LLM Data Pipeline

Anisio Lacerda
Rodrygo Santos

Data fundamentally limits model performance

> A key principle in AI is that **the quality of a model is fundamentally limited by the quality of its training data.**

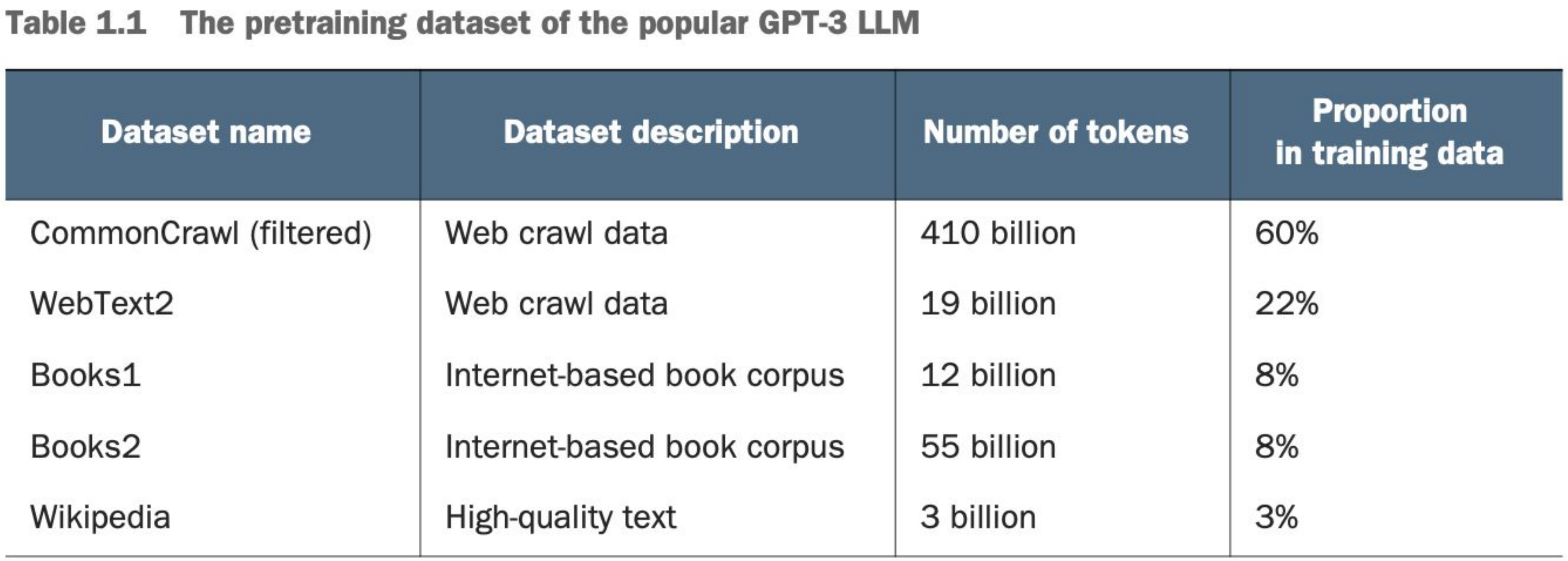


Table 1.1 The pretraining dataset of the popular GPT-3 LLM

Dataset name	Dataset description	Number of tokens	Proportion in training data
CommonCrawl (filtered)	Web crawl data	410 billion	60%
WebText2	Web crawl data	19 billion	22%
Books1	Internet-based book corpus	12 billion	8%
Books2	Internet-based book corpus	55 billion	8%
Wikipedia	High-quality text	3 billion	3%

Data fundamentally limits model performance

> A key principle in AI is that **the quality of a model is fundamentally limited by the quality of its training data.**

Data-centric AI

E.g.: OpenAI significantly increased its data efforts from **GPT-3** (**2** people credited) to **GPT-4** (**80** people, not including external annotators), even dedicating **senior researchers** to seemingly **simple formatting tasks**.

Model-centric AI

Improving models through new architectures, increased sizes, or training techniques.

The Importance of Data Quality Over Quantity (alone)

While Large Language Models (LLMs) are still pre-trained on "vast amounts of text"

E.g., Llama 3 on 16 trillion tokens.

Yi model family, found that:

"10K carefully crafted instructions are superior to hundreds of thousands of noisy instructions".

The LIMA (Less Is More for Alignment) paper demonstrated that a 65B-parameter Llama model

fine-tuned with "1,000 carefully curated prompts and responses"

produce **answers** preferred to GPT-4 in **43%** of cases **by human annotators**

The Importance of Data Quality Over Quantity (alone)

Llama 3 team arrived at the same conclusion

Llama 3's performance gains were "primarily driven by **improvements** in **data quality** and diversity as well as by **increased training scale**"

"**human-generated data** is more **prone to errors** and **inconsistencies**, particularly for nuanced safety policies"

"develop **AI-assisted annotation tools** to ensure **high data quality**"

> A small amount of high-quality data can outperform a large amount of noisy data, e.g., data that is irrelevant or inconsistent.

