LLaMA: Open and Efficient Foundation Language Models

AI 키워드	
날짜	@2024년 11월 26일
콘텐츠	논문
태그	유런

Abstract

7B에서 65B의 파라미터를 가진 모델들의 집합

trillions of tokens

공개 데이터셋으로 sota 달성

대부분의 벤치마크에서 LLaMA-13B는 GPT3 넘어섬

LLaMA-65B는 최고 성능 모델들(Chinchilla-70B and PaLM-540B)과 견줄만함

모든 모델 공개

Introduction

few shot properties(=여러 다른 태스크를 수행할 수 있는 능력): 모델과 데이터 크기가 충분히 크면 나타남

→ 모델 확장에 집중하게 됨

하이퍼 파라미터 많아질수록 더 좋은 성능낸다고 생각해옴

Training compute-optimal large language models

> 한정된 자원 내에서 가장 큰 모델이 꼭 최고의 성능을 내는 것은 아님. 모델은 작게 하되 데이터 크기를 키우는 것이 더 유용함 = chinchilla optimum or chinchilla scailing laws
>
> scaling laws의 목표는 데이터셋과 모델 사이즈를 “훈련” 계산 능력에 알맞게 설정하는 것
>
> 이 목표는 “추론” 계산 능력은 고려하지 않음 → 모델 서빙 시 문제 발생

성능 목표치가 정해지면 가장 선호되는 모델은 학습이 가장 빠른 모델이 아니라 추론이 가장 빠른 모델

큰 모델을 학습 시키는 것이 더 저렴할 수 있어도 작은 모델이 추론할 때 더 저렴할 수 있음

실제로 위 논문에서 제시한 10B 모델을 200B개의 토큰으로 학습시키는 것보다 7B 모델을 1T 토큰으로 학습시키는 게 더 나았다는 결과를 얻음

이 논문에서 집중한 부분은 다양한 추론 예산에 따라 더 많은 양의 토큰을 사용함으로써 다양한 모델의 최고 성능을 얻는 것 → 7B에서 65B의 모델

LLaMA-13B는 하나의 GPU에서도 구동 가능

공개 데이터셋만 사용

transformer 구조에서 변경한 부분, 학습 과정들 보여줄 예정

다른 LLM과 벤치마크 기준으로 비교

마지막으로 responsible AI 커뮤니티의 벤치마크를 활용하여 편향과 독성을 드러냄

Approach

GPT3, PaLM과 유사하게 학습

chinchilla scaling laws를 따름

큰 트랜스포머를 큰 텍스트 데이터에 기본적인 옵티마이저로 학습시킴

Pre-training Data

> LLM 만들 때 전처리 개빡세게 해야 함. 고퍼 논문에 잘 나와있음

공개 데이터셋

다른 LLM에서 썼던 것들을 재사용

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

English CommonCrawl [67%]

5개의 commoncrawl dump

2017~2020

CCNet pipeline

- 문장 레벨에서 중복 삭제
- fastText linear cliassifier로 영어아닌 페이지 삭제
- n-gram language model로 질 낮은 콘텐츠 필터링

위키피디아에서 참고로 사용된 페이지와 무작위 샘플링된 페이지를 분류하기 위해 선형 모델을 훈련시킴 → 참고로 분류되지 않은 페이지는 폐기

C4 [15%]

CommonCrawl 데이터셋이 성능 향상에 도움을 줬으므로 C4 사용

punctuation mark의 존재나 웹페이지 내의 단어/문장 개수를 heuristic하게 질적 필터링

Github [4.5%]

Google BigQuery에서 나온 데이터셋

Apache, BSD and MIT licenses의 데이터셋만 활용

문장 길이, 알파벳의 비율을 heuristic하게 질적 필터링

정규화식(re)으로 header같은 서식들은 삭제함

파일 레벨에서 중복 삭제

Wikipedia [4.5%]

2022 6~8월

20개의 언어. 라틴, 키릴 분자 사용

하이퍼링크, 댓글, 다른 서식 삭제

Gutenberg and Books3 [4.5%]

Gutenberg Project, Books3 section of ThePile

책 레벨로 중복 삭제. 90% 이상 겹칠 때

ArXiv [2.5%]

latex file

과학적 데이터 추가하기 위함

.tex 파일의 코멘트, 서식 제거

Stack Exchange [2%]

다양한 도메인(컴퓨터과학, 화학)에서의 질문-답변

HTML 태그 삭제

답변 점수 높은 것부터 정렬

Tokenizer

SentencePiece 활용해서 BPE

모든 숫자를 각개의 숫자로 분리

알 수 없는 UTF-8 문자를 분해하기 위해 바이트로 대체

전체 데이터셋은 1.4T의 토큰을 가짐

학습데이터 대부분에서 각 토큰은 한번만 쓰임.

