INTRODUCTION TO NATURAL LANGUAGE PROCESSING

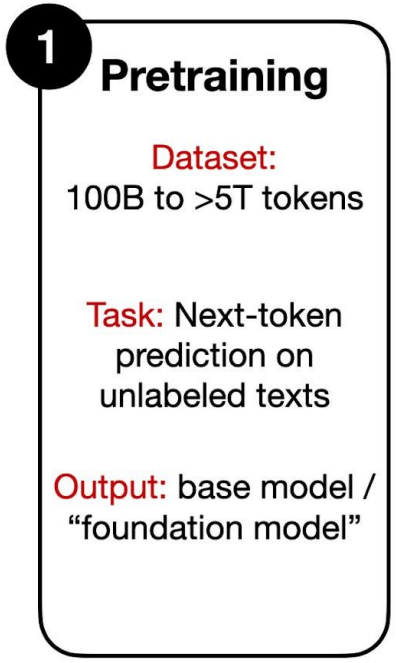
Pretraining Large Language Models

Victor Zhong

Introduction to Natural Language Processing

What is a Large Language Model (LLM)?

- **Definition:** A computational agent capable of conversational interaction and text generation.
- **Fundamentally:** A probabilistic model that predicts the next token in a sequence based on context.
- **Core Mechanism:** Conditional Generation.
  - **Input:** Context (Prefix)
  - **Output:** Probability distribution over next tokens



Project Gutenberg (PG) is a volunteer effort to digitize and archive cultural works, as well as to "encourage the creation and distribution of eBooks." It was founded in 1971 by American writer Michael S. Hart and is the oldest digital **library**. Most of the items in its collection are the full texts of books or individual stories in the public domain. All files can be accessed for free under an open format layout, available on almost any computer. As of 3 October 2015, Project Gutenberg had reached 50,000 items in its collection of free eBooks.

Visualizing the conditional probability of the next word given a context history.

https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

The Evolution of "Chat"

ELIZA (1966)

Joseph Weizenbaum's rule-based system simulating a Rogerian psychologist.

- **The ELIZA Effect:** Humans easily form emotional connections with machines.
- **Limitation:** No real understanding; purely pattern matching/rules.

Modern LLMs

Unlike rule-based systems, these learn language patterns and world knowledge from vast corpora.

ELIZA (1966): The first "chatbot" relied on pattern matching, not learning.

The Knowledge Bottleneck

Human Learning

Children learn ~7–10 words/day to reach adult vocabularies of 30k–100k words.

Most knowledge is acquired as a by-product of reading and contextual processing.

Machine Learning

Distributional Hypothesis: We learn meaning from the company words keep.

The NLP revolution relies on this principle: learning syntax, semantics, and facts from data distribution.

Pretraining: The Core Idea

Definition

Learning knowledge about language and the world by iteratively predicting tokens in massive text corpora.

The Result

A "Pretrained" model containing rich representations of syntax, semantics, and facts.

Foundation

Serves as the base for downstream tasks (QA, translation). This phase is **self-supervised**.

Learning from Context

What does a model learn by predicting the blank?

"With roses, dahlias, and peonies, I was surrounded by ____"

Learns: Ontology (Category: Flowers)

"The room wasn't just big it was ____"

Learns: Semantics/Intensity (enormous > big)

"The square root of 4 is ____"

Learns: Arithmetic/Math

"The author of 'A Room of One's Own' is ____"

Learns: World Knowledge (Virginia Woolf)

Three Major Architectures

Decoder-only

Generative (GPT-4, Llama)

Encoder-only

Understanding (BERT)

