What are LLM’s?

A large language model (LLM) refers to a type of artificial intelligence model specialized in processing and then generating human-like responses based on it’s training data, parameters, and the input it receives. But what does that really mean? Let’s find out!

Imagine a vast, expansive library with billions and billions of books, articles, and documents. These are the countless pieces of training data from human history that our LLM aka the librarian that has never ventured outside the walls of this vast library has learned from. And whenever you ask a question or seek information aka write a prompt, a specialized librarian (the LLM) scours through the shelves, pulling out relevant snippets from various books found in it’s library (training data) and crafts a coherent and informative response to your question. However, the librarian doesn't have personal experiences or opinions outside the library, or the ability to fully understand the text like a human would. It can only provide responses based on the patterns and information present in the texts it has read. So LLMs learn by identifying patterns in the training data. For instance, they learn the structure of sentences, common phrases, the relationship between words and contexts, etc. They can then use this learned pattern recognition to generate text that only mimics human-like language.

Imagine a skilled musician who has extensively studied and memorized countless pieces of music from inside this library but doesn't have the mental ability to understand the theory behind the music or have the awareness to fully comprehend it the way a human would. When asked to compose a new piece of music, the musician cleverly rearranges and combines snippets from the memorized pieces to create something that sounds original and harmonious to listeners. It can sound amazing and be correctly produced but it is important to note that it is a remix of what has been studied and memorized from the past. Similarly, an LLM, having analyzed and stored patterns from vast amounts of data, rearranges and combines these learned snippets to generate new, coherent outputs that mimics human-like language, even though it doesn't truly understand the meaning behind the words and library it’s been trained on.

Like a librarian who has read extensively yet lacks personal experience, the LLM can provide detailed information on a wide array of topics, yet its understanding is confined to the patterns learned from the text within the library's walls.

So what makes LLM's so impactful in today's world?

- Because of their ability to process and generate human-like responses on an unprecedented scale, this makes the interactions between humans and machines much more natural, intuitive, and effective. Imagine in our library metaphor, not having a librarian at your fingertips. The result would be billions of pieces of information without an effective or intuitive way to interact with it.

How ChatGPT was Created

GPT stands for "Generative Pre-trained Transformer." It reflects the model's nature and training process: "Generative" signifies its ability to generate text, "Pre-trained" indicates that it is trained on a large corpus of text data before fine-tuning on specific tasks, and "Transformer" denotes the underlying architecture used for training the model.

Creating ChatGPT entailed a multifaceted process starting with data collection from diverse sources, which is crucial for training robust Large Language Models like GPT-3 or GPT-4. The data undergoes preprocessing, including tokenization before the base model training on a Transformer architecture, which is adept at handling sequential data for language modeling tasks. Post-training, the model is fine-tuned on specific tasks using supervised learning, evaluated for performance, deployed for user accessibility, and continuously monitored and updated to ensure its accuracy, safety, and relevance over time. Here's a high-level overview:

Data Collection:

- **How**: Which is done by scraping and aggregating text from various sources like books, articles, and websites, a vast dataset is compiled. GPT-3 was trained on several datasets, including the Common Crawl dataset, which is a compilation web-based of data collected from the internet. The Common Crawl dataset contains petabytes (equivalent to 1,000,000 gigs) of data collected over 12 years of web crawling. The corpus contains raw web page data, metadata, and text extracts.
- A web crawler is a computer program that's used to search and automatically index website content and other information over the

internet. These programs, or bots, are most commonly used to create entries for a search engine index.
- It was also likely trained on some proprietary and licensed datasets that was collected and organized by a third-party provider.
- This data is essential for training Large Language Models like GPT-3 or GPT-4. For GPT-3, 45 terabytes of text data were collected from various sources, 1 terabyte is equivalent to 83 million pages of text. An absurdly higher amount of data was used for GPT-4, so you do the math. This was then preprocessed down to 570GB of high-quality, usable data.
- **Why**: Data is the foundation of any machine learning model. It provides the raw material from which models learn patterns and make predictions or decisions. A diverse and large dataset is crucial for training robust and capable models.

Data Preprocessing:
- **How**: This is done to clean and organize the data collected and ensure consistency and remove any irrelevant or sensitive information. An element of this is called tokenization, which involves breaking down text into smaller pieces called tokens. This process helps to convert the raw text into a format that can be used by the model for training in an optimized fashion.
- **Why**: It aids the removal of any personally identifiable information to adhere to privacy standards and helps in reducing the noise in the data which may affect the model's performance.

