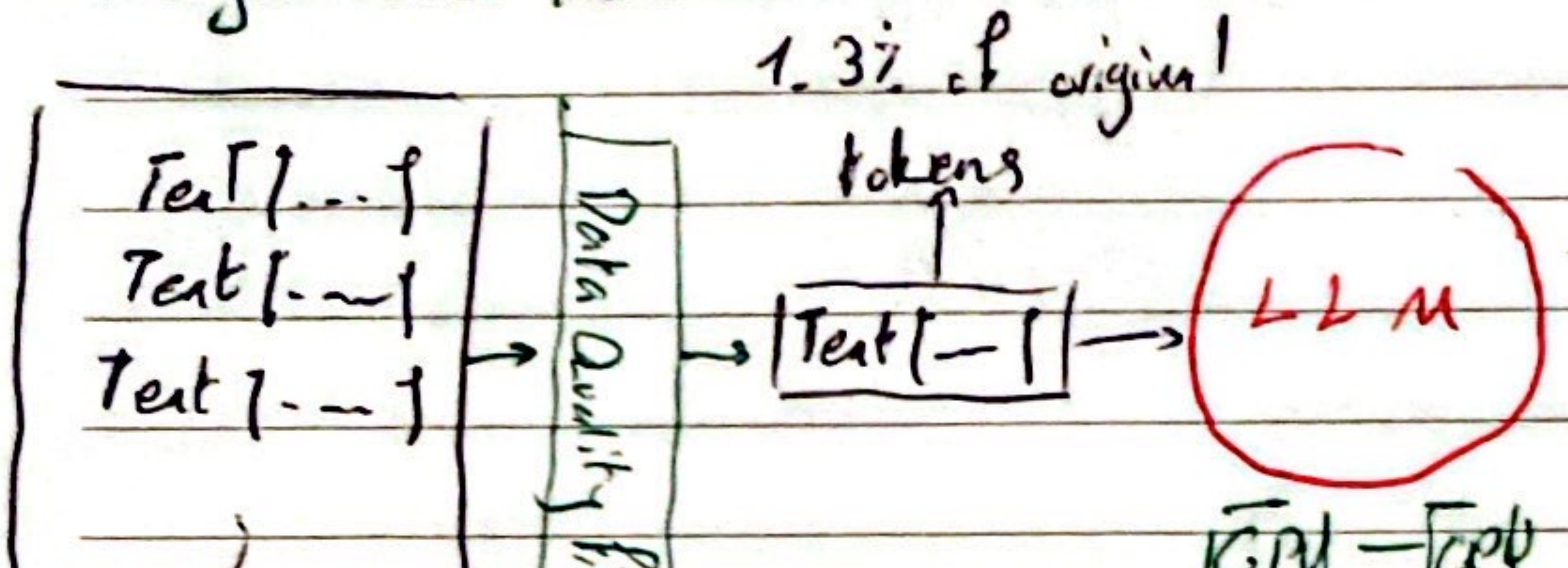
LLM Pre-training and Scaling laws

Model hubs after defining a usecase, most of the times choosing from a series of pretrained models, are better than training one.
A full list of available pre-trained models, can be found on Hugging Face hub, TF Hub, ...

How Are LLMs Pre-train?

A high level look:



GB - TB - PB
of unstructured
data

\+

Token string	Token ID	Embedding
'_The'	37	[-0.05?, ...]
'_Teacher'	11852	
'_Teaches'	19741	
'_the'	1236	

Vocabulary

Pre-Training based on LLM Types: each type of LLm structure, is best for a certain set of tasks, and is pre-trained differently:

A) Auto encoding models (Encoder-only LLM)

Masked language modeling (MLM)

| The | teacher | ~~teaching~~ | the | students |

<mask>

encoder-only LLM

Good use cases:
- sentimental analysis
- named entity recognition
- word classification

Example models:
- BERT
- ROBERTA

Objective: Reconstruct text ("denoising")

| The | teacher | <mask> | the | students |

teaching

bidirectional Context
(uses both the context (tokens) before and after to predict masked tokens)