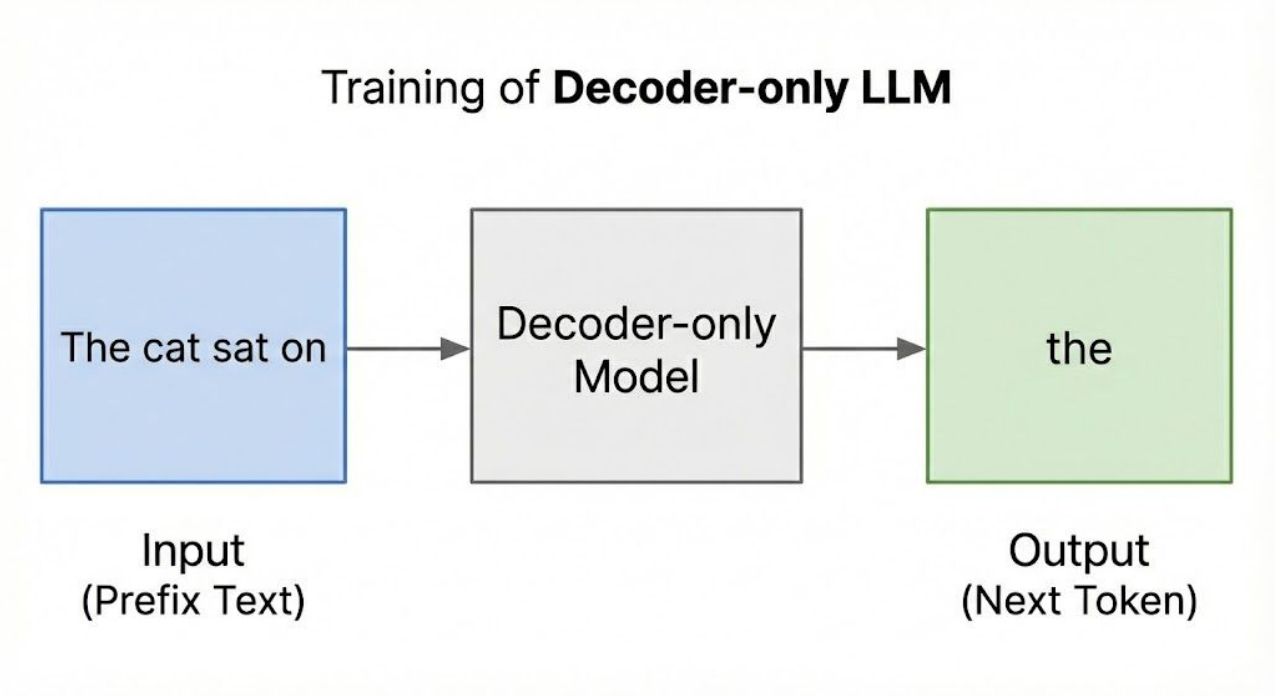
Encoder-Decoder

Translation (T5, BART)

1. Decoder Architecture

- **Mechanism:** Auto-regressive generation. Takes tokens as input and generates tokens one by one (left-to-right).
- **Causal:** Masked attention ensures it can only see past tokens, not future ones.
- **Use Case:** Generative tasks (text generation, code completion).
- **Examples:** GPT-3, GPT-4, Llama, Claude, Mistral.

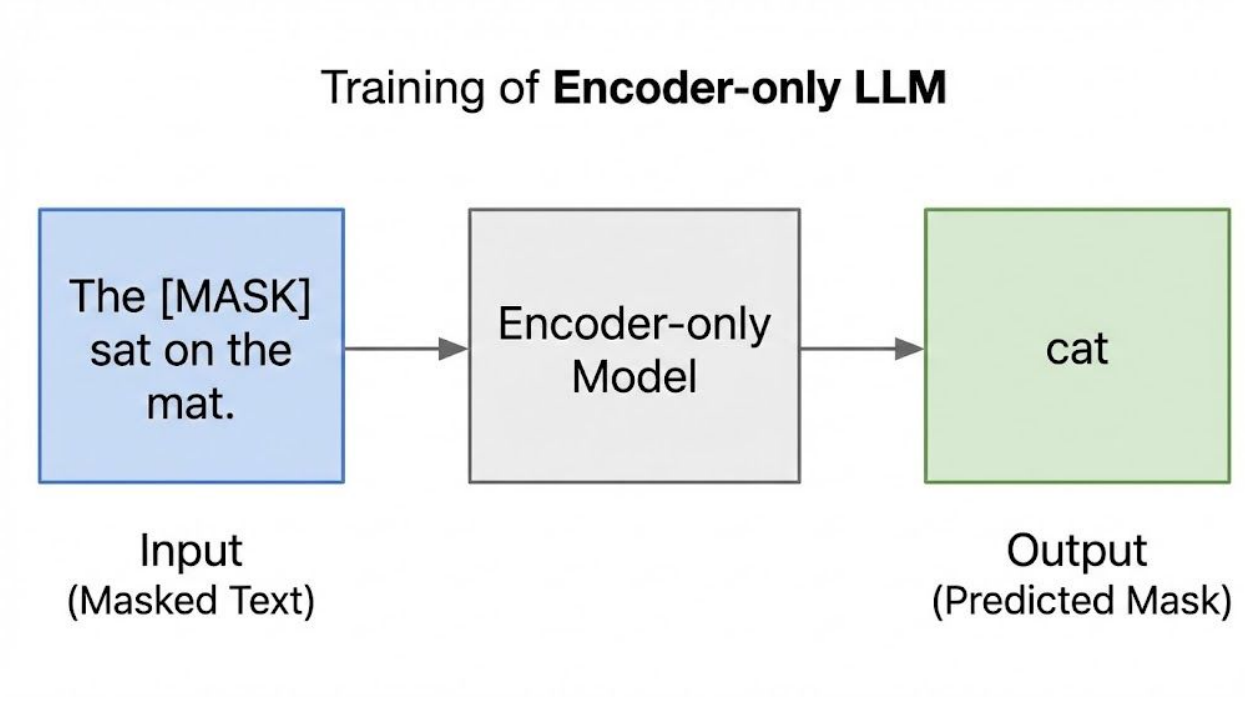
> **Focus:** We focus on decoders as they are the standard for generative LLMs.



