Training Large Language Models (LLMs)

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 3: Data

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-628 (Spring 2025)

License Information for Training LLMs Slides

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Outline

- Motivation
- Data cleaning
- Tokenization
- Data selection
- Data mixing
- Distillation
- Reasoning data

Motivation: Data as a resource

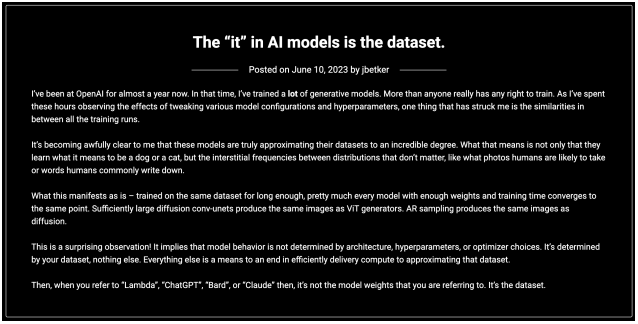


Figure: (Left) Economist 2017 article on the importance of data. (Right) The relative importance of data within the model.

- Architectures determine
  - inference resources
  - inference capabilities
- Algorithms determine
  - training resources
  - generalization
- Data determines
  - generalization
  - training resources

Data processing pipeline for LLM pretraining

- Modern LLMs (e.g., ChatGPT, Gemini, Claude, DeepSeek,...) employ the following general data pipeline:

1. Data cleaning
   - Deduplication
   - Privacy and copyright
   - Quality filtering
   - Data age
2. Data selection
   - Data masking
   - Document packing
   - Data ordering
   - Synthetic data
3. Data mixing
   - Find optimal domain mixtures
   - Target evaluations
   - Online mixing approaches
   - Offline mixing approaches

Data cleaning: Deduplication

- Datasets curated from internet contain substantial amounts of duplicate and near-duplicate content.
- This redundancy can negatively impact the performance of language models [51].
  - Redundancy skews the model's learning, resulting in overemphasis on repeated patterns.
- Effective deduplication helps to create a more balanced and varied training set.
  - It helps with model's generalization and alignment.
  - It reduces storage, which in turn reduces costs associated with infrastructure.
- We discuss three deduplication techniques in the sequel:
  1. Exact matching
  2. Approximate matching
  3. Model-based matching

Deduplication techniques: Exact matching

- We can use hashing for each document.
  - Two documents are duplicates, if their hashes coincide.
  - Secure hash algorithm (SHA)-1, or its first 64 bits, used in CCNet [89].
  - A simple hash function on large scale datasets has extremely low collision probability [24].
- We can use exact substring matching to identify duplicate substring.
  - [51] uses suffix arrays for efficient computation.

Deduplication techniques: Approximate matching

Dataset	Example	Near-Duplicate Example
Wiki-40B	\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. [...]	\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. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	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!	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!

Figure: Examples of near-duplicates identified by NEARDUP from multiple datasets (from [51]).

- NEARDUP: Efficiently identifies near-duplicates by estimating Jaccard similarity between documents [51].
- Edit Similarity: Similarity based on number of edits (insertions, deletions, substitutions) [51].
- Other similarity metrics used in practice include Levenshtein distance in PaLM [14].

Deduplication techniques: Model-based

- Use learned representations from pretrained models to identify semantically similar documents.
- SEMDEDUP [1]:
  - Uses a pretrained language model to generate embeddings.
  - Calculates a representation for each data point (e.g., last layer embedding of final token).
  - Clusters data points to find groups of similar documents.
  - Computes pairwise cosine similarity within each cluster.
- Document De-Duplication and Diversification (D4) [83]:
  - Combines hashing and clustering.
  - Removes "most prototypical" examples within each cluster, enhancing diversity [76].

Data cleaning: Privacy and copyright

- Privacy (anonymization)
  - Challenge: Web-scraped data often contains Personally Identifiable Information (PII), such as names, phone numbers, addresses, and emails.
  - Risk: An adversary can attack the LLM and extract training-data PPI [12, 53].
  - Mitigation:
    - Rule-based: Identify and replace PII with placeholders (e.g., using regular expressions).
    - Differential Privacy: Add noise to the data to protect individual privacy while preserving utility [22, 23].
- Copyright
  - Challenge: Much of the web content, e.g., books and articles, is copyrighted material.
  - Risk: Training LLMs on copyrighted data without permission or license can lead to legal issues.
  - Mitigation:
    - Obtain explicit permission from copyright holders.
    - Utilize publicly available datasets.

Example (Rule-based anonymization)

Original Text: "John Doe's phone number is 555-123-4567 and his email is john.doe@example.com."

Anonymized Text: "[NAME]'s phone number is [PHONE] and his email is [EMAIL]."

Data cleaning: Bias

- Real-world datasets are biased.
- Data imbalances hurt out-of-distribution performance [28].
- Bias is also within the same research vein as fairness and AI safety [15].

Example (Biased data)

A language model trained predominantly on news articles may exhibit bias towards formal language and struggle with informal slang or dialectical variations.

Remark:

- Mitigation techniques include
  - data augmentation with examples representing underrepresented groups [102]
  - contrastive objectives [34]
  - dropout regularization [87]
  - counterfactual data augmentation [87, 49, 61]

Data cleaning: Filtering low-quality data

- Pretraining datasets for LLMs often contain substantial amounts of low-quality, noisy, or undesirable content.
- Modern LLMs use quality and/or toxicity filtering to achieve optimal model performance and desired behavior.

Example

Consider the Common Crawl dataset, a massive web scrape. It includes

- advertisements and promotional material;
- repetitive, non-human-readable text;
- toxic content.

Filtering approaches

Two primary approaches are used, often in combination:

1. Heuristic-based filtering: Rule-based methods that remove documents based on easily computable features.
2. Classifier-based filtering: Models trained to identify and remove specific types of undesirable content.

Quality filtering: Heuristic-based

- Heuristic methods deploy rules based on simple text features [67, 95, 48, 100].
  - Sentence length: Remove documents with sentences that are too short or too long.
  - Specific-word presence: Filter documents based on the presence of specific words with low-quality content.
  - Repetitiveness: Eliminate documents with excessive repetition of characters or phrases.
  - Language: Ensure the document is primarily in the desired target language (e.g., English-only).

Example

Colossal, cleaned version of Common Crawl's web crawl corpus' (C4) heuristic filtering include [68]:

- Removing lines not ending in a terminal punctuation mark.
- Filtering documents with Javascript alerts.
- Removing documents with fewer than 5 sentences.
- Removing documents containing curly brackets (to filter out code).
- Filtering out filler text, such as "lorem ipsum," or boilerplate policy notes like "terms of use."

Quality filtering: Classifier-based

- Classifier-based methods use machine learning models to identify and filter undesirable content.
  - We train classifiers to distinguish between "high-quality" and "low-quality" documents.
  - High-quality training data is often derived from curated sources like Wikipedia or Books.

Example (Classifier-based quality filtering)

- PaLM and GLAM use a linear classifier based on features, e.g., similar to Wikipedia and Books [14, 21].
- LLaMA uses pages referenced within Wikipedia as high-quality examples [84].

Remarks:

- Classifier-based filtering might be too strict; we can use stochasticity in the selection [10]
  - to allow some lower-scoring documents to be included.

Quality filtering: Impact

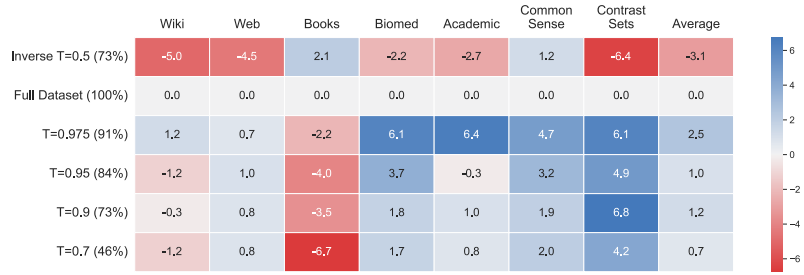


Figure: Performance when applying quality filtering to C4. The quality filter threshold is on the x-axis, with percentage of remaining training data in parenthesis. "Full Dataset" is unfiltered, "Inverse" removes highest quality data. From [59].

Remarks:

- Quality filtering improves performance on almost all downstream tasks.
- Removing $\approx 10\%$ of training data can significantly improve performance.

Data cleaning: Data age

- Language evolves, impacting the LLM performance on temporally misaligned data [59].
  - Temporal misalignment: Discrepancy between the time periods in training and evaluation [50, 60, 4].

Example

An LLM trained on 2019 data may perform poorly on a 2023 evaluation set containing information about events happened after 2019. Similarly, a model trained on 2023 data may struggle with nuances of language and information relevant to a 2019 evaluation set.

Remarks:
- Performance degradation is pronounced when the evaluation year is *after* the pretraining year [60].
- Larger models exhibit greater sensitivity to temporal misalignment than smaller models.
- Temporal misalignment complicates evaluation.
  - Older evaluations may underestimate newer models.
  - Newer evaluations may underestimate older models w.r.t. their original context.

Data age: Impact on evaluations

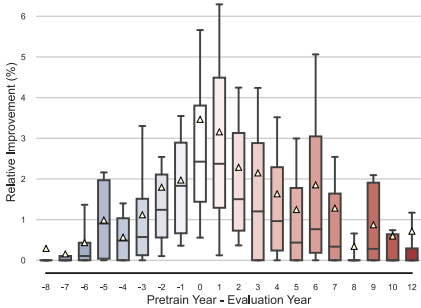


Figure: Asymmetry in data age. From [59].

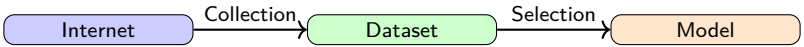
Remarks:

- Relative performance (y-axis) increases as temporal misalignment (x-axis) approaches zero.
- Performance degradation is asymmetric:
  - It is steeper when the evaluation year is after the pretraining year (red) than before (blue).
- Newer evaluations appear more difficult than old evaluations when applied to older models.
  - Older evaluations may underestimate the capabilities of newer models.