B) Autoregressive models: (Decoder-Only)

Causal language modeling (CLM)
Aka. full " "

| The | teacher | ? |

Decoder-Only model

Objective: Predict next token

| The | teacher | teaches |

unidirectional Context
(meaning the model only uses Previous tokens to generate new token)

Good use cases:
- Text generation
- other emergent behaviour (Depends on the model)

Example models:
- GPT
- BLOOM

C) Sequence-to-sequence models: (encoder-decoder models)

Span Corruption

| The | teacher | <mask> | <mask> | students |

| The | teacher | <x> | students |

sentient token

Enc-Dec LLM

Objective: reconstruct span

| <x> | teaches | the |

Predictive tokens

Good use-cases
- Translation
- Summarization
- Question answering

Example models:
- T5
- BART

Computational Challenges to train an LLM

- to run an LLM, we need 4 bytes for each parameter

- to train it 20 bytes per parameter + original 4 bytes { adam optimizer (+8 byte) / Gradients (+4 byte) / Activation and temp memory (+8 bytes)

So approx for a 1B para model: 4GB @ 32bit. full Percision to store model → 80GB @ 32 bit full percision to train the model

What we can do to store and train models on a smaller GPU-Ram?

Quantization: the process of projecting parameters to smaller dtypes/sizes to reduce the store/train size of LLM. The original dtype to store paras is FP32, which requires 4 bytes, but we try to project it's valve to smaller dtypes such as FP16, BFloat 16 or INT8:

	Bits	Exponent	Fraction	Memory needed to store on Valve
FP32	32	8	23	4 byte
FP16	16	5	10	2 byte
BFLOAT16	16	8	7	2 byte
INT8	8	-/-	7	1 byte

→ best structure for Quantization for now (less loss than FP16)

- So we Quantize, to reduce memory needed for storing and training LLM, at the Cost of Percision

- Quantization-aware training (QAT) learns the Quantization scaling factors during training for instance, google Flan T5 is trained by QAT on BFLOAT 16

model size : Parameter size
Dataset " : Token size

Scaling laws and Compute optimal models:

The final goal of training any model is to maximize model performance by decreasing model loss. (Aka. Compute optimal models)

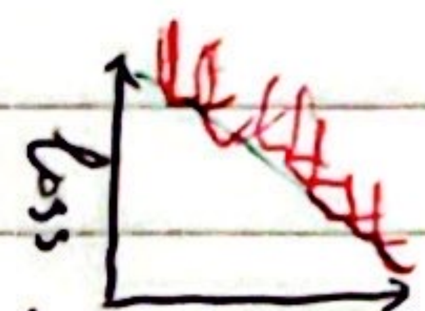
To do so, we can increase:
- → Dataset size (number of tokens)
- → Model size (number of parameters)

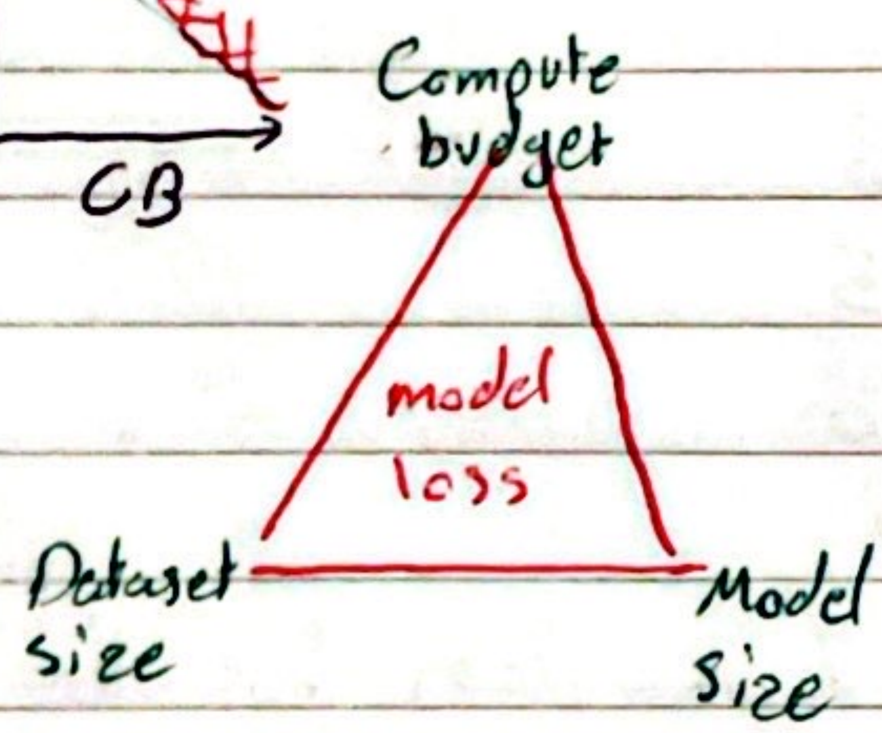
but we also have compute budget (GPUs, training time, Cost) as Constraints.