The Decoder block uses masked self-attention to predict the next token.

2. Encoder Architecture

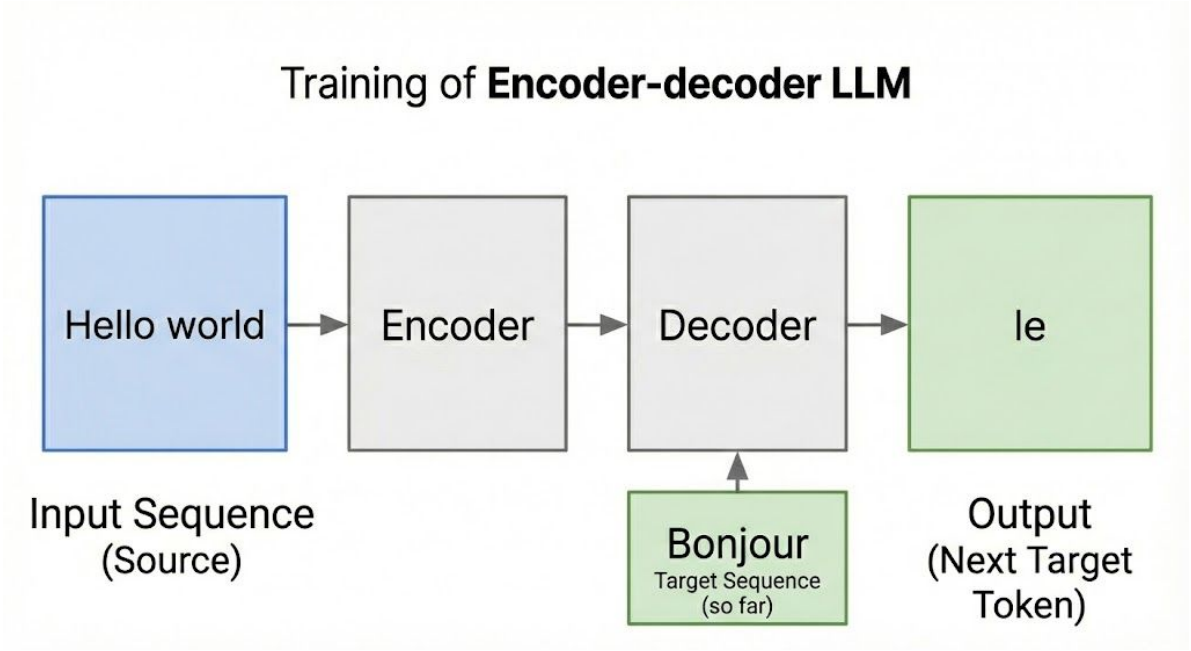
- **Mechanism:** Outputs a vector representation (encoding) for each input token.
- **Bidirectional:** Can see context from both left and right directions simultaneously.
- **Training:** Typically Masked Language Modeling (predicting missing words in the middle).
- **Use Case:** Classification, Sentiment Analysis, NER.
- **Examples:** BERT, RoBERTa.



Encoders process the entire sequence at once to build rich representations.

3. Encoder-Decoder Architecture

- **Mechanism:** Maps an input sequence of tokens to a (potentially different) output sequence.
- **Key Feature:** Decouples input understanding from output generation. Input and output can have different lengths.
- **Use Case:** Translation (English → French), Summarization.
- **Examples:** T5, BART.



Combines an Encoder to understand input and a Decoder to generate output.

The "Black Box" View

Input

Sequence of tokens (Context).

