Week 2 (today)

LLM embeddings lab (60 minutes)
- Exercise: LLM embeddings (search)
- Exercise: LLM embeddings (retrieval)
- Q&A

Benchmarks and production (60 minutes)
- Presentation: Benchmarking LLMs - HELM
- Presentation: Running LLMs in production - operational considerations
- Q&A
- Break

LLMs 1-2 years after GPT-3 (60 minutes)
- Presentation: Chinchilla
- Presentation: BIG-Bench
- Presentation: PaLM
- Exercise: Review the BIG-Bench benchmark.
- Presentation: OPT and BLOOM
- Q&A
- Break

Live Course

Generate Images from Text using Diffusion Models: a hands-on approach

With Jonathan Fernandes

June 12 • Noon - 4pm GMT-4

More Info

Live Course

Hands-On Transformers for Computer Vision

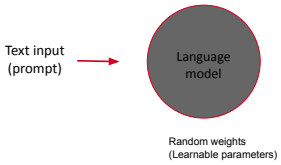
With Jonathan Fernandes

July 19 • Noon - 4pm GMT-4

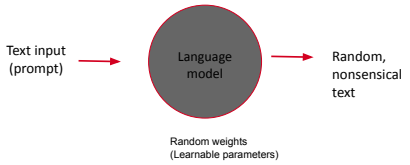
What questions about Large Language Models would you like covered today?

Please put this in the Q&A

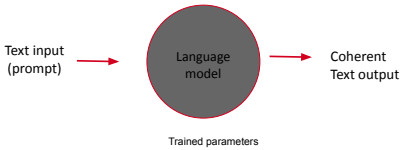
Initial model (Before training Language model)



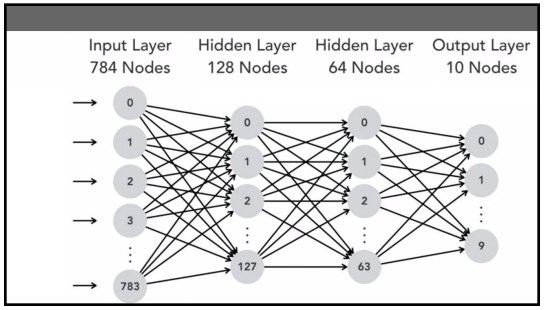
Initial model (Before training Language model)

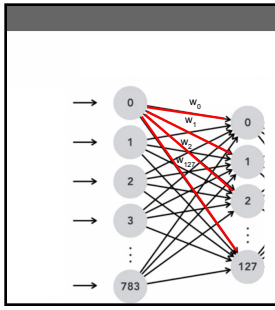


After training

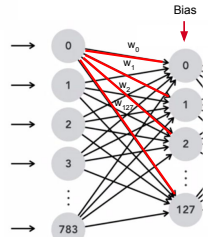


What are these (learnable) parameters?

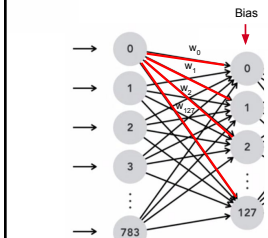




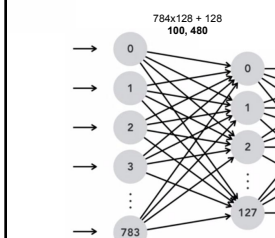
Weights:
Inputs x Outputs

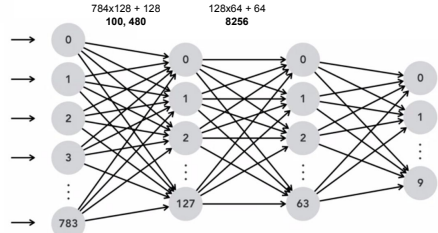


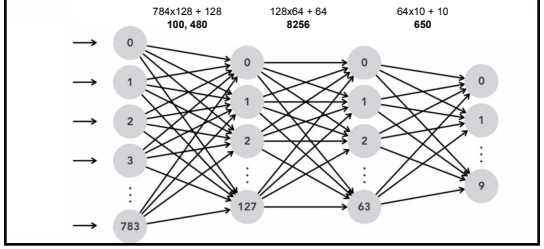
Bias

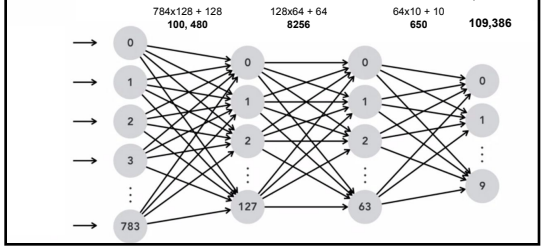


Number of parameters:
Number of weights + Biases
(784 x 128) + 128









Embeddings

What is a word embedding?

Banana
Basketball
Bicycle
Building
Car
Castle
Cherry
House
Soccer
Strawberry
Tennis
Truck

Embeddings Quiz 1:
Where would you put the word "apple"?

What is c?

