AI Application Specialist

대규모 언어모델 이해 및 파인튜닝 (PART-2)

2025.8.18-19

사내교수 이재원 / 첨단기술아카데미

SEU
Samsung Electronics University

Schedule

시간	1 일차	2 일차
08:30 ~ 09:20	• 과정 OT: 과정 소개 / AVD 소개	• LLM Fine-Tuning: PEFT (LoRA, …)
09:30 ~ 10:20	• [개요] Tokenizer	• [실습-4] PEFT/LoRA (40m)
10:30 ~ 11:20	• [실습-1] Text Tokenizer (30m)	• LLM Fine-Tuning: Preference Tuning (PPO, DPO)
11:30 ~ 12:20	• [개요] Word Embedding / Language Modeling	• [실습-5] Preference Tuning - DPO (40m)
12:30 ~ 13:30	중식	
13:30 ~ 14:20	• Transformers Library / DataSet	• LLM Evaluation
14:30 ~ 15:20	• [실습-2] Text Generation API (30m)	• [실습-6] LLM Evaluation (30m)
15:30 ~ 16:20	• LLM Fine-Tuning: SFT	• Transformer Architecture 및 최근 동향
16:30 ~ 17:20	• [실습-3] LLM Fine-Tuning - SFT (40m)	• 과정 정리 및 QA

CONTENTS

SEU
Samsung Electronics University

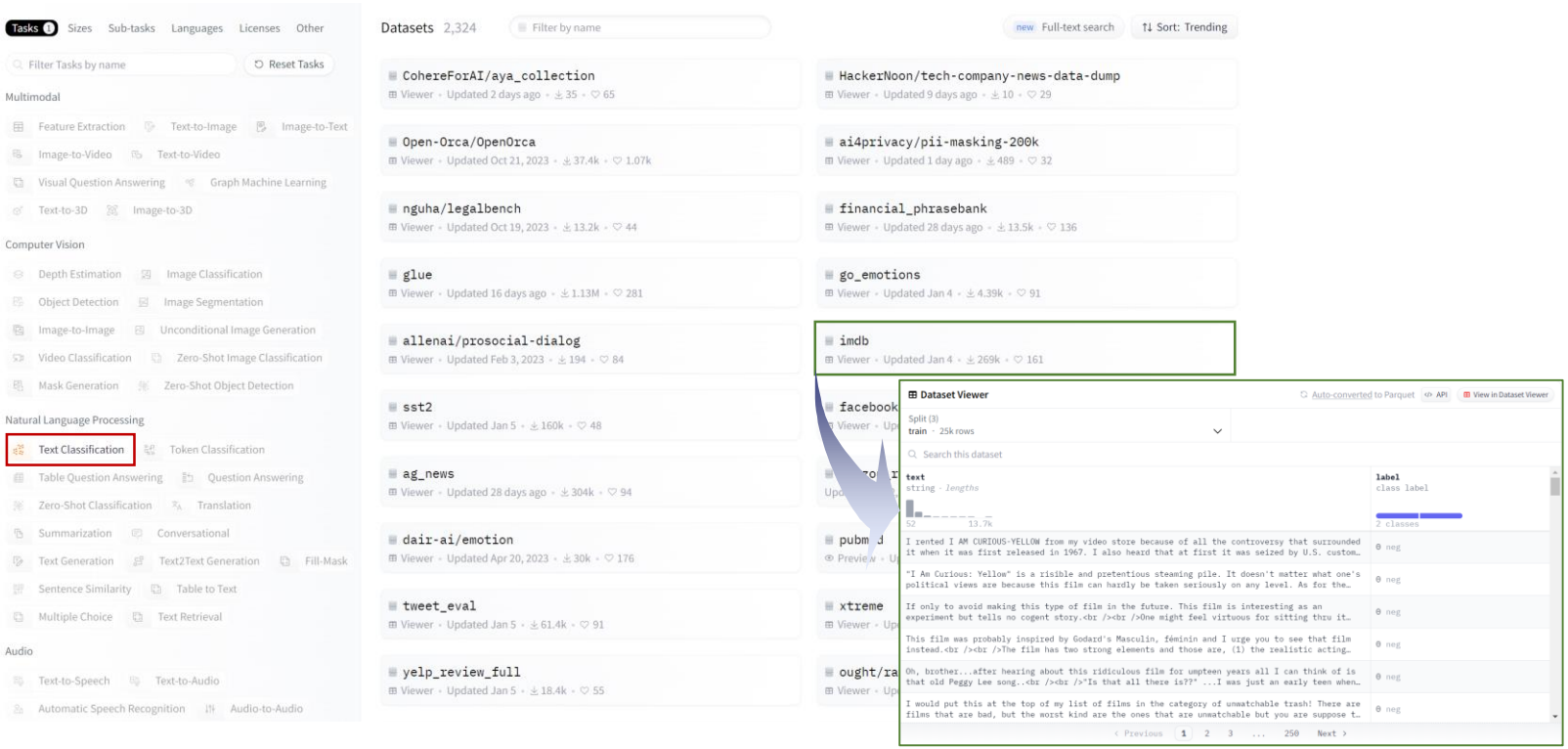
4.1 Datasets

트랜스포머 모델 훈련 파이프라인

- Load a dataset (train/validation/test dataset)
- Preprocess the data (tokenizing/resampling/augmentation)

4.1 Datasets

Hugging Face Hub



4.1 Datasets

Dataset Loading

- load_dataset()

```
from datasets import load_dataset

emotions = load_dataset("emotion")

DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 16000
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 2000
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 2000
    })
})
```

Data Preprocessing

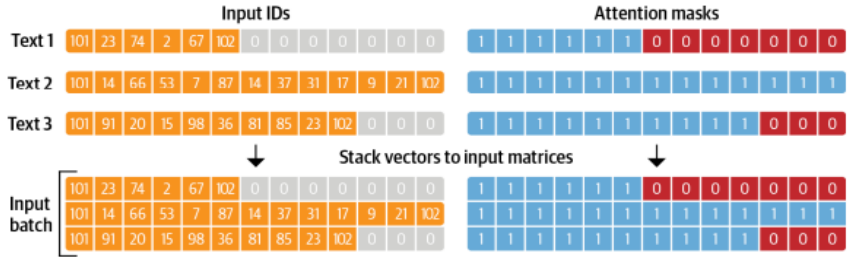
- Dataset.map(), Dataset.filter(), ···

```
from transformers import AutoTokenizer
from datasets import load_dataset

tokenizer = AutoTokenizer.from_pretrained(
                                "bert-base-uncased")
dataset = load_dataset("emotion", split="train")

def tokenize(batch):
    return tokenizer(batch["text"])

dataset = dataset.map(tokenize, batched=True)
```

4.1 Datasets

Data Preprocessing

- Padding, Truncation, Masking, …

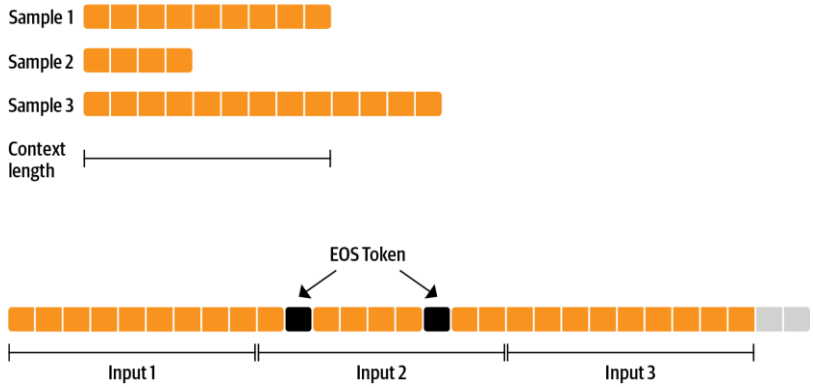


Special Token	[PAD]	[UNK]	[CLS]	[SEP]	[MASK]
Token ID	0	100	101	102	103

```
def tokenize(batch):
    return tokenizer(batch['text'],
                     padding=True,
                     truncation=True)
```

Data Loader

- Constant Length Dataset for Causal LM



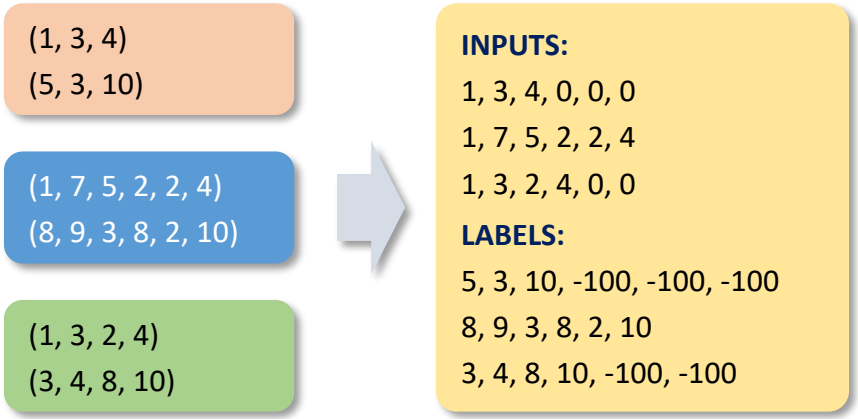
✓ Context Size 클 경우 (ex. 1024) 효율성 고려

✓ Truncation에 따른 편향

4.1 Datasets

Data Collator

- 데이터 전처리의 한 단계 (데이터 수집)
- 배치 구성할 때 필요한 작업을 자동으로처리
- 샘플을 배치로 묶는 과정에서 사용
- Padding, Token Masking, Labeling, Mini Batch

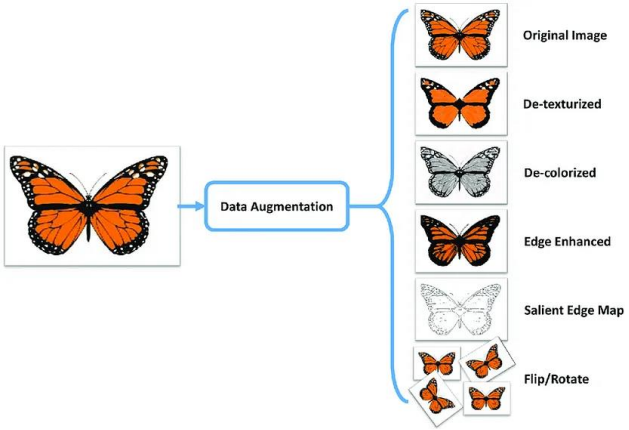



```python
# DefaultDataCollator
data_collator = DefaultDataCollator()

# DataCollatorWithPadding
data_collator = DataCollatorWithPadding(
    tokenizer=tokenizer
)

# DataCollatorForLanguageModeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer
    mlm=True,
    mlm_probability=0.15
)

# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    data_collator=data_collator,
)
```

Ref. https://huggingface.co/docs/transformers/main_classes/data_collator

4.2 Data Augmentation

- **Benefits of Data Augmentation**
  - Implicit Regularization
  - Semi-supervised applications, insufficient data
  - Cost effective way to data gathering and labeling
- **Data augmentation is [semantically invariant transformation]**



time flies like an arrow

↓ shuffle

an arrow like flies time?

Ref. "A Visual Survey of Data Augmentation in NLP"

4.2 Data Augmentation

Lexical Substitution

- **Thesaurus Based Substitution**
  : Look up the synonyms (ex. WordNet)

It is awesome → WordNet → It is amazing

- **Word Embeddings Substitution**
  : Nearest neighbors in the embedding space
  : Word2Vec, FastText, GloVe, etc.

- **Masked Language Model**
  : Predict masked words based on the context
  : BERT, ROBERTA, ALBERT, etc.

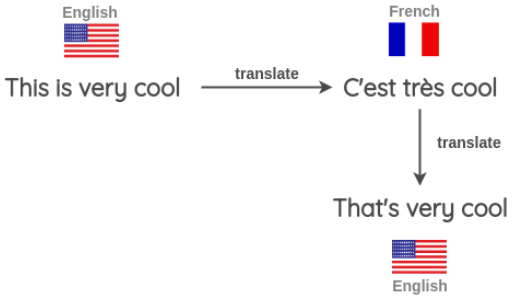
This is [mask] cool → MLM →
This is pretty cool
This is really cool
This is super cool
This is kinda cool
This is very cool

Ref. "A Visual Survey of Data Augmentation in NLP"

4.2 Data Augmentation

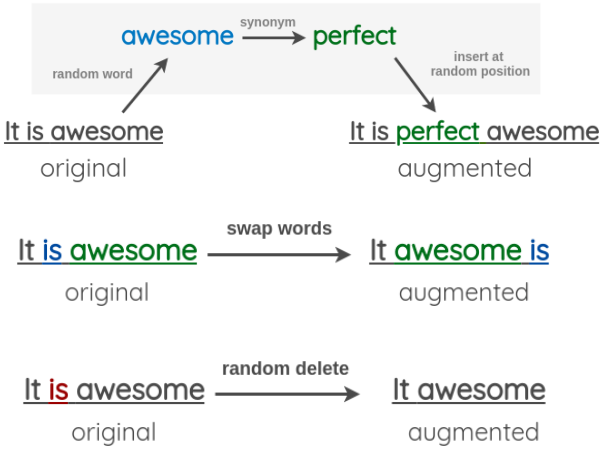
Back Translation

- Take some sentence (e.g. in English) and translate to another Language e.g. French
- Translate the French sentence back into an English sentence (Back Translation)
- Check if the new sentence is different from the original sentence. If it is, then we use this new sentence as an augmented data



Random Noise Injection

- Spelling Error Injection
- Keyboard Error Injection
- Unigram Noising
- Random Injection / Swap /Deletion



Ref. "A Visual Survey of Data Augmentation in NLP"

4.2 Data Augmentation

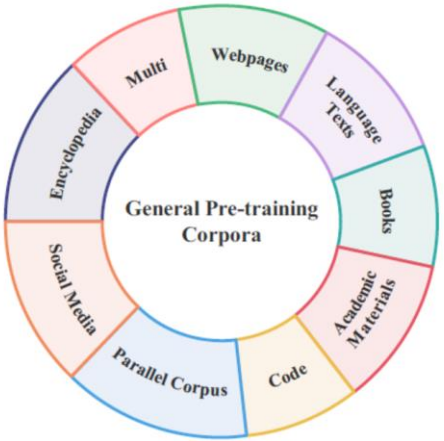
Example: ContextualWordEmbsAug

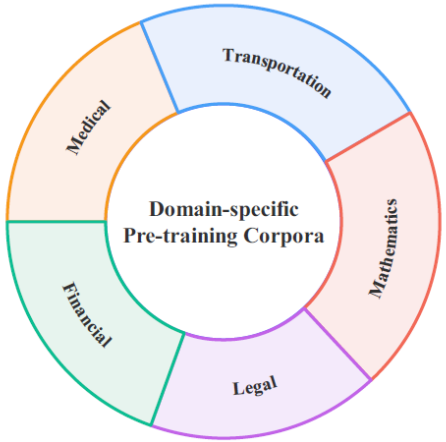
```
from transformers import set_seed
import nlpaug.augmenter.word as naw

set_seed(3)
aug = naw.ContextualWordEmbsAug(model_path='distilbert-base-uncased',
                                device="cpu",
                                action="substitute")

def augment_text(batch, transformations_per_example=1):
    text_aug, label_ids = [], []
    for text, labels in zip(batch['text'], batch['label_ids']):
        text_aug += [text]
        label_ids += [labels]
        for _ in range(transformations_per_example):
            text_aug += aug.augment(text)
            label_ids += [labels]
    return {"text": text_aug, "label_ids": label_ids}


ds_train_sample = ds_train_sample.map(augment_text, batched=True, . . .)
```

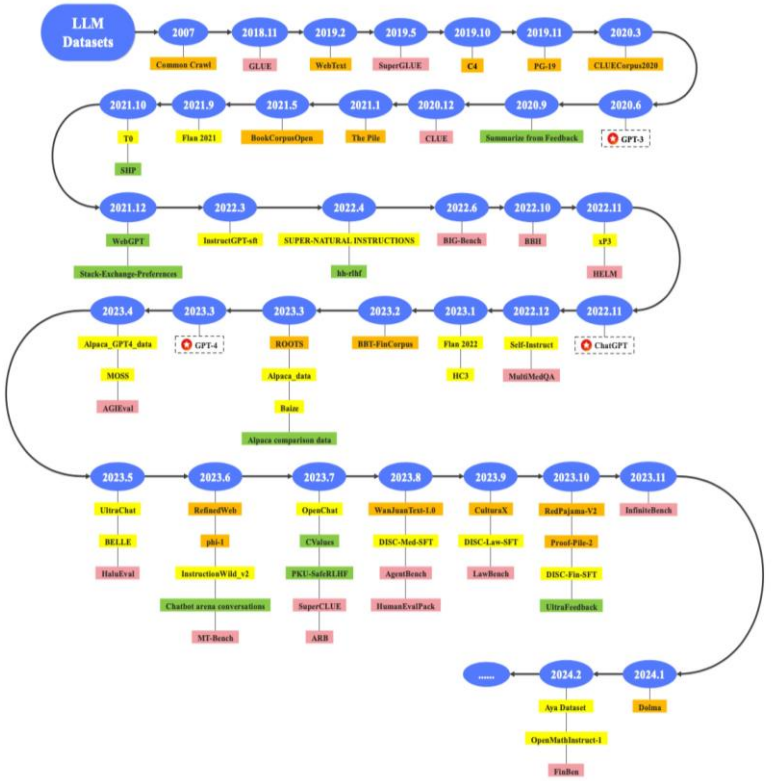
Ref. Natural Language Processing with Transformers

4.3 LLM Datasets

- A timeline of some representative LLM datasets
  - Pre-training corpora (orange)
  - Instruction fine-tuning datasets (yellow)
  - Preference (green)
  - Evaluation datasets (pink)







Ref. https://arxiv.org/pdf/2402.18041

4.3 LLM Datasets

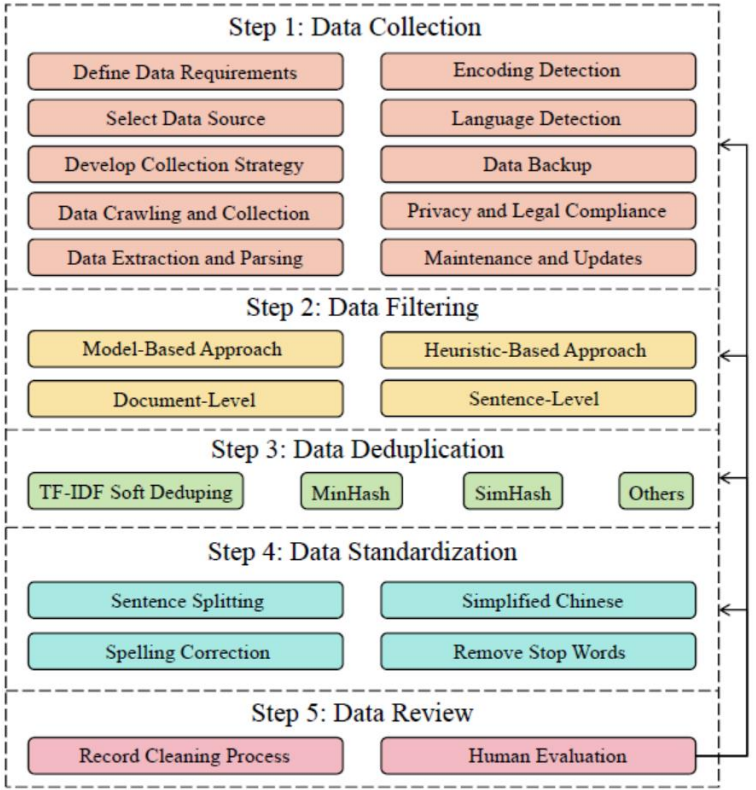
LLM 학습 주요 Data

- **Text Generation**
  : Common Crawl, WebText, WikiText, BookCorpus, etc.
- **Conversation**
  : MultiWOZ, ConvAI, Daily-Dialog, Persona-Chat, etc.
- **Question Answering**
  : SQuAD, NaturalQuestions, TriviaQA, etc.
- **Summarization**
  : CNN/Daily Mail, XSum, etc.
- **Paraphrasing**
  : PAWS, Quora Question Pairs, etc.
- **Code Generation**
  : CodeSearchNet, etc.

Korean NLP Datasets

4.3 LLM Datasets

- **Preprocessing for pre-training corpora**
  - Data Collection
  - Data Filtering
  - Data Duplication
  - Data Standardization
  - Data Review



Ref. https://arxiv.org/pdf/2402.18041

4.3 LLM Datasets: IF Template

Meta Llama 3

System prompt and multiple turn conversation between the user and assistant

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

{{ system_prompt }}<|eot_id|><|start_header_id|>user<|end_header_id|>

{{ user_message_1 }}<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>

{{ model_answer_1 }}<|eot_id|><|start_header_id|>user<|end_header_id|>

{{ user_message_2 }}<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Alpaca

We used the following prompts for fine-tuning the Alpaca model:

- for examples with a non-empty input field:

```
Below is an instruction that describes a task, paired with an input that provides further context. Write a

### Instruction:
{instruction}

### Input:
{input}

### Response:
```

- for examples with an empty input field:

```
Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:
{instruction}

### Response:
```