Base Model Training:
- **How**: Training a Transformer-based language model (like GPT-3 or GPT-4) on the collected and preprocessed data using a form of unsupervised learning is known as language modeling. This involves adjusting the model's parameters to minimize the difference between the predicted output and actual next word/token that's output in a sequence. "Don't worry, we'll have an entire lecture on ChatGPT's predictive ability to choose it's next word while writing a given output and we'll also learn exactly what Transformer Architecture is.
- **Why**: This base model training step aims to take the data it's been trained on and deliver quality outputs by adjusting the model's parameters to minimize prediction error.

Fine-Tuning:

- **How**: At this stage, we have a well-performing tool but it lacks the nuances of human subtleties and characteristics. To get a product that is ready to be used by the public, a smaller, more specific dataset with supervised learning, to guide the model towards desired behavior, such as a better writing format to present detailed information, better accuracy with word choice, or dialing in it's parameters for safer responses.
- Supervised learning is similar to a teacher grading a students work. The student outputs a written paper and the teacher then gives a value based on how correct and properly the model performed.
- **Why**: Fine-tuning tailors the base model to perform well on specific tasks and aligns it with certain safety and ethical guidelines. Fine-tuning also helps in aligning the model's outputs with human values and improving its safety by reducing harmful or biased outputs.

Evaluation and Iteration:

- **How**: Various metrics and feedback from human evaluators are used to assess the model's performance, and iterative fine-tuning and evaluations are performed for continuous improvement.
- **Why**: Evaluation measures the model's performance and identifies areas for improvement, ensuring the model meets desired standards and real-world expectations. This stage goes on and on until the model is safely ready for public use.

Deployment and Updating:

- **How**: Feedback from users and monitoring systems are collected and analyzed to identify issues, and the model is updated and re-trained as necessary.
- Continuous monitoring helps in identifying and rectifying issues like emerging biases or unexpected behaviors.
- There's countless intricacies about each one of these steps that we could spend weeks learning but this high-level view of the process will set you up for the following lessons.

What is Transformer Architecture

Transformer architecture is a sophisticated type of neural network architecture designed to handle sequential data efficiently, making it particularly well-suited for tasks in natural language processing (NLP).

Natural Language Processing, is a field of technology that helps computers understand, interpret, and respond to human language in a valuable way. It's like teaching computers to understand our language and communicate with us using text or speech. Through NLP, computers can read text, hear speech, interpret it, measure sentiment, and determine which parts are important. So, whether it's Siri understanding a question you asked, or Google Translate turning English text into Spanish, that's NLP at work!

Now let's get into a fun example that explains what a Neural Network like transformer architecture is!

A Neural Network is like a team of workers trying to solve a puzzle. Each different type of team will excel or fall short on certain puzzles. And the more puzzles a teams tries to solve, the better their performance becomes.

Input Layer (Puzzle Pieces):

- The puzzle pieces (input data) are handed out to the team (neural network).

Hidden Layers (Team Members):

- The team members (hidden layers) work together, passing around pieces and making connections based on the shape and image on the pieces.
- These hidden layers of the neural network assign parameters to the input data and enable a neural network to learn complex patterns and make predictions or decisions based on what it gathers from the input data.

Activation Functions (Decision-Making Process):

- Each member makes decisions on whether a connection between pieces is correct aka making the best conclusion possible with the available pieces of information.

Output Layer (Finished Puzzle):

- As pieces fit together, the puzzle (output) starts to take shape, and in the end, the completed puzzle represents the solution (output data) to the original input.

Learning (Practice):

- The more puzzles the team solves, the better they get at finding connections faster and more accurately, representing the learning process of the neural network.

<u>Here are some of the different teams of neural networks</u>

Transformer Architecture:

- **Main Use**: Excels with text-based data due to its ability to quickly understand and analyze the relationships between different words in a sentence, no matter how far apart they are.

Convolutional Neural Networks (CNNs):

- **Main Use**: Image and video recognition, image classification, and processing geospatial data.
- They are good at this because of their ability to scan through pictures, focusing on small pieces at a time, to identify patterns and features like edges or colors which help in recognizing objects.

