

언어모델의 이해와 실습

2025년 1학기

[UOS 고교학점제 정규교과 수업지원 프로그램]

강민설

1교시: 언어모델 개요

AI모델은 무엇일까요?

- 사람은 데이터를 만들고 컴퓨터는 함수를 만듭니다
- 이차방정식 함수 각 항에 대한 계수를 사람이 만들지 않습니다
- 사람은 x 에 따른 계산 결과(y)를 컴퓨터에게 제공합니다
- 컴퓨터는 x 와 y 의 관계를 학습합니다
- 여러 케이스에 대해 학습이 되면 실제 각 항의 계수에 근접한 값을 알 수 있습니다

사람이 만드는 함수와 AI가 만드는 모델

- 수학시간에 이차방정식의 계수는 사람이 정의했습니다

$$y = 2x^2 + 3x + 4$$

- AI에서는 x 와 y 를 사람이 정의하고 이차방정식의 계수를 컴퓨터가 만듭니다

$$y = 2x^2 + 3x + 4$$

사람이 만드는 함수와 AI가 만드는 모델

- 이 방식은 함수만 한번 정의하면 모든 x 에 대해 정확한 y 가 산출 됩니다

$$y = 2x^2 + 3x + 4$$

- 이 방식은 하나의 x, y 쌍만 가지고는 모든 x 에 대한 y 의 관계를 서술할 수 없습니다

$$y = 2x^2 + 3x + 4$$

사람이 만드는 함수와 시가 만드는 모델

- $(x, y) = (0, 4)$ 으로 아래의 이차방정식 계수를 만들어봅시다

$$y = ?x^2 + ?x + ?$$

- $a = 0, b = 0, c = 4$ 이면 되겠습니다

$$y = 0x^2 + 0x + 4$$

- 그러나 이 계수로는 $(x, y) = (1, 9)$ 를 충족할 수 없습니다

$$y = 2x^2 + 3x + 4$$

$$y = 2 + 3 + 4(x = 1)$$

사람이 만드는 함수와 AI가 만드는 모델

- $(x, y) = (0, 4)$ 과 $(x, y) = (1, 9)$ 2개 데이터로 계수를 만들어봅시다
- 연립방정식 해법을 고려하지 않고 생각합니다

$$y = 3x^2 + 2x + 4$$

- 그러나 이 함수는 $(x, y) = (-1, 3)$ 은 만족하지 않습니다
- 우리가 원했던 계수는 아래와 같았습니다
- 그러나 주어진 데이터만으로는 아래 계수를 도출하기 어렵습니다

$$y = \boxed{2}x^2 + \boxed{3}x + \boxed{4}$$

사람이 만드는 함수와 AI가 만드는 모델

- 그러나 우리는 여기서 드는 생각이 있을 것입니다
- 하나의 데이터만 더 있으면 연립방정식 해법으로 3개 계수를 모두 맞출 수 있지 않았을까요?
- 그러나 AI가 만드는 모델이라는 것은 연립방정식 해법을 사용하지 않습니다
- 임의의 계수를 설정하고 그 계수로 연산을 해본 후 오차를 보정하는 방식으로 계수를 맞추어 나갑니다
- 우리는 이것을 학습이라고 합니다

실험을 통해 알아봅시다

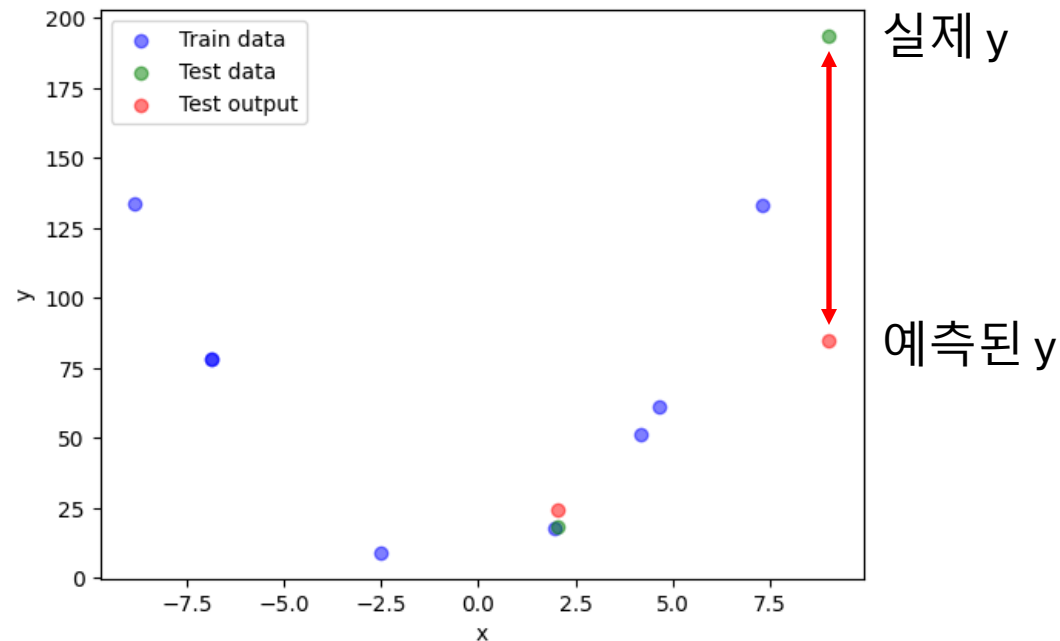
- 아래 식을 만족하는 (x, y) 쌍을 여러 개 만듭니다
- 아래 각 항의 계수는 컴퓨터에게 절대 알려주지 않습니다

$$y = 2x^2 + 3x + 4$$

- (x, y) 쌍의 80%를 이용해서 학습을 시킵니다
- (x, y) 쌍의 20%를 이용해 x를 대입하여 y가 맞게 나오는지 확인해봅니다

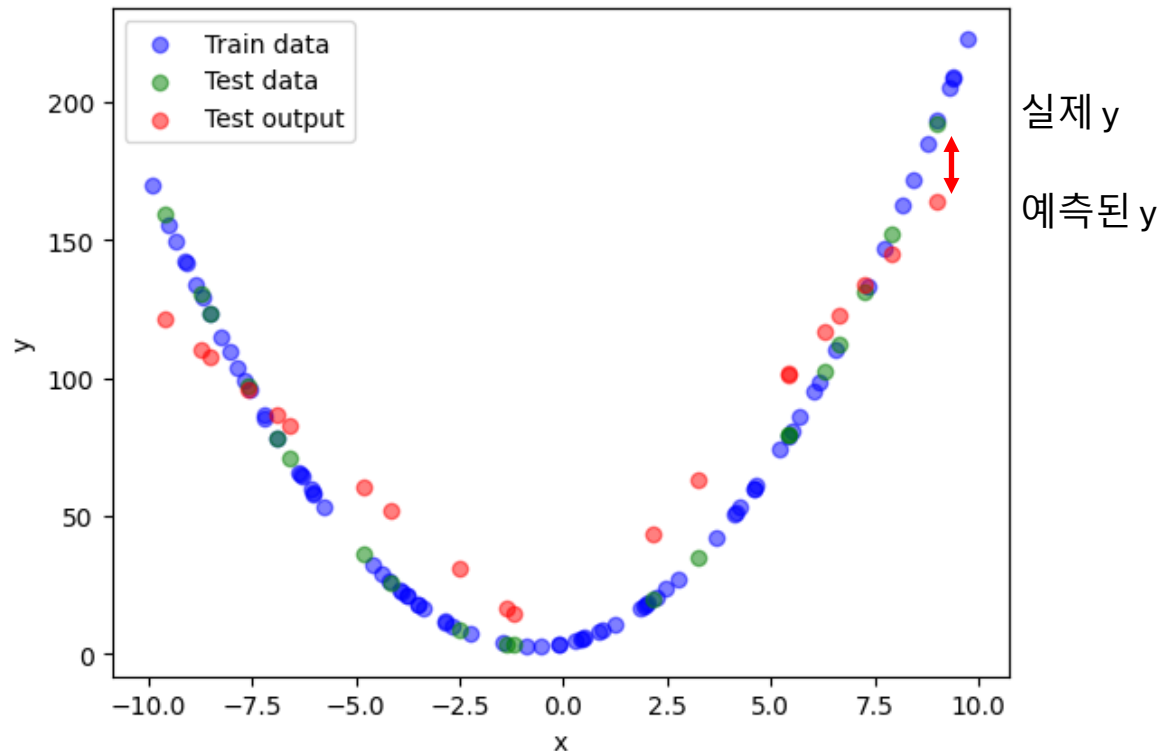
사람이 만드는 함수와 AI가 만드는 모델

- 데이터를 10개 사용했을 때 이차방정식의 계수를 맞추기
- 파란색 점은 학습에 사용된 (x, y) 쌍입니다
- 초록색 점은 테스트에 사용된 (x, y) 쌍입니다
- 붉은색 점은 테스트에 사용된 x 를 모델에 대입했을 때 출력된 y 입니다



