



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

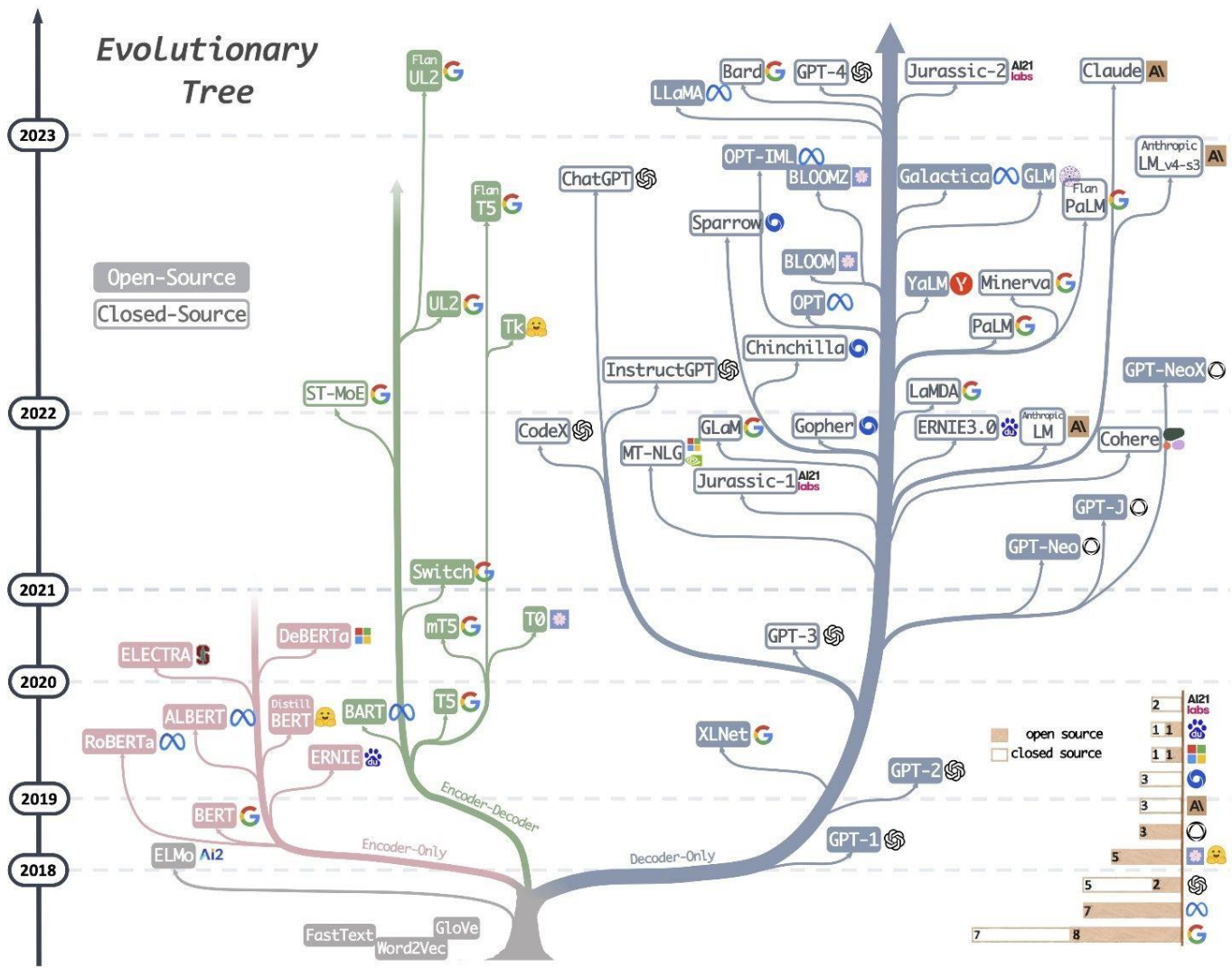
Transformer Language Models

CSCI 601-771 (NLP: Self-Supervised Models)

<https://self-supervised.cs.jhu.edu/fa2025/>

After Transformer ...

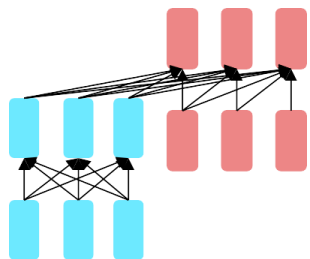




Yang et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, 2023

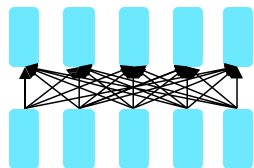
Impact of Transformers

- A building block for a variety of LMs



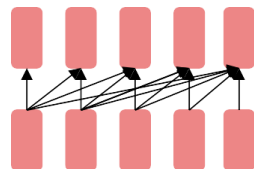
Encoder-
Decoders

- ❖ Examples: Transformer, T5, Meena
- ❖ What's the best way to pretrain them?



Encoders

- ❖ Examples: BERT, RoBERTa, ModernBERT.
- ❖ Captures bidirectional context. Wait, how do we pretrain them?



Decoders

- ❖ Examples: GPT-3, Gemini
- ❖ Other name: causal or auto-regressive language model
- ❖ Nice to generate from; can't condition on future words



How consistent are the architectures
used in existing LLMs?



Another View of Architectural Variations

Aa Name	# Year	⊙ Norm	⊙ Parallel Layer	☑ Pre-norm	⊙ Position embedding	⊙ Activations
Original transformer	2017	LayerNorm	Serial	<input type="checkbox"/>	Sine	ReLU
GPT	2018	LayerNorm	Serial	<input type="checkbox"/>	Absolute	GeLU
T5 (11B)	2019	RMSNorm	Serial	<input checked="" type="checkbox"/>	Relative	ReLU
GPT2	2019	LayerNorm	Serial	<input checked="" type="checkbox"/>	Sine	GeLU
T5 (XXL 11B) v1.1	2020	RMSNorm	Serial	<input checked="" type="checkbox"/>	Relative	GeGLU
mT5	2020	RMSNorm	Serial	<input checked="" type="checkbox"/>	Relative	GeGLU
GPT3 (175B)	2020	LayerNorm	Serial	<input checked="" type="checkbox"/>	Sine	GeLU
GPTJ	2021	LayerNorm	Parallel	<input checked="" type="checkbox"/>	RoPE	GeLU
LaMDA	2021			<input checked="" type="checkbox"/>	Relative	GeGLU
Gopher (280B)	2021	RMSNorm	Serial	<input checked="" type="checkbox"/>	Relative	ReLU
GPT-NeoX	2022	LayerNorm	Parallel	<input checked="" type="checkbox"/>	RoPE	GeLU
BLOOM (175B)	2022	LayerNorm	Serial	<input checked="" type="checkbox"/>	Alibi	GeLU
OPT (175B)	2022	LayerNorm	Serial	<input checked="" type="checkbox"/>	Absolute	ReLU
PaLM (540B)	2022	RMSNorm	Parallel	<input checked="" type="checkbox"/>	RoPE	SwiGLU
Chinchilla	2022	RMSNorm	Serial	<input checked="" type="checkbox"/>	Relative	ReLU
Mistral (7B)	2023	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU
LLaMA2 (70B)	2023	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU
LLaMA (65B)	2023	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU
Qwen (14B)	2024	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU
DeepSeek (67B)	2024	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU
Yi (34B)	2024	RMSNorm	Serial	<input checked="" type="checkbox"/>	RoPE	SwiGLU

Low consensus
(except pre-norm)

Most try to follow
previous successful
choices.

[Slide credit: Tatsu Hashimoto]



When should we do
normalization?