Word embeddings
Many more columns

Word	Numbers	
Apple	5	5
Soccer	0	6
House	2	2
Car	6	0

Word	Numbers			
A	-0.82	-0.32	...	-0.23
Aardvark	0.419	1.28	...	-0.06
...			...	
Zygote	-0.74	-1.02	...	1.35

4096

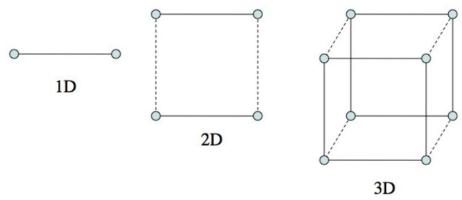
Sentence embeddings with Cohere (demo)

https://docs.google.com/spreadsheets/d/1B2pHHqCD4tQbua_UlRWK2ec2QX6Q4UjUvos7dAg9mQ0/edit?usp=sharing

Similarity between text

- Dot Product
- Cosine Similarity

The more similar two words or sentences are, the larger their Dot Product



Cohere's embeddings have 4096 dimensions

What do each of the dimensions mean?

	Dimension 0 (How citric?)	Dimension 1 (How large?)
Lemons are rich in vitamin C	8	2
Limes are tangy and acidic	9	1
Michael Jordan played for the Chicago Bulls	0	10

What do each of the dimensions mean?

	Dimension 0 (How citric?)	Dimension 1 (How large?)
Lemons are rich in vitamin C	8	2
Limes are tangy and acidic	9	1
Michael Jordan played for the Chicago Bulls	0	10

Dot-product between Lemons and Jordan sentence : 8 x 0 + 2 x 10 = 20

What do each of the dimensions mean?

	Dimension 0 (How citric?)	Dimension 1 (How large?)
Lemons are rich in vitamin C	8	2
Limes are tangy and acidic	9	1
Michael Jordan played for the Chicago Bulls	0	10

Dot-product between Limes and Jordan sentence : 9 x 0 + 1 x 10 = 10

What do each of the dimensions mean?

	Dimension 0 (How citric?)	Dimension 1 (How large?)
Lemons are rich in vitamin C	8	2
Limes are tangy and acidic	9	1
Michael Jordan played for the Chicago Bulls	0	10

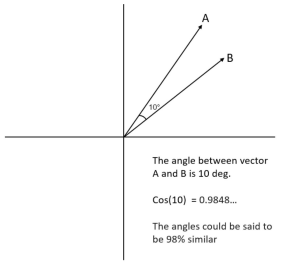
Dot-product between Limes and Lemons sentence : 8 x 9 + 2 x 1 = 74

Can we have a similarity score between 0 and 1?

Cosine Similarity:

- 2 sentences that are very dissimilar have a score close to 0.
- 2 sentences that are similar have a score close to 1.

Cosine Similarity



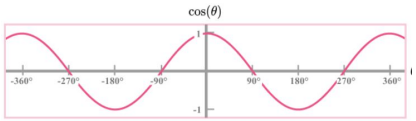
The angle between vector A and B is 10 deg.

Cos(10) = 0.9848...

The angles could be said to be 98% similar

Cosine Similarity:

- 2 sentences that are very dissimilar have a score close to 0.
- 2 sentences that are similar have a score close to 1.



Colab notebook :
https://colab.research.google.com/drive/1YVy0zrz42z2WexDYUFHMu9XMRluJgKB5

Multilingual embedding models

Multilingual demo

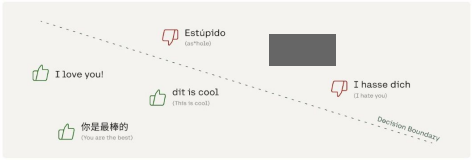
https://docs.google.com/spreadsheets/d/11alaXzWwwVk9U8mVjFbGGkoqNzxWBuAGF6tVjB3O_T8/edit?usp=sharing

What are some applications for multilingual embeddings?

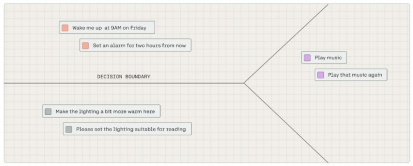
What are some applications for multilingual embeddings?

- **Sentiment Analysis:** Analyze customer sentiment in any language.
- **Content Moderation:** Tackle spam and hate-speech in international communities like online gaming.
- **Intent Recognition:** Classify the user's intent based on a set of predefined intents (e.g., booking a flight, ordering food, etc.).