Data age: Mitigation techniques

- Trade-offs between using the most recent data (recency) and using data relevant to the task (relevance).
  - Data refreshment: Regularly update training data with newer information [33, 32].
  - Dynamic evaluation: Evaluate on datasets covering various time periods to assess robustness [105, 104].
  - Retrieval-augmented generation: Include recent documents in the context, e.g., web search results [30].

From datasets to algorithms



- We have seen how to construct a dataset from the Internet through collection, filtering, and processing.
- In the sequel, we will focus on how to efficiently feed this data to training algorithms.
- The process of transforming a raw dataset into this format involves several steps:
  - Tokenization: Converting text into numerical tokens in the vocabulary size.
  - Data masking: Methods to select most useful samples.
  - Document packing: Combining multiple documents for computational efficiency.
  - Data order: The order of samples during pretraining impacts learning dynamics.
  - Data mixing: Combining diverse data sources according to specific ratios.
  - Batching: Grouping sequences with a batch size B and a sequence length S.

Tokenization

- Tokenization is the process of decomposing a text into a sequence of individual "tokens" [103].
- Word-based tokenization: Simple, but suffers from out-of-vocabulary issues and large vocabulary size.
- Character-based tokenization: Leads to long sequences and fails to capture higher-level linguistic units.
- Subword tokenization: Solves the previous issues.

Example

The sentence "Tokenization is crucial." could be tokenized as:

- Word-based: ["Tokenization" "is" "crucial" "."]
- Character-based: ["T" "o" "k" "e" "n" "i" "z" "a" "t" "i" "o" "n" " " "i" "s" " " "c" "r" "u" "c" "i" "a" "l" "."]
- Subword-based: ["Token" "ization" "is" "crucial" "."]

Subword tokenization

- In subword tokenization, each word is split into a sequence of known subwords.

- Byte-pair encoding (BPE): Iteratively merges frequent pairs of consecutive tokens [27].
  - Starts with basic characters.
  - Selection criterion: Highest co-occurrence frequency.
  - Used in GPT-2, BART, LLaMA.
- WordPiece: Similar to BPE, but uses a likelihood-based selection criterion [19].
  - Scores pairs based on the impact on the training data likelihood.
  - Used in BERT.

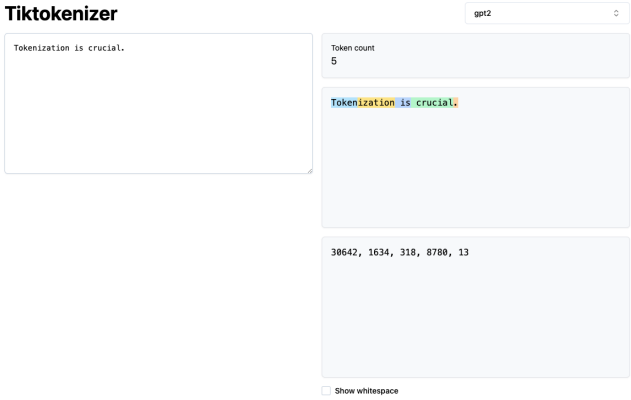


Figure: From Tiktokenizer.

Advanced tokenizers

- SentencePiece: Treats input as unique raw stream, including whitespaces in the set of characters to use [47].
  - Applicable to languages that do not use spaces to separate words (e.g., Chinese).
  - Also useful for languages where the meaning of a word strongly depends on the context.
- Commercial LLMs often use ad-hoc tokenizers.
  - GPT-4o uses the `o200k_base` tokenizer [62].
  - Faster token generation than previous GPTs.
  - Superior handling of Indic languages, with significant token reduction.

Feeding the tokenized data to an algorithm

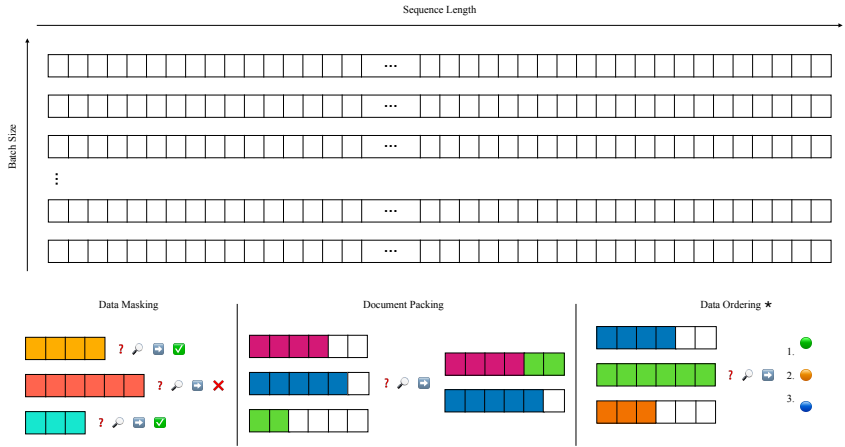


Figure: Overview of data selection techniques to efficiently feed data to models, including masking, packing, and ordering strategies to optimize learning efficiency. *Data Ordering can also be applied after Data Masking.

Motivation for data selection [64, 6]

- Why does data selection matter?
  - Costs: Training on large data is computationally expensive and produces significant carbon emissions
  - Quality over quantity: Smaller, high-quality datasets can yield better performance
  - Bias mitigation: Real-world datasets are biased; selection can help address imbalances
  - Robustness: Selection can reduce impact of corrupted examples and noisy data

- We will cover the following approaches in the sequel:
  - Data masking: Focus computation on informative tokens/samples.
  - Document packing: Optimize computational efficiency during training / reduce waste.
  - Data ordering: Structure training sequences for improved learning.

Data masking: Selection methods and results

- Perplexity correlation [82]: Samples with stable perplexity across open-source model checkpoints.
  - Perplexity is estimated in samples of 25 documents & webpages per domain.
  - Results in good performance on benchmark datasets: ARC Easy [16], PIQA [8], LAMBADA [65].
- Token-level selection (RHO-1) [54]: Filter tokens based on gradient importance scores.
  - A smaller "reference" model is used to obtain the importance scores (next slide).
  - Matches DeepSeekMath with 3% of tokens.
- Coreset methods [63]: Select small subsets that maintain gradient fidelity.
  - The subsets are updated during training.
  - 90% data reduction with higher accuracy.
- Importance resampling [91]: Re-weight examples based on similarity to a small target dataset distribution.
  - Prevents catastrophic forgetting compared to training on target dataset.
  - 5-15% gains on low-resource tasks.

Selective language modeling: Addressing token-level noise

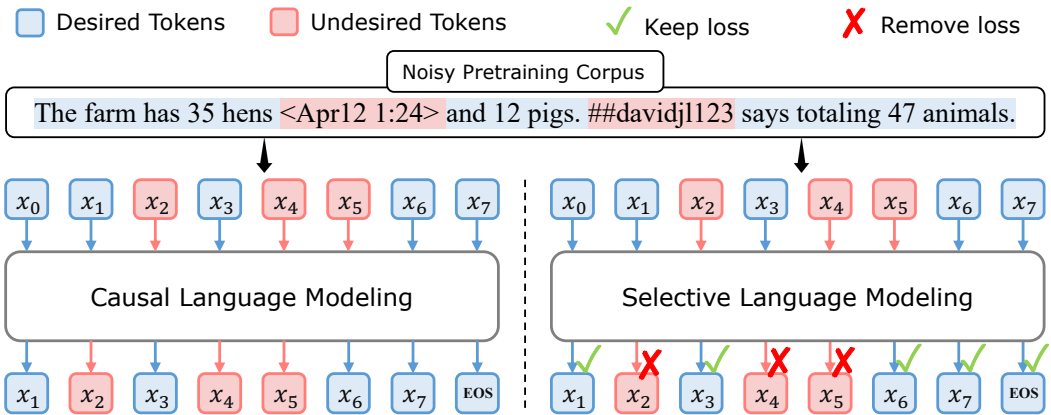


Figure: Selective Language Modeling approach. Even an extensively filtered pretraining corpus contains token-level noise. The proposed Selective Language Modeling (SLM) selectively applies loss on those useful and clean tokens. [54]

Document packing: Overview

- Training sequences often contain significant padding, wasting computation.
- Document packing combines multiple short sequences into single longer sequences to:
  - Reduce padding waste.
  - Increase computational efficiency.
  - Improve training throughput.
- Different approaches address this problem:
  - Standard packing [86]: Simple concatenation of sequences.
  - Cross-contamination prevention [46]: Mask attention to avoid leakage from independent sequences.
  - Optimized implementations [55]: Models like DeepSeek-V3 incorporate packing while preserving integrity.
- Document packing can yield 2-3$\times$ training speedup without sacrificing model quality.

Document packing: Implementation & results

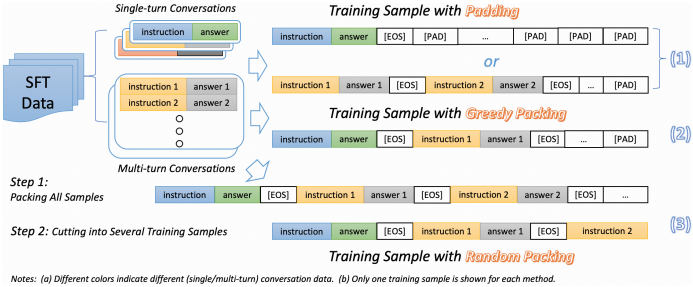


Figure: Packing multiple documents reduces padding tokens, improving training efficiency by 2-3× [86]. We may also giveup space to pack documents with semantic coherence, e.g., do not split instructions from their corresponding examples (Greedy Packing).

- Key considerations for effective document packing:
  - Maximum benefit when dataset contains many short documents.
  - More effective for larger models (benefits scale with parameter count).
  - Must balance efficiency gains against potential cross-document interference.