Generative Adversarial Networks (GANs):

- **Main Use**: Generating new data and content from any given dataset. Commonly used in image generation, style transfer, and sometimes in audio generation.
- This adversarial network is like having two artists in a competition; one tries to create realistic artwork, while the other judges if the artwork is real or fake, and through this back-and-forth, both get better, leading to the creation of very convincing, often visually indistinguishable, fake images or data.

Lastly, we have Autoencoders:

- **Main Use**: Data compression, denoising, and anomaly detection. They are used in image, text, and audio processing.
- Autoencoders are good for finding patterns and reducing noise in data by compressing it into a simpler form and then expanding it back, which can be helpful in tasks like spotting anomalies or organizing data more efficiently.

How the transformer architecture works

In the Transformer architecture, the encoder reads and processes a sequence of data (like a sentence) to create a sort of summary that captures the important information. The decoder then takes this summary and generates a new sequence of data (like a translated sentence) based on it. Throughout this process, both the encoder and decoder pay attention to different parts of the input data to understand the relationships between words or elements, making sure the output is meaningful and relevant to the input.

Imagine a bustling, busy airport where airplanes from various places land and take off. In this metaphor, the Transformer architecture is like the entire airport operations, the encoder is the arrival terminal, and the decoder is the departure terminal.

Encoder (Arrival Terminal):

- When planes (data) arrive at the airport (Transformer), they first enter the arrival terminal (encoder).
- At the arrival terminal, each plane's details (individual pieces of data) are logged, and passengers (features of the data) are screened and processed. This terminal has several counters and security checks (layers of the encoder) to understand who and what arrived.
- The personnel (self-attention mechanism) at the terminal pay attention to different groups of passengers (different parts of the data) based on their needs, destinations, or other criteria.
- By the time all planes and passengers are processed, the arrival terminal has a comprehensive understanding (high-dimensional representation) of who and what has arrived and their respective details.

Decoder (Departure Terminal):

- Now, passengers need to board new planes (generate output) to their next destinations. They move to the departure terminal (decoder).
- At the departure terminal, there are also several counters and security checks (layers of the decoder) to ensure everyone reaches the right plane.
- There's a special team (additional sub-layer performing multi-head attention over the encoder's output) that coordinates with the arrival terminal to get insights about any special requirements or conditions of passengers, ensuring a smooth transition.
- The personnel (self-attention mechanism) in this terminal also pay attention to different groups of passengers, ensuring they board the right

planes at the right time, considering their previous and upcoming journeys (context from the input).
- As planes (output data) take off from the departure terminal, they carry passengers (generated output) to their next destinations, based on all the information processed from the time they arrived, their preferences, and the coordination between both terminals.

In this manner, the Transformer's encoder processes and understands the input data, while the decoder generates the output data, each with its layers and self-attention mechanism ensuring the right focus and processing at every step.

How ChatGPT Answers Your Questions

GPT (Generative Pre-trained Transformer) doesn't "read" words in the way that humans do, but it processes them through a series of computational steps. Here's how it handles the words inputted by a user:

1. **Tokenization**:
   - Initially, the text input from the user is broken down into smaller pieces, much like splitting a sentence into individual words or even smaller parts. This process is called tokenization. For example, the sentence "ChatGPT is great!" might be split into three pieces: "ChatGPT", "is", and "great!". Each of these pieces is referred to as a token, and this breakdown makes it easier for the model to analyze the text and understand its structure.

2. **Embedding**:
   - After the text is broken down into smaller pieces, each piece is then converted into a numerical form, like a unique code, so that the computer can understand and work with it. This process is known as embedding. It's like translating the words into a language that the model can understand, where each word or piece gets its own unique numerical identifier. This way, GPT can process the text in a format that's suitable for mathematical operations, which are used in the later steps to analyze the text and generate a response.
   - Within these unique codes are numerical values pertaining to semantic and syntactic information.
     - **Semantic Information**:

        - Semantic information relates to the meaning of words and their relationships with other words. For instance, consider the words "doctor," "nurse," and "hospital." Semantically, these words are related because they all pertain to healthcare. In the embedding space, they might be represented by vectors that are close to each other, indicating their related meanings. On the other hand, a word like "apple" which is unrelated to healthcare, might have a vector far from those of "doctor," "nurse," and "hospital."
    - **Syntactic Information**:
        - Syntactic information is about the arrangement of words and phrases to create well-formed sentences in a language. It's like the framework or rules that dictate how sentences are constructed.
        - In English, the typical word order is Subject-Verb-Object (SVO). So, in the sentence "John eats apples," "John" is the subject, "eats" is the verb, and "apples" is the object. Syntactic information helps identify the correct order of words to make meaningful sentences.