The cat sat on the

Output

Probability distribution over the vocabulary for the next token.

mat: 0.5

bench: 0.3

dog: 0.01

$$P(w_t = i \mid w_{<t}) = \frac{\exp(f_\theta(w_{<t})_i)}{\sum_{j=1}^{|V|} \exp(f_\theta(w_{<t})_j)}$$

$$P(w_t \mid w_1, \ldots, w_{t-1}) = \text{Softmax}(f_\theta(w_1, \ldots, w_{t-1}))$$

We sample from this distribution to generate the next word.

Self-Supervised Learning

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{<t}; \theta)$$

- **The Idea:** We do not need manually labeled data (e.g., "this sentence is positive").
- **The Supervision:** The text *itself* is the supervision.
- **Task:** At every step t, predict the next word w_{t+1} given the history $w_{1:t}$.
- **Ground Truth:** Since we possess the source text, we always know the correct next token.

Cross-entropy loss

We train the network to minimize **Cross-Entropy Loss**.

It measures the difference between the model's predicted distribution and the true distribution.

The loss simplifies to the negative log probability of the correct next word.

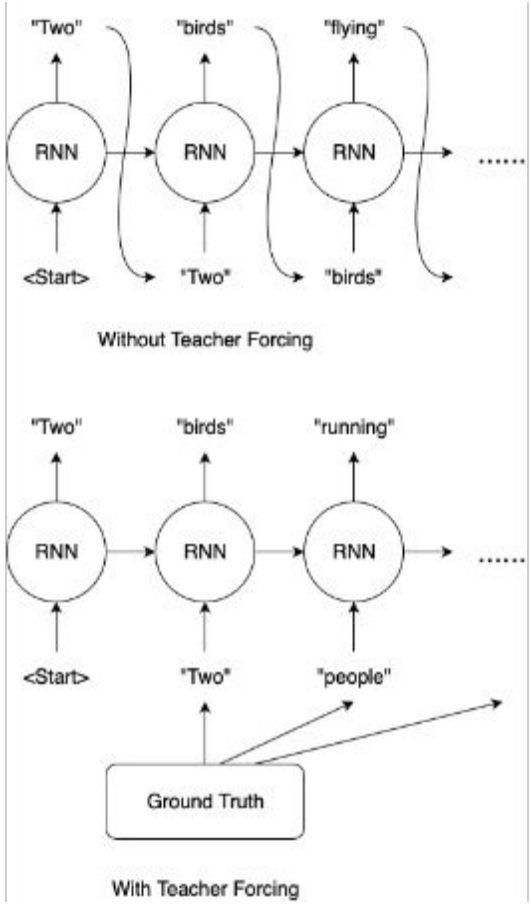
$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{<t}; \theta)$$

Teacher Forcing

Inference vs. Training

- **Inference:** Errors accumulate as model predicts its own future context.
- **Training (Teacher Forcing):**
  - We do not use the model's own prediction.
  - We always feed the **correct history sequence**.

Advantage: Allows massive parallelization (all tokens trained simultaneously).



Comparison of Autoregressive Inference (Left) vs Teacher Forcing (Right).

Training Visualization

1. **Input:** Sequence of tokens $w_1, \ldots, w_T$.

2. **Forward Pass:** Model computes distribution $\hat{y}_t$ for every position t simultaneously.

3. **Update:** Calculate Loss, Batch, and Backpropagate to update weights.

Conditional Generation

$$w_t \sim P(w_t \mid w_{<t}, \theta)$$

We model NLP tasks as conditional text generation.

Probability of token given all previous tokens.

$$P(W) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1})$$

Sampling Loop

- Compute probability distribution
- Sample token
- Add to context
- Repeat

How To Go From Logits to Text

1. Logits

Raw scores from model

2. Softmax

Convert to Probabilities

3. Decoding

Select next token

Greedy Decoding

Algorithm

Always select the single token with the highest probability.

$$\hat{w} = \text{argmax}_w P (w \mid \text{context})$$

Pros

Deterministic, optimal for short factual queries.

Cons

Often leads to repetitive, degenerate text. Misses creative paths.

Random Sampling

Algorithm

Sample the next token randomly according to the probability distribution $\mathrm{P}(\mathrm{w})$.

Effect: A word with 10% probability is chosen 10% of the time.

Pros

High diversity, human-like variance.