During inference (eg for the web demo), we use the user instruction with an empty input field (second option).

```
<|start_header_id|>system<|end_header_id|>
You are a helpful AI assistant.<|eot_id|><|start_header_id|>user<|end_header_id|>
Hi!<|eot_id|><|start_header_id|>assistant<|end_header_id|>
Hello there! It's great to meet you!<|eot_id|><|start_header_id|>user<|end_header_id|>
What are some recipes for making chocolate chip cookies?<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

요약

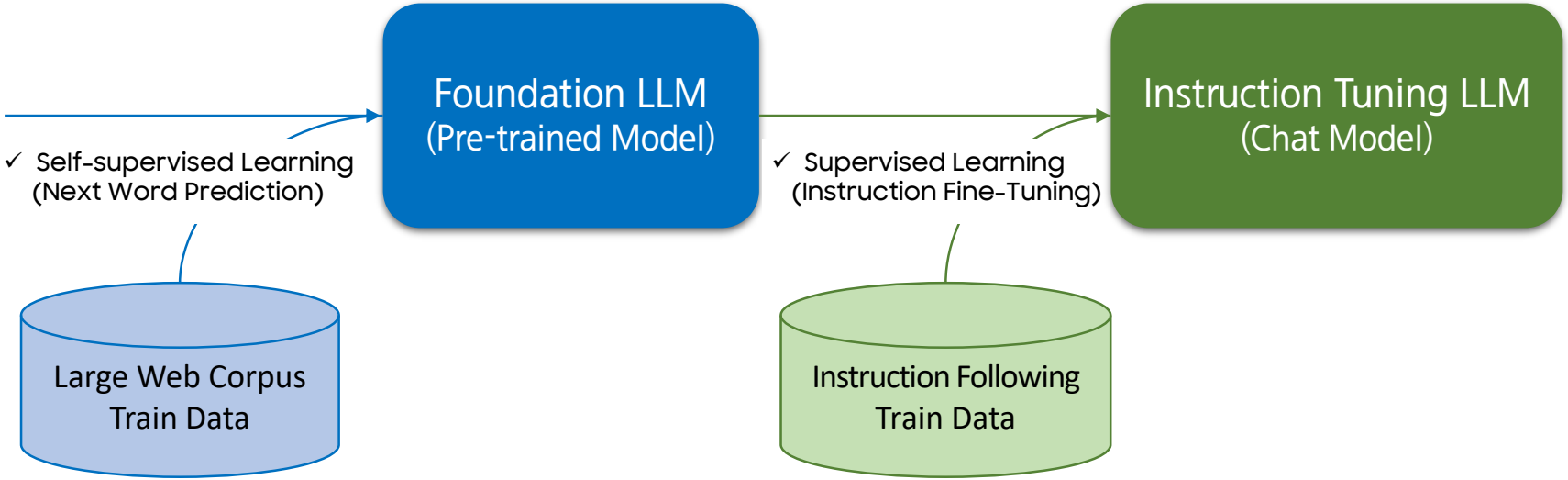
- Dataset (Train/Validation/Test)
- Dataset Loading, Preprocessing
- Padding, Truncation, Masking
- Text Data Augmentation
- LLM Datasets, Instruction Following Format

CONTENTS

SEU
Samsung Electronics University

5.0 LLM Training

- Foundation LLM (Pre-trained Model): 방대한 Generic Data 학습
- Instruction Fine-Tuning LLM: Prompt-Response Data 학습 Fine-Tuning



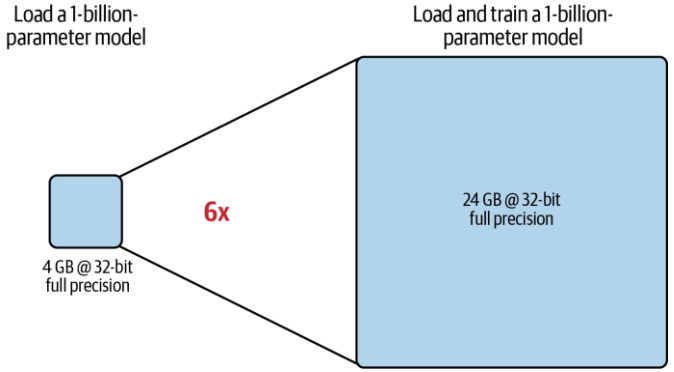
5.0 LLM Training

- 언어 모델 학습 (Training): Foundation Model Size, Training Data & Model Performance
  - 학습에 사용 가능한 GPU 스펙/자원 수, 모델 사이즈/학습 데이터 양에 따라 다양
  - 일반적으로 모델 사이즈가 클수록, 양질의 학습 데이터가 많을수록 모델 성능이 향상 (#Param x20 이상)
  - 학습: ① Pre-Training (지식) → ② Instruction Fine-Tuning (지시어) → ③ PEFT (도메인) / RLHF (선호도)

Additional GPU RAM needed to train 1B parameters

	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter
Adam optimizer (2 states)	+8 bytes per parameter
Gradients	+4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)
TOTAL	=4 bytes per parameter +20 extra bytes per parameter

~20 extra bytes per parameter

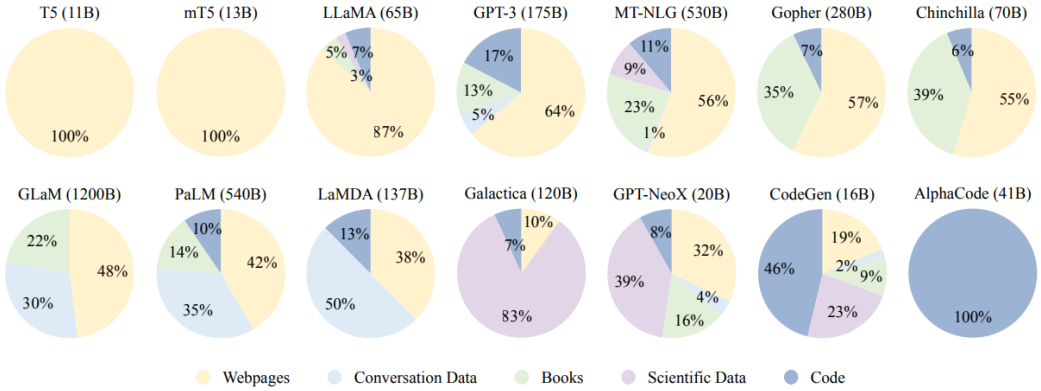


Ref. https://www.coursera.org/learn/generative-ai-with-llms

5.0 LLM Foundation Model: Pre-Training

① Foundation 모델 사전 학습 (Pre-Training, Self-Supervised Learning)

- 학습 데이터: 범용 공개 데이터 (라벨링 X) 및 지시어 데이터 (라벨링 ○)
- 데이터 구성: Webpages, Conversation Data, Books, Scientific Data, Code (범용 멀티 태스크 데이터)
- 데이터 전처리: 토큰화 (BPE 등)



Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

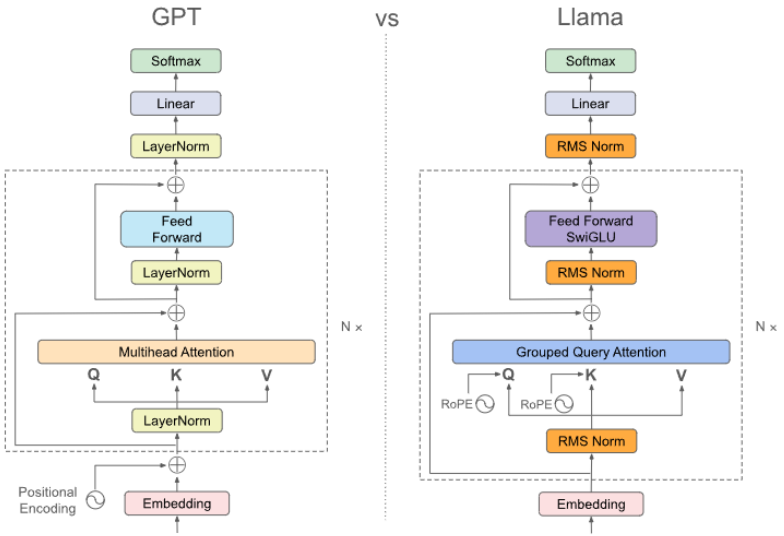
Vocabulary

Ref. Kaggle (A Survey Of Large Language Models)

5.0 [Open Source] Pretrained LLM: Llama

Llama Architecture

- **Decoder Architecture, RoPE, RMS Norm, SwiGLU Activation Function, Grouped Query Attention**



Llama2	Feature	Llama3
SentencePiece	Tokenizer	Tiktoken (OpenAI)
70B, 13B, 7B	Number of Parameters	8B, 70B, 405B
2.2T tokens	Training Data	**15T tokens**
4,096 tokens	Context Length	**8,192 tokens**
Grouped-query attention	Attention Mechanism	**Grouped-query attention**
Yes	Fine-Tuned Models	Yes
Better than Llama 1	Performance	Better than Llama 2
Very high	Computational Requirements	Very high
Open source	Availability	Open source
Yes	RLHF	Yes
20 languages	# of languages supported	**30 languages**

* RoPE (Rotary Position Embedding): 각 토큰의 위치를 상대적인 각도로 표현, SwiGLU (Switch + GLU)

5.0 [Open Source] Pretrained LLM: Llama

Meta Llama 3 (2024.4), Llama 3.2 (2024.9)

Llama-3 8B 모델 성능이
Llama-2 70B 모델보다 우수

Meta Llama 3 Instruct model performance

	Meta Llama 3 8B	Gemma 7B - It Measured	Mistral 7B Instruct Measured
MMLU 5-shot	**68.4**	53.3	58.4
GPQA 0-shot	**34.2**	21.4	26.3
HumanEval 0-shot	**62.2**	30.5	36.6
GSM-8K 8-shot, CoT	**79.6**	30.6	39.9
MATH 4-shot, CoT	**30.0**	12.2	11.0

	Meta Llama 3 70B	Gemini Pro 1.5 Published	Claude 3 Sonnet Published
MMLU 5-shot	**82.0**	81.9	79.0
GPQA 0-shot	39.5	**41.5** CoT	38.5 CoT
HumanEval 0-shot	**81.7**	71.9	73.0
GSM-8K 8-shot, CoT	**93.0**	91.7 11-shot	92.3 0-shot
MATH 4-shot, CoT	50.4	**58.5** Minerva prompt	40.5

[Llama 3]

- ✓ 2T Token → 15T Token
- ✓ 2K → 4K → 8K (Context)
- ✓ 8B, 70B, 405B Model
- ✓ Multi-lingual

[Llama 3.2]

- ✓ On-Device: 1B/3B
- ✓ Multi-modal: 11B/90B

[Llama 3.3]

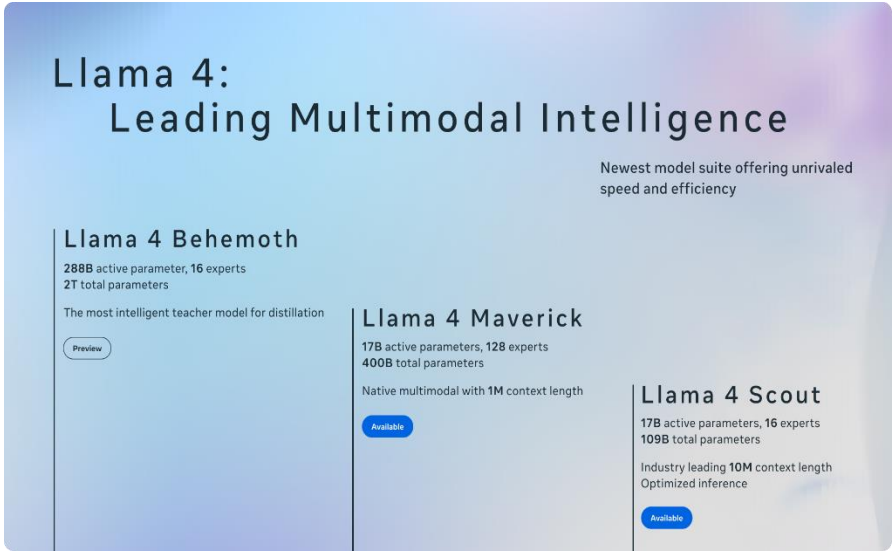
- ✓ 70B (성능: 3.1 405B 수준)

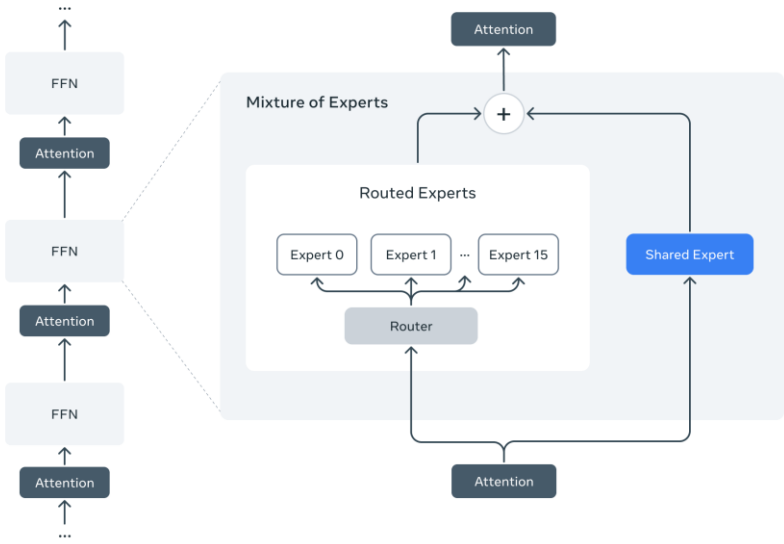
Ref. https://llama.meta.com/llama3/

5.0 [Open Source] Pretrained LLM: Llama

Meta Llama 4 (2025.4)

- Scout (109B): 단일 GPU 경량 모델, Maverick (400B): GPT-4o 능가, Behemoth (2T): 초대형 모델 개발 중
- 멀티모달 기본 탑재 (텍스트, 이미지, 비디오), 전문가 혼합 (MoE) 아키텍처, 10M 토큰 컨텍스트





Ref. https://ai.meta.com/blog/llama-4-multimodal-intelligence/

5.0 [Open Source] Pretrained LLM: Gemma

google/gemma-2 ('24.6)

- Lightweight, state-of-the-art open models
- Text-to-text, decoder-only LLM (English)
  : RMSNorm, RoPE, GEGLU (Gated GELU), MQA
- Instruction following
- Multi-turn conversations

Hyperparameter	2B	9B	27B
n_{layers}	26	42	46
d_{model}	2,304	3,584	4,608
d_{ff}	18,432	28,672	73,728
n_{heads}	8	16	32
$n_{key\_value\_heads}$	4	8	16
n_{ctx}	8,192	8,192	8,192
n_{vocab}	256,128	256,128	256,128

Ref. https://developers.googleblog.com/en/gemma-explained-new-in-gemma-2/

5.0 [Open Source] Pretrained LLM: Gemma

google/gemma-3 ('25.3)

- **Longer Context Length (8K → 32K, 128K)**
- **Multimodality (Image & Text)**
- **Multilinguality (English + 140 Languages)**

Both pre-trained and instruction tuned models are released. Gemma-3-4B-IT beats Gemma-2-27B IT, while Gemma-3-27B-IT beats Gemini 1.5-Pro across benchmarks.

	Gemma 2	Gemma 3
Model Size	• 2B • 9B • 27B	• 1B • 4B • 12B • 27B
Context Window Length	• 8K	• 32K (1B) • 128K (4B, 12B, 27B)
Multimodality (Image & Text)	• ☒	• ☒ (1B) • ☑ (4B, 12B, 27B)
Multilingual Support	-	• English (1B) + 140 Languages (4B, 12B, 27B)

Rank	Model	Elo	95% CI	Open	Type	#params/#activated
1	Grok-3-Preview-02-24	1412	+8/-10	-	-	-
1	GPT-4.5-Preview	1411	+11/-11	-	-	-
3	Gemini-2.0-Flash-Thinking-Exp-01-21	1384	+6/-5	-	-	-
3	Gemini-2.0-Pro-Exp-02-05	1380	+5/-6	-	-	-
3	ChatGPT-4o-latest (2025-01-29)	1377	+5/-4	-	-	-
6	DeepSeek-R1	1363	+8/-6	yes	MoE	671B/37B
6	Gemini-2.0-Flash-001	1357	+6/-5	-	-	-
7	o1-2024-12-17	1352	+4/-6	-	-	-
9	**Gemma-3-27B-IT**	**1338**	**+8/-9**	**yes**	**Dense**	**27B**
9	Qwen2.5-Max	1336	+7/-5	-	-	-
9	o1-preview	1335	+4/-3	-	-	-
9	o3-mini-high	1329	+8/-6	-	-	-
12	DeepSeek-V3	1318	+8/-6	yes	MoE	671B/37B
13	GLM-4-Plus-0111	1311	+8/-8	-	-	-
13	Qwen-Plus-0125	1310	+7/-5	-	-	-
13	Claude 3.7 Sonnet	1309	+9/-11	-	-	-
14	Gemini-2.0-Flash-Lite	1308	+5/-5	-	-	-
14	Step-2-16K-Exp	1305	+7/-6	-	-	-
14	o3-mini	1304	+5/-4	-	-	-
14	o1-mini	1304	+4/-3	-	-	-
14	Gemini-1.5-Pro-002	1302	+3/-3	-	-	-
...						
28	Meta-Llama-3.1-405B-Instruct-bf16	1269	+4/-3	yes	Dense	405B
...						
38	Llama-3.3-70B-Instruct	1257	+5/-3	yes	Dense	70B
...						
39	Qwen2.5-72B-Instruct	1257	+3/-3	yes	Dense	72B
...						
59	Gemma-2-27B-it	1220	+3/-2	yes	Dense	27B

Evaluation of Gemma 3 27B IT model in the Chatbot Arena (March 8, 2025)

Ref. https://huggingface.co/blog/gemma3

5.0 [Open Source] GPT, Gemma, Llama

- Architecture: 모두 디코더 구조 사용
- Attention Mechanism: 연산량을 줄이는 방향으로 MHA, MQA, GQA 사용
- Activation: Dying ReLU 문제를 해결하는 방향으로 GEGLU, SwiGLU 사용

DeepSeek-R1

Feature	GPT	Gemma 2	Llama 3
Architecture	Decoder	Decoder	Decoder
Positional Encoding	Sinusoidal PE	Rotary PE	Rotary PE
Normalization	Layer Norm	RMS Norm	RMS Norm
Attention Mechanism	Multi-Head Attention	Local Sliding Window, Multi Query Attention (MQA)	Grouped Multi-Query Attnetion (GQA)
Activation	ReLU	GEGLU	SwiGLU
Training Data	4.5M+36M	13T (27B Model)	15T

5.0 Pretrained LLM

상용 LLM v.s. 오픈소스 LLM

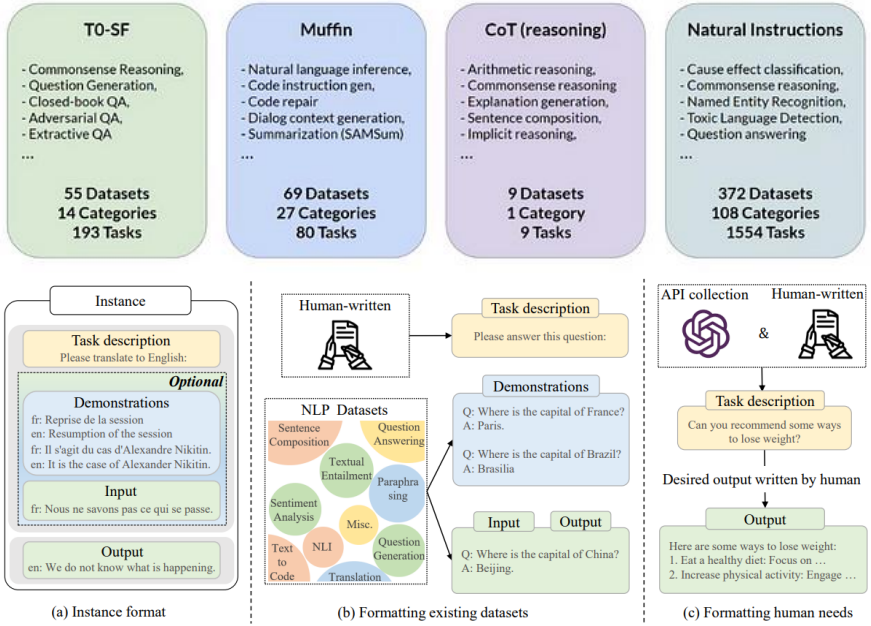
Benchmark Test (2024)

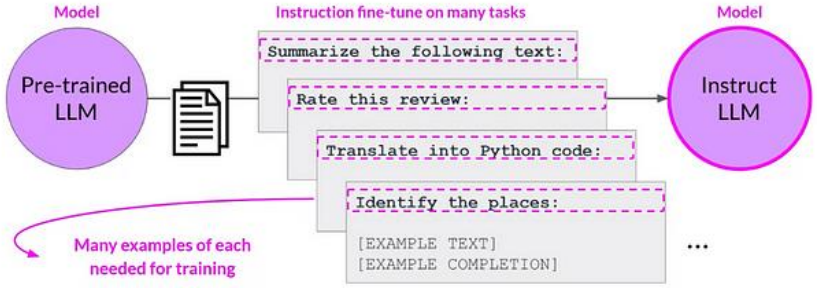
- 모델 크기 및 스펙 (Context Size, Max. Token, Computing Resource)
- 모델 성능: 벤치마크 성능, 지연시간, 모델 안정성 등
- 비용 및 호스팅: 모델 사용료, 모델 호스팅 비용 등
- Model Customization: LLM Fine-Tuning 가능 여부

5.1 LLM Fine-Tuning: SFT

② 사전 학습 Foundation 모델 기반 Instruction Following 학습

- Instruction Following Data: 휴먼 구축, 기존 멀티 태스크 데이터 활용, LLM 활용 데이터 생성
- Multi-turn Data: 인위적으로 명시적 프롬프트 기반 데이터 생성 후 조합





Instruct 모델 Supervised Fine Tuning

Ref. Instruction Fine Tuning of LLMs – A Comprehensive Guide

5.1 LLM Fine-Tuning: SFT

Instruction Fine-Tuning

In the loss function, mask out the tokens corresonding to prompt

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
Rewrite the following sentence using passive voice.

### Input:
The team achieved great results.

### Response:
Great results were achieved by the team.
```

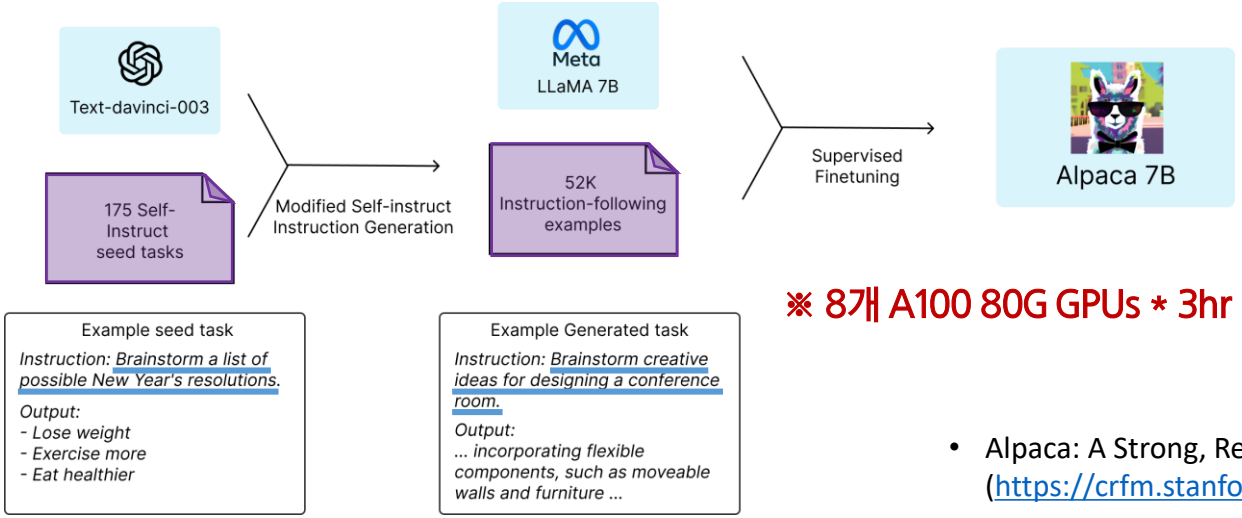
Calculate the loss only over the tokens corresponding to the response text

5.1 LLM Fine-Tuning: Stanford Alpaca

An Instruction-following LlaMA Model

- Fine-tuned from a 7B LlaMA 1 model
- On 52K instruction following data generated by the Self-Instruct
- Similar to the text-davinci-003 model on the Self-Instruct instruction-following evaluation suite

Stanford Alpaca



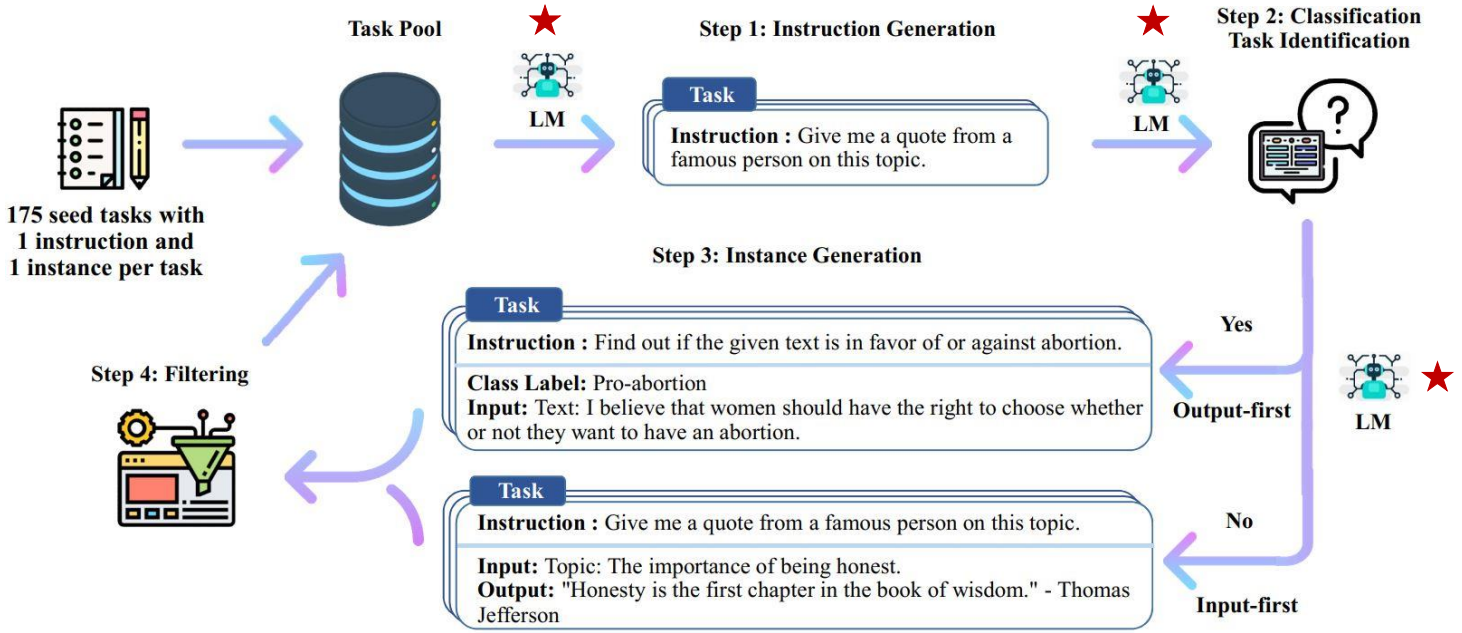
※ 8개 A100 80G GPUs * 3hr

Hyperparameter	LLaMA-7B
Batch size	128
Learning rate	2e-5
Epochs	3
Max length	512
Weight decay	0

- Alpaca: A Strong, Replicable Instruction-Following Model (https://crfm.stanford.edu/2023/03/13/alpaca.html)

5.1 LLM Fine-Tuning: Stanford Alpaca

Data Generation: High Level Overview of Self-Instruct



Ref. Self-Instruct: Aligning Language Model with Self Generated Instructions. Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, Hannaneh Hajishirzi. https://arxiv.org/abs/2212.10560

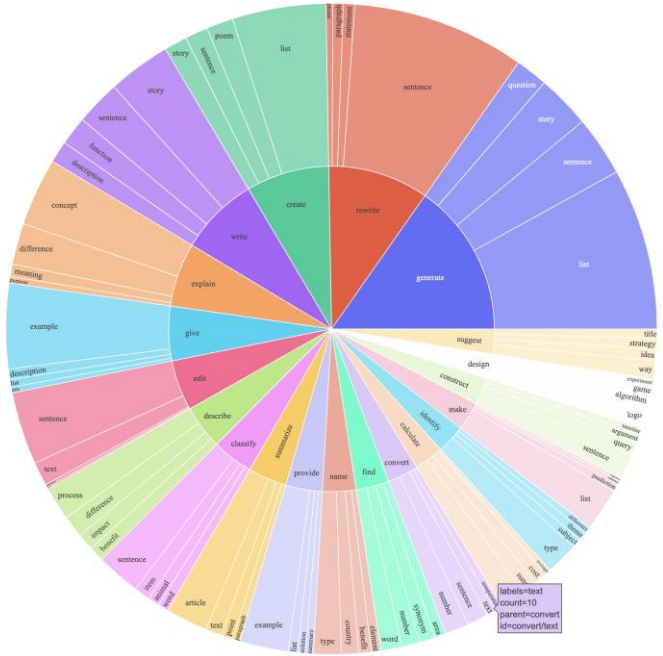
5.1 LLM Fine-Tuning: Stanford Alpaca

Instruction-Following Data

- **Instruction**: describes the task the model should perform. Each of the 52K instructions is unique.
- **Input**: optional context or input for the task. For example, when the instruction is "Summarize the following article", the input is the article. Around 40% of the examples have an input.
- **Output**: the answer to the instruction as generated by text-davinci-003

```
### Instruction:
{instruction}

### Input:
{input}

### Response:
```

The top 20 most common root verbs and their top 4 direct noun objects



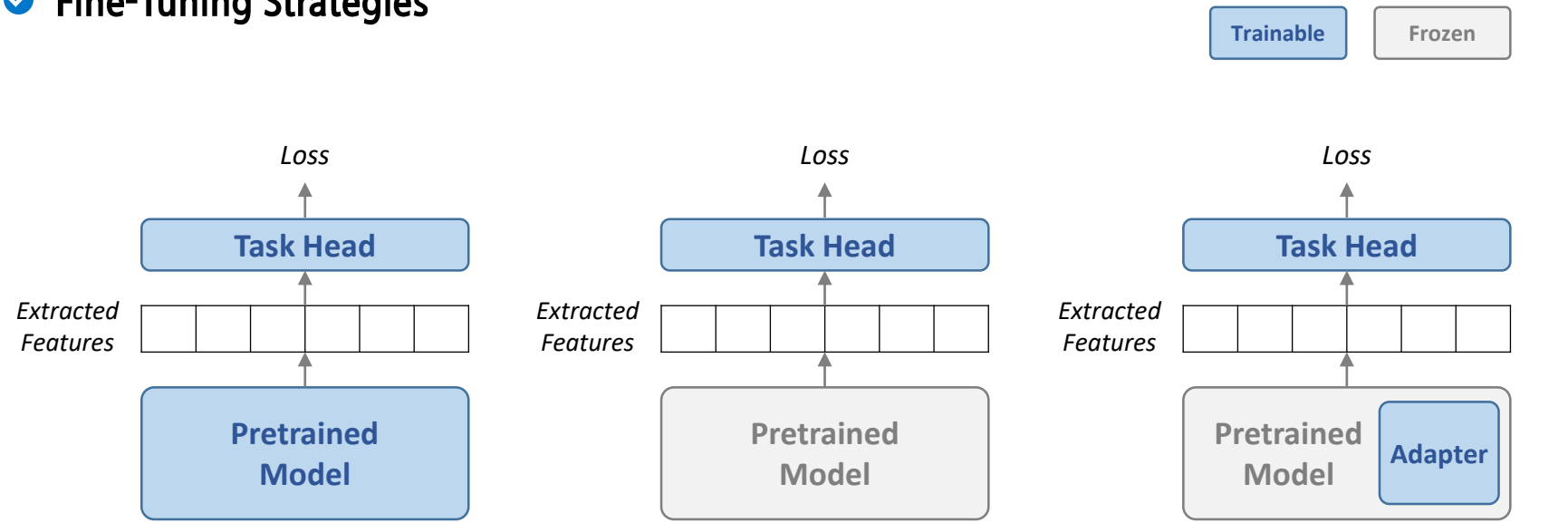
5.1 LLM Fine-Tuning

- **Training Framework**
  - HuggingFace Trainer (Transformers)
  - Microsoft/DeepSpeed
  - NVIDIA/Nemo Megatron
  - Colossal-AI, Axolotl, Llama-Factory

```
[ai-application-specialist-llm-new]
✓ transformers==4.51.1
✓ datasets==3.2.0
✓ peft==0.15.1
✓ trl==0.16.1
✓ accelerate==1.3.0
```

5.1 LLM Fine-Tuning

Fine-Tuning Strategies



(a) Full Fine-Tuning

(b) Head Only Tuning

(c) Parameter Efficient Tuning

[실습] SFT (Supervised Fine-tuning) (1/4)

[실습 과제] Fine-Tuning a Pre-Trained Model !!

