About LLM

JunHyeok Choi

Index

Introduction

In late 2022, OpenAI's ChatGPT changed the world almost overnight. It suddenly appeared, showing all of us the possibility of the AI we'd only seen in science fiction. And, contrary to what many expected, it is even threatening our future jobs.

Despite this recent surge into the mainstream, this form of AI has already become one of humanity's most powerful tools. And yet, for many of us, these LLMs remain a mystery.

Let's think about it. What allows this mass of code, the LLM, to speak as fluently as (or even more fluently than) a human, debug code, and answer complex questions? But when we ask these questions, the answers that come back are often just a stream of technical terms we can't understand. This article is a guide for anyone who has these same questions.

First, we will trace the history of LLMs, from World War II to the present day. Then, we will dive into the fundamental concepts behind them. Finally, we will learn about key modern technologies like QLoRA and RAG that build on those foundations.

Before you read on, please note that this article is intended to give people a general understanding of LLMs, so it simplifies complex topics and is not intended to be a deeply technical reference. Furthermore, I wrote this for my own study, so some parts might be incorrect.

Thank you for reading my humble article.

History of LLM

1. Dreaming concept of AI

The history of Large Language Models (LLMs) began much earlier than most people expect. During World War II, **Alan Turing** developed one of the first computers. However, at the initial stage, computers were merely giant calculators used for military purposes such as calculating ballistics and deciphering codes.

Ironically, the war accelerated the development of computer technology. Later, **John von Neumann** laid the foundation of the modern computer with the **Von Neumann Architecture**, which allowed faster progress in computer science.

By the 1950s, people began to imagine machines that could think — the early concept of Artificial Intelligence (AI). This idea aligned perfectly with government interests, as they wanted machines capable of translating Russian documents and identifying objects from images. From this point, the initial ideas of **language models** and **image recognition AI** were born.

2. Initial research and AI winter

After government started to invest money on AI, tons of research were started. In this time (1950 ~ 1970), the fundamental concepts of modern AI were made. For example, the concept of **artificial neural network** (**Perceptron** model, Frank Rosenblatt, 1958) and **NLP** (Natural Language Processing) were born. Surprisingly, researchers made a chat bot named **ELIZA** in 1966.

Even though human made huge progress with AI, after 1970, people realized Perceptron model (early artificial neural network) has huge limitations. Because of this realization, AI winter came. The main reasons of this phenomenon were **lack of computing power, lack of data and high cost**.

3. Rerise of AI

Although there was language model like Statistical Language Models in the 1990s, AI didn’t get much attention. Amid public indifference, only Canada invested in AI. From the University of Toronto, the "Godfather of AI", Geoffrey Hinton emerged.

During the AI winter, he developed the **Backpropagation Algorithm** for **multi-layer networks**, laying the fundamentals of deep learning. In 2006, he and his colleagues introduced the **Deep Belief Network (DBN)**, a probabilistic generative model inspired by **how the human brain processes** information and proved deep learning can be done in the real world. This announcement shocked a lot of people and resulted resuming research on AI. His work on neural networks and distributed representations became the critical foundation for future breakthroughs like **Word2Vec** and the **Transformer** architecture.

4. Renaissance of AI

After Geoffrey Hinton made DBN, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton won the **ImageNet** Large Scale Visual Recognition Challenge (ILSVRC) with **AlexNet** in 2012. The **AlexNet**, a **deep convolutional neural network (CNN) architecture**, shocked the world with an error rate of only 15.3%. This was a breakthrough in object recognition and proved that deep learning could outperform traditional machine learning methods.

There were three main reasons for this revolution: **parallel computing with GPUs**, **the availability of massive amounts of data on the Internet**, and **rapid advances in computing technology**. After this success, **deep learning became the standard approach** in AI research, leading to major progress in fields like speech recognition, natural language processing, and autonomous vehicles.

5. Fetal movement of LLM

Despite great progress in image recognition, AI still struggled with language understanding. But emergence of **Word2Vec in 2013** and **Transformer in 2017** paved the silk road of LLM.

Around 2013, Google researchers introduced **Word2Vec**, a model can **represent word's meaning into vector** and they called this **embedding**. Since the concept of the **embedding is change words into series of numbers**, people can teach AI natural languages easily.