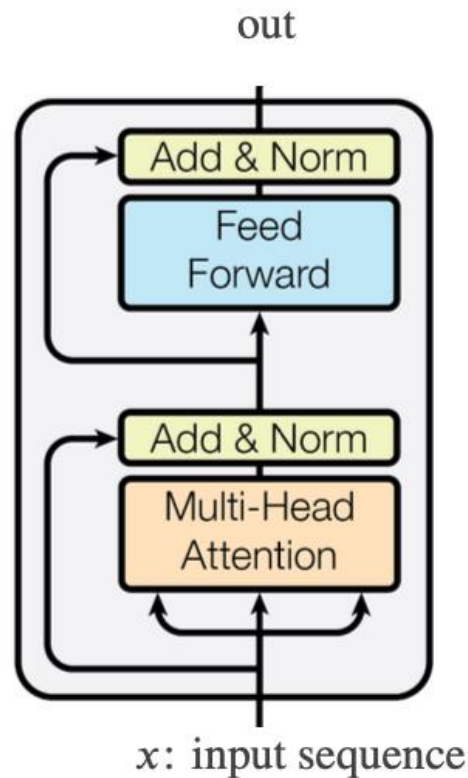


Quiz: Pre-norm vs Post-norm

- Which is the original implementation?
- Which one is your favorite?

$$\text{LayerNorm}(x + \text{SubLayer}(x))$$

$$x + \text{SubLayer}(\text{LayerNorm}(x)),$$



Pre-norm vs Post-norm

- Pre-norm (right) is set up so that LayerNorm does not disrupt the residual stream (in gray).
- In theory, both should work fine.
- In practice, however, **Pre-norm is preferred over Post-norm.**

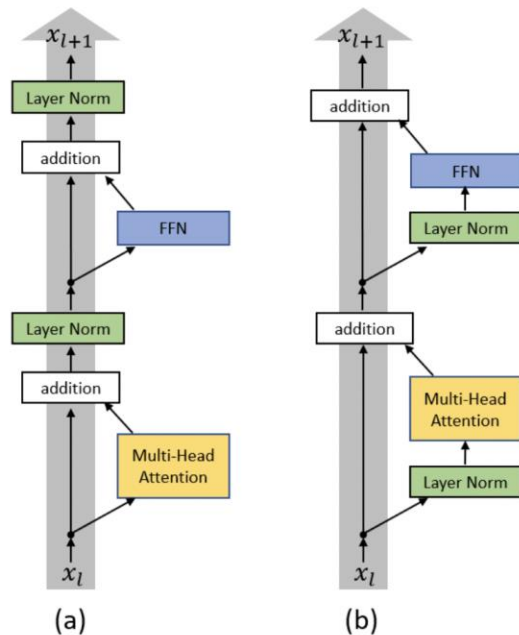
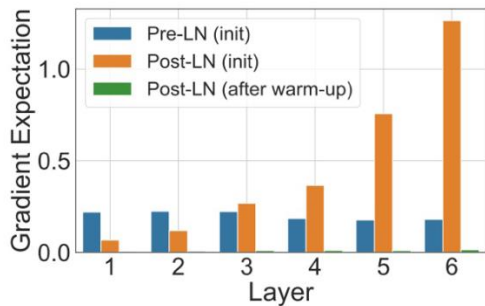


Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

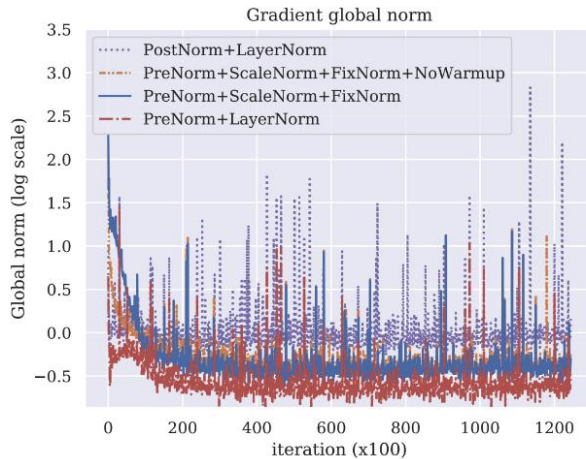
Pre-norm vs Post-norm — Explanation?

- Stability, larger LR for large networks and no need for warm up.

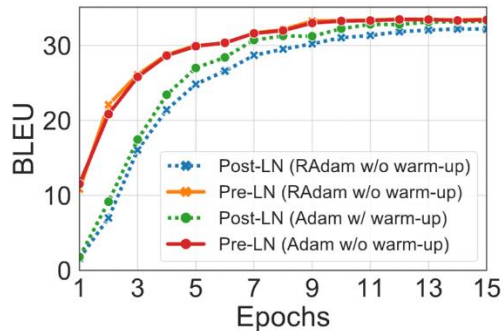
Gradient attenuation



Gradient spikes



No need for warm-up stage



(b) BLEU (IWSLT)



Serial vs Parallel layers



Serial vs Parallel Layer

Notable models:

GPT-J, PaLM, GPT-NeoX

- Normal transformer blocks are serial – they compute attention, then the MLP
 - Can they be parallelized? GPT-J introduced a simple change to do so!

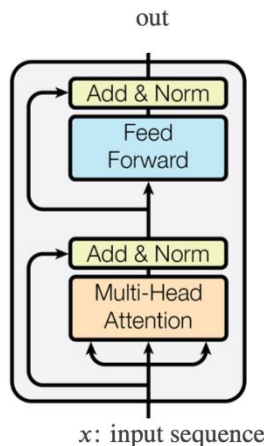
- The standard “serial” formulation:

$$y = x + \text{MLP}(\text{LayerNorm}(x + \text{Attention}(\text{LayerNorm}(x))))$$

- The parallel formulation:

$$y = x + \text{MLP}(\text{LayerNorm}(x)) + \text{Attention}(\text{LayerNorm}(x))$$

- Note, LayerNorm can be shared, and matrix multiplies can be fused
- From PaLM paper: *“The parallel formulation results in roughly 15% faster training speed at large scales ... Ablation experiments showed a small quality degradation at 8B scale but no quality degradation at 62B scale”*



Recap

- **Pre-vs-post norm:**
 - Everyone does pre-norm (except OPT350M).
- **LayerNorm vs RMSnorm:**
 - RMSnorm has clear compute wins, sometimes even performance.
- **Gating:**
 - GLUs seem generally better, though differences are small
- **Serial vs parallel layers:**
 - No extremely serious ablations; but parallel layers have a compute win.

Architecture Hyperparams

There are a ton of question regarding architecture hyperparameters:

- How much bigger should the feedforward size be compared to hidden size?
- How many heads? Should # of heads always divide hidden size?
- Should we make our model wide or deep?

The Surprising Consensus #1: FFN Dimension Ratio

- **Feedforward – model dimension ratio:**

$$\begin{aligned}\text{FFN}(\mathbf{x}) &= f(\mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2 \\ \mathbf{W}_1 &\in \mathbb{R}^{d \times d_{\text{ff}}}, \\ \mathbf{W}_2 &\in \mathbb{R}^{d_{\text{ff}} \times d}\end{aligned}$$

- There are two dimensions that are relevant – the feedforward dim (d_{ff}) and model dim (d). What should their relationship be?

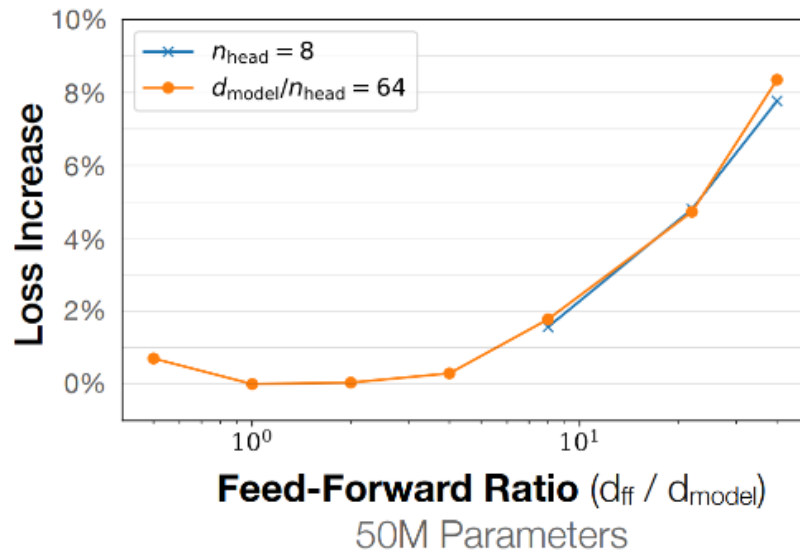
$$d_{\text{ff}} = 4d$$

- This is *almost* always true. There's just a few exceptions.

[Slide credit: Tatsu Hashimoto]

Why this range of multipliers?

- Empirically, there's a basin between 1-10 where this hyperparameter is near-optimal.



Summary of LLM architectures

- There are many architectural variations.
- Major differences? Position embeddings, activations, tokenization
- This is an evolving field; a lot of empirical analysis is going into identifying best practices.

