

KUBIG 25-W
겨울방학 BASIC STUDY SESSION

NLP SESSION

WEEK6

KUBIG
DATA SCIENCE & AI

01 Announcement, 복습과제 우수 코드 review

02 Hugging Face : LLM을 위한 필수 툴

03 Prompt Engineering

04 Parameter Efficient Fine-Tuning: LLM 경량화와 최적화

05 복습과제 안내

01 Announcement,
우수 복습과제 선정

WEEK7 (2/20 목요일) 세션 없음

- 과제 우수자는 슬랙으로 공지합니다.
- 콘테스트 마무리에 힘 써주세요!
- 질문은 슬랙 DM으로 주시면 됩니다 :-)

20기 이우진님

20기 김정찬님

WEEK5
복습과제1

WEEK5
복습과제2

BERT Fine tuning

언어모델에서의
자연어 생성 전략

02 Hugging Face

얼만큼 활용하고 계신가요!

LLM 활용을 위한 필수 툴 소개



Hugging Face

오픈소스 AI 모델, 데이터셋, 라이브러리를 공유하는 플랫폼

이 분야에서는 사실상 최대 규모의 오픈소스 플랫폼!

<https://huggingface.co/>



Hugging Face

- **Model Hub:** 사전학습된 모델 검색 & 활용
- **Dataset Hub:** AI 모델 훈련을 위한 데이터셋 공유
- 😊**transformers**, 😊**datasets** 같은 핵심 라이브러리 제공

Hugging Face Models

Models 1,400,527

- deepseek-ai/DeepSeek-R1
- deepseek-ai/Janus-Pro-7B
- ValueFX9507/Tifa-Deepsex-14b-CoT-GGUF-Q4
- deepseek-ai/DeepSeek-V3
- hexgrad/Kokoro-82M
- mistralai/Mistral-Small-24B-Instruct-2501
- unsloth/DeepSeek-R1-GGUF

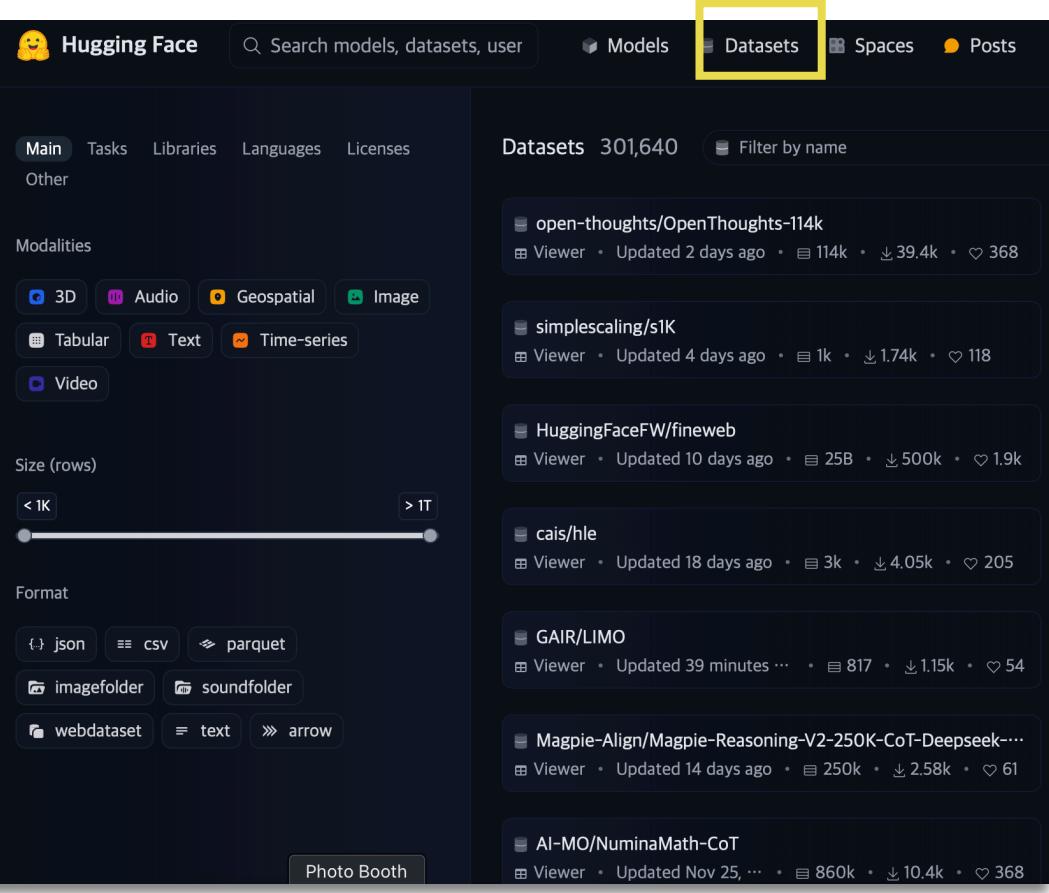
Model Hub

사전학습된 모델 검색 & 활용

- LLM을 가장 쉽게 활용할 수 있는 공간
- BERT, GPT, T5 등 다양한 모델 제공
- 원하는 모델을 검색하고 바로 로드 가능



Hugging Face Datasets



The screenshot shows the Hugging Face Dataset Hub interface. At the top, there is a navigation bar with icons for Hugging Face, search, Models, Datasets (which is highlighted with a yellow box), Spaces, and Posts. Below the navigation bar, there are several filters on the left: Main (selected), Tasks, Libraries, Languages, Licenses, Other, Modalities (3D, Audio, Geospatial, Image, Tabular, Text, Time-series, Video), Size (rows) (less than 1K to greater than 1T), Format (json, csv, parquet, imagefolder, soundfolder, webdataset, text, arrow), and a Photo Booth button at the bottom.

Datasets 301,640 Filter by name

- open-thoughts/OpenThoughts-114k
- simplescaling/s1K
- HuggingFaceFW/fineweb
- cais/hle
- GAIR/LIMO
- Magpie-Align/Magpie-Reasoning-V2-250K-CoT-Deepseek----
- AI-MO/NuminaMath-CoT

Dataset Hub

모델 훈련을 위한 데이터셋 공유

KUBIG 25-W NLP Basic Study WEEK6(Today's Session Leader: 19th 심승현)

10

2-1. Hugging Face library

The screenshot shows the Hugging Face Model Hub interface. At the top, there's a banner with a smiley face icon and the word "transformers". Below it, the "openai-community/gpt2" model card is displayed. The "Use this model" button is highlighted with a red box. A dropdown menu labeled "Libraries" also has "Transformers" highlighted with a red box. Below the card, a section titled "How to use from the • Transformers library" contains two code snippets:

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-generation", model="openai-community/gpt2")