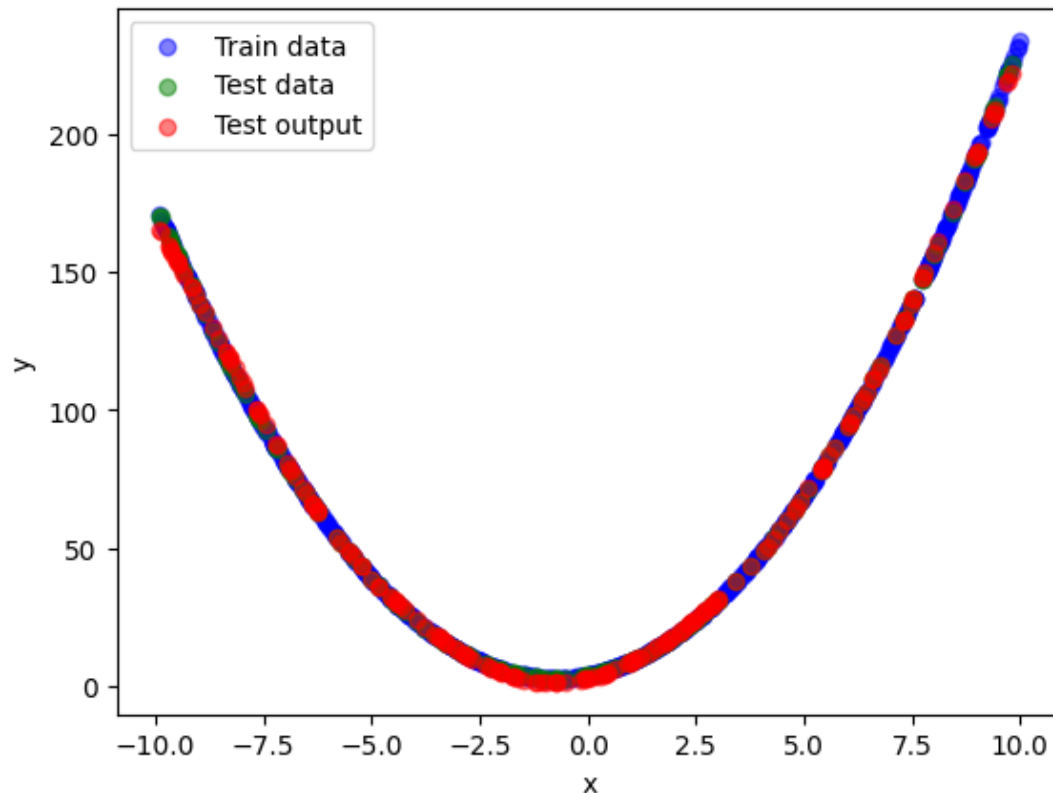
사람이 만드는 함수와 시가 만드는 모델

- 데이터를 100개 사용했을 때 이차방정식의 계수를 맞추기
- 10개 데이터로 학습했을 때보다 예측 오차가 줄어들었으나 여전히 오차가 있습니다



사람이 만드는 함수와 AI가 만드는 모델

- 데이터를 1000개 사용했을 때 이차방정식의 계수를 맞추기
- 테스트에 사용한 x로 모델이 연산한 결과와 실제 y가 거의 일치합니다

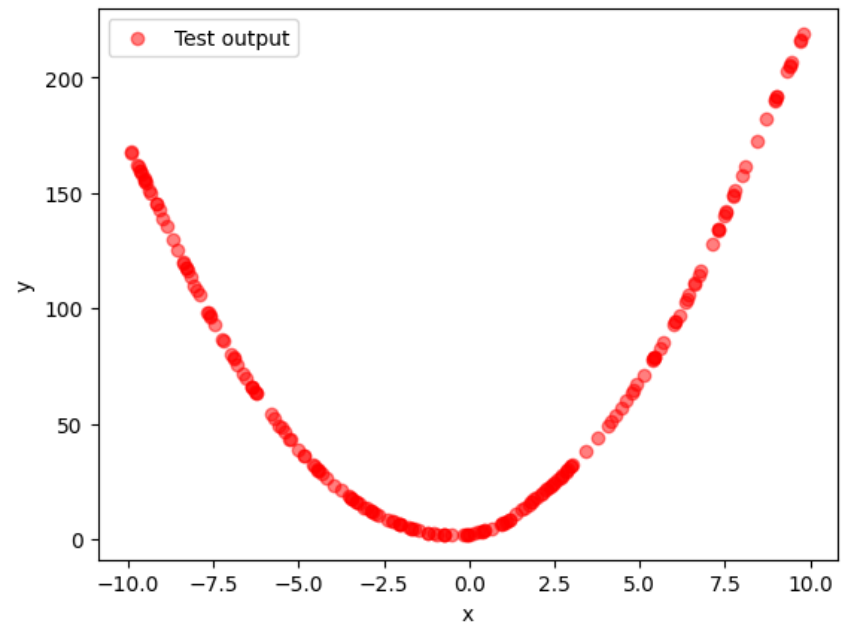
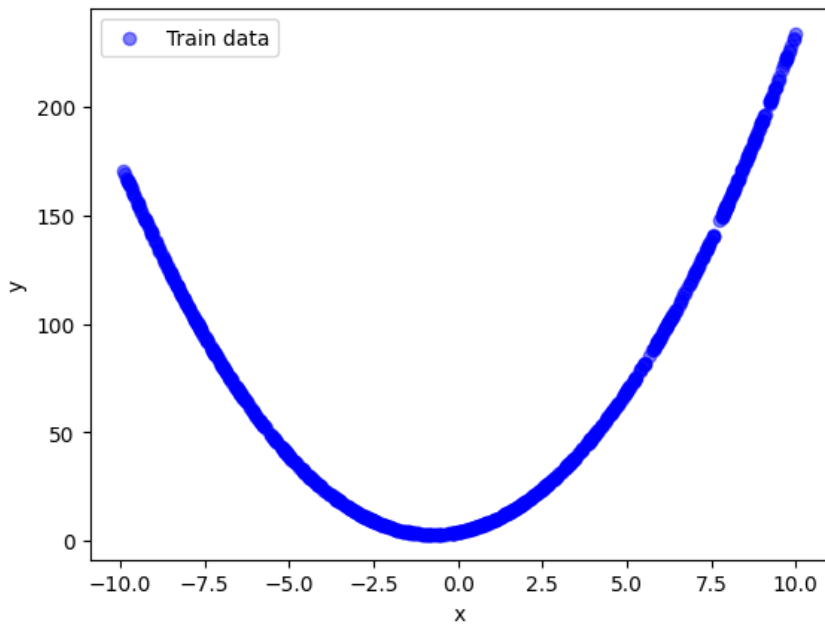