Cons

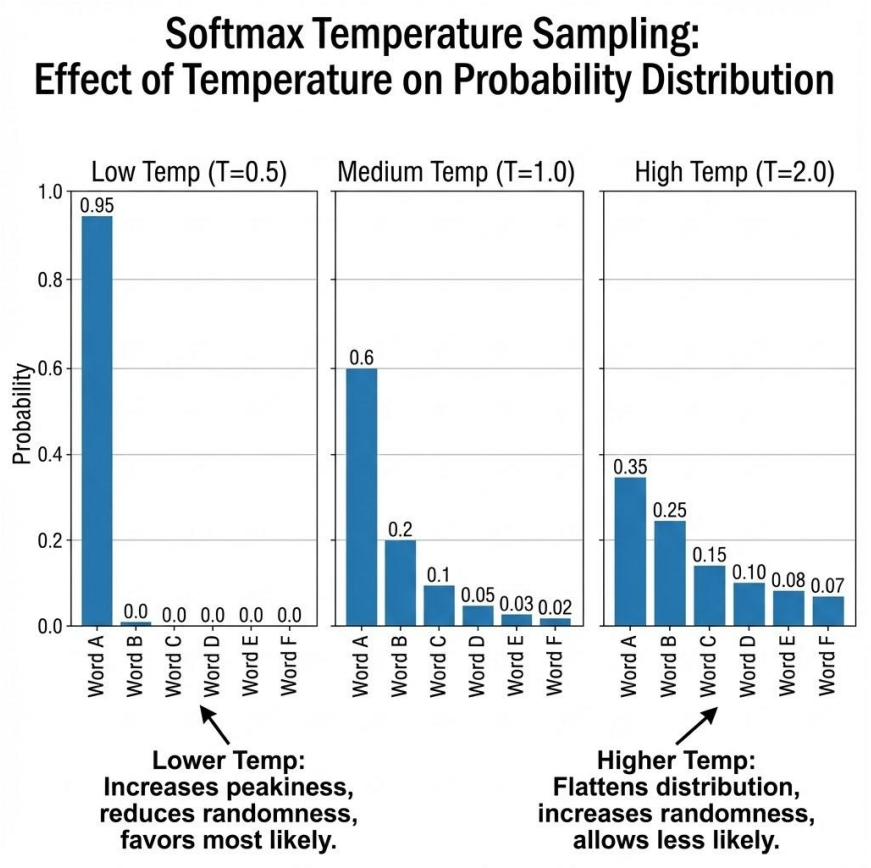
"Tail risk"—sampling a very low probability word that derails coherence.

Temperature Sampling

Rescale logits by a temperature parameter τ

$$P_i = \frac{\exp(u_i / \tau)}{\sum_j \exp(u_j / \tau)}$$

- **Low ($\tau < 1$):** Confident/Greedy.
- **High ($\tau > 1$):** Diverse/Creative.



Effect of temperature on probability distribution.

Top-k and Nucleus Sampling

Top-k

Only sample from the top k most likely words (e.g., top 50).

Cuts off the long tail of bad words.

Nucleus (Top-p)

Sample from the smallest set of words whose cumulative probability exceeds p (e.g., 0.9).

Dynamic: Adapts to uncertainty.

Preview: Task Modeling

Almost any task can be cast as next-token prediction.

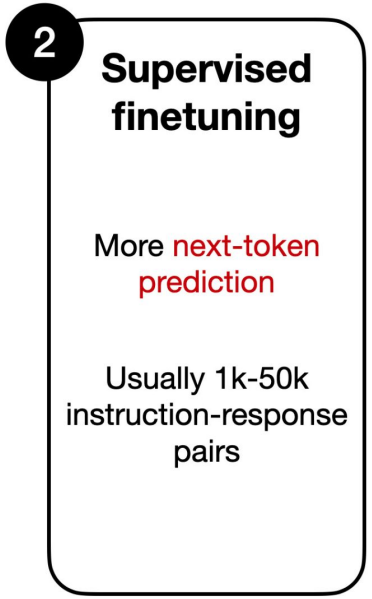
Sentiment Analysis Example

Input: "The sentiment of the sentence 'I like Jackie Chan' is:"

Model Compares:

P("positive" | context) vs. P("negative" | context)