Importance of data order in LLM training

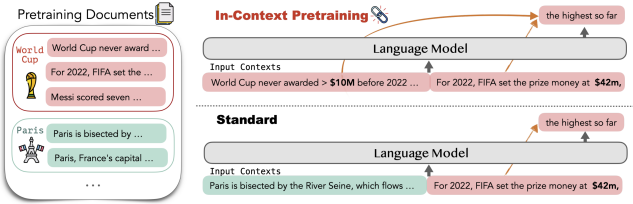


Figure: Unlike standard pretraining with shuffled documents, In-context pretraining groups related texts together, improving cross-document reasoning [74].

- The order of training data affects how language models learn patterns and dependencies [10, 85].
- Random shuffling disrupts inter-document relationships, limiting long-context reasoning [68].
- Keeping related documents together improves:
  - Information flow [38]
  - Coherence in conversations [101]
  - Memory retention [58]
  - Generalization [10]

Data order: Finding and structuring related documents

- Retrieving nearest neighbors
  - Each document d_i is mapped to an embedding $E(d_i)$ using a Contriever [39] model.
  - The similarity between documents is computed using cosine similarity:

$$s(d_i, d_j) = \cos(E(d_i), E(d_j)).$$

  - FAISS (Facebook AI Similarity Search) [43] is used for efficient approximate nearest-neighbor search.
  - Product quantization [41] and inverted file indexing [20] reduce memory usage and improve retrieval speed.

- Constructing a document graph
  - Documents are represented as nodes, edges exist between nearest neighbors.
  - Edge weights correspond to document similarity scores.
  - The graph structure ensures related documents appear close together in input sequences.

Data order: Optimizing document ordering for input contexts

- In graph traversal approach, we find a single path that maximizes similarity between consecutive documents.
- Modeled as the maximum traveling salesman problem (TSP):

$$P^* = \arg\max_P \sum_{(d_i, d_j) \in P} s(d_i, d_j).$$

- A greedy algorithm approximates an optimal path:
  - Start from the document with the fewest connections.
  - Iteratively select the most similar unvisited neighbor.
  - If no unvisited neighbors remain, connect to another low-degree document.
- Segmenting into Input Contexts:
  - The optimized document path is divided into fixed-size input contexts (e.g., 8192 tokens).
  - This ensures related documents are grouped without redundancy.

Data order: Experimental results of ICLM

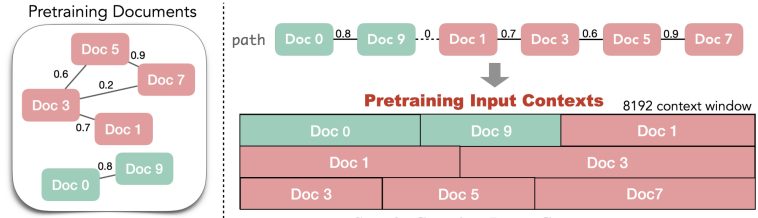


Figure: Overview of In-Context Pretraining: First, related documents are retrieved at scale to construct a document graph. Then, a traversal algorithm sequences documents to maximize contextual continuity, ensuring semantic coherence [74].

Remarks:

- Trained LLaMA-based models (0.3B-7B parameters) on 300B tokens from CommonCrawl.
- Performance improvements compared to standard pretraining:
  - Zero-shot generalization: +12%
  - Few-shot prompting accuracy: +10%
  - Faithfulness to prior context: +16%
  - Computational efficiency (faster convergence): -20% training steps

Data mixing problem

- Pretraining an LLM requires large-scale data from diverse sources, such as the Pile and SlimPajama [29, 73].

Data Source	Sampling (%)	Train Tokens	Validation Tokens	Train (% of Total)
CommonCrawl	67.0	200.82B	214.72M	66.99
C4	15.0	44.98B	48.06M	15.00
GitHub	4.5	13.51B	14.42M	4.51
Books	4.5	13.49B	14.39M	4.50
Wikipedia	4.5	13.50B	14.41M	4.50
ArXiv	2.5	7.49B	8.01M	2.50
StackExchange	2.0	5.99B	6.41M	2.00
Total	100.0	299.78B	320.42M	100.00

Table: SlimPajama dataset [73]

- Denote k domains for pretraining as $D_{\text{train}} = \{D_1, D_2, \ldots, D_k\}$.
- The sampling distribution is $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \in \triangle^k$, the probability simplex over k domains.
- How to find the optimal data mixture weight $\boldsymbol{\alpha}$?

Suboptimal heuristics data mixing methods

- Heuristics data mixing methods:
  - Uniform sampling across domains: $\alpha_i = \frac{1}{k}$
  - Sample-level uniform sampling: $\alpha_i = \frac{|D_i|}{|D_{\text{train}}|}$
  - Manual selection: Upweight high-quality sources like Wikipedia and academic texts [10, 29]
- Most current well-known LLMs still use manual selection strategies.
  - Staged training with various mixtures like Gemini [78, 79], Qwen [93, 94], Gemma [80, 81].
  - DeepSeek-V3 increases focus on math and programming samples compared to DeepSeek-V2 [18].
  - Qwen 2.5 uses Qwen2-Instruct models to classify and balance content across different domains [94].
- These heuristic methods may be suboptimal since they ignore:
  - Relevance of topics
  - Variance data quality
  - Toxicity, duplication etc.

Suboptimal heuristics data mixing methods

- Good domain mixing methods can greatly speed up LLMs training.

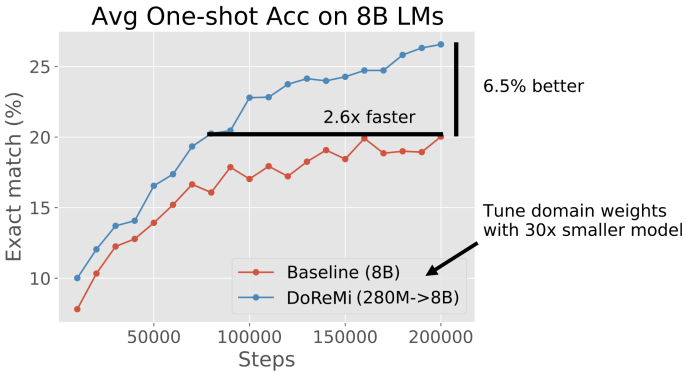


Figure: Comparison of domain mixing algorithm and heuristic method [90]

Proxy-based algorithms

- Objective: Train a model $\mathbf{x}$ minimizing average loss across all domains $\frac{1}{k}\sum_{i=1}^{k} f_i(\mathbf{x})$.
  - $f_i(\mathbf{x}) = \frac{1}{|D_i|}\sum_{\mathbf{a}\in D_i} \mathcal{L}(h_{\mathbf{x}}(\mathbf{a}_i, b_i))$
- Some works [90, 26] empirically show that a good $\boldsymbol{\alpha}$ can be transferred across model sizes.
- For efficiency, we can train small proxy models to obtain $\boldsymbol{\alpha}$ and then apply it to train a large model [90, 26].

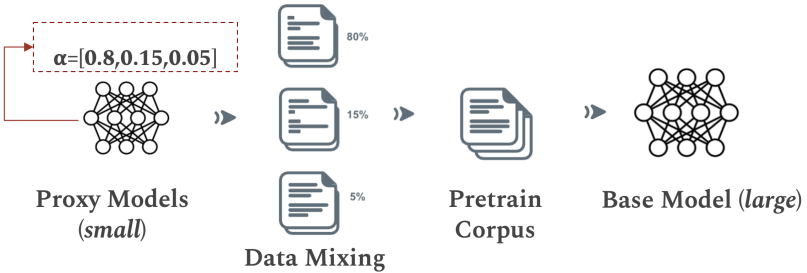


Figure: Proxy-based algorithms' pipeline [26].

Proxy-based algorithms: DoReMi [90]

- DoReMi is a three-step algorithms.
  1. Train a small reference model $\mathbf{x}_{\text{ref}}$ with uniform domain weights.
  2. Train proxy model with group distributionally robust optimization [69] to obtain domain weights.
     - Trains a robust model by optimizing the worst-case loss over domains:

$$\min_{\mathbf{x}} \max_{\boldsymbol{\alpha} \in \triangle^k} L(\mathbf{x}, \boldsymbol{\alpha}) := \sum_{i=1}^{k} \alpha_i \cdot [f_i(\mathbf{x}) - f_i(\mathbf{x}_{\text{ref}})]$$

     - The domain weights are guided by loss difference. At step t,

$$\alpha_i^t \propto \alpha_i^{t-1} \exp\left(f_i(\mathbf{x}^t) - f_i(\mathbf{x}_{\text{ref}})\right).$$

  3. Train large model with new domain weights.

Proxy-based algorithms: DoReMi [90]

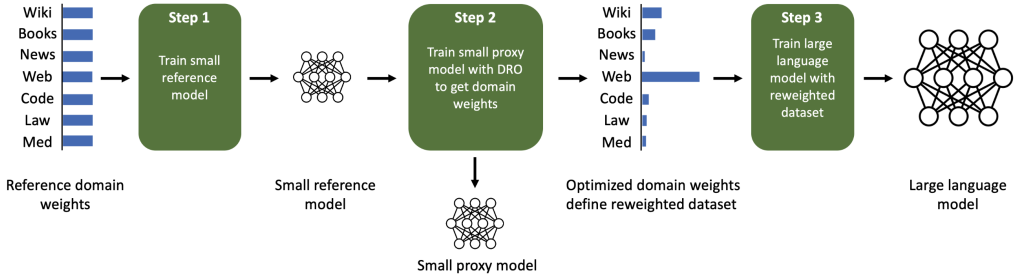


Figure: DoReMi's pipeline [90]

Remarks:

- DoReMi is not efficient enough since it requires two auxiliary models.
- α highly depends on the performance of reference model.

Proxy-based algorithms: DoGE [26]

- Compared with DoReMi, DoGE gets rid of training the reference model.
- DoGE is a two-step algorithm.