AI학습의 효율성

- 우리는 3개의 (x, y) 쌍만 가지고 계수를 정확히 도출할 수 있습니다
- 그러나 AI는 무려 1,000개의 데이터가 필요했습니다
- 연립방정식만 놓고 보면 비효율적입니다
- 단, 연립방정식을 몰라도 계수에 근접한 값을 알아낼 수 있습니다
- 데이터만 충분하면 이차방정식보다 더 복잡한 문제를 근사적으로 풀 수 있다는 것입니다

근사적이라는 말의 의미

- 아래 붉은 점은 이차방정식 계수에 의해 y 를 계산해 낸 것이 아닙니다
- 학습된 데이터를 기반으로 추정했을 때 확률적으로 가장 근사적인 y 가 도출된 것입니다



이차방정식보다 복잡한 문제

- 이차방정식보다 복잡한 문제는 너무나 많습니다
- 대표적인 것이 우리의 언어입니다
- 우리가 어떠한 말을 하면 그 다음 이어질 문장은 무엇일까요?
- 언어모델이 바로 이를 학습한 모델입니다
- 우리는 문장을 숫자로 바꿀 수 있습니다
- 그러나 이것을 그대로 이차방정식에 대입해서는 풀기 어렵습니다

$$\text{이어질문장} \neq a(\text{제시한문장})^2 + b(\text{제시한문장}) + c$$

언어모델에서의 x와 y

- “오늘 날씨가 참 좋네요. 산책을 나가볼까요?” 라는 문장이 있다고 가정해봅시다
- 각 문장은 아래와 같이 숫자로 변환될 수 있습니다
- 각 문장이 x와 y가 된다고 가정하면 학습을 시킬 수 있겠습니다

오늘 날씨가 참 좋네요.

['_오늘', '_날씨가', '_참', '_좋', '_네', '_요.']
[10070, 34018, 9338, 9677, 7098, 25856]

산책을 나가볼까요?

['_산', '_책을', '_나가', '_볼', '_까', '_요', '_?']
[9145, 12115, 12312, 7656, 6969, 8084, 406]

Vocab과 Tokenizer

- 앞서 문장이 숫자로 변환된 것은 무엇을 근거로 할까요?
- 각 글자에 대해 숫자로 변환되는 규칙이 있는 것입니다
- 그 규칙을 언어모델에서는 vocab 이라고 합니다
- Vocab을 따라 숫자로 변환하는 행동을 벡터화라고 합니다
- Vocab을 이용해 문장을 벡터화 해주는 것을 Tokenizer라고 합니다

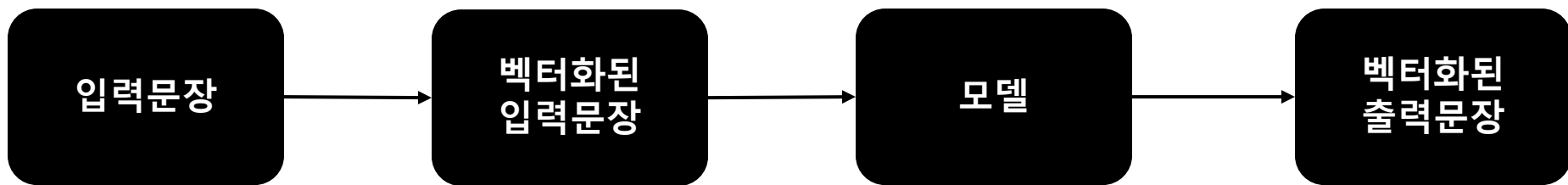
오늘 날씨가 참 좋네요.

['_오늘', '_날씨가', '_참', '_좋', '_네', '_요.']
[10070, 34018, 9338, 9677, 7098, 25856]

Tokenizer에 의해 Vocab 체계를 따라 벡터화

모델

- 이차방정식에서 모델은 x 에 대응되는 가장 근사적인 y 를 만들어주었습니다
- 이어지는 문장만들기에서 모델은 벡터화된 입력문장에 대응되는 가장 근사적인 출력문장을 만들어줍니다
- 이 때의 출력문장은 벡터화된 출력문장입니다



모델

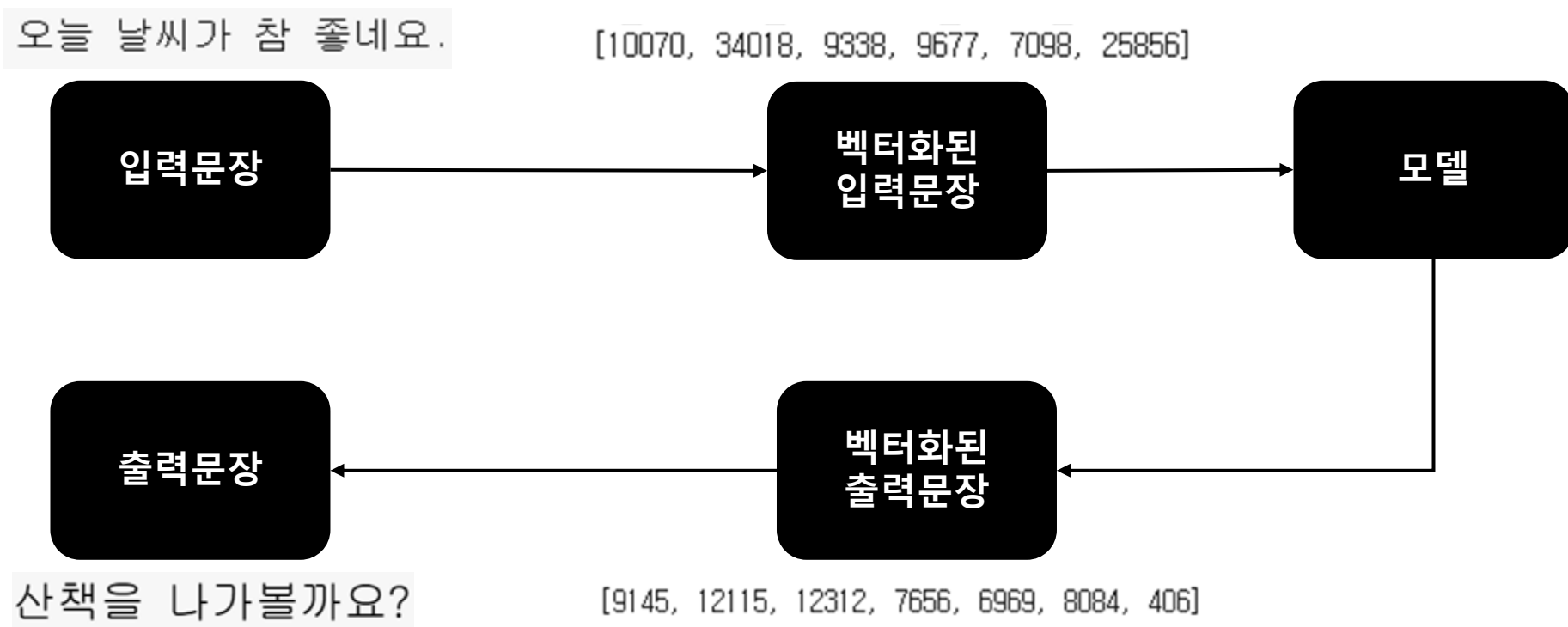
- 언어모델에서 모델이라는 것은 분명히 어떠한 수식을 가지고 있습니다
 - 그 수식을 표현하는 방법에 따라 모델의 이름이 정해집니다
 - GPT, Llama, Deepseek와 같은 것입니다
 - 그 수식을 표현하는 계수와 상수들을 파라미터라고 합니다
 - 파라미터의 수를 모델의 이름에 함께 기재하기도 합니다
-
- Llama-3.1-8B: Llama-3.1에서 정의한 구조를 따라 수식을 표현했고 파라미터의 수는 80억개
 - GPT3-175B: GPT3에서 정의한 구조를 따라 수식을 표현했고 파라미터의 수는 1750억개

