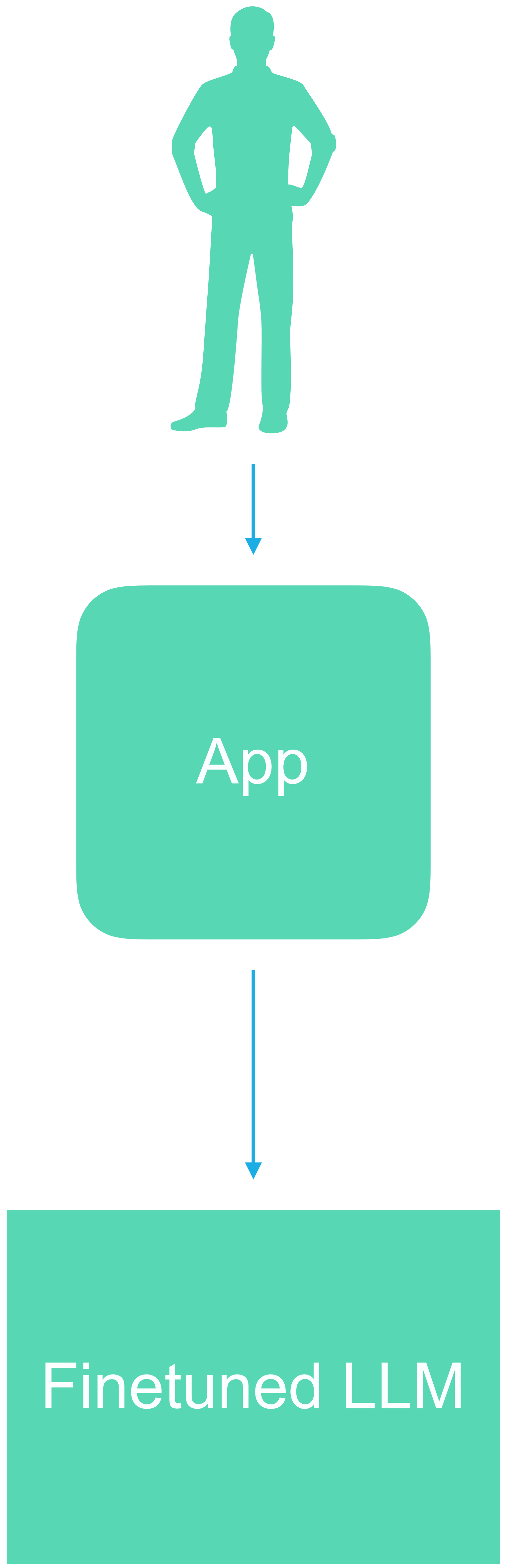
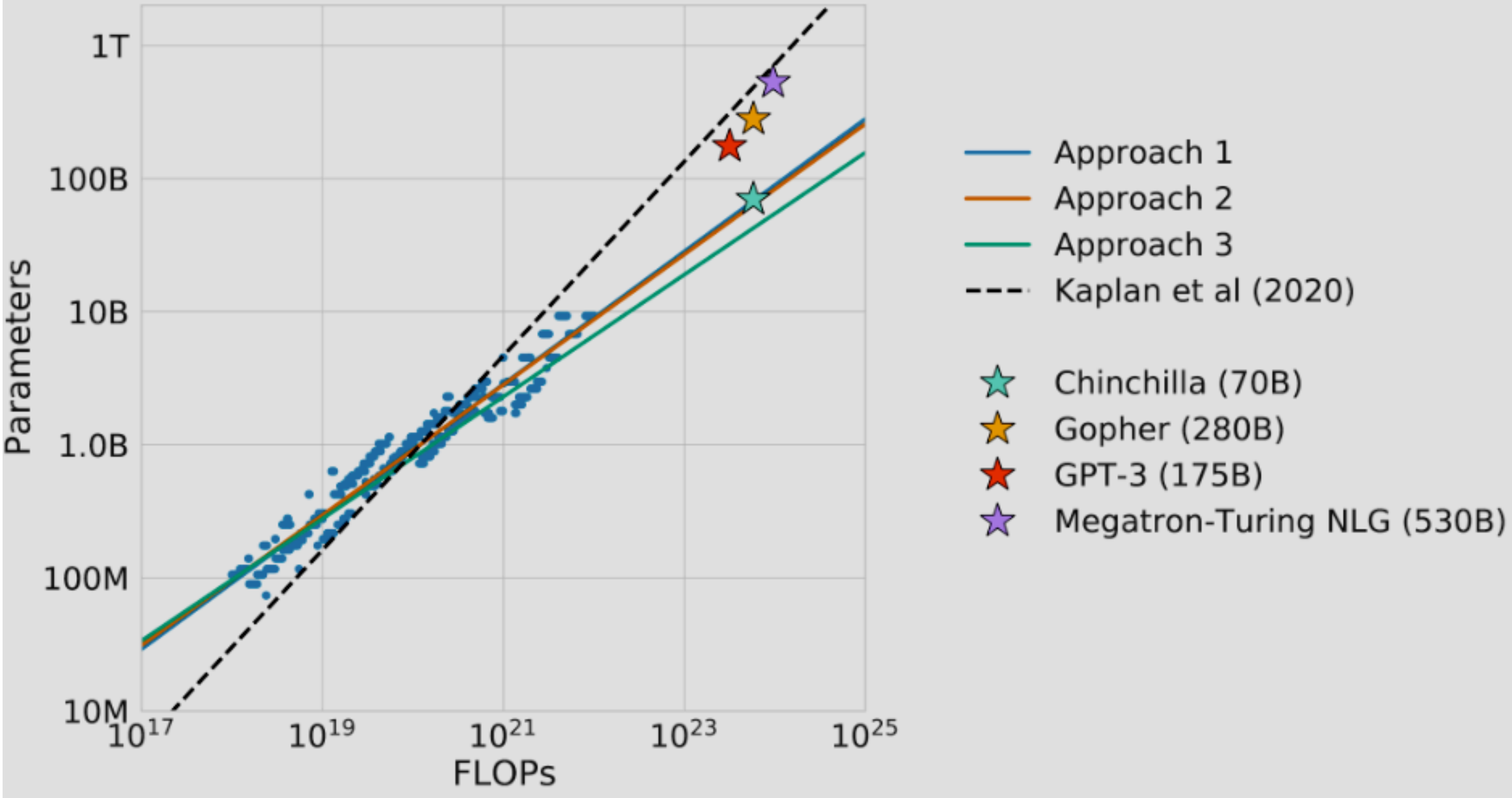
Behind these GenAI tools there is a model