Decision: Choose the token with higher probability.




```
{
    "instruction": "Write a limerick about a
                    pelican.”,
    "input": "",
    "output": "There once was a pelican so fine,
              \nHis beak was as colorful as
              sunshine,\nHe would fish all day,\nIn
              a very unique way,\nThis pelican was
              truly divine!\n\n\n"
},

{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```


https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

Preview: Question Answering

Example Flow

Input: "Q: Who wrote the book 'The Origin of Species'? A:"

Prediction 1: The model computes $P(w \mid \text{context})$.
Result: High probability for **"Charles"**.

Next Step: Add "Charles" to context.
New Input: "... A: Charles"
Prediction 2: High probability for **"Darwin"**.

What Determines Performance?

Performance (Loss) is determined by three main power-law factors:

N

Number of parameters (Model Size)

D

Dataset size (Number of tokens)

C

Compute budget (FLOPS)

Pretraining Corpora

LLMs are trained on massive datasets (trillions of tokens).

Web Crawls

Common Crawl, C4 (Colossal Clean Crawled Corpus).

Quality Data

Wikipedia, Books (fiction/non-fiction), arXiv papers.

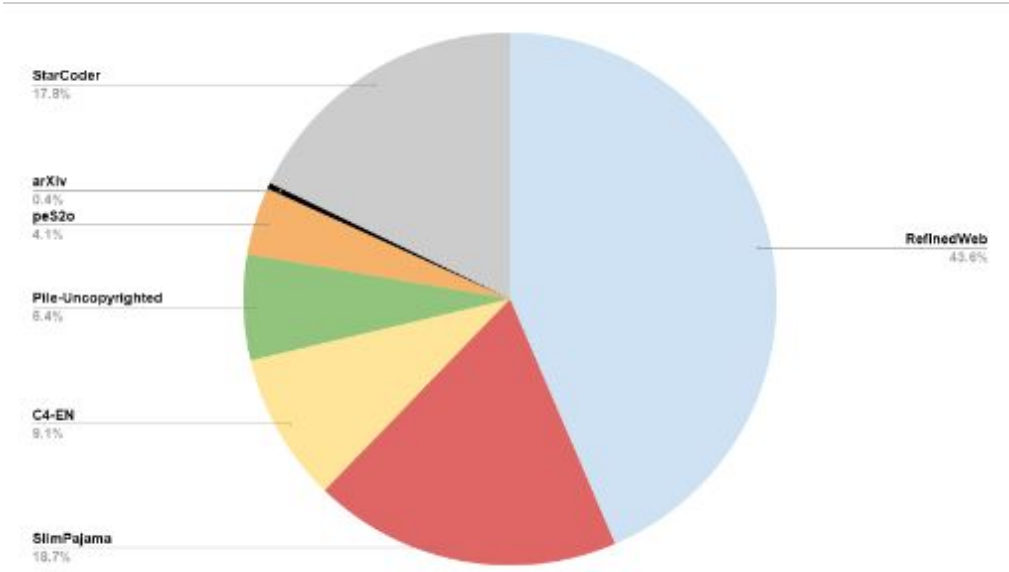
Code

GitHub repositories (critical for reasoning capabilities).

Dataset Examples: The Pile

The Pile (Gao et al., 2020): An 825 GB corpus of diverse text sources.

- **Academic:** PubMed, arXiv, USPTO Patents.
- **Internet:** CommonCrawl, OpenWebText, Wikipedia.
- **Prose:** BookCorpus, Gutenberg (Books).
- **Dialogue:** Subtitles, Chat logs.



Composition of The Pile: Balancing high-quality academic/prose with diverse web text.

Filtering: Quality

Raw web text is noisy. We must filter for quality.

Heuristics

Removing short documents, excessive symbols, or malformed text.

Model-Based

Train classifiers to distinguish "high-quality" text (e.g., Wikipedia-like) from "low-quality" text.

Deduplication

Remove duplicate documents. Reduces memorization and improves generalization.

Filtering: Safety & Ethics

PII Removal

Scrubbing Personally Identifiable Information (phones, emails, SSNs).

Toxicity Filtering

Removing hate speech and abusive content.

Challenges

- **Bias:** Classifiers may be biased against minority dialects (e.g., AAE).
- **Trade-off:** Models trained on sanitized data may be worse at detecting toxicity themselves.

Ethical & Legal Issues

- **Copyright:** Training on copyrighted books/code. Is it "fair use"? (Currently being litigated).
- **Consent:** Websites opting out of crawling (robots.txt).
- **Privacy:** Models can "leak" training data (extraction attacks).
- **Skew:** Data is disproportionately from the US/Global North, skewing model perspectives and values.

The Power Law of Scaling

Kaplan et al. (2020) empirically showed that loss L scales as a power law:

$$L(N, D) = E + \frac{A}{N^a} + \frac{B}{D^b}$$

- **Implication:** To improve performance, we must scale up N and D simultaneously.

- **Diminishing Returns:** To halve the error, you need exponentially more compute.

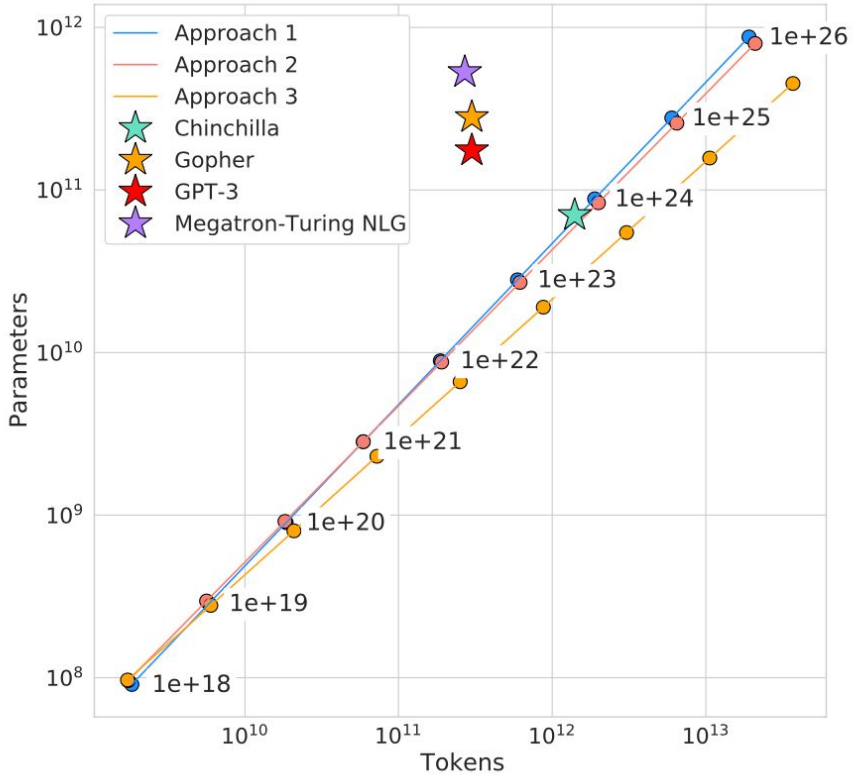
Compute-Optimal Scaling (Chinchilla)

The Chinchilla Ratio: To train optimally for a fixed compute budget, scale parameters (N) and data (D) equally.

~20 tokens per parameter

Example: A 70B model → 1.4 Trillion tokens.

Current Trend: Models are "over-trained" (Llama 3 used >200 tokens/param) to make inference cheaper.



The Chinchilla frontier defines the optimal model size for a given compute budget.

Approximate Parameter Count

For a Transformer model, the number of non-embedding parameters N is roughly:

$$N \approx 12 n_{layer} d^2$$

- n_{layer} : Number of layers
- d : Model dimensionality

Example: GPT-3

96 layers, $d = 12288$

≈ 175 Billion parameters

Evaluation

How do we know it works?

Intrinsic

Perplexity (Mathematical certainty)

Extrinsic

Zero-shot / Few-shot performance on downstream tasks.

Perplexity (Intrinsic)

Definition

The inverse probability of the test set, normalized by the number of words.

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, \ldots, w_{i-1})}}$$

Interpretation

The "branching factor" of the model.

If **PP = 10**, the model is as confused as if it were choosing uniformly from 10 words.

Goal: Lower is better.

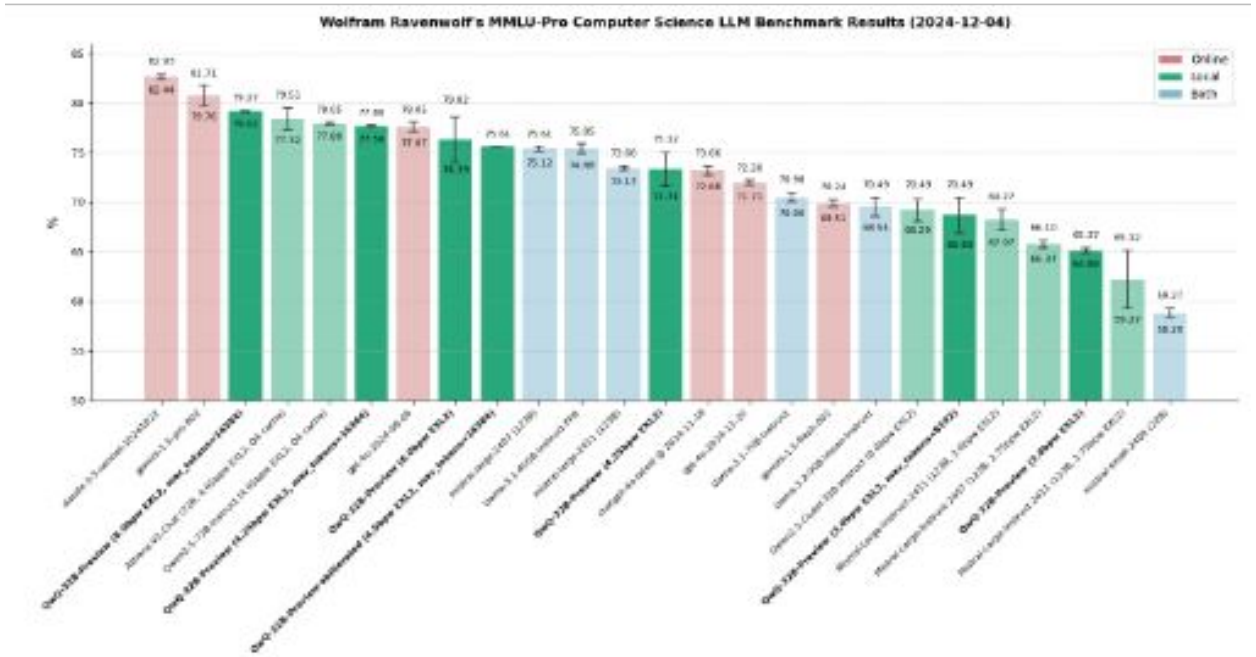
Downstream Benchmarks (Extrinsic)

Perplexity does not always predict reasoning ability.

We use standard benchmarks:

- **MMLU:** 15k+ multiple choice questions (Math, Law, Medicine).
- **Code:** HumanEval.
- **Reasoning:** GSM8K (Grade School Math).

Method: Zero-shot or Few-shot prompting.



MMLU scores demonstrate the reasoning capabilities of scaled models.

Data Contamination

The Problem

Did the model memorize the answer?
Contamination: When test set questions appear in the training data (e.g., inside Common Crawl).

Consequence & Mitigation

Consequence: High scores but fails to generalize to new problems.
Mitigation: Rigorous decontamination (n-gram matching) before training.

Using Pretrained Models: Prompting

Prompt

The input text provided to the model to elicit a specific response.

System Prompt

A hidden prefix instruction that defines the model's persona (e.g., "You are a helpful, harmless assistant.").

Prompt Engineering

The empirical science of designing prompts to maximize model performance.

In-Context Learning (ICL)

Definition

The ability of a model to improve performance on a specific task given context in the prompt, **without updating model parameters**.

Mechanism

The model "learns" the pattern or format from the input buffer activations.

Modes: Zero-shot and Few-shot.

Zero-shot Prompting

Providing the task instruction without any examples.

Example