모델의 학습

- 우리가 이차방정식의 계수를 맞추기 위해 데이터를 1개, 2개 사용하면서 바꾼 것과 비슷합니다
- 기존의 계수와 상수 의해 산출된 결과와 실제 결과와의 오차를 보정하는 방식으로 계수와 상수를 조정합니다
- 이를 학습이라고 합니다
- GPT3-175B 모델을 기준으로 보면 1750억개의 계수와 상수를 보정하는 것입니다
- 우리 손으로는 하기 어렵습니다
- 그래서 컴퓨터 하드웨어, 특히 GPU 라는 장비의 힘을 사용합니다

Detokenize

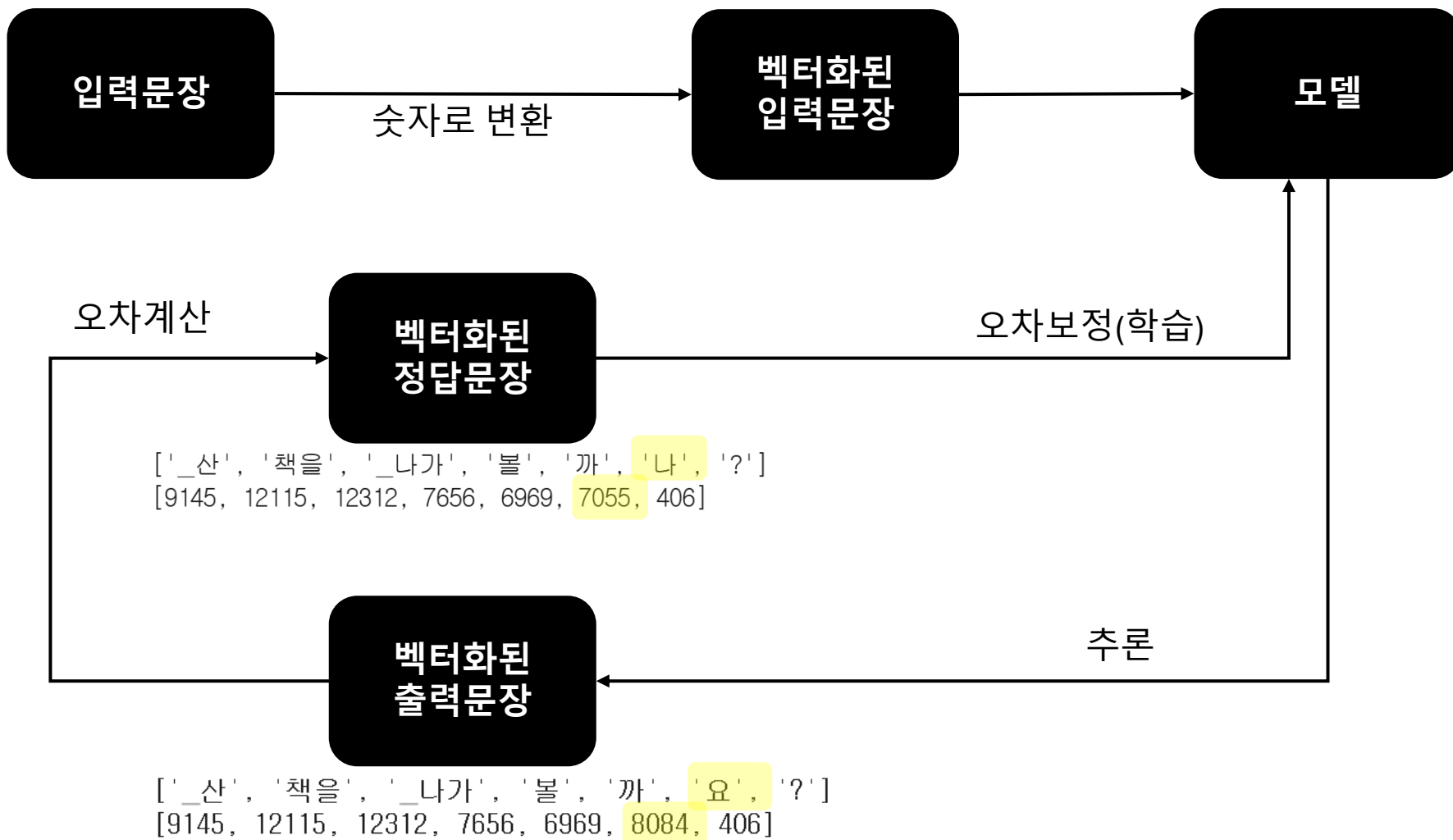
- 벡터화된, 숫자로 된 출력문장을 우리의 언어로 바꾸기 위해서는 Tokenizer를 이용하면 됩니다
- 이를 Detokenize라고 합니다



언어모델의 학습 과정 요약

오늘 날씨가 참 좋네요.

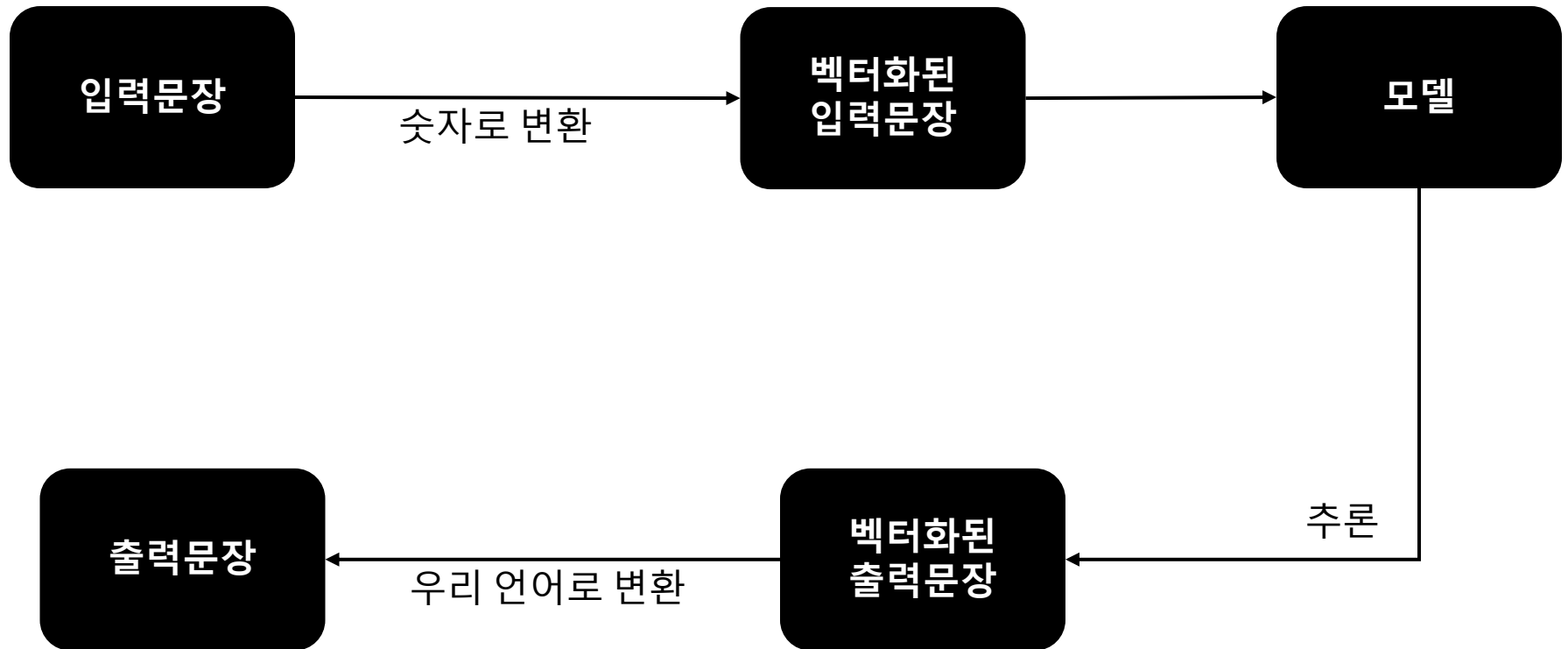
[10070, 34018, 9338, 9677, 7098, 25856]



언어모델의 추론 과정 요약

오늘 날씨가 참 좋네요.