"Training Compute-Optimal Large Language Models" Hoffman et. Al.

Data Trainers

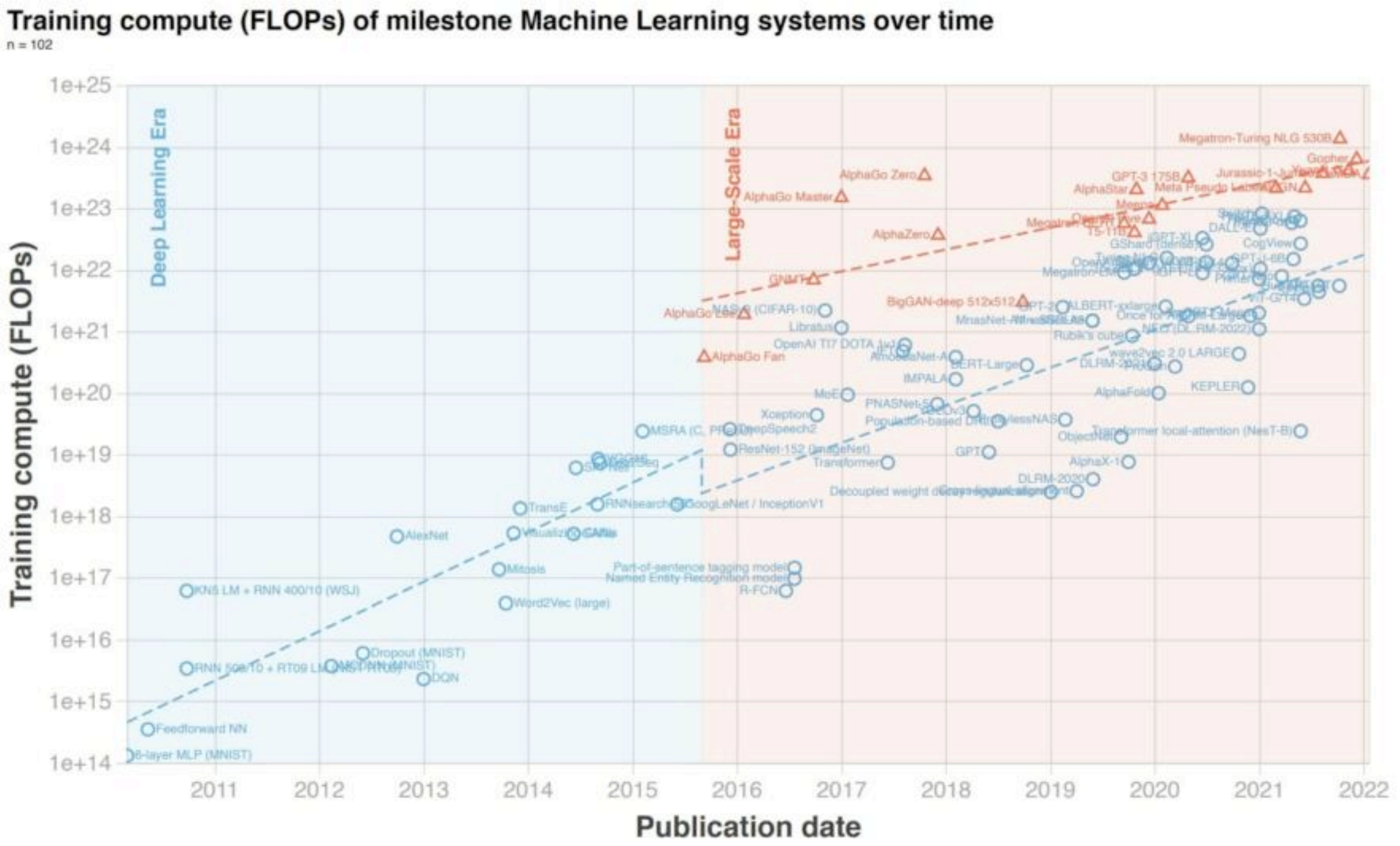