The Importance of Data Diversity

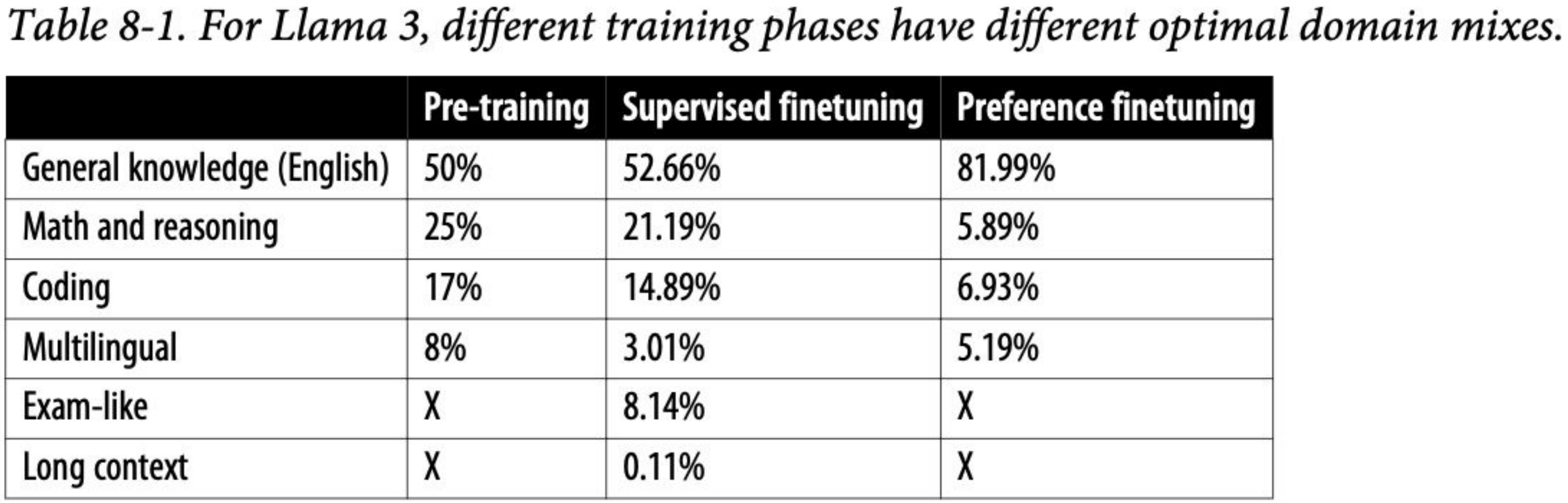


Table 8-1. For Llama 3, different training phases have different optimal domain mixes.

	Pre-training	Supervised finetuning	Preference finetuning
General knowledge (English)	50%	52.66%	81.99%
Math and reasoning	25%	21.19%	5.89%
Coding	17%	14.89%	6.93%
Multilingual	8%	3.01%	5.19%
Exam-like	X	8.14%	X
Long context	X	0.11%	X

> A small amount of high-quality data can outperform a large amount of noisy data, e.g., data that is irrelevant or inconsistent.

Roadmap

To master the art of creating high-impact datasets

Part I: The Atoms of Language - Text Representation

Fine-grained mechanics of converting text into structured numerical tensors.

Part II: Data Augmentation and Synthesis

We will now address how to get more data. We'll differentiate between data **augmentation** (creating new data from existing real data) and data **synthesis** (generating new data from scratch).

Part III: The Research Frontier

Model collapse—the risk of models degrading when recursively trained on their own output—and the importance of **mixing synthetic data with real human data** to maintain quality and diversity.

Part I: The Atoms of Language - Text Representation

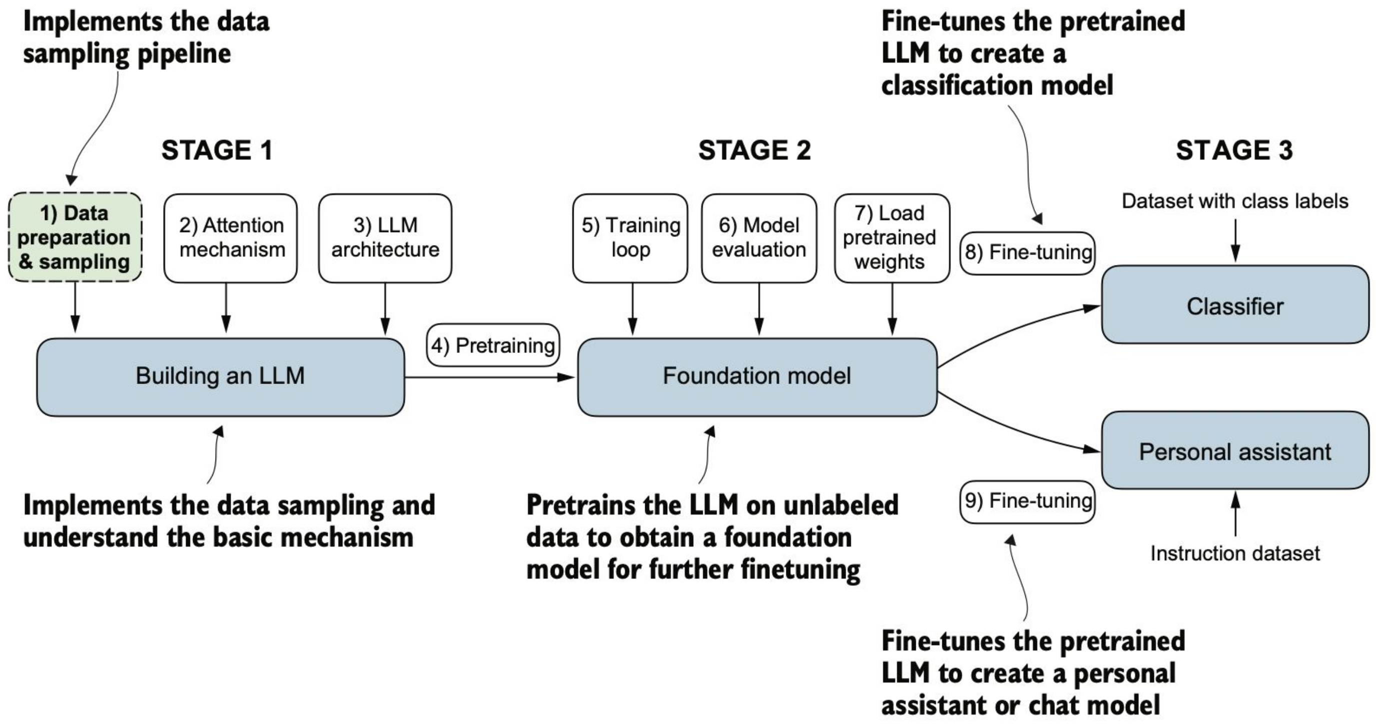

Implements the data sampling pipeline
STAGE 1
1) Data preparation & sampling
2) Attention mechanism
3) LLM architecture
Building an LLM
4) Pretraining
Implements the data sampling and understand the basic mechanism
STAGE 2
5) Training loop
6) Model evaluation
7) Load pretrained weights
Foundation model
Pretrains the LLM on unlabeled data to obtain a foundation model for further finetuning
Fine-tunes the pretrained LLM to create a classification model
STAGE 3
Dataset with class labels
8) Fine-tuning
Classifier
Personal assistant
9) Fine-tuning
Instruction dataset
Fine-tunes the pretrained LLM to create a personal assistant or chat model

Figure 2.1 The three main stages of coding an LLM. This chapter focuses on step 1 of stage 1: implementing the data sample pipeline.

