60 Days of LLM Development from Scratch

Day 2: Fine-Tuning vs. Pre-Training & Tokenization for LLM

Elias Hossain

Graduate Student, Mississippi State University

Email: elias.hossain191@gmail.com

On Day 2, we will focus on the differences between fine-tuning and pre-training and explore tokenization and preprocessing for training LLMs.

Fine-Tuning vs. Pre-Training: Theoretical Understanding

What is Pre-Training?

Pre-training is the process of training a language model on a large corpus of text data using self-supervised learning. The goal is to create a generalized model that understands language patterns, semantics, and syntax.

Key Characteristics of Pre-Training:

- ✓ Uses unsupervised/self-supervised learning (e.g., predicting masked words in BERT, next token prediction in GPT).
- ✓ Trained on massive datasets (e.g., Common Crawl, Wikipedia, BooksCorpus).
- ✓ Produces a foundational model that can be adapted for multiple NLP tasks.
- ✓ Requires high computational resources (TPUs, GPUs).

Examples of Pre-Trained LLMs:

- ✓ BERT (Bidirectional Encoder Representations from Transformers) – Trained on masked language modeling (MLM).
- ✓ GPT (Generative Pre-trained Transformer) – Trained on causal language modeling (CLM).
- ✓ T5 (Text-to-Text Transfer Transformer) – Trained for text generation and NLP tasks.

What is Fine-Tuning?

Fine-tuning is the process of adapting a pre-trained model to a specific NLP task (e.g., sentiment analysis, question-answering). Instead of training a model from scratch, fine-tuning modifies the pre-trained model's weights using labeled data.

Key Characteristics of Fine-Tuning:

- ✓ Uses supervised learning (with labeled task-specific datasets).
- ✓ Requires fewer resources than pre-training.
- ✓ Fine-tuned models inherit knowledge from pre-trained models but adapt to a domain-specific use case.
- ✓ Can be done using techniques like LoRA (Low-Rank Adaptation), Prefix Tuning, and Prompt Tuning.

Examples of Fine-Tuned Models:

- ✓ BERT fine-tuned for Named Entity Recognition (NER)
- ✓ GPT fine-tuned for Code Generation (Codex, GPT-4 Code Interpreter)
- ✓ T5 fine-tuned for Summarization (e.g., news summarization tasks)

When to Use Pre-Training vs. Fine-Tuning?

Scenario	Pre-Training?	Fine-Tuning?
You want a general-purpose LLM	✅ Yes	❌ No
You have a task-specific dataset (e.g., legal, medical)	❌ No	✅ Yes
You have a large corpus but no labeled data	✅ Yes	❌ No
You need a custom model for a business problem	❌ No	✅ Yes

Tokenization and Preprocessing for LLMs

Before training or fine-tuning a Large Language Model (LLM), the raw text data must be converted into a numerical format that the model can process. This involves tokenization, handling special tokens, padding, and truncation.

Understanding Tokenization

Tokenization is the process of breaking down text into smaller units (tokens) that a model can understand. Tokens can be words, subwords, or even individual characters.

Types of Tokenization

1. **Word-Level Tokenization**
   - Splits text into individual words.
   - **Example: "Large Language Models are powerful!"**

     → ["Large", "Language", "Models", "are", "powerful", "!"]
   - **Limitations:** Fails with **Out-of-Vocabulary (OOV)** words and cannot handle morphological variations effectively.

2. **Subword Tokenization** (Most Common in LLMs)

   Splits text into meaningful subwords to balance efficiency and vocabulary size such as:
   - **Byte Pair Encoding (BPE)** – Used in **GPT, OpenAI models**.
   - **WordPiece Tokenization** – Used in **BERT, RoBERTa**.
   - **Unigram Tokenization** – Used in **T5, mBART**.
   - **Example**: "Transformer" → ["Trans", "##former"]

     "Unbelievable" → ["Un", "##believ", "##able"]

3. **Character-Level Tokenization**

   Splits text into individual characters.
   - Example: "hello" → ["h", "e", "l", "l", "o"]
   - Used in: Low-resource languages, Speech-to-text and text-to-image applications.

Special Tokens in LLMs

Modern LLMs require **special tokens** to structure input properly. These tokens help the model understand text boundaries, separate input segments, and apply masking strategies.

Token	Description	Example
[CLS]	Start of the input sequence	BERT, RoBERTa
[SEP]	Separates two text segments	BERT, T5
[MASK]	Used for masked language modeling	BERT

[PAD]	Padding token for batch processing	GPT, BERT
<s> ... </s>	Sentence start and end tokens	GPT, T5

Example: Text: "Machine learning is amazing!"

Tokenized: ["[CLS]", "Machine", "learning", "is", "amazing", "!", "[SEP]"]

Preprocessing Steps for LLMS

Step 1: Text Cleaning and Normalization

Step 2: Tokenization

Step 3: Convert Tokens to IDs

Step 4: Padding & Truncation

Key Takeaways

- ✓ Tokenization converts text into numerical form for processing.
- ✓ Subword tokenization (BPE, WordPiece) is commonly used in LLMs.
- ✓ Special tokens ([CLS], [SEP], [MASK]) are essential for structured input.
- ✓ Padding and truncation ensure consistent input lengths for training.

Next Step: Implementing Tokenization in Code (Day 3 - Hands-on)