Lesson 2: What Are LLMs?

Type	@datasciencebrain

1. What is a Large Language Model?

A **Large Language Model (LLM)** is a type of machine learning model designed to understand and generate human-like text. It is trained on massive datasets of text — including books, articles, websites, code, and more.

The model's primary goal is **to predict the next word (or token)** given a sequence of previous words. That's it. But when done at scale — with billions of parameters and enormous datasets — this single ability enables surprisingly powerful applications.

Why "Large"?

The word “large” refers to:

- The **size of the dataset** the model is trained on
- The **number of parameters** (weights) in the model, often in the billions
- The **computational power** needed to train it

This scale allows the model to generalize across a wide range of topics, tasks, and domains.

---

2. How Does an LLM Work?

At the heart of modern LLMs is a neural network architecture called the **Transformer**, introduced in the paper *“Attention is All You Need”* (2017). This architecture completely replaced older models like RNNs and LSTMs because of its efficiency and scalability.

Let’s understand how a transformer enables an LLM to generate text.

Key Components of a Transformer

1. **Tokens**

   Text is broken down into smaller units called tokens (these may be words, parts of words, or even characters).

   Example:

   `"Hello world"` → `["Hello", "world"]` or `["Hel", "lo", "world"]`

2. **Embedding Layer**

   Each token is converted into a high-dimensional vector using an embedding table. This allows the model to work with numbers instead of raw text.

3. **Self-Attention**

   This is the core innovation. The model looks at **every token in the input** and learns **how much attention to pay to each one** when generating the next token.

   Example: In the sentence *“The dog chased the ball because it was fast”*, the model learns whether "it" refers to the dog or the ball by paying attention to context.

4. **Feedforward Layers**

   After attention, the model applies regular neural network layers to each token's representation to process further information.

5. **Stacking Layers**

   Transformers consist of many such attention + feedforward layers stacked together (often 12, 24, or more) to capture complex language patterns.

6. **Output Layer**

   Finally, the model computes a probability distribution over the vocabulary and picks the next most likely token (or word).

This entire process is repeated step-by-step for every word it generates, using the previous tokens as context.

---

3. Prompt and Completion

All LLM interaction follows this simple structure:

- You give it a **prompt** (some input text)
- It gives you a **completion** (generated output)

Example:

- Prompt: *"The capital of France is"*
- Completion: *"Paris"*

While it may look like the model is "answering" you, in reality it's doing nothing more than **predicting the most statistically likely continuation** based on training data.

LLMs don't truly "understand" in the human sense — they're extremely good at **pattern matching and prediction**, not reasoning or fact-checking.

---

4. LLMs Have Limitations

LLMs are powerful, but we must understand what they **can't** do on their own:

- They **don't access real-time data**

If you ask, “What’s the score in today’s match?”, it may give a guess or outdated answer.

- They **can’t run code**

  If you ask for a math operation, it may solve it incorrectly unless it’s seen similar examples.

- They **don’t have memory between sessions**

  Each prompt is treated independently. Unless you explicitly include prior context in the input, the model doesn’t remember.

- They **can’t use tools or APIs**

  If you want it to call an external service like Google Search or a weather API, it can’t. That’s where agents come in.

So although LLMs can appear smart, they’re still limited to generating text only.

---

5. LLM vs Agent

This is a critical distinction.

- A **Language Model** is a text prediction engine. It reads a prompt and generates a continuation.
- An **Agent** is a system built on top of a language model. It reasons about tasks, chooses what tool to use, and executes actions.

Example:

- Ask an LLM: *“Plot a graph of y = sin(x) from -10 to 10”*

  It may generate Python code but won’t actually run it.

- Ask an Agent:

  It uses the LLM to decide that plotting is required, selects a **PythonTool**, runs the code, and shows the output.

So:

> LLM = text generator

> Agent = decision-maker + LLM + tools

Agents are built by **giving the LLM access to tools** and asking it to reason about how to solve the problem using them.

---

6. Why Study LLMs Before Agents?

Before building intelligent agents that interact with the real world, we must fully understand how the underlying language model behaves.

Key reasons:

- To write better prompts
- To know what outputs to expect
- To anticipate model errors or limitations
- To make agents that are reliable and interpretable

You wouldn’t program a robot without first understanding the motor. In the same way, we don’t build agents without mastering the LLM that drives them.

Summary

- LLMs are trained to predict the next word in a sentence using massive text corpora.
- They are built using transformer models, which rely heavily on attention mechanisms.
- They only generate text — they don’t take actions or access external information.
- Agents extend LLMs by adding tools and decision logic.
- Understanding LLMs is foundational before working with agents, as they are the reasoning engine behind every decision an agent makes.