```
"Translate the following sentence into French: The cat sat on the mat."
```

Reliance: Relies entirely on the model's pretraining data to understand the instruction verb ("Translate").

Few-shot Prompting

Providing k examples (demonstrations) of the task before the actual query.

Example (2-shot)

```
Translate English to French: Dog → Chien Cheese → Fromage Cat →
```

Effect: Significantly improves performance on complex or novel tasks by demonstrating the expected output format.

Why Do Demonstrations Work?

Format Constraints

They teach the model the output structure (e.g., "Answer with one word").

Task Location

They help the model "locate" the specific task manifold in its parameter space.

Counter-intuitive Finding

Correctness of labels matters less than format. Models improve even with incorrect labels (Min et al., 2022).

Summary

- **LLMs** are probabilistic models trained to predict the next token in a sequence.
- **Decoder architectures** are the standard for generative tasks.
- **Pretraining** is self-supervised: the data itself provides the labels.
- **Data** is vast, sourced from the web, and requires rigorous filtering for quality and safety.
- **Scaling Laws** tell us that performance improves predictably with more data, parameters, and compute.

The Three Stages of LLM Training

1. Pretraining (This Lecture)

Self-supervised next-word prediction on massive text.
Result: Base Model.

2. Instruction Tuning (SFT)

Supervised training on (Instruction, Response) pairs.

3. Alignment (RLHF/DPO)

Optimizing for human preferences (helpful, honest, harmless).