Part I: The Atoms of Language - Text Representation

I.I - Tokenization - The Fundamental Unit

The First Step: We'll start with **tokenization**—the process of **breaking text** into a **sequence** of **discrete units** (words, subwords, or characters).

Vocabulary and Token IDs: Each **unique token** is **mapped** to an **integer ID** via a vocabulary. We'll discuss how simple methods struggle with **out-of-vocabulary words**, requiring **a special `<unk>` token**.

The Modern Solution - BPE: We will focus on **Byte-Pair Encoding (BPE)**, the **subword algorithm** used by **GPT models** that handles any word by breaking it into smaller, known pieces, making the **`<unk>` token unnecessary**.

A Comparative Look: We'll analyze **how different real-world tokenizers** (from BERT, GPT-4, StarCoder2) **handle the same text.**

Part I: The Atoms of Language - Text Representation

I.II - Vectorization - Turning Tokens into Tensors

Token Embeddings: We will move **from integer token IDs** to **continuous vector representations** using a **learnable** embedding layer, which functions as a large lookup table.

The Position Problem: A crucial concept is that the Transformer's **self-attention mechanism is position-agnostic. Without explicit positional** information, **it cannot understand word order**.

Positional Embeddings: The solution is to add a second, **learnable positional embedding vector** to the token embedding. The **final model input** is the **sum of these two vectors**, encoding both **semantic** meaning and **sequential order**.

Part I: The Atoms of Language - Text Representation

I.III - Preparing Data for Model Training

The Autoregressive Objective: LLMs are trained to **predict** the **next token** in a **sequence**.

The Sliding Window: We'll detail **how a long text sequence** is converted **into** a multitude of (**input, target**) **pairs** for **training** using a sliding window approach.

The Data Loader: In practice, this data is managed by a Data Loader, which **creates batches for efficient**, **parallelized training** on a **GPU**.

Part I: The Atoms of Language - Text Representation

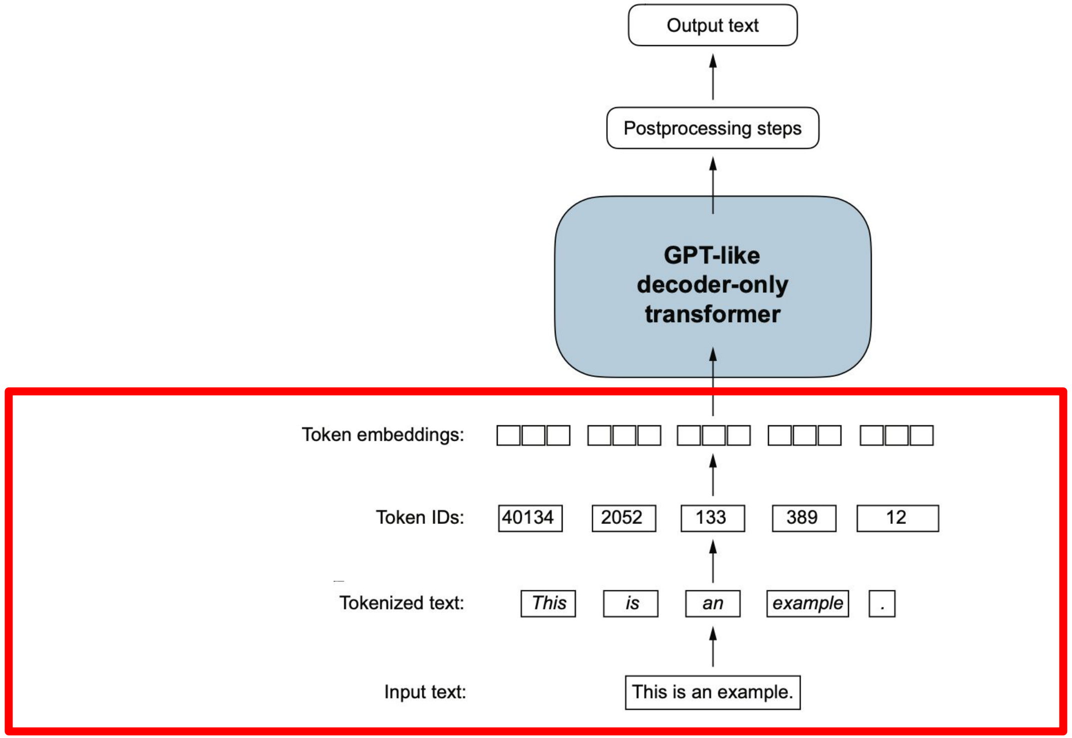


Figure 2.4 A view of the text processing steps in the context of an LLM. Here, we split an input text into individual tokens, which are either words or special characters, such as punctuation characters.

Understanding embeddings

An embedding is a **mapping** from **discrete objects** to **points** in a continuous vector space.

This **conversion** is **crucial** because **LLMs operate** on **mathematical principles**, and **raw text** (or any discrete data) needs to be **represented numerically** for **computation.**

Embeddings are not limited to text

Different data formats, such as video, audio, and text, require distinct embedding models to convert them into dense vector representations.

Embeddings are also useful outside of text and language generation

e.g., recommender engines and robotics.

