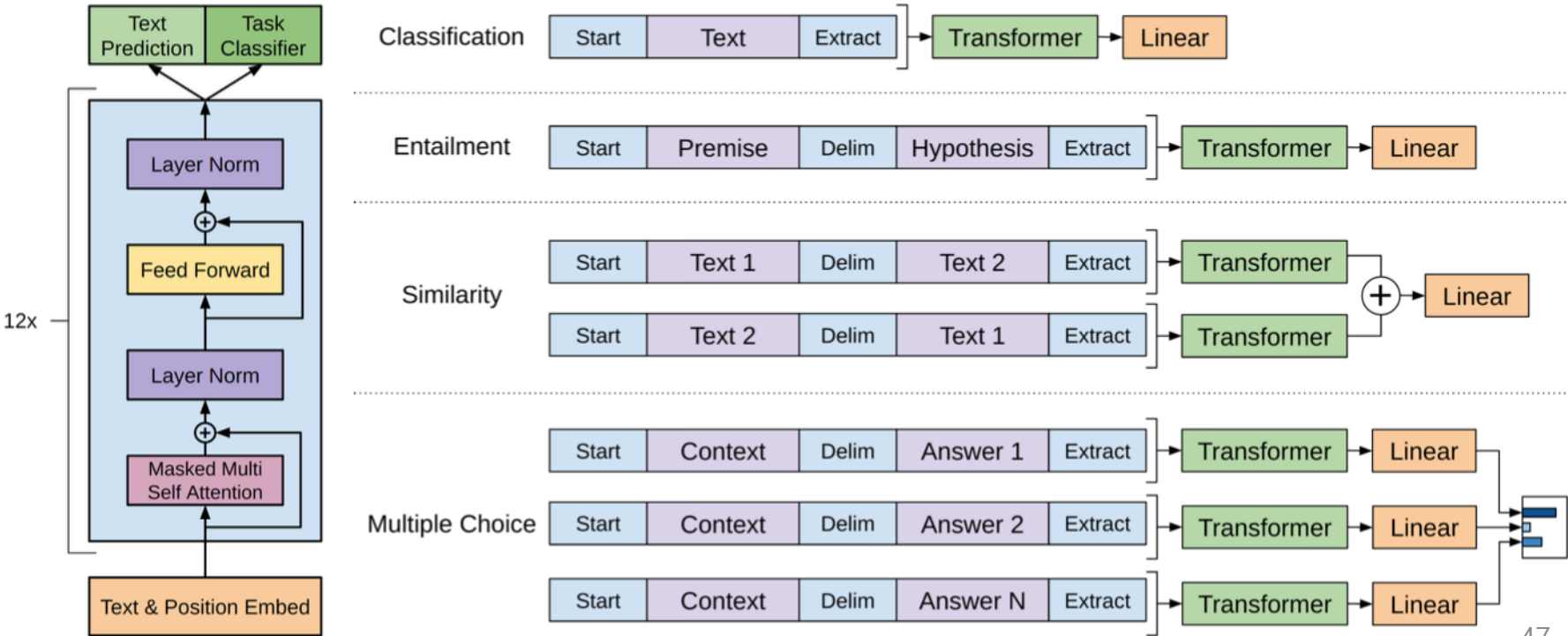
### 비지도 사전 학습 단계 unsupervised pretraining

- Transformer의 attention 중 Decoder에 있는 Masked Multi-headed Self-attention 만 활용함
- Decoder만 활용. GPT1 모델은 12개의 Decoder로 구성
- 비지도 학습에서는 일정 단어들의 순서를 전달하고 다음 단계에 출력될 단어의 확률을 Softmax 결과와 비교하여 학습함
- 비지도 학습을 통해 언어모델은 상당 부분 언어 관계를 이해