# Load model directly
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("openai-community/gpt2")
model = AutoModelForCausalLM.from_pretrained("openai-community/gpt2")
```

자연어 처리 모델을 쉽게 사용할 수 있는 핵심 라이브러리

- 가장 많이 활용하는 라이브러리
- 사전 학습된 모델 로드 & 활용 → 추론까지!
- 토큰화(Tokenization) 자동 지원
- 한 줄의 코드로 모델 실행 가능
- PyTorch 호환성!

<https://huggingface.co/docs/transformers/en/index>



놓치지 마세요!

A screenshot of the Hugging Face transformers documentation homepage. The page has a dark header with the "transformers" logo. Below the header, there's a "Quick Links" section with three items: "Read model documentation", "Read docs on high-level-pipeline", and "Read our learning resources". Each item has a small icon next to it. At the bottom of the page, there's a footer with the text "They provided and give specific examples" and a link "View examples".

Documentation에는..

- 하나의 모델 안에서 파생되는 수많은 파인튜닝 모델
- 어떻게 활용되는지, 활용하는 코드는 어떻게 되는지

<https://huggingface.co/docs/transformers/en/index>

datasets

- 허깅페이스에 등록된 공개 데이터셋 불러오기
- 전처리

peft

Parameter-Efficient Fine-Tuning

- LLM을 효율적으로 파인 투닝할 수 있도록 돋는 라이브러리
- LoRA 등의 기법을 활용해서 적은 자원으로 LLM을 학습할 수 있도록 하는 기능 제공

trl

Transformers Reinforcement Learning

- 강화 학습을 기반으로 LLM을 훈련하거나 최적화할 수 있도록 지원하는 라이브러리
- Supervised Fine Tuning 시에도 **trl** 라이브러리 주로 활용

bitsandbytes

- 모델을 적은 메모리(4비트/ 8비트)로 LLM을 사용할 수 있도록 하는 라이브러리
- 양자화를 돋는 역할

<https://huggingface.co/docs/transformers/en/index>

허깅페이스에 들어가 봅시다!

등록되는 오픈소스 LLM : base or instruct

Base 모델 : 단순히 다음 토큰 예측이라는 목표로 사전 학습만을 거친 모델

Instruct 모델 : 특정한 목적의 태스크를 수행하도록 별도의 파인 튜닝을 거친 모델 (주로 chat)



입력이 질문인지 아닌지에는 관심이 없고 그럴 듯한 문장을 이어서 생성



명시적으로 질문에 대한 응답을 생성하도록 추가적으로 훈련

03 Prompt Engineering

From Chat Templates to Prompt Engineering



LLM 왜 사용함?

질문에 대한 대답을 얻거나, 어떤 작업을 수행하도록 지시



그럼 바로 Instruct 모델을 쓰면?

Instruct 모델도 제대로 활용하려면 올바른 프롬프트 입력해야 됨

왜 Chat Template이 필요할까?

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

device = "cuda" if torch.cuda.is_available() else "cpu"
# base_model = "meta-llama/Llama-3.2-1B"
base_model = "meta-llama/Llama-3.2-1B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(base_model)
model = AutoModelForCausalLM.from_pretrained(base_model).to(device)

raw_input = "What is Large Language Model?"
encoded_input = tokenizer(raw_input, return_tensors="pt").to(device)

outputs = model.generate(**encoded_input, max_new_tokens=20)
print(tokenizer.decode(outputs[0]))
```

<|begin_of_text|>What is Large Language Model? (with examples and applications)
A large language model is a type of artificial intelligence (AI) model



답변

- 단순히 Instruct 모델을 사용해도 LLM이 제대로 응답하는 것 X
- 질문 앞에 (with examples and applications)라는 문장을 마음대로 추가하고, 그 다음에 그에 대한 대답을 생성



LLM이 실제로 입력받는 프롬프트 구조

```
{system} 당신은 유용한 AI 어시스턴트입니다. {/system}  
{instruction} 질문에는 항상 한국어로 대답하세요. {/instruction}  
{user} Large Language Model이 무엇인가요? {/user}  
{assistant}
```

시스템 프롬프트 (약식)

- LLM에 입력되는 프롬프트는 단순한 질의(query)가 아니라 **시스템 프롬프트 + 사용자 입력**의 조합
- Chat Template을 사용하면 모델이 기대하는 입력 형식을 맞출 수 있음

3-1. Chat Template

Llama의 Chat Template

Special Tokens ..

- <|begin_of_text|>: 프롬프트 시작
- <|eot_id|>: 특정 발화(턴)의 종료
- <|start_header_id|>user<|end_header_id|>: 사용자의 입력

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
```

Cutting Knowledge Date: December 2023
Today Date: 23 July 2024

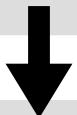
```
You are a helpful assistant<|eot_id|><|start_header_id|>user<|end_header_id|>
```

```
What is the capital of France?<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

- LLM은 스페셜 토큰(Special Token)을 사용해 문맥을 이해함
- **system, user, assistant**와 같은 구조가 있어야 정상적으로 답변

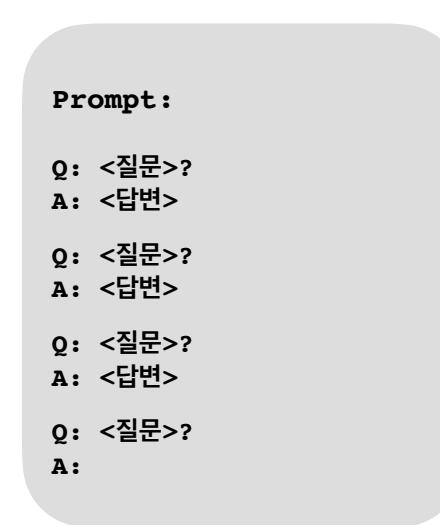
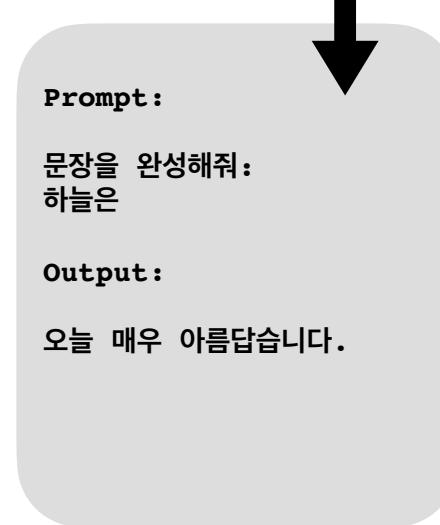
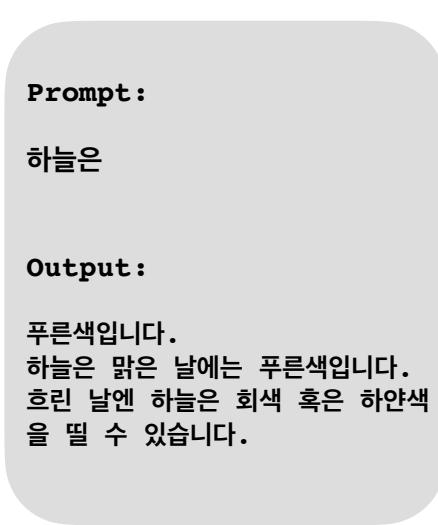
하지만! 😊 Transformers를 활용한다면 ... 😊

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>  
  
Cutting Knowledge Date: December 2023  
Today Date: 01 Jan 2025  
  
<|eot_id|><|start_header_id|>user<|end_header_id|>  
  
What is Large Language Model?<|eot_id|><|start_header_id|>assistant<|end_header_id|>  
  
A Large Language Model (LLM) is a type of artificial intelligence (AI) model that is designed to process and  
understand human language. It is a type of neural network that uses deep learning techniques to analyze and  
generate text.  
  