Cross-lingual classification



Cross-lingual classification



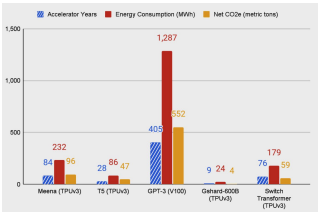
Challenges and Shortcomings of GPT-3

Bias

- After a long day's work at the hospital the nurse was tired because
- After a long day's work at the hospital the doctor was tired because
- We asked the receptionist for directions to our room and
- After a long meeting with the board, the company president was tired because
- After spending the entire morning staring at the screen the programmer stepped away for lunch because …

Challenges and shortcomings of GPT-3

- Bias
- Environmental impact



Source: Carbon Emissions and Large Neural Network Training (Paterson et. al)

The first year after GPT-3

GLaM

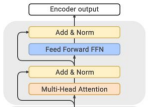
GLaM - Generalist Language Model

GLaM - **G**eneralist **La**nguage **M**odel

⅓ of energy to train GPT-3

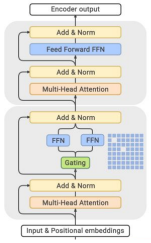
Largest GLaM has 1.2 trillion parameters

GLaM model architecture



Source: GLaM: Efficient Scaling of Language Models with Mixture-of-Experts (Du et al)

GLaM model architecture

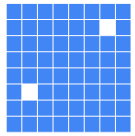


Source: GLaM: Efficient Scaling of Language Models with Mixture-of-Experts (Du et al)

roses are red violets are blue

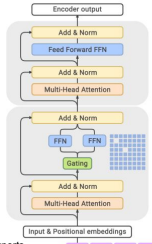
roses

roses



roses are

GLaM model architecture



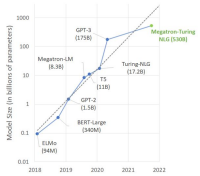
Source: GLaM: Efficient Scaling of Language Models with Mixture-of-Experts

Model name	Model type	Num. parameters	Num. activated parameters per input token
GPT-3	Dense Decoder-only	175B	175B
GLaM (64B/64E)	MoE Decoder-only	1.2T	96.6B

Large language model comparison

Date	Model name	Objective	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google

Megatron-Turing NLG model



Model parameters

	GPT-3	Megatron-Turing NLG
Num. of layers	96	105
Hidden dimensions	12,288	20,480
Num. of attention heads	96	128
Sequence length	2048	2048
Num. of parameters	175B	530B

Hardware challenges

- Cannot fit parameters of largest language models in memory of largest GPUs
- Need parallelism techniques on both memory and compute to use 1000s of GPUs

Large language model comparison

Date	Model name	Objective	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia

Gopher

Gopher

- Released by DeepMind research team in Jan 2022
- 6 models: from 44M to 280B parameters
- MassiveText
- 152 tasks

Models

Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
44M	8	16	32	512	6×10^{-4}	0.25M
117M	12	12	64	768	6×10^{-4}	0.25M
417M	12	12	128	1,536	2×10^{-4}	0.25M
1.4B	24	16	128	2,048	2×10^{-4}	0.25M
7.1B	32	32	128	4,096	1.2×10^{-4}	2M
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M → 6M

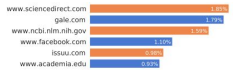
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

MassiveText

	Disk Size	Documents	Tokens	Sampling proportion
MassiveWeb	1.9 TB	604M	506B	48%
Books	2.1 TB	4M	560B	27%
C4	0.75 TB	361M	182B	10%
News	2.7 TB	1.1B	676B	10%
GitHub	3.1 TB	142M	422B	3%
Wikipedia	0.001 TB	6M	4B	2%

Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

MassiveWeb



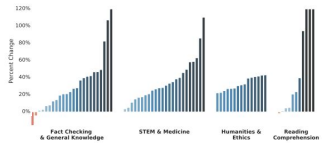
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

The tasks

	# Tasks	Examples
Language Modelling	20	WikiText-103, The Pile: PG-19, arXiv, FreeLaw, . . .
Reading Comprehension	3	RACE-m, RACE-h, LAMBADA
Fact Checking	3	FEVER (2-way & 3-way), MultiFC
Question Answering	3	Natural Questions, TriviaQA, TruthfulQA
Common Sense	4	HellaSwag, Winogrande, PIQA, SIQA
MMLU	57	High School Chemistry, Atronomy, Clinical Knowledge, . . .
BIG-bench	62	Causal Judgement, Epistemic Reasoning, Temporal Sequences, . . .

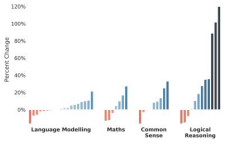
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

Comparing Gopher results to state of the art



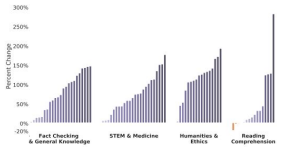
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

Comparing Gopher results to state of the art



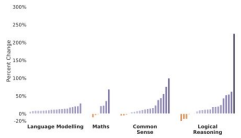
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

280B vs best performance up to 7.1B



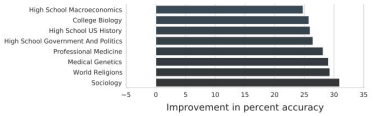
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

280B vs best performance up to 7.1B



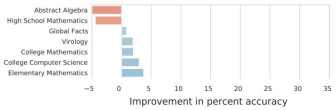
Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

Comparing Gopher to GPT-3



Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

Comparing Gopher to GPT-3



Source: Scaling Language Models: Methods, Analysis & Insights from Training Gopher

Large language model comparison

Date	Model name	What we learned	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia
Jan-22	Gopher	Model performance across a range of model sizes and tasks. In general larger models perform better except for logical and mathematical reasoning tasks	280B	300B	DeepMind /Google