1. Train a small-scale proxy model to find optimal domain weights.
   - Formulate domain reweighting as a bilevel-optimization problem:

$$\boldsymbol{\alpha} \in \arg\min_{\boldsymbol{\alpha} \in \triangle^k} \sum_{i \in [k]} f_i(\mathbf{x}^\star(\boldsymbol{\alpha})) \quad \text{s.t.} \quad \mathbf{x}^\star(\boldsymbol{\alpha}) \in \arg\min_{\mathbf{x}} \sum_{i \in [k]} \alpha_i f_i(\mathbf{x})$$

   - Domain weights are updated by the first-order gradients in the following fashion:

$$\alpha_i^t \propto \alpha_i^{t-1} \exp\left(\langle \nabla f_i(\mathbf{x}^t), \sum_{j \in [k]} \nabla f_j(\mathbf{x}^t) \rangle \right)$$

2. Train large-scale base LLM with resampled pretrain corpus according to the finalized domain weights.

Remarks:
- The domain weight update uses approximations based on "influence functions."
- There are more principled ways of performing the updates! (reach out to me)

Proxy-based algorithms: DoGE [26]

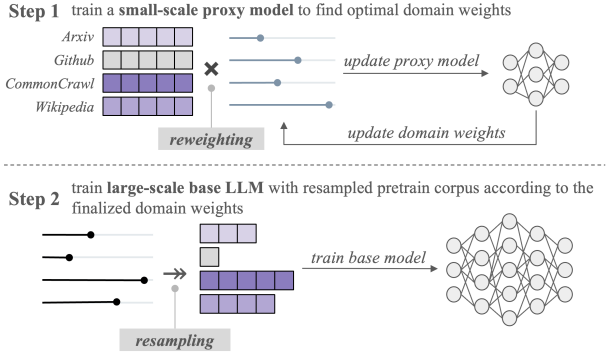


Figure: DoGE's pipeline [26]

Remarks:

- DoGE expands to out-of-domain scenarios.
- The proxy model requires k gradient computations per update, slowing training.

Proxy-based algorithms: CHAMELEON [92]

- DoReMi and DoGE derive domain weights from proxy model training process, which may be unstable [7, 42].
- CHAMELEON leverages domain characteristics directly to obtain domain weights.

1. Train a proxy model and extract domain embeddings $v_i \in \mathbb{R}^p$ for $i \in [k]$, where p is embedding dimension.
2. Define the kernel $\kappa(v_i, v_j) = v_i^\top v_j$ for domain similarity and construct the affinity matrix:

$$\Omega = [\kappa(v_i, v_j)]_{i,j=1}^k = VV^\top.$$

3. Compute Kernel Ridge Leverage Scores (KRLS) [5]:

$$S(D_i) = [\Omega(\Omega + kI)^{-1}]_{ii}.$$

4. Compute domain weights according to training phases.
   - Pretraining aims to obtain general knowledge: $\alpha_i = \text{softmax}\left(S^{-1}(D_i)\right)$
   - Finetuning specializes on specific tasks : $\alpha_i = \text{softmax}\left(S(D_i)\right)$

Proxy-based algorithms: CHAMELEON [92]

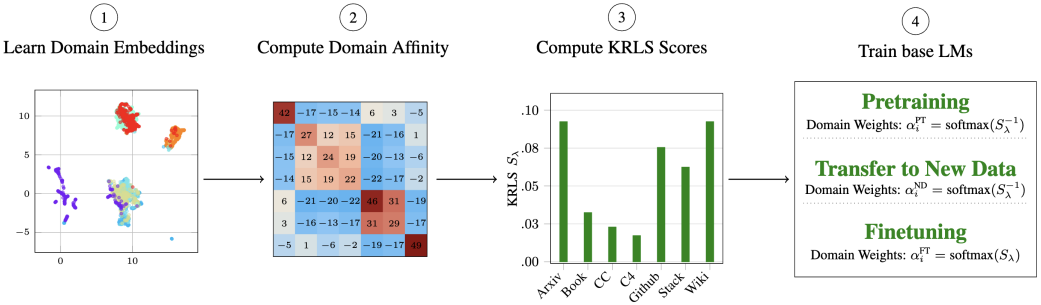


Figure: CHAMELEON's pipeline

Remarks:

- CHAMELEON is more stable since domain weights are independent of the training process.
- The time for step 1-3 in the above figure is negligible compared to proxy model training.
- It indicates that the training phases affect optimal domain weights.
- CHAMELEON can adapt to changes in domain composition without re-training proxy models.

Reliability of proxies

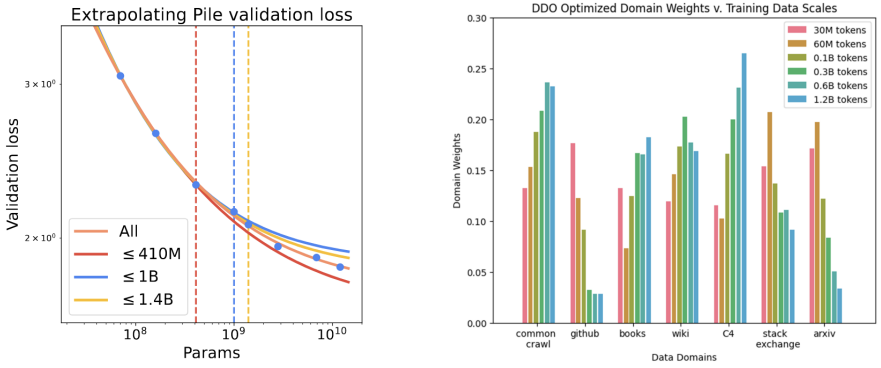


Figure: Optimal domain weights can depend on model and data size [42, 45]

- A proxy model may not faithfully mirror the target model's performance.
  - Scale with model sizes (left figure) [42, 97].
  - Scale with data size (right figure) [45].

Optimal domain weights are predictable

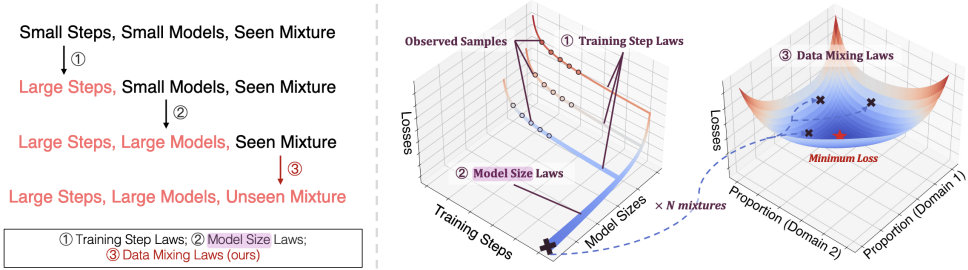


Figure: Data mixing law [97]

- [97, 45] find that the domain losses regarding the domain proportion precisely fit into an exponential law.
- [57] grid searches domain proportions and predicts optimal weights via regression.
- [45] investigates the scaling law of data size in the domain mixing problem.
- [97] predicts domain weights through using scaling laws of training steps, model sizes, and data mixing laws.

Online data mixing algorithms

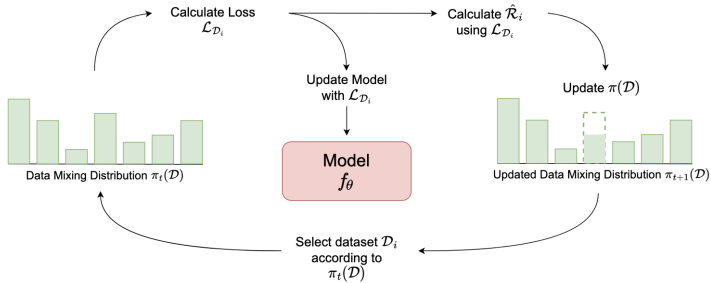


Figure: Online data mixing [7]

- Online algorithms offer an alternative to avoid proxy model pitfalls.
- [7, 42] propose loss-guided online data mixing methods.

Task-oriented methods

- Instead of averaging the loss across all domains, we can optionally use a specific task as the objective.
- Dynamic gradient alignment [25] is similar to DoGE but operates online with a target loss:

$$\alpha_i^t \propto \alpha_i^{t-1} \exp\left(\langle \nabla f_i(\mathbf{x}^t), \nabla f_{\text{target}}(\mathbf{x}^t) \rangle\right).$$

- Perplexity may not directly relate to downstream tasks' performance.
  - RegMix [57] finds that Common Crawl (CC) is highly relevant to downstream tasks' performance.
  - It uses the validation loss of Pile-CC [29] as objective.

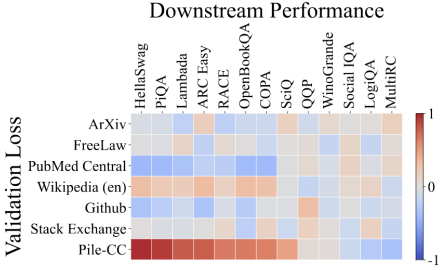


Figure: Correlation between validation loss by domains of the Pile and downstream performance.

From real to synthetic data: Data augmentation vs. Knowledge distillation

- Data augmentation [75]
  - Generates additional data samples from existing datasets.
  - Includes transformations, paraphrasing, or synthetic text generation.
  - Often aims to improve generalization in data-limited scenarios.
- Mathematically:

$$\mathcal{D}_{\text{aug}} = f(\mathcal{D}_{\text{orig}})$$

where f is a transformation function that increases dataset diversity.

- Knowledge distillation [37]
  - Transfers knowledge from a large teacher model (T) to a smaller student model (S).
  - Involves a loss function comparing student predictions with soft labels from the teacher.
  - Can use KL-divergence, mean squared error, or other objectives.
- Mathematically:

$$\mathcal{L}_{\text{KD}} = \text{KL}(\sigma(z_{\text{Teacher}}) \parallel \sigma(z_{\text{Student}}))$$

where σ is the softmax function, $z_{\text{Teacher}}, z_{\text{Student}}$ are teacher and student logits.