Aa Name	Has pa...	Link	# Year	Tokenizer type	# Vocab count	Norm	Parallel Layer	Pre-norm	Position embedding	Activations	MoE	# MLP factor	# num_layers	# model_dim
Original transformer	Yes	arxiv.org/abs_03762	2017	BPE	37000	LayerNorm	Serial		Sine	ReLU		4	6	
GPT	Yes	cdn.openai.com/res_...er.pdf	2018	BPE	40257	LayerNorm	Serial		Absolute	GeLU		4	12	
GPT2	Yes	cdn.openai.com/bet_...rs.pdf	2019	BPE	50257	LayerNorm	Serial		Sine	GeLU		4	48	
T5 (11B)	Yes	arxiv.org/abs_10683	2019	SentencePiece	32128	RMSNorm	Serial		Relative	ReLU		64	24	
GPT3 (175B)	Yes	arxiv.org/abs_14165	2020	BPE	50257	LayerNorm	Serial		Sine	GeLU		4	96	
mT5	Yes	arxiv.org/abs_11934	2020	SentencePiece	250000	RMSNorm	Serial		Relative	GeGLU		2.5	24	
T5 (XXL 11B) v1.1	Kind of	github.com/goo_d#511	2020	SentencePiece	32128	RMSNorm	Serial		Relative	GeGLU		2.5	24	
Gopher (280B)	Yes	arxiv.org/abs_11446	2021	SentencePiece	32000	RMSNorm	Serial		Relative	ReLU		4	80	
Anthropic LM (not claude)	Yes	arxiv.org/abs_00861	2021	BPE	65536							4	64	
LaMDA	Yes	arxiv.org/abs_08239	2021	BPE	32000				Relative	GeGLU		8	64	
GPTJ	Kind of	huggingface.co/Ele_t-j-6b	2021	BPE	50257	LayerNorm	Parallel		RoPE	GeLU			28	
Chinchilla	Yes	arxiv.org/abs_15556	2022	SentencePiece	32000	RMSNorm	Serial		Relative	ReLU		4	80	
PaLM (540B)	Yes	arxiv.org/abs_02311	2022	SentencePiece	256000	RMSNorm	Parallel		RoPE	SwiGLU		4	118	
OPT (175B)	Yes	arxiv.org/abs_01068	2022	BPE	50272	LayerNorm	Serial		Absolute	ReLU		4	96	
BLOOM (175B)	Yes	arxiv.org/abs_05100	2022	BPE	250680	LayerNorm	Serial		Alibi	GeLU		4	70	
GPT-NeoX	Yes	arxiv.org/pdf_45.pdf	2022	BPE	50257	LayerNorm	Parallel		RoPE	GeLU		4	44	
GPT4	OPEN	Ad arxiv.org/abs_08774	2023	BPE	100000									
LLaMA (65B)	Yes	arxiv.org/abs_13971	2023	BPE	32000	RMSNorm	Serial		RoPE	SwiGLU		2.6875	80	
LLaMA2 (70B)	Yes	arxiv.org/abs_09288	2023	BPE	32000	RMSNorm	Serial		RoPE	SwiGLU		3.5	80	
Mistral (7B)	Yes	arxiv.org/abs_06825	2023	BPE	32000	RMSNorm	Serial		RoPE	SwiGLU		3.5	32	

Pre-training data

The pre-training data size and sources

- They vary quite a bit!
- They used to be in billions of tokens; now they're north of trillions.

Model Name	Release	Pre-training data #Tokens	Training Dataset
BERT	2018	3.3B	BooksCorpus (800M), English Wikipedia (2.5B)
GPT-1	2018	13B	BooksCorpus
GPT-2	2019	40B	WebText: scraping outbound links from Reddit post with ≥ 3 karma
T5	2019	34B	C4 which is the cleaned up version of CommonCrawl
GPT-3	2020	400B	Common Crawl (filtered), WebText2, Myrstry books!! (Books1, Books2), Wikipedia
Gopher	2021	1.4T	MassiveText
BLOOM	2022	350B	ROOTS corpus, a dataset comprising hundreds of sources in 46 natural and 13 programming languages (59 in total)
PaLM	2022	2.81T	Web documents, books, Wikipedia, conversations, GitHub code
LaMDA	2022	1.56T	Public dialog data and web documents
Chinchilla	2022	1.4T	MassiveText
LLaMA2	2023	2.0T	A new mix of publicly available online data
GPT-4	2023	?	?
Claude-3	2023	?	?
OLMo 2	2024	5.6T	OLMo-Mix-1124(stage1) + Dolmino-Mix-1124(stage 2)
Qwen2.5	2024	7T	
DeepSeek (V3)	2024	14.8T	GitHub's Markdown and StackExchange
LLaMA3	2024	15T	A new mix of publicly available online data

Where do we begin to collect data?

- Where do I find a very large dataset?
 - Crawling web is non-trivial (unless you're OpenAI or Google with ton of resources).
 - But if you have to do it, be aware that websites have their own permissions regarding which parts of their content, if any, can be crawled. (next slide)
- The alternative is to look for websites that have done the crawling for you.

CommonCrawl



- A non-profit organization that release a new crawl of the internet **every month**.
 - So far, there have been ~100 crawls from 2008-2024.
 - In 2016, a crawl took 10-12 days on 100 machines. They used [Apache Nutch](#).
 - This is **not** a complete of the internet. Crawls have some overlap but try to diversify.
 - Common Crawl follows links from previously crawled pages.
 - Also note, it respects robots.txt
- CC is a common sources of pre-training data.
 - **WARC**: The raw HTTP responses, including full web pages.
 - **WAT**: The metadata summary from WARC files.
 - **WET**: The extracted plaintext from WARC files, stripping out HTML and other non-textual content.

Data Type	File List	#Files	Total Size Compressed (TiB)
Segments	segment.paths.gz	100	
WARC	warc.paths.gz	90000	76.08
WAT	wat.paths.gz	90000	17.68
WET	wet.paths.gz	90000	7.00
Robots.txt files	robotstxt.paths.gz	90000	0.15
Non-200 responses	non200responses.paths.gz	90000	2.59
URL index files	cc-index.paths.gz	302	0.19
Columnar URL index files	cc-index-table.paths.gz	900	0.22

<https://data.commoncrawl.org/crawl-data/CC-MAIN-2024-30/index.html>



CC is messy. Is that a concern?



Besides quantity, the choice of dataset is also critical

Garbage in Garbage OUT



C4: A cleaned up pre-training dataset

- C4: Colossal Clean Crawled Corpus
 - The corpus is CommonCrawl.
 - English language only
 - 750GB after ton of filtering

Data set	Size
★ C4	745GB
C4, unfiltered	6.1TB

- Notice that the unfiltered data is quite large.
 - Common Crawl is mostly not useful natural language

C4: The Data

Remove any:

- References to Javascript
- Pages with "{" (no code), "[Lorem ipsum](#)" text (dummy text), "terms of use", etc.
- Pages with "[bad words](#)".

Menu

Lemon

Introduction

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae. The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lemon is unknown, though

Retain:

- Sentences with terminal punctuation marks
- Pages with at least 5 sentences, sentences with at least 3 words