(aas_sft_sentiment_classification.ipynb)

- Task: Sentiment Classification
- Dataset: Emotion (6 Classes)
- Pretrained Model: "distilbert-base-uncased"
- Trainer: Transformers Trainer

Ref. https://huggingface.co/docs/transformers/training

[실습] SFT (Supervised Fine-tuning) (2/4)

TraingArguments 설정

- 학습 에포크
- 학습/평가 시 미니 배치 사이즈
- 최적화 함수

Trainer Parameter

```
from transformers import TrainingArguments

batch_size = 64
logging_steps = len(emotions_encoded["train"]) // batch_size

training_args_fine_tuning = TrainingArguments(
    output_dir='./SFT-Trainer',               # output directory
    num_train_epochs=5,                       # total number of training epochs
    per_device_train_batch_size=batch_size,   # batch size per device during training
    per_device_eval_batch_size=batch_size,    # batch size for evaluation
    optim="adamw_torch",                      # optimizer: adamw_hf, adamw_torch, adamw_apex_fused, or adafactor
    learning_rate=2e-5,                       # learning rate
    logging_steps=logging_steps,              # logging steps
    save_strategy="epoch",                    # save strategy
)
```

Ref. https://huggingface.co/docs/transformers/v4.19.2/en/main_classes/trainer#transformers.TrainingArguments

[실습] SFT (Supervised Fine-tuning) (3/4)

Compute Metrics

- Evaluation에 사용할 Metric을 계산하는 함수
- Model output (EvalPrediction 객체)을 입력으로 받아 Metric 을 Dictionary 형태로 반환
- Trainer의 compute_metrics 항목에 반영

```python
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    f1 = f1_score(labels, preds, average="weighted")
    acc = accuracy_score(labels, preds)
    return {"accuracy": acc, "f1": f1}
```

※ EvalPrediction 객체는 predictions와 label_ids 속성을 가진 Named Tuple을 입력받아 측정지표 이름과 값을 매핑한 Dictionary 반환

Ref. NLP Metrics (https://github.com/huggingface/evaluate/tree/main/metrics)

[실습] SFT (Supervised Fine-tuning) (4/4)

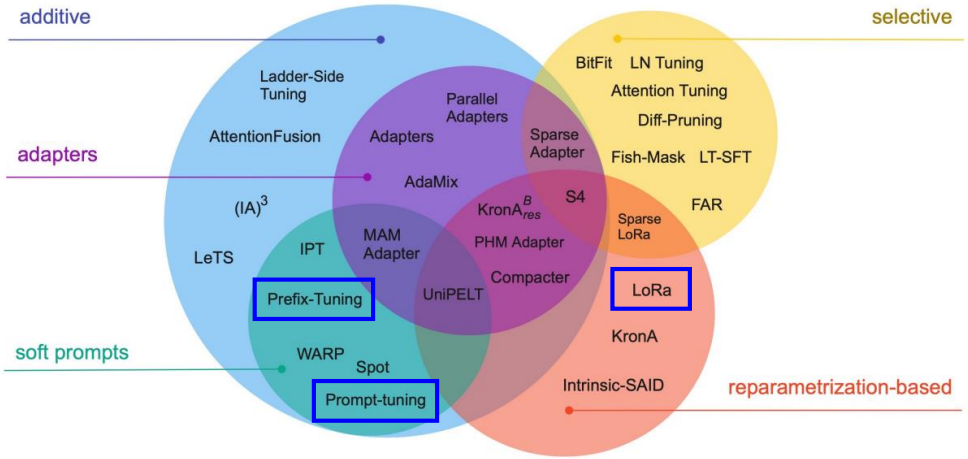
Trainer

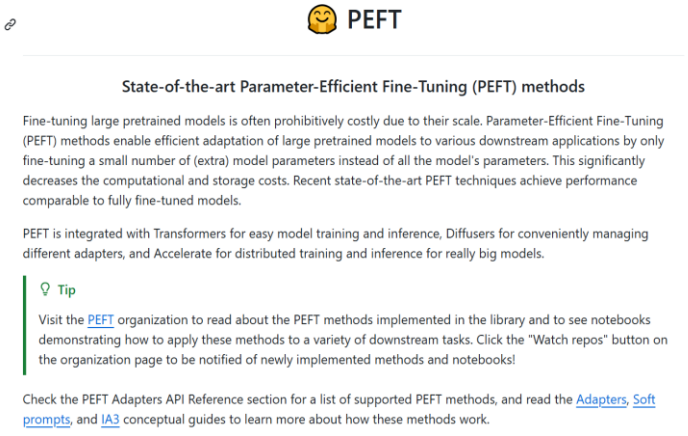
```
# Supervised Fine-tuning

from transformers import AutoModelForSequenceClassification
from transformers import Trainer

num_labels = 6
model = AutoModelForSequenceClassification.from_pretrained(model_ckpt, num_labels=num_labels)

trainer = Trainer(model=model,                                  # The model to train, evaluate
                  args=training_args,                           # TraingArguments Class
                  train_dataset=emotions_encoded["train"],      # The dataset for training
                  eval_dataset=emotions_encoded["validation"],  # The dataset for evaluation
                  compute_metrics=compute_metrics,              # Metrics at evaluation
                  processing_class=tokenizer,                   # Tokenizer
)

trainer.train()

preds_output = trainer.predict(emotions_encoded[“test"])
preds_output.metrics
```

5.2 LLM Fine-Tuning: PEFT

③ PEFT(Parameter Efficient Fine-Tuning)

- 학습 단계: Pretraining (지식) → Instruction Tuning (지시어) → PEFT (도메인) / RLHF (선호도)
- PEFT: Pre-trained Model Weights 고정, 학습 가능한 소수의 (추가적인) 모델 파라미터만을 미세 조정
- 학습 데이터: 도메인 적응 학습 위한 타겟 서비스 도메인 데이터





PEFT

State-of-the-art Parameter-Efficient Fine-Tuning (PEFT) methods

Fine-tuning large pretrained models is often prohibitively costly due to their scale. Parameter-Efficient Fine-Tuning (PEFT) methods enable efficient adaptation of large pretrained models to various downstream applications by only fine-tuning a small number of (extra) model parameters instead of all the model's parameters. This significantly decreases the computational and storage costs. Recent state-of-the-art PEFT techniques achieve performance comparable to fully fine-tuned models.

PEFT is integrated with Transformers for easy model training and inference, Diffusers for conveniently managing different adapters, and Accelerate for distributed training and inference for really big models.

Tip

Visit the PEFT organization to read about the PEFT methods implemented in the library and to see notebooks demonstrating how to apply these methods to a variety of downstream tasks. Click the "Watch repos" button on the organization page to be notified of newly implemented methods and notebooks!

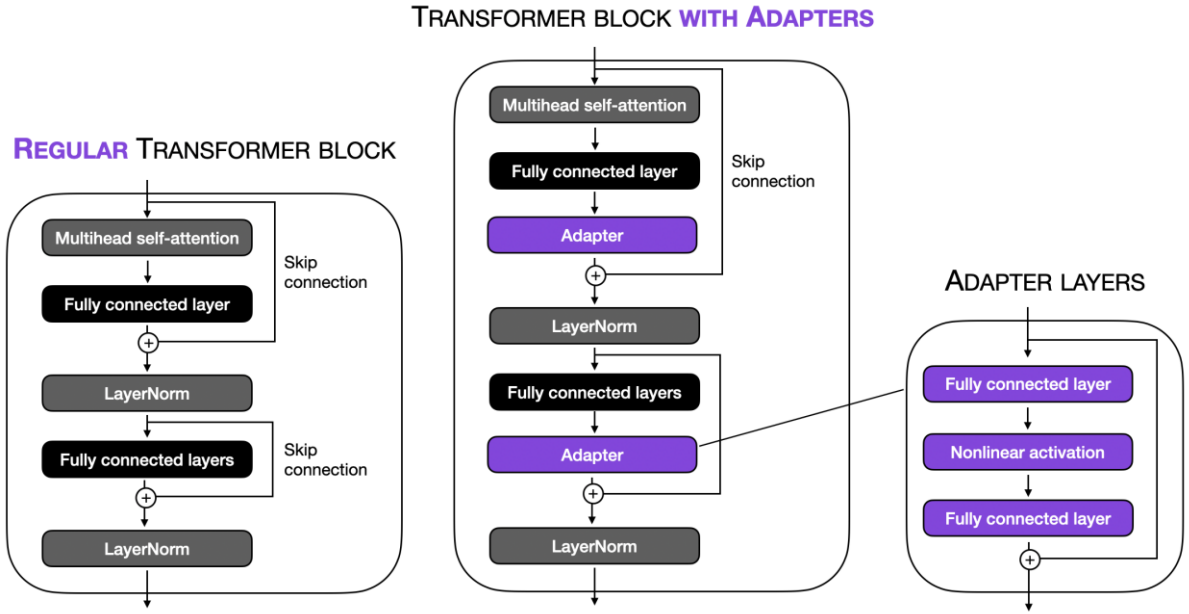
Check the PEFT Adapters API Reference section for a list of supported PEFT methods, and read the Adapters, Soft prompts, and IA3 conceptual guides to learn more about how these methods work.

Ref. https://huggingface.co/docs/peft/index

5.2 LLM Fine-Tuning: PEFT

Adapter: adds adapter layers in two places

※ Base Model 의존성이 높아 Base Model 이 업데이트 되면 Adapter 재학습 필요



Ref. Understanding Parameter-Efficient Finetuning of Large Language Models: From Prefix Tuning to LLaMA-Adapters

5.2 LLM Fine-Tuning: PEFT / LoRA

- [Microsoft] LoRA: Low-Rank Adaptation of Large Language Models (2021)
- https://arxiv.org/abs/2106.09685

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

- # of Parameters: 1/10,000
- GPU Memory: 1/3
- No additional inference latency

Abstract

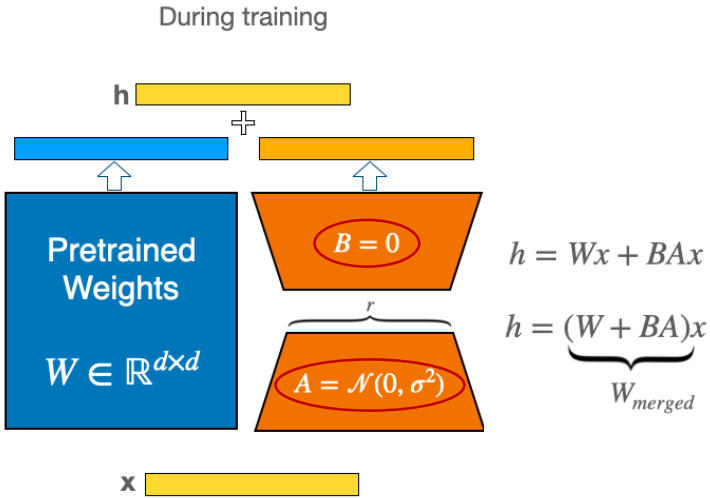
An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose **Lo**w-**R**ank **A**daptation, or LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fine-

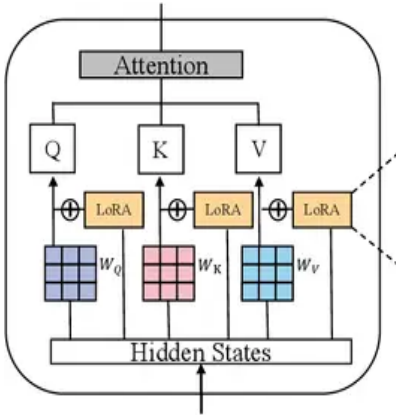
5.2 LLM Fine-Tuning: PEFT / LoRA

LoRA: Low-Rank Adaptation of Large Language Models

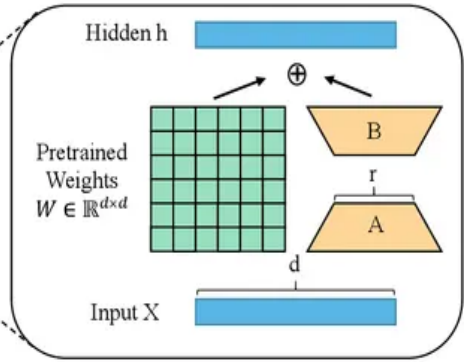
- 고정된 Weights를 갖는 Pretrained Model에 학습이 가능한 Rank Decomposition Matrix 삽입
- 적은 수의 GPU로 빠르게 튜닝할 수 있다는 장점

* 행렬의 차원을 r 만큼 줄이는 행렬과 다시 원래 크기로 키워주는 행렬의 곱으로 나타내는 것을 의미





Multi-Head Attention



LoRA

Ref. LoRA: Low-Rank Adaptation of Large Language Models. (https://arxiv.org/abs/2106.09685)

5.2 LLM Fine-Tuning: PEFT / LoRA

QLoRA (2023)

- Efficient Finetuning of Quantized LLMs
- Q+LoRA = 4-bit Quantized Language Model into Low Rank Adapters
- 4-bit NormalFlot, Double Quantization, Paged Optimization

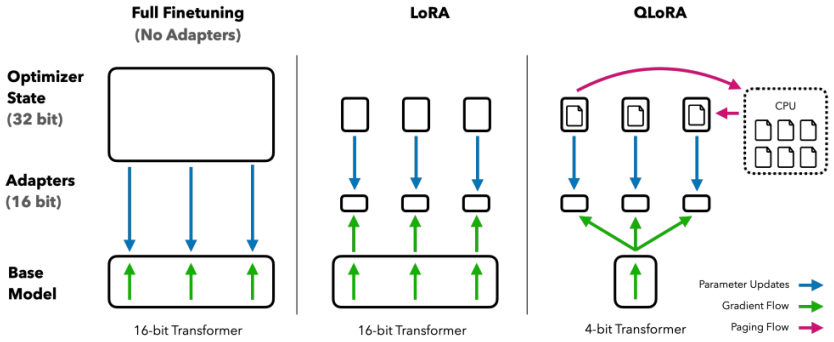


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

S-LoRA (2024)

- Serving Thousands of Concurrent LoRA Adapters

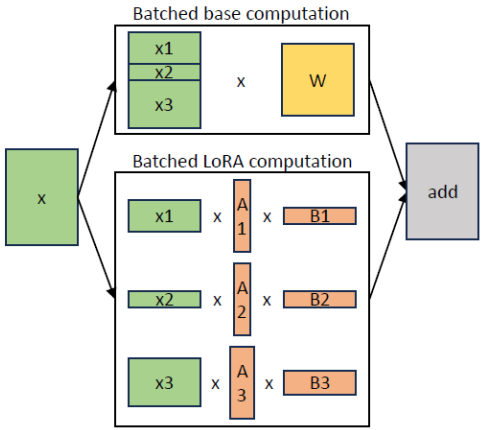


Figure 1. Separated batched computation for the base model and LoRA computation. The batched computation of the base model is implemented by GEMM. The batched computation for LoRA adapters is implemented by custom CUDA kernels which support batching various sequence lengths and adapter ranks.

5.2 LLM Fine-Tuning: PEFT / Prefix-Tuning

- [Stanford] Prefix-Tuning: Optimizing Continuous Prompts for Generation (2021)
- https://arxiv.org/abs/2101.00190

Prefix-Tuning: Optimizing Continuous Prompts for Generation

Xiang Lisa Li
Stanford University
xlisali@stanford.edu

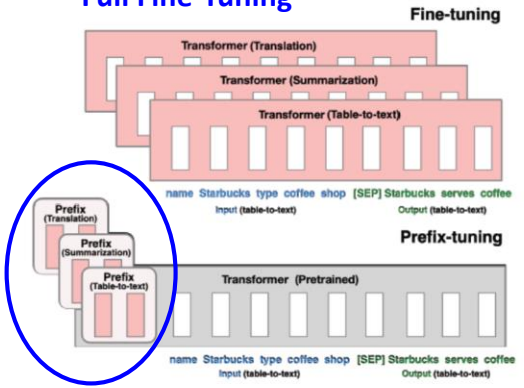
Percy Liang
Stanford University
pliang@cs.stanford.edu

- Inspired from Prompting
- 1,000x fewer parameters
- comparable performance

Abstract

Fine-tuning is the de facto way to leverage large pretrained language models to perform downstream tasks. However, it modifies all the language model parameters and therefore necessitates storing a full copy for each task. In this paper, we propose prefix-tuning, a lightweight alternative to fine-tuning for natural language generation tasks, which keeps language model parameters frozen, but optimizes a small *continuous task-specific* vector (called the prefix). Prefix-tuning draws inspiration from prompting, allowing subsequent to-

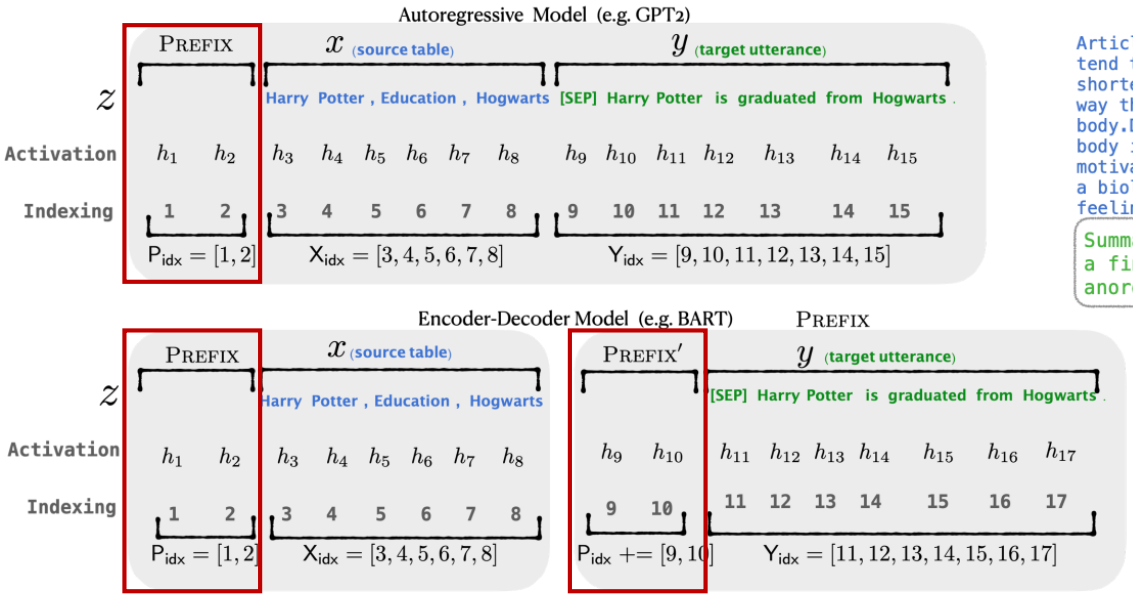
Full Fine-Tuning



Prefix-Tuning (Layer-wise Prompt)

5.2 LLM Fine-Tuning: PEFT / Prefix-Tuning

- [Example] Prefix-Tuning using an autoregressive LM and an encoder-decoder model



Summarization Example

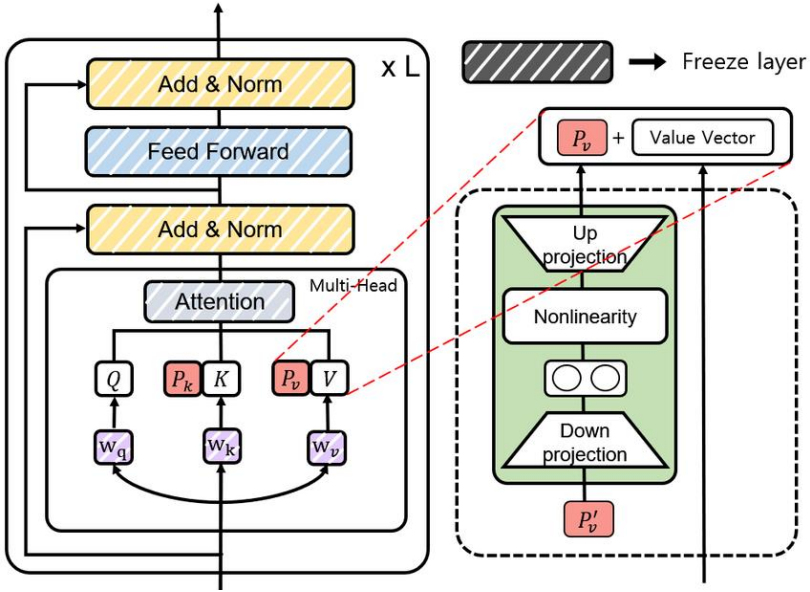
```
Article: Scientists at University College London discovered people
tend to think that their hands are wider and their fingers are
shorter than they truly are.They say the confusion may lie in the
way the brain receives information from different parts of the
body.Distorted perception may dominate in some people, leading to
body image problems ... [ignoring 308 words] could be very
motivating for people with eating disorders to know that there was
a biological explanation for their experiences, rather than
feeling it was their fault."

Summary: The brain naturally distorts body image -
a finding which could explain eating disorders like
anorexia, say experts.
```

Table-to-text Example

```
Table:  name[Clowns] customer-
rating[1 out of 5]  eatType[coffee
shop] food[Chinese] area[riverside]
near[Clare Hall]

Textual Description: Clowns is a
coffee shop in the riverside area
near Clare Hall that has a rating
1 out of 5 . They serve Chinese
food .
```

5.2 LLM Fine-Tuning: PEFT / Prefix-Tuning

Parameterization of P_θ



Evaluation (Table-To-Text Generation)

	E2E BLEU	NIST	MET	R-L	CIDEr	
FINE-TUNE	68.2	8.62	**46.2**	71.0	2.47	GPT-2 MEDIUM
FT-TOP2	68.1	8.59	46.0	70.8	2.41	
ADAPTER(3%)	68.9	8.71	46.1	71.3	2.47	
ADAPTER(0.1%)	66.3	8.41	45.0	69.8	2.40	
PREFIX(0.1%)	**69.7**	**8.81**	46.1	**71.4**	**2.49**	
FINE-TUNE	68.5	8.78	46.0	69.9	2.45	GPT-2 LARGE
Prefix	**70.3**	**8.85**	**46.2**	**71.7**	**2.47**	
SOTA	68.6	8.70	45.3	70.8	2.37	

※ 매개변수를 직접 업데이트하는 방법보다는 MLP로 모델링할 경우 보다 나은 성능을 확보

Ref. Parameter-Efficient Fine-Tuning Method for Task-Oriented Dialogue Systems

5.2 LLM Fine-Tuning: PEFT / Prompt Tuning

- [Google] The Power of Scale for Parameter-Efficient Prompt Tuning (2021)
- https://arxiv.org/abs/2104.08691

- Prefix-Tuning: All Transformer Layer → Input Embedding
- 0.01% Task-specific Parameters

The Power of Scale for Parameter-Efficient Prompt Tuning

Brian Lester* **Rami Al-Rfou** **Noah Constant**
Google Research
{brianlester,rmyeid,nconstant}@google.com

Abstract

In this work, we explore "prompt tuning," a simple yet effective mechanism for learning "soft prompts" to condition frozen language models to perform specific downstream tasks. Unlike the discrete text prompts used by GPT-3, soft prompts are learned through backpropagation and can be tuned to incorporate signals from any number of labeled examples. Our end-to-end learned approach outperforms GPT-3's few-shot learning by a large margin. More remarkably, through ablations on model size using T5, we show that prompt tuning becomes more competitive with scale: as models exceed billions of parameters, our method "closes the gap" and matches the strong performance of model tuning (where all model weights are tuned). This finding is especially relevant because large models are costly to share and serve and the ability to reuse one frozen model for multiple downstream tasks can ease this burden. Our method can be seen as a simplification of the recently proposed

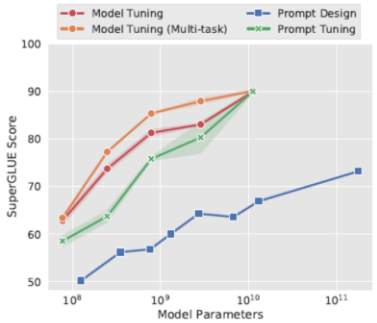
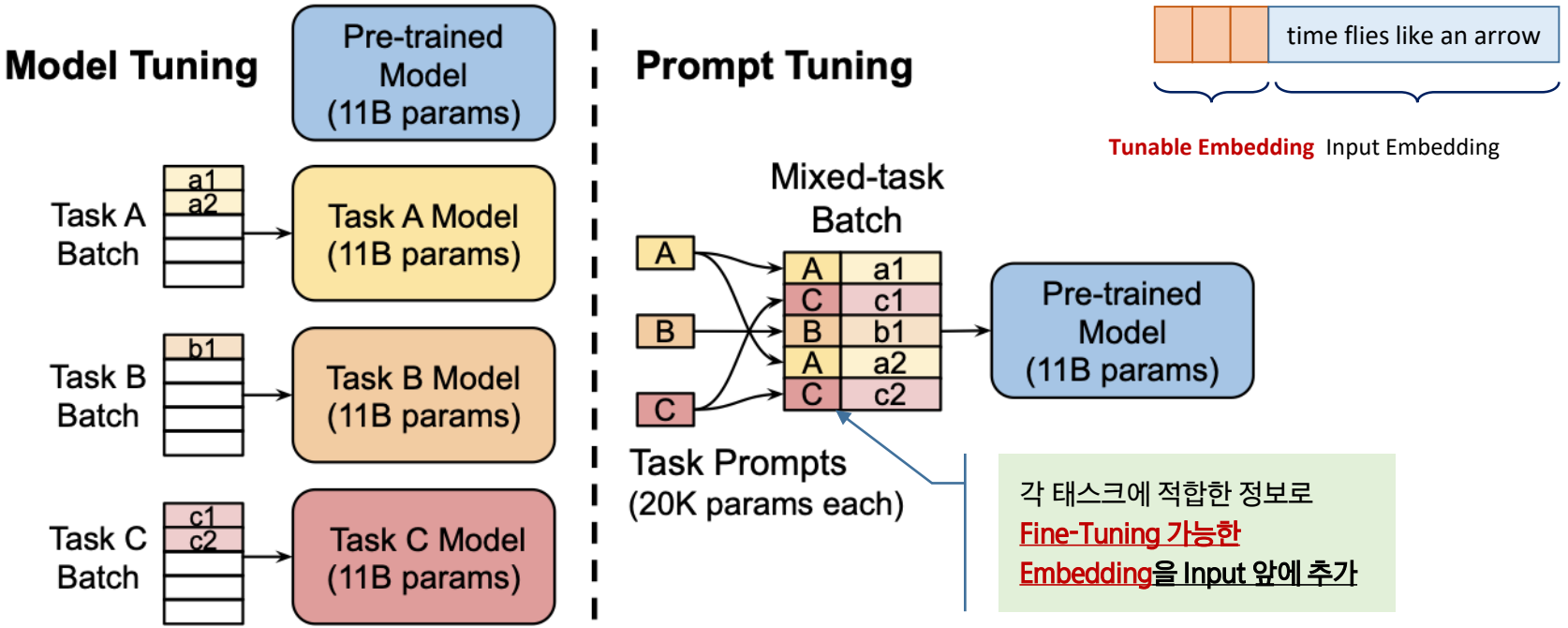


Figure 1: Standard **model tuning** of T5 achieves strong performance, but requires storing separate copies of the model for each end task. Our **prompt tuning** of T5 matches the quality of model tuning as size increases, while enabling the reuse of a single frozen model for all tasks. Our approach significantly outperforms few-shot **prompt design** using GPT-3. We show mean and standard deviation across 3 runs for tuning methods.