# GPT1 (Generative Pre-Training of a Language Model)

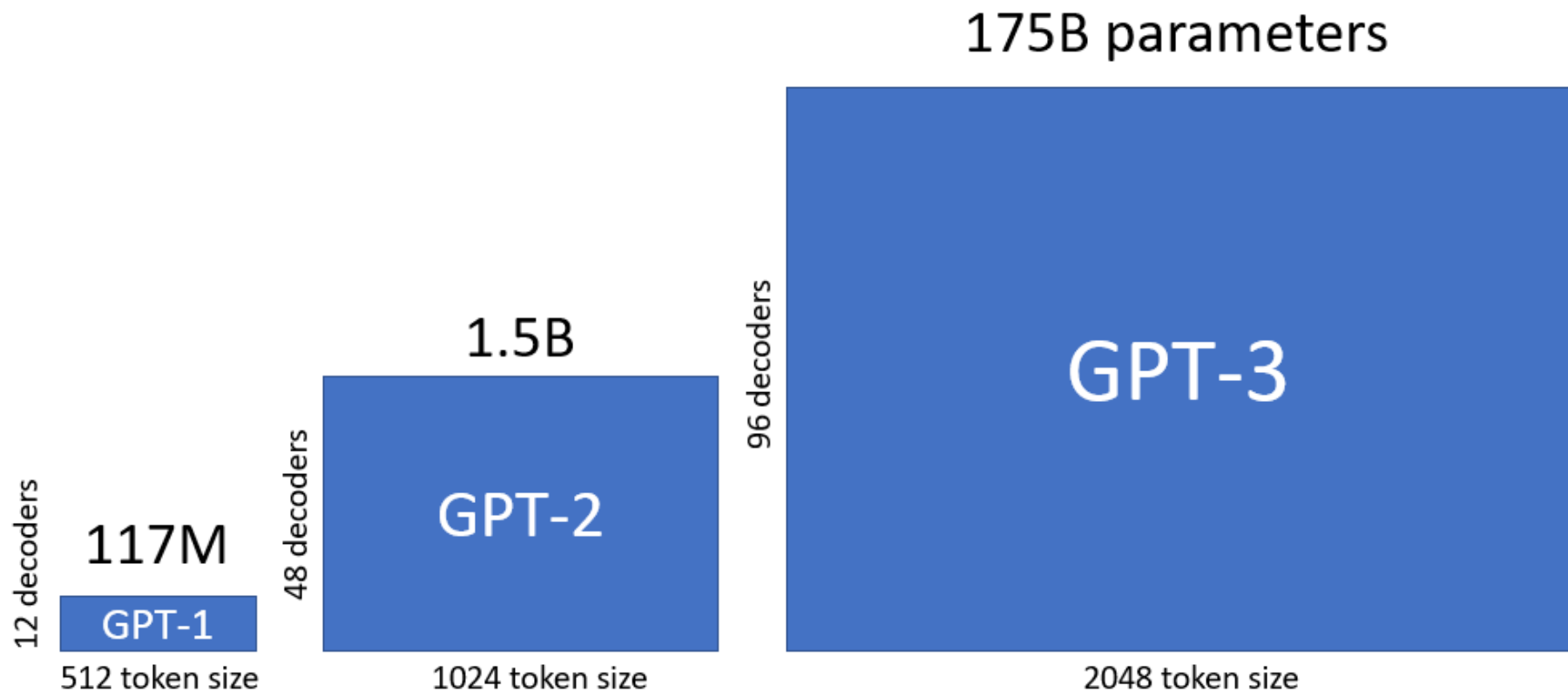
## 지도 전이 학습 단계 Supervised fine-tuning

- 비지도 사전학습에서 얻은 디코더는 학습을 시키지 않는다.
- 디코더 층을 통과하고 나온 softmax 결과에 linear 레이어를 연결하여 전이학습
- 테스트에 따라 다양한 형태로 지도학습을 진행하여 수행



## GPT1 > GPT2 > GPT3

- GPT는 버전이 올라가면서 학습에 활용하는 텍스트 데이터 분야를 늘려 나가면서 디코더 개수, 파라미터 수, 토큰사이즈를 계속 키워 나감
- GPT2에서는 데이터를 늘려서 이전보다 특정 분야에 편중되지 않은 언어모델(general language model)을 만들어내는 목표 진행, 그러나 zero-shot, few-shot 대한 성능이 저조했음
- GPT3는 fine-tuning을 완전히 없애고 성능을 대폭 높임