Please enable JavaScript to use our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, \$3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae. The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur in tempus quam. In mollis et ante at consectetur. Aliquam erat volutpat. Donec at lacinia est. Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit. Fusce quis blandit lectus. Mauris at mauris a turpis tristique lacinia at nec ante. Aenean in scelerisque tellus, a efficitur ipsum. Integer justo enim, ornare vitae sem non, mollis fermentum lectus. Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {  
  this.radius = r;  
  this.area = pi * r ** 2;  
  this.show = function(){  
    drawCircle(r);  
  }  
}
```

Pre-training Data: Experiment

- Takeaway:
 - Clean and compact data is better than large, but noisy data.
 - Pre-training on in-domain data helps.

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	19.24	80.88	71.36	26.98	39.82	27.65
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21



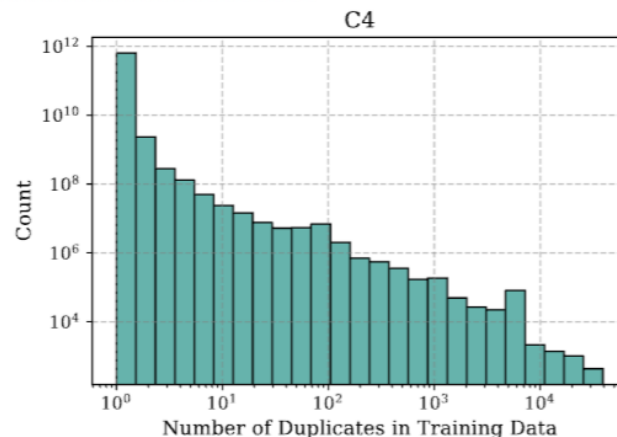
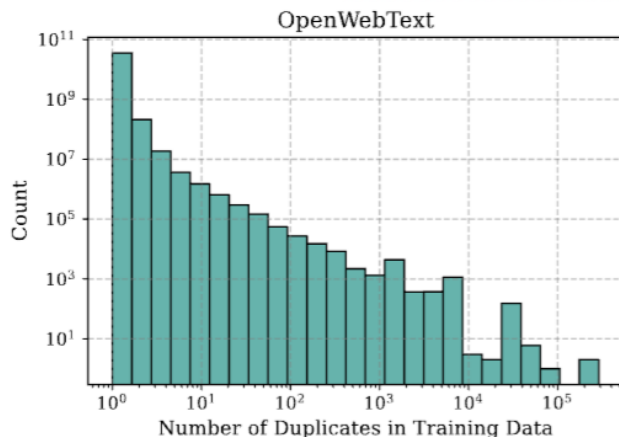
Does it matter that my
data has ton of repetitions?



Pre-training Data Duplicates

- There is a non-negligible number of duplicates in any pre-training data.

	% train examples with dup in train		% valid with dup in train
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%



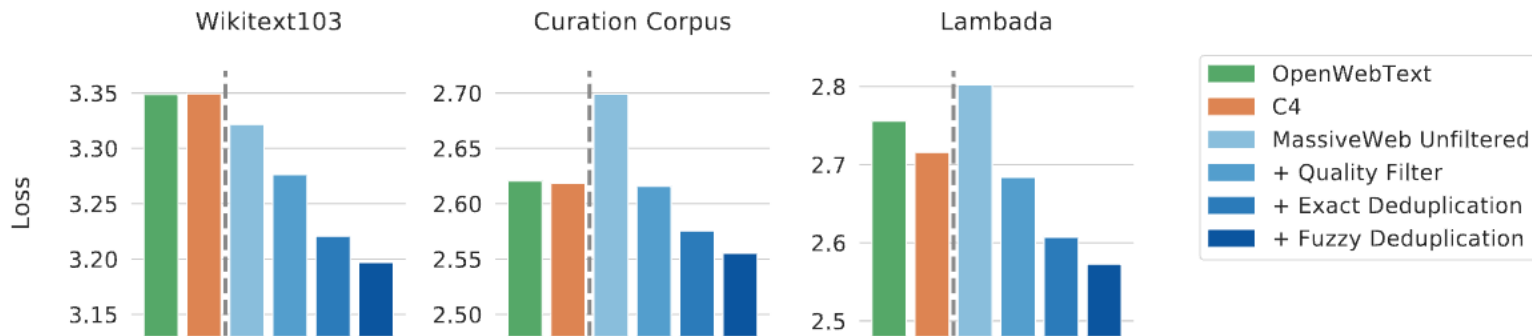
Pre-training Data Duplicates

- There is a non-negligible number of duplicates in any pre-training data.
- Maybe we should not spend our training budget re-learning things we have already seen.

Dataset	Example	Near-Duplicate Example
Wiki-40B	<code>\n_START_ARTICLE\nHum Award for Most Impactful Character \n_START_SECTION\nWinners and nominees\n_START_PARAGRAPH\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]</code>	<code>\n_START_ARTICLE\nHum Award for Best Actor in a Negative Role \n_START_SECTION\nWinners and nominees\n_START_PARAGRAPH\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]</code>
LM1B	<code>I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .</code>	<code>I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .</code>
C4	<code>Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!</code>	<code>Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!</code>

Deduplicating Data Improves LMs

- Another evidence from Gopher paper: Performance of 1.4B parameter models (lower is better) trained on OpenWebText, C4, and versions of MassiveWeb with progressively more pre-processing stages added.
- Applying a quality filter and de-duplication stages significantly improves quality.





How can I do my own
deduplication?



How do you scale data deduplication?

- Pre-training is huge. Naively deduplicating the data is going to take forever!!
- How do you deduplicate it? Here are a few options:
 - Naively hashing each document (a good baseline)
 - SuffixArray
 - MinHash
 - BloomFilters
 - Embedding-based dedup

Comparison between dedup algorithms

- Single methods: BF better than any other method standalone.
