Large Language Models (LLMs) - Backbone of AI Applications

Author: Shiven Saini
Portfolio: https://shiven.one

Some terms we should know to set up the context of this document

Machine Learning: Imagine a computer programmer writing an application with programming languages(traditional programming), the tasks are predefined. To complete or perform a certain action, the instructions are completely laid out in a format that the computer can understand.

But here is a catch, what if the environment and tasks to perform are not predictable. Like, if the user input is ambiguous or cannot be predicted at the time it was programmed.

Here comes the **ML Algorithms**, as the name suggests it is about *Machine & Learning*, we provide the computer or an algorithm a huge amount of examples and what the response should be like. The algorithm will learn patterns from these provided examples and be able to solve novel questions as well that *rely on same pattern*.

ML Analogy: Teaching a child to identify fruits

- **Traditional Algorithms (Rule-based):** Imagine you teach a child to identify an apple by giving explicit rules:
  - If the fruit is red, round, and about the size of your fist, then it is likely an apple.
  - If not, check if it's yellow and long—it might be a banana.

  Here, you (the programmer) set clear rules for the child to follow. The child can only identify fruits that fit these exact rules. If they see a green apple or a differently shaped apple, they might get confused.
- **Machine Learning Algorithms:** Instead, you show the child hundreds of pictures of apples and bananas, telling them which is which. Over time, the child starts to recognize apples and bananas—even if the apple is green or oddly shaped—by learning from examples. Here, the child develops their own *internal rules* based on patterns in the examples you provide, not from explicit instructions. They can handle unfamiliar cases better because they've learned from data.

> **Note:** As AI engineers, our primary focus is not on the inner workings or training processes of AI models, but rather on leveraging them to build impactful applications. However, having foundational knowledge of how these models operate can make the work more engaging and deepen our appreciation for the effort that goes into their development.

ChatGPT: This is *not an AI Model* but a product by the company OpenAI

Generative AI: An umbrella term for all the applications that leverages the AI Capabilities using API from different providers and building upon it.

Prompt: The instructions and input text you provide to the model. Prompting can have a significant impact on the quality of output from the LLMs.

Evolution of AI: From Research Labs to Mainstream Adoption

OpenAI began as a non-profit organization and, in many respects, continues to operate with a mission-driven approach. As pioneers in the artificial intelligence domain, OpenAI conducted significant research and development, training a variety of models and becoming one of the first organizations around the world to deliver advanced AI capabilities directly to end users.

While the field of artificial intelligence is not new—its origins can be traced back to research laboratories in the 1960s—access to AI was largely restricted to academic and research settings for decades. This landscape shifted dramatically in November 2022, when OpenAI launched the preview version of ChatGPT, powered by the GPT-3.5 model. ChatGPT quickly became a cultural phenomenon, setting a record as the fastest-growing consumer application in history by reaching one million users within its first five days of launch.

Following this groundbreaking success, numerous new platforms emerged, further advancing the field:

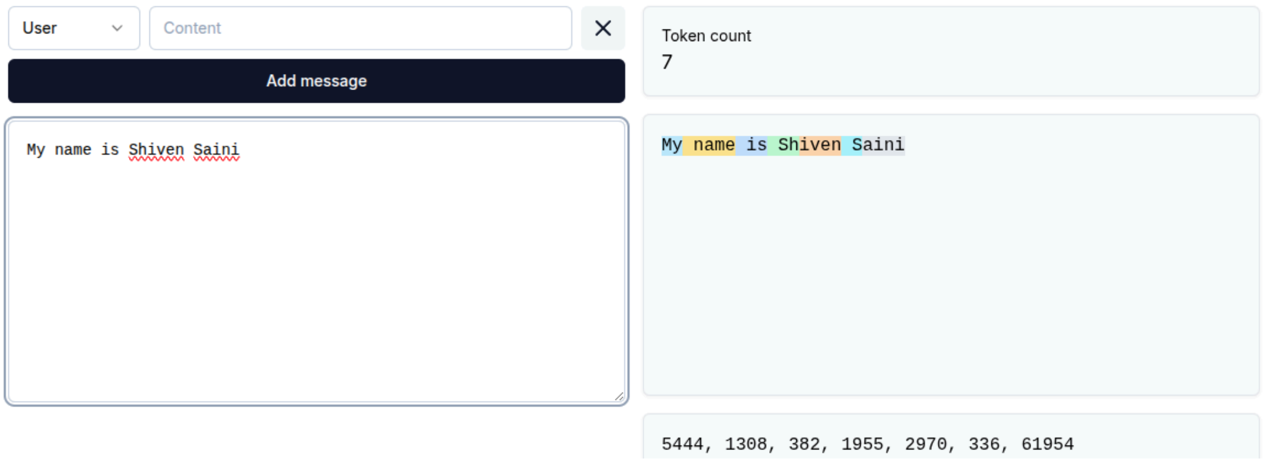
- **Anthropic**: Gained recognition for its state-of-the-art (SOTA) coding models, including the `Sonnet` and `Opus` series.
- **Google**: Released the `Gemini` models and the open-source `Gemma` series.
- **xAI**: Introduced the `Grok` family of models.
- **Deepseek:** Introduced cost-efficient SOTA models like `deepseek-v3` & `deepseek-r1`.