# GPT3, GPT3.5, chatGPT

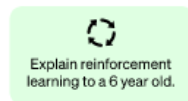
<https://velog.io/@easter423/GPT-3-vs-GPT-3.5-vs-ChatGPT>

- GPT3(davinci)는 base 모델로 최초 학습된 1750억개의 파라미터를 제공, 속도가 느리고 고비용
- GPT3.5(davinci-003, instructGPT)는 RLHF(Rainforce Learning from Human Feedback) 기법을 도입하여 사람의 피드백을 통해 인간이 원하지 않는 응답(폭력,욕설,혐오 등)을 제거함.
- chatGPT는 GPT3.5와 마찬가지로 RLHF를 사용하였으나 최초의 강화학습 부문을 supervisor가 제시하여 이전버전들에 비해서 대화에서 원하지 않는 응답 확률을 낮추었다.

## Step 1

**Collect demonstration data and train a supervised policy.**

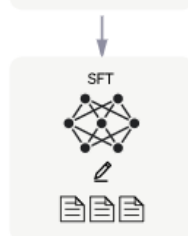
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



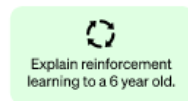
This data is used to fine-tune GPT-3.5 with supervised learning.



## Step 2

**Collect comparison data and train a reward model.**

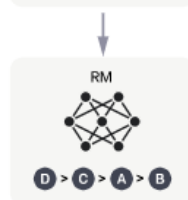
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



## Step 3

**Optimize a policy against the reward model using the PPO reinforcement learning algorithm.**

A new prompt is sampled from the dataset.