- Combination: The competitive approaches are last row (exact -> MH -> SA) and BF-only. The former leads to more compact data.

	Exact Dedup	MinHash	Suffix Array	Bloom Filter	Tokens	Removal Rate	CORE	Δ from Baseline
Individual technique	X	X	X	X	76B	00%	40.1	+0.0
	✓	X	X	X	66B	13%	41.0	+0.9
	X	✓	X	X	62B	18%	40.9	+0.8
	X	X	✓	X	51B	33%	41.4	+1.3
	X	X	X	✓	56B	26%	41.7	+1.6
Combined techniques	✓	✓	X	X	58B	24%	40.2	+0.1
	✓	X	✓	X	49B	36%	41.3	+1.3
	X	✓	✓	X	48B	37%	41.2	+1.2
	✓	✓	✓	X	45B	41%	41.7	+1.6

Deduplication: Recap

- Does it matter that my data has ton of repetitions? Yes, one should do careful dedup.
- How can I do my own deduplication?
 - Scaling it up requires advanced data structures.
 - So far, there is no clear winner between these algorithms. A “kitchen sink” approach that mixes dedup algorithms is generally best, but it’s an empirical exercise.
 - BF is generally preferred since it’s cheaper/faster.

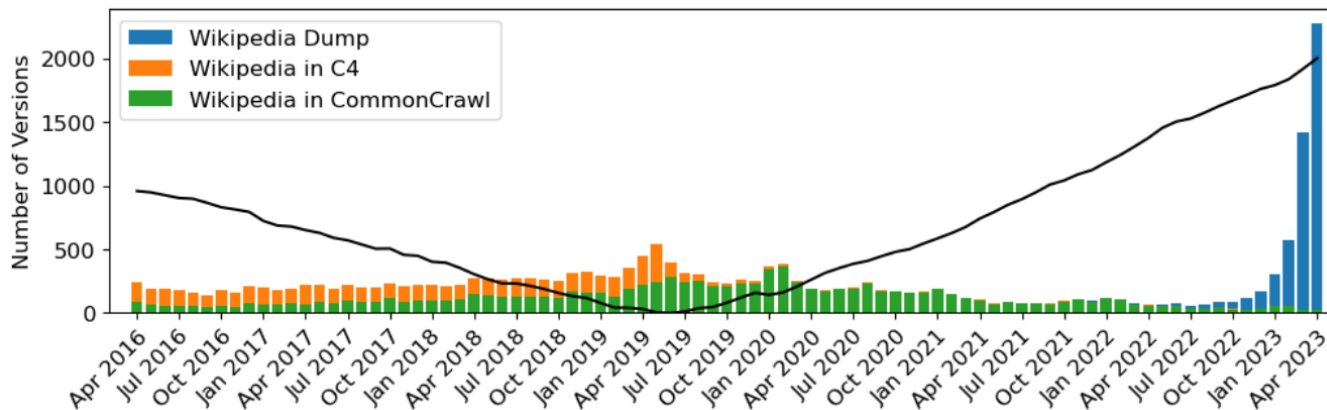


Should I worry about old data
in my pre-training?



Prevalence of stale data: RedPejamas

- Breakdown of old versions of Wikipedia in RedPejamas
 - It is based on dumps from C4, CC and a recent Wikipedia dump.
- The bars blow show the breakdown of older versions of Wikipedia in RedPajamas.
 - There is a ton of old Wikipedia versions in RedPejamas! 🤖
- The solid trend is the perplexity of a pre-trained model on temporal instances of Wikipedia.
 - The significant stale training data in has skewed PPL toward older versions of Wikipedia.



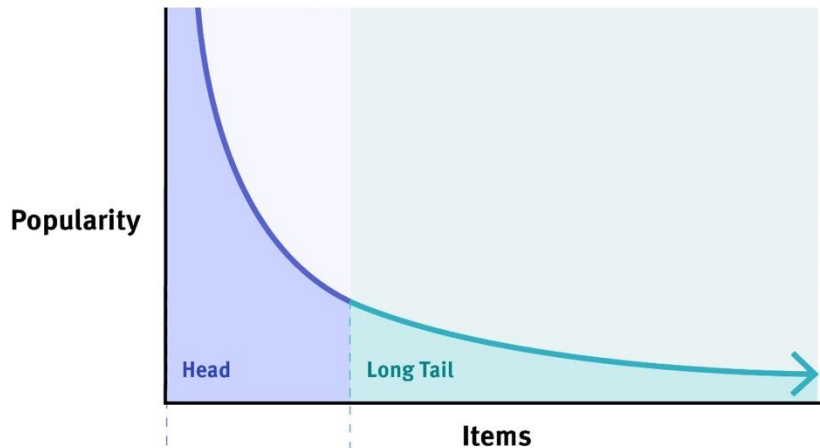


Should I worry about skew of
the data mixtures in my pre-training?



Data mixtures (and the long tail)

- Your dataset mixture will determine the versatility of the resulting model.
- Data in the world is always skewed. For example,
 - English has a lot more language than other domains.
 - Reddit is a lot larger than science papers.



Data mixtures (and the long tail)

- Your dataset mixture will determine the versatility of the resulting model.
- Data in the world is always skewed. For example,
 - English has a lot more language than other domains.
 - Reddit is a lot larger than science papers.
- A uniform “weight” of data during pre-training is not good since overrepresented domains would dominate (e.g., your model would be a better at English than Azeri).
- Overamplifying underrepresented domains also runs risk of overfitting.
- So, there is a lot of research on finding **a good balance**.

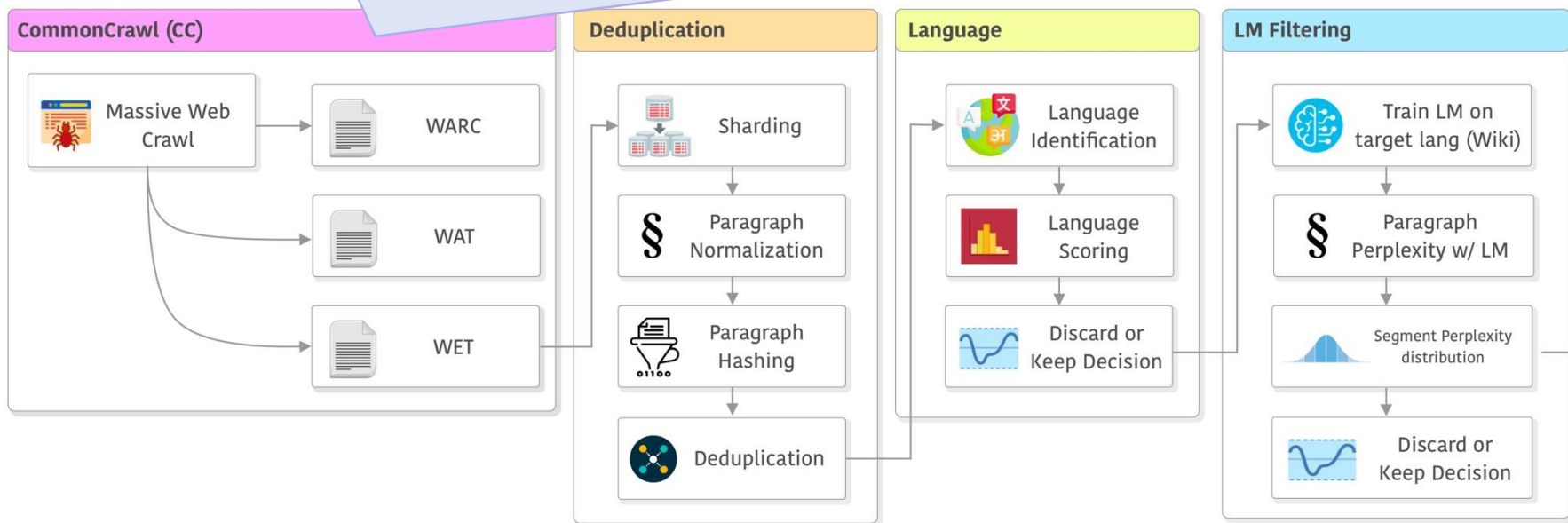


Few notable data pipelines



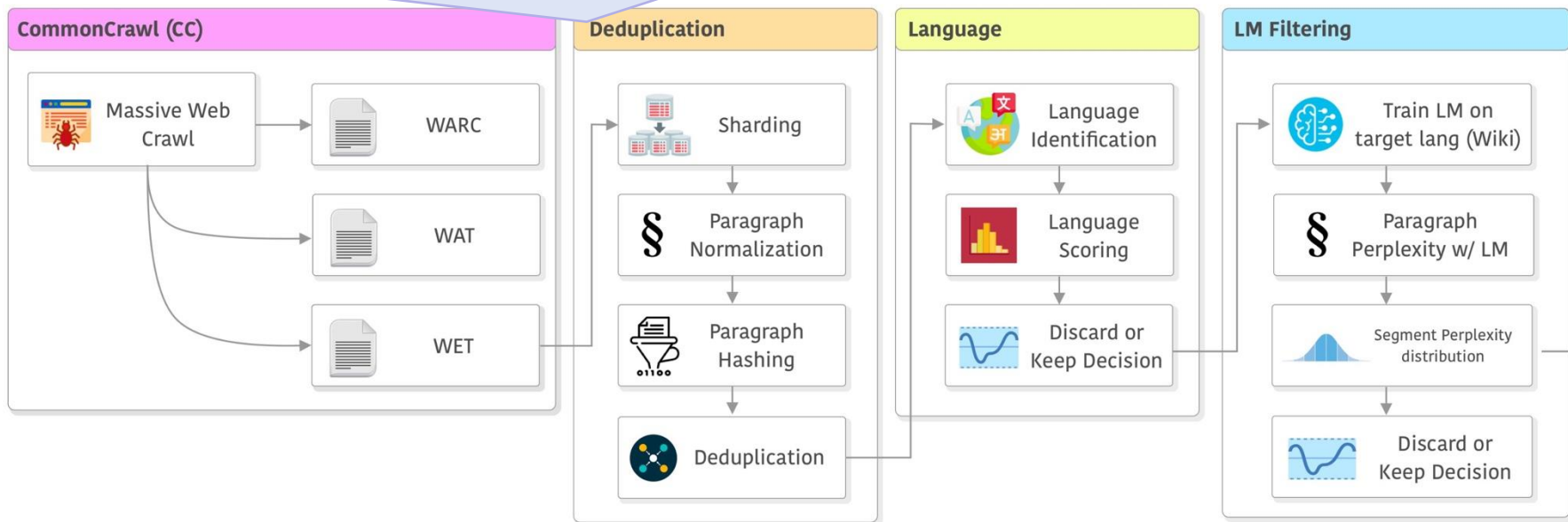
LLaMA 1's Data Pipeline

Starts with the massive crawled data by CommonCrawl.
The WET format that contains textual information.
WARC is raw, WAT is metadata, WET is text+some metadata.



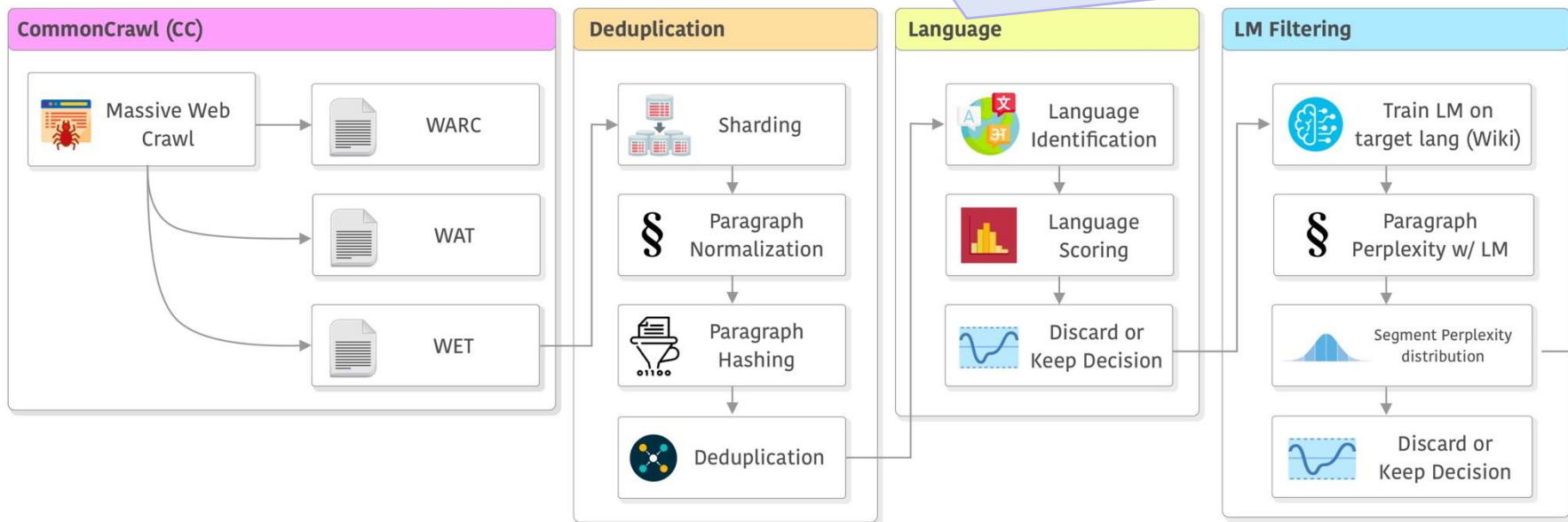
LLaMA 1's Data Pipeline

Shard WET content into shards of 5GB each (one CC snapshot can have 30TB). Then you normalize paragraphs (lowercasing, numbers as placeholders, etc), compute per-paragraph hashes and then duplicate them.



LLaMA 1's Data Pipeline

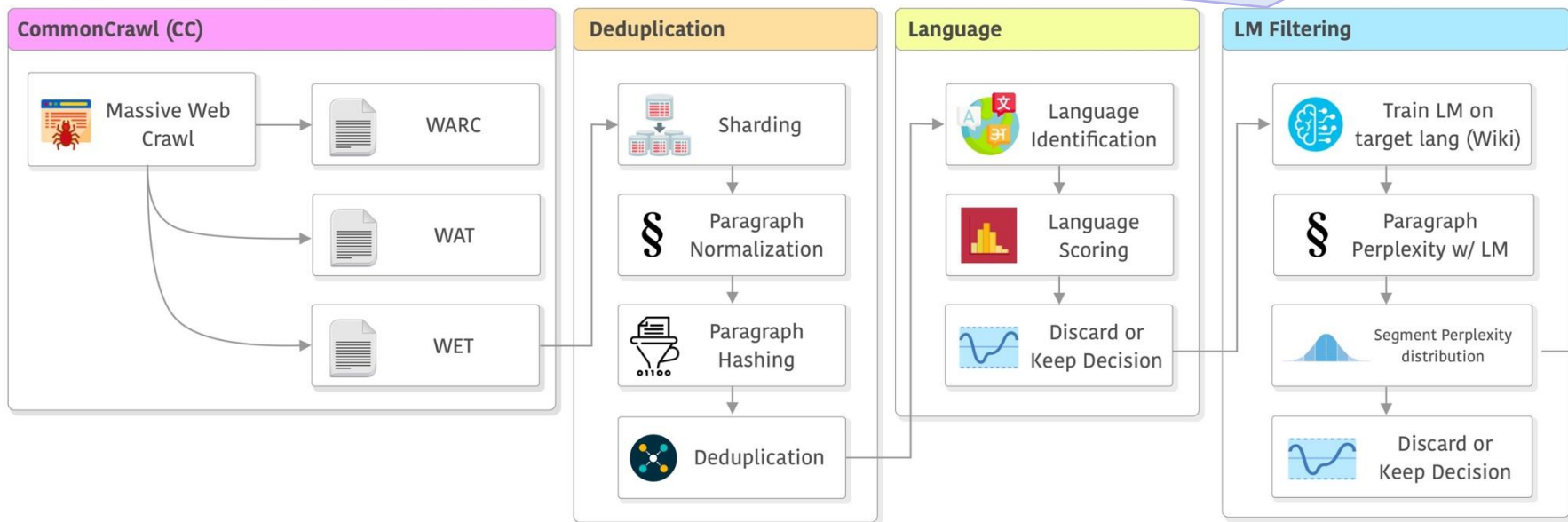