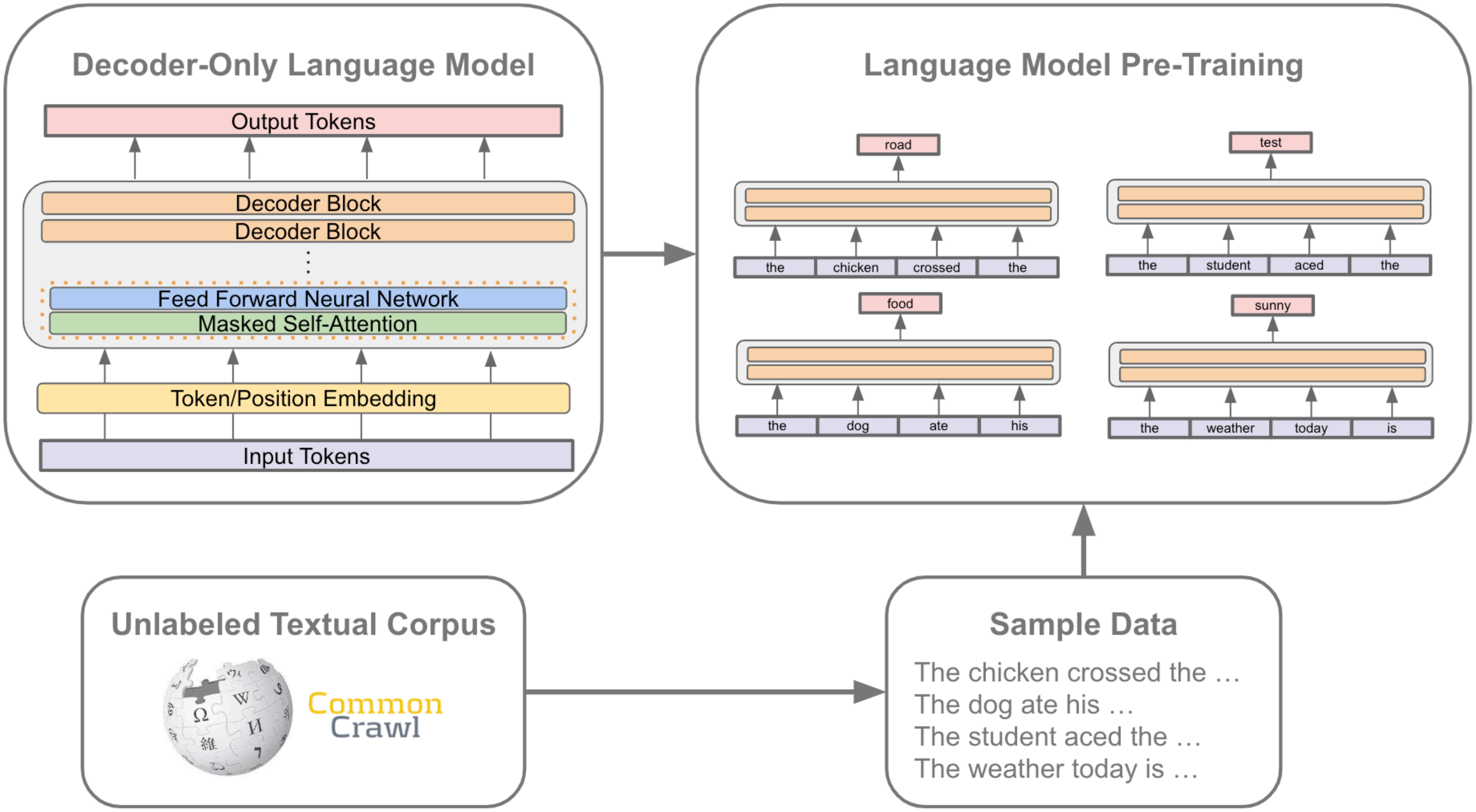
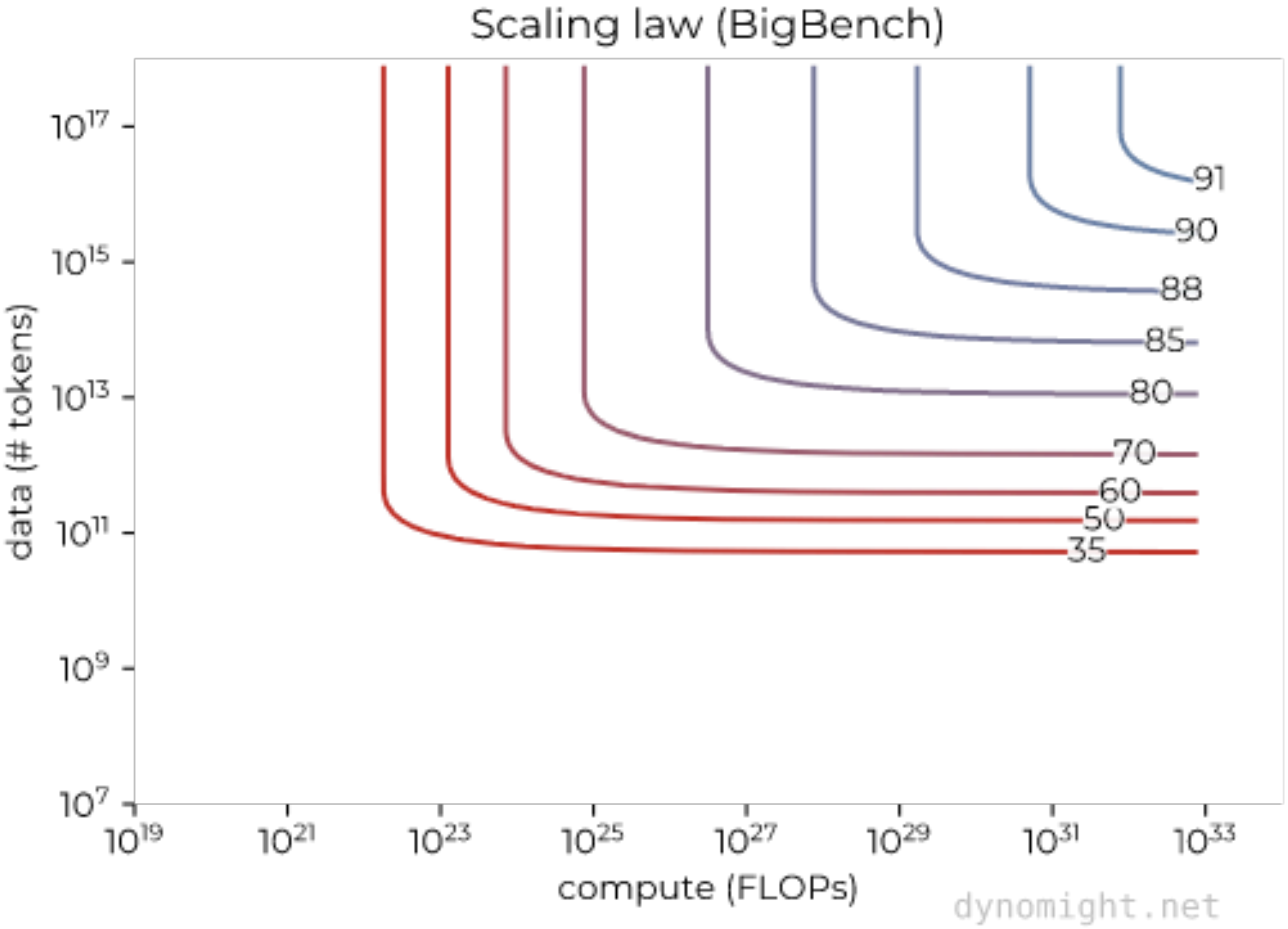


