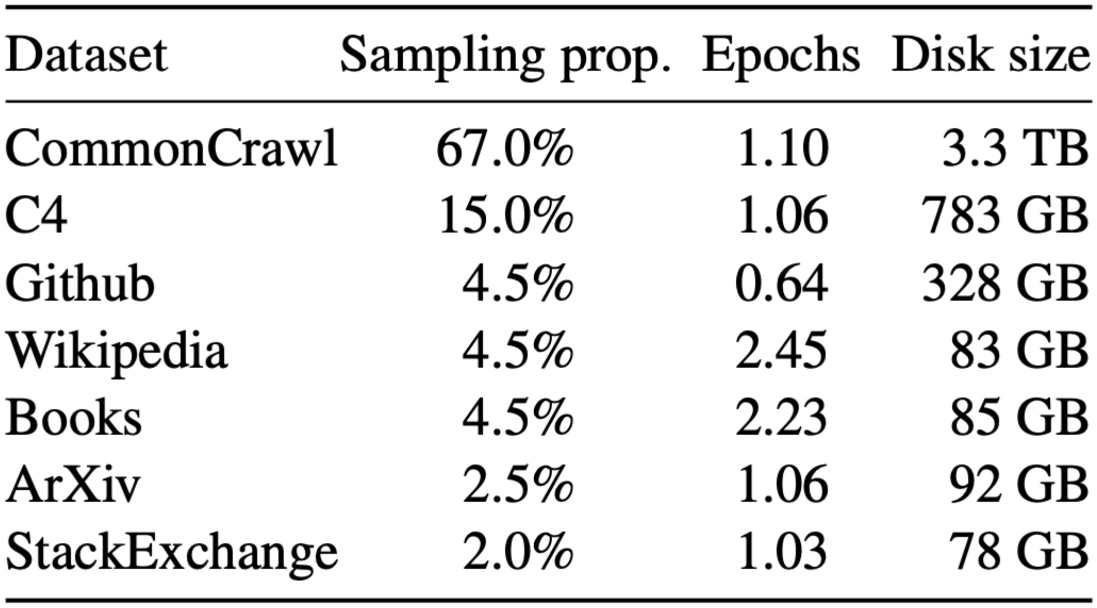
How a Large Language Model Works: The 5-Minute Explanation

How It Learns: Training on the Internet

An LLM learns its skills by being "trained" on an enormous amount of text data, a significant portion of the public internet, books, articles, and other text sources. This training process has one main objective:

1. **Input:** The model is given a piece of text with a word missing (e.g., "The cat sat on the ___").
2. **Prediction:** It makes a guess to fill in the blank.
3. **Correction:** It compares its guess to the actual word ("mat"). If it's wrong, it adjusts its internal parameters (millions of them, called **weights** and **biases**) to make it slightly more likely to guess "mat" in a similar context next time.

By repeating this process billions of times, the model builds an incredibly complex statistical understanding of how words, grammar, syntax, facts, and even reasoning styles are related.



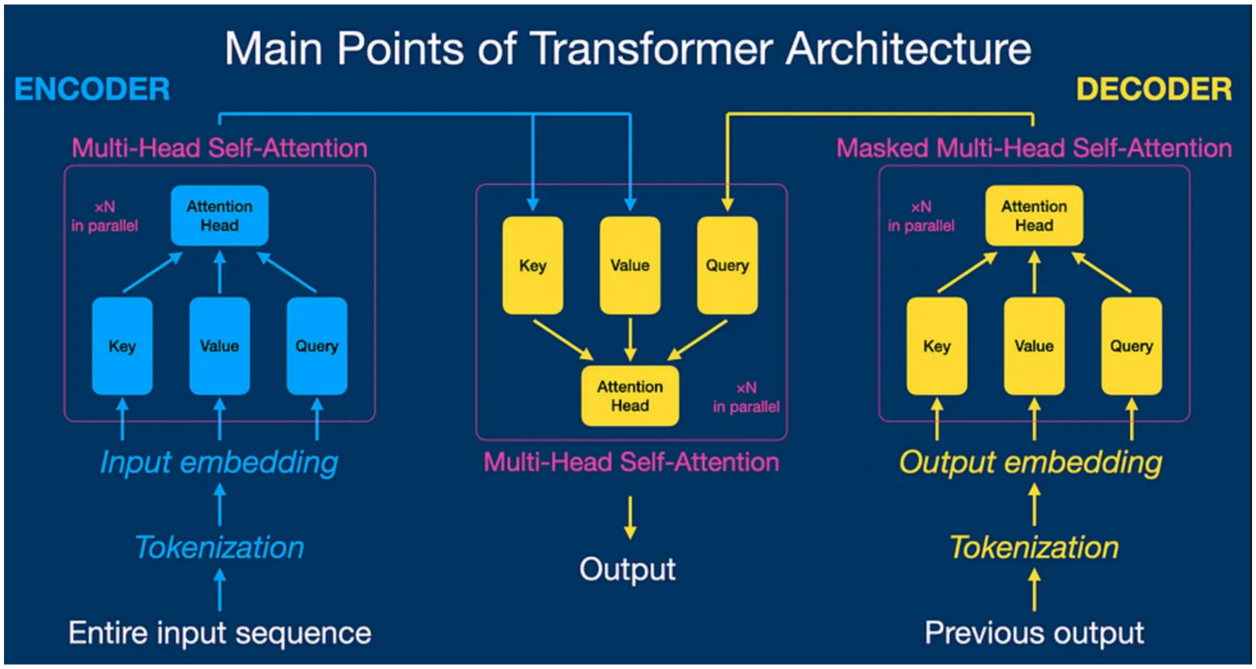
Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

The "Brain": The Transformer Architecture

The key technology that makes modern LLMs possible is the **Transformer architecture**. Its most important innovation is a mechanism called **attention**.

- **Attention Model:** It allows the model to weigh the importance of different words in the input text when it's deciding what to generate next. For example, in the sentence, "The delivery truck was late because **it** got a flat tire," the attention mechanism helps the model understand that "**it**" refers to the "**truck**," not the "**delivery**." This ability to track relationships between words, even far apart in a text, is crucial for generating coherent and contextually relevant responses.



How It "Thinks": From Prompt to Response

When you give an LLM a prompt, it doesn't "understand" it in a human sense. Instead, it does the following:

1. **Tokenization:** It breaks your prompt down into smaller pieces called **tokens** (which can be words or parts of words).
2. **Processing:** These tokens are converted into numbers and fed through its neural network. The "attention" mechanism identifies which parts of your prompt are most relevant.
3. **Prediction Loop:** The model then calculates the probability for every possible token that could come next. It picks one of the most likely ones, adds it to the sequence, and then feeds the *entire new sequence* back into itself to predict the *next* token.

It repeats this loop **predict, append, repeat** until it reaches a stopping point, generating a complete response one token at a time.