[10070, 34018, 9338, 9677, 7098, 25856]



산책을 나가볼까요?

[9145, 12115, 12312, 7656, 6969, 8084, 406]

정리

- 이차방정식을 AI모델이 학습하는 것은 연립방정식의 해법을 사용하는 것이 아니라 여러 x, y 의 쌍으로부터 추정하여 임의의 x 에 대한 근사적인 y 를 도출하는 것입니다
- 작은 문제는 AI모델로 풀기에는 비효율적이지만 사람이 함수를 만들어 정의하기에 어려운 문제를 풀기에는 좋은 방법입니다
- 사람이 함수를 만들어 정의하기에 어려운 문제중 하나가 이어지는 문장 만들기 이고 이것을 AI모델로 풀인 것이 언어모델 입니다
- 언어모델이란 입력문장에 이어지는 가장 확률 높은 문장을 출력하는 수식체계 입니다
- 모델의 이름에 따라 수식체계의 구조가 다르고 이를 구성하는 계수와 상수 즉, 파라미터의 수가 다릅니다

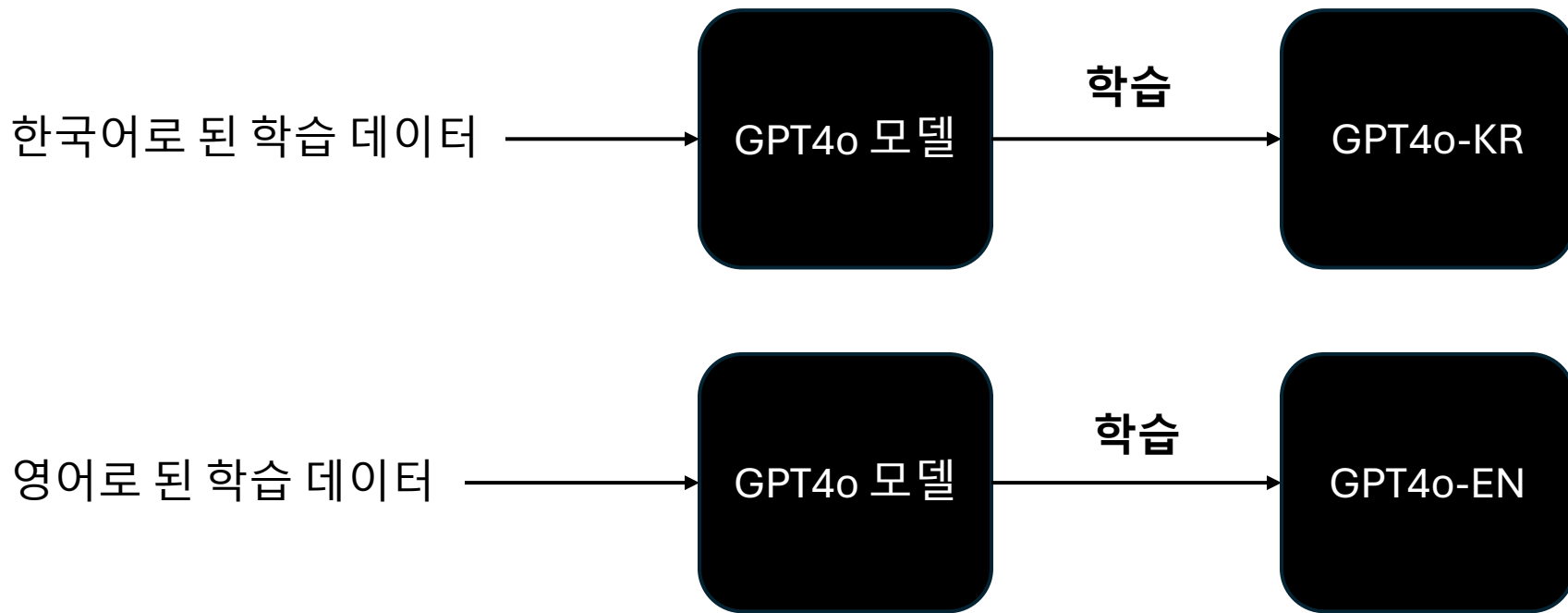
2교시: 언어모델을 내 컴퓨터에 설치하기

ChatGPT에서 사용된 모델

- ChatGPT는 서비스 이름입니다
- 여기에도 언어모델이 사용되었습니다
- 초창기의 ChatGPT에는 GPT3-175B가 사용되었습니다
- GPT3-175B 모델은 어떻게 만들어졌을까요?

모델을 구성하는 요소

- 아래 2개 모델은 같은 모델일까요?
- 모델의 수식체계가 같아도 제공되는 데이터에 따라 계수와 상수는 다르게 조정됩니다
- 따라서 아래 2개 모델은 다른 모델입니다



모델을 구성하는 요소

- 수식 체계: 이차방정식으로 구성되었는지, 삼차방정식으로 구성되었는지에 따라 항의 차수가 다르고 항의 수가 다른 것과 같습니다
- 계수와 상수의 값: 같은 이차방정식이라고 해도 계수와 상수의 값이 다르면 다른 방정식입니다
- 우리는 AI로 이차방정식을 학습할 때 주어지는 데이터에 따라 모델이 달라지는 것을 이미 확인했습니다
- 따라서 모델의 구조가 동일하더라도 학습에 사용한 데이터가 다르면 다른 모델입니다