Understanding embeddings

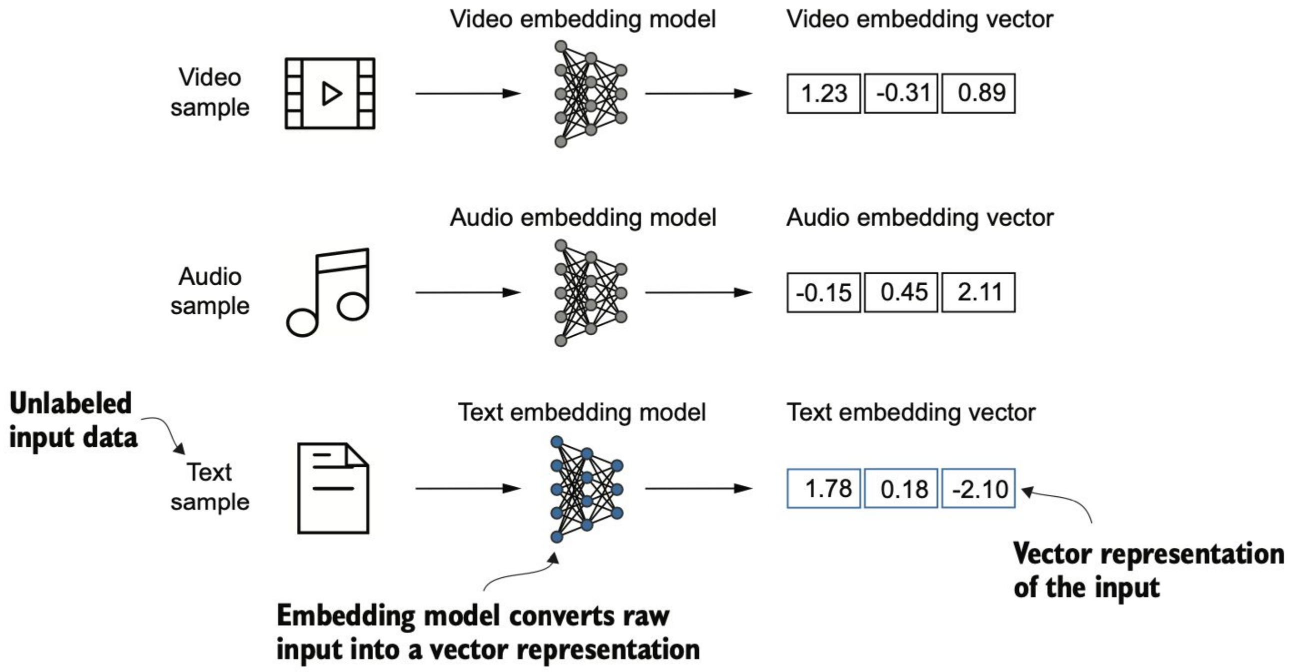


Figure 2.2 Deep learning models cannot process data formats like video, audio, and text in their raw form. Thus, we use an embedding model to transform this raw data into a dense vector representation that deep learning architectures can easily understand and process. Specifically, this figure illustrates the process of converting raw data into a three-dimensional numerical vector.

Understanding embeddings

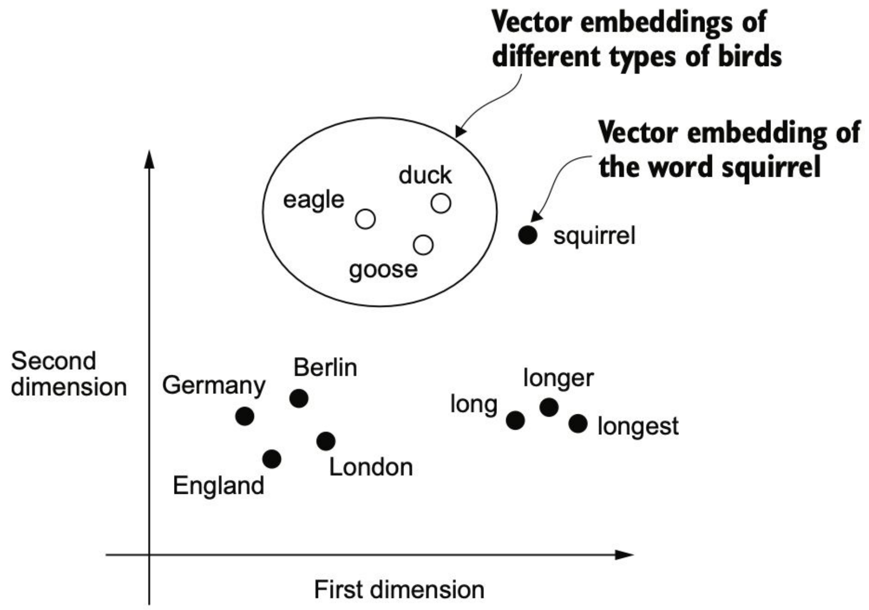


Figure 2.3 If word embeddings are two-dimensional, we can plot them in a two-dimensional scatterplot for visualization purposes as shown here. When using word embedding techniques, such as Word2Vec, words corresponding to similar concepts often appear close to each other in the embedding space. For instance, different types of birds appear closer to each other in the embedding space than in countries and cities.

Part I: The Atoms of Language - Text Representation

Notebook!

