Training LLM

Train Neural Network for Language Model

- **Probability of a Word Sequence (주어진 sequence가 나올 확률) :**
  - Given a sequence of words $W = w_1, w_2, \dots, w_t$,
  - $P(W) = P(w_1, w_2, \dots, w_t) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \cdots P(w_t|w_1, w_2, \dots, w_{t-1})$
- **Train the network Θ to maximize log-likelihood $log\ P(W; \Theta)$:**
  - $\log P(W; \Theta) = \log P(w_1; \Theta) + \log P(w_2|w_1; \Theta) + \cdots + \log P(w_t|w_1, w_2, \cdots, w_{t-1}; \Theta)$
  - 각 log-term은 지난 단어들 다음에 현재 단어가 나올 log-probability
- **cross-entropy-loss를 minimize하는 것으로 위 목적 달성**
  - NN으로 n-gram 한계를 압도적으로 뛰어 넘음.
- **perplexity를 minimize하는 것과 같음**
  - $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}}$

Training GPTs

1

Large datasets?

GPT-1 (2018)
Dataset: BooksCorpus
Size: 5GB

GPT-2 (2019)
Dataset: WebText
Size: 40GB

GPT-3 (2020)
Dataset: Common Crawl, WebText2, BookCorpus, Wikipedia, Other large text datasets
Size: 570 GB of text data

LLaMA 1 (2023)
Dataset: Common Crawl, C4, Wikipedia, BooksCorpus, GitHub, Stack Exchange
Size: 1.4 trillion tokens

LLaMA 2 (2023)
Dataset: Common Crawl, Research papers and academic content, Code-related datasets, Books, Wikipedia, News articles, Multilingual data
Size: 2 trillion tokens

LLaMA (2024)
Dataset: Not disclosed
Size: 15 trillions, LLaMA 2의 7배

Maintaining Web Scale Datasets

Common Crawl maintains a free, open repository of web crawl data that can be used by anyone.

Over 250 billion pages spanning 18 years.

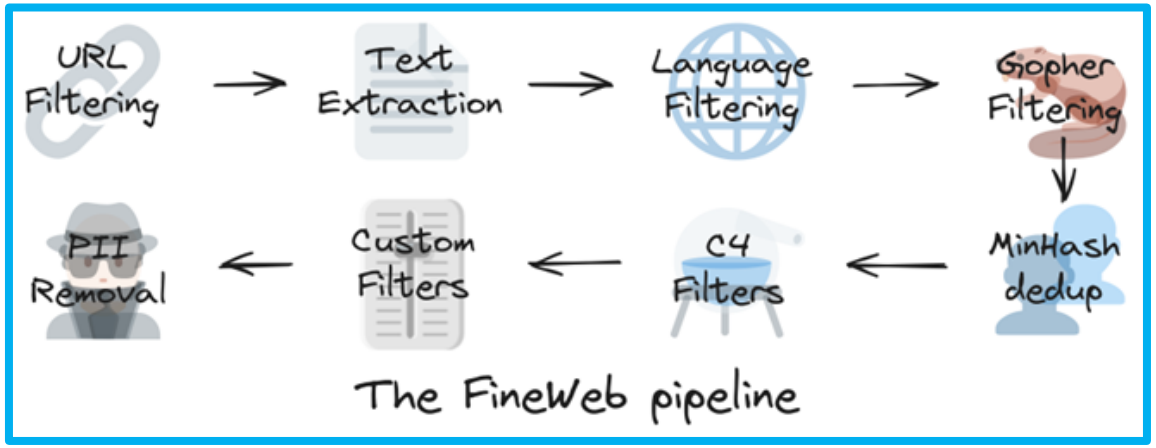
Free and open corpus since 2007.

3–5 billion new pages added each month.

🍷 **FineWeb:**

decanting the web for the finest text data at scale

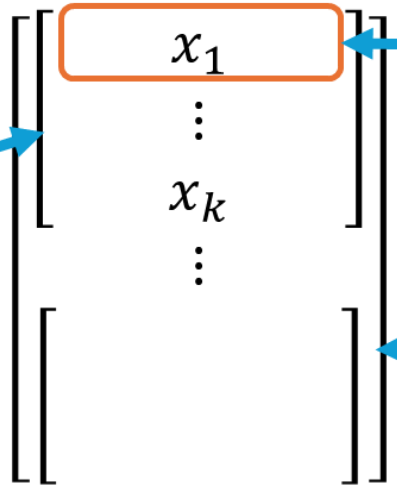
15-trillion tokens, 44TB disk space



Training Input Tensor: batch-seq-dim

We can also use the existing language functionality in the model to perform sentiment analysis. For the Stanford Sentiment Treebank dataset, which consists of sentences from positive and negative movie reviews, we can use the language model to guess whether a review is positive or negative by inputting the word "very" after the sentence and seeing whether the model predicts the word "positive" or "negative" as more likely. This approach, without adapting the model at all to the task, performs on par with classic baselines ~80% accuracy.