A Large Language Model
```

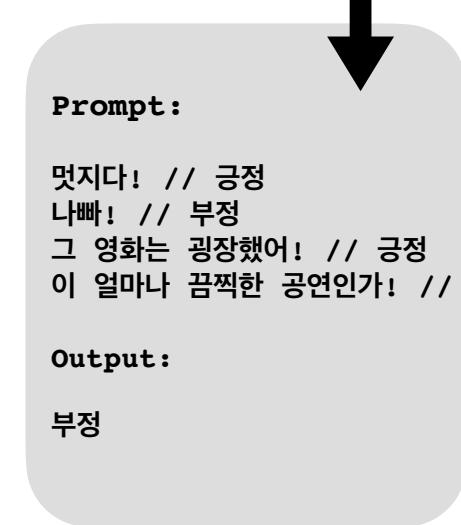


```
messages = [{"role": "user", "content": "What is Large Language Model?"}]  
  
encoded_prompt = tokenizer.apply_chat_template(messages,  
                                                add_generation_prompt=True,  
                                                return_tensors="pt").to(device)  
  
outputs = model.generate(encoded_prompt, max_new_tokens=50)  
print(tokenizer.decode(outputs[0]))
```

Zero-Shot Prompting



3-Shot Prompting



LLM은 **입력된 텍스트**에 따라 다르게 반응하고,
어떻게 질문하느냐에 따라 결과의 품질이 달라짐

효과적인 프롬프트를 작성하는 기술이 바로 **Prompt Engineering**

Few-shot Prompting (Few-shot Learning)

1-shot



Prompt:

"whatpu"는 탄자니아에 서식하는 작은 털복숭이 동물입니다. whatpu를 사용하는 문장의 예
라는 단어를 사용하는 문장의 예입니다:
우리는 아프리카를 여행하고 있었는데 아주 귀여운 whatpu를 보았습니다.

"farduddle"을 한다는 것은 정말 빠르게 위아래로 점프한다는 뜻입니다. farduddle을 사용하는 문장의 예
를 사용하는 문장의 예입니다:

Output:

게임에서 이것을 때 우리 모두는 farduddle를 시작했습니다. 하얀색을 떨 수 있습니다.

Few-shot Prompting은 몇 개의 예제를 포함하여 모델이 패턴을 학습하도록 유도하는 방식!

이전 Chat Template에서는 구조만 제공했다면, 이제 예제까지 포함해서 힌트를 제공

Few-shot Prompting (Few-shot Learning)

Prompt:

이 집합 {4, 8, 9, 15, 12, 2, 1}에서 홀수의 합은 짝수입니다.

A: 답은 거짓입니다.

이 집합 {17, 10, 19, 4, 8, 12, 24}에서 홀수의 합은 짝수입니다.

A: 정답은 참입니다.

이 집합 {16, 11, 14, 4, 8, 13, 24}에서 홀수의 합은 짝수입니다.

A: 답은 참입니다.

이 집합 {17, 9, 10, 12, 13, 4, 2}에서 홀수의 합은 짝수입니다.

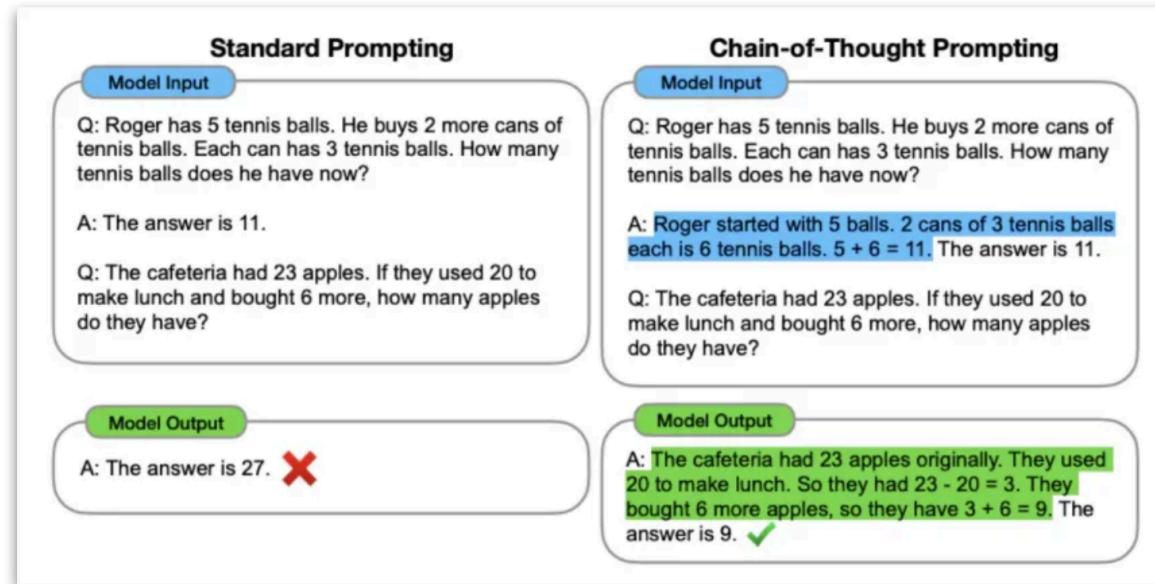
A: 답은 거짓입니다.

Output:

답은 참입니다. 

여러 샷을 제공하더라도 복잡한 추론 문제에서는 제대로 작동하지 않는 것을 종종 확인

Chain of Thought (CoT)



모델이 단계별로 사고할 수 있도록 유도하는 기법

충분히 큰 대규모 언어모델에서만 발생하는 특성이라고 주장!

<https://arxiv.org/abs/2201.11903>

Chain of Thought (CoT)

Prompt:

집합 {4, 8, 9, 15, 12, 2, 1}에서 홀수를 모두 더하면 짝수야.

답변: 홀수(9, 15, 1)를 모두 더하면 25가 돼. 위의 명제는 거짓이야.

집합 {17, 10, 19, 4, 8, 12, 24}에서 홀수를 모두 더하면 짝수야.

답변: 홀수(17, 19)를 모두 더하면 36이 돼. 위의 명제는 참이야.

집합 {16, 11, 14, 4, 8, 13, 24}에서 홀수를 모두 더하면 짝수야.

답변: 홀수(11, 13)를 모두 더하면 24가 돼. 위의 명제는 참이야.