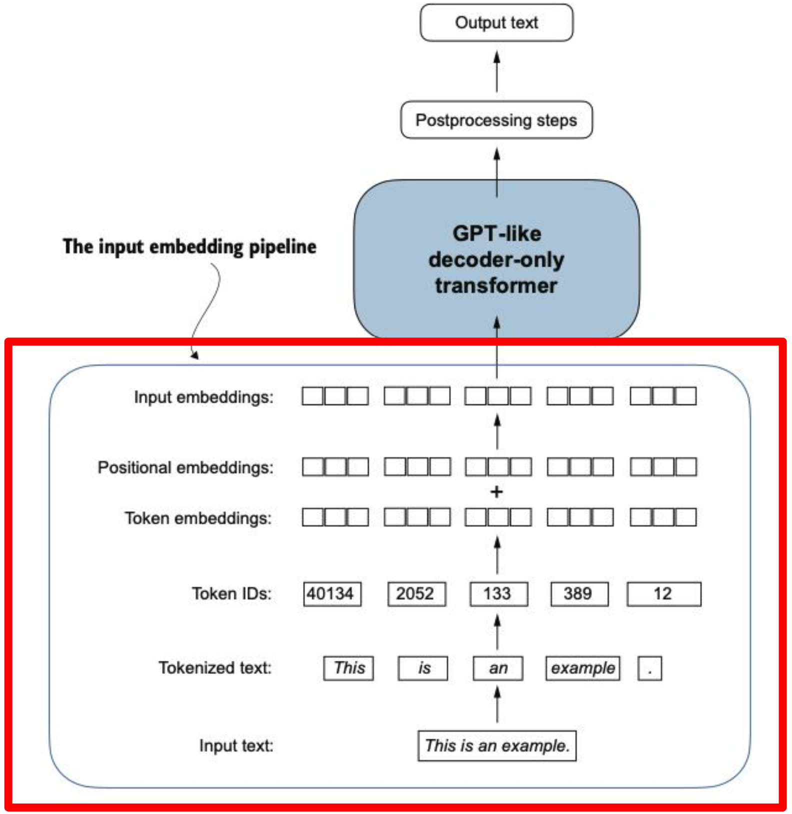
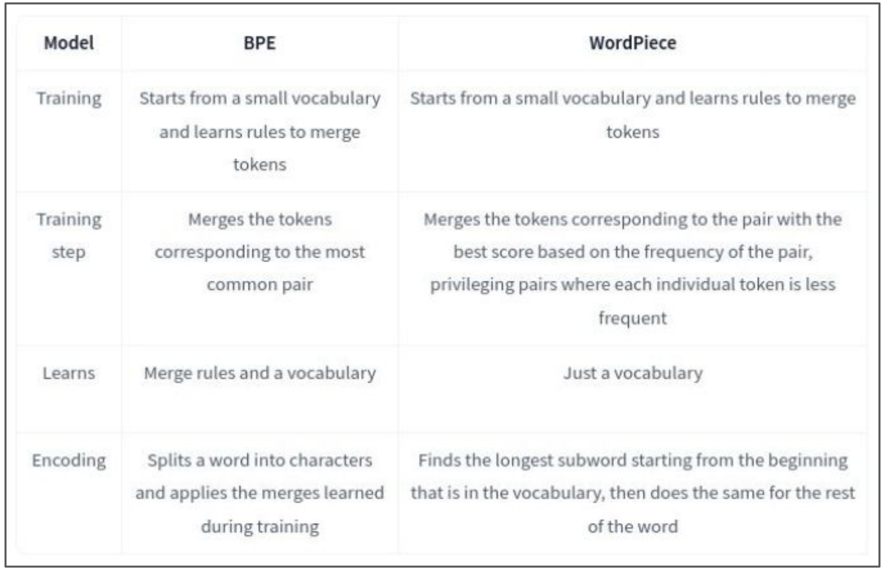


Figure 2.19 As part of the input processing pipeline, input text is first broken up into individual tokens. These tokens are then converted into token IDs using a vocabulary. The token IDs are converted into embedding vectors to which positional embeddings of a similar size are added, resulting in input embeddings that are used as input for the main LLM layers.

Tokenization Algorithms - Two components

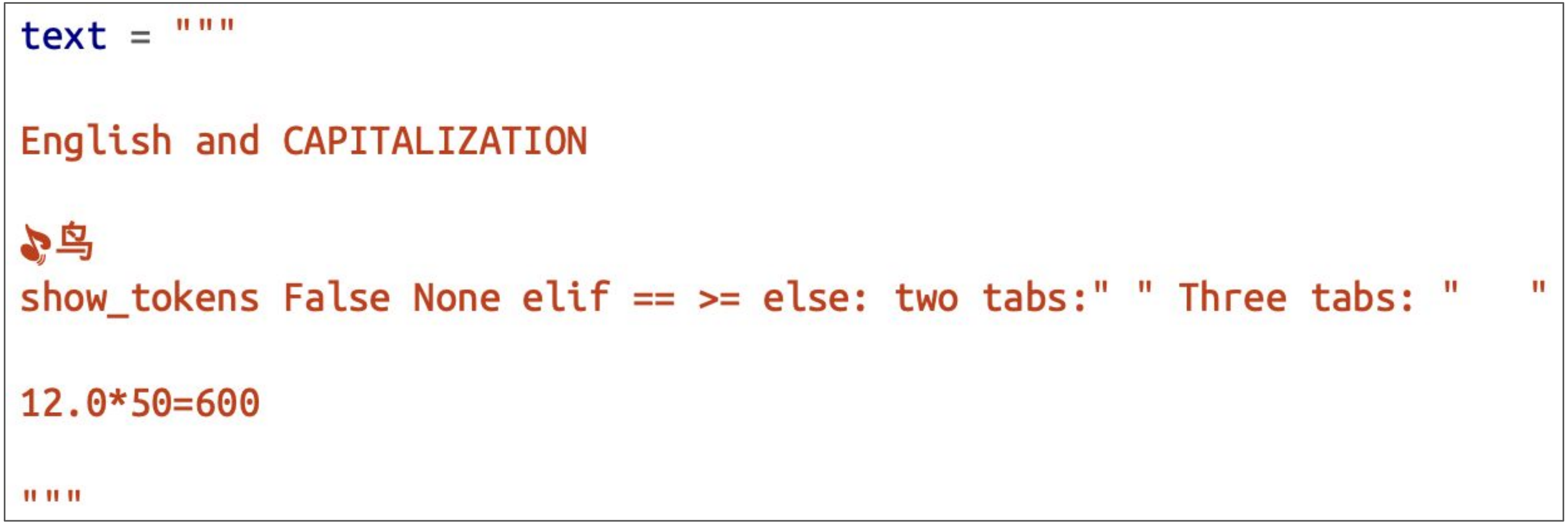
The *training* algorithm: preprocessing on a training set, to determine what will be the tokens.

The ***tokenization*** algorithm: at run time, transforming text inputs into sequences of tokens.