Later, in 2017, Google also introduced **Transformer architecture**. The Transformer model revolutionized natural language processing by **allowing AI to understand relationships between words across entire sentences**.

These two innovations—embeddings and transformers—became the backbone of modern **Large Language Models (LLMs)**, enabling machines to understand and generate human-like language.

6. The Rise of LLMs

Building upon the deep learning foundations that researchers like Geoffrey Hinton helped establish, the development of large language models marked a major technological leap.

In 2018, **GPT-1** introduced the **concept of pretraining on massive text corpora** followed by task-specific fine-tuning, showing that general linguistic knowledge could be transferred to various applications.

The following year, **GPT-2**, with 1.5 billion parameters, **demonstrated remarkable fluency and contextual understanding**, sparking global interest in the potential of generative AI.

By 2020, as everybody knows, **GPT-3** was introduced. Unlike its brothers and sisters, it caught international attention and represented a clear glimpse into the future. GPT-3 scaled up to 175 billion parameters and achieved **few-shot learning**, enabling the model to adapt to new tasks simply through prompts rather than retraining, and finally opened the current era—the **era of Large Language Models**.

Foundation of LLM

1. The backbone of LLM
   a. Machine Learning
   b. Deep Learning
   c. Artificial Neural Network, ANN

2. LLM architecture
   a. Transformer
      i. Self-Attention
      ii. Positional Encoding
      iii. Softmax

3. LLM training pipeline
   a. Tokenization
   b. Embedding
   c. Pre-training
   d. Fine-tuning
   e. RLHF

1. The backbone of LLM

a. Machine Learning

The traditional way to solve real world problems is making rule with code. However, for many complex problems (like recognizing a cat or understanding language), it's nearly **impossible to write all the rules**. These problems required a more flexible approach. To fix this, scientists came up with an idea, **Machine Learning (ML)**.

The concept of ML is “If we show lots of Data (ex. Image of cat) to computer, it might be able to find the pattern and learn it”. For example, to create a spam filter, you wouldn't write thousands of rules for every spam word. Instead, you would show an ML model 100,000 examples of spam emails and 100,000 examples of normal emails. The model ***learns*** the patterns that separate one from the other.

This is the "learning" in "Large Language Model." We don't teach an LLM the rules of grammar. We feed it a massive dataset (like every English book human have ever written), and it learns the patterns, context, and structure of English all by itself.

b. Deep Learning

Then what is Deep Learning? It is simply one of types of the Machine Learning. The major difference between ML and Deep Learning is how it learns. Unlike traditional ML, Deep Learning uses **Artificial Neural Network (ANN)**. (it will be explained in the next section)

The **"Deep"** in Deep Learning simply means the ANN has **tons of layers** stacked on top of each other—sometimes hundreds or thousands deep. This deep structure allows the model to learn incredibly complex patterns, like the structure of language or recognizing objects in images. And because this structure is so complex (with billions of parameters), it **requires massive amounts of data** to be trained effectively.

All modern LLMs are Deep Learning models.

c. Artificial Neural Network, ANN

The ANN is the "how" behind Deep Learning, and it is **directly inspired by the human brain.**

Think of your brain: it has billions of **neurons** that are **connected** to each other. When you learn a new skill (ex. riding a bike), the strength of the connections between specific neurons changes.

An ANN mimics this exact mechanism. It is built from a few key components:

1. **Nodes (or "Neurons"):** Small computational units, grouped into layers.
2. **Layers:**
    1. **Input Layer:** Where the network receives information (e.g., the pixels of an image).
    2. **Hidden Layers:** These layers process the data from the input layer. The more hidden layers there are, the "Deeper" the network is, and the more complex patterns it can learn.
    3. **Output Layer:** Where the final decision or prediction is made (e.g., "Cat" or "Dog").
3. **Weights (or "Parameters"):** These are the **connections between the nodes**. Each connection has a "weight," which is just a number (ex. 0.8, -0.2) representing its **strength**.

The "learning" is all about adjusting these weights.

This connects everything: A **Deep Learning** model is just an ANN with many Hidden Layers. And an **LLM** is a Deep Learning model with *billions* of these **weights/parameters**.