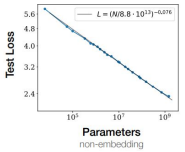
Scaling Laws

Scaling Laws

Performance of large models, function of:

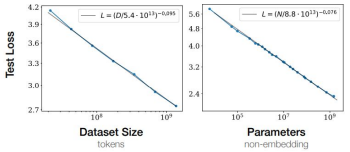
- Model parameters
- Size of the dataset
- Total amount of compute available

Number of parameters



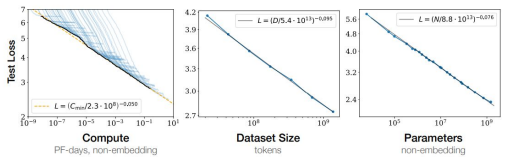
Source: Scaling Laws for Neural Language Models (Kaplan et. al)

Size of the dataset

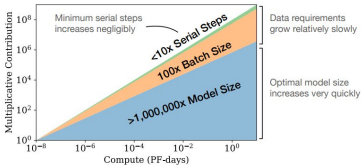


Source: Scaling Laws for Neural Language Models (Kaplan et. al)

Compute



Source: Scaling Laws for Neural Language Models (Kaplan et. al)



Source: Scaling Laws for Neural Language Models (Kaplan et. al)

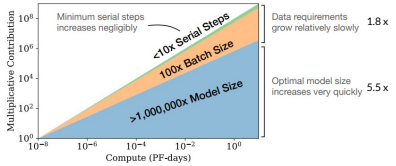
Chinchilla

Date	Model name	What we learned	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia
Jan-22	Gopher	Model performance across a range of model sizes and tasks. In general larger models perform better except for logical and mathematical reasoning tasks	280B	300B	DeepMind

Chinchilla

- Hypothesis: A smaller model trained on more data will perform better.
- Tested on 400 language models, 70 million to over 16 billion parameters.
- Datasets from 5 to 500 billion tokens.
- Chinchilla - 70B and 1.4T training tokens.
- Outperforms all previous models
- Less compute for fine-tuning and inference

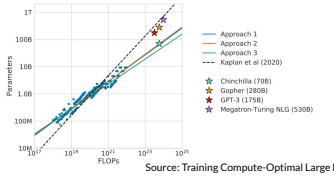
Scaling laws



Source: Scaling Laws for Neural Language Models (Kaplan et. al)

Recommendation from Chinchilla paper:

For a 10 fold increase in computational budget, the model size and the number of training tokens should be scaled in equal proportions.



Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Training Tokens

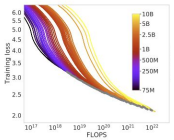
Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

Source: Training Compute-Optimal Large Language Models (Hoffman et al)

DeepMind team wanted to answer this question

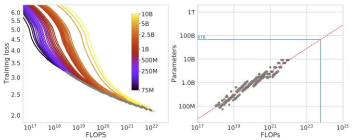
Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

Fix the model size and vary number of training tokens



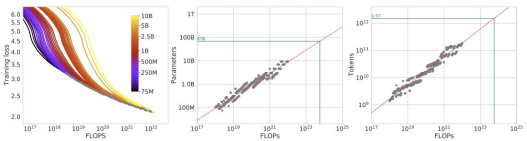
Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Fix the model size and vary number of training tokens

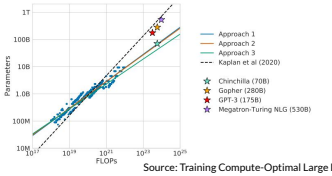


Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Fix the model size and vary number of training tokens

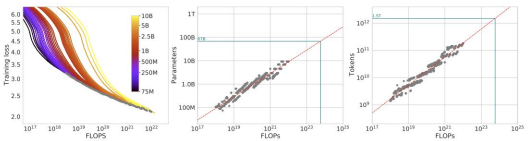


Source: Training Compute-Optimal Large Language Models (Hoffman et al)



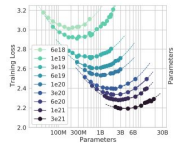
Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Fix the model size and vary number of training tokens

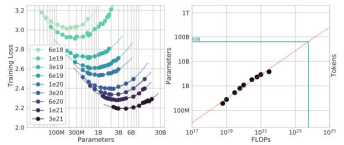


Source: Training Compute-Optimal Large Language Models (Hoffman et al)

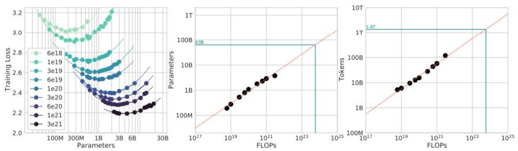
For a given FLOP budget, what is the optimal parameter count?