Perform language identification and decide whether to keep or discard languages. The order of when you do this in the pipeline can impact the language discrimination quality.



LLaMA 1's Data Pipeline

Do further quality filtering: Train a simple LM (n-gram) on target languages using Wikipedia, then compute per-paragraph perplexity on the rest of the data:

- Very high PPL: Very different than Wiki and likely low-quality → Drop
- Very low PPL: Very similar or near duplicates to Wiki → Drop



DataDecomp-LM filtering pipeline

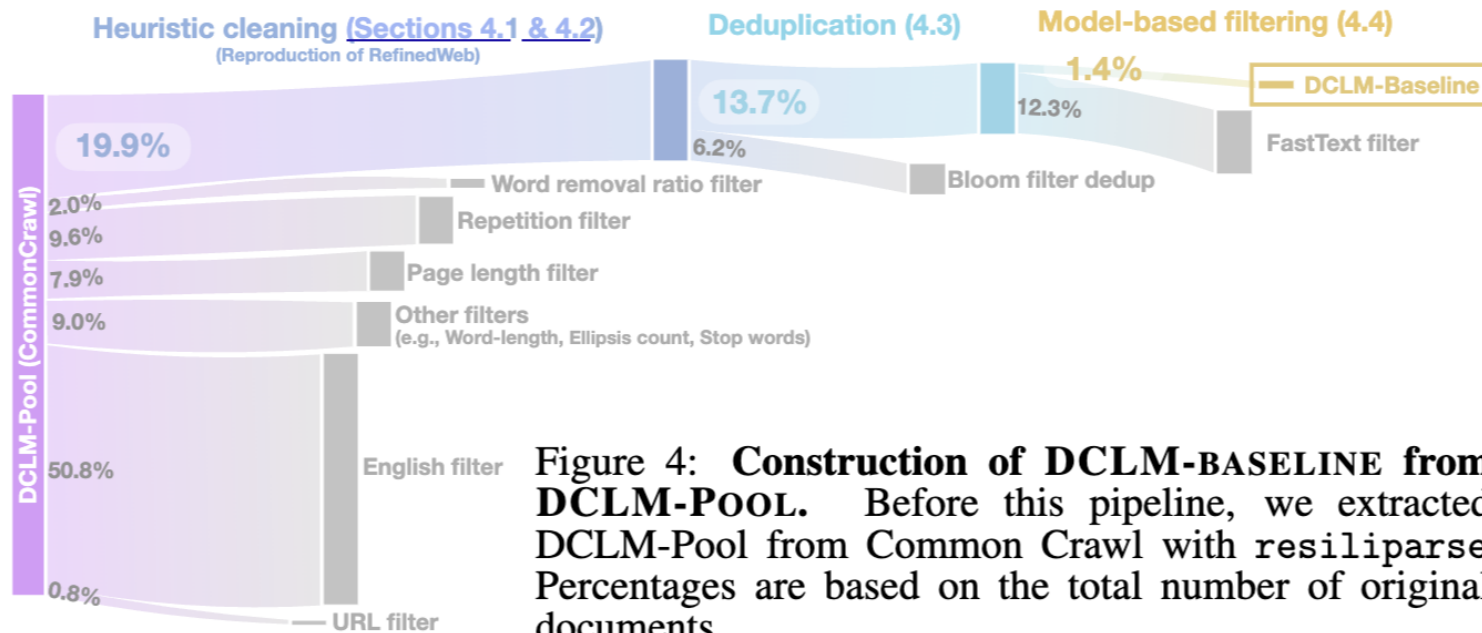


Figure 4: **Construction of DCLM-BASELINE from DCLM-POOL.** Before this pipeline, we extracted DCLM-Pool from Common Crawl with resiliiparse. Percentages are based on the total number of original documents.

Few cleaned-up pre-training datasets

Dataset	Example models	Tokens	Source	License	Lang
C4 (Raffel et al. 2020)	T5	165B	CC	ODC-BY	English
The Pile (Gao et al. 2020)	GPT-J, Pythia	300B	22 datasets including CC, books, code, news	Varies by dataset subset	English
RedPajamas (Weber et al. 2024)	Llama 1	1.2T	CC, C4, Github, Arxiv, Books, Wikipedia, StackExchange	Varies by dataset subset	English
RefinedWeb (Penedo et al. 2023)	Falcon	600B	CC	ODC-BY 1.0	English
Dolma (Soldaini et al. 2024)	OLMo	3T	CC, C4, Gutenberg, Github, Wikipedia, Wikibooks	ImpACT MR	English
DataComp-LM (Li et al. 2024)	SmolLM2, DCLM	240T	CC	?	English

The Pile

- Pile-CC: From Common Crawl; uses [justText](#) to extract useful text.
- PubMed Central: 5M NIH funded papers and public.
- arXiv: preprint for research papers since 1991 (uses latex).
- Gutenberg [PG-19](#): Online books (before 2019) with copyright clearance.
- Books3 is a collection of ~200K books. Has been [subject of lawsuits](#).
- StackExchange: Q&A format is close to real applications.
- Github: Content is not just the code.
 - Note, [GH archive](#) has regular snapshots of Github (commits, forks, etc.)

Component	Raw Size
Pile-CC	227.12 GiB
PubMed Central	90.27 GiB
Books3 [†]	100.96 GiB
OpenWebText2	62.77 GiB
ArXiv	56.21 GiB
Github	95.16 GiB
FreeLaw	51.15 GiB
Stack Exchange	32.20 GiB
USPTO Backgrounds	22.90 GiB
PubMed Abstracts	19.26 GiB
Gutenberg (PG-19) [†]	10.88 GiB
OpenSubtitles [†]	12.98 GiB
Wikipedia (en) [†]	6.38 GiB
DM Mathematics [†]	7.75 GiB
Ubuntu IRC	5.52 GiB
BookCorpus2	6.30 GiB
EuroParl [†]	4.59 GiB
HackerNews	3.90 GiB
YoutubeSubtitles	3.73 GiB
PhilPapers	2.38 GiB
NIH ExPorter	1.89 GiB
Enron Emails [†]	0.88 GiB
The Pile	825.18 GiB

Summary: preparing pre-training data

- Data does not fall from the sky. You have to work to get it!
- **Finding large data:** CommonCrawl has a ton of crawled dumps, but not the only one.
- **Cleaning data** can save tons of compute and even give you gains.
- **Repetitions** are often a waste of compute and deteriorate model quality.
- **Scaling deduplication** requires advanced data structures.
- **Old data** old data may skew your model predictions, but it depends on your application.
- **Data mixtures** are quite important, though depend on your downstream application.

The actual pre-training



How should we select the