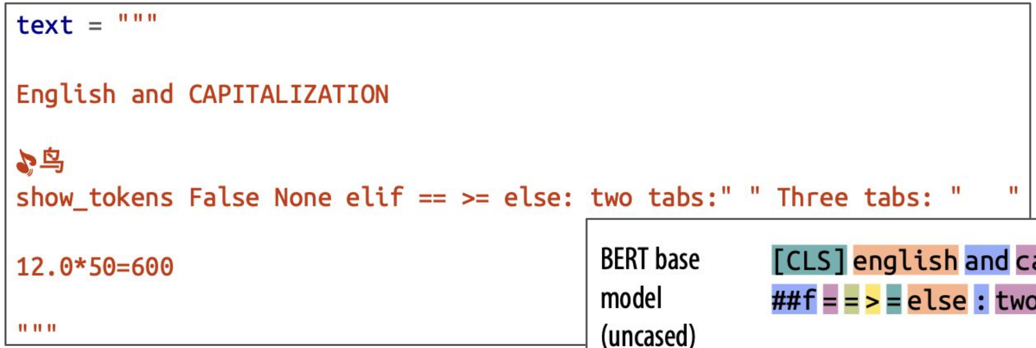
Model	BPE	WordPiece
Training	Starts from a small vocabulary and learns rules to merge tokens	Starts from a small vocabulary and learns rules to merge tokens
Training step	Merges the tokens corresponding to the most common pair	Merges the tokens corresponding to the pair with the best score based on the frequency of the pair, privileging pairs where each individual token is less frequent
Learns	Merge rules and a vocabulary	Just a vocabulary
Encoding	Splits a word into characters and applies the merges learned during training	Finds the longest subword starting from the beginning that is in the vocabulary, then does the same for the rest of the word

Comparing Trained LLM Tokenizers

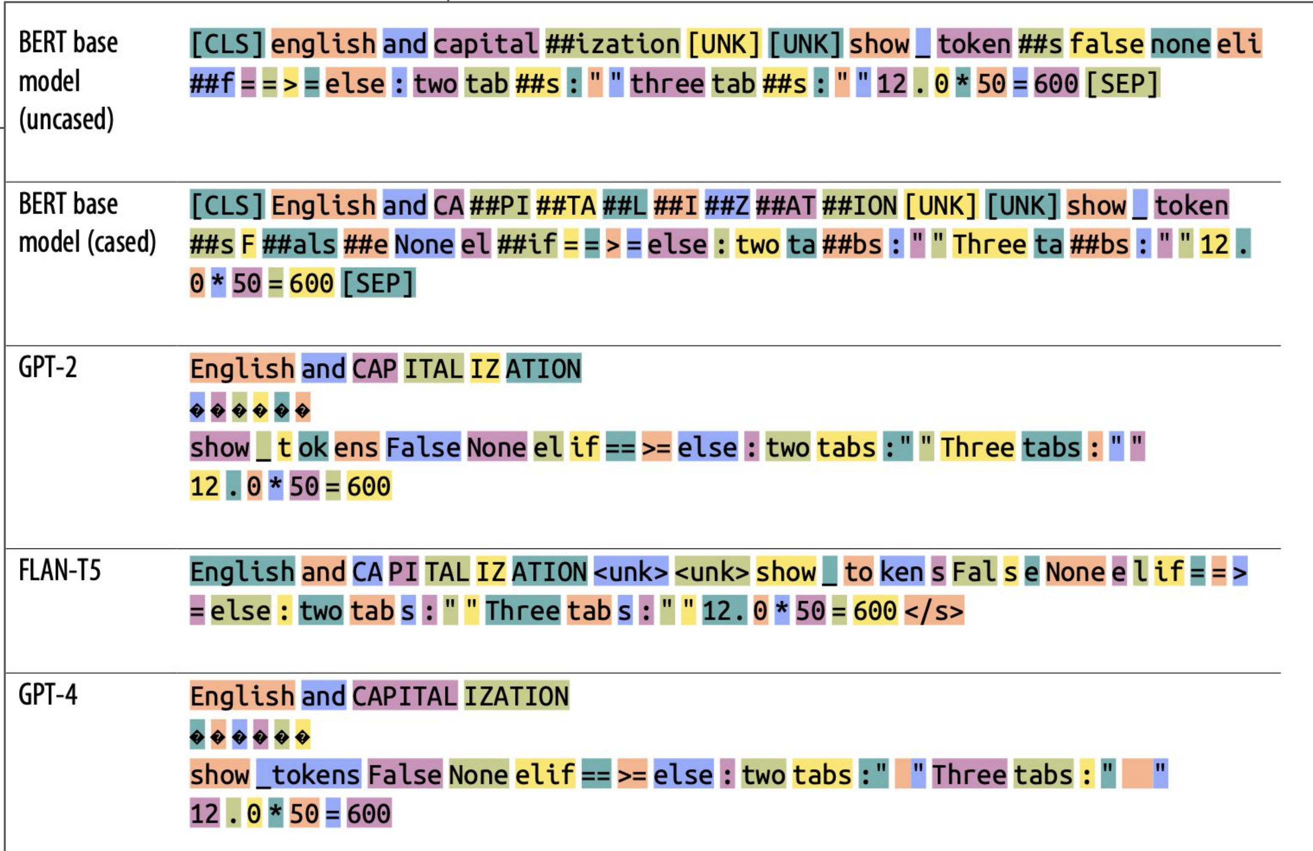


```
text = """

English and CAPITALIZATION

🎶鸟
show_tokens False None elif == >= else: two tabs:"  " Three tabs: "    "

12.0*50=600

"""
```