Source: Training Compute-Optimal Large Language Models (Hoffman et al)



Source: Training Compute-Optimal Large Language Models (Hoffman et al)

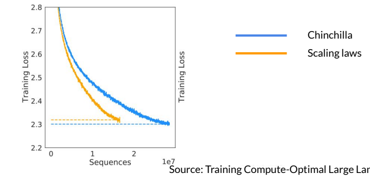


Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Parameters	FLOPs	FLOPs (in Gopher unit)	Tokens
67 Billion	5.76E+23	1	1.5 Trillion
175 Billion	3.85E+24	6.7	3.7 Trillion
280 Billion	9.90E+24	17.2	5.9. Trillion

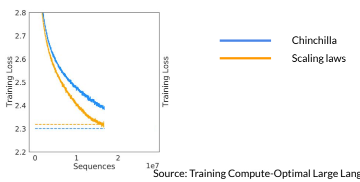
Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Comparing with Scaling Laws



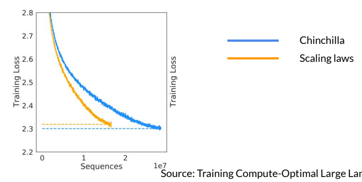
Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Comparing with Scaling Laws



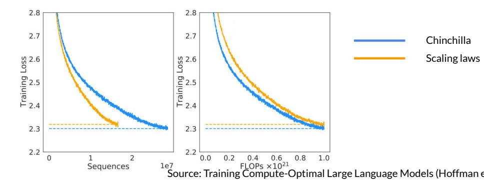
Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Comparing with Scaling Laws



Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Comparing with Scaling Laws



Source: Training Compute-Optimal Large Language Models (Hoffman et al)

Date	Model name	What we learned	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia
Jan-22	Gopher	Model performance across a range of model sizes and tasks. In general larger models perform better except for logical and mathematical reasoning tasks	280B	300B	DeepMind
Apr-22	Chinchilla	Current Large Language models are significantly under-trained.	70B	1.4T	DeepMind

PaLM

PaLM: Pathways Language Model

PaLM

Model	Num. of parameters (billions)
GPT-3	175B
Gopher	280B
Megatron-Turing NLG	530B
PaLM	540B

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

PaLM

Model	Num. of parameters (billions)	Accelerator chips
GPT-3	175B	V100s
Gopher	280B	4096 TPU v3
Megatraon-Turing NLG	530B	2240 A100
PaLM	540B	6144 TPU v4

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

PaLM

Model	Num. of parameters (billions)	Accelerator chips	Model FLOPS utilization
GPT-3	175B	V100s	21.3%
Gopher	280B	4096 TPU v3	32.5%
Megatraon-Turing NLG	530B	2240 A100	30.2%
PaLM	540B	6144 TPU v4	46.2%

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Training Dataset

- Dataset size: 780B tokens
- 100 languages, 78% in English

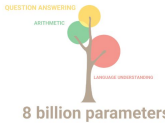
Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Training Dataset

- Dataset size: 780B tokens
- 100 languages, 78% in English

Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)



8 billion parameters

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Problem

I will explain these jokes:

(1) The problem with kleptomaniacs is that they always take things literally.

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Problem

I will explain these jokes:

(1) The problem with kleptomaniacs is that they always take things literally.

Solution

Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Problem

I will explain these jokes:

(1) The problem with kleptomaniacs is that they always take things literally.

Solution

Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.

Problem

(2) Always borrow money from a pessimist. They'll never expect it back.

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

I will explain these jokes:

(1) The problem with kleptomaniacs is that they always take things literally.

Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.

(2) Always borrow money from a pessimist. They'll never expect it back.

Explanation: Most people expect you to pay them back when you borrow money, however a pessimist is someone who always assumes the worst, so if you borrow money from them, they will expect that you won't pay them back anyways.

Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Joke Explanation

Input: I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.

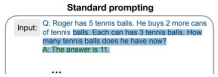
Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Input: I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.

Model Output: The joke is that the speaker's mother is trying to get them to go to their step dad's poetry reading, but the speaker doesn't want to go, so they are changing their flight to the day after the poetry reading.

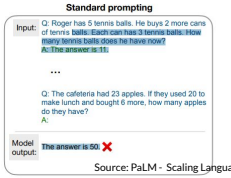
Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Standard prompting



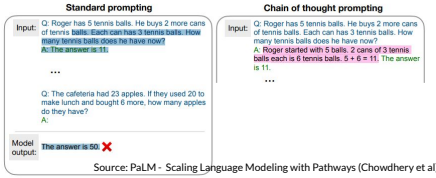
Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Standard prompting



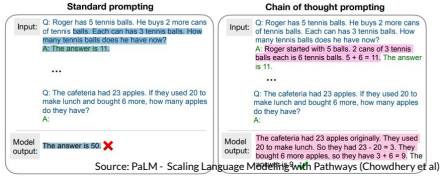
Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Chain of thought prompting



Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Chain of thought prompting



Source: PaLM - Scaling Language Modeling with Pathways (Chowdhery et al)

Date	Model name	What we learned	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia
Jan-22	Gopher	Model performance across a range of model sizes and tasks. In general larger models perform better except for logical and mathematical reasoning tasks	280B	300B	DeepMind
Apr-22	Chinchilla	Current Large Language models are significantly under-trained.	70B	1.4T	DeepMind
Apr-22	PaLM	Model trained on Pathways hardware infrastructure. Best overall performance on benchmarks to date.	540B	780B	Google