3. **Encoding**:
In step 3, where the tokens are fed into the GPT model, we are essentially moving through a multi-layered network that refines the understanding of each token based on the tokens around it. So, let's break this down further:

Layers:

- GPT has a series of layers (like floors in a building), and each layer performs special operations on the incoming tokens. Imagine each layer as a kind of a workshop where tokens get refined or reshaped based on the surrounding context. GPT-3 for example has 175 billion parameters distributed across 96 Transformer layers.

Transformation:

- In each layer, tokens are transformed through mathematical operations. Think of this as a kind of translation from one language to another, helping to build a richer understanding of each token.

Attention Mechanism:

- Within each layer, there's a mechanism called "attention" which allows each token to "look at" other tokens in the input and adjust its own representation based on what it "sees." For instance, the word "bank" might adjust its representation based on whether the surrounding words relate to a financial institution bank or the side of a river bank.

Contextual Adjustment:

- The attention mechanism helps in adjusting the representation of each token based on its context, making sure that the model understands each word in a way that makes sense given the words around it.

Passing Through Layers:

- Now, these refined tokens move up to the next layer (the next workshop) and go through a similar process again and again. With each layer they pass through, they gain a deeper understanding of the context.

Aggregation of Information:

- As tokens move through the layers, they aggregate and gather more and more information from the surrounding tokens, which helps in building a rich, contextual understanding of the entire input.

By the time tokens reach the final layer, they have been significantly refined and carry a detailed understanding of the input text based on the context provided by the surrounding words. Each layer has contributed to building this deep understanding of the input, making the tokens ready for the next step, which is output generation.

4. Output Generation:

After processing the user's input, GPT begins the task of creating a response. It starts with the tokens provided in the input and then works to predict what comes next, one token at a time.

- **Prediction and Selection**:
  - For each new token, GPT looks at all the tokens that have come before it (including the user's input and any tokens it has generated so far) to predict the most likely next token. It makes this prediction based on patterns it has learned from the vast amount of text data it was trained on.
  - And then GPT selects the one that it calculates to be the most likely next token, based on its understanding from the training data.

When processing the input phrase "Albert Einstein was the world's most...", GPT would go through the following steps to predict and select the next word:

Tokenization:

- The input phrase is tokenized into individual tokens, e.g., ["Albert", "Einstein", "was", "the", "world's", "most", ...].

Embedding:

- Each token is converted into a numerical vector using an embedding matrix.

Processing Through Layers:

- The values are processed through the multiple layers of the GPT model. Through self-attention mechanisms, the model identifies relationships between tokens, understanding, for instance, that "Albert Einstein" refers to a notable individual.

Probability Distribution:

- Based on the context, the model computes a probability distribution over the vocabulary for the next token after "most". Common continuations like "famous", "influential", or "brilliant" might receive high probabilities.
- Now what is a probability distribution?

A probability distribution is like a rule that tells us how likely different outcomes are in a situation involving chance.

Imagine you have a jar of colored candies: red, blue, and green. If the candies are mixed equally, the chance of picking each color is the same, 1 out of 3, or about 33.3%.

Now, if we write down these chances for each color, we get a simple probability distribution:

- Chance of red: 33.3%
- Chance of blue: 33.3%
- Chance of green: 33.3%

In the case of GPT generating a word, think of it like the model has a huge jar of words, and some words are more likely to be picked next based on the previous words. The probability distribution is the rule that tells GPT how likely each word is to be picked next.

Selection of 'influential':

- The model may then select the token "influential" as it has a high probability and fits well with common descriptions of Albert Einstein.

Continuation:

- The model continues this process for each subsequent token, using the growing context to inform the prediction and selection of the next token. For instance, after "influential", it might predict "physicist" with high probability given the context of Albert Einstein.

Stopping Criterion:

- The model continues generating tokens until a stopping criterion is met, such as an end-of-sentence punctuation or a maximum token limit.

This process continues until we create the phrase = Albert Einstein was the world's most influential physicist of the 20th century, whose work revolutionized our understanding of the fundamental laws of the universe.