

Lecture 6: Stages of Building an LLM from Scratch

1. The Course Roadmap: Three Critical Stages

The lecturer outlines that the remainder of the course will be divided into three specific stages to build a functional LLM. This structure is heavily influenced by Sebastian Raschka's book, *Building a Large Language Model from Scratch*.

Stage 1: Building the Architecture (Data & Mechanism) Before training can begin, the fundamental building blocks must be coded from scratch.

- **Data Preparation:**
 - **Tokenization:** Breaking sentences down into individual units (tokens).
 - **Vector Embeddings:** Converting tokens into high-dimensional vectors so that semantic meaning is captured (e.g., ensuring "apple" and "banana" are mathematically closer than "apple" and "king").
 - **Positional Encoding:** Encoding the order of words, as the position in a sentence changes the meaning.
 - **Batching:** Creating data batches to feed the model efficiently during training.
- **The Attention Mechanism:**
 - Coding "Multi-head Attention" and "Masked Multi-head Attention" from scratch.
 - Understanding concepts like Key, Query, and Value.
- **LLM Architecture:** Stacking these layers to form the skeleton of the model.

Stage 2: Pre-training (Creating the Foundational Model) The goal of this stage is to train the architecture on a large, **unlabeled dataset** to create a "Foundational Model".

- **The Training Loop:** Implementing the code to calculate loss, compute gradients, and update parameters over multiple epochs.
- **Weight Management:** Writing functions to save and load model weights. This is crucial because training is expensive; saving weights allows you to pause and resume without starting from scratch.
- **Integration:** Loading pre-trained weights from OpenAI into the custom architecture to leverage existing powerful models.

Stage 3: Fine-tuning (Specialization & Applications) The goal of this stage is to refine the foundational model on a **labeled dataset** for specific real-world tasks. The course will focus on building two specific applications:

1. **Spam Classifier:** An email classification tool that determines if an email is "Spam" or "Not Spam" (e.g., distinguishing a lottery win scam from a dinner invite).
 2. **Personal Assistant / Chatbot:** A bot trained to answer specific queries based on instruction-response pairs.
-

2. Why Fine-tuning is Essential

The lecture emphasizes that while students often rely on pre-trained models (like generic ChatGPT), building **production-ready** applications requires fine-tuning.

- **The Limitation:** Pre-trained models provide generic answers.
 - **The Solution:** Companies (banks, airlines, educators) use fine-tuning to train the model on their private, labeled data to ensure accuracy and specific domain knowledge.
 - **Performance:** A fine-tuned model often outperforms a much larger pre-trained model on the specific task it was trained for.
-

3. Comprehensive Recap (Lectures 1–5)

The lecturer concludes the theory portion of the course with a summary of the five key takeaways from the previous lectures:

1. **LLMs Transformed NLP:**
 - Previously, NLP required separate models for translation, sentiment analysis, etc.
 - LLMs are **generalists**; one model can perform a wide range of tasks.
2. **The Two-Step Process:**
 - Modern LLMs are built via **Pre-training** (on massive unlabeled data/raw text) followed by **Fine-tuning** (on smaller, labeled datasets).
3. **The Secret Sauce (Transformers):**
 - The core mechanism is **Attention**, which allows the model to give weight to specific words across the entire input sequence, capturing long-range dependencies and context.
4. **Evolution of Architecture:**
 - The original 2017 Transformer had an **Encoder** and a **Decoder**.
 - The GPT family (2018 onwards) utilizes **only the Decoder**.
5. **Emergent Behavior:**
 - Although these models are technically trained *only* to predict the next word, they spontaneously develop capabilities they were not explicitly trained for, such as translation, coding, and summarization.

4. Next Steps

- From **Lecture 7**, the course shifts to **Hands-on Coding**.
- The immediate next topic will be "**Working with Text Data**," covering loading datasets and implementing tokenization in Python.