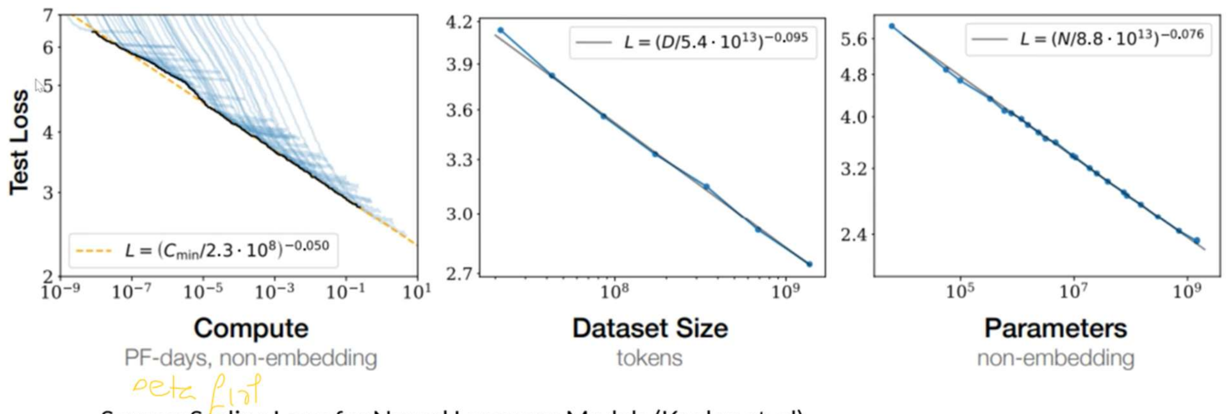
Fundamentals of Large Language Models explores various **LLMs**, detailing their scale, corporate origins, training methodologies, core strengths, and overall industry significance.

The course provides a concise overview of the **Transformer architecture**, highlighting the functional differences between its components. Encoder-only models, such as BERT, excel at language understanding tasks like sentence classification and Named Entity Recognition (NER). In contrast, the addition of a decoder block enables models to perform generative tasks. While GPT models utilize decoder-only architecture, models like BART and T5 incorporate both encoder and decoder blocks; both configurations are highly effective for generative AI applications.

The text also explores the concept of **embeddings**, specifically word and sentence embeddings. It explains how mathematical operations, such as the dot product and cosine similarity, are utilized to measure and identify semantic similarities between different texts.

While embedding ad LLMs are based on similar neural network architecture (like the transformer), embedding models and LLMs serve completely different purposes in an AI system. Think of embedding models as translator that turns text into a secret numerical code (vectors), and an LLM as a storyteller that uses that code to write back to you. The common use case of an embedding model includes search, recommendation and RAG while a LLMs are mostly used in chatbot, summarization and coding.

A key topic in the course is Scaling Law, which demonstrates that the performance of large models is a direct function of three critical variables: the number of model parameters, the size of the training dataset, and the total compute power allocated.

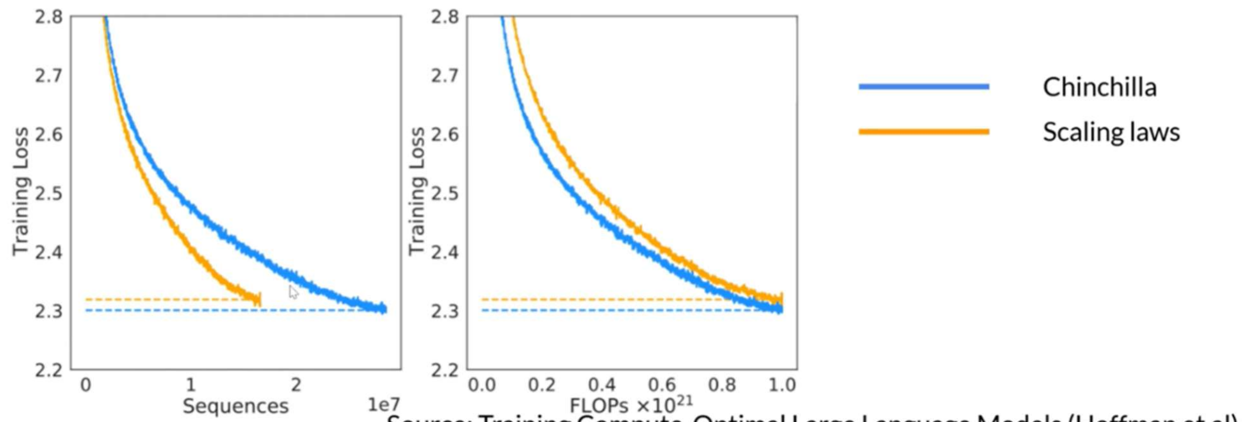


Source: Scaling Laws for Neural Language Models (Kaplan et. al)

Chinchilla model released by Google DeepMind in March 2022 changed how industry thinks about AI scaling. The model provided a clear answer to a **critical scaling question:** for a fixed budget, how should one balance model capacity against training duration? Before Chinchilla, the trend was to make models as large as possible (like the 175B GPT-3 or the 280B Gopher). DeepMind's research proved that many of these massive models were actually **"undertrained"** because they didn't have enough data to match their size.

The "**Chinchilla Scaling Laws**" provide a mathematical roadmap for training the most efficient model possible within a fixed "compute budget" (the total amount of GPU power and time available).

- **The 20:1 Ratio:** DeepMind found that for every doubling of model size, the training data should also double. Specifically, they estimated that a compute-optimal model should be trained on roughly **20 tokens for every 1 parameter**.
- **Small but Mighty:** To prove this, they built Chinchilla with only **70 billion parameters** (much smaller than GPT-3) but trained it on **1.4 trillion tokens** (much more data).
- **Superior Performance:** Despite being 2.5x smaller than GPT-3 and 4x smaller than Gopher, Chinchilla outperformed both on almost every major benchmark, including MMLU.



Source: Training Compute-Optimal Large Language Models (Hoffman et al)

While **LLM benchmarks** were mentioned throughout the course, they were specifically emphasized during the deep dives into the **Hugging Face** and **HELM** (Holistic Evaluation of Language Models) leaderboards. It detailed several limitations that have hindered HELM's popularity, including feature completeness & fine tuning, price, latency, and platform update has hindered HELM popularity.

It was insightful to learn that '**Open**' does not always mean '**Open Source**'. While many models are simply 'open weight,' true open-source models offer a complete package: the weights, the training data, and the intermediate checkpoints necessary to replicate or extend the training process. Finally, the course touched upon reasoning models, noting how Meta's Llama remains opaque regarding its specific training tokens despite serving as the foundation for many Chinese LLMs. Similarly, while DeepSeek has shared its 'reasoning tokens,' this data remains insufficient to replicate their experiments or determine the optimal path for developing and sizing high-performance reasoning models.

Although the course was not code-intensive, it provided practical demonstrations and sample code for tokenization and embedding using the **tiktoken** and **Cohere** Python packages.