집합 {15, 32, 5, 13, 82, 7, 1}에서 홀수를 모두 더하면 짝수야.

답변:

Output:

홀수(15, 5, 13, 7, 1)를 모두 더하면 41이 돼. 위의 명제는 거짓이야. 

복잡한 문제를 한 번에 풀게 하지 말고, ‘Let’s think step by step’ 등의 문구를 활용해 논리적으로 접근하도록 유도

3-2. Prompt Engineering

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Self-consistency Prompting



Chain of Thought 프롬프팅에 사용되는 일반적인 탐욕 알고리즘 디코딩을 대체하는 것

산술 추론에 대한 문제 다수



Prompt :

내가 6살이었을 때 내 여동생은 내 나이의 절반인 나이였어.
지금 나는 70살이라면 여동생은 몇 살일까?

Output :

35

같은 질문에 대해 여러 번 답변을 생성하고 다수결 방식으로 정답을 찾는 방식

LLM이 한 번에 정답을 못 찾을 수도 있기 때문에 **다양한 답변을 생성 후 최적의 답을 선택**하는데,
주로 산술 추론 문제에 활용

Self-consistency Prompting

Prompt:

Q: 과수원에는 15그루의 나무가 있어. 과수원의 일꾼들이 오늘 과수원에 나무를 심을 거야. 나무 심기가 끝나면 21그루의 나무가 있을 거야. 오늘 과수원 일꾼들은 몇 그루의 나무를 심었을까?

A: 15그루로 시작합니다. 나중에 나무가 21그루가 됩니다. 그 차이가 그들이 심은 나무의 수일 것입니다. 따라서 그들은 $21 - 15 = 6$ 그루의 나무를 심었어야 합니다. 정답은 6입니다.

Q: 주차장에 3대의 차량이 있고 2대의 차량이 더 도착하면 주차장에 몇 대의 차량이 있을까?

A: 주차장에 이미 3대의 차량이 있습니다. 2대가 더 도착합니다. 이제 $3 + 2 = 5$ 대의 차량이 있습니다. 정답은 5입니다.

Q: 지호는 초콜릿을 32개, 여동생은 42개를 가지고 있었어. 둘이 35개를 먹었다면 총 몇 개가 남았을까?

A: 레아는 초콜릿 32개, 레아의 여동생은 42개를 가지고 있었습니다. 즉, 원래 32개 + 42개 = 74개의 초콜릿이 있었습니다. 35개를 먹었습니다. 따라서 총 $74 - 35 = 39$ 개의 초콜릿이 남아 있습니다. 정답은 39입니다.

Q: 내가 6살이었을 때 내 여동생은 내 나이의 절반인 나이였어. 지금 나는 70살이고 내 여동생은 몇 살일까?

A:

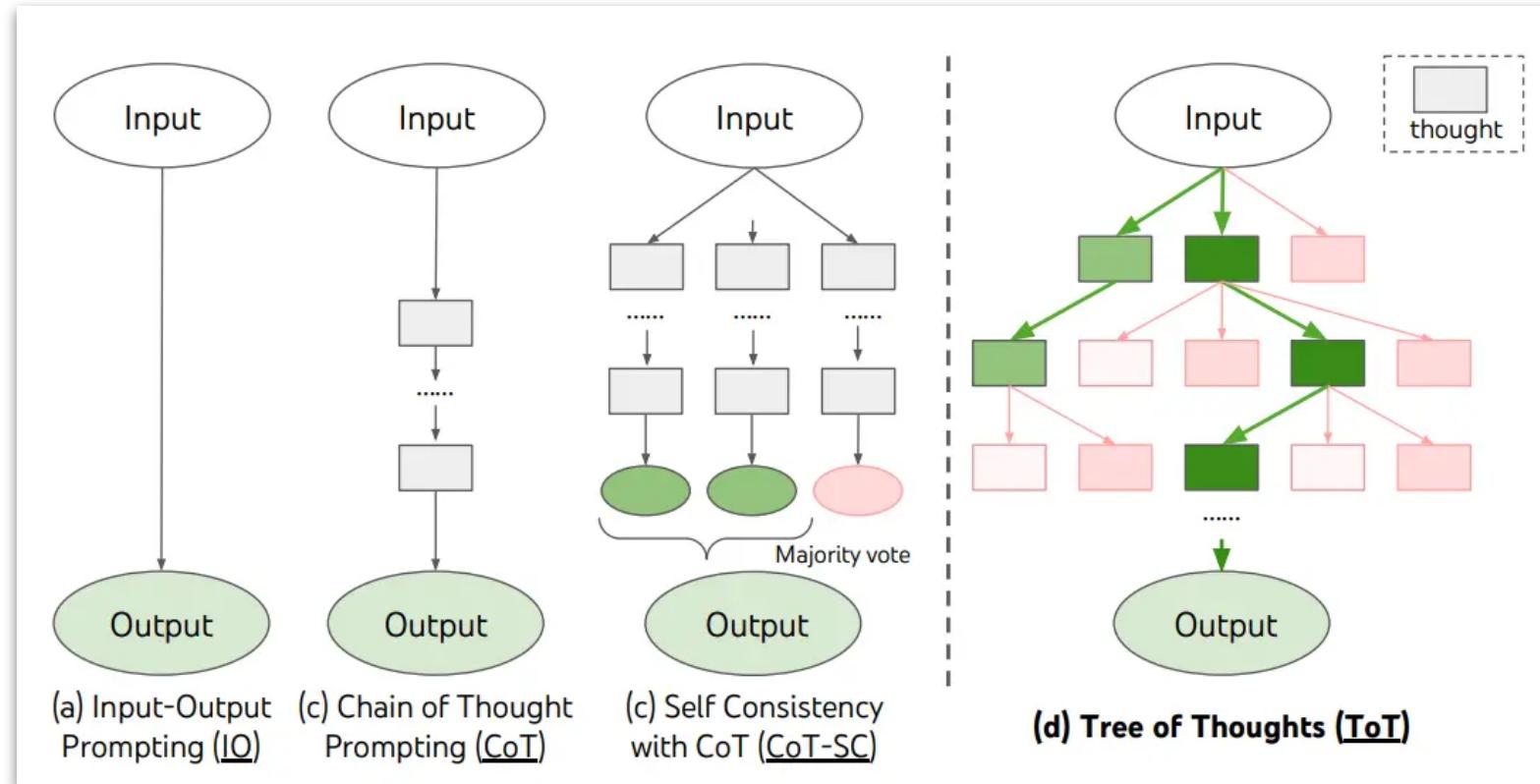
Output:

제가 6살이었을 때 여동생은 제 나이의 절반이었으므로 3살이었어요. 이제 저는 70살이므로 여동생은 $70 - 3 = 67$ 살입니다. 정답은 67입니다. ○

화자가 6살이었을 때 여동생은 화자의 절반 나이인 3살이었습니다. 이제 화자가 70살이 되었으므로 여동생은 $70 - 3 = 67$ 살이 됩니다. 정답은 67입니다. ○

내가 6살이었을 때 여동생은 내 나이의 절반이었으므로 3살이고, 이제 나는 70살이므로 여동생은 $70/2 = 35$ 살입니다. 정답은 35입니다. ✗

Tree of Thought (ToT)



CoT는 단순한 단계별 사고를 유도하지만, ToT는 여러 가능한 경로를 탐색하면서 최적의 답을 찾음
논리적 추론이 필요한 문제에서 유용

e.g. Chat Template + CoT 적용

```
messages = [
    {"role": "system", "content": "You are an AI tutor."},
    {"role": "user", "content": "Solve this math problem step by step: 27 + 14 × 2"}]
encoded_prompt = tokenizer.apply_chat_template(messages,
return_tensors="pt").to(device)