2. LLM architecture

a. Transformer

The Transformer was introduced in a famous 2017 paper from Google titled **"Attention Is All You Need."** Before the Transformer, models (like RNNs/LSTMs) had a huge problem: they processed sentences **word-by-word in sequence**. This was slow, and they often "forgot" what was said at the beginning of a long sentence.

The Transformer revolutionized this by processing **all words in a sentence at the same time.**

i. Self-Attention: Context

This is the key feature of the Transformer and the “attention” in the paper’s title. **Self-Attention** is a mechanism that allows every single word in a sentence to "look at" and "pay attention to" every other word in that same sentence. By doing this, it learns which words are most important to each other.

- **Example:** "The cat sat on the mat, as **it** was sleepy."
- **How it works:** Self-Attention learns to connect the word "**it**" most strongly to "**the cat**," not "the mat." If the sentence were "...as **it** was dirty," the model would learn to connect "**it**" to "**the mat**."

This is how LLMs understand deep context and ambiguity.

ii. Positional Encoding (The "Order" Mechanism)

Self-Attention has one major flaw: by looking at all words at once, it has **no idea what order they are in**. To a "pure" attention model, "The dog chased the cat" and "The cat chased the dog" would look identical.

Positional Encoding solves this.

- **How it works:** Before the words are fed into the Transformer, a small piece of mathematical information (a vector) is "stamped" onto each word. This stamp explicitly tells the model the position of the word (e.g., "you are word #1," "you are word #2," etc.).
- **Analogy:** It's like adding a unique timestamp or GPS coordinate to each word. This gives the model the sense of sequence and grammar that it otherwise lacks.

iii. Softmax (The "Decision" Function)

Softmax is a mathematical function used to **make a clear choice**. It's used in two key places, but its most important job is at the very end of the LLM.

- **How it works:** When an LLM is about to generate the next word, it first calculates a "score" (called a *logit*) for every possible word in its vocabulary. (e.g., "apple": 2.1, "banana": 0.5, "cat": 9.8, "dog": 9.6).
- **Softmax's Job:** It takes this messy list of scores and converts them into clean **probabilities** that all add up to 100%.
- **Example:** "apple": 1%, "banana": 0%, "**cat**": 51%, "dog": 48%.

This allows the model to make a clear, probabilistic decision about which word to output next. It's the final "decision-maker" that turns the model's "thoughts" into an actual word.

3. LLM training pipeline

If the LLM Architecture is the "blueprint," the **training pipeline** is the step-by-step factory assembly line that builds the model. This is how we take a "dumb" network and turn it into a smart assistant.

a. Tokenization (Step 1: Preparing the Raw Materials)

The first problem is that computers don't read words; they only read numbers. Tokenization is the process of breaking down raw text into smaller, manageable pieces called "tokens."

- Example: The sentence "LLMs are powerful"
- Might become: ["LL", "Ms", " are", " powerful"]

Each of these unique tokens is then assigned an ID number (e.g., [5012, 621, 389, 11978]). This is the only format the machine can understand.

b. Embedding (Step 2: Giving the Numbers Meaning)

Now we have number IDs, but they are just labels. The number 5012 has no meaning. So, we need to convert this with **Embedding** to add meaning.

An **Embedding** layer converts each token ID into a long list of numbers (a **vector**). This vector isn't just a label; it represents the token's meaning, context, and relationship to other words.

This is the rich, mathematical "language" that the Transformer architecture processes.

c. Pre-training (Step 3: The "General Education")

This is the most expensive, time-consuming, and **"Large"** part of "Large Language Model."

Pre-training is where the model gets its 'general knowledge' of the world. We feed the model trillions of tokens (from the entire internet, books, etc.) and give it one simple task over and over: **"Guess the next word."**

- **Example:** Given "The cat sat on the...", the model must learn to predict "mat."

By doing this billions of times, the model is forced to learn everything about human language: grammar, facts, common sense, reasoning, and even bias. The result is a **"base model".** It's incredibly knowledgeable but not yet specialized for any useful task.

d. Fine-tuning (Step 4: The "Specialized Job Training")

The base model is a "jack-of-all-trades" but a master of none. It knows about language, but it doesn't know how to be a helpful chatbot.

Fine-tuning is the 'specialized training.' We take the pre-trained base model and train it again, but this time on a much smaller, high-quality dataset designed for a specific task.