context window is a chunk of tokens from training data



embedding vector for each token in a context window

minibatch is a group of context windows

GPT-1: $X = [64 \times 512 \times 768]$

model	number of tokens	embedding dimension	context window (tokens in a batch)	number of batch
GPT-1	40,458	768	512	64
GPT-2	50,257	1,600	1024	512
GPT-3	50,257? (unknown)	12,288	2048	1600? 3.2M in tokens

2

How to Train LLM – InstructGPT

Difficulties in Training LLMs

- **Computational Requirements**: High Compute Demand, Extended Training Time
- **Memory Constraints**: GPU Memory Limitations
- **Large-Scale Data Handling:** Data Size, Date Quality
- **Optimization Challenges:** Gradient Instability, Hyperparameter Sensitivity
- **Communication Overhead:** Distributed Training Bottlenecks
- **Cost and Resource Management:** Financial Costs, Resource Allocation
- **Engineering Complexity:** Infrastructure Setup, Debugging Difficulty
- **Ethical and Legal Considerations:** Data Privacy, Bias and Fairness
- **Evaluation and Validation:** Performance Metrics, Generalization

Instruct GPT (2022, OpenAI)

- Pretrained LLMs은 사람의 명령이나 요구에 부합하지 않는 생성형 모델
  - 'untruthful', 'toxic', 'not helpful' to the user
  - 이를 'not *aligned* with users' 라고 말한다.
- Users에 aligned 시키기 위한 일련의 노력 결과: Instruct GPT
  - Supervised Fine Tuning을 위한 Prompt Dataset 작성 및 수집
  - Reinforcement Learning Fine Tuning을 위한 Reward Model 학습
- InstructGPT – 1.3B parameters로 학습한 결과
  - Human evaluation에서 GPT-3 175B 결과보다 좋게 평가
  - truthfulness 가 향상, toxic output 감소
  - 이 논문 발표이후 LLM 은 이 방향으로 연구되고 있음

Overall Process

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

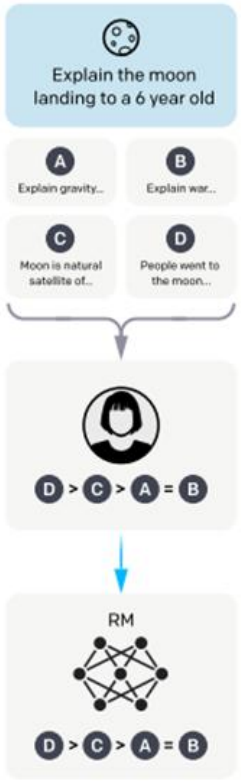
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

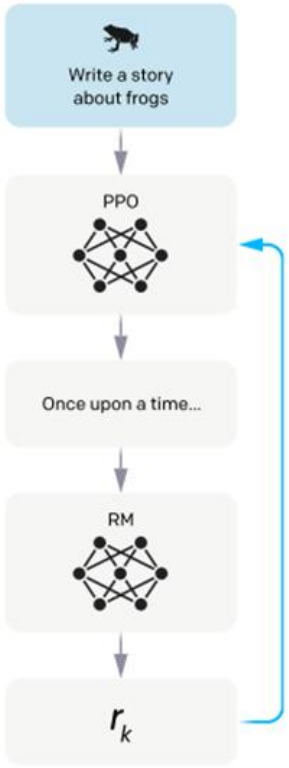
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Dataset Sources for Fine-tuning

1. **OpenAI API에 submitted된 prompt dataset:**
   - GPT-3 Playground에서 사용자들이 API를 통해서 질문한 prompts를 데이터로 활용 – 한 user ID 당 최대 200 prompts
   - Deduplication 후에 사용
2. **Labeler를 고용해서 데이터 생성**
   - Plain: labeler에게 자유롭게 prompt를 생성하도록 요청
   - Few-shot: one instruction, multiple query/response pairs of instruction 작성 요청.
   - User-based: OpenAI API에 waitlists에 있는 user-cases를 labeler가 직접 prompts를 작성하도록 요청

Data Distribution & Example

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

Human Evaluations of InstructGPT

Dataset Sizes

1. **SFT data** (labeler dataset 포함)
2. **RM data** (Reward Model) dataset (labeler가 ranking을 매김)
3. **PPO data** (사람의 간섭이 없이 Reinforcement Learning으로 학습)

Table 6: Dataset sizes, in terms of number of prompts.