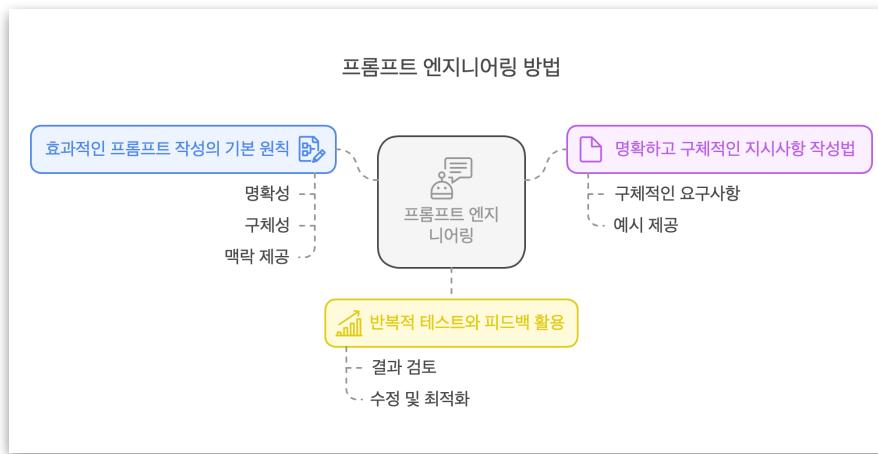
output = model.generate(encoded_prompt, max_new_tokens=100)
```

04 Parameter Efficient Fine-Tuning

PEFT로 효율적인 모델 만들기

4-1. Parameter Efficient Fine-Tuning

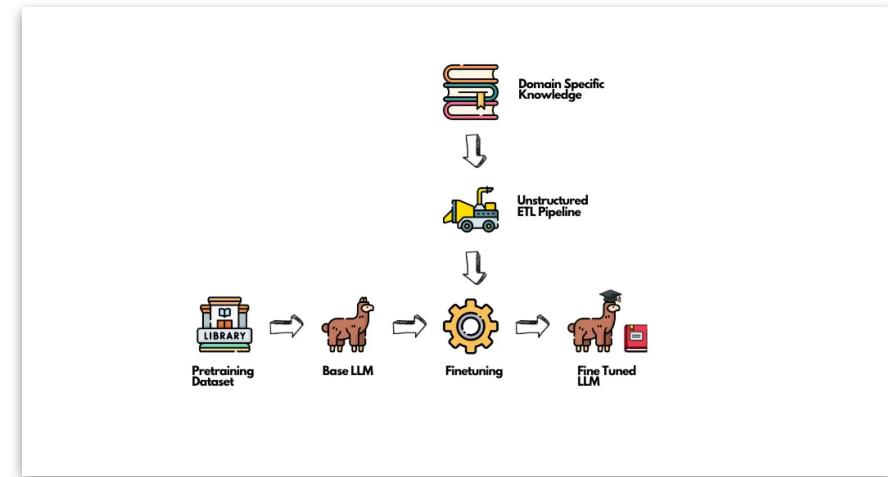
Prompt Engineering



모델을 출력을 조작하는 용도

Fine-Tuning

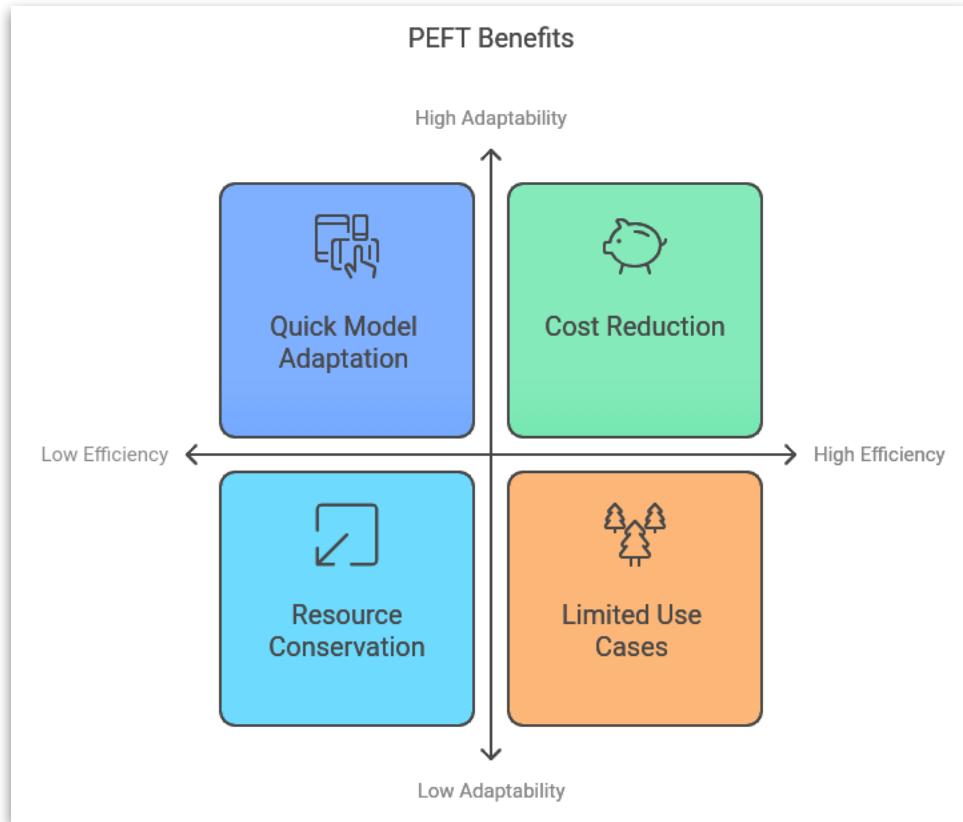
모든 파라미터를 미세조정



새로운 작업에 대한 예측 정확도 향상

▲ 자원 측면에서 한계

4-1. Parameter Efficient Fine-Tuning



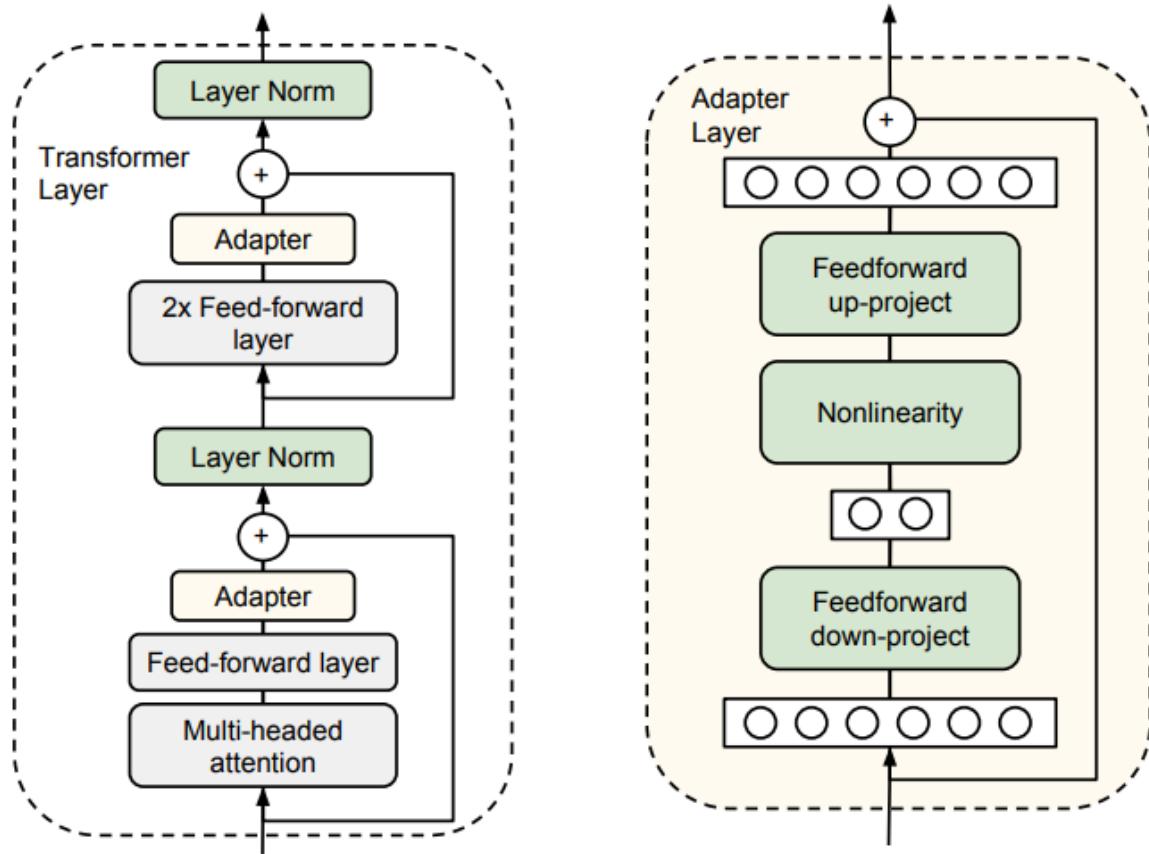
- 필요한 연산량 감소
- 추론 속도의 향상
- 추론 비용 절감
- 성능 유지 or 향상
- 리소스가 제한된 디바이스에서의 배포 개선



PEFT는 대규모 모델의 전체가 아닌 일부 매개변수만 조정하는 여러 가지 방법을 도입하여 대규모 모델을 보다 효율적이고 경제적으로 활용

4-2. Adaptive PEFT

* Adaptive : 적응할 수 있는

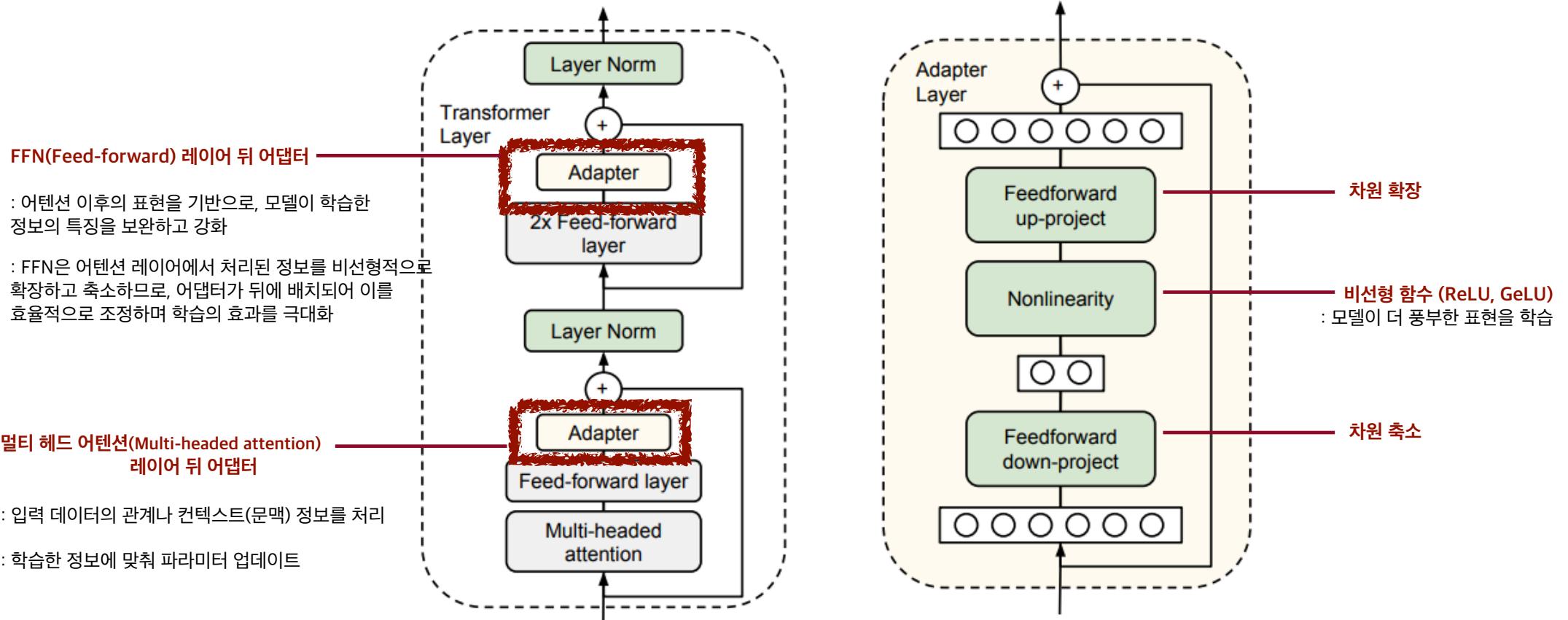


사전 학습된 백본을 변경하지 않음

- 특정 다운스트림을 위해 파인 투닝하면 추가된 어댑터의 파라미터만 업데이트
- 저장 공간과 메모리를 크게 절약

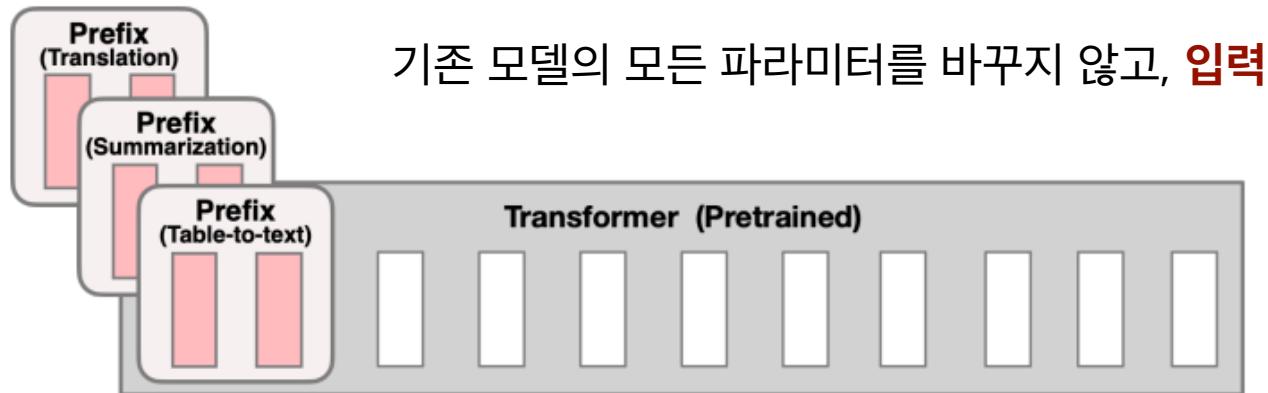
특정 작업이나 데이터 셋의 요구사항에 맞게 모델 아키텍처 내에 **학습 가능한 어댑터를 추가**하는 방법

4-2. Adaptive PEFT