Figure 3: Trends in training compute of $n102$ milestone ML systems between 2010 and 2022. Notice the emergence of a possible new trend of large-scale models around 2016. The trend in the remaining models stays the same before and after 2016.

“Training Compute-Optimal Large Language Models” Hoffman et. Al.



Decoder-Only Language Model
Output Tokens
Decoder Block
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Token/Position Embedding
Input Tokens
Language Model Pre-Training
road
the
chicken
crossed
the
test
the
student
aced
the
food
the
dog
ate
his
sunny
the
weather
today
is
Unlabeled Textual Corpus
Common
Crawl
Sample Data
The chicken crossed the …
The dog ate his …
The student aced the …
The weather today is …


Scaling law (BigBench)
data (# tokens)
compute (FLOPs)
10^{17}
10^{15}
10^{13}
10^{11}
10^{9}
10^{7}
10^{19}
10^{21}
10^{23}
10^{25}
10^{27}
10^{29}
10^{31}
10^{33}
91
90
88
85
80
70
60
50
35
dynomight.net

Data Trainers

Model	Provider	Open-Source	Speed	Quality	Params	Fine-Tuneability
gpt-4	OpenAI	No	★☆☆	★★★★	-	No
gpt-3.5-turbo	OpenAI	No	★★☆	★★★☆	175B	No
gpt-3	OpenAI	No	★☆☆	★★★☆	175B	No
ada, babbage, curie	OpenAI	No	★★★	★☆☆☆	350M - 7B	Yes
claude	Anthropic	Yes	★★☆	★★★☆	52B	no
claude-instant	Anthropic	Yes	★★★	★★☆☆	52B	No
command-xlarge	Cohere	No	★★☆	★★☆☆	50B	Yes
command-medium	Cohere	No	★★★	★☆☆☆	6B	Yes
BERT	Google	Yes	★★★	★☆☆☆	345M	Yes
T5	Google	Yes	★★☆	★☆☆☆	11B	Yes
PaLM	Google	Yes	★☆☆	★★☆☆	540B	Yes
LLaMA	Meta AI	Yes	★★☆	★★☆☆	65B	Yes
CTRL	Salesforce	Yes	★★★	★☆☆☆	1.6B	Yes
Dolly 2.0	Databricks	Yes	★★☆	★★☆☆	12B	Yes