Since then, a wave of cutting-edge models—such as the reasoning-focused `01` and `03` series —have entered the market, each offering significant improvements in performance and capabilities.

Tokenizers & Tokens

You can think of tokens as the currency of AI models. AI models are essentially *mathematical functions* that can only process numbers, while humans communicate in text. To bridge this gap, **tokenizers** are used—they convert words, phrases, or characters into numerical representations called **tokens**.

- **Tokens** are not the same as words. For example, "ChatGPT" might be a single token, while "artificial intelligence" could be split into several tokens. On average, a token represents about 0.7 words in English text. **1 Token ~ 0.7 Word(On Average)**
- **Tokenizers** are special algorithms that break down text into these tokens according to specific rules, often based on how commonly certain sequences appear in language. Common words might be a single token, while rare or invented words are split into smaller pieces.
- **Example:**



7

```
My name is Shiven Saini
```

```
My name is Shiven Saini
```

```
5444, 1308, 382, 1955, 2970, 336, 61954
```

Even if the total word count is 5, it is still split up into 7 tokens.

> **Note:** A word is not always a token. The number of tokens depends on the tokenizer and the language. Tokens can be as short as a single character or as long as an entire word.

Why Does This Matter?

- **Efficiency:** Tokenizing text allows models to process information more efficiently and handle different languages, special characters, or even code.
- **Limits:** Most AI models have a token limit per request (for example, 4,096 tokens), which affects how much text you can input or generate at a time.
- **Costs:** Many AI APIs and tools charge based on the number of tokens processed rather than words or characters.

You can experiment with how text is broken into tokens using this tool: tiktokenizer.vercel.app

Brief primer on LLMs

In layman's terms, LLMs are trained algorithms that receive text input and predict and generate human like text output(Literally, that's it). Essentially, they behave like the familiar autocomplete feature found on our smartphones keyboard (try writing a sentence on WhatsApp using Gboard), but taken to an extreme level.

Large refers to the size of these models in terms of training data and parameters used during the learning process. For example, OpenAI's `GPT-3` model contains 175 billion *parameters*, which were learned from training on *45 terabytes* of text data.

Parameters in a neural network model are made up of the numbers that control the output of each *neuron* and the relative weight of its connections with its neighbouring neurons. Trying to mimick how our own Human brain works. However, this is really an oversimplification of concepts. But should suffice to set up the base for further exploration.

Language model refers to a computer algorithm trained to receive written text (in English or other languages) and produce output also as written text (in the same language or a different one). These are *neural networks*, a type of ML model which resembles a stylized conception of the human brain, with the final output resulting from the combination of the individual outputs of many simple mathematical functions, called *neurons*, and their interconnections. If many of these neurons are organized in specific ways, with the right training process and the right training data, this produces a model that is capable of interpreting the meaning of individual words and sentences, which makes it possible to use them for generating plausible, readable, written text.

> 🤔 Note: I know this sound a lot of technical jargon. But keep up with me, it's just a fancy way to say that they just receive a text input and then predict the new text part based on the training patterns they learned.

> 💡Fun Fact: Researchers have discovered that large language models (LLMs) can transfer their semantic understanding across languages. This means that even if an LLM is primarily trained on data in one language, such as English, it can still perform reasonably well in other languages—provided it understands their grammar. Remarkably, this capability emerges even when the model has not been exposed to text examples in those other languages during training.

LLMs can perform various text based tasks including text generation, summarization, translation, sentiment-analysis, classification and much more; which makes them much more versatile.

Example of Text Generation :-
Prompt: The capital of India is ___________

Answer: The capital of India is New Delhi.

It tries to relate with words to connect the context of each word, like it knows about Capital, but capital of what? The Parsing the text semantics , it is realizing that the model is being asked to tell the capital of India.

The driving engine behind LLMs' predictive power is known as the *transformer neural network architecture*. The transformer architecture enables models to handle sequences of data, such as sentences or lines of code, and make predictions about the likeliest next word(s) in the sequence. Transformers are designed to understand the context of each word in a sentence by considering it in relation to every other word. This allows the model to build a comprehensive understanding of the meaning of a sentence, paragraph, and so on (in other words, a sequence of words) as the joint meaning of its parts in relation to each other.

I can never capture the beauty of how these models work intrinsically. Following are some ground-breaking research papers that played a key role in the development of Transformer architecture and LLMs in general.

1. Attention is all you need : Explains the *Transformer* architecture.
2. Language Models are Few Shot Learners: Explains the Prompting techniques

Modern LLMs come in different forms, each building on the previous version to become more helpful and user-friendly. Here's a breakdown in simple language:

---

1. Base (Pretrained) LLMs

What are they?
These are language models trained on huge amounts of text from the internet, books, articles, code, and more. They learn by predicting missing or next words in sentences, but are not specifically trained to answer questions or follow instructions.