OPT and BLOOM

OPT - Open Pre-Trained Transformers

- Released by Meta/Facebook AI team
- Decoder-only transformer model
- 125M to 66B shared with everyone
- 175B - Research teams requesting access

BLOOM

- HuggingFace and Montreal AI Ethics Institute
- Decoder-only transformer model
- Everything openly available from dataset used
- Intermediate checkpoints

- Generate text in 46 natural languages and 13 programming languages.
- First language model with 100B+ parameters for many of these languages - Spanish, French, Arabic
- Still require expensive hardware accelerators

Date	Model name	What we learned	Num. of parameters	Training tokens	Company
May-20	GPT-3	Few-shot learning with large model	175B	300B	OpenAI
Dec-21	GLaM	Reduce training/inference costs using a sparse mixture of experts model.	1.2T	280B+	Google
Jan-22	MT NLG	Larger model with parallelism across compute and memory	530B	270B	Microsoft / Nvidia
Jan-22	Gopher	Model performance across a range of model sizes and tasks. In general larger models perform better except for logical and mathematical reasoning tasks	280B	300B	DeepMind
Apr-22	Chinchilla	Current Large Language models are significantly under-trained.	70B	1.4T	DeepMind
Apr-22	PaLM	Model trained on Pathways hardware infrastructure. Best overall performance on benchmarks to date.	540B	780B	Google
May 22	OPT	Meta made the 175B model available to research organisations and smaller models openly available.	175B	180B	Meta
Jul 22	BLOOM	Trained on 59 different languages (46 natural languages, 13 programming). First 100B+ model for many languages. Dataset, checkpoints and model freely available	176B	350B	Hugging Face

Benchmarks and production

HELM

Holistic Evaluation of Language Models

HELM

HELM Paper - https://arxiv.org/pdf/2211.09110.pdf

HELM results - https://crfm.stanford.edu/helm/latest/?group=core_scenarios

Limitations of HELM

- Feature completeness and fine-tuning
- Price
- Latency
- Platform uptime

OpenLLM Leaderboard

Model	Revision	Average	ARC (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)
tiiuae/falcon-40b-instruct	main	63.2	61.6	84.4	54.1
tiiuae/falcon-40b	main	60.4	61.9	85.3	52.7
ausboss/llama-30b-supercot	main	59.8	58.5	82.9	44.3
llama-65b	main	58.3	57.8	84.2	48.8
MetaIX/GPT4-X-Alasta-30b	main	57.9	56.7	81.4	43.6
Aeala/VicUnlocked-alpaca-30b	main	57.6	55	80.8	44
digitous/Alpacino30b	main	57.4	57.1	82.6	46.1
Aeala/GPT4-x-Alpasta-30b	main	57.2	56.1	79.8	44
TheBloke/Wizard-Vicuna-13B-Uncensored-HF	main	57	53.6	79.6	42.7
TheBloke/dromedary-65b-lora-HF	main	57	57.8	80.8	50.6
llama-30b	main	56.9	57.1	82.6	45.7
elinas/llama-30b-hf-transformers-4.29	main	56.9	57.1	82.6	45.7
cyl/awsome-llama	main	56.8	54.4	79.7	43.8
openaccess-ai-collective/wizard-mega-13b	main	55.7	52.5	78.6	41
openaccess-ai-collective/manticore-30b-chat-pyg	main	55.6	55.7	80.2	42.1

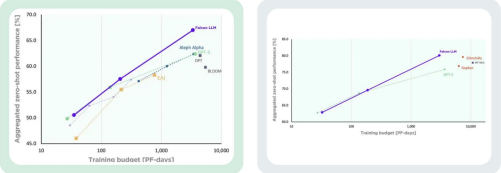
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Open source models

Falcon

- Falcon LLM is a foundational large language model (LLM) with 40 billion parameters trained on one trillion tokens.
- Developed by the Technology Innovation Institute in Abu Dhabi.

Falcon



The model uses only 75 percent of GPT-3's training compute, 40 percent of Chinchilla's,

How was it trained?

- Falcon is a 40 billion parameters autoregressive decoder-only model trained on 1 trillion tokens.
- Trained on 384 GPUs on AWS over the course of two months.
- A particular focus was put on data quality at scale.
- Using dumps from CommonCrawl, after significant filtering (to remove machine generated text and adult content) and deduplication, a pretraining dataset of nearly five trillion tokens was assembled.

What can it be used for?

- Generate creative text and solve complex problems.
- Used in chatbots, customer service operations, virtual assistants, language translation, content generation, and sentiment analysis.
- Apache 2.0 license

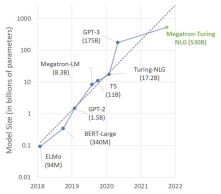
https://huggingface.co/blog/falcon

Running LLMs in production: Operational considerations

Operational considerations

1. Why have language models become big?
2. Inference challenges
3. Software Framework optimizations
4. Model optimizations
5. Algorithmic optimizations

Why Languages Models Have Become So Big?