The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



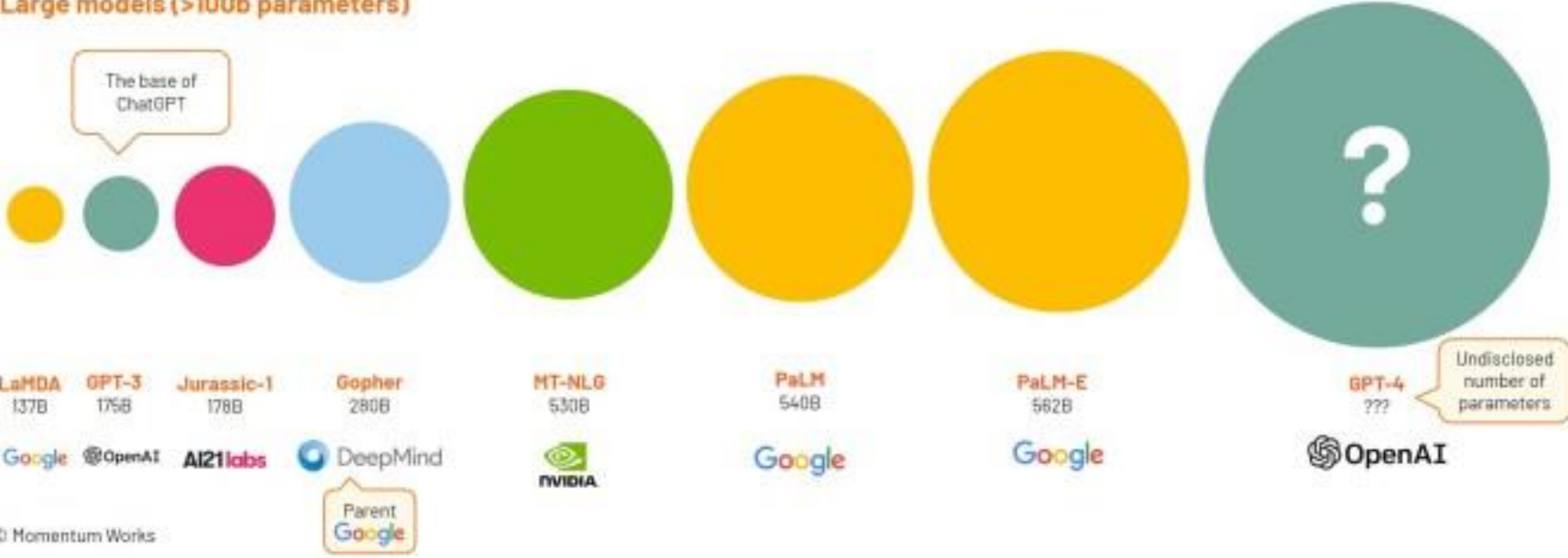
# LLM(Large Language Model) – 초거대AI

## Large Language Models are becoming very large indeed

### Small models (<= 100b parameters)



### Large models (>100b parameters)



## Chat GPT 온도에 따른 Random 선택

**Q : 한국건설기술연구원에 대해서 한문장으로 설명해줘 temp = 1.0**

- 한국건설기술연구원(KICT)은 건설 산업 발전을 위한 기술 연구 및 개발, 기술 지원, 국제협력 등을 수행하는 정부 출연 연구기관입니다.
- 한국건설기술연구원은 건설 산업의 기술 발전 및 혁신을 촉진하는 연구 개발 기관입니다.
- 한국건설기술연구원은 건설 기술의 연구, 개발 및 지식 보급을 통해 건설 산업 발전에 기여하는 국가 연구소입니다.
- 한국건설기술연구원(KICT)은 국내 건설 분야의 기술 발전과 산업 선도를 위한 기술 연구 및 개발을 수행하는 기관입니다.
- 한국건설기술연구원은 건설 분야 기술 연구 및 개발, 인허가 및 검사, 기술 정보 제공 등 건설 산업의 기술적 발전과 지속 가능한 발전을 추진하는 기관입니다.

**Q : 한국건설기술연구원에 대해서 한문장으로 설명해줘 temp = 0.0**

- 한국건설기술연구원은 건설 분야의 기술 연구 및 개발을 수행하는 국가 연구기관입니다.
- 한국건설기술연구원은 건설 분야의 기술 연구 및 개발을 수행하는 국가 연구기관입니다.
- 한국건설기술연구원은 건설 분야의 기술 연구 및 개발을 수행하는 국가 연구기관입니다.
- 한국건설기술연구원은 건설 분야의 기술 연구 및 개발을 수행하는 국가 연구기관입니다.
- 한국건설기술연구원은 건설 분야의 기술 연구 및 개발을 수행하는 국가 연구기관입니다.

## ChatML (Markup Language)

<https://cobusgreyling.medium.com/the-introduction-of-chat-markup-language-chatml-is-important-for-a-number-of-reasons-5061f6fe2a85>

- ChatGPT에 채팅 내용을 전달하기 위해 만들어진 구조체
- role과 content 키값으로 이루어진 딕셔너리 리스트를 전달
- Role은 system, assistant, user 로 구분하며, system을 이용하여 챗봇을 내가 원하는 방식으로 설정 가능

```
{"role": "system", "content": "너는 초등학교 1학년이고, 반말로 대답한다."}
```

```
{"role": "user", "content": "안녕 넌 누구니?"}
```

```
{"role": "assistant", "content": "나는 AI 챗봇이야. 네 명령을 받아들이고 대답할 수 있어."}
```

```
{"role": "system", "content": "정중한 말투로 답변한다. "}
```

```
{"role": "user", "content": "안녕 넌 누구니?"}
```

```
{"role": "assistant", "content": "안녕하세요, 저는 OpenAI의 인공지능 어시스턴트입니다. 무엇을 도와드릴까요?"}
```

```
{"role": "system", "content": "친근한 말투로 답변해줘 "}
```


```
{"role": "user", "content": "안녕 넌 누구니?"}
```

```
{"role": "assistant", "content": "안녕하세요! 저는 당신이 지금 이 채팅방에서 만난 인공지능 챗봇입니다. 무엇을 도와드릴까요? :)}"}
```


# ChatGPT Plugin

<https://openai.com/blog/chatgpt-plugins?fbclid=IwAR003-NBN9Waq9qqAoqyWo6ILUXxIWG9q0g-CbcNZABsaSIhVXKqFz7kYyU>


- 최신 데이터 연계, 개인화 등 chatGPT의 한계를 플러그인으로 대체 가능




**Expedia**  
Bring your trip plans to life—get there, stay there, find things to see and do.




**FiscalNote**  
Provides and enables access to select market-leading, real-time data sets for legal, political, and regulatory data and information.