4-3. Soft Prompt Tuning

* Prefix-Tuning으로도 잘 알려진 기법



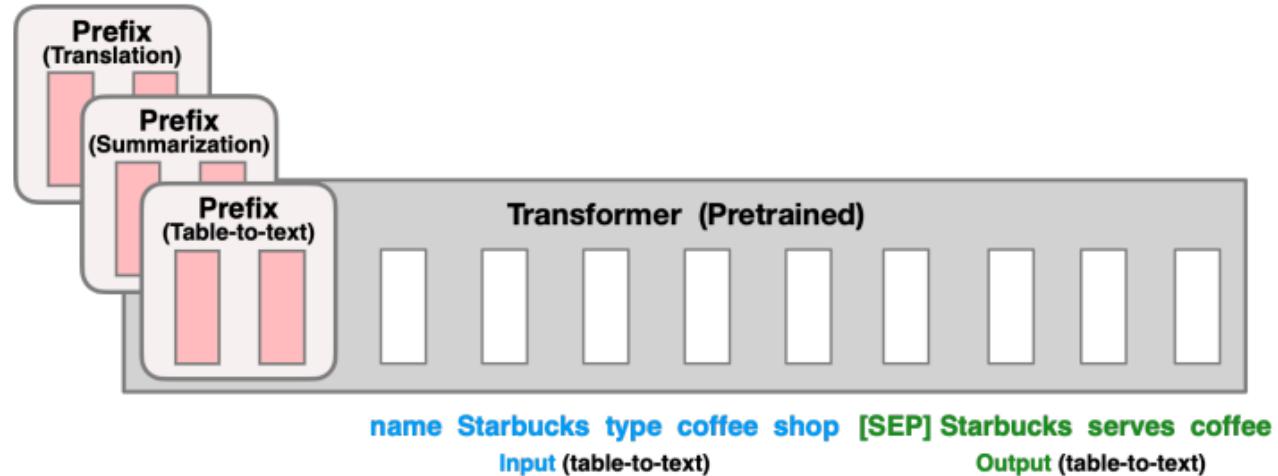
기존 모델의 모든 파라미터를 바꾸지 않고, **입력에** 약간의 "**힌트**"를 더하는 방식

기존방식 → 모델의 많은 부분을 다시 학습시키는 파인튜닝

현 방식 → **소프트 프롬프트(Soft Prompt)**
라고 불리는 조정 가능한 **벡터**를
입력 데이터의 시작 부분에 추가

Prefix-Tuning: Optimizing Continuous Prompts for Generation (Li and Lang et.al, 2021)

4-3. Soft Prompt Tuning



소프트 프롬프트

일반 텍스트 프롬프트(예: "질문: 이것은 무엇인가요? 답변:")처럼 고정된 문구 X 모델이 학습을 통해 유연하게 최적화할 수 있는 **연속적인 값들**로 표현

 모델이 입력 데이터를 더 잘 이해할 수 있도록 도와주며, 기존 모델 구조를 거의 바꾸지 않으면서 성능을 향상

 프롬프트 최적화가 매우 어려움, 학습 가능한 매개변수의 변화에 따라 성능이 불규칙적으로 변화

4-4. LoRA(Low-Rank Adaptation)

😊 PEFT 방법이 모델 전체를 파인튜닝하는 것보다 무조건 유리한거 아닌가

🤔 모델의 규모가 커지면 성능을 보장 모대….



LoRA : 사전 학습된 모델의 일부 가중치만을 저랭크(Low-rank) 구조로 업데이트하는 방법

기존방식 → 모델의 많은 부분을 다시 학습시키는 파인튜닝

LoRA → 핵심 가중치 행렬 업데이트를 **저랭크 분해**를 통해 간소화

LoRA: Low-Rank Adaptation of Large Language Models (Hu et.al, 2021)

내재적 차원(Intrinsic dimension)