SFT Data			RM Data			PPO Data		
split	source	size	split	source	size	split	source	size
train	labeler	11,295	train	labeler	6,623	train	customer	31,144
train	customer	1,430	train	customer	26,584	valid	customer	16,185
valid	labeler	1,550	valid	labeler	3,488			
valid	customer	103	valid	customer	14,399			

Human Evaluations of InstructGPT

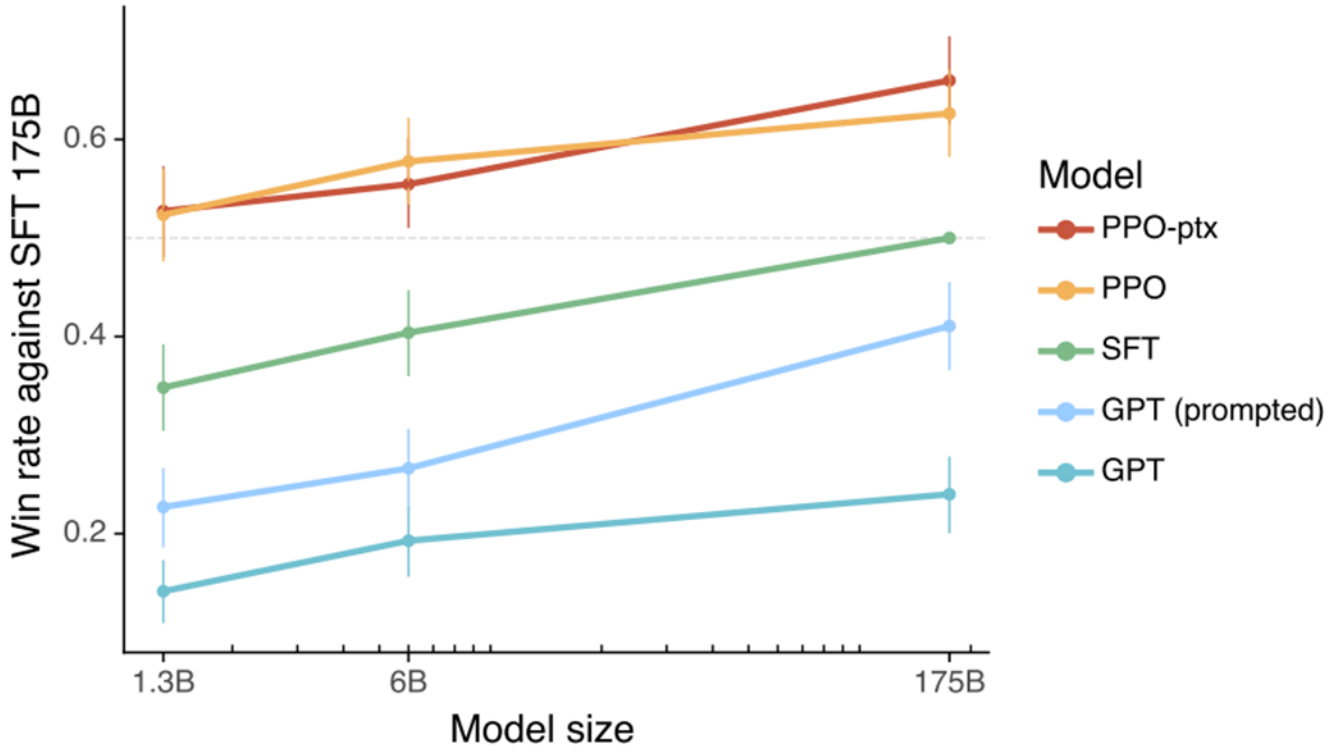


Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

3

PEFT (Parameter Efficient Fine Tunning)

PEFT (Parameter-Efficient Fine-Tunning) 이란?

- PEFT는 거대한 사전 학습 모델(Pre-trained Model)의 모든 파라미터(가중치)를 전부 미세 조정하는 대신, 아주 작은 일부의 파라미터만 수정하거나 추가하여 모델을 특정 작업에 맞게 조정하는 기법을 통칭

- 주요 목표:
- 계산 자원 절약: GPU 메모리 사용량과 학습 시간을 크게 줄임
- 저장 공간 효율화: 전체 모델(수십~수백 GB)을 복사할 필요 없이, 추가된 작은 파라미터(수십~수백 MB)만 저장
- 망각 방지: 기존 모델의 지식은 그대로 유지하면서 새로운 작업에 대한 지식만 추가하므로, 원래 모델의 성능 저하 방지