$$y = ax^2 + bx + c$$

모델의 구조

$$y = 2x^2 + 3x + 4$$

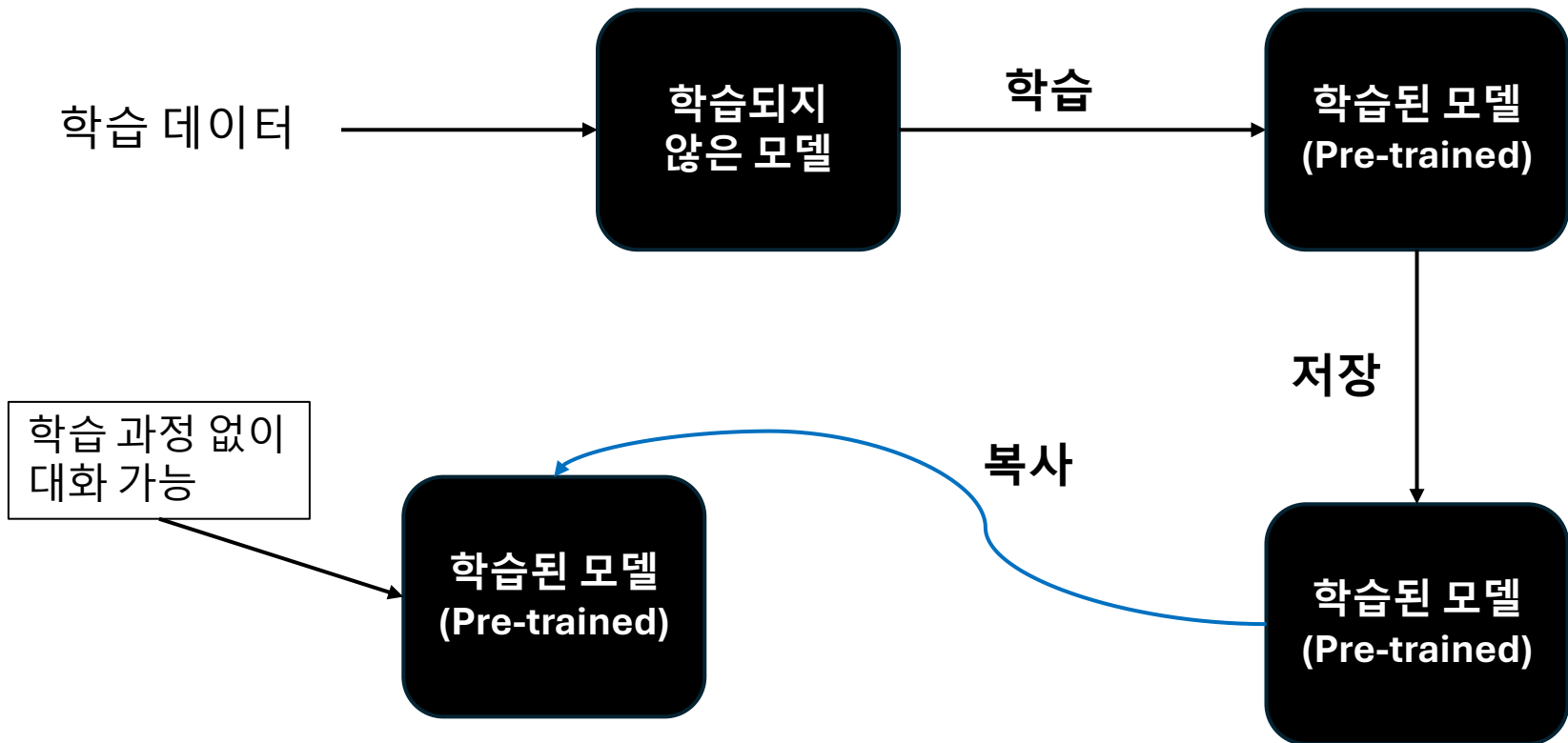
학습된 계수와 상수에 의해 정의된 방정식

GPT3-175B 모델 학습에 사용된 시간

- ChatGPT에서 사용된 GPT3-175B 라는 모델을 만들기 위해 모델의 구조를 정의하는 과정 뿐만 아니라 방대한 데이터를 준비하여 학습하는 과정 또한 존재했습니다
- GPT3 논문을 따르면 총 학습된 계산량이 3.14×10^{23} 제곱번의 실수 연산을 했다고 합니다
- 고가의 GPU 카드인 RTX 4090 1장을 사용했을 때 130년 정도 걸리는 연산량 입니다
- 물론 시간이 흘러 더 적은 연산량으로도 좋은 성능을 내는 모델들이 나왔습니다
- 그러나 여전히 개인이 진행하기에는 너무나 큰 작업입니다

Pre-trained 모델

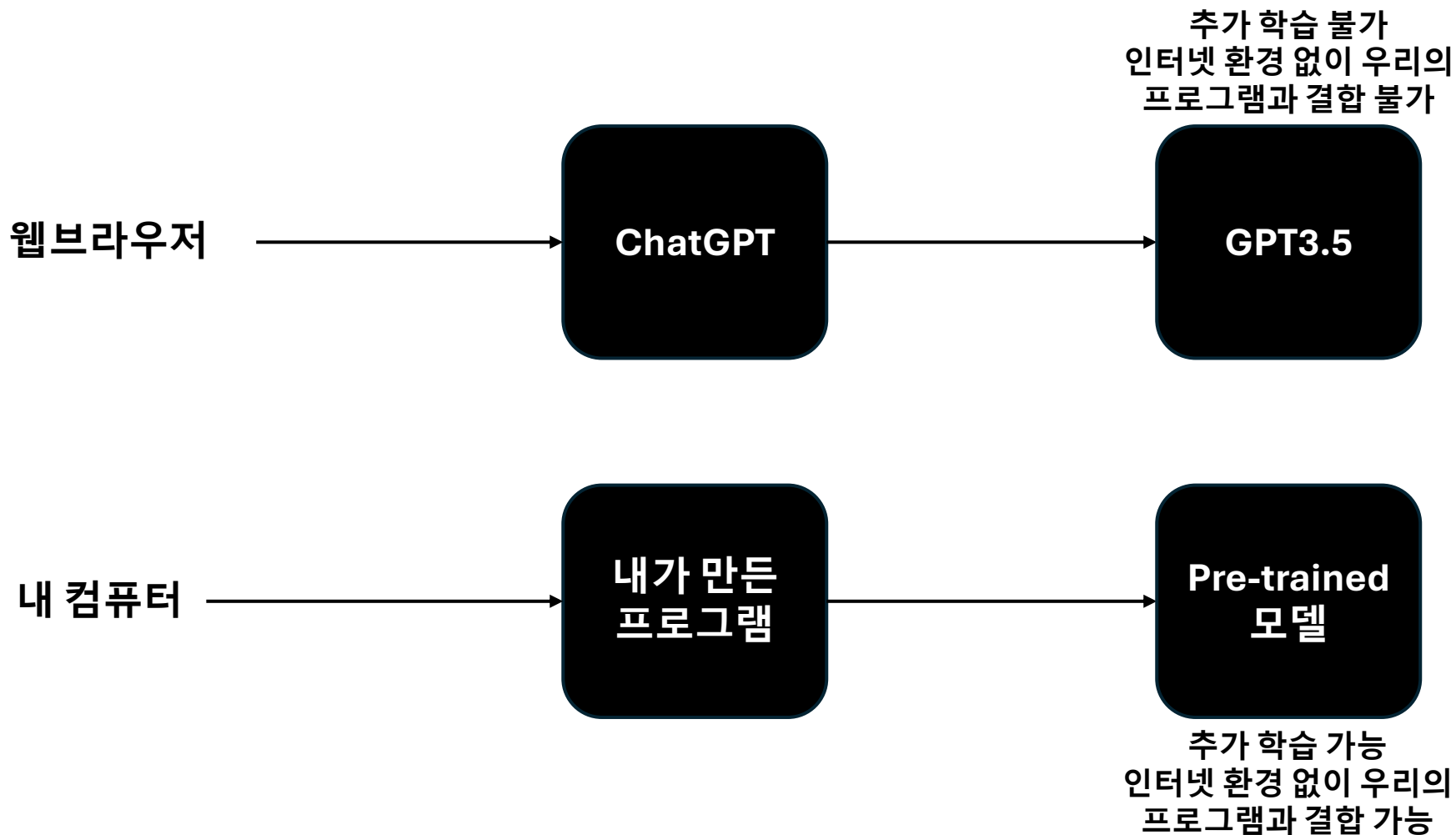
- 그래서 학습이라는 과정을 완료하여 저장된 것을 활용하면 돈과 시간을 절약할 수 있습니다
- 이것을 Pre-trained 모델이라고 합니다