Remark: ○ One approach is to generate synthetic data from the teacher model for data augmentation.

Using teacher models for data generation and Phi models

- Instead of distillation, teacher models generate synthetic training data, improving diversity and generalization.
- We can use the teacher model to generate the next token

$$\mathcal{D}_{\text{synth}} = \{(\mathbf{a}_i, h_{\text{Teacher}}(\mathbf{a}_i)) | \mathbf{a}_i \sim h_{\text{Teacher}}\},$$

where $h_{\text{Teacher}}(\mathbf{a})$ is the teacher model.

- Phi Models: Learning from synthetic data
  - Phi-1.5 [52], Phi-2 [40], and Phi-3 [2] use synthetic data but do not rely on KL-divergence distillation.
  - These models leverage curated synthetic datasets (from GPT-4 variants etc.) to improve sample efficiency.
  - They demonstrate strong generalization abilities and few-shot learning performance.

- Training Objective:

$$\mathcal{L}_{\text{Phi}}(\mathbf{x}) = \sum_{(\mathbf{a},b)\in\mathcal{D}_{\text{synth}}} \mathcal{L}(h_{\mathbf{x}}(\mathbf{a}), h_{\text{Teacher}}(\mathbf{a})),$$

where $h_{\mathbf{x}}$ is the student model, and $\mathcal{L}$ is the supervised training loss applied over the synthetic dataset $\mathcal{D}_{\text{synth}}$.

Synthetic continued pretraining (Synthetic CPT) [96]

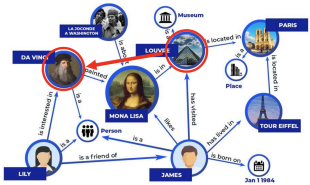


Figure: Illustration of Synthetic CPT acquiring new implicit fact as data using the EntiGraph method (The Louvre contains many works by DaVinci...) [96].

- LLMs struggle to acquire knowledge from small datasets efficiently.
- Synthetic CPT generates a large synthetic dataset from a small domain-specific dataset.
- Instead of training on a small dataset $\mathcal{D}_{\text{source}}$, Synthetic CPT expands it as follows:

$$A_{\text{synth}} : \mathcal{D}_{\text{source}} \rightarrow \mathcal{D}_{\text{synth}},$$

  - A_{synth} is an augmentation algorithm that diversifies knowledge representations.
- The resulting synthetic corpus helps the model generalize with various knowledge representations.

EntiGraph: Entity-centric augmentation in Synthetic CPT [96]

- This expansion is made using the EntiGraph method
  - Extracts key entities from $\mathcal{D}_{\text{source}}$:

$$\{E_1, E_2, ..., E_n\} \sim \text{LM}_{\text{extract}}(\mathcal{D}_{\text{source}})$$

    where $\text{LM}_{\text{extract}}$ is a pretrained model for entity identification.
  - For example, if $\mathcal{D}_{\text{source}}$ covers Linear Algebra, then $E_1 = \text{Linear Space}, E_2 = \text{Vector}, E_3 = \text{SVD}, \ldots$
  - Identifies relationships between entities:

$$\mathcal{D}_{E_i, E_j} \sim \text{LM}_{\text{relate}}(\mathcal{D}_{\text{source}}, E_i, E_j)$$

    where $\text{LM}_{\text{relate}}$ structures entity relationships for factual consistency.
  - Uses these structured relationships to generate diverse synthetic data, unlike simple paraphrasing.

Synthetic continued pretraining results [96]

- Balancing synthetic and real data
  - Synthetic data helps but has diminishing returns:

$$\mathcal{L}(N_{\text{synth}}) = A + B\log(N_{\text{synth}})$$

  - Synthetic CPT uses a balanced approach:

$$\min_{\theta} \sum_{(\mathbf{a},b)\in\mathcal{D}_{\text{synth}}} \mathcal{L}(h_{\mathbf{x}}(\mathbf{a}),b) + \lambda \sum_{(\mathbf{a},b)\in\mathcal{D}_{\text{real}}} \mathcal{L}(h_{\mathbf{x}}(\mathbf{a}),b)$$

where λ is the balance factor.

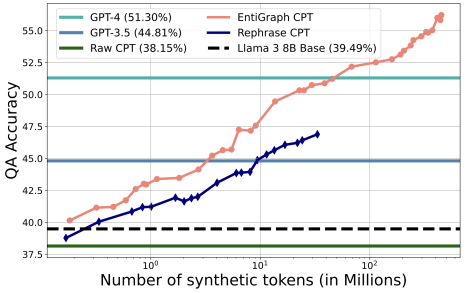


Figure: Impact of synthetic continued pretraining on QA accuracy. The x-axis shows the synthetic token count, and the y-axis shows model accuracy [96].

Knowledge distillation

- Knowledge distillation (KD) [37, 71] transfers knowledge from a large teacher model to a student model.

Knowledge distillation

Given a teacher model with logits z_{Teacher} and a student model with logits z_{Student}, distillation loss is given by

$$\mathcal{L}_{\text{KD}} = \alpha \text{CE}(b, z_{\text{Student}}) + (1 - \alpha) \cdot d(z_{\text{Teacher}}, z_{\text{Student}}),$$

where α balances supervision from labels of the distillation dataset and teacher outputs. $d(z_{\text{Teacher}}, z_{\text{Student}})$ measures the difference between the teacher and student:

- Cross-entropy: $d(z_{\text{Teacher}}, z_{\text{Student}}) = \text{CE}(z_{\text{Teacher}}, z_{\text{Student}})$
- KL divergence: $d(z_{\text{Teacher}}, z_{\text{Student}}) = \text{KL}(\sigma(z_{\text{Teacher}}) \parallel \sigma(z_{\text{Student}}))$, where $\sigma(\cdot)$ is the softmax.

Remarks:

- Reduces model size while maintaining accuracy.
- Transfers knowledge from a model to a more efficient one (e.g., linearized attention [99]).
- Speeds up inference using lightweight models.

MiniLLM: Reverse KL in knowledge distillation [31]

- Standard knowledge distillation minimizes the forward KL:

$$\mathcal{L}_{\text{KD}} = \text{KL}(\sigma(z_{\text{Teacher}}) \parallel \sigma(z_{\text{Student}})),$$

  which forces the student to match modes of the teacher, including low-likelihood outputs.

- Instead, MiniLLM minimizes the reverse KL:

$$\mathcal{L}_{\text{MiniLLM}} = \underset{\longleftarrow}{\text{KL}}(\sigma(z_{\text{Student}}) \parallel \sigma(z_{\text{Teacher}}))$$

- This leads to:
  - Mode-seeking behavior: The student prioritizes high-likelihood outputs.
  - Better generalization: Avoids copying uncertain, noisy predictions from the teacher.
  - More stable generation: Reduces exposure bias by penalizing improbable outputs.

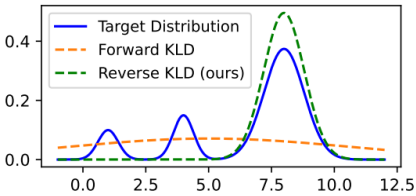


Figure: Fitting a Gaussian mixture with a single Gaussian using forward and reverse KLD [31].

MiniLLM: Optimization strategies for distillation

- Direct optimization of reverse KL is unstable; MiniLLM refines it with:
  - Uses Policy Gradient Optimization to compute gradients:

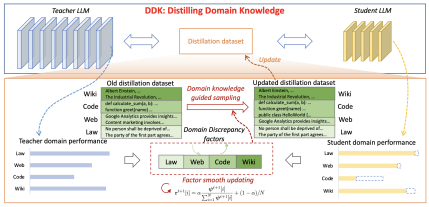
$$\nabla L = -\mathbb{E}_{\mathbf{a}\sim h_{\text{Teacher}}, b\sim h_{\text{Student}}(\mathbf{a})} \sum_t R_t \nabla \log h_{\text{Student}}(b_t|b_{<t}, \mathbf{a}),$$

    where R_t is a reward function based on the teacher's likelihood.
  - Reduces variance by computing per-token quality directly as a single-step decomposition.
  - Prevents degenerate text generation by mixing teacher's outputs.
  - Adjusts for sentence length bias to improve long-form generation using length normalization.

Remarks:

- Reverse KL distillation allows smaller models to focus on key knowledge.
- The optimization strategy stabilizes training and improves response quality.
- MiniLLM outperforms standard sequence-level KD across multiple datasets.

DDK: Distilling domain knowledge [56]



- Data mixing methods also benefit distillation.
- Standard KD treats all domains equally, ignoring performance gaps.
- Distilling domain knowledge (DDK) is an online data mixing algorithm for distillation [56].
  - The weight of domain i (α_i) is decided by the discrepancy between student and teacher models.
  - Domain discrepancy factor with b_i being the ground truth label for domain i:

$$\alpha_i = \frac{\exp(\text{CE}(h_{\text{Student}}(\mathbf{a}_i), b_i)/\text{CE}(h_{\text{Teacher}}(\mathbf{a}_i), b_i))}{\sum_{j=1}^{k} \exp(\text{CE}(h_{\text{Student}}(\mathbf{a}_j), b_j)/\text{CE}(h_{\text{Teacher}}(\mathbf{a}_j), b_j))}.$$

  - Use the updated domain weights to sample data from different domains to do distillation with $\mathcal{L}_{\text{KD}}$.

Distillation scaling laws [11]

Question:
- If I want a small, capable model, should I distill from a more powerful model, or train from scratch?

Remarks:
- Compared to supervised training:
  - It is only more efficient to distill if the teacher training cost does not matter.
  - Efficiency benefits vanish when enough compute/data is available.
  - Distillation cannot produce lower cross-entropies when enough compute/data is available.
- In practice the distillation is preferred as the large teacher model is often already available.