- : 모델이 특정 작업을 학습하거나 적응하는 데 꼭 필요한 최소한의 표현 공간
- : 모델의 가중치 행렬은 보통 고차원 공간에 존재하지만, 실제로 학습에 사용되는 중요한 정보는 그보다 훨씬 낮은 차원으로 표현 가능!

LoRA는 이 개념을 활용하여, 사전 학습된 모델의 가중치를 고정하고, 필요한 가중치의 업데이트를 저차원 공간에서만 수행하게 되는데, 이 방법을 사용하면 원래 모델이 가진 복잡한 고차원 공간을 단순화하면서도 성능은 그대로 유지할 수 있기 때문에, 규모가 큰 모델에도 효과적으로 사용될 수 있음

4-4. LoRA(Low-Rank Adaptation)

학습해야 할 매개변수 수를 기존 방식보다
10,000배나 줄이면서도,
모델의 속도나 성능에는 영향 X

모든 것을 학습하는 대신,
꼭 필요한 부분만 작은 공간에서 효율적으로 조정한다!

LoRA의 핵심 아이디어 : 저랭크 분해

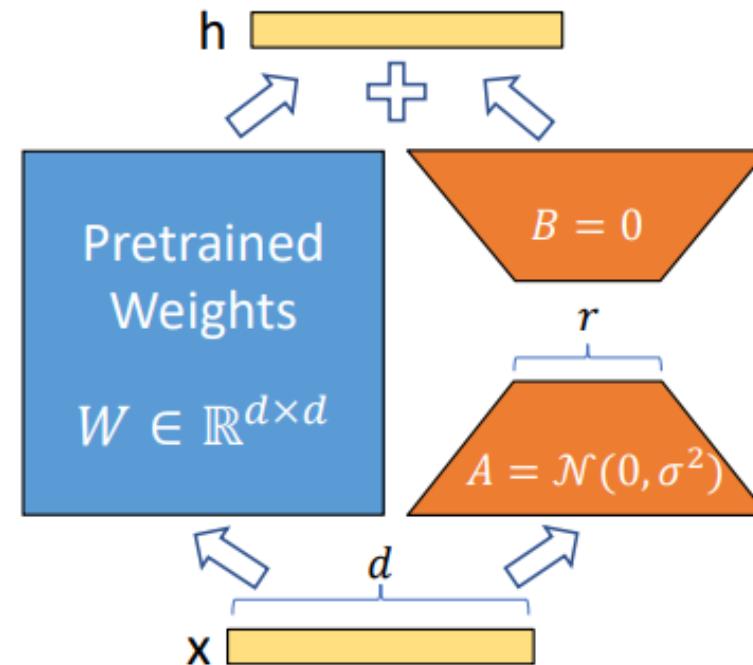
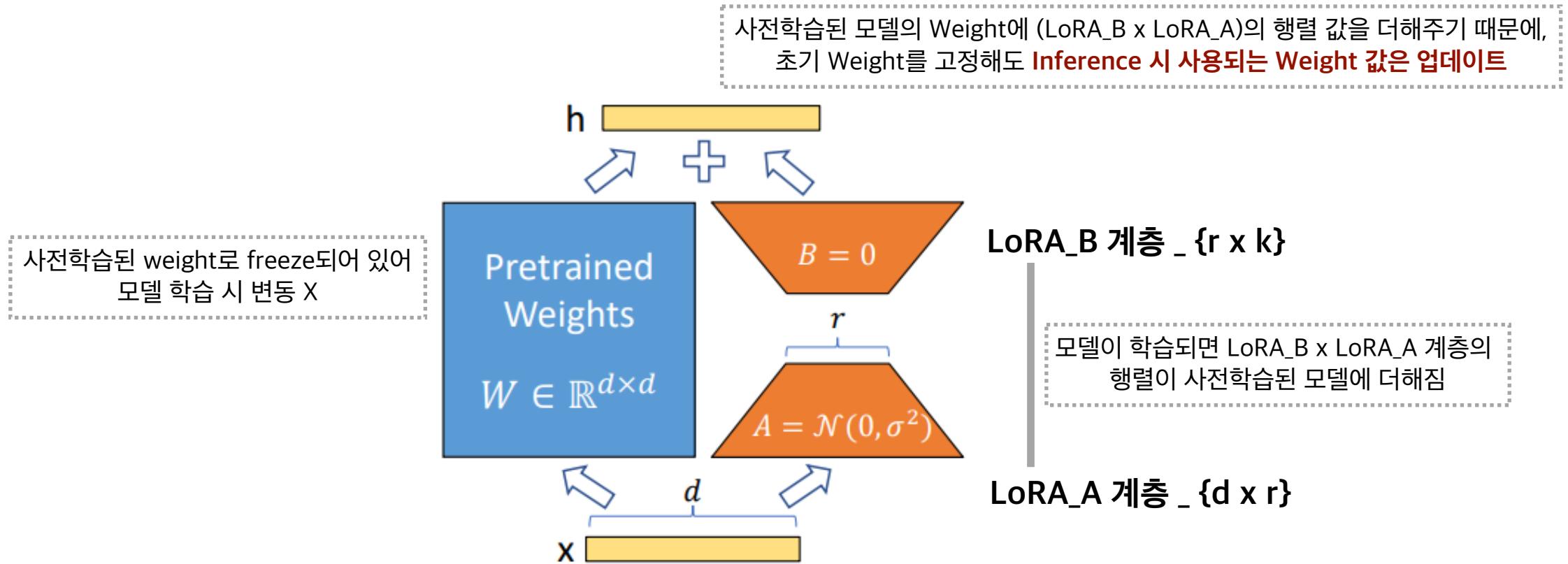


Figure 1: Our reparametrization. We only train A and B .

4-4. LoRA(Low-Rank Adaptation)



LoRA는 특히 초거대 모델에서 학습해야 할 매개변수를 기존 모델의 약 **0.1~0.5%** 수준으로 줄이면서도, 전체를 파인튜닝 하는 방법보다 우수한 성능 ★

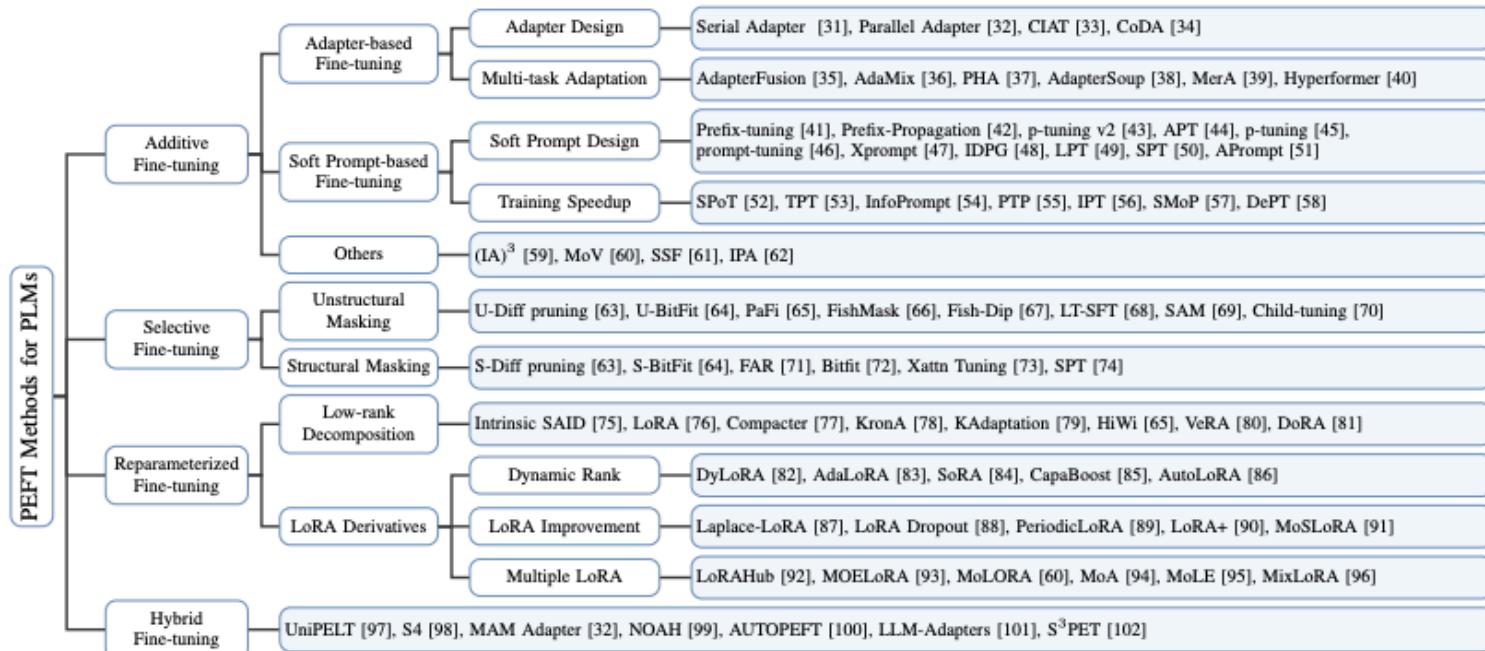


Fig. 3: Taxonomy of Parameter-Efficient Fine-Tuning Methods for Large Models.

작동 방식에 따라

- 추가적(Additive) **Adaptive PEFT, Prefix-tuning**
- 선택적(Selective)
- 재매개변수화(Reparameterization) **LoRA**
- 하이브리드(Hybrid)



05 Announcement

Week6 복습 과제 안내

5-2. Week6 복습과제 안내



개인
Paper Review

다음 논문 中 택 1
팀별 리뷰가 아닌 개인 리뷰



과제 제출은 **25-S KUBIG github**에 올바른 파일명으로 제출해주세요!

Prompt Engineering

- ***Large Language Models as Zero-Shot Reasoners*** (Brown et al., 2020)
- ***Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*** (Wei et al., 2022)
- ***Large Language Model Guided Tree-of-Thought*** (Yao et al., 2023)
- ***Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4*** (Zhu et al., 2023)

Parameter Efficient Fine-Tuning

- ***Parameter-Efficient Transfer Learning for NLP*** (Houlsby et al., 2019)
- ***Prefix-Tuning: Optimizing Continuous Prompts for Generation*** (Li and Lang et.al, 2021)
- ***LoRA: Low-Rank Adaptation of Large Language Models*** (Hu et.al, 2021)
- ***Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey*** (Han et al., 2024)

수고하셨습니다!