오픈소스 Pre-trained 모델

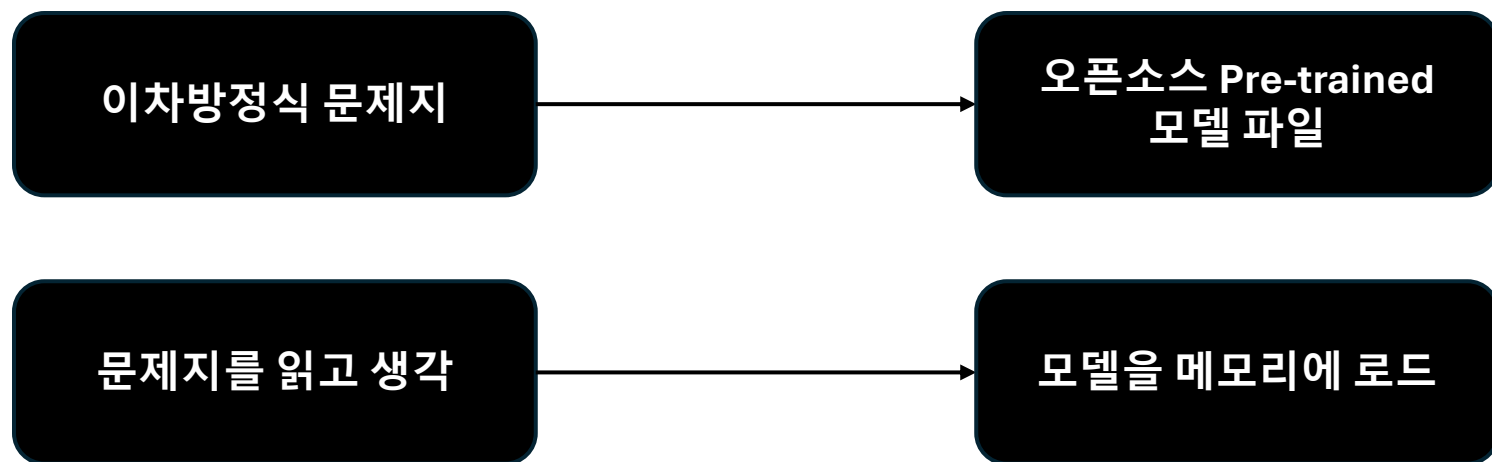
- 아쉽게도 GPT3와 그 이후로는 공개가 되지 않은 상태입니다
- 그러나 이와 비슷한 성능을 가진 모델이 공개가 되었습니다
- Meta의 Llama 입니다
- Llama의 공개를 기점으로 하여 많은 기업들에서 언어모델을 활용한 서비스 개발이 이루어졌습니다

언어모델을 직접 다운로드 받아 사용한 다는 것의 의미



오픈소스 Pre-trained 모델의 구동

- 이를 위해서는 두가지가 필요합니다
- 먼저 오픈소스 Pre-trained 모델 파일을 다운로드 받아야 합니다
- 그리고 오픈소스 Pre-trained 모델 파일을 메모리에 올려놓아야 합니다
- 오픈소스 Pre-trained 모델 파일에 저장된 것은 모델의 구조와 계수, 상수의 값이고 이를 메모리에 올려야 합니다
- 이차방정식을 계산하기 위해서는 이차방정식이 서술된 문제지와 이 수식을 읽고 우리의 머리속으로 생각하는 과정이 필요한 것과 같습니다



LLM 프레임워크

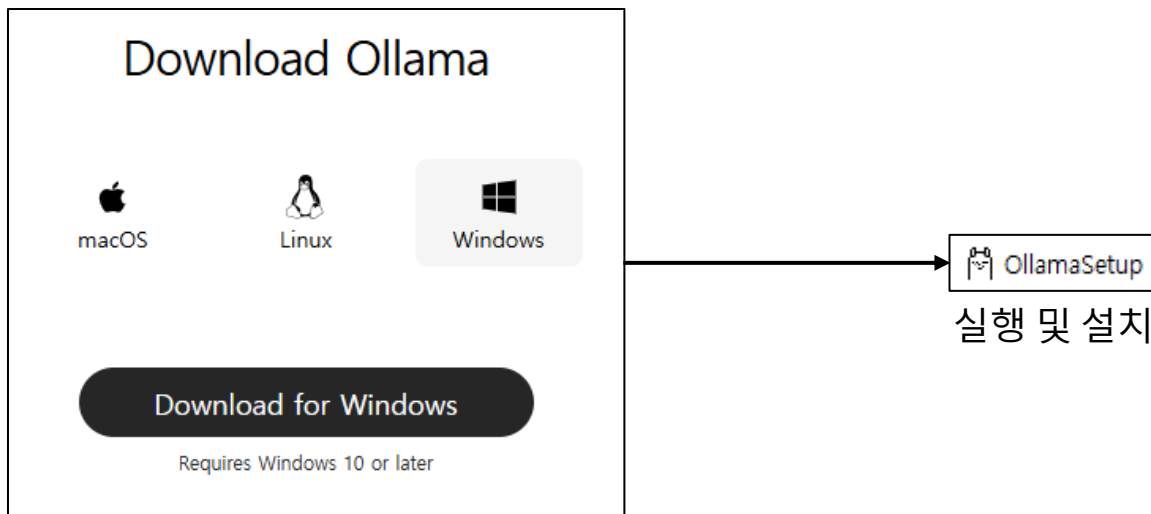
- 이러한 과정을 도와주는 도구를 LLM 프레임워크라고 합니다
- 프레임워크를 사용함으로써 복잡한 사용과정을 단순화하여 쉽게 사용 가능합니다
- 대신 프레임워크가 정한 방식대로 사용해야 하는 제약은 있습니다
- 프레임이라는 단어를 생각해보면 당연한 말입니다
- 그러나 우리에게 그 제약은 사소하기 때문에 LLM 프레임워크를 사용해보도록 합니다

ollama

- LLM 을 쉽게 다운로드 받고 테스트할 수 있도록 해주는 도구 중 하나입니다
- ollama를 통해 다운로드 받을 수 있는 모델은 <https://ollama.com/library> 를 참조 합니다
- 우리는 Llama3.2:1b (<https://ollama.com/library/llama3.2>)를 사용해보겠습니다
- 모델을 구성하는 계수와 상수를 지금부터는 파라미터라고 표현합니다
- Llama3.2:1b는 파라미터의 수가 10억개밖에(?) 되지 않는 경량 모델이며 GPU가 없는 환경에서도 사용 가능합니다

ollama 다운로드

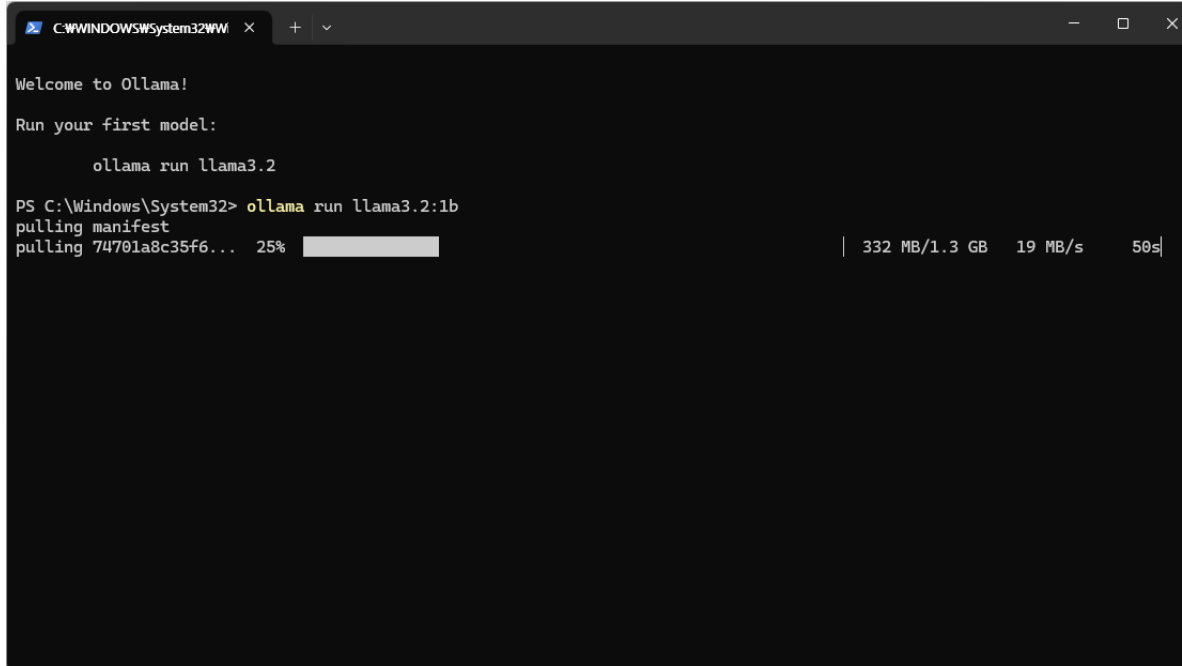
- Windows PC 기준으로 ollama는 설치파일을 다운로드 받아서 설치할 수 있습니다
- <https://ollama.com/download/windows> 에서 아래와 같이 Download for Windows 를 눌러서 다운로드 받습니다