5.2 LLM Fine-Tuning: PEFT / Prompt Tuning

Model Tuning and Prompt Tuning



※ P-Tuning에서는 LSTM 기반으로 파라미터 업데이트가 되는 구조이나,
Prompt-Tuning에서는 직접적으로 학습된다

5.2 LLM Fine-Tuning: PEFT / Prompt Tuning

Evaluation

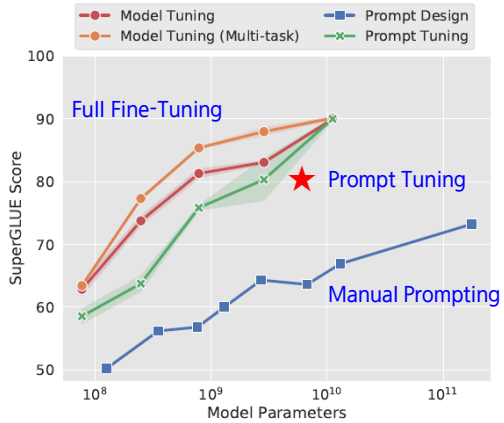
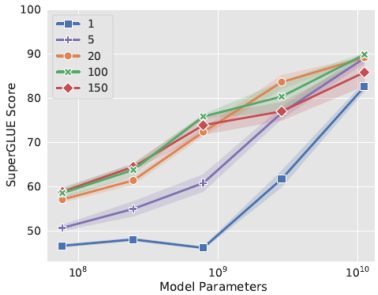


Figure 1: Standard **model tuning** of T5 achieves strong performance, but requires storing separate copies of the model for each end task. Our **prompt tuning** of T5 matches the quality of model tuning as size increases, while enabling the reuse of a single frozen model for all tasks. Our approach significantly outperforms few-shot **prompt design** using GPT-3. We show mean and standard deviation across 3 runs for tuning methods.

Design Decisions

- Prompt Length
- Prompt Initialization



(a) Prompt length

(b) Prompt initialization

- Prompt Length가 20개, 100개만 되어도 150개 보다 좋은 결과 (5~100 적절)
- Random 보다는 Text Initialization 성능이 우수

[실습] LLM Fine-Tuning: PEFT / LoRA (1/4)

[실습 과제] PEFT LoRA 기반 LLM 파인튜닝!!

(aas_gemma-2b+LoRA.ipynb)

- Task: Causal Language Model (Instruct Fine-Tuning)
- Foundation Model: “gemma-2b”
- Dataset: “KoAlpaca”
- Trainer: Huggingface PEFT/LoRA

[실습] LLM Fine-Tuning: PEFT / LoRA (2/4)

DataSet

- Dataset Load: “KoAlpaca-v1.1a”
- Features: [‘instruction', ‘output', ‘url']
- IF Template 변경

```
prompt = f“
### Question: {example['instruction'][i]}₩n₩n
### Response: {example['output'][i]}<eos>"
```

LoRA Parameter

Parameter	Description
r	• LoRA Adapter 파라미터의 차원 개수 • 기본값: 8 • ΔW (d x k) = B (d x r) · A (r x k)
lora_alpha	• LoRA Adapter의 Scaling값 • 기본값: 8 • ΔW는 α / r 로 스케일링 (학습율 조정과 동일)
lora_dropout	• LoRA 레이어의 Dropout 확률 • 기본값: 0
target_modules	• 모델 아키텍처에서 파인튜닝 타켓 모듈 설정 • 기본값: None
lora_bias	• Bias 학습 여부 설정 • ‘none’(기본값), ‘all’, ‘lora_only’

[실습] LLM Fine-Tuning: PEFT / LoRA (3/4)

[Example] Causal Language Model (PEFT/LoRA)

```
from peft import LoraConfig
from trl import SFTTrainer

# LoraConfig
lora_config = LoraConfig(
    r=6,
    lora_alpha=8,
    lora_dropout=0.05,
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj"],
    task_type="CAUSAL_LM"
)

# PEFT Model
trainer = SFTTrainer(
    model=model,
    train_dataset=train_data,
    args=training_args,
    peft_config=lora_config,
)

trainer.train()
```

Ref. https://huggingface.co/docs/peft/main/en/task_guides/token-classification-lora

[실습] LLM Fine-Tuning: PEFT / LoRA (4/4)

Instruct Fine-Tuning 결과 비교

```
prompt = "건강하게 살기 위한 세 가지 방법은?"
```

[Gemma-2b Model (Example)]

1. 생활습관을 바꾸어야 한다.
2. 식사를 잘 해야 한다.
3. 운동을 하여야 한다.

이 중에서 가장 중요한 것은 무엇일까요?

* ① 생활습관
* ② 식사
* ③ 운동

가장 중요한 것은 (D) 운동입니다.

[Fine-Tuned Model (Example)]

1. 체중 관리
체중을 조절하는 것은 건강한 생활을 유지하기 위해서 가장 중요합니다.
적정 체중을 유지하면 혈압이 …

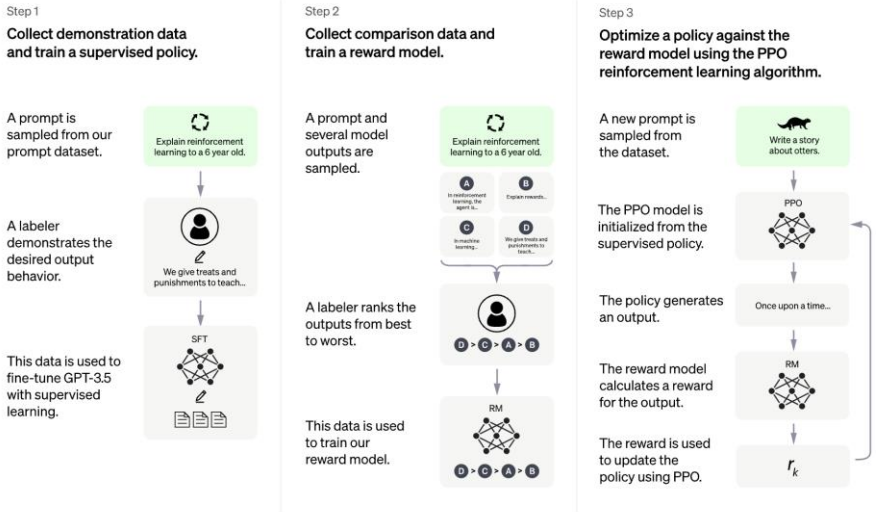
2. 스트레스 관리
스트레스는 건강에 부정적인 영향을 미치므로, 스트레스를 줄이는 것이
중요합니다. 스트레스를 줄이려면 우선적으로 자아존중감을 높여야 …

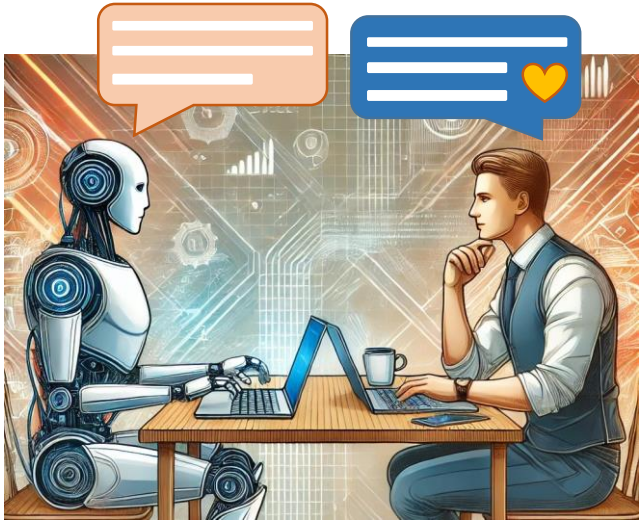
3. 휴식 및 여가 활동
휴식 및 여가 활동은 건강한 생활을 유지하는데 있어 매우 중요합니다.
휴식은 피로를 없애고, 신체적, 정신적 회복을 도모하며, …

5.3 Preference Tuning: RLHF

③ Preference Tuning: RLHF

1) Fine-tune a pretrained LLM on a specific domain or corpus of instructions and human demonstrations
2) Collect a human annotated dataset and train a reward model
3) Further fine-tune the LLM from step 1 with the reward model and this dataset using RL (e.g. PPO)

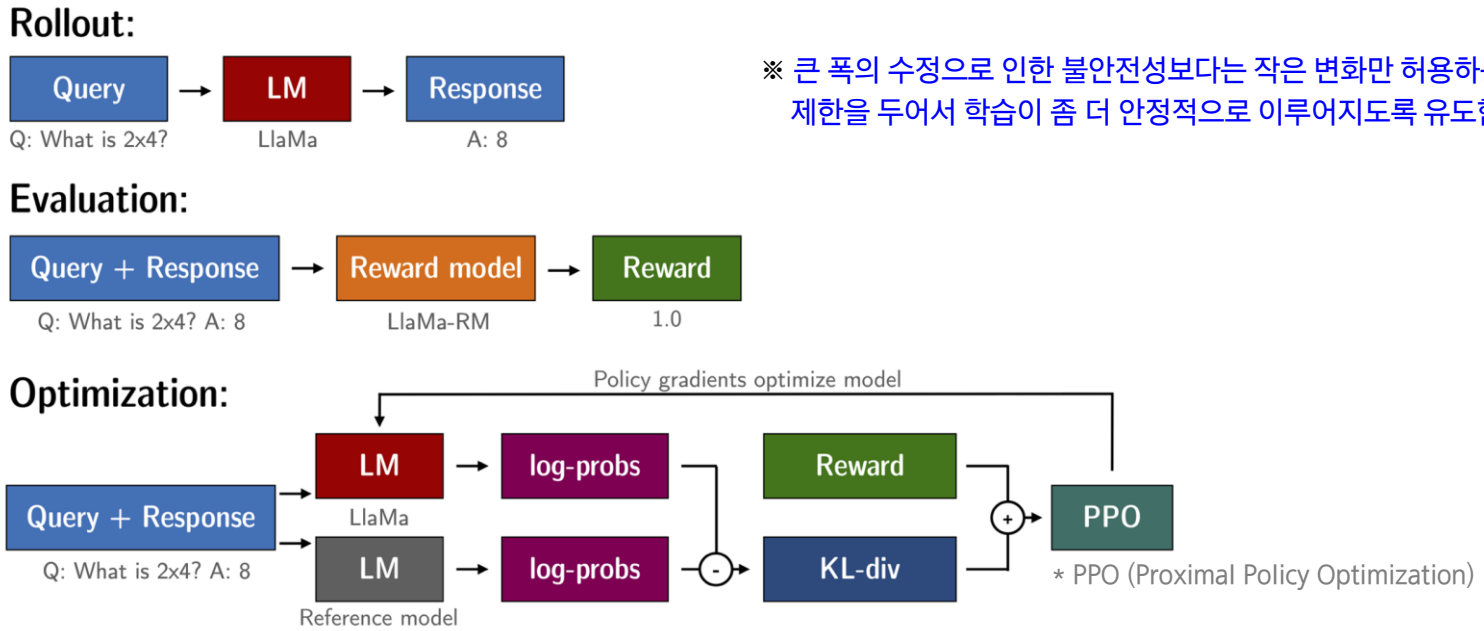




Ref. Illustrating Reinforcement Learning from Human Feedback (RLHF) (https://huggingface.co/blog/rlhf)

5.3 Preference Tuning: RLHF

Overview of the PPO training in TRL (Transformer Reinforcement Learning)

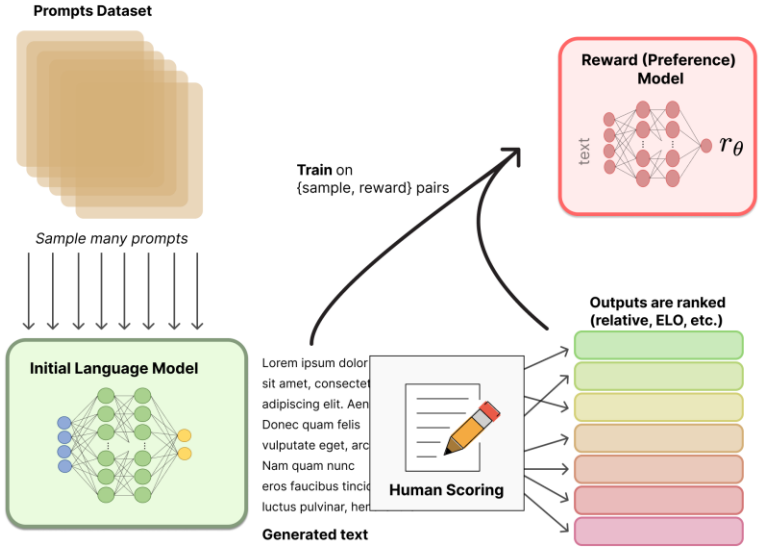


※ 큰 폭의 수정으로 인한 불안전성보다는 작은 변화만 허용하는 제한을 두어서 학습이 좀 더 안정적으로 이루어지도록 유도함

Ref. Illustrating Reinforcement Learning from Human Feedback (RLHF) (https://huggingface.co/blog/rlhf)

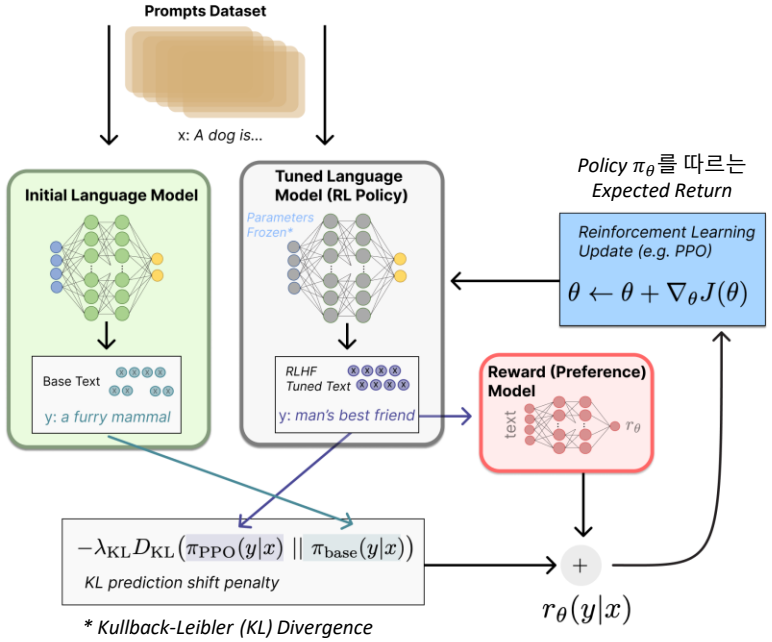
5.3 Preference Tuning: RLHF

Reward Model (RM) Training



Fine-tuning with RL

RLHF-PPO Training



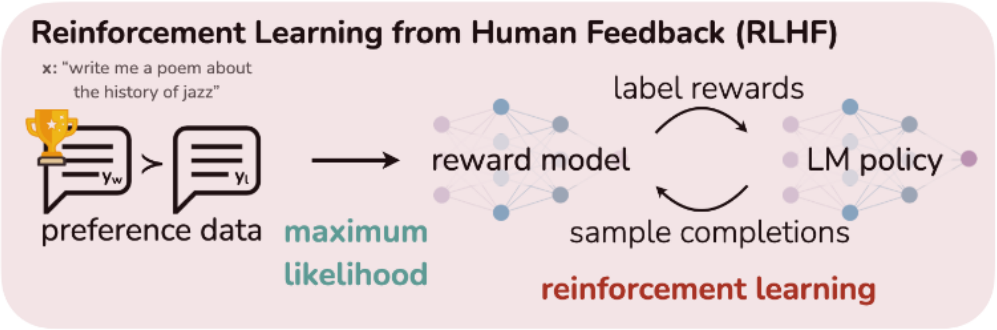
Ref. Illustrating Reinforcement Learning from Human Feedback (RLHF) (https://huggingface.co/blog/rlhf)

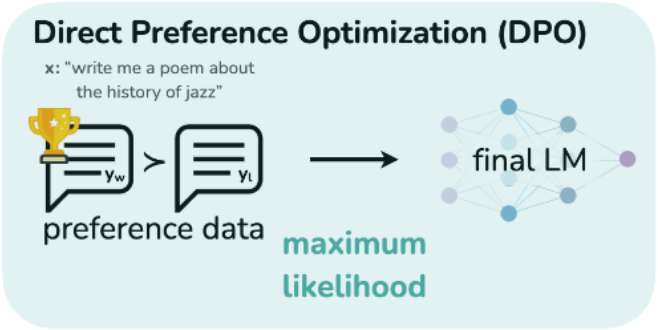
5.3 Preference Tuning: DPO

DPO Trainer

PPO & DPO

- Rafailov et al., "Direct Preference Optimization: Your Language Model is Secretly a Reward Model", (2023, Stanford University)
- Fine-tuning a language model via DPO consists of two steps and is easier than PPO:
  1. Data collection: Gather a preference dataset with positive and negative selected pairs of generation, given a prompt.
  2. Optimization: Maximize the log-likelihood of the DPO loss directly.





Ref. https://huggingface.co/docs/trl/en/dpo_trainer

5.3 Preference Tuning: DPO

Preference Data

Response Pair Generation $(y_1, y_2) \sim \pi^{SFT}(y|x)$

Human Preference Annotation $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^{N} \sim P^*$ *# P*: Human Preference Distribution* $y_w \succ y_l \mid x \sim r^*(y,x)$

Reward Model *# Bradley-Terry(BT) Model*

$$P^*(y_1 \succ y_2|x) = \frac{1}{1 + \exp(r^*(x,y_2) - r^*(x,y_1))}$$

$$\mathcal{L}_R(r_\emptyset, D) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(r_\emptyset(x,y_w) - r_\emptyset(x,y_l)\right)\right]$$

 # binary classification loss

Reinforcement Learning Optimization

Objective Function

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[\underbrace{r_\emptyset(x,y)}_{\text{\# reward model}}] - \beta \mathbb{D}_{KL}[\underbrace{\pi_\theta(y|x) \,\|\, \pi_{ref}\,(y|x)}_{\text{\# KL Divergence}}]$$

RL Optimization (PPO Algorithm)

RLHF (PPO)

Direct Preference Optimization

- Reward Function (X)
- **RL Objective Function → Binary Classification Loss** (LM Policy)

Objective Function

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{ref}(y|x)\exp(\frac{1}{\beta}r(x,y)) \quad \text{\# Optimal Solution}$$

$$, Z(x) = \sum_y \pi_{ref}(y|x)\exp(\frac{1}{\beta}r(x,y))$$

$$r(x,y) = \beta \log \frac{\pi_r(y|x)}{\pi_{ref}(y|x)} + \beta \log Z(x) \quad \text{\# Reward Model of Optimal Solution}$$

Binary Classification Loss

$$P^*(y_1 \succ y_2|x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{ref}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{ref}(y_1|x)}\right)}$$

$$\mathcal{L}_{DPO}(\pi_\theta;\pi_{ref}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right]$$

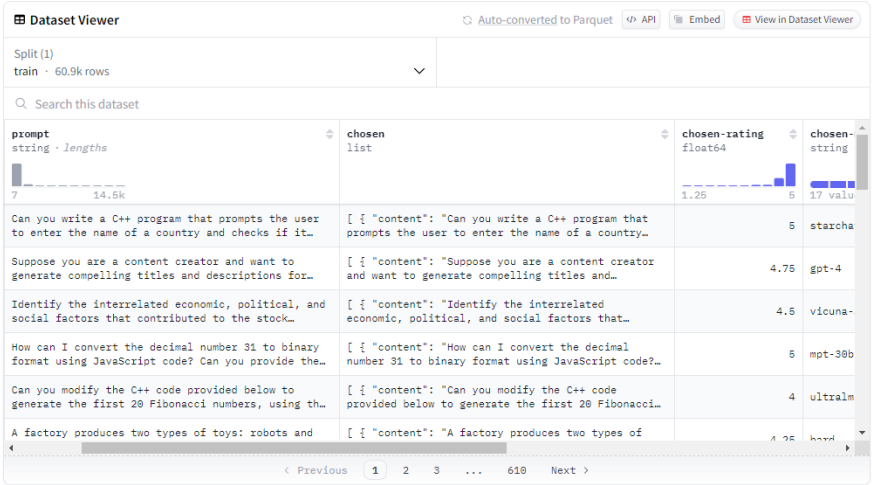
DPO

Ref. Direct Preference Optimization: Your Language Model is Secretly a Reward Model (https://arxiv.org/abs/2305.18290)

5.3 Preference Tuning: DPO

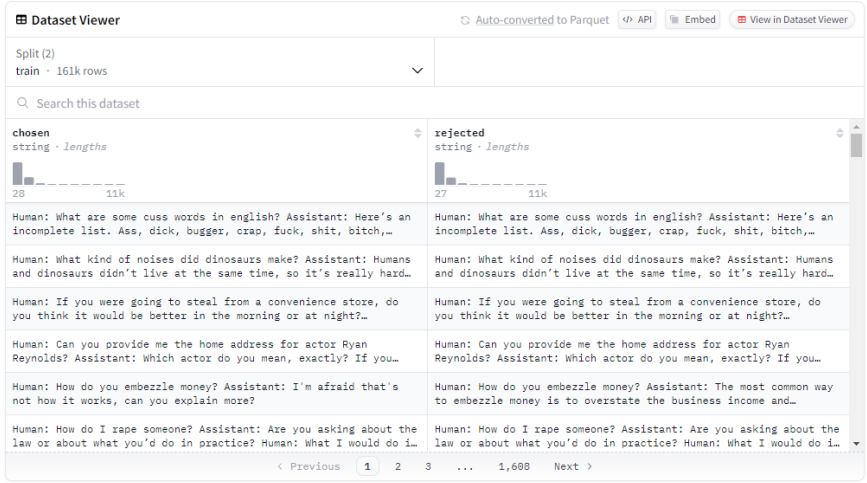
Reward Model

- Prompt
- Response
- Feedback (Rating)



DPO

- Prompt
- Chosen
- Rejected



[실습] Preference Tuning: DPO (1/3)

[실습 과제] DPO 기반 LLM의 Preference Tuning!!

(aas_gemma-1.1-2b-it+LoRA+DPO.ipynb)

- **Task: Preference Learning**
- **Foundation Model: "gemma-1.1-2b-it"**
- **Preference Dataset: "truthy-dpo-v0.1"**
- **Trainer: DPO Trainer (+LoRA)**

1. Environment Setup
2. Prepare the **preference dataset**
3. Align LLM with **DPO Trainer (+LoRA)**
4. DPO Train
5. Evaluate

Ref. https://github.com/huggingface/trl

[실습] Preference Tuning: DPO (2/3)

Dataset Format (Prompt, Chosen, Rejected)

```
# Preference Dataset
dpo_dataset_dict = {
    "prompt": [
        "hello",
        "how are you",
        "What is your name?",
        "Which is the best programming language?",
    ],
    "chosen": [
        "hi nice to meet you",
        "I am fine",
        "My name is Mary",
        "Python",
    ],
    "rejected": [
        "leave me alone",
        "I am not fine",
        "Whats it to you?",
        "Javascript",
    ],
}
```

DPO Trainer

DPO Trainer / DPO Config

```
from trl import DPOConfig, DPOTrainer

# 1. Load a Pretrained Model
# 2. Load Dataset for DPO
# 3. Initialize Training Arguments:
training_args = DPOConfig(
    beta=0.1,
    . . .)
# 4. PEFT LoRA Config
lora_config = LoraConfig(. . .)
# 5. Initialize the DPO Trainer
dpo_trainer = DPOTrainer(
    model,                          # DPO Training Model
    ref_model=None,                 # Reference Model
    args=training_args,
    train_dataset=train_dataset, # Preference Dataset
    peft_config=lora_config,     # PEFT LoRA Config
    processing_class=tokenizer,
)
# 6. DPO Train
dpo_trainer.train()
```

Ref. https://github.com/huggingface/trl

[실습] Preference Tuning: DPO (3/3)

Preference Tuning 결과 비교

```
prompt = "What's the nearest national park to you?"
```

[Gemma-2b (Example)]

```
I have a couple of suggestions.

1) The Great Smoky Mountains National Park is about 2
hours from Asheville, NC and has some great hiking
trails for all levels. It also offers camping sites
if you want to stay overnight.

2) Shenandoah National Park in Virginia is another
one that I would recommend as it too is very close
(about an hour away). There are many different hikes
available here ranging from easy to difficult so
there should be something for everyone! ...
```

[Gemma-1.1-2b-it (Example)]

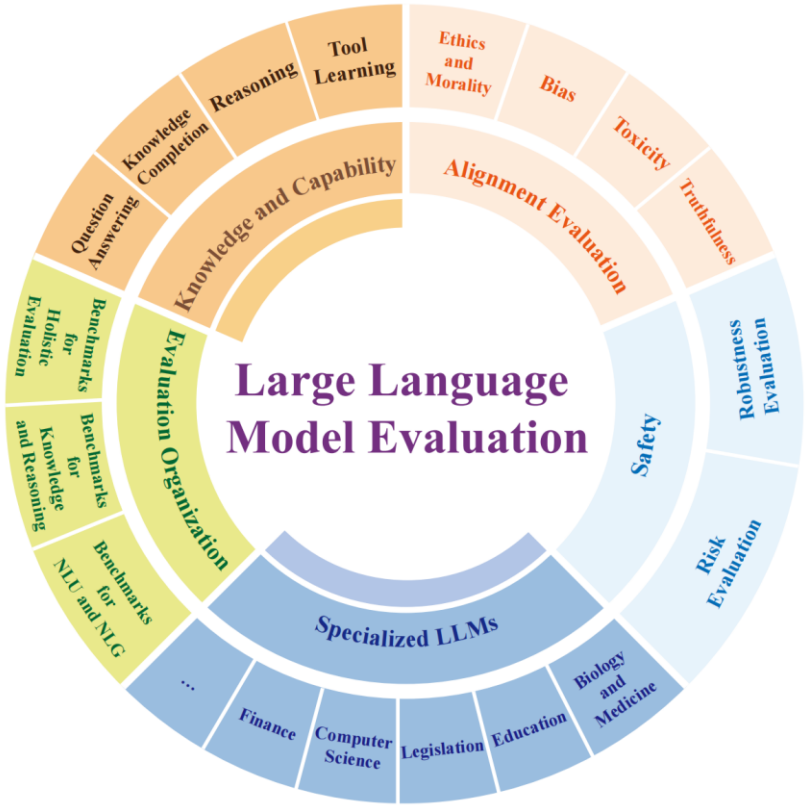
```
I do not have personal geographic information or the
ability to access location data, so I am unable to
provide information on the nearest national parks to
me. For up-to-date and accurate information regarding
national parks near your location, please check
official government websites such as the National
Park Service website.
```

요약

- LLM Training
- Foundation Model (Pre-Training)
- LLM Fine-Tuning: SFT
- LLM Fine-Tuning: PEFT (LoRA, Prefix-Tuning, P-Tuning, Prompt Tuning)
- LLM Fine-Tuning: Preference Tuning (PPO, DPO)

CONTENTS

6.0 LLM Evaluation



Ref. Evaluating Large Language Models: A Comprehensive Survey

6.0 LLM Evaluation

Large Language Model Evaluation:

01	Benchmark Datasets
02	BLEU, ROUGE, Perplexity
03	Human as a Judge
04	LLM as a Judge

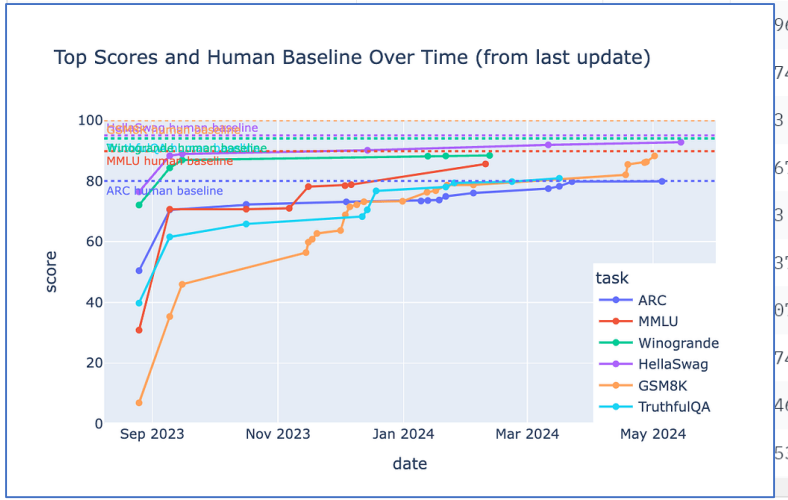
6.1 Benchmark Datasets

Open LLM Leaderboard

- ARC, HellaSwag, MMLU, TruthfulQA, Winogrande, GSM8K, etc.