Wikipedia와 Books 도메인을 제외하고는 약 두번의 에포크 수행

Architecture

트랜스포머 구조

이후에 나온 방법들을 추가로 적용

Pre-normalization [GPT3]

학습 안정화를 위함

각 서브레이어의 출력이 아니라 입력을 normalize

RMSNorm 사용

SwiGLU activation function [PaLM]

ReLU를 SwiGLU로 변경

본래의 4d가 아니라 $\frac{2}{3}4d$ 차원 사용

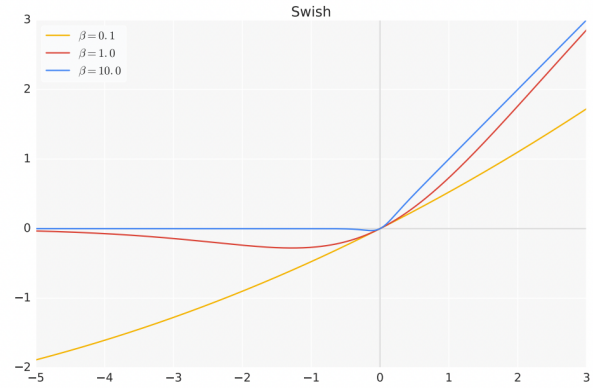

Figure 4: The Swish activation function.

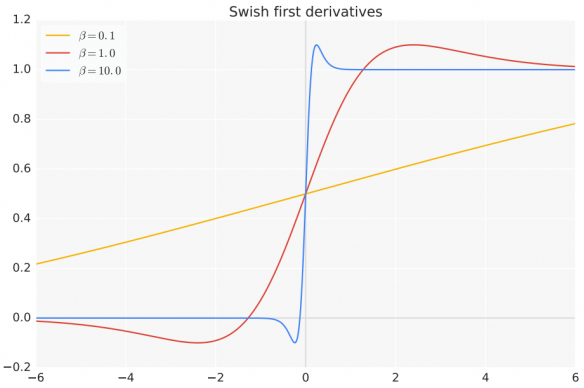

Figure 5: First derivatives of Swish.

Rotary Embeddings [GPTNeo]

absolute positional embeddings(=positional encoding) 제거하고 각 레이어에 RoPE 사용

Optimizer

AdamW(β1 = 0.9, β2 = 0.95)

cosine learning rate schedule - 마지막 학습률이 최대 학습률의 10%와 같도록

weight decay 0.1

gradient clipping 1.0

2, 000 warmup steps

학습률과 배치사이즈는 모델 크기에 따라 달리함

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Efficient implementation

학습 속도 개선

1) causal multi-head attention의 효율적인 구현 사용

    메모리 사용량, 런타임 감소

    xformers library에서 사용 가능

    논문에서 영감 받음

    flashattention의 backward 사용

        attention score 저장하지 않고 마스킹된 key/query score 계산하지 않음

2) 체크포인트 활용하여 역전파시에 activation을 다시 계산하지 않도록 함

    pytorch autograd 쓰는 대신 역전파 함수 직접 구현

    이 효과를 최대화하기 위해 모델과 시퀀스의 병렬처리를 활용하여 모델의 메모리 사용량을 줄임

3) GPU 간의 연결과 활성화함수의 계산을 최대한 중첩시킴

65B 모델 학습할 때 2048 A100 GPU, 80GB RAM에서 380 tokens/sec/GPU

1.4T 토큰을 사용하면서 21일 소요

Instruction Finetuning

LLaMA-65B는 finetuning 안해도 instruction 따를 수 있지만 매우 작은 양의 파인튜닝으로 MMLU에서 성능향상이 있다는 것을 확인함

Scaling instruction-finetuned language models의 프로토콜을 따름

간단한 파인튜닝임에도 불구하고 68.9% 달성 → 적당한 크기의 모델 성능 능가, sota(GPT 3.5)에는 못 미침

Bias, Toxicity and Misinformation

llm은 학습 데이터의 편향을 학습하고 유해한 콘텐츠를 생성함

web에서 가져온 데이터가 많기 때문에 이런 콘텐츠를 생성하지 않는지 확인해야 함

다양한 벤치마크 활용했지만 충분하지 않음