PEFT 기법 종류

- **어댑터 (Adapters):** 트랜스포머 블록 사이에 작은 신경망 모듈을 추가하여 학습하는 방식.
- **프롬프트 튜닝 (Prompt Tuning):** 모델의 입력 프롬프트에 학습 가능한 가상 토큰(soft prompt)을 추가하는 방식. (가상의 token을 주입하고 그에 해당하는 임베딩 벡터를 학습)
- **프리픽스 튜닝 (Prefix Tuning):** 어텐션 블록의 Key와 Value에 학습 가능한 벡터 시퀀스(prefix)를 추가하는 방식. (특정 task에 적합한 가상의 KV cache를 학습한다고 상상해도 좋음)
- **LoRA(Low-Rank Adaptation):** 기존 모델의 전체 가중치(weight)를 모두 업데이트하는 대신, 훨씬 작은 크기의 추가 파라미터만 학습하여 시간과 메모리 사용량을 크게 줄이는 방식.

LoRA (Low-Rank Adaptation)

- **LoRA의 핵심 아이디어는 '저차원 행렬 분해(Low-Rank Decomposition)'에 기반**
- 사전 학습된 거대 모델의 가중치 행렬(W)은 이미 다양한 특징을 학습했기 때문에, ① 새로운 작업에 맞게 미세 조정할 때 가중치의 변화량(ΔW)은 본질적으로 낮은 '내재적 차원(intrinsic rank)'을 가질 것이라는 가설에서 출발 ② 즉, 방대한 파라미터 전체가 아닌, 일부 핵심적인 부분만 수정해도 충분히 새로운 작업을 학습할 수 있다는 가정에 기반한 파인 튜닝(fine-tunning) 방법

LoRA 이론

- **LoRA는 이 변화량(ΔW)을 두 개의 작은 행렬, 저차원(low-rank) 행렬A와 B의 곱(BA)으로 근사화**
- **원래의 가중치 행렬: $W \in R^{d \times k}$**
- **변화량 행렬: $\Delta W \in R^{d \times k}$**
- **LoRA 행렬: $A \in R^{d \times r}$ 와 $B \in R^{r \times k}$**

$$W' = W + \Delta W = W + AB$$

여기서 r은 랭크(rank)를 의미하며, 원래 행렬의 차원인 d나 k보다 훨씬 작은 값($r \ll d, k$) ① 예를 들어, 1024x1024 크기의 행렬을 미세 조정할 때 랭크를 8로 설정하면, ② 기존에는 1024x1024 ≈ 100만 개의 파라미터를 업데이트 ③ LoRA를 사용하면 (1024x8)+(8x1024) ≈ 1.6만 개의 파라미터만 학습

트랜스포머 블록 수정

- 트랜스포머의 어느 블록 가중치 행렬에 LoRA를 적용할 것인가? LoRA는 주로 Query, Key, Value을 계산하는 선형 레이어의 가중치에 적용

① 기존 가중치 고정: 사전 학습된 원래 모델의 가중치 행렬(W)은 학습 과정에서 변경되지 않도록 동결(freeze)
→ 재앙적 망각을 방지

② LoRA 행렬 추가: 원래의 가중치 행렬 옆에 새로운 학습 가능한 LoRA 행렬 A와 B를 병렬로 추가

③ 결과 결합: 입력 텐서(x)가 들어오면, 원래의 가중치(Wx)와 LoRA 행렬을 통과한 결과(Bax)를 서로 더하여 최종 출력을 계산

$$h = xW + xAB = x(W + AB)$$

이 구조 덕분에 학습 시에는 A와 B만 업데이트하고, 추론(inference) 시에는 W'=W+AB를 미리 계산하여 하나의 행렬로 합칠 수 있음 → 추론 시 추가적인 계산 지연(latency)이 거의 발생하지 않는 큰 장점

멀티 LoRA(Multiple Adapters)

A. 단일 로딩(하나만 선택해서 사용)

- 가장 단순한 방법
- Inference 시점에 원하는 LoRA만 로드해서 사용

B. 멀티 LoRA 병합(Merging)

- 두 개 이상의 LoRA 파라미터를 원본 모델 가중치에 병합하여 하나의 새로운 모델로 만들어 사용
- 예: 번역 + 요약 LoRA를 합쳐서 번역 요약 전문 모델을 만들 수 있음

C. 런타인 조합(On-the-fly composition)

- 최근 프레임워크(PeFT, Hugging Face Transformers 등)는 여러 LoRA를 동시에 로드하고, 상황에 따라 가중치를 섞어서 사용

D. Router/Selector 기반 활용

- 여러 LoRA를 동시에 메모리에 올려두고,입력에 따라 적합한 LoRA를 자동 선택하는 방식.
- 예: 의학 전문 LoRA, 법률 전문 LoRA, 일반 LoRA