**KAYAK**  
Search for flights, stays and rental cars. Get recommendations for all the places you can go within your budget.




**Klarna Shopping**  
Search and compare prices from thousands of online shops.




**OpenTable**  
Search for restaurants available for booking dining experiences.



**Shop**  
Search for millions of products from the world's greatest brands.




**Wolfram**  
Access computation, math, curated knowledge & real-time data through Wolfram|Alpha and Wolfram Language.



**Zapier**  
Interact with over 5,000+ apps like Google Sheets, Trello, Gmail, HubSpot, Salesforce, and more.

J

plot function 1/sin(x)



Finished calculating

Show work

▼

👍

👎

Graph of 1/sin(x)



Here is the graph of the function  $1/\sin(x)$ . Please note that the graph has vertical asymptotes where  $\sin(x)$  is zero. For clarity, the y-axis range has been limited to -10 and 10.

# AGI (Artificial General Intelligence)

OpenAI CEO 샘 올트만 AGI에 대하여 (openAI 사이트)

<https://openai.com/blog/planning-for-agi-and-beyond>

"첫째, 더욱 강력한 시스템을 지속적으로 개발하면서 이를 배포하고 실제 환경에서 운영 경험을 쌓는 것입니다. 이것이 AGI를 신중하게 도입할 수 있는 가장 좋은 방법이라고 생각합니다. AGI가 있는 세상으로의 점진적인 전환이 갑작스러운 전환보다 낫기 때문입니다. 강력한 AI가 세상의 발전 속도를 훨씬 더 빠르게 만들 것으로 기대하며, 점진적으로 적응하는 것이 더 낫다고 생각합니다."

DeepMind CEO 데미스 하사비스 (GTC 2023 강연 슬라이드)

## 강연스크립트

기술 업계가 오랫동안 지켜온 일종의 진리인 '빨리 움직여서 무언가를 깨뜨리지 말아야 한다'는 조언을 하고 싶습니다. 물론 이는 엄청난 성공을 거두었고 많은 발전을 가져왔습니다. 반드시 느리게 움직여야 한다는 것이 아니라 빠르게 움직여야 하지만 우리가 하는 일에 대해 신중하게 생각해야 한다는 것입니다

Live A B test not working out well, it's society I don't think is really a viable approach going forward with something like AI. And I think we're already starting to see the consequences of that.



OpenAI CEO 샘 올트만 인터뷰

[https://www.youtube.com/watch?v=L\\_Guz73e6fw](https://www.youtube.com/watch?v=L_Guz73e6fw)

샘: 제가 말하고 싶은 것은 이게 소수의 사람들이 하고 있는 것이 아니라는 겁니다. 이것은 놀라운 양의 인간 노력이 결합된 결과라고 할 수 있죠. 이렇게 될 수 있도록 모든 것이 어떻게 뭉쳐야 했는지 생각해 보세요. 40년대에 트랜지스터를 발견한 사람들이 이렇게 되길 바라고 있었을까요? 그 첫 번째 트랜지스터부터 칩에 수많은 숫자를 집어넣고, 이 모든 것을 함께 연결하는 방법을 찾아내고, 이 모든 것에 들어가는 에너지와 과학, 그 모든 과정을 거친 수십만, 수백만의 사람들의 노력입니다. 이것은 우리 모두의 결과물이예요. 그리고 그게 꽤 멋진 것 같아요.

샘: 그리고 그 모든 것이 이 하나의 지수 곡선 위에 있었죠.