- **Example:** For a chatbot, this dataset would be thousands of high-quality, "Question-and-Answer" pairs.
- **Result:** This step teaches the model how to behave, how to follow instructions, and how to use its knowledge in a helpful way. (Note: **QLoRA** is an efficient method for doing this fine-tuning.)

e. RLHF (Step 5: The "Polishing and Safety" Step)

This is the final "polishing" step that made ChatGPT's behavior so impressive. **RLHF (Reinforcement Learning from Human Feedback)** makes the model safer and more aligned with what humans prefer.

1. The model generates a few different answers to a prompt (e.g., Answer A, B, C).
2. A **human rater** looks at the answers and ranks them from best to worst.
3. This human feedback is used to train a separate "Reward Model" (a small AI "judge").
4. This Reward Model then trains the main LLM, acting as a 'carrot and stick' to teach it to produce answers that humans would rate highly (i.e., be helpful, honest, and harmless).

This entire pipeline—**from Tokenization to RLHF**—is how we get the powerful, usable AI assistants we have today.

Modern LLM Technologies (QLoRA, RAG)

1. Background

After the emergence of GPT-3, the focus of AI research shifted from simply increasing model size to **improving efficiency, adaptability, and accessibility**.

While large-scale models showed impressive capabilities, their **enormous computational cost and data requirements limited real-world use**.

To overcome these challenges, researchers began developing techniques that make large language models more efficient and specialized. Among these innovations, **QLoRA (Quantized Low-Rank Adaptation)** and **RAG (Retrieval-Augmented Generation)** stand out as two major advancements that revolutionized how modern LLMs are trained and utilized.

2. QLoRA : Efficient Fine-tuning Method

Traditionally, fine-tuning a large model required enormous GPU memory, making it nearly impossible for individuals or small institutions to adapt LLMs for specific tasks.

QLoRA, introduced in 2023, solved this problem by combining **quantization** (compressing model weights into lower precision formats) with **LoRA** (Low-Rank Adaptation), which trains only a small number of additional parameters instead of updating the entire model.

Because this method is far more cost-efficient than full-scale fine-tuning, QLoRA has quickly become a **standard approach** for adapting large models.

In particular, the **combination of QLoRA and RAG** (Retrieval-Augmented Generation) is now widely used in **agent-based AI systems**, enabling them to learn efficiently while maintaining access to external knowledge sources.

If you want to see details → https://ar5iv.labs.arxiv.org/html/2305.14314

3. RAG: For Accuracy

One of the biggest limitations of standard LLMs is that their knowledge is "stopped" in time, **limited to the data they were trained on**. This leads to two major problems: **hallucinations** and an **inability to access current or private data.**

To solve this accuracy problem, **RAG (Retrieval-Augmented Generation)** was introduced by Facebook AI Research in 2020. The concept of RAG can be likened to giving the AI an **open-book test**.

When an LLM with RAG receives a question, it doesn't just "make up" an answer from its trained memory. Instead, it follows a simple 2-step process:

i. **Retrieve:** First, the system searches an external knowledge source—like a company's internal documents, the latest news, or a specific database—for **facts related to the question**.

ii. **Augment & Generate:** Next, it passes these retrieved "grounding data" to the LLM as a reference. The LLM then **generates an answer based *on* these facts**.

This method dramatically improves accuracy by **"grounding"** the AI's answer in external, verifiable facts.

If you want to see details → https://arxiv.org/abs/2005.11401

Concluding the article…….

I sincerely hope this article has helped answer your questions or sparked your interest in LLMs and AI.

The world of LLMs is vast, so this article certainly couldn't cover everything. If you have any questions while reading, please feel free to ask. I'll gladly answer them to the best of my knowledge.

If, like me, or after reading this, you feel intrigued by LLMs and want to try developing with them, I'd like to carefully recommend joining my project. A friend and I are currently working on a project using **local LLMs** and are attempting to build our own implementations of **QLoRA and RAG**, which I mentioned earlier. You can also gain hands-on experience with **cloud computing using GCP**.

I can't say our project will be easy. However, I believe that as computer science students—that is, as engineers—**our job is to bring these complex concepts to life**.

Thank you once again for reading my humble article.

Special thanks to Gemini