Comparison of training options

Technique	Pros	Cons	Cost	Risk mitigation with RLHF
Prompt Engineering	Cost effective Quick Deployment Wide Applicability	Limited by model capability Time consuming prompts	Low, API fees Good prompts imply up to 8x cost reduction	Nope
Finetuned LLM	Customised Performance Competitive Advantage	Higher Cost Medium Complexity Medium Data Needs Need team for model maintenance	Moderate (More later)	Can use, consider cost of human annotation
Finetuned LLM with PEFT	Customised Performance Competitive Advantage	Medium Cost Medium Complexity Need team for model maintenance	Lower than full finetuning (More later)	Can use, consider cost of human annotation
LLM from scratch	Full Control and customisation Potential for innovation	Highest Cost Risk of inferior performance Data Cost Team of specialists	Highest	Doing RHLF on a new model is complex

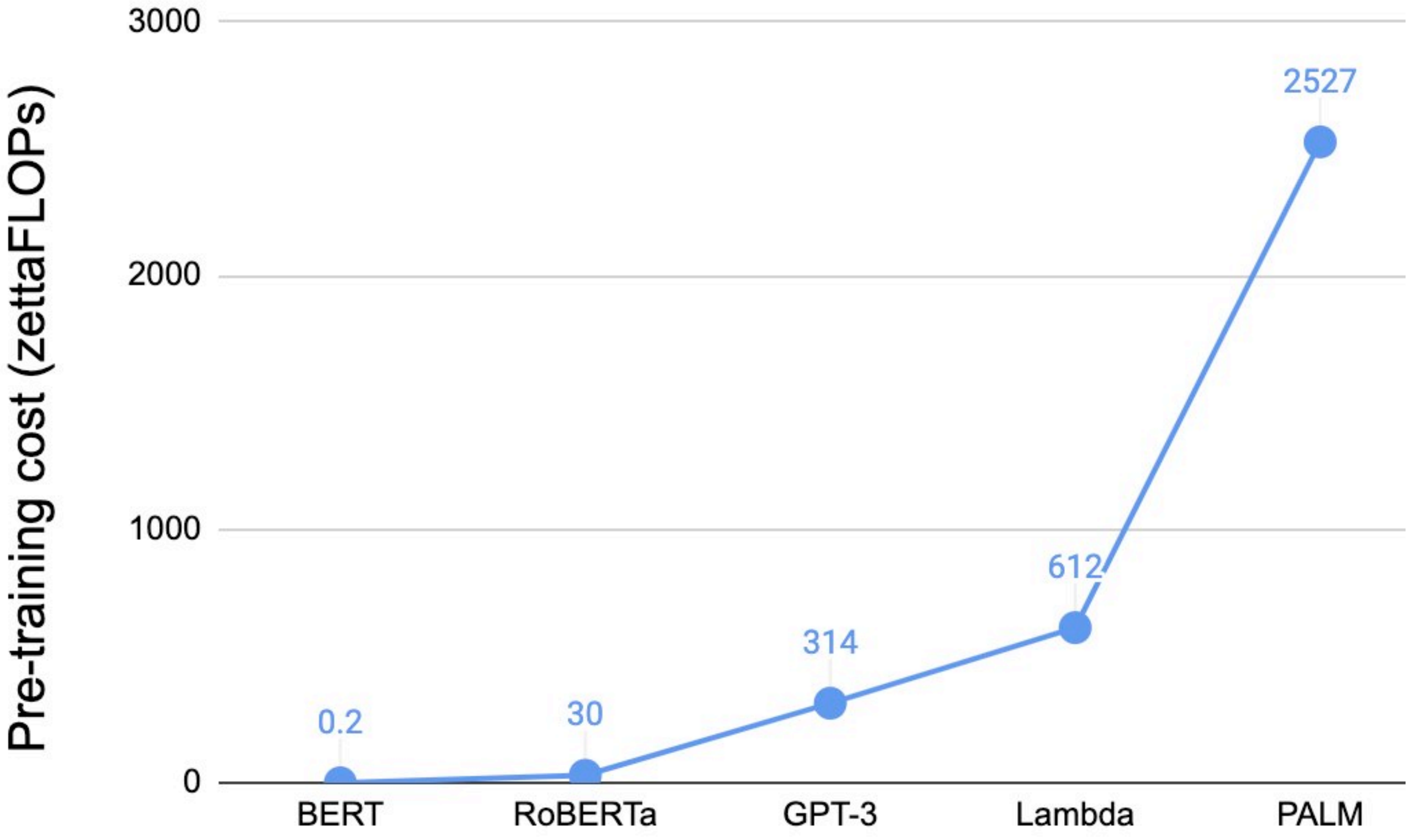
Comparison of training options

Pretrained / Fine Tuned	Model Name	Params*	Fine tuning Cost ($)	Input Cost ($)	Output Cost ($)	Total Cost ($)
Pretrained	GPT-4 32K	1 Tn +	NA	360k	360k	720k
	GPT-4 8K	1 Tn +	NA	180k	180k	360k
	DaVinci	175 Bn	NA	120k	60k	180k
	Claude v1	52 Bn	NA	66k	96k	162k
	Curie	13 Bn	NA	12k	6k	18k
	Self-hosted 7B	7 Bn	NA	350	1750	2.1k
Fine Tuned	DaVinci	175 Bn	180k	720k	360k	1.26M
	Curie	13 Bn	18k	72k	36k	126k
	Self-hosted 7B	7 Bn	1400	350	1750	3.5k

Comparison of training options: Training from scratch