- we use petaflops/s-day as a unit to show the Compute budget need for training each model
  - → 1 "pflop/s-day" = floating point ops performed at rate of 1 petaFLOP/s for one day
  - → equal to 8 Nvidia V100s or 2 Nvidia A100s

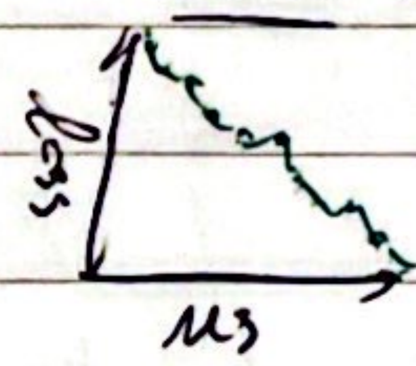
How is the relation of each variable in model loss?

* by Considering Dataset size and model size fixed, loss decreases with improved CB

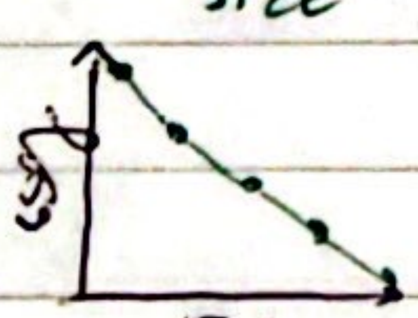




* and yet again, increase in model size ends up with lower loss



* again, by fixing two other variables, loss decreases with larger DS



And is there any relation/optimal size and relationship between them?

In Chinchilla paper, it is shown that there is a ≃ 20X relation between model size and Dataset size; to have an optimal model with size of xB parameters, we need 20x tokens:

Model	# of params	Compute optimal # of tokens (20x)	Actual # tokens	
Chinchilla	70B	~1.4T	1.4 B	
LLaMa-65B	65B	~1.3T	1.4 B ✓	
GPT-3	175B	~3.5T	300 B	under-trained
OPT-175B	175B	~3.5T	180B	under-trained
BLOOM	176B	~3.5T	350B	under-trained

- So in contrast of what we thought, just having bigger models won't end us up with better performance

* in one new paper, it is shown that not only chichilla rule is correct, but having a higher quality dataset also may end up in better performance models, with smaller datasets (~5x) than chinchilla - models
it is achived by { - running a classifier on data set and hand picking related texts for model
- separating more useful texts by keyword search.

Pre-training for domain adaptation

If the domain we want to work on, has an specific vocabulary and language structure which is uncommon, Pre-train is necessary.

for instance, legal-based LLMs should be pre-trained on legal data, due to their specific terms, vocab and language structure.

Another great exemple is Bloomberg GPT, which strickly tries to apply chichilla scaling rule, trained on { 51% Public and Private financial data
49% Public (other domain) data

↳ it has specific rules for
Compute budget and model structure data