right hyperparams?

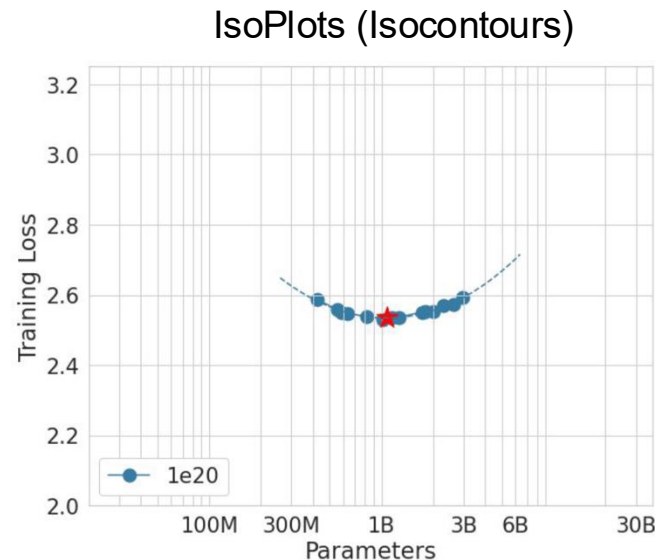


Q: What would you do?

- Zuckerberg gave you a \$500M budget for training Llama-10.
- You set aside \$10M for finding the best architecture at smaller scale, assuming that your ultimate model will be much larger.
- This way, you pick your parameters with rigorous experiments at small scale:
 - Optimal training params: Learning rate, warmup, weight decay, etc.
 - Architecture configs by scaling (each x50) the optimal values at small scale.
- Q: What you like (or don't) about this recipe?
- Optimal depth/width, lr, batch size, weight decay, init, and residual scaling are **not scale-invariant**.

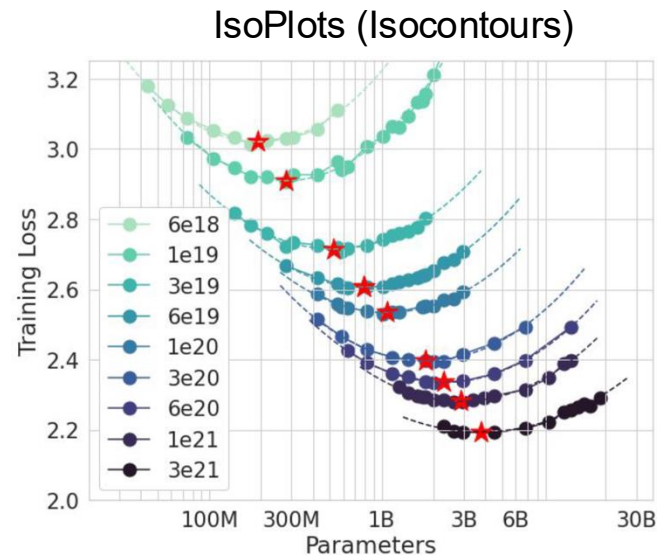
IsoPlots: Tradeoffs at a smaller scale

- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
 1. Fix a compute budget FLOP
 2. Train a few models and vary their size
 3. Fit a parabola and find the minimum



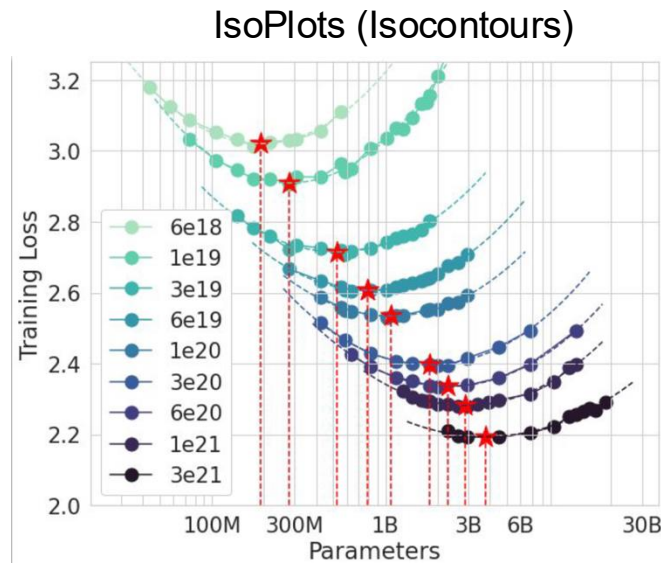
IsoPlots: Tradeoffs at a smaller scale

- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
 1. Fix a compute budget FLOP
 2. Train a few models and vary their size
 3. Fit a parabola and find the minimum
 4. Repeat 1-3 for various FLOP budgets



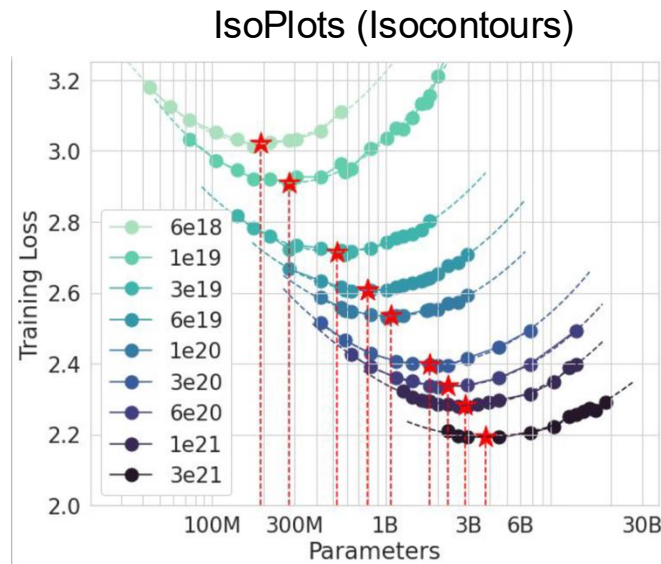
IsoPlots: Tradeoffs at a smaller scale


- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
 1. Fix a compute budget FLOP
 2. Train a few models and vary their size
 3. Fit a parabola and find the minimum
 4. Repeat 1-3 for various FLOP budgets
- It's good to change various parameter (e.g., training data, size, or other hyperparams) and see how it's quality (loss) changes.




Predictive models for parameters

- Overall, IsoPlots show how loss depends on three axes: model size (parameters N), dataset size (tokens D), and compute budget C .
- They're a tool for reasoning about *how to spend your training budget efficiently*.
- But it also allows one to see track effective hyperparams (LR, batch size, etc.) changes with N , C , or D .
- **Lesson:**
 - Don't overtune your hyperparameters at small scales and expect to use them at large
 - Instead develop predictive metrics based on parameters:



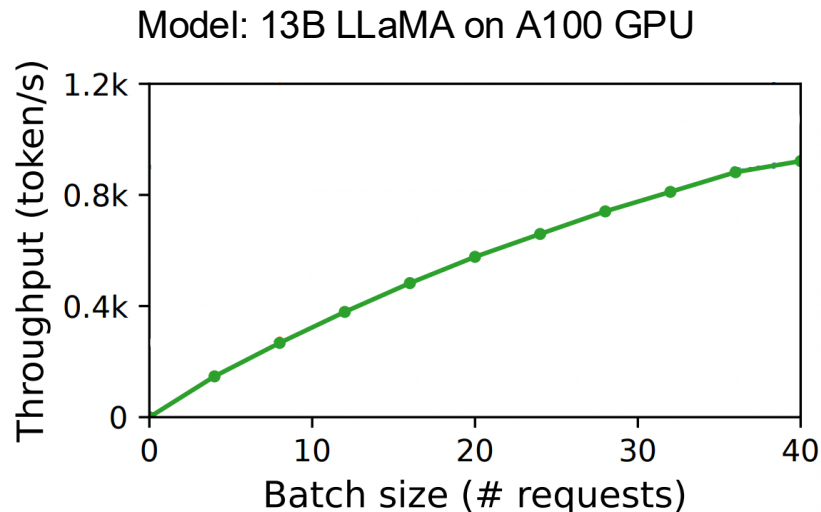


How should I
train the model?



Batching Data

- Previously we talked about the importance of batching data
- GPUs are faster at Tensor operations and hence, we want to do batch processing
- The larger batch of data, the faster they get processed.
- Alas, the speedup is often sub-linear (e.g., 2x larger batch leads to less than 2x speedup).
- If you can afford larger batch (larger GPU), it's generally worth it.



Another side of batching: gradient quality

- It's also about quality of your estimated gradients.
- Small batch sizes will result in **noisy gradients estimates**.
 - Therefore, the model may not be able to converge to the optimal performances.
- A large batch size while giving very accurate gradient estimations will tend to make **less use of each training token**
 - Slower convergence and potentially wasting compute.

Batch sizes: some known statistics