ollama 를 이용한 llama3.2:1b 모델 구동

- 명령 프롬프트를 실행하고 아래와 같이 구동합니다
- 모델 파일을 다운로드 받고 컴퓨터의 메모리로 로드할 때까지 기다립니다

```
ollama run llama3.2:1b
```



```
Welcome to Ollama!
Run your first model:
ollama run llama3.2
PS C:\Windows\System32> ollama run llama3.2:1b
pulling manifest
pulling 74701a8c35f6... 25% | 332 MB/1.3 GB 19 MB/s 50s
```

llama3.2:1b 와 채팅

- success 라는 문구와 함께 >>> 이 나오면 모델의 로드가 완료된 것입니다
- 이제 입력문장을 전달하여 출력문장을 생성할 수 있습니다
- ChatGPT와 대화하는 것처럼 사용할 수 있습니다
- ChatGPT 서버가 아닌 여러분의 컴퓨터에서 실행되고 있는 것입니다

```
C:\WINDOWS\system32\cmd.exe
Welcome to Ollama!

Run your first model:

ollama run llama3.2

PS C:\WINDOWS\system32> ollama run llama3.2:1b
pulling manifest
pulling 74701a8c35f6... 100% 1.3 GB
pulling 966de95ca8a6... 100% 1.4 KB
pulling fcc5a6bec9da... 100% 7.7 KB
pulling a70ff7e570d9... 100% 6.0 KB
pulling 4f659a1e86d7... 100% 485 B
verifying sha256 digest
writing manifest
success
>>> what is language model?
A language model is a computer system designed to understand, interpret, and generate human language. It's a type of artificial intelligence (AI) that uses machine learning algorithms to process and analyze natural language data.

Language models are trained on vast amounts of text data, such as books, articles, conversations, and more. This training enables the model to learn patterns, relationships, and context within language, allowing it to generate coherent and sometimes even creative text.

There are different types of language models, including:

1. **NLP (Natural Language Processing) Models**: These models focus on understanding and generating human language, such as
```

정리

- 학습 데이터가 다르면 파라미터의 값이 다르게 조정되므로 학습 데이터가 다르면 다른 모델이 됩니다
- 우리가 사용하는 ChatGPT는 방대한 양의 데이터로 학습되었습니다
- ChatGPT와 같은 거대언어모델을 우리가 직접 학습시키기에는 어렵습니다
- 이러한 학습의 결과를 공개한 오픈소스 Pre-trained 모델이 있습니다
- 그리고 이를 쉽게 다운로드 및 구동할 수 있는 LLM 프레임워크가 있습니다
- ollama라고 하는 LLM 프레임워크를 이용하면 llama3.2:1b 모델을 다운로드 하여 우리의 컴퓨터에서 구동하고 채팅을 진행해볼 수 있습니다

3교시: 언어모델을 내 프로그램과 결합하기

내 프로그램과 언어모델을 결합

- 언어모델을 프로그래밍 언어와 결합하는 것을 의미합니다
- 이를 이용해서 ChatGPT 서비스에서는 할 수 없는 무언가를 해보고자 합니다
- 이번 시간에는 2개의 다른 언어모델을 다운로드 받고 프로그래밍 언어를 이용하여 이들 둘이 서로 대화를 하도록 시켜보겠습니다
- 프로그래밍언어는 python언어를 사용합니다

언어모델과 프로그래밍언어간의 연결고리

- 언어모델이 실행되는 것은 언어모델 서버를 구동하는 형태입니다
- 프로그래밍언어가 이 언어모델 서버와 통신할 수 있게 도와주는 연결고리가 있습니다
- 이것을 ollama python 라이브러리라고 합니다
- 따라서 ollama python 라이브러리를 먼저 설치해야 합니다
- 실습 파일을 열고 아래와 같이 입력하여 라이브러리를 설치합니다

```
! pip install ollama
```

추가 모델 다운로드

- 명령 프롬프트를 열고 아래와 같이 추가 모델을 다운로드 받습니다
- <https://ollama.com/library> 에서 적당한 모델을 선택해봅니다
- 1b 사이즈 내외의 모델을 선택하는 것이 좋습니다
- 예를 들면 <https://ollama.com/library/qwen> 을 참고하여 아래와 같이 다운로드 받습니다

```
ollama pull qwen:0.5b
```

각 언어모델 테스트

- 아래와 같이 model 변수의 값을 바꾸면서 정상 동작 하는지 확인합니다

```
import ollama

model='llama3.2:1b'
# model='qwen:0.5b'

messages = [{
    'role': 'user',
    'content': "Tell me this situation as a simple one word, 'I got A+ in my Math class!'"
}]

response = ollama.chat(model=model, messages=messages)
content = response['message']['content']

print(content)
```

✓ 2.5s

Proudly

두 모델간의 대화

- 시작 문장에 따라 이들 둘의 대화가 달라질 것입니다
- 또한 어떤 모델을 쓰는지에 따라 이들 둘의 대화가 달라질 것입니다
- 시작 문장을 바꾸면서, 모델을 바꾸면서, 대화의 순서를 바꾸면서 반응을 확인해봅시다

```
import ollama

# 아래와 같이 2개의 모델을 사용합니다
model_1 = "qwen:0.5b"
model_2 = "llama3.2:1b"

# model1 에게 먼저 대화를 시작합니다
message_from_model_1 = ""
message_from_model_2 = "Now, let's have an argument with anything. Any topic OK but one sentence per one turn. If you or I tell 'I lost' then I or you win"

# 대화를 10번 반복합니다
for i in range(10):
    # 모델1이 먼저 말합니다
    print(f"Model 2 ({model_2}): {message_from_model_2}")
    response_1 = ollama.chat(model=model_1, messages=[{"role": "user", "content": message_from_model_2}])
    message_from_model_1 = response_1['message']['content'] # Get the response from Model 1

    # 모델1의 응답을 받아 모델2가 말하고 이를 모델1에게 전달합니다
    print(f"Model 1 ({model_1}): {message_from_model_1}")
    response_2 = ollama.chat(model=model_2, messages=[{"role": "user", "content": message_from_model_1}])
    message_from_model_2 = response_2['message']['content'] # Get the response from Model 2
```

Model 2 (llama3.2:1b): Now, let's have an argument with anything. Any topic OK but one sentence per one turn. If you or I tell 'I lost' then I or you win

Model 1 (qwen:0.5b): You lose.

Model 2 (llama3.2:1b): That's okay, I didn't win anyway. Would you like to play again?

Model 1 (qwen:0.5b): Yes, please let's play again!

Model 2 (llama3.2:1b): I'd be happy to play another round with you. What kind of game would you like to play? We can try again with a different activity or keep going with the same one if you'd like.

Here are some options:

1. 20 Questions: I'll think of an object, and you try to guess what it is by asking me up to 20 yes-or-no questions.
2. Word Association: I'll give you a word, and you respond with a word that's associated with it.
3. Would You Rather: I'll give you two options, and you choose which one you'd prefer.
4. Trivia: I can provide you with trivia questions on a topic of your choice (e.g., history, science, sports, music, etc.).
5. Storytelling: I'll start telling a story, and then stop at a cliffhanger. You can then continue the story in your own words, and I'll respond with the next part of the story.

정리

- 프로그래밍언어를 이용하여 두 언어모델이 서로 대화하는 장면을 연출하였습니다
- 언어모델이 바뀌면 대화의 성향이 바뀌는 것을 확인하실 수 있을 것입니다
- 이러한 방식으로 오픈소스 언어모델을 사용하면 비용 부담 없이 우리의 프로그램과 언어모델을 결합한 시나리오를 구현할 수 있습니다