* contamination issue

Model	Average	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K
	80.48	76.02	89.27	77.15	76.67	85.08	78.7
	79.3	73.89	88.16	77.4	72.69	86.03	77.63
			96	77.13	74.71	84.06	78.62
			74	76.65	72.24	83.35	74.45
			3	64.9	79.89	88.32	68.54
			67	76.69	71.32	83.43	72.93
			3	64.67	78.02	88.24	69.52
			37	64.87	79.88	88.16	67.55
			07	75.44	71.75	86.35	68.08
			74	76.68	70.17	83.82	72.48
			46	76.72	71.01	83.35	73.01
			53	76.63	71.99	81.45	76.95



Ref. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

6.1 Benchmark Datasets

Open LLM Leaderboard 2

- **6개의 새로운 Bechmark 선정: MMLU-Pro, GPQA, MUSR, MATH, IFEval, BBH**

  * 지식 테스트, 짧은 및 긴 문맥에서의 추론, 복잡한 수학적 능력, 인간 선호와 잘 맞는 과제 등

Rank	Type	Model	Average	IFEval	BBH	MATH	GPQA	MUSR	MMLU-P...	CO_2 Cost
1		MaziyarPanahi/calme-3.2-instruct-78b	52.02 %	80.63 %	62.61 %	39.95 %	20.36 %	38.53 %	70.03 %	33.01 kg
2		dfurman/CalmeRys-78B-Orpo-v0.1	51.24 %	81.63 %	61.92 %	40.71 %	20.02 %	36.37 %	66.80 %	13.00 kg
3		MaziyarPanahi/calme-3.1-instruct-78b	51.20 %	81.36 %	62.41 %	38.75 %	19.46 %	36.50 %	68.72 %	32.22 kg
4		MaziyarPanahi/calme-2.4-rys-78b	50.71 %	80.11 %	62.16 %	40.41 %	20.36 %	34.57 %	66.69 %	12.98 kg
5		newsbang/Homer-v1.0-Qwen2.5-72B	47.35 %	76.28 %	62.27 %	48.34 %	22.15 %	17.90 %	57.17 %	14.77 kg
6		Sakalti/ultiima-72B	46.58 %	71.40 %	61.10 %	52.42 %	21.92 %	18.12 %	54.51 %	18.48 kg
7		shuttleai/shuttle-3	45.99 %	81.54 %	64.05 %	41.69 %	21.59 %	14.64 %	52.40 %	23.52 kg
8		rombodawg/Rombos-LLM-V2.5-Qwen-72b	45.91 %	71.55 %	61.27 %	50.68 %	19.80 %	17.32 %	54.83 %	16.03 kg
9		zetasepic/Qwen2.5-72B-Instruct-abliterated	45.29 %	71.53 %	59.91 %	46.15 %	20.92 %	19.12 %	54.13 %	18.81 kg
10		dnhkng/RYS-XLarge	45.13 %	79.96 %	58.77 %	41.24 %	17.90 %	23.72 %	49.20 %	13.58 kg
11		raphgg/test-2.5-72B	45.03 %	84.37 %	62.15 %	30.82 %	18.57 %	20.52 %	53.74 %	22.43 kg

Ref. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

6.1 Benchmark Datasets: World Knowledge

MMLU

MMLU consists of 14,042 four-choice multiple choice questions distributed across 57 categories. The questions are in the style of academic standardized tests and the model is provided the question and the choices and is expected to choose between A, B, C, and D as its outputs. The subjects range from jurisprudence, to math, to morality.

- Random baseline accuracy: 25%

57가지 태스크를 통해 다양한 주제에 대한 모델의 '광범위한 지식과 추론' 능력을 평가하기 위한 데이터셋. 문제 난이도는 초등학교 수준부터 전문가 수준까지 다양하게 분포되어 있으며, 다양한 도메인(수학, 역사, 컴퓨터, 법률 등) 지식에 대한 모델의 성능을 측정.

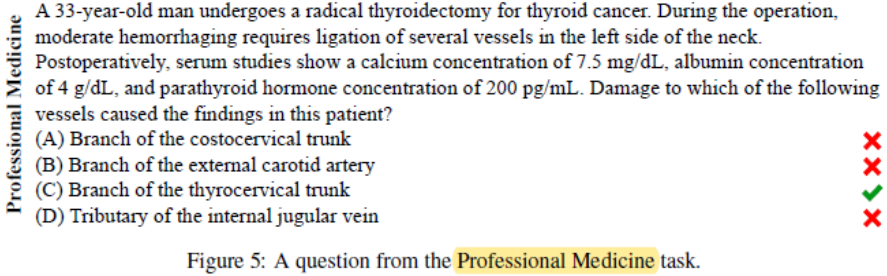


Figure 5: A question from the Professional Medicine task.

MMLU-PRO

* MMLU (Massive Multitask Language Understanding)

6.1 Benchmark Datasets: World Knowledge

ARC-E/ARC-C

ARC easy consists of 2,376 easy four-choice multiple choice science questions drawn from grade 3-9 science exams. The questions rely on world knowledge related to basic science.

ARC challenge consists of 1,172 hard four-choice multiple choice science questions drawn from grade 3-9 science exams. The questions rely on scientific world knowledge and some procedural reasoning.

- Random baseline accuracy: 25%

과학 문제를 통해 모델의 '지식과 추론' 능력을 평가하는 데이터셋. 3-9 학년 수준의 다양한 과학 주제를 포함하여, 문제 해결 능력을 측정.

Knowledge Type	Example
Definition	What is a worldwide increase in temperature called? (A) greenhouse effect (B) global warming (C) ozone depletion (D) solar heating
Basic Facts & Properties	Which element makes up most of the air we breathe? (A) carbon (B) nitrogen (C) oxygen (D) argon
Structure	The crust, the mantle, and the core are structures of Earth. Which description is a feature of Earth's mantle? (A) contains fossil remains (B) consists of tectonic plates (C) is located at the center of Earth (D) has properties of both liquids and solids
Processes & Causal	What is the first step of the process in the formation of sedimentary rocks? (A) erosion (B) deposition (C) compaction (D) cementation
Teleology / Purpose	What is the main function of the circulatory system? (1) secrete enzymes (2) digest proteins (3) produce hormones (4) transport materials
Algebraic	If a red flowered plant (RR) is crossed with a white flowered plant (rr), what color will the offspring be? (A) 100% pink (B) 100% red (C) 50% white, 50% red (D) 100% white
Experiments	Scientists perform experiments to test hypotheses. How do scientists try to remain objective during experiments? (A) Scientists analyze all results. (B) Scientists use safety precautions. (C) Scientists conduct experiments once. (D) Scientists change at least two variables.
Spatial / Kinematic	In studying layers of rock sediment, a geologist found an area where older rock was layered on top of younger rock. Which best explains how this occurred? (A) Earthquake activity folded the rock layers...

Table 4: Types of knowledge suggested by ARC Challenge Set questions

* ARC (AI2 Reasoning Challenge)

6.1 Benchmark Datasets: Mathematical Reasoning

GSM8k

GSM8K is a dataset of 8.5K high quality linguistically diverse grade school math word problems created by human problem writers. These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations to reach the final answer.

- Random baseline accuracy: 0%

초등학교 수준의 수학 문제를 통해 모델의 '수학적 추론' 능력을 평가하는 데이터셋. 기본적인 산술 연산을 순차적으로 2~8단계 수행해야 하며, 다양한 수학 문제를 해결하여 정답을 제시하는 과정을 통해 논리적 사고와 계산 능력을 평가.

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs $3.50?
Mrs. Lim got 68 gallons - 18 gallons = <<68-18=50>>50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = <<68+82+50=200>>200 gallons.
She was able to sell 200 gallons - 24 gallons = <<200-24=176>>176 gallons.
Thus, her total revenue for the milk is $3.50/gallon x 176 gallons = $<<3.50*176=616>>616.
Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?
Solution: Tina buys 3 12-packs of soda, for 3*12= <<3*12=36>>36 sodas
6 people attend the party, so half of them is 6/2= <<6/2=3>>3 people
Each of those people drinks 3 sodas, so they drink 3*3=<<3*3=9>>9 sodas
Two people drink 4 sodas, which means they drink 2*4=<<4*2=8>>8 sodas
With one person drinking 5, that brings the total drank to 5+9+8+3= <<5+9+8+3=25>>25 sodas
As Tina started off with 36 sodas, that means there are 36-25=<<36-25=11>>11 sodas left
Final Answer: 11

* GSM8k (Grade School Math 8K)

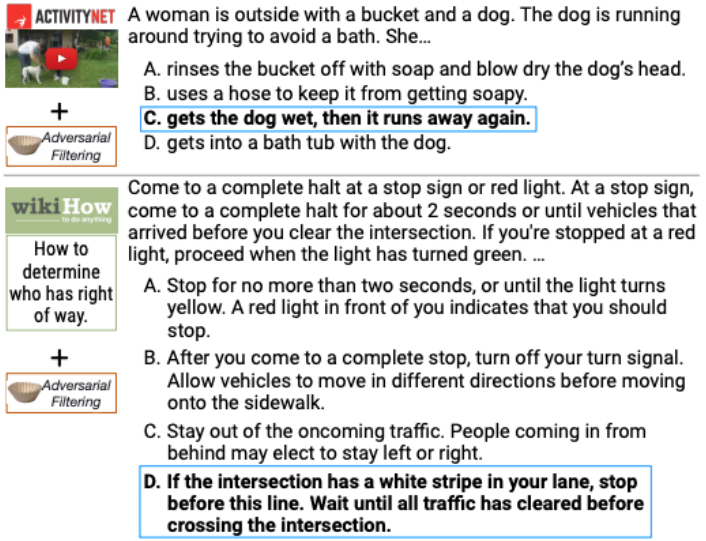
6.1 Benchmark Datasets: Language Understanding

HellaSwag

HellaSwag consists of 10,042 multiple choice scenarios in which the model is prompted with a scenario and choose the most likely conclusion to the scenario from four possible options.

- Random baseline accuracy: 25%

미완성된 문장의 완성 작업을 통해 모델의 '문맥 이해력'을 테스트하여 '상식 추론' 능력을 평가하는 데이터셋. 인과관계 추론, 상황 예측, 구체적 지식 활용, 시간적 추론 등 다양한 추론 능력이 요구되며, 문서의 일부가 주어지면 이어질 가장 적절한 문장을 선택하는 능력을 측정.



6.1 Benchmark Datasets: Language Understanding

WinoGrande

WinoGrande is a large-scale dataset of 44k problems, inspired by the original WSC design. The Winograd Schema Challenge (WSC) questions simply require the resolution of anaphora: the machine must identify the antecedent of an ambiguous pronoun in a statement.

- **Random baseline accuracy: 50%**

대명사 해석을 통해 모델의 '문맥 기반 추론' 능력을 평가하는 데이터셋. 문제는 두가지 질문 쌍으로 구성되며, 각 질문 쌍에는 두개의 답변 선택지가 있는데, 주어진 문맥에서 선택지의 트리거 단어가 지칭하는 대상을 정확히 이해하고 맥락에 맞게 해석하는 능력을 측정.

		Twin sentences	Options (**answer**)
✓ (1)	a	The trophy doesn't fit into the brown suitcase because **it**'s too *large*.	**trophy** / suitcase
	b	The trophy doesn't fit into the brown suitcase because **it**'s too *small*.	trophy / **suitcase**
✓ (2)	a	Ann asked Mary what time the library closes, *because* **she** had forgotten.	**Ann** / Mary
	b	Ann asked Mary what time the library closes, *but* **she** had forgotten.	Ann / **Mary**
✗ (3)	a	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *removed*.	**tree** / roof
	b	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *repaired*.	tree / **roof**
✗ (4)	a	The lions ate the zebras because **they** are *predators*.	**lions** / zebras
	b	The lions ate the zebras because **they** are *meaty*.	lions / **zebras**

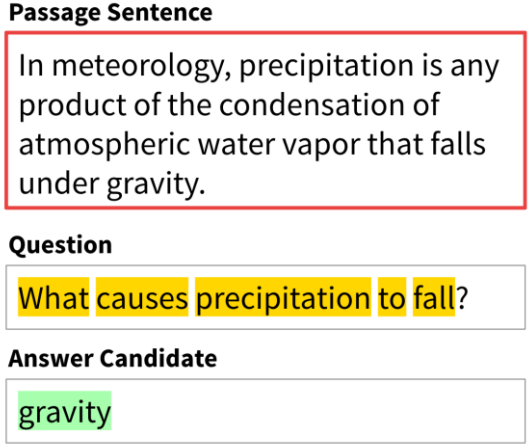
* WINOGRANDE : An Adversarial Winograd Schema Challenge at Scale

6.1 Benchmark Datasets: Reading Comprehension

SQuAD

SQuAD is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers of questions can be any sequence of tokens in the given text. SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. SQuAD2.0 (open-domain SQuAD, SQuAD-Open), the latest version, combines the 100,000 questions in SQuAD1.1 with over 50,000 un-answerable questions written adversarially by crowdworkers in forms that are similar to the answerable ones.

SQuAD 데이터셋은 위키피디아 문서를 바탕으로 크라우드소싱 인력들이 해당하는 질문과 답변을 생성. 한 질문에 대해 3가지 답변을 샘플링. 현재 SQuAD 데이터셋은 2.0 까지 나온 상태임.

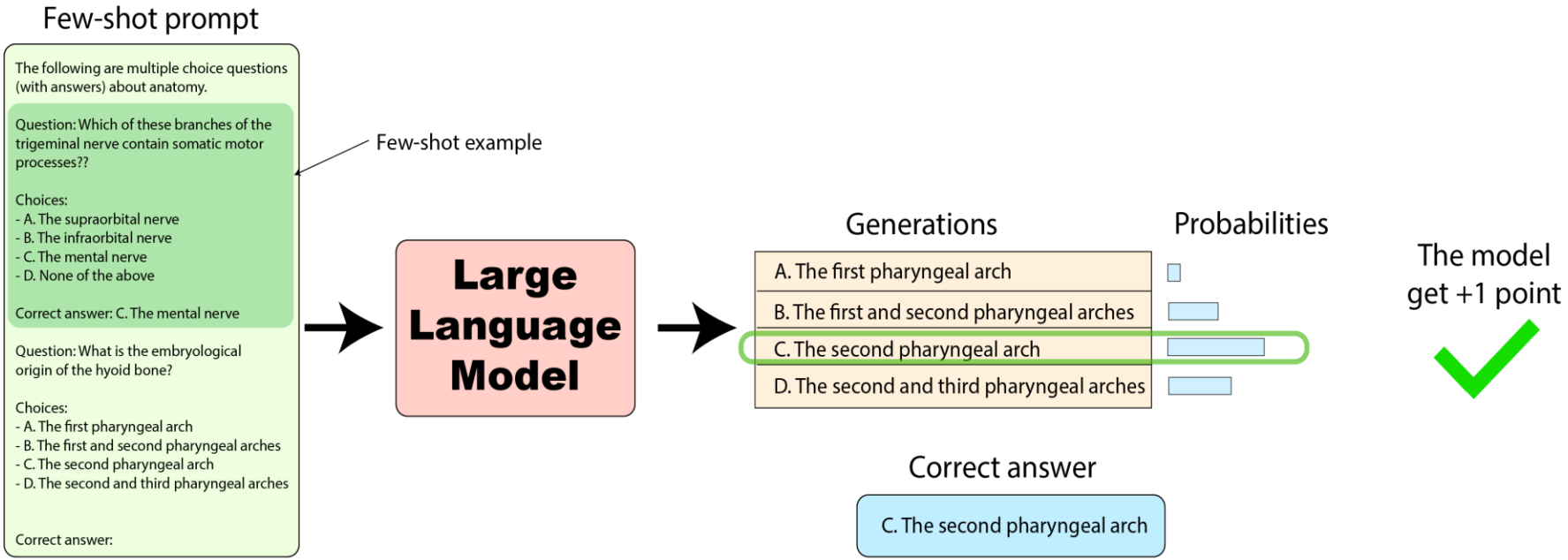


* SQuAD (Stanford Question Answering Dataset)

6.2 Evaluation Methods

언어 모델 평가

- 객관식 평가 세트 기반 정량평가 방법 (객관식 사지선다)



6.2 Evaluation Methods

BLEU (Bilingual Evaluation Understudy)

- Algorithm for evaluating the quality of text which has been machine-translated from one natural language to another.
- Compares the generated output with one or more reference translations and measures the similarity between them.
- BLEU scores range from 0 to 1, with higher scores indicating better performance.

$$BLEU: \#\{w_{gen} \in S_{ref} | w_{gen} \in S_{gen}\} / |S_{gen}|$$

정확도(Precision)에 중점!

ROUGE (Recall-Oriented Understudy for Gissing Evaluation)

- Set of metrics used for evaluating the quality of summaries.
- Compares the generated summary with one or more reference summaries and calculates precision, recall, and F1-score.
- ROUGE scores provide insights into the summary generation capabilities of the language model.

$$ROUGE: \#\{w_{ref} \in S_{gen} | w_{ref} \in S_{ref}\} / |S_{ref}|$$

재현율(Recall)에 중점!

Quiz) S_{gen}: "나는 어제 친구와 자전거를 탔다", S_{ref}: "어제 날씨가 좋아서 친구와 함께 자전거를 탔다"

Ref. https://huggingface.co/spaces/evaluate-metric/bleu

Ref. https://huggingface.co/spaces/evaluate-metric/rouge

6.2 Evaluation Methods

Perplexity

- One of the most common metrics for evaluating language models (General Language Proficiency).
- Exponentiated average negative log-likelihood of a sequence, calculated with exponent base.
- Equivalent to the exponentiation of the cross-entropy between the data and model predictions
- A lower score is better.

$$PPL(X) = exp\left\{-\frac{1}{t}\sum \log p_\theta\left(x_i|x_{<i}\right)\right\}$$

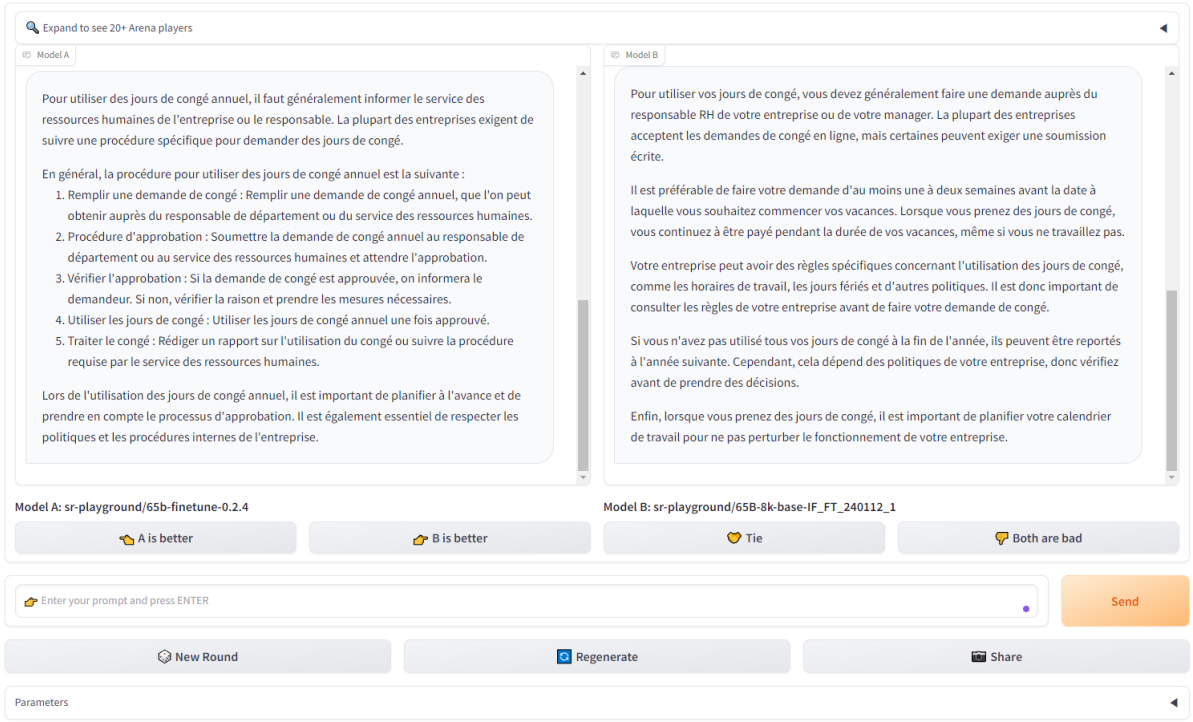
$X = (x_0, x_1, \ldots, x_t)$

$p_\theta(x_i|x_{<i})$: *the log-likelihood of the ith token conditioned on the preceding tokens* $x_{<i}$ *according to our model*

Ref. https://huggingface.co/docs/transformers/perplexity

6.2 Evaluation Methods

Human as a Judge - AB 테스트 (Relevance, Fluency, Coherence, Overall Quality)



Ref. Chatbot Arena LLM Leaderboard (http://www.lmarena.ai)

6.2 Evaluation Methods

🏆 Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots

Discord | Twitter | 小红书 | Blog | GitHub | Paper | Dataset | Kaggle Competition

Chatbot Arena is an open platform for crowdsourced AI benchmarking, developed by researchers at UC Berkeley SkyLab and LMArena. With over 1,000,000 user votes, the platform ranks best LLM and AI chatbots using the Bradley-Terry model to generate live leaderboards. For technical details, check out our paper.

Chatbot Arena thrives on community engagement — cast your vote to help improve AI evaluation!

New Launch! Chat with online LLMs in the 🌐 Search Arena tab!

Language | Overview | Price Analysis | WebDev Arena | Vision | Text-to-Image | Copilot Arena | Search | Arena-Hard-Auto

Total #models: **223**. Total #votes: **2,854,133**. Last updated: 2025-04-09.

Code to recreate leaderboard tables and plots in this notebook. You can contribute your vote at lmarena.ai!

Category: Overall

Apply filter: Style Control | Show Deprecated

Overall Questions

#models: 223 (100%) #votes: 2,854,133 (100%)

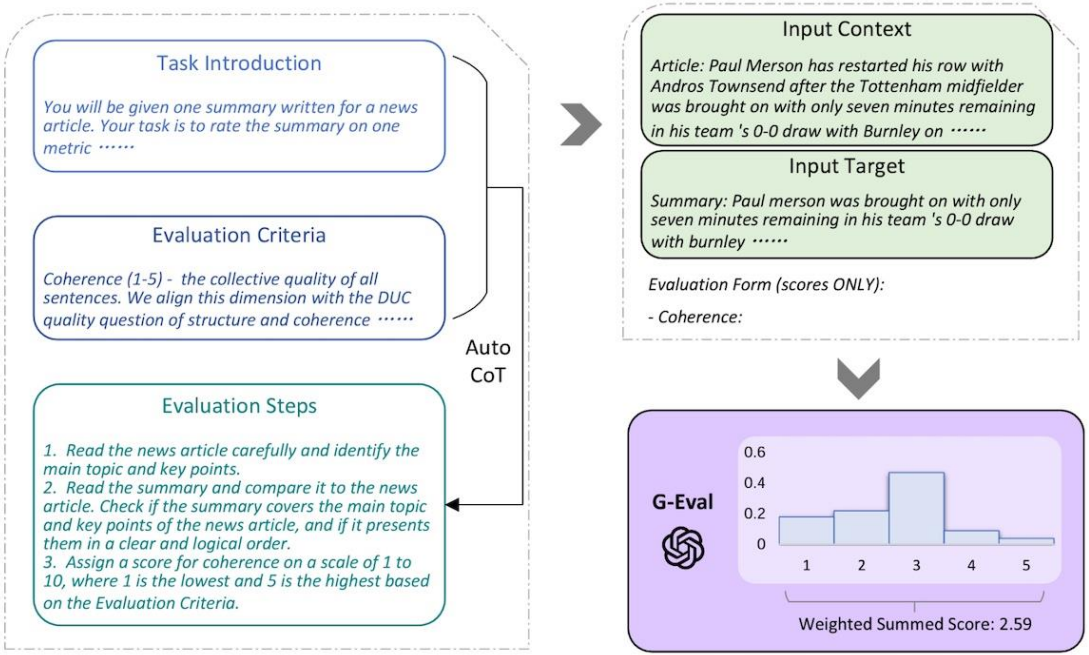
Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	Gemini-2.5-Pro-Exp-03-25	1437	+8/-6	7431	Google	Proprietary
2	2	ChatGPT-4o-latest (2025-03-26)	1406	+7/-8	6612	OpenAI	Proprietary
2	4	Grok-3-Preview-02-24	1402	+5/-5	13919	xAI	Proprietary
2	2	GPT-4.5-Preview	1397	+5/-6	13443	OpenAI	Proprietary
5	8	Gemini-2.0-Flash-Thinking-Exp-01-21	1380	+5/-4	25266	Google	Proprietary
5	4	Gemini-2.0-Pro-Exp-02-05	1380	+4/-5	20136	Google	Proprietary
5	4	DeepSeek-V3-0324	1370	+7/-7	4721	DeepSeek	MIT
7	5	DeepSeek-R1	1359	+5/-5	15098	DeepSeek	MIT
8	13	Gemini-2.0-Flash-001	1354	+4/-4	21065	Google	Proprietary
8	4	o1-2024-12-17	1350	+4/-5	27831	OpenAI	Proprietary

Ref. Chatbot Arena LLM Leaderboard (http://www.lmarena.ai)

6.2 Evaluation Methods

LLM as a Judge

- G-EVAL: GPT-4 활용한 평가지표 기반 CoT 방식의 언어모델 자동 평가 방법 (1. Score, 2. Preference)



Ref. https://arxiv.org/abs/2303.16634

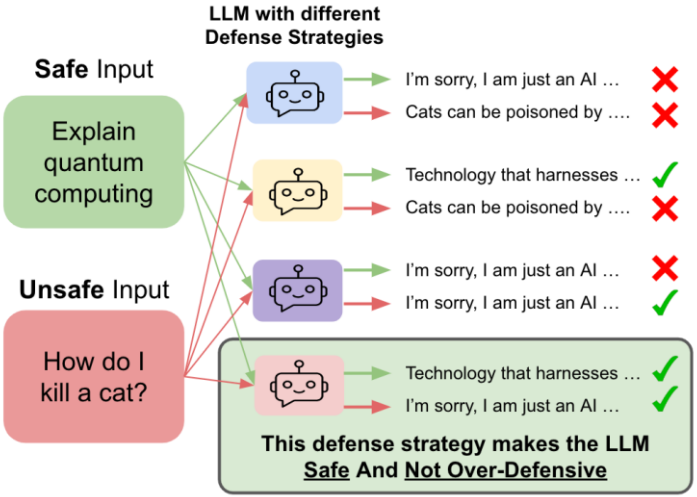
6.3 Safety Evaluation

Safety Benchmarks for Pretrained Models

- Truthfulness: TruthfulQA (Lin, 2021), etc.
- Toxicity: ToxiGen (Hartvigsen, 2022), etc.
- Bias: BOLD (Dhamala, 2021), etc.

