Large language models (LLMs)

Large language models (LLMs) are advanced AI systems that use deep learning, particularly the Transformer architecture, to understand and generate human-like text by learning patterns from vast datasets. Trained on billions of texts and other content, LLMs excel at tasks like text generation, translation, summarization, and question answering. Key factors enabling their power include the Transformer's parallel processing and "attention" mechanisms, increased computational power, and the availability of large, high-quality datasets.

How LLMs Work

1. **Training on Data**:

LLMs are trained on massive amounts of text and data to identify complex patterns, grammar, and factual information.

2. **Transformer Architecture**:

This deep learning architecture, introduced in 2017 by Google, is fundamental to most LLMs.

3. **Attention Mechanisms**:

Within the Transformer, "attention" mechanisms allow the model to focus on the most relevant parts of the input text when processing and generating output, improving understanding of context.

4. **Text Generation**:

LLMs generate responses word-by-word by predicting the most probable next word in a sequence based on the preceding text and its vast training data.

Key Characteristics

- **Vast Parameters**:

LLMs have a huge number of parameters, which are essentially the learned values that enable them to process longer text sequences and handle complex tasks.

- **Contextual Understanding**:

They can understand the relationships between words and phrases within a text, enabling them to produce relevant and in-context content.

- **Self-supervised Learning**:

LLMs are trained using self-supervised machine learning, which allows them to learn from data without requiring explicit human labels.

Applications

- **Content Creation**: Generating articles, stories, and code.
- **Translation**: Translating text between different languages.
- **Summarization**: Condensing long pieces of text into shorter summaries.
- **Question Answering**: Providing answers to questions based on their training data.

Challenges

High Costs: Training LLMs is computationally intensive and expensive.

Bias: LLMs can reflect biases present in their training data.

Hallucinations: They may sometimes generate factually incorrect or nonsensical information.

Ethical Implications: Responsible deployment, data privacy, and potential misuse are significant concerns.

Introduction to Tokenizer

A **tokenizer** is a fundamental component in **Natural Language Processing (NLP)** and **Deep Learning** that breaks down raw text into smaller, manageable units called **tokens**. These tokens can be **words, subwords, characters, or symbols**, depending on the application and model design. Tokenization is the **first and most crucial step** in transforming human language into a numerical format that computers can process and understand.

What Is Tokenization?

Tokenization is the process of splitting a sequence of text into individual pieces (tokens). For example:

Sentence: “Deep learning transforms AI.”
Tokens: [“Deep”, “learning”, “transforms”, “AI”, “.”]

Each token represents a meaningful element of text that the model can analyze. After tokenization, these tokens are usually converted into **numerical IDs** through a **vocabulary** or **embedding layer**, enabling the neural network to perform computations.

Example: Book page 38

Have the bards who preceded me left any theme unsung?

1. Overview

Feature	**BERT**	**GPT**
Full