How do they learn?

- **Predict the next word:**
  - Example:
    - Input: "The capital of India is ___"
    - Expected Output: "New Delhi"
- **Predict a missing word:**
  - Example:

- Input: "The ___ of India is New Delhi"
- Expected Output: "capital"

Limitation:
They work best if you prompt them in a way similar to their training ("The capital of India is"), but may struggle with direct questions ("What is the capital of India?").

---

2. Instruction-Tuned LLMs

What are they?
These models start as base LLMs, but get additional training on question-answer pairs made by humans technically called *Supervised Fine-tuning*. This teaches them to respond better to direct questions or instructions.

How do they learn?

- ⭐**Task-specific datasets:**
  - Example:
    - Q: "What is the capital of India?"
    - A: "The capital of India is New Delhi."
- **Reinforcement Learning from Human Feedback (RLHF):**
  - Example:
    - If users prefer one answer over another, the model learns to favor the preferred style.

Benefit:
Now you can ask more natural questions and get helpful responses.

> **Note:** On platforms such as HuggingFace, instruction-tuned LLMs are typically identified by the word "instruct" in their model names.

---

3. Dialogue-Tuned LLMs

What are they?
These are instruction-tuned models further optimized for conversations and chat.

How do they learn?

- **Dialogue datasets:**
  - Example:
    - User: "What is the capital of India?"
    - Assistant: "The capital of India is New Delhi."
    - User: "And what about France?"
    - Assistant: "The capital of France is Paris."
- **Chat format structure:**
  - The conversation is organized by roles:
    - **System:** Gives instructions or context (e.g., “You are a helpful assistant.”)
    - **User:** Asks questions.
    - **Assistant:** Responds.

Benefit:
These models handle multi-turn conversations and can keep track of context, making them perfect for chatbots.

> Note: We will discuss about the concept of roles at a later time.

---

4. Fine-Tuned LLMs

What are they?
These are LLMs specifically trained further for a particular task or domain by a developer or company.

How do they learn?

- They are trained on specialized datasets, like financial reports, medical texts, or customer support conversations.

Example:

- A company fine-tunes an LLM to:
  - Extract the sentiment from reviews.
  - Identify risk factors in business reports.
  - Summarize legal contracts.

Benefit & Trade-off:
They become very good at the chosen task, but may lose some ability to answer unrelated questions.

Summary Table

Model Type	Main Training	Best at	Example Prompt	Example Output
Base (Pretrained)	Huge text datasets	Predicting next/missing words	"The capital of India is ___"	"New Delhi"
Instruction-Tuned	Q&A pairs, RLHF	Following instructions	"What is the capital of India?"	"The capital of India is New Delhi."
Dialogue-Tuned	Chat datasets	Multi-turn conversations	"and what about France?"	"The capital of France is Paris." (and remembers follow-ups)
Fine-Tuned	Custom datasets	Specific tasks/domains	"Extract all risks from this report."	"Risks: 1. Market volatility..."

In short:

- **Base LLMs** are like students who read a lot but haven't practiced answering questions.
- **Instruction-tuned LLMs** are those students who have practiced answering test questions.
- **Dialogue-tuned LLMs** are those who are great at having a conversation and remembering context.
- **Fine-tuned LLMs** are specialists trained for a particular job.

> **Note:** Throughout these notes, when I refer to "LLMs," I am specifically referring to instruction-tuned LLMs, as these are the most practically useful. Similarly, when I mention a "chat model," I am by default referring to a dialogue-tuned model.

Basic terms used in prompting techniques

1. Zero-shot prompting:
You give the model a task or question without providing any examples. You rely entirely on the

model's existing knowledge and abilities.
Example: "Translate 'hello' to French."

2. One-shot prompting:
You provide one example along with your prompt. This helps guide the model with a single sample of the format or task you want.
Example:
English: "cat" → French: "chat"
English: "dog" → French: ?

3. Few-shot prompting:
You provide a few (typically 2-5) examples to show the model how to perform the task. This helps the model understand the pattern or style you expect.
Example:
English: "cat" → French: "chat"
English: "dog" → French: "chien"
English: "bird" → French: ?

4. Chain-of-thought prompting:
You encourage the model to "think aloud" and show its reasoning process step by step before giving a final answer.
Example:
Question: "If Alice has 3 apples and gives 1 to Bob, how many does she have left?"
Let's think step by step. First, Alice starts with 3 apples. She gives 1 to Bob, so she has 2 apples left.
Final answer: 2 apples.

Few-shot and multi-shot prompting help models give more accurate, relevant, and formatted answers by providing clear examples to follow. This is especially helpful when the task is new or ambiguous.

> 🤷 Fun Fact: Large language models (LLMs) with chain-of-thought (CoT) prompting can perform worse on tasks where humans also perform worse with explicit thinking.

You can read the research paper if you are interested here:
[Mind Your Step (by Step): Chain-of-Thought can Reduce Performance on Tasks where Thinking Makes Humans Worse]