Model	Parameters	Jump Factor	Chinchilla Tokens (B)	Jump Factor	CS-2 Config	Days To Train	Jump Factor	Price To Train	Jump Factor	Cost Per 1M Parameters
GPT-3XL	1.3		26		4 * CS-2	0.4		$2,500		$1.92
GPT-J	6	4.6 X	120	4.6 X	4 * CS-2	8	20.0 X	$45,000	18.0 X	$7.50
GPT-3 6.7B	6.7	1.1 X	134	1.1 X	4 * CS-2	11	1.4 X	$40,000	0.9 X	$5.97
T-5 11B	11	1.6 X	*34*	0.3 X	4 * CS-2	9	0.8 X	$60,000	1.5 X	$5.45
GPT-3 13B	13	1.2 X	260	7.6 X	4 * CS-2	39	4.3 X	$150,000	2.5 X	$11.54
GPT NeoX	20	1.5 X	400	1.5 X	4 * CS-2	47	1.2 X	$525,000	3.5 X	$26.25
GPT NeoX	*20*	*1.5 X*	***400***	*1.5 X*	***16 * CS-2***	***11.1***	*0.3 X*	***$656,250***	*4.4 X*	***$32.81***
GPT 70B	70	3.5 X	1,400	3.5 X	4 * CS-2	85	1.8 X	$2,500,000	4.8 X	$35.71
GPT 70B	*70*	*3.5 X*	***1,400***	*3.5 X*	***16 * CS-2***	***21.3***	*0.3 X*	***$3,125,000***	*6.0 X*	***$44.64***
GPT 175B	175	2.5 X	3,500	2.5 X	4 * CS-2	***110.5***	1.3 X	***$8,750,000***	3.5 X	***$50.00***
GPT 175B	*175*	*2.5 X*	***3,500***	*2.5 X*	***16 * CS-2***	***27.6***	*0.3 X*	***$10,937,500***	*4.4 X*	***$62.50***

Growth of training cost for large language models



Finetuning with PEFT: LoRA

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter^{H})	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter^{H})	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

“LoRA: Low-Rank Adaptation of Large Language Models” Hu et.al.

So what do we do?

We try to use the LLMs and use Prompt Engineering as much as possible. Once we reached a ceiling, only then we think of fine-tuning an LLM

(there are techniques to avoid doing a full finetuning)