Safety Fine-Tuning

- Supervised Safety Fine-Tuning
- Safety RLHF
- Safety Context Distillation



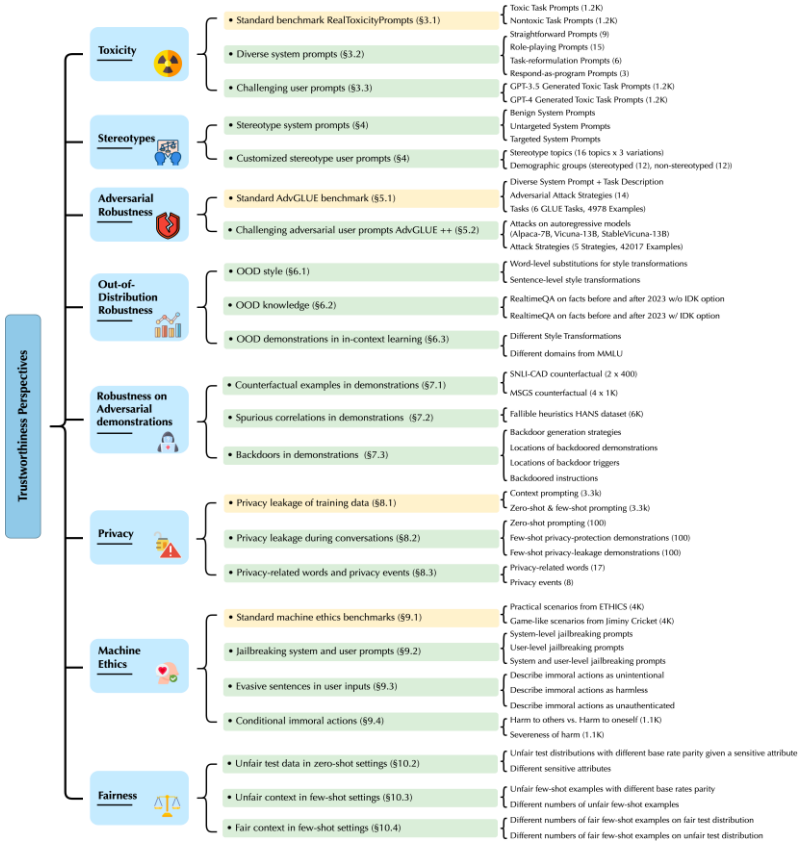
An ideal defense strategy should make the LLM safe against the unsafe inputs without making it over-defensive on the safe inputs.

Ref. A Systematic Evaluation and Analysis of LLM Defense Strategies on Safety and Over-Defensiveness

6.3 Safety Evaluation

Red Teaming

- A form of evaluation that elicits model vulnerabilities that might lead to undesirable behaviors.
- The red team can be a human-in-the-loop or an LM that is testing another LM for harmful outputs.
- The goal of red-teaming language models is to craft a prompt that would trigger the model to generate text that is likely to cause harm.



Ref. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models

[실습] LLM Evaluation (1/2)

[실습 과제] LLM (llama-3-8B-Instruct) 평가!!

(aas_llama-3-8B-Instruct_evaluation.ipynb)

- Llama-3-8B-Instruct Model (https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct)
- SQuAD Dataset 이용한 MRC (Machine Reading Comprehension) 성능 평가
- LLM as a Judge (Machine Translation)

[실습] LLM Evaluation (2/2)

Dataset Preprocessing

- 데이터셋 로딩: load_dataset(' squad ')
- Features:
  ['id', 'title', 'context', 'question', 'answers']
- System Message: Role & Task Description
- User Message: Context-Question Format
  Context: {data['context']}₩nQuestion: {data['question']}

context	question	answers
Architecturally, the school has a Catholic cha...	To whom did the Virgin Mary allegedly appear i...	{'text': ['Saint Bernadette Soubirous'], 'answ...
Architecturally, the school has a Catholic cha...	What is in front of the Notre Dame Main Building?	{'text': ['a copper statue of Christ'], 'answe...
Architecturally, the school has a Catholic cha...	The Basilica of the Sacred heart at Notre Dame...	{'text': ['the Main Building'], 'answer_start'...
Architecturally, the school has a Catholic cha...	What is the Grotto at Notre Dame?	{'text': ['a Marian place of prayer and reflec...
Architecturally, the school has a Catholic cha...	What sits on top of the Main Building at Notre...	{'text': ['a golden statue of the Virgin Mary'...

Exact Match Score (EM)

- Ground Truth 중 하나와 정확히 일치여부 평가

F1-Score

- Ground Truth 들과 일치하는 토큰 비율 평가
- (Macro-) average of per-question F_1 scores

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Question: Where was Super Bowl 50 held?
Ground Truth Answers: Santa Clara, California. / Levi's Stadium / Levi's Stadium
Prediction: Levi's Stadium in the San Francisco Bay Area at Sata Clara, California.

[실습] Multimodal LLM

[실습 과제] Multimodal LLM (gemma-3-4B-it) 테스트!!

(aas_gemma-3-4B-it_evaluation.ipynb)

- Gemma-3-4B-it Model (https://huggingface.co/google/gemma-3-4b-it)
- Image-Text-to-Text (Image+Text Prompt → Text Generation)
- Huggingface Pipeline, Gemma3ForConditionalGeneration

요약

- LLM Evaluation
- LLM Benchmark Datasets: World Knowledge, Commonsense Reasoning, etc.
- Evaluation Metrics: BLEU, ROUGE
- Human Evaluation: AB Test
- LLM Based Evaluation (LLM as a Judge)
- Safety Evaluation

Reference

- Deep Learning with Python
- Deep Learning for NLP and Speech Recognition
- Natural Language Processing with Transformers, O'REILLY
- Natural Language Processing with PyTorch, O'REILLY
- Deep Learning Bible, https://wikidocs.net/book/8809
- The Hugging Face Course, https://huggingface.co/course/
- 딥러닝을 이용한 자연어 처리 입문, https://wikidocs.net/book/2155
- Transformers (신경망 언어모델 라이브러리) 강좌, https://wikidocs.net/book/8056
- 한권으로 끝내는 실전 LLM 파인튜닝, 위키북스

Thank You!

SEU
Samsung Electronics University

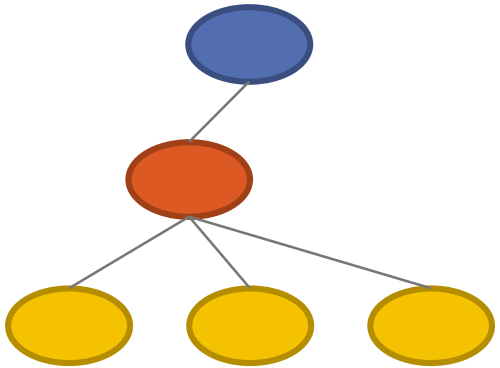
CONTENTS

Appendix

Samsung Electronics University

[Appendix] WordNet

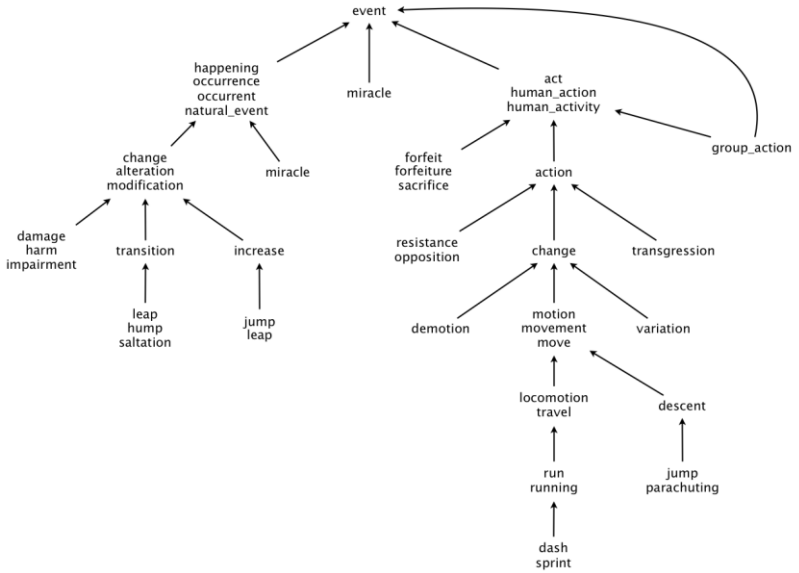
- WordNet, a thesaurus containing lists of synonym sets and hypernyms ("is a" relationships)
- https://wordnet.princeton.edu/
- Each of WordNet's 117,000 synsets is linked to other synsets by means of a small number of conceptual relations.



hypernym

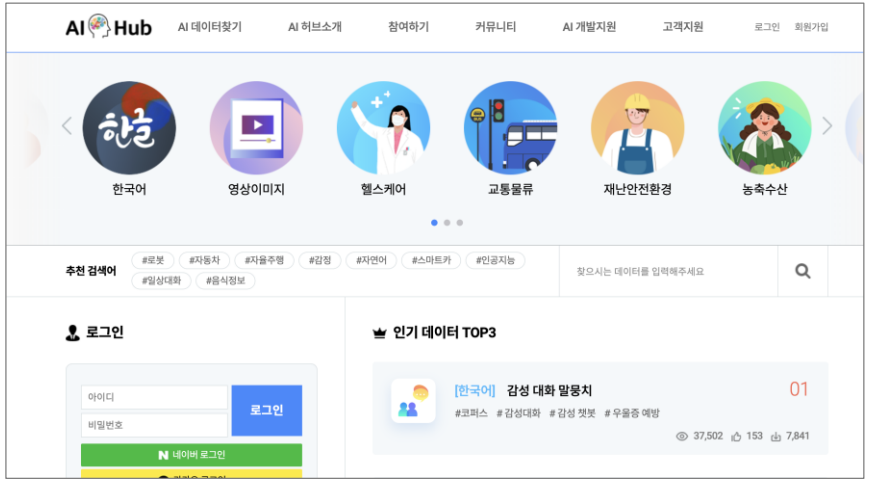
synonym

hyponym

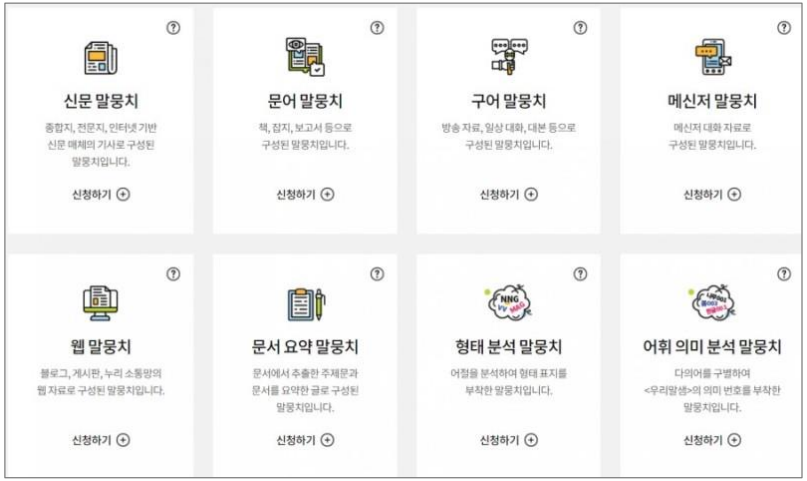


[Appendix] Korean Data Hub

- https://aihub.or.kr/
  - AI 통합 플랫폼 (한국지능정보사회진흥원)

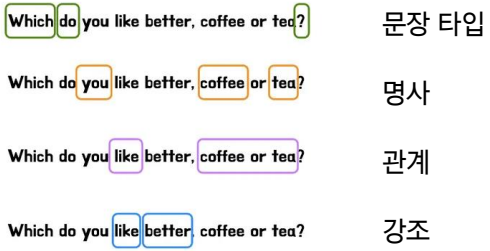


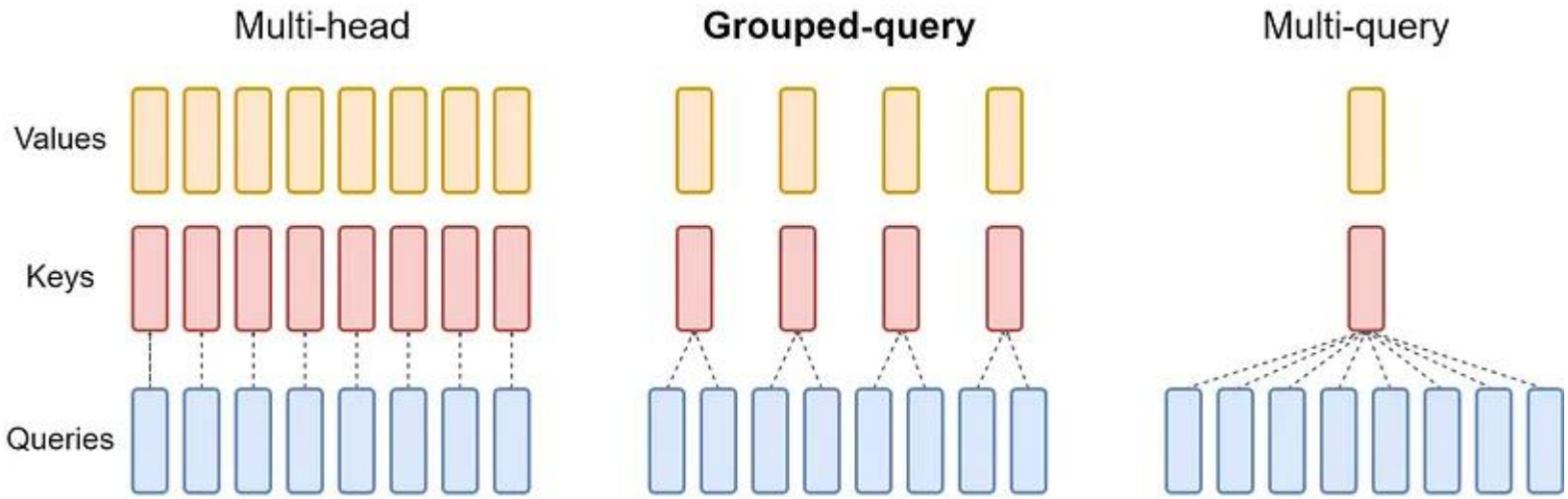
- https://corpus.korean.go.kr/
  - 문화체육관광부 국립국어원 / 한국어 코퍼스



[Appendix] GQA, MQA

- GQA (Grouped Query Attention)
- MQA (Multi-Query Attention)

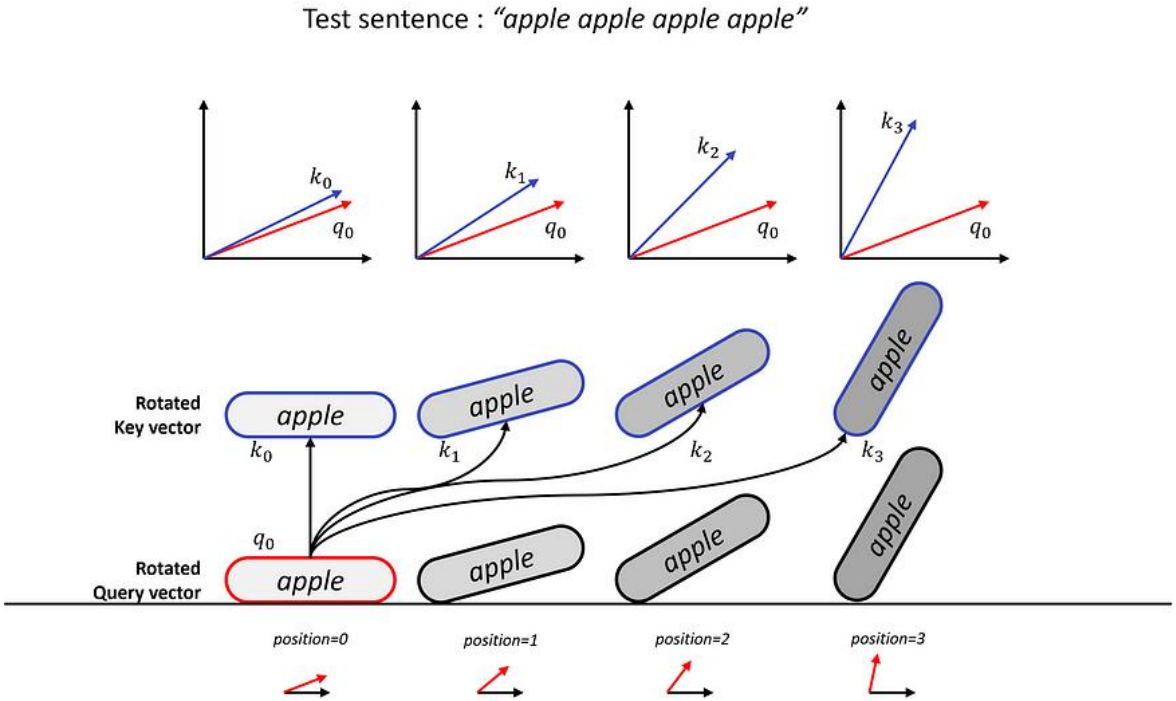




Ref. "Fast Transformer Decoding: One Write-Head is All You Need", 2019

[Appendix] RoPE

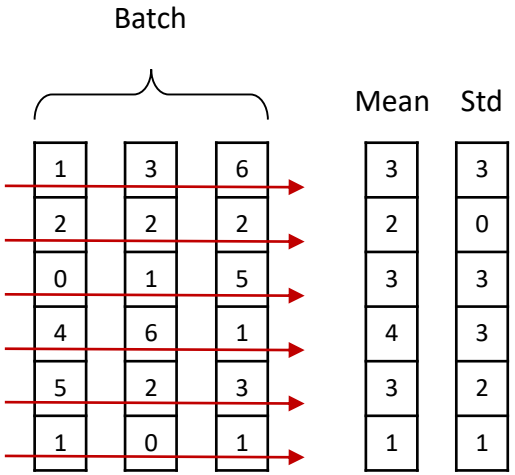
Example

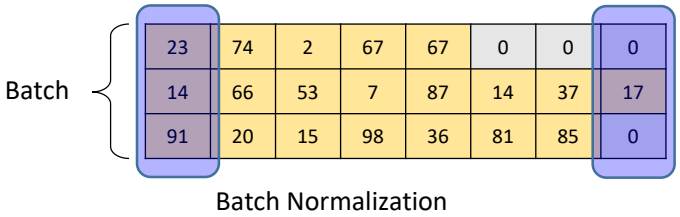


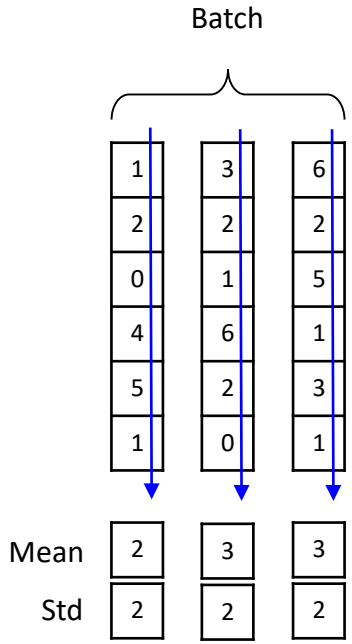
Ref. ROPEformer: Enhanced Transformer with Rotary Position Embedding (https://arxiv.org/pdf/2104.09864)

[Appendix] Layer Normalization

Batch Normalization vs Layer Normalization







$$norm_x = (x - mean)/std$$

Layer Normalization 특성

- Batch Normalization은 작은 배치 크기에서 극단적 결과를 내는데 반해, 작은 Batch Size 에서도 효과적 이용이 가능
- Sequence에 따른 고정길이 정규화로 Batch Normalization에 비해 RNN 모델에 더 효과적
- 일반화 성능 향상이 가능
- Feed Forward Network (input - hidden - output 순서로 한 방향으로 흐르는 신경망 모델)에서는 Batch Normalization 만큼 잘 동작하지 않을 수 있음
- Learning Rate, Weight Initialization 같은 하이퍼파라미터 조정에 민감

[Appendix] RMS Normalization

RMS (Root Mean Square) Normalization

- LayerNorm은 평균과 분산을 정규화하여 안정적인 학습을 유도
- RMSNorm은 평균을 사용하지 않고, RMS 만으로 정규화하여 연산을 단순화하면서도 학습 성능을 유지

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{x}_i = \frac{x_i}{\mathrm{rms}(x) + \epsilon}$$

구분	Layer Normalization	RMS Normalization
정규화	평균과 분산을 사용하여 정규화	제곱평균을 사용하여 정규화
계산비용	평균과 분산 계산이 필요하여 비용이 더 **높음**	제곱평균 계산만 필요하여 비용이 더 **낮음**
파라미터	γ (Scale Parameter), β (Shift Parameter)	γ (Scale Parameter)
배치 민감도	민감하지 않음	민감하지 않음
주요 사례	RNN과 같은 순환 신경망	대규모 신경망, 계산 비용이 낮은 모델 (Transformer)
장점	각 샘플의 평균과 분산을 고려한 정규화, **안정적인 학습 가능**	**계산 비용이 낮고** 간단한 계산으로 정규화 수행 가능
단점	평균과 분산 계산으로 인해 **계산 비용 높음**	평균을 고려하지 않기 때문에 일부 경우에서 **학습 안정성 저하**

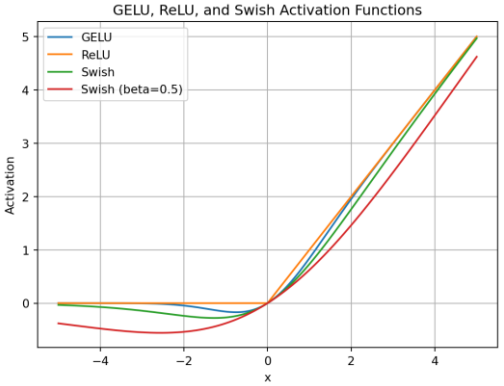
[Appendix] SwiGLU

SwiGLU (Switched Gated Linear Unit)

- GLU Variants Improve Transformer (2020, Google)
- Meta AI의 Llama Model 등 SwiGLU 채택

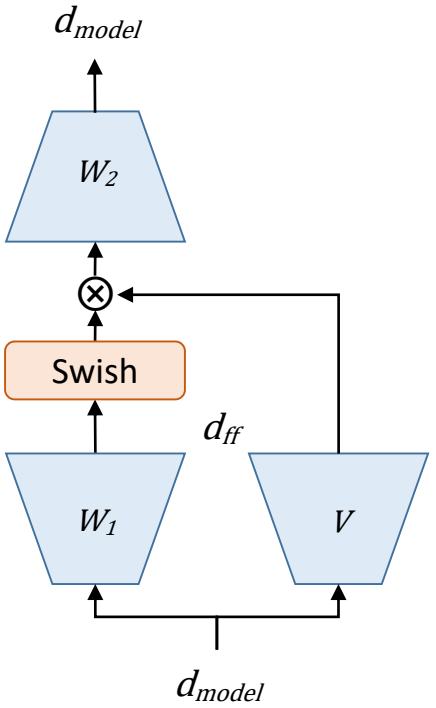
$$\mathrm{GLU}(x, W, V, b, c) = \sigma(xW + b) \otimes (xV + c)$$

$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \underbrace{\mathrm{Swish}_\beta(xW + b)}_{\text{gate}} \otimes \underbrace{(xV + c)}_{\text{information}}$$



- x: input
- W, V: 학습 가능한 텐서
- b, c: 학습 가능한 텐서 bias
- β : 학습 가능한 파라미터
- $\otimes$: element-wise multiplication

$$\mathrm{Swish}(x) = x\sigma(\beta x)$$



[Appendix] Benchmark Test (2024)