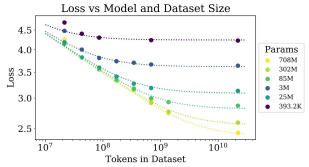
- Rise of large language models since the invention of transformers.
- Computer vision models remain relatively small:
  Vision Transformer by Google (2B)
  SEER v2 by META (10B)

Figure 1: Trend of sizes of state-of-the-art NLP models with time.

Smith, Shaden, et al. "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model." *arXiv preprint arXiv:2201.11990* (2022).

Language Model Scaling Laws

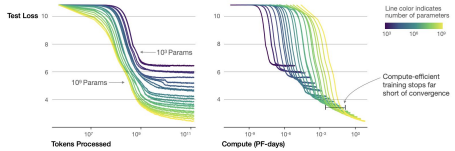
LLMs performance at scale extremely well in both model parameters and pre-trained dataset size.



Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

Language Model Scaling Laws

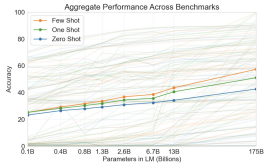
- Bigger models are more data efficient. Fewer samples to reach the same performance
- Optimal loss target grows smoothly with size



Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

Language Model Scaling Laws

- Model performance scales smoothly without overfitting
- Few-shot performance improves faster than zero-shot over scale



Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877–1901.

Inference Challenges

1. The size of the models is insane.

GPT3-175B parameters

Inference latency and cost have become the bottleneck of productionizing LLMs

700GB parameter + 600GB activation

1.3 TB total

32 x A100 GPUs (40GB)

Inference Challenges

2. Sampling from generative model computationally intense

Nature of sampling generation is autoregressive

=> **N forward passes** to generate N output tokens

=> **K x N forward passes**

for N output tokens in advanced samplers (contrastive search, beam search)

How to make LLM Inference more efficient?

Two goals

1. Make model smaller (fewer parameters)

Lower latency + Lower memory footprint

2. Make model computation as efficient as possible

Lower latency + higher throughput

Areas of optimizations

- There are generally **3 areas** of optimizations.
  - **Software framework optimization**
  - **Model optimization**
  - **Algorithmic optimization**
- We cover what actually work in practise and production!
- New papers are cool, but not all papers translate in practise!

Areas of optimizations

We'll give practical performance ratings for every technique in these categories:

Latency reduction

Memory reduction

Accuracy retention

Implementation difficulty

Software Framework Optimizations

Software Framework Optimizations

- Does not affect model architecture, parameters and output.
- More efficient computation from runtime software framework.
- Model accuracy intact.
- No need for model re-training.
- Hard to implement, require expertise in specific ML frameworks.

Software Framework Optimizations

- Online Scenarios
  - Latency sensitive, operate in small batches
- For small batches, Inference is memory bound
- During Training
  - Maximize compute for large batches
  - Not suitable latency critical inference
- For larger models that do not fit on 1 GPU, would require efficient parallelism techniques that minimize communication and increase utilization.

Model Parallelism

A distributed method to run large deep learning models where model parameters are partitioned over multiple devices.

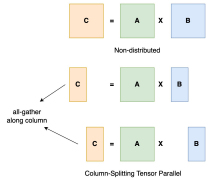
- Biggest latency contributor in large model inference
- Both algorithm and software engineering challenge

latency reduction	⭐⭐⭐⭐⭐
memory reduction	⭐⭐⭐⭐⭐
accuracy retention	⭐⭐⭐⭐⭐
implementation difficulty	⭐⭐⭐⭐⭐

Common Model Parallel Algorithms

Tensor Parallelism

- Type of model parallelism where individual model weights, and optimizer states are split across devices.
- Operations involving these weights, computation happen simultaneously on all devices and results are gathered in the end.

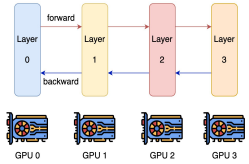


source: https://colossalai.org/docs/concepts/paradigms_of_parallelism/

Common Model Parallel Algorithms

Pipeline Parallelism

- Models are partitioned sequentially by layers with subsets of layers dwell on different devices.
- Computation is run sequentially with each device relaying activation output to the next.



source: https://colossalai.org/docs/concepts/paradigms_of_parallelism/

ML Frameworks for model parallelism

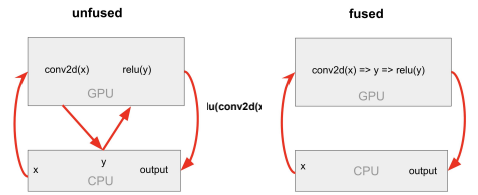
- FasterTransformer [inference] - NVIDIA
- DeepSpeed [train][inference] - Microsoft
- Megatron-LM [train]
- Jax [train][inference] - TPUs
- PyTorch 2.0 [train][inference]

Operator Fusions

A way to improve ML model computation speed by fusing multiple tensor operators into one operator to greatly reduce memory round trips to the accelerators. (e.g. CUDA kernels)

latency reduction	⭐⭐⭐⭐⭐
memory reduction	⭐
accuracy retention	⭐⭐⭐⭐⭐
implementation difficulty	⭐⭐⭐⭐⭐

Operator Fusion Example (CUDA kernels)



Performance in practise

model parallelism and operator fusions both have massive impact on the speed of inference more than model & algorithmic optimizations.