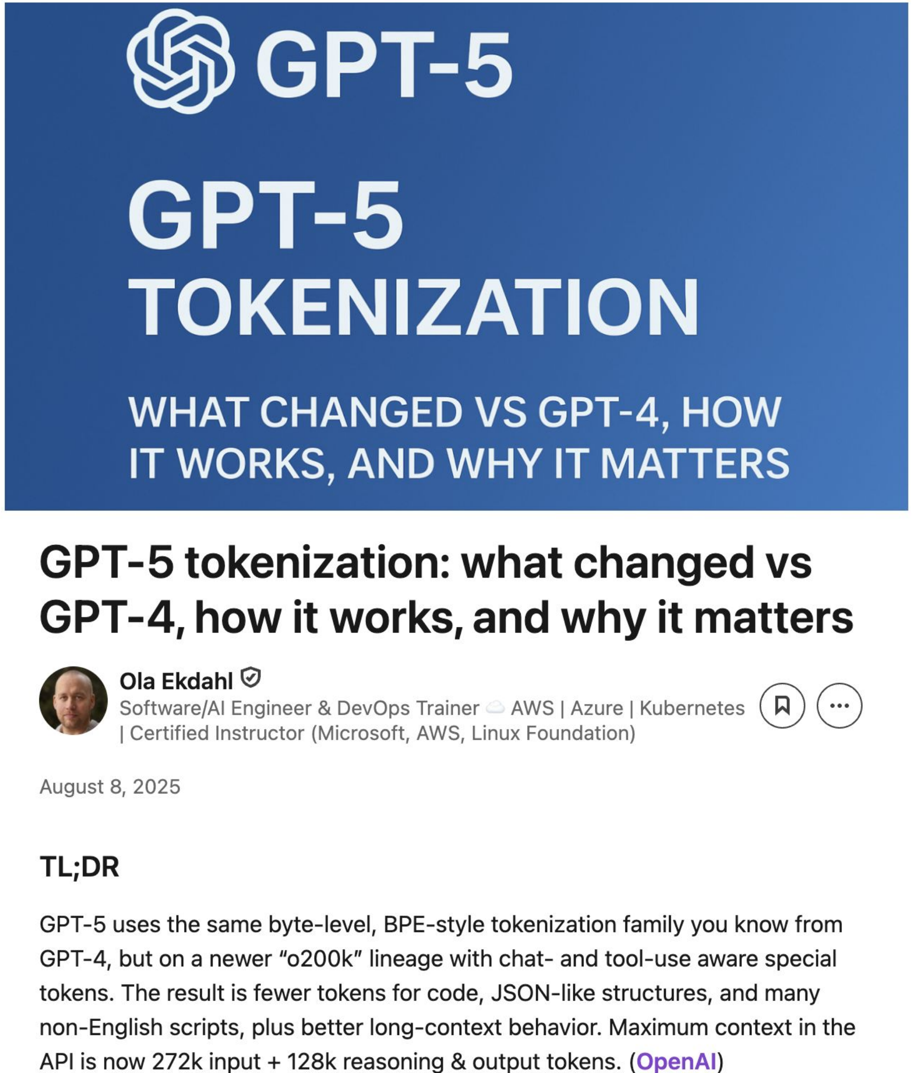
▌Capitalization ▌Emojis ▌Programming code ▌Numbers and digits



```
text = """

English and CAPITALIZATION

🎵鸟
show_tokens False None elif == >= else: two tabs:"    " Three tabs: "      "

12.0*50=600

"""
```



BERT base model (uncased)	[CLS] english and capital ##ization [UNK] [UNK] show _ token ##s false none eli ##f = = > = else : two tab ##s : " " three tab ##s : " " 12 . 0 * 50 = 600 [SEP]
BERT base model (cased)	[CLS] English and CA ##PI ##TA ##L ##I ##Z ##AT ##ION [UNK] [UNK] show _ token ##s F ##als ##e None el ##if = = > = else : two ta ##bs : " " Three ta ##bs : " " 12 . 0 * 50 = 600 [SEP]
GPT-2	English and CAP ITAL IZ ATION � � � � � � show _ t ok ens False None el if == >= else : two tabs :" " Three tabs : " " 12 . 0 * 50 = 600
FLAN-T5	English and CA PI TAL IZ ATION <unk> <unk> show _ to ken s Fal s e None e l if = = > = else : two tab s : " " Three tab s : " " 12. 0 * 50 = 600 </s>
GPT-4	English and CAPITAL IZATION � � � � � � show _tokens False None elif == >= else : two tabs :" " Three tabs : " " 12 . 0 * 50 = 600



GPT-5

GPT-5
TOKENIZATION

WHAT CHANGED VS GPT-4, HOW
IT WORKS, AND WHY IT MATTERS

GPT-5 tokenization: what changed vs GPT-4, how it works, and why it matters

Ola Ekdahl
Software/AI Engineer & DevOps Trainer ☁ AWS | Azure | Kubernetes | Certified Instructor (Microsoft, AWS, Linux Foundation)

August 8, 2025

TL;DR

GPT-5 uses the same byte-level, BPE-style tokenization family you know from GPT-4, but on a newer "o200k" lineage with chat- and tool-use aware special tokens. The result is fewer tokens for code, JSON-like structures, and many non-English scripts, plus better long-context behavior. Maximum context in the API is now 272k input + 128k reasoning & output tokens. (OpenAI)

https://www.linkedin.com/pulse/gpt-5-tokenization-what-changed-vs-gpt-4-how-works-why-ola-ekdahl-iewyc/

Roadmap

To master the art of creating high-impact datasets

Part I: The Atoms of Language - Text Representation

Fine-grained mechanics of converting text into structured numerical tensors.

Part II: Data Augmentation and Synthesis

We will now address how to get more data. We'll differentiate between data **augmentation** (creating new data from existing real data) and data **synthesis** (generating new data from scratch).

Part III: The Research Frontier

Model collapse—the risk of models degrading when recursively trained on their own output—and the importance of **mixing synthetic data with real human data** to maintain quality and diversity.

The role of synthetic data

Data augmentation

involves creating new data from existing, real data (e.g., flipping an image of a cat to create a new image of the same cat).

Data synthesis

generates data that mimics the properties of real data (e.g., simulating how a mouse moves through a webpage to generate bot movement data).

> While distinct, the terms "data augmentation" and "data synthesis" are sometimes **used interchangeably** because both aim to **automate data creation.**

Position: Will we run out of data? Limits of LLM scaling based on human-generated data

Pablo Villalobos [1] Anson Ho [1] Jaime Sevilla [1 2] Tamay Besiroglu [1 3] Lennart Heim [1 4] Marius Hobbhahn [1 5]

Abstract

We investigate the potential constraints on LLM scaling posed by the availability of public human-generated text data. We forecast the growing demand for training data based on current trends and estimate the total stock of public human text data. Our findings indicate that if current LLM development trends continue, models will be trained on datasets roughly equal in size to the available stock of public human text data between 2026 and 2032, or slightly earlier if models are overtrained. We explore how progress in language modeling can continue when human-generated text datasets cannot be scaled any further. We argue that synthetic data generation, transfer learning from data-rich domains, and data efficiency improvements might support further progress.