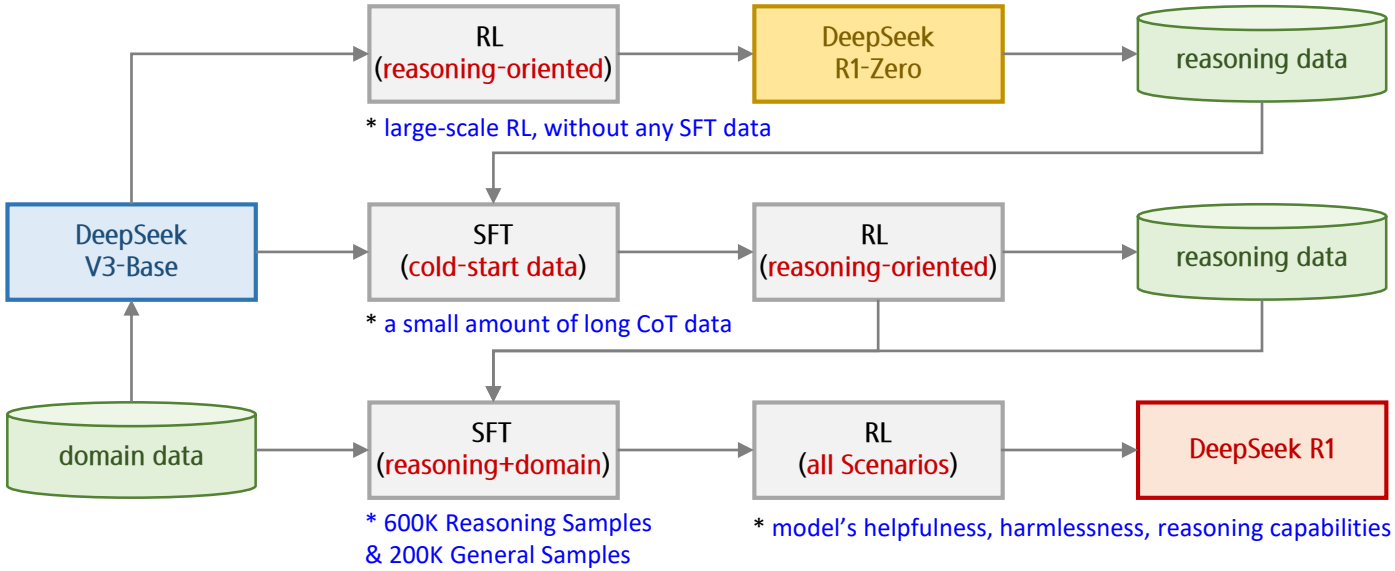
벤치마크	1 위	2 위	3 위
멀티태스크 추론 (MMLU)	GPT-4o	Llama 3.1 405B	Claude 3.5 Sonnet
코딩 (HumanEval)	Claude 3.5 Sonnet	GPT-4o	Llama 3.1 405B
수학 (MATH)	GPT-4o	Llama 3.1 405B	GPT-4-Turbo
지연 시간 (Latency)	Llama 3.1 8B	GPT-3.5-Turbo	Llama 3.1 405B
비용 (Cost)	Llama 3.1 8B	Gemini 1.5 Flash	GPT-4 Mini
컨텍스트 창 (Context Window)	Gemini 1.5 Flash	Claude 3 & 3.5	GPT-4-Turbo & GPT-4o
사실 정확성 (Factual Accuracy)	Claude 3.5 Sonnet	GPT-4o	Llama 3.1 405B
정렬 (Alignment)	Claude 3.5 Sonnet	GPT-4o	Llama 3.1 405B
적대적 프롬프트에 대한 안정성	Claude 3.5 Sonnet	GPT-4o	Llama 3.1 405B
다국어 부문	GPT-4o	Claude 3.5 Sonnet	Llama 3.1 405B
지식 보유 및 장문 생성	Claude 3.5 Sonnet	GPT-4o	Gemini 1.5 Flash
제로 및 퓨샷 학습	GPT-4o	Claude 3.5 Sonnet	Llama 3.1 405B
편견 및 독성 출력	Claude 3.5 Sonnet	GPT-4o	Llama 3.1 405B

Ref. Top Large Language Models (MarkTechPost, 2024)

[Appendix] DeepSeek (1/3)

DeepSeek R1-Zero & R1

- Post-Training: Large-Scale Reinforcement Learning on the Base Model
- Distillation: Smaller Models Can Be Powerful Too

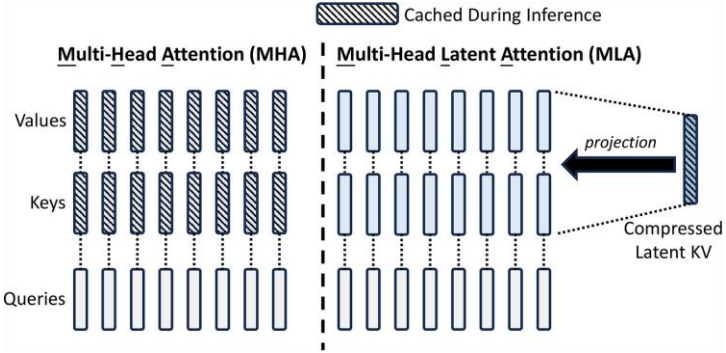


Ref. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning (DeepSeek-AI, 2025)

[Appendix] DeepSeek (2/3)

Architecture

- **MoE (Mixture of Experts)**
  : 37B of 671B parameters active per token
  : 256 Experts
- **MLA (Multi-head Latent Attention)**
  : Low-Rank Key-Value Joint Compression
  * Instead of reducing KV-heads as in MQA and GQA



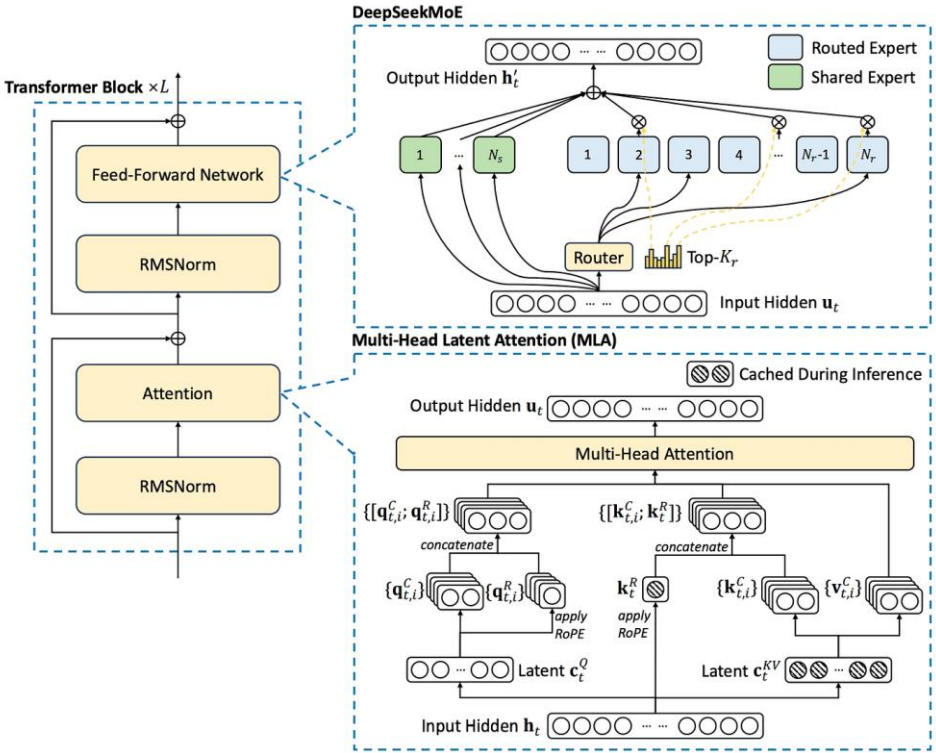


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

Ref. DeepSeek-V3 Technical Report (DeepSeek-AI, 2024)

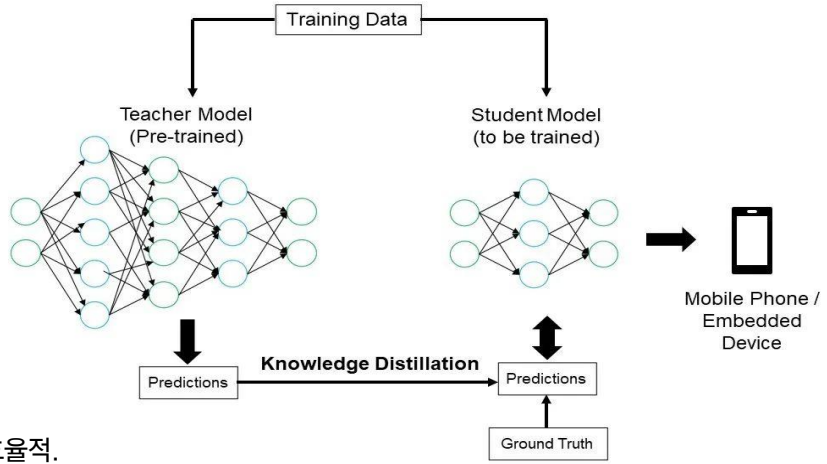
[Appendix] DeepSeek (3/3)

Reinforcement Learning

- DeepSeek-R1-Zero
  : Group Relative Policy Optimization (GRPO*)
  : Rule-based Reward System
  : Self-evolution Process
- DeepSeek-R1
  : Cold start (small amount of long CoT data)
  : Reasoning-oriented RL
  : Rejection Sampling and SFT
  : Reinforcement Learning for all Scenarios

Knowledge Distillation

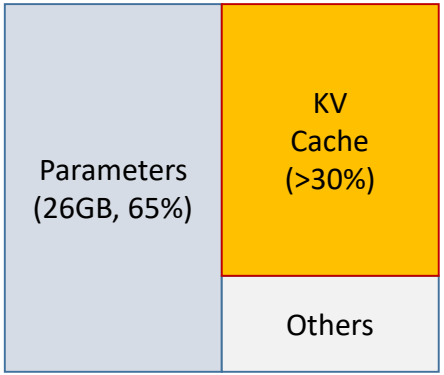
- Fine-tuned open-source models
  (Qwen2.5, Llama-3.3)
- 800k samples curated with DeepSeek-R1
- Apply only SFT and do not include an RL stage



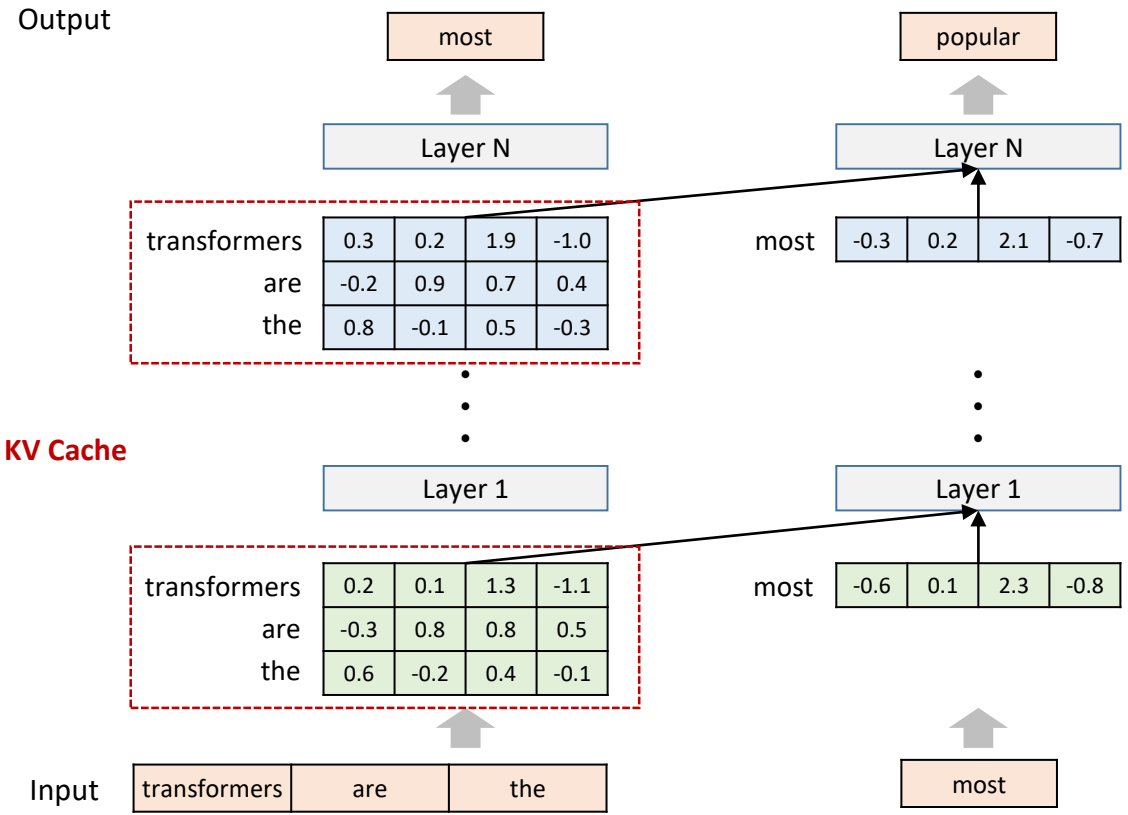
* 답변을 소수의 그룹으로 나눠 평가하는 강화학습 방식. PPO 대비 메모리 사용량이 효율적.

Ref. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning (DeepSeek-AI, 2025)

[Appendix] KV Cache



13B LLM on A100-40GB*



Ref. Efficient Memory Management for Large Language Model Serving with PagedAttention

[Appendix] LLM 개발

LLM requires a large team to develop all components.

- Data Gathering
- Data Cleaning (noise, HAP, etc)
- PII Filtering
- Deduplication
- Multi-lingual corpus
- Data governance (license, legality)

Pre-training corpus

- LLM architecture
- HP search / decision
- Training Stability
- 3D Parallel training with >1K GPUs
- GPU failure tolerance
- Long Context support

Pre-training

- Instruct Following / Plugin / RAG corpus generation
- Reasoning corpus generation
- Quality management

Fine-tune corpus

- HP search
- Loss masking
- Ghost Attention
- Evaluation / testing

Supervised Fine-tuning

- Reward signal design
- Feedback gathering
- Reward models
- RL training

Reinforcement Learning

- Orchestration (Plugin / RAG / LangChain)
- Chat mode
- REST API
- Watermark

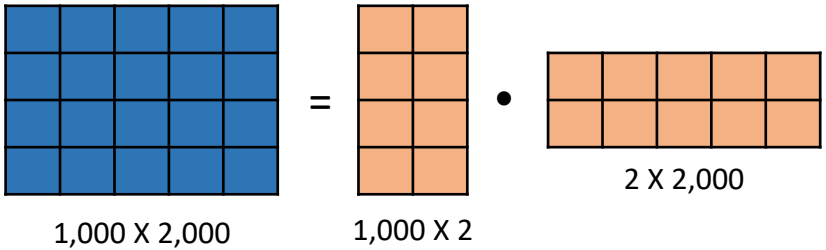
LLM Service

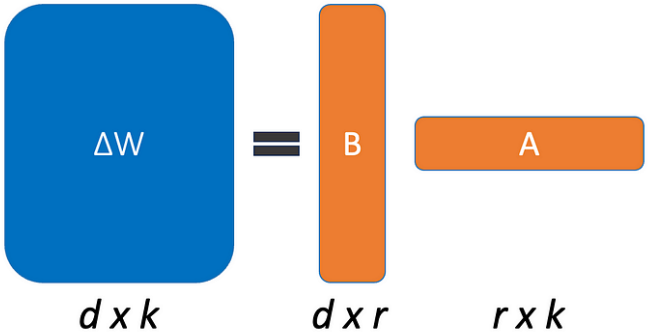
[출처] SAIF 2023

[Appendix] PEFT/LoRA

LoRA

- 각 레이어에 d × k 가중치 업데이트 행렬 ΔW
- 낮은 분해 순위 BA로 표현
- B는 d × r 행렬, A는 r × k 행렬
- 분해의 순위 r은 min(d, k)보다 훨씬 작음
- A는 무작위 가우스 숫자로, B는 0으로 초기화





Parameter	Description
r	• LoRA Adapter 파라미터의 차원 개수 • 기본값: 8 • ΔW (d x k) = B (d x r) · A (r x k)
lora_alpha	• LoRA Adapter의 Scaling값 • 기본값: 8 • ΔW는 α / r 로 스케일링 (학습율 조정과 동일)
lora_dropout	• LoRA 레이어의 Dropout 확률 • 기본값: 0
target_modules	• 모델 아키텍처에서 파인튜닝 타켓 모듈 설정 • 기본값: None
lora_bias	• Bias 학습 여부 설정 • 'none'(기본값), 'all', 'lora_only'

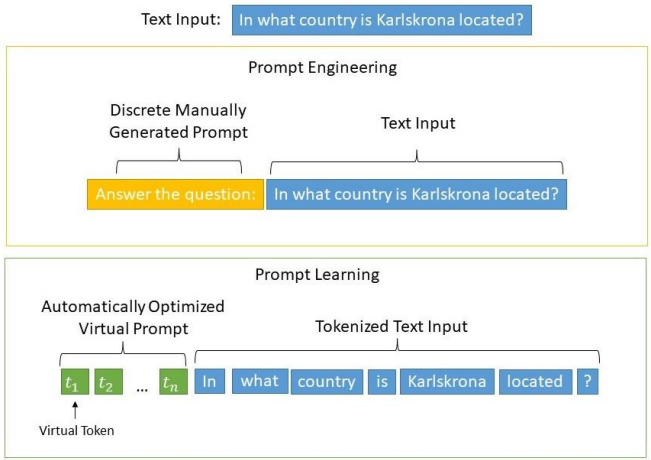
[Appendix] LLM Fine-Tuning: PEFT / P-Tuning

- [Tsinghua] P-Tuning:
  GPT Understands, Too (2023)
- https://arxiv.org/abs/2103.10385
- https://github.com/THUDM/P-tuning

GPT Understands, Too

※ GPT 계열은 NLU Task에 열세였으나, Prompting이 문제 해결에 효과적

Xiao Liu[1*], Yanan Zheng[1*], Zhengxiao Du[1], Ming Ding[1], Yujie Qian[2],
Zhilin Yang[1†], Jie Tang[1†]
[1]Tsinghua University [2]Massachusetts Institute of Technology



Abstract

Prompting a pretrained language model with natural language patterns has been proved effective for natural language understanding (NLU). However, our preliminary study reveals that manual discrete prompts often lead to unstable performance—e.g., changing a single word in the prompt might result in substantial performance drop. We propose a novel method P-Tuning that employs trainable continuous prompt embeddings in concatenation with discrete prompts. Empirically, P-Tuning not only stabilizes training by minimizing the gap between various discrete prompts, but also improves performance by a sizeable margin on a wide range of NLU tasks including LAMA

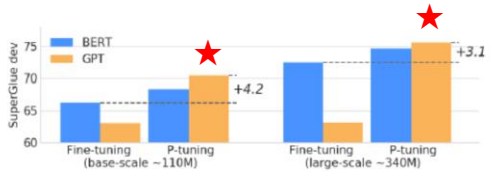


Figure 1: Average scores on 7 dev datasets of SuperGLUE using P-Tuning.

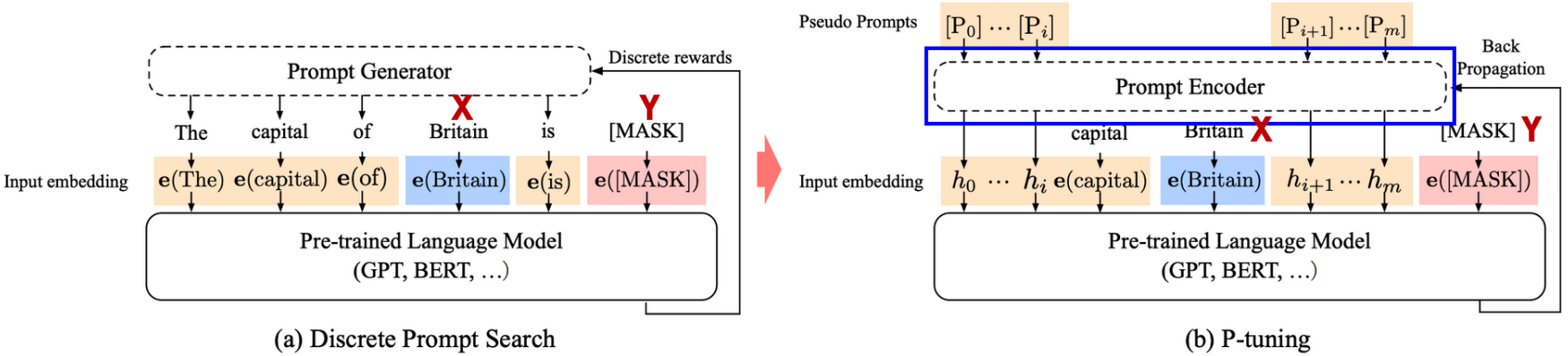
Prompt	P@1 w/o PT	P@1 w/ PT
[X] is located in [Y]. *(original)*	31.3	57.8
[X] is located in which country or state? [Y].	19.8	57.8
[X] is located in which country? [Y].	31.4	58.1
[X] is located in which country? In [Y].	51.1	58.1

Ref. Adapting P-Tuning to Solve Non-English Downstream Tasks

[Appendix] LLM Fine-Tuning: PEFT / P-tuning

- **Discrete Prompt Search: Prompt Generator (AutoPrompts)**
- **P-tuning: Prompt Encoder (LSTM)**

※ Handcrafted Prompts 방식이 아닌 LSTM 학습을 통해 자동으로 Prompt Tuning 하는 방식을 제안

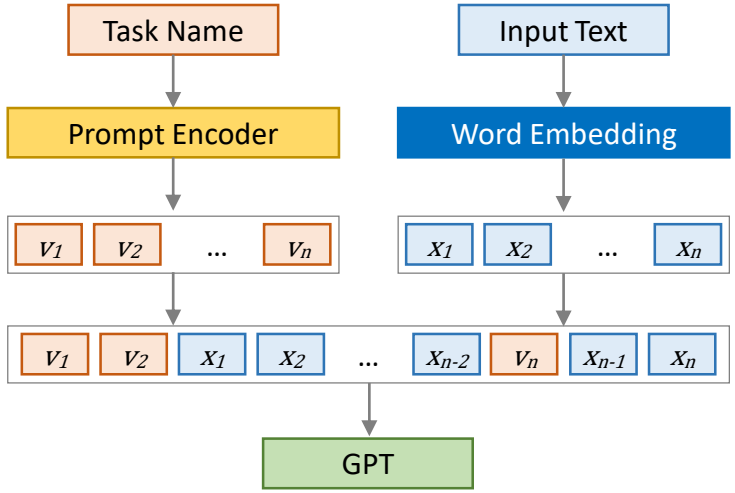


[Appendix] LLM Fine-Tuning: PEFT / P-tuning

Experiments

Prompt type	Model	P@1
Original (MP)	BERT-base	31.1
	BERT-large	32.3
	E-BERT	36.2
Discrete	LPAQA (BERT-base)	34.1
	LPAQA (BERT-large)	39.4
	AutoPrompt (BERT-base)	43.3
P-tuning	BERT-base	48.3
	BERT-large	**50.6**

Model	MP	P-tuning
BERT-base (109M)	31.7	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	45.2
BERT-large (335M)	33.5	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	40.0
RoBERTa-large (355M)	22.1	53.5 (+31.4)
GPT2-medium (345M)	20.3	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	54.4 (+31.6)
MegatronLM (11B)	23.1	**64.2** (+41.1)



※ Handcrafted Prompt 방법 대비 연속적인 공간에서 자동으로 Prompt를 검색하는 새로운 방법

* MP(Manual Prompting)

[Appendix] LLM Fine-Tuning: RLHF (1/3)

TRL (Transformer Reinforcement Learning) Pseudo Code

```
from transformers import AutoTokenizer
from trl import AutoModelForCausalLMWithValueHead, PPOConfig, PPOTrainer

config = PPOConfig(
    model_name="gpt2",
    learning_rate=1.41e-5,
)

model = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)
tokenizer = AutoTokenizer.from_pretrained(config.model_name)

reward_model = pipeline("text-classification", model="reward_model")

from trl import PPOTrainer

ppo_trainer = PPOTrainer(
    model=model,
    config=config,
    dataset=dataset,
    tokenizer=tokenizer,
)
```

Ref. https://huggingface.co/blog/rlhf

[Appendix] LLM Fine-Tuning: RLHF (2/3)

TRL (Transformer Reinforcement Learning) Pseudo Code

```
from tqdm import tqdm

for epoch in tqdm(range(epochs), "epoch: "):
    for batch in tqdm(ppo_trainer.dataloader):
        query_tensors = batch["input_ids"]

        #### Get response from SFTModel
        response_tensors = ppo_trainer.generate(query_tensors, **generation_kwargs)
        batch["response"] = [tokenizer.decode(r.squeeze()) for r in response_tensors]

        #### Compute reward score
        texts = [q + r for q, r in zip(batch["query"], batch["response"])]
        pipe_outputs = reward_model(texts)
        rewards = [torch.tensor(output[1]["score"]) for output in pipe_outputs]

        #### Run PPO step
        stats = ppo_trainer.step(query_tensors, response_tensors, rewards)
        ppo_trainer.log_stats(stats, batch, rewards)

#### Save model
ppo_trainer.save_pretrained("my_ppo_model")
```

Ref. https://huggingface.co/blog/rlhf

[Appendix] LLM Fine-Tuning: RLHF (3/3)

Reward Modeling

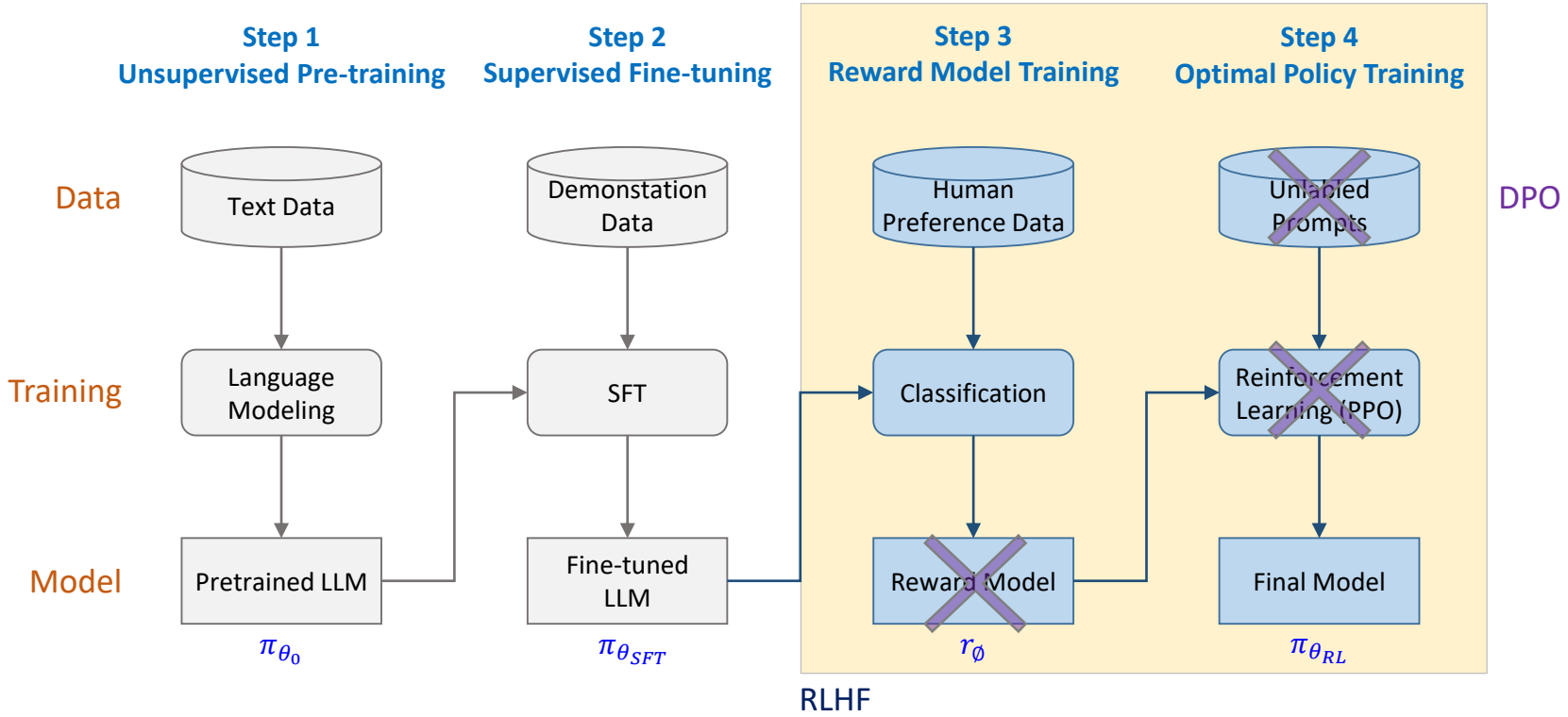
```
from trl import ModelConfig, RewardConfig,
                RewardTrainer, get_peft_config

model = AutoModelForSequenceClassification.from...

# Preprocess and filter out the raw dataset
...
train_dataset = raw_datasets["train"]
eval_dataset = raw_datasets["test"]

trainer = RewardTrainer(
    model=model,
    tokenizer=tokenizer,
    args=reward_config,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    peft_config=get_peft_config(model_config),
)

trainer.train()
```

TRL + PEFT

```
from peft import LoraConfig
from trl import AutoModelForCausalLMWithValueHead

lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
)

pretrained_model =
    AutoModelForCausalLMWithValueHead.from_pretrained(
    config.model_name,
    peft_config=lora_config,
)
```

Ref. https://huggingface.co/blog/rlhf

[Appendix] RLHF(PPO) & DPO



Ref. https://huyenchip.com/2023/05/02/rlhf.html

[Appendix] Trainer v.s. SFTTrainer

- **Trainer**: If you have a large dataset and need extensive customization for your training loop or complex training workflows.
- **SFTTrainer**: If you have a pre-trained model and a relatively smaller dataset, and want a simpler and faster fine-tuning experience with efficient memory usage.