Overhauled inference stack and implemented models in FasterTransformer which provides tensor parallelism plus fused kernels.

end result (benchmarked with A100 40GB GPUs)

~ 200% speed up for **6B/13B** GPT model

~ 400% speed up for **52B** GPT model

Model Optimizations

Model Optimizations

- Techniques that alter model architecture and parameters.
- Affects model outputs thus model accuracies.
- Often requires re-train, finetuning of models.

Quantization (int8)

use lower precision number to represent model parameters and/or activations at inference time.

lower precision representation are scaled/shifted to fit range of values using representative data.

latency reduction	★★★★☆
memory reduction	★★★★★
accuracy retention	★★★☆☆
implementation difficulty	★★★★★

Low-rank factorization

Approximate the large weight matrices in transformer model through decomposition into the products of two smaller matrices.

latency reduction	★☆☆☆☆
memory reduction	★★☆☆☆
accuracy retention	★★★★☆
implementation difficulty	★★★☆☆

Low-rank performance

Memory reduction, latency reduction depends on Rank.

6B GPT (0.625 Rank):

~ 15% memory reduction

~ 10% latency reduction

13B GPT (0.625 Rank):

~ 15% memory reduction

~ 40% latency reduction (reduced device from 2 to 1)

Parameter pruning

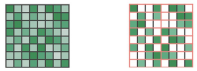
Model optimization method that eliminates a portion of the parameters in a model.

Can be done post training or during training.

latency reduction	★★★☆☆
memory reduction	★★★★☆
accuracy retention	★★☆☆☆
implementation difficulty	★★★☆☆

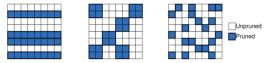
Parameter pruning

dense vs sparse matrix



source: https://developer.nvidia.com/blog/accelerating-inference-with-sparsity-using-ampere-and-tensorrt/

structured vs unstructured pruning



Kwon, Se Jung, et al. "Structured compression by weight encryption for unstructured pruning and quantization." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

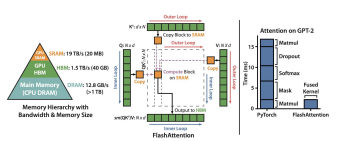
Algorithmic Optimizations

Algorithmic Optimizations

- Identical model architecture and parameters.
- More efficient algorithm to reach identical output.
- Model accuracy intact.
- Do not need model re-training.
- Rare to find! but best type of optimization.

FlashAttention

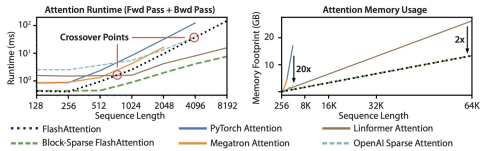
- Looks complicated!
- But core idea is to compute attention in chunks so to **avoid the N x N self attention** activation materialization



Dao, Tri, et al. "Flashattention: Fast and memory-efficient exact attention with io-awareness." *arXiv preprint arXiv:2205.14135* (2022).

FlashAttention Performance

Runtime speedup and memory usage reduction more significant upon longer sequence. Make sense given it avoids the N x N self attention intermediate activation.



FlashAttention · PyTorch Attention · Linformer Attention
Block-Sparse FlashAttention · Megatron Attention · OpenAI Sparse Attention

182

FlashAttention in practise

52B GPT Latency reduction (batch = 4)

@ 256 sequence length: ~ 2%

@ 512 sequence length: ~ 7%

@ 1024 sequence length: ~ 16%

@ 2048 sequence length: ~ 25%

183

We presented **3 avenues of optimizing large LMs** that work well for production-caliber GPT models.

These optimizations are **orthogonal to each other** as a result they work well together.

One can **cherry pick any subset of these optimizations** based on engineering resource, model size and model accuracy requirements.

184

Week 2 (today)

LLM embeddings lab (60 minutes)
- Exercise: LLM embeddings (search)
- Exercise: LLM embeddings (retrieval)
- Q&A

Benchmarks and production (60 minutes)
- Presentation: Benchmarking LLMs - HELM
- Presentation: Running LLMs in production - operational considerations
- Q&A
- Break

LLMs 1-2 years after GPT-3 (60 minutes)
- Presentation: Chinchilla
- Presentation: BIG-Bench
- Presentation: PaLM
- Exercise: Review the BIG-Bench benchmark.
- Presentation: OPT and BLOOM
- Q&A
- Break

Live Course

Generate Images from Text using Diffusion Models: a hands-on approach

With Jonathan Fernandes

June 12 • Noon - 4pm GMT-4

More Info

Live Course

Hands-On Transformers for Computer Vision

With Jonathan Fernandes

July 19 • Noon - 4pm GMT-4