1. Introduction

Recent progress in language modeling has relied heavily on

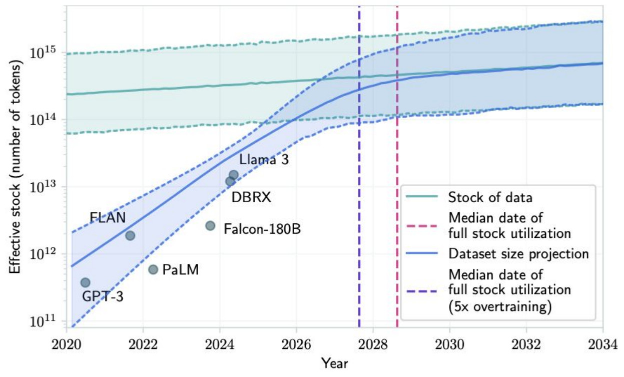


Figure 1. Projections of the effective stock of human-generated public text and dataset sizes used to train notable LLMs. The intersection of the stock and dataset size projection lines indicates the median year (2028) in which the stock is expected to be fully utilized if current LLM development trends continue. At this point, models will be trained on dataset sizes approaching the total effective stock of
tokens, corresponding to t
non-overtrained models. In
specific notable models. Th

In this position paper, we argue that human-generated public text data cannot sustain scaling beyond this decade. To support this conclusion, we develop a model of

Why Programmatically Generate Data?

E.g., instruction Data Synthesis

For **supervised fine-tuning**, AI can synthesize

both **instructions** and **responses**,

or even **generate prompts** for existing high-quality content (reverse instruction) to avoid AI-generated **hallucinations** in responses.

This includes generating **complex coding problems** and **solutions**, along with **unit tests**, which can be **programmatically** verified for **correctness**.

Why Programmatically Generate Data?

Increase data quantity

Produce data at scale, for training and testing.

In theory, helps models generalize to a wider range of tasks.

Increase data coverage

Focus on targeted characteristics or get a model to express specific behaviors.

E.g., conversations that contain toxic phrases for toxic detection model.

In Perez'22, Anthropic discussed various data synthesis techniques to generate specific datasets that can test 154 AI behaviors

- personality traits, political views, and social biases.

Why Programmatically Generate Data?

Increase data quality

- Sometimes, humans might have fundamental limitations:
  - AI can generate math questions that are far more complex than avg humans.
- What about a AI-generated content detector?

Mitigate privacy concerns

- E.g., healthcare regulations.

To distill models

- Train a model to imitate the behavior of another model.
- Create a cheaper and/or faster model (distilled model).

Challenges of Full Programmatic Data Generation - E.g.,

Superficial imitation

In Gudihande'23, shows that imitation models are good at mimicking the style of the teacher models but might struggle with factual accuracy and generalization to tasks outside the training data.

Quality Control

AI-generated data can be of low quality ("garbage in, garbage out"), and verifying its quality remains a challenge.

Roadmap

To master the art of creating high-impact datasets

Part I: The Atoms of Language - Text Representation

Fine-grained mechanics of converting text into structured numerical tensors.

Part II: Data Augmentation and Synthesis

We will now address how to get more data. We'll differentiate between data **augmentation** (creating new data from existing real data) and data **synthesis** (generating new data from scratch).

Part III: The Research Frontier

Model collapse—the risk of models degrading when recursively trained on their own output—and the importance of **mixing synthetic data with real human data** to maintain quality and diversity.

What is Model Collapse?

Model collapse

Irreversible defects that can emerge in models when they are recursively trained on AI-generated data.

The model's performance degrades over iterations of training with data that was itself generated by AI.

> This phenomenon has been observed in **various models**, including **Variational Autoencoders**, **Gaussian mixture models**, and Large Language Models (**LLMs**), and can occur during both **pre-training** and **post-training** phases.

The causes of Model Collapse

AI models tend to generate probable or common events more frequently than improbable or rare events.

Over multiple iterations of training on such synthetic data

probable events become **over-represented** in the generated datasets, while **improbable or rare events** become **under-represented** or are **"forgotten"** by the models.

E.g., an AI model might generate

data predominantly about **"not having cancer"** (a probable event) and much less about **"having cancer"** (an improbable event).

If **subsequent models** are then **trained** on this **skewed synthetic data**, they will **increasingly output common events** and **lose** the **ability** to **predict** or represent **rare** ones

Mitigation Strategies

Although **there isn't** a definitive recommendation for the **optimal proportion of synthetic to real data.**

Some research has shown that **large amounts of synthetic data can still be effective**, with synthetic data being "nearly as effective as real data" in finetuning models for **specific tasks** like math problems.

E.g., Nemotron-4 340B-Instruct used 98% synthetic data during its instruction and preference finetuning.

• The Llama 3 paper also noted that while **training on data from a more competent model can improve performance**,

Indiscriminately training on self-generated data can degrade it.

However, they found that by introducing **mechanisms to verify the quality of synthetic data** and **using only verified data**, they could **improve a model using its own generated data**.

Recap

Part I: The Atoms of Language - Text Representation

Fine-grained mechanics of converting text into structured numerical tensors.

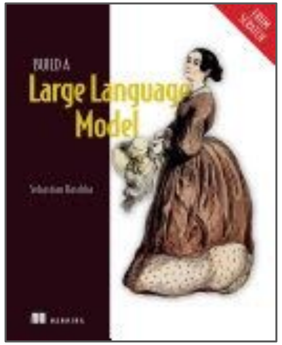
Part II: Data Augmentation and Synthesis

We will now address how to get more data. We'll differentiate between data **augmentation** (creating new data from existing real data) and data **synthesis** (generating new data from scratch).

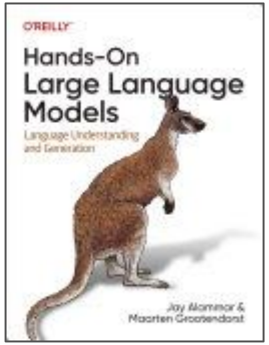
Part III: The Research Frontier

Model collapse—the risk of models degrading when recursively trained on their own output—and the importance of **mixing synthetic data with real human data** to maintain quality and diversity.

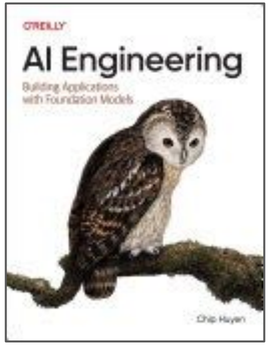
References



Ch 2



Ch 2



Ch 8

Github
Research papers