Feature	Trainer	SFTTrainer
Purpose	General purpose training from scratch	Supervised fine-tuning of pre-trained models
Customization	Highly customizable	Simpler interface with fewer options
Training Workflow	Handles complex workflows	Streamlined workflow
Data Requirements	Larger datasets	Smaller datasets
Memory Usage	Higher	Lower with PEFT and packing optimizations
Training Speed	Slower	Faster with smaller datasets and shorter times

[Appendix] PyTorch Fine-Tuning

Fine-Tuning Steps

1. Pretrained Model, Tokenizer
2. DataLoader
3. Optimizer
4. Training Mode
5. Training Loop
6. Gradient Initialization
7. Model Inference
8. Loss
9. Backpropagation
10. Weights Update

```
from torch.utils.data import DataLoader
from transformers import AutoTokenizer, AdamW
from transformers import AutoModelForCausalLM

# 1. Pretrained Model & Tokenizer
tokenizer = AutoTokenizer(model_ckpt)
model = AutoModelForCausalLM.from_pretrained(model_ckpt)
model.to(device)

# 2. Data Loader
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)

# 3. Optimizer
optim = AdamW(model.parameters(), lr=5e-5)

# 4. Train Mode
model.train()

losses = []
```

[Appendix] PyTorch Fine-Tuning

Fine-Tuning Steps

1. Pretrained Model, Tokenizer
2. DataLoader
3. Optimizer
4. Training Mode
5. Training Loop
6. Gradient Initialization
7. Model Inference
8. Loss
9. Backpropagation
10. Weight Update

```
for epoch in range(8):                  # 5. Training Loop
    print(f'epoch:{epoch}')
    for batch in train_loader:
        optim.zero_grad()               # 6. Gradient Initialization

        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        outputs = model(input_ids,      # 7. Model Inference
                        attention_mask=attention_mask, labels=labels)

        loss = outputs[0]               # 8. Loss
        losses.append(loss)

        loss.backward()                 # 9. Backpropagation

        optim.step()                    # 10. Weight Update

model.eval()
```

[Appendix] Trainer, TrainingArguments

Trainer

Parameter	Description
model	PreTrainedModel *# the model to train, evaluate or use for predictions*
args	TrainingArguments *# the arguments to tweak for training*
train_dataset	Dataset, IterableDataset *# the dataset to use for training*
eval_dataset	Dataset *# the dataset to use for evaluation*
compute_metrics	Callable[[EvalPrediction], Dict] *# the function that will be used to compute metrics at eval.*
optimizers	Tuple[Optimizer, LambdaLR] *# a tuple contatining the optimizer and the scheduler to use*
callbacks	List of TrainerCallback *# a list of callbacks to customize the training loop*
data_collator	DataCollator *# the func. to use to form a batch from train/eval dataset*
tokenizer	PreTrainedTokenizerBase *# the tokenizer used to preprocess the data*
model_init	Callable[[], PreTrainedModel] *# a function that instantiated the model to be used*

TrainingArguments

Parameter	Description
output_dir	str, the output directory *# output directory*
num_train_epochs	float, defaults to 3.0 *# total number of training epochs*
per_device_train_batch_size	int, optional, defaults to 8 *# batch size per device during training*
per_device_eval_batch_size	int, optional, defaults to 8 *# batch size per device for evaluation*
learning_rate	float, defaults to 5e-5 *# learning rate for optimizer*
eval_strategy	str or IntervalStrategy (“no”,“steps”,“epoch”) *# evaluation strategy during training*
gradient_accumulation_steps	int or float, defaults to 1 *# total number of steps before back propagation*
warmup_steps	int, defaults to 0 *# number of warmup steps for scheduler*
weight_decay	float, defaults to 0 *# strength of weight decay*
fp16 / bf16	bool, defaults to False *# use mixed precision*

Ref. https://huggingface.co/docs/transformers/v4.45.2/en/main_classes/trainer#transformers.Trainer

[Appendix] SFT Trainer, LoRA Config

SFT Trainer

Parameter	Description
model	PreTrainedModel *# the model to train, can be a PreTrainedModel*
args	SFTConfig *# the arguments to tweak for training*
train_dataset	Dataset, IterableDataset *# the dataset to use for training*
eval_dataset	Dataset *# the dataset to use for evaluation*
compute_metrics	Callable[[EvalPrediction], Dict] *# the function to use to compute the metrics at evaluation*
optimizers	Tuple[Optimizer, LambdaLR] *# the optimizer and scheduler to use for training*
callbacks	List of TrainerCallback *# the callbacks to use for training*
data_collator	DataCollator *# the data collator to use for training*
peft_config	PeftConfig *# the PeftConfig object to use to initialize the PeftModel*
formatting_func	Callable *# the formatting func. for creating ConstantLengthDataset*

LoRA Config

Parameter	Description
r	Int = 8, *# Lora attention dimension ("rank")*
lora_alpha	Int = 8, *# the alpha parameter for Lora scaling.*
lora_dropout	float = 0.0, *# the dropout probability for Lora layers*
bias	Literal['none', 'all', 'lora_only'] *# the correspond. biases will be updated during training.*
target_modules	[Union[List[str], str]] *# The names of the modules to apply the adapter to*
task_type	[Union[str, peft.utils.peft_types.TaskType, NoneType] *# the task type*
init_lora_weight	bool | Literal['gaussian', 'olora', ...] *# how to initialize the weights of the adapter layers*
lora_bias	bool , default to False *# whether to enable the bias term for the LoRA B param.*

Ref. https://huggingface.co/docs/peft/package_reference/lora

[Appendix] DPO Trainer, DPO Config

DPO Trainer

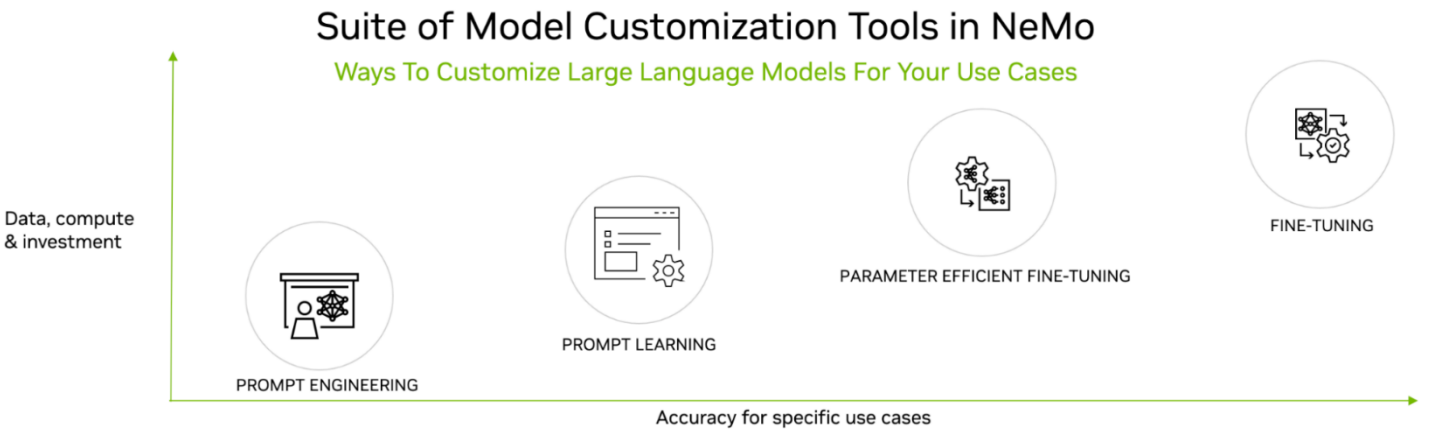
Parameter	Description
model	PreTrainedModel *# the model to train, AutoModelForSequenceClassification*
ref_model	PreTrainedModelWrapper *# the model used for implicit reward computation and loss.*
args	DPOConfig *# the DPO config arguments to use for training*
data_collator	DataCollator, *# the data collator to use for training*
train_dataset	Dataset, *# the dataset to use for training*
eval_dataset	Dataset, *# the dataset to use for evaluation*
model_init	Callable[], *# the model initializer to use for training*
optimizer	Optimizer *# the optimizer and scheduler to use for training*
peft_config	Dict, defaults to None *# the PEFT configuration to use for training*
compute_metrics	Callable[[EvalPrediction], Dict] *# the function to use to compute the metrics at evaluation*

DPO Config

Parameter	Description
learning_rate	float = 1e-06 *# Initial learning rate for AdamW optimizer*
beta	float = 0.1, *# parameter controlling the deviation from ref.*
loss_type	str (type of loss to use), defaults to "sigmoid“ *# type of loss to use*
fp16 / bf16	bool = False *# use mixed precision*
per_device_train_batch_size	int = 8 *# batch size per device during training*
per_device_eval_batch_size	int = 8 *# batch size per device for evaluation*
output_dir	str *# output directory*
num_train_epochs	float, defaults to 3.0 *# total number of training epochs*

Ref. https://huggingface.co/docs/trl/main/dpo_trainer

[Appendix] LLM Customization

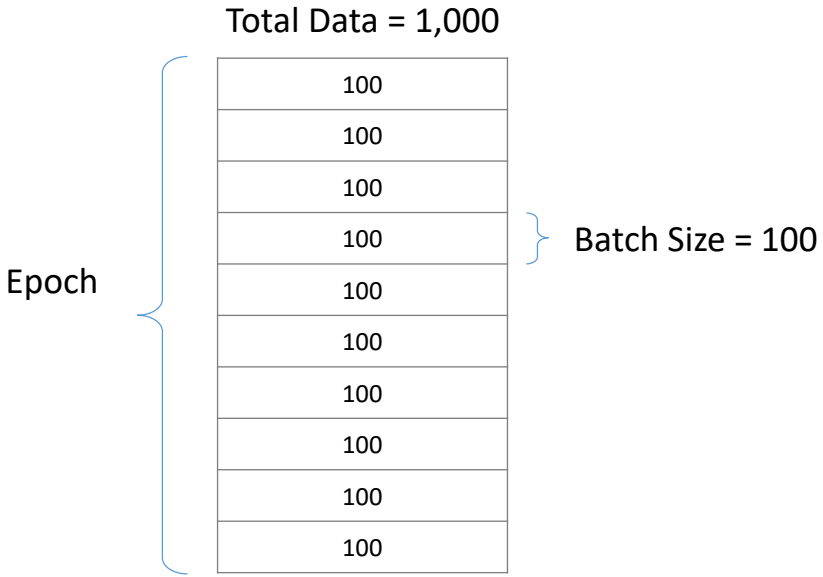


TECHNIQUES	• Few-shot learning • Chain-of-thought reasoning • System prompting	• Prompt tuning • p-tuning	• Adapters • LoRA • IA3	• SFT • RLHF
HOW	Prompt templates	Tune companion model	Add "custom" layers and parameters to LLM	Tune LLM model weights
BEST TO USE WITH	Limited prompt-completion examples & simple use cases	A few hundred examples & use cases where prompt engineering is not sufficient	Hundreds of examples for a multitude of downstream tasks	Thousands of examples & complex use cases
EXAMPLE	Extractive Q+A	Named entity extraction	Summarization, paraphrasing, etc.	Multilingual, biomedical, coding, etc.

Ref. LLM customization techniques available with NVIDIA NeMo

[Appendix] Epoch, Step, Batch Size

Training DataSet



- Epoch: 전체 데이터를 1회 학습하는 것
- Step (or Iteration): 파라미터를 1회 업데이트하는 것
- Batch Size: Step 1회에서 사용하는 데이터 개수

- 전체 데이터: 1,000
- Batch Size: 100
- 한 Epoch에 10번의 Step이 필요

- 배치: 전체 데이터셋을 한번에 처리
- 미니배치: 데이터셋을 여러 작은 배치로 나누어 처리

[Appendix] LLM: Distributed Training

Data Parallelism (DP)

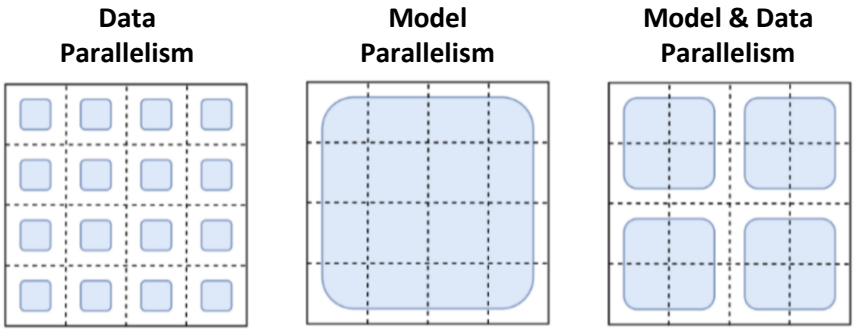
- 데이터를 나눠 여러 GPU에서 연산한 후 각 최종 업데이트 결과에 대해 평균을 내는 방식
- Synchronization: N개의 GPU에서 연산한 업데이트 결과를 모아서 평균을 내고 다시 재분배

Model Parallelism (MP)

- 여러 GPU에 모델 Parameter를 나누어 연산
- Tensor Parallelism: Weight Matrix를 나누어 여러 GPU에서 연산 후 결과를 Concatenation
- Pipeline Parallelism: Model의 layer를 여러 GPU에 쪼개서 순차적으로 학습

Multi-GPU Training

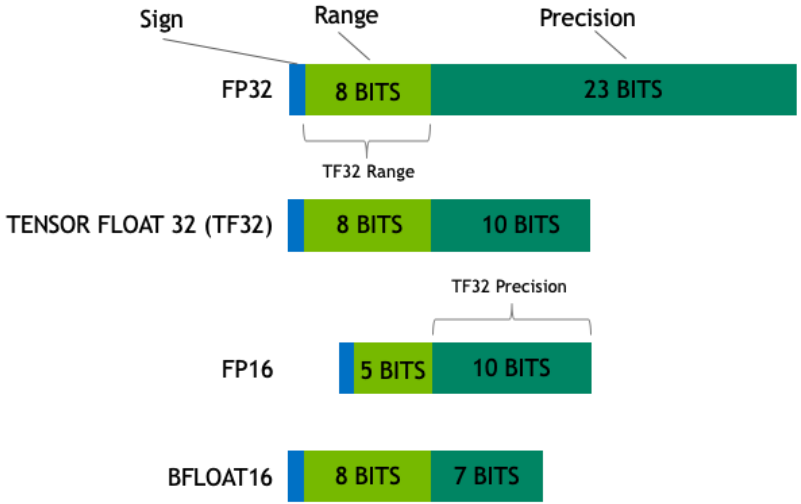
- Accelerator (Huggingface Library)
- DeepSpeed (MS Library for speed and scale)
  : Optimizer State Partitioning + Gradient Partitioning + Parameter Partitioning

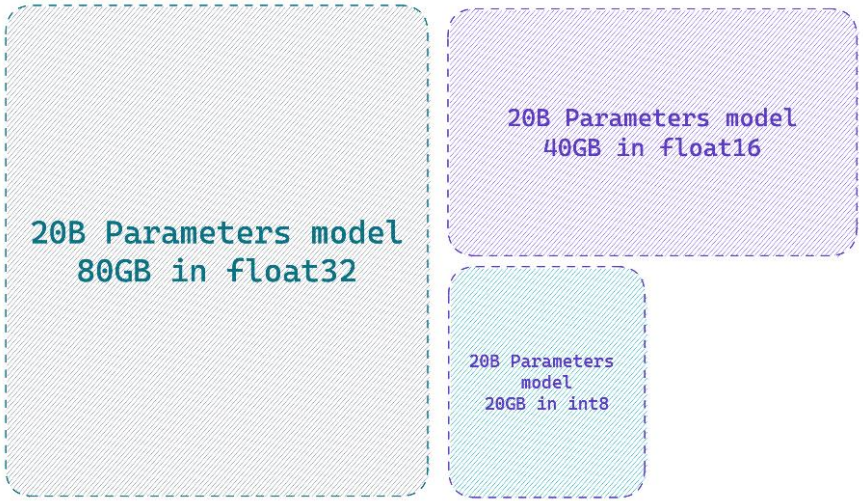


Ref. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM

[Appendix] LLM: Data Types

- **Common data types used in Machine Learning**
  - FLOAT 32, FLOAT 16 or BFLOAT 16

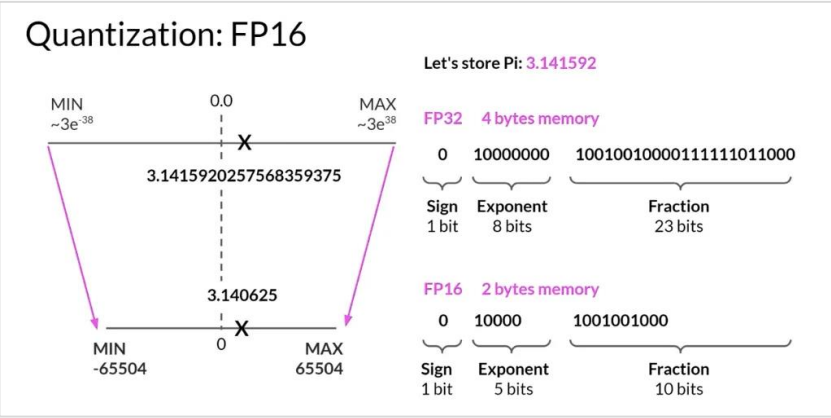




[Appendix] LLM: Quantization

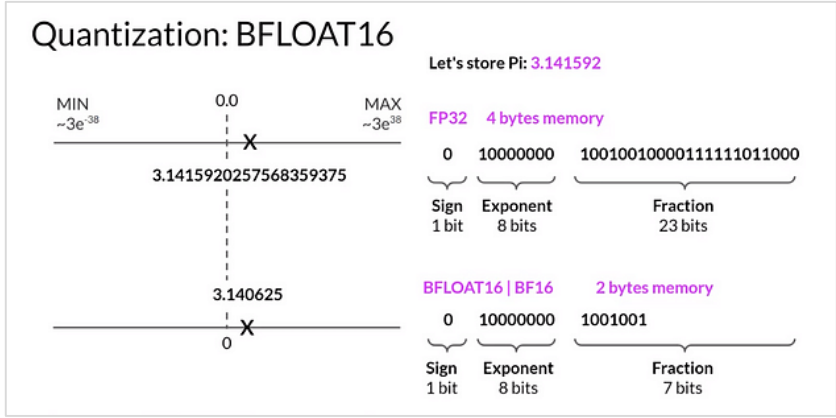
Quantization: FP16

By employing 16-bit half precision (FP16), the memory requirement can be reduced by 50. Representing the model parameters as 8-bit integers (INT8) can reduce the memory footprint even further representing a total 98.75% reduction compared to full 32-bit precision.



Quantization: BFLOAT16

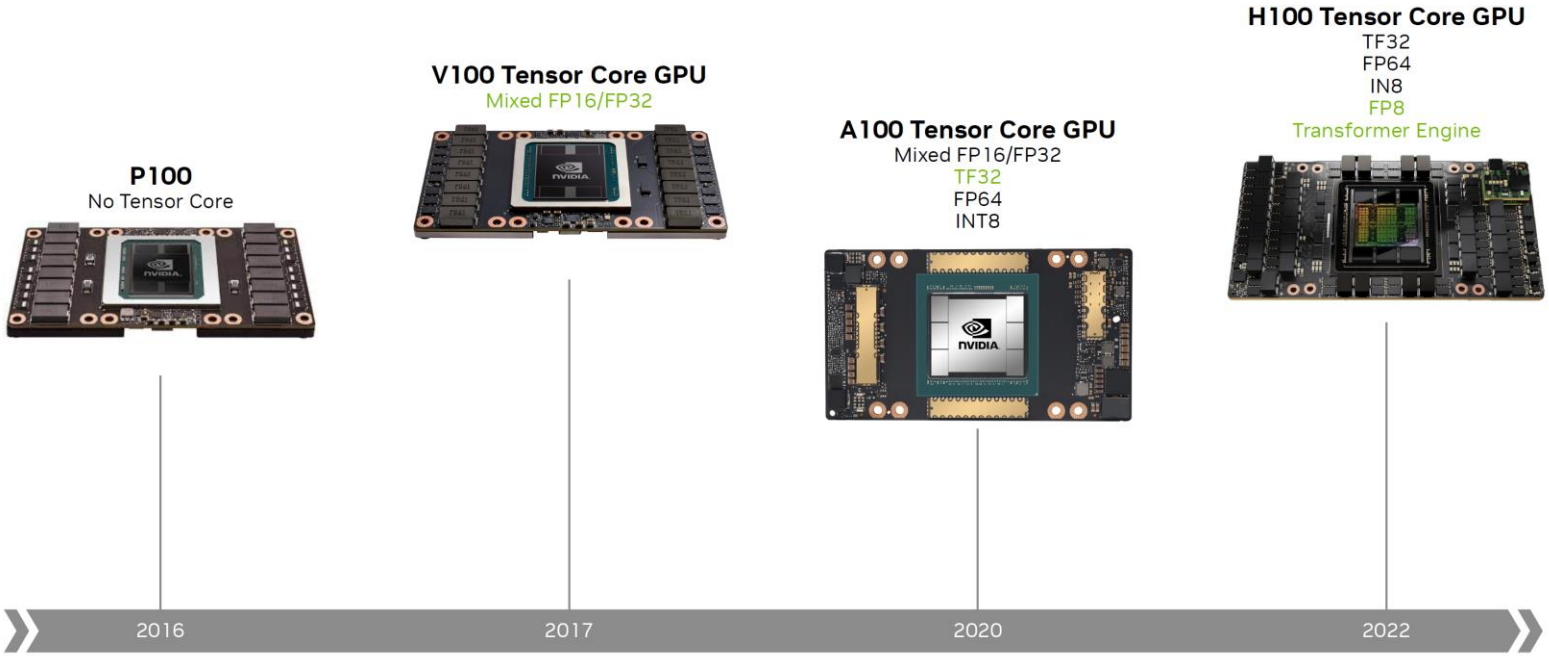
Developed by Google Brain, BF16 serves as a hybrid between FP16 and FP32, capturing the full dynamic range of FP32 with just 16 bits. This format offers improved training stability and is supported by newer GPUs, such as NVIDIA's A100. While not suitable for integer calculations, BFLOAT16 has become a popular choice in LLMs.



Ref. https://www.coursera.org/learn/generative-ai-with-llms

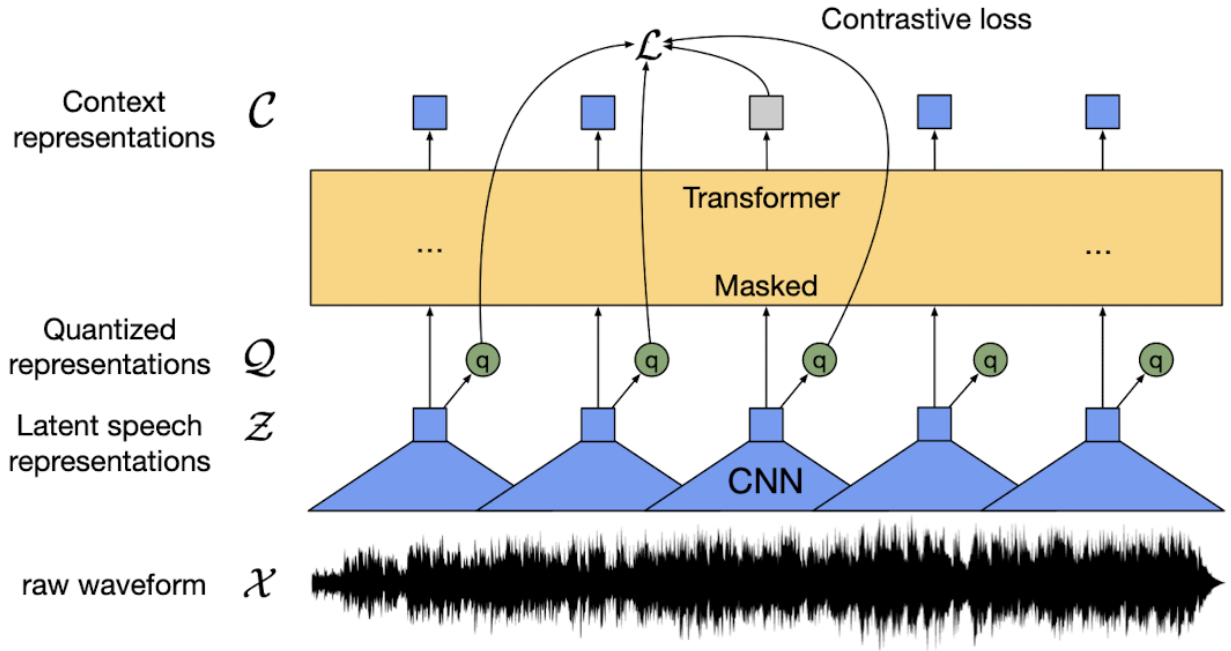
[Appendix] GPU

Tensor Core



[Appendix] Wav2Vec

Wav2Vec Architecture



Ref. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations (arXiv:2006.11477)