[LLaMA: Open and Efficient Foundation Language Models, 2023](#)

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

[The Llama 3 Herd of Models, 2024](#)

GPUs	TP	CP	PP	DP	Seq. Len.	Batch size/DP	Tokens/Batch	TFLOPs/GPU	BF16 MFU
8,192	8	1	16	64	8,192	32	16M	430	43%
16,384	8	1	16	128	8,192	16	16M	400	41%
16,384	8	16	16	8	131,072	16	16M	380	38%

Table 4 Scaling configurations and MFU for each stage of Llama 3 405B pre-training. See text and Figure 5 for descriptions of each type of parallelism.

[DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model, 2024](#)

is set to 1.0. We do not employ the batch size scheduling strategy for it, and it is trained with a constant batch size of 4608 sequences. During pre-training, we set the maximum sequence

Dropout and other regularization

- Do we need regularization during pretraining?
- Arguments against:
 - There is *a lot* of data (trillions of tokens), more than parameters.
 - SGD only does a single pass on a corpus (hard to memorize)
- This is all quite reasonable.... but what do people do in practice?

Dropout and weight decay in practice

Model	Dropout*	Weight decay
Original transformer	0.1	0
GPT2	0.1	0.1
T5	0.1	0
GPT3	0.1	0.1
T5 v1.1	0	0
PaLM	0	(variable)
OPT	0.1	0.1
LLaMA	0	0.1
Qwen 14B	0.1	0.1

Many older models used dropout during pretraining

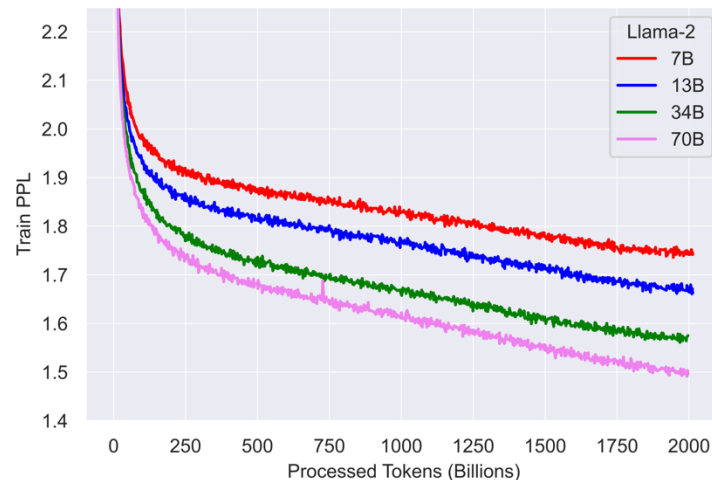
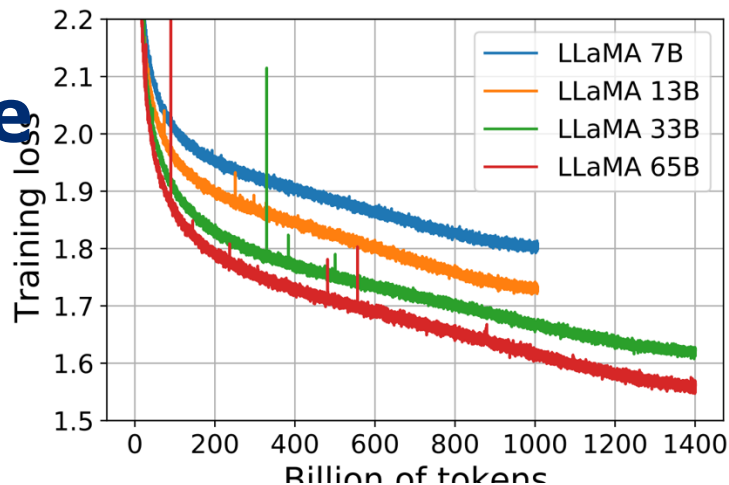
Newer models (except Qwen) rely only on weight decay

* Most of the times papers just don't discuss dropout. On open models, this closely matches not doing dropout. This may not be true of closed models.

[Slide credit: Tatsu Hashimoto]

Monitoring the convergence

- In practice, your model's loss should continue to go down with more training on more data.
- So, the real bottlenecks are:
 - (1) compute
 - (2) data
- Sometimes training diverges (spikes in the loss), at which point practitioners usually restart training from an earlier checkpoint.



Monitoring the convergence with end tasks

- Some works have also monitored **end task** performance during pre-training.
- Use **likelihood** of the correct answer rather than accuracy
 - (you don't even need to consider the incorrect answers)
- Very similar to the "cloze mmlu" trend where you use the probability of the full answer instead of A, B, C, D.
 - Not discrete metrics (e.g., Accuracy)

Monitoring the convergence with end tasks

- These two typically correlate, but not always.

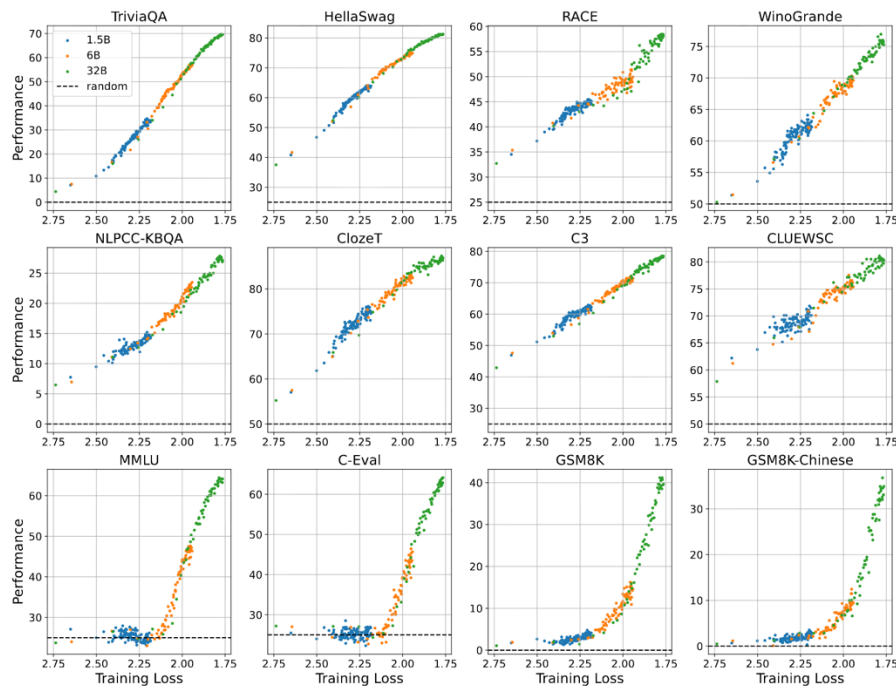


Figure 1: **The performance-vs-loss curves of 1.5B, 6B, and 32B models.** Each data point is the loss (x-axis) and performance (y-axis) of the intermediate checkpoint of one of the three models. We mark the results of random guess in black dashed lines.

Recap of training LLMs

- **IsoPlots:** for a fixed compute, which combination of parameters give you the best bang for the buck.
- Careful batching makes your training go brrr!
- Memory usage can be tricky since there are various moving parts.
 - More on distributed training later on.
- Dropout is less common but you still 'regularize' LMs via large-scale training.