Reasoning data

- State-of-the-art LLMs have demonstrated impressive improvements in in mathematical reasoning.
  - GPT-o1 models, DeepSeek, etc.
- Specialized datasets are used for training LLMs with superior mathematical reasoning abilities.
- Pretraining, SFT, and RL approaches are all used for reasoning capabilities.

Reasoning data: DeepSeek-Math corpus

- DeepSeekMath corpus [72] construction:
  1. Initial seed corpus of high-quality mathematical text (OpenWebMath [66]).
  2. Train a classifier [44] using the seed corpus as positive samples and CC web pages as negative.
  3. The top-ranked web pages are added to the positive samples until 120B math tokens are obtained.
- The pretrained base model, DeepSeekMath-Base 7B, shows strong reasoning abilities [72].

Common reasoning benchmarks for LLMs

- Mathematical Reasoning Benchmarks
  - MATH [36]: Includes high school and competition-level problems.
  - GSM8K [17]: Grade school arithmetic word problems.
  - ProofWriter [77]: Tests theorem proving abilities.
- Sample Problem from MATH [36]:

$$\text{If } x + 2 = 5, \text{ solve for } x.$$

- Commonsense Reasoning Benchmarks
  - HellaSwag [98]: Tests physical commonsense understanding.
  - WinoGrande [70]: Pronoun resolution in ambiguous contexts.
  - PIQA [8]: Physical interaction question answering.
- Example Question from WinoGrande [70]:

  The trophy does not fit in the suitcase because it is too big/small. What does it refer to?

Common reasoning benchmarks for LLMs: MMLU and ARC

- MMLU: Massive Multitask Language Understanding [35]
  - Evaluates general knowledge across 57 subjects.
  - Covers humanities, STEM, social sciences, and other disciplines.
  - Used to assess generalization and robustness of LLMs.
- ARC: AI2 Reasoning Challenge [16]
  - Designed to assess advanced commonsense reasoning.
  - Includes science questions from elementary and middle school exams.
  - Requires multi-step inference and world knowledge.
- Example Question from ARC [16]:

  What is the main function of a leaf?

  A) *Store food*
  B) *Absorb water*
  C) *Produce oxygen*
  D) *Provide support*

Benchmark performance trends and prompting

- Performance Insights

  - Reasoning accuracy improves with model size, and training objectives such as SFT and RLHF [3].
  - Few-shot prompting significantly boosts performance on GSM8K and MATH [10].
  - Chain-of-thought (CoT) prompting improves multi-step logical reasoning [88].

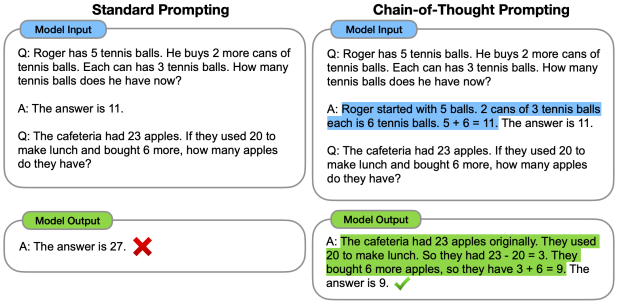


Figure: Standard vs chain-of-thought prompting [88].

Wrap up

- The data-mixture selection is an open topic (lots of conflicting conclusions!)
- Project discussions next Thursday!
- We will also have a brief introduction to the CSCS cluster

Supplementary Material

Deduplication techniques: Approximate matching

- NEARDUP: Efficiently identifies near-duplicates by estimating Jaccard similarity.
  - Represent the i-th document by its set of n-grams d_i.
  - Jaccard similarity:

$$J(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i \cup d_j|}$$

  - MinHash [9] approximates the Jaccard index using hash functions. Let $h_1, ..., h_n$ be a set of n hash functions. The MinHash signature $M(d)$ of a set d is:

$$M(d) = \Big[\min\{h_1(s) : s \in d\}, \min\{h_2(s) : s \in d\}, \ldots, \min\{h_n(s) : s \in d\}\Big].$$

  - *Key Property:* For any hash function h_i,

$$P\Big(\min\{h_i(d_i)\} = \min\{h_i(d_j)\}\Big) = J(d_i, d_j).$$

  - Modern deduplication methods like MinHashLSH [10] and SimHash [13] utilize locality sensitive hashing for more efficient near-duplicate detection.
- **Edit Similarity:** Similarity based on number of edits (insertions, deletions, substitutions) [51]

$$\text{EditSim}(x_i, x_j) = 1 - \frac{\text{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)}.$$

  - Other distances are used in practice (e.g., Levenshtein in PaLM [14]).
  - Documents are near-duplicates if Edit Similarity exceeds a threshold (e.g., 0.8).

Deduplication techniques: Model-Based

- Use learned representations from pretrained models to identify semantically similar documents.
- **SEMDEDUP [1]:**
  - Uses a pretrained language model to generate embeddings.
  - Calculates a representation for each data point (e.g., last layer embedding of final token).

$$e_i = \text{LM}(x_i)$$

  - Clusters data points to find groups of similar documents.

$$C = \text{Clustering}(\{e_i\})$$

  - Computes pairwise cosine similarity within each cluster.
  - Can be combined with other methods (e.g., MinHash) for efficiency.
- **Document De-Duplication and Diversification (D4) [83]:**
  - Combines MinHash, SemDeDup, and clustering.
  - Removes "most prototypical" examples within each cluster using [76], enhancing diversity.

Quality filtering: Classifier-based

Definition (Quality Score)

Let $q : \mathcal{D} \to [0, 1]$ be a quality classifier, where $\mathcal{D}$ is the space of all documents. $q(d)$ represents the predicted quality of document d, with higher values indicating higher quality. A threshold is used to filter:

$$\mathcal{D}_{\text{filtered}} = \{d \in \mathcal{D} | q(d) \geq \tau\}. \tag{1}$$

Theorem (Stochastic Selection (Pareto Sampling))

A common approach [10] is to keep a document d if:

$$r > 1 - q(d), \tag{2}$$

where $r \sim$ Pareto(α) is a random variable drawn from a Pareto distribution with shape parameter α, and $q(d)$ is the document's quality score.

- Pareto distribution introduces stochasticity, allowing some lower-scoring documents to be included.
- The shape parameter α controls the degree of stochasticity.

Data age: Temporal degradation

- Performance degradation is quantified using Temporal Degradation (TD).
- Average rate of performance deterioration per unit time delta between training and evaluation data. Higher TD indicates greater sensitivity to temporal shifts.

Definition: Temporal Degradation (TD) [60]

Let $S_{t,t'}$ be the performance of a model trained on data from time t and evaluated on data from time t'. Let $D(t \to t') = -(S_{t,t'} - S_{t',t'}) \cdot \text{sign}(t - t')$, which indicates the performance delta between the time-aligned and misaligned models. The TD score at evaluation time t' for training times $\mathcal{T}$ is:

$$\text{TD}(\mathcal{T} \to t') = \left| \frac{\sum_{t \in \mathcal{T}} (D(t \to t') - \bar{D})(t - \bar{t})}{\sum_{t \in \mathcal{T}} (t - \bar{t})^2} \right|,$$

where $\bar{t} = \text{avg}_{t \in \mathcal{T}} t$ and $\bar{D} = \text{avg}_{t \in \mathcal{T}} D(t \to t')$.

Data filtering details

Model	Filtering Details
Bert	"ignore lists, tables, and headers"
GPT-2	removed Wikipedia
RoBerta	CC filtered to news and Winograd-like subsets
XLNet	"heuristics to aggressively filter out short or low-quality articles"
T5	Heuristic quality, toxicity, and length filters; code removed
GPT-3	Filtered based on similarity to high-quality reference corpora.
GPT-J/Neo	Uses fasttext classifier on Pile-CC, with OpenWebText2 as the high-quality reference.
GLaM	Classifier with Wikipedia, books and selected websites as positive examples
LaMDA	"LaMDA SSI and safety discriminators are also used to score and filter 2.5M turns of dialog data sampled from the pre-training dataset", which are then trained on.
AlphaCode	Filtering heuristics to exclude automatically generated code
CodeGen	Heuristic filters for code quality
Chinchilla	Heuristic-based quality filtering, SafeSearch filter
Minerva	Same as PaLM for non-academic data
BLOOM	heuristic-based quality and porn filtering
PaLM	Same as GlaM
Galactica	Apply several quality filters: exclude papers from journals with certain keywords or low journal impact factor
LLaMA	Classifier to filter out low-quality and un-Wikipedia-like text

Figure: Additional notes on each model's filtering details from [59].

LoLCATs: Low-Rank Linear Conversion via Attention Transfer

- Traditional output-based distillation hardly scales to very large models.
- A recent work called LoLCATs [99] replaces quadratic softmax attention with linearized attention.
- Utilizes attention distillation that acts on the hidden layers, and low-rank adaptation.

Mathematical Formulation: Attention Transfer

The original Transformer attention function:

$$y_n = \sum_{i=1}^{n} \frac{\exp(q_n^\top k_i / \sqrt{d})}{\sum_{j=1}^{n} \exp(q_n^\top k_j / \sqrt{d})} v_i$$

is approximated in LoLCATs using a linear feature transformation ϕ:

$$\hat{y}_n = \sum_{i=1}^{n} \frac{\phi(q_n)^\top \phi(k_i)}{\sum_{j=1}^{n} \phi(q_n)^\top \phi(k_j)} v_i$$

The function ϕ (a learnable linear layer) is trained via distillation loss for each layer m and attention head h:

$$\ell_{\text{MSE}} = \frac{1}{MH} \sum_{m=1}^{M} \sum_{h=1}^{H} \frac{1}{d} \sum_{n=1}^{d} (y_n - \hat{y}_n)^2$$

Low-Rank Fine-Tuning in LoLCATs

Step 2: Low-Rank Adaptation (LoRA)

After replacing softmax attention with linear attention, LoLCATs fine-tunes only the projection matrices using Low-Rank Adaptation (LoRA):

$$W' = W + \Delta W, \quad \text{where} \quad \Delta W = BA$$

where:

- W' is the updated projection matrix,
- $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$ are low-rank matrices (with $r \ll d$).

Training Objective

In the LoRA step, LoLCATs minimizes the cross-entropy loss for next-token prediction:

$$\ell_{\text{xent}} = -\sum \log P_{\Theta}(u_{t+1}|u_{1:t})$$

where P_{Θ} is the modified LLM with linearized attention.

LoLCATs Framework General Picture

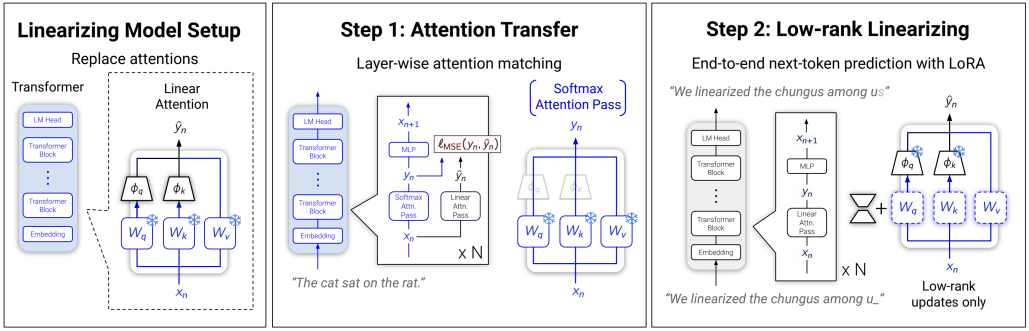


Figure: Illustration of the LoLCATs framework. [99].

- Scalability: Enables the linearization of 70B, up to 405B models.
- Efficiency: It takes under half the total GPU hours than prior methods linearize 8B models.

Distillation Scaling Laws: Formulation [11]

- The student cross-entropy follows a power-law relation:

$$L_S(N_S, D_S, L_T) = L_T + \frac{1}{L_T^c}\left(1 + \left(\frac{L_T}{\tilde{L}_S}\right)^{1/f}\right)^{-cf}\left(\frac{A}{N_S^\alpha} + \frac{B}{D_S^\beta}\right)^\gamma$$

where:

- L_T and $\tilde{L}_S$: teacher and student cross-entropy.
- N_S, D_S: student model and data size.
- A, B, c, f: scaling and transition factors.
- α, β, γ: model, data, and loss scaling.

Remarks:

- If N_S or D_S is too small, distillation significantly outperforms direct supervision.
- For large N_S and D_S direct training is preferable rather than distillation.
- There exists a transition where supervised learning overtakes distillation as compute scales.
- Optimal teacher selection depends on L_T: too strong teacher may not yield further benefits.

References I

[1] Amro Abbas, Colin Parkinson, Yuqing Dai, Bahram Rafiee, Somayeh Jafarian, and James Zou.
Semdedup: Data-efficient learning at web-scale through semantic deduplication.
arXiv preprint arXiv:2303.09540, 2023.
(Cited on pages 9 and 65.)

[2] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al.
Phi-3 technical report: A highly capable language model locally on your phone.
arXiv preprint arXiv:2404.14219, 2024.
(Cited on page 48.)

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al.
Gpt-4 technical report.
arXiv preprint arXiv:2303.08774, 2023.
(Cited on page 61.)

[4] Oshin Agarwal and Ani Nenkova.
Temporal effects on pre-trained models for language processing tasks, 2022.
(Cited on page 16.)

References II

[5] Ahmed El Alaoui and Michael W. Mahoney.
Fast Randomized Kernel Methods With Statistical Guarantees, 2015.
(Cited on page 41.)

[6] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang.
A survey on data selection for language models, 2024.
(Cited on page 24.)

[7] Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang.
Efficient online data mixing for language model pre-training.
In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models Workshop*, 2023.
(Cited on pages 41 and 45.)

[8] Yonatan Bisk, Rowan Zellers, Rowan Le Bras, Jianfeng Gao, and Yejin Choi.
Piqa: Reasoning about physical commonsense in natural language.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.
(Cited on pages 25 and 59.)

References III

[9] Andrei Z Broder.
On the resemblance and containment of documents.
In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.
(Cited on page 64.)

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.
Language models are few-shot learners.
Advances in neural information processing systems, 33:1877–1901, 2020.
(Cited on pages 14, 29, 34, 61, 64, and 66.)

[11] Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb.
Distillation scaling laws, 2025.
(Cited on pages 56 and 72.)

[12] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer.
Membership inference attacks from first principles, 2022.
(Cited on page 10.)

References IV

[13] Moses S Charikar.
Similarity estimation techniques from rounding algorithms.
In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.
(Cited on page 64.)

[14] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al.
Palm: Scaling language modeling with pathways.
Journal of Machine Learning Research, 24(240):1–113, 2023.
(Cited on pages 8, 14, and 64.)

[15] Jaymari Chua, Yun Li, Shiyi Yang, Chen Wang, and Lina Yao.
Ai safety in generative ai large language models: A survey.
arXiv preprint arXiv:2407.18369, 2024.
(Cited on page 11.)

[16] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord.
Think you have solved question answering? try arc, the ai2 reasoning challenge.
arXiv preprint arXiv:1803.05457, 2018.
(Cited on pages 25 and 60.)

References V

[17] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al.
Training verifiers to solve math word problems.
arXiv preprint arXiv:2110.14168, 2021.
(Cited on page 59.)

[18] DeepSeek-AI.
Deepseek-v3 technical report, 2024.
(Cited on page 34.)

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
(Cited on page 21.)

[20] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou.
The faiss library.
arXiv preprint arXiv:2401.08281, 2024.
(Cited on page 30.)

References VI

[21] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al.
Glam: Efficient scaling of language models with mixture-of-experts.
In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.
(Cited on page 14.)

[22] Cynthia Dwork.
Differential privacy: A survey of results.
In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.
(Cited on page 10.)

[23] Cynthia Dwork.
Differential Privacy, pages 338–340.
Springer US, Boston, MA, 2011.
(Cited on page 10.)

References VII

[24] Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hannaneh Hajishirzi, Noah A. Smith, and Jesse Dodge.
What's in my big data?
In *The Twelfth International Conference on Learning Representations*, 2024.
(Cited on page 7.)

[25] Simin Fan, David Grangier, and Pierre Ablin.
Dynamic gradient alignment for online data mixing.
arXiv preprint arXiv:2410.02498, 2024.
(Cited on page 46.)

[26] Simin Fan, Matteo Pagliardini, and Martin Jaggi.
DOGE: Domain reweighting with generalization estimation.
2024.
(Cited on pages 36, 39, and 40.)

[27] Philip Gage.
A new algorithm for data compression.
The C Users Journal, 12(2):23–38, 1994.
(Cited on page 21.)

References VIII

[28] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed.
Bias and fairness in large language models: A survey.
Computational Linguistics, 50(3):1097–1179, 09 2024.
(Cited on page 11.)

[29] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al.
The pile: An 800gb dataset of diverse text for language modeling.
arXiv preprint arXiv:2101.00027, 2020.
(Cited on pages 33, 34, and 46.)

[30] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang.
Retrieval-augmented generation for large language models: A survey, 2024.
(Cited on page 18.)

[31] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang.
Minillm: Knowledge distillation of large language models, 2024.
(Cited on page 53.)

References IX

[32] Yiduo Guo, Jie Fu, Huishuai Zhang, Dongyan Zhao, and Yikang Shen.
Efficient continual pre-training by mitigating the stability gap, 2024.
(Cited on page 18.)

[33] Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort.
Continual pre-training of large language models: How to (re)warm your model?, 2023.
(Cited on page 18.)

[34] Jacqueline He, Mengzhou Xia, Christiane Fellbaum, and Danqi Chen.
Mabel: Attenuating gender bias using textual entailment data.
arXiv preprint arXiv:2210.14975, 2022.
(Cited on page 11.)

[35] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.
Measuring massive multitask language understanding.
arXiv preprint arXiv:2009.03300, 2020.
(Cited on page 60.)

References X

[36] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt.
Measuring mathematical problem solving with the math dataset.
arXiv preprint arXiv:2103.03874, 2021.
(Cited on page 59.)

[37] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean.
Distilling the knowledge in a neural network.
arXiv preprint arXiv:1503.02531, 2015.
(Cited on pages 47 and 52.)

[38] Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.
(Cited on page 29.)

[39] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave.
Unsupervised dense information retrieval with contrastive learning.
Transactions on Machine Learning Research.
(Cited on page 30.)

References XI

[40] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al.
Phi-2: The surprising power of small language models.
Microsoft Research Blog, 1(3):3, 2023.
(Cited on page 48.)

[41] Herve Jegou, Matthijs Douze, and Cordelia Schmid.
Product quantization for nearest neighbor search.
IEEE transactions on pattern analysis and machine intelligence, 33(1):117–128, 2010.
(Cited on page 30.)

[42] Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J Zico Kolter.
Adaptive data optimization: Dynamic sample selection with scaling laws.
arXiv preprint arXiv:2410.11820, 2024.
(Cited on pages 41, 43, and 45.)

[43] Jeff Johnson, Matthijs Douze, and Hervé Jégou.
Billion-scale similarity search with gpus.
IEEE Transactions on Big Data, 7(3):535–547, 2019.
(Cited on page 30.)

References XII

[44] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov.
Fasttext. zip: Compressing text classification models.
arXiv preprint arXiv:1612.03651, 2016.
(Cited on page 58.)

[45] Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia.
Autoscale: Automatic prediction of compute-optimal data composition for training llms.
arXiv preprint arXiv:2407.20177, 2024.
(Cited on pages 43 and 44.)

[46] Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon.
Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance, 2022.
(Cited on page 27.)

[47] Taku Kudo and John Richardson.
SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.
In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.
(Cited on page 22.)

References XIII

[48] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al.
The bigscience roots corpus: A 1.6 tb composite multilingual dataset.
Advances in Neural Information Processing Systems, 35:31809–31826, 2022.
(Cited on page 13.)

[49] Anne Lauscher, Tobias Lueken, and Goran Glavavs.
Sustainable modular debiasing of language models.
arXiv preprint arXiv:2109.03646, 2021.
(Cited on page 11.)

[50] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom.
Mind the gap: Assessing temporal generalization in neural language models, 2021.
(Cited on page 16.)

[51] Katherine Lee, Mert Yuksekgonul, Aditi Raghunathan, Jacob Steinhardt, and James Zou.
Deduplicating training data makes language models better.
arXiv preprint arXiv:2107.06499, 2021.
(Cited on pages 6, 7, 8, and 64.)

References XIV

[52] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee.
Textbooks are all you need ii: phi-1.5 technical report.
arXiv preprint arXiv:2309.05463, 2023.
(Cited on page 48.)

[53] Zhan Li, Yongtao Wu, Yihang Chen, Francesco Tonin, Elias Abad Rocamora, and Volkan Cevher.
Membership inference attacks against large vision-language models, 2024.
(Cited on page 10.)

[54] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al.
Rho-1: Not all tokens are what you need.
arXiv preprint arXiv:2404.07965, 2024.
(Cited on pages 25 and 26.)

[55] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al.
Deepseek-v3 technical report.
arXiv preprint arXiv:2412.19437, 2024.
(Cited on page 27.)

References XV

[56] Jiaheng Liu, Chenchen Zhang, Jinyang Guo, Yuanxing Zhang, Haoran Que, Ken Deng, Jie Liu, Ge Zhang, Yanan Wu, Congnan Liu, et al.
Ddk: Distilling domain knowledge for efficient large language models.
Advances in Neural Information Processing Systems, 37:98297–98319, 2025.
(Cited on page 55.)

[57] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin.
Regmix: Data mixture as regression for language model pre-training.
arXiv preprint arXiv:2407.01492, 2024.
(Cited on pages 44 and 46.)

[58] Xiaodong Liu, Jie Yuan, Shujie Fu, Min Jiang, Hiroaki Hayashi, Graham Neubig, Hao Cheng, Wanjun Lv, Linjun Song, Yichong Xu, et al.
Gpt understands, too.
arXiv preprint arXiv:2103.10385, 2021.
(Cited on page 29.)

References XVI

[59] Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito.
A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, and toxicity, 2023.
(Cited on pages 15, 16, 17, and 68.)

[60] Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith.
Time waits for no one! analysis and challenges of temporal misalignment, 2022.
(Cited on pages 16 and 67.)

[61] Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy.
An empirical survey of the effectiveness of debiasing techniques for pre-trained language models.
arXiv preprint arXiv:2110.08527, 2021.
(Cited on page 11.)

[62] Microsoft.
Exploring the new frontier of ai: Openai's gpt-4 o for indic languages, 2024.
Accessed: 2025-03-05.
(Cited on page 22.)

References XVII

[63] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec.
Coresets for data-efficient training of machine learning models.
In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.
(Cited on page 25.)

[64] Baharan Mirzasoleiman and Siddharth Joshi.
Foundations of data-efficient learning.
ICML 2024 Tutorial, 2024.
Presented at ICML 2024.
(Cited on page 24.)

[65] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez.
The LAMBADA dataset: Word prediction requiring a broad discourse context.
In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics.
(Cited on page 25.)

[66] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba.
Openwebmath: An open dataset of high-quality mathematical web text.
arXiv preprint arXiv:2310.06786, 2023.
(Cited on page 58.)

References XVIII

[67] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al.
Scaling language models: Methods, analysis & insights from training gopher.
arXiv preprint arXiv:2112.11446, 2021.
(Cited on page 13.)

[68] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu.
Exploring the limits of transfer learning with a unified text-to-text transformer.
Journal of machine learning research, 21(140):1–67, 2020.
(Cited on pages 13 and 29.)

[69] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang.
Distributionally robust neural networks.
2020.
(Cited on page 37.)

[70] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi.
Winogrande: An adversarial winograd schema challenge at scale.
Communications of the ACM, 64(9):99–106, 2021.
(Cited on page 59.)

References XIX

[71] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf.
Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.
arXiv preprint arXiv:1910.01108, 2019.
(Cited on page 52.)

[72] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al.
Deepseekmath: Pushing the limits of mathematical reasoning in open language models.
arXiv preprint arXiv:2402.03300, 2024.
(Cited on page 58.)

[73] Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, et al.
Slimpajama-dc: Understanding data combinations for llm training.
arXiv preprint arXiv:2309.10818, 2023.
(Cited on page 33.)

[74] Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Gergely Szilvasy, Rich James, Xi Victoria Lin, Noah A Smith, Luke Zettlemoyer, et al.
In-context pretraining: Language modeling beyond document boundaries.
arXiv preprint arXiv:2310.10638, 2023.
(Cited on pages 29 and 32.)

References XX

[75] Connor Shorten and Taghi M Khoshgoftaar.
A survey on image data augmentation for deep learning.
Journal of big data, 6(1):1–48, 2019.
(Cited on page 47.)

[76] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos.
Beyond neural scaling laws: beating power law scaling via data pruning.
Advances in Neural Information Processing Systems, 35:19523–19536, 2022.
(Cited on pages 9 and 65.)

[77] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark.
Proofwriter: Generating implications, proofs, and abductive statements over natural language.
arXiv preprint arXiv:2012.13048, 2020.
(Cited on page 59.)

[78] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al.
Gemini: a family of highly capable multimodal models.
arXiv preprint arXiv:2312.11805, 2023.
(Cited on page 34.)

References XXI

[79] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al.
Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.
arXiv preprint arXiv:2403.05530, 2024.
(Cited on page 34.)

[80] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al.
Gemma: Open models based on gemini research and technology.
arXiv preprint arXiv:2403.08295, 2024.
(Cited on page 34.)

[81] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al.
Gemma 2: Improving open language models at a practical size.
arXiv preprint arXiv:2408.00118, 2024.
(Cited on page 34.)

[82] Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto.
Improving pretraining data using perplexity correlations.
In *The Thirteenth International Conference on Learning Representations*, 2025.
(Cited on page 25.)

References XXII

[83] Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos.
D4: Improving llm pretraining via document de-duplication and diversification.
Advances in Neural Information Processing Systems, 36, 2024.
(Cited on pages 9 and 65.)

[84] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample.
Llama: Open and efficient foundation language models, 2023.
(Cited on page 14.)

[85] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008, 2017.
(Cited on page 29.)

[86] Shuhe Wang, Guoyin Wang, Yizhong Wang, Jiwei Li, Eduard Hovy, and Chen Guo.
Packing analysis: Packing is more appropriate for large models or datasets in supervised fine-tuning.
arXiv preprint arXiv:2410.08081, 2024.
(Cited on pages 27 and 28.)

References XXIII

[87] Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, Ed Chi, and Slav Petrov.
Measuring and reducing gendered correlations in pre-trained models.
arXiv preprint arXiv:2010.06032, 2020.
(Cited on page 11.)

[88] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al.
Chain-of-thought prompting elicits reasoning in large language models.
Advances in neural information processing systems, 35:24824–24837, 2022.
(Cited on page 61.)

[89] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave.
Ccnet: Extracting high quality monolingual datasets from web crawl data.
In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4003–4012, 2020.
(Cited on page 7.)

References XXIV

[90] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu.
DoReMi: Optimizing data mixtures speeds up language model pretraining.
2023.
(Cited on pages 35, 36, 37, and 38.)

[91] Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang.
Data selection for language models via importance resampling.
In *Advances in Neural Information Processing Systems*, 2023.
(Cited on page 25.)

[92] Wanyun Xie, Francesco Tonin, and Volkan Cevher.
Chameleon: A flexible data-mixing framework for language model pretraining and finetuning.
Under review, 2024.
(Cited on pages 41 and 42.)

References XXV

[93] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan.
Qwen2 technical report.
arXiv preprint arXiv:2407.10671, 2024.
(Cited on page 34.)

[94] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu.
Qwen2.5 technical report.
arXiv preprint arXiv:2412.15115, 2024.
(Cited on page 34.)

References XXVI

[95] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le.
XInet: Generalized autoregressive pretraining for language understanding, 2020.
(Cited on page 13.)

[96] Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto.
Synthetic continued pretraining.
arXiv preprint arXiv:2409.07431, 2024.
(Cited on pages 49, 50, and 51.)

[97] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu.
Data mixing laws: Optimizing data mixtures by predicting language modeling performance.
arXiv preprint arXiv:2403.16952, 2024.
(Cited on pages 43 and 44.)

[98] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi.
Hellaswag: Can a machine really finish your sentence?
arXiv preprint arXiv:1905.07830, 2019.
(Cited on page 59.)

References XXVII

[99] Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré.
Lolcats: On low-rank linearizing of large language models.
arXiv preprint arXiv:2410.10254, 2024.
(Cited on pages 52, 69, and 71.)

[100] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al.
Opt: Open pre-trained transformer language models.
arXiv preprint arXiv:2205.01068, 2022.
(Cited on page 13.)

[101] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan Liu.
Dialogpt: Large-scale generative pre-training for conversational response generation.
arXiv preprint arXiv:1911.00536, 2020.
(Cited on page 29.)

[102] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang.
Gender bias in coreference resolution: Evaluation and debiasing methods.
arXiv preprint arXiv:1804.06876, 2018.
(Cited on page 11.)

References XXVIII

[103] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al.
A survey of large language models.
arXiv preprint arXiv:2303.18223, 2023.
(Cited on page 20.)

[104] Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang.
Is your llm outdated? evaluating llms at temporal generalization, 2024.
(Cited on page 18.)

[105] Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie.
Dyval: Dynamic evaluation of large language models for reasoning tasks.
In *The Twelfth International Conference on Learning Representations*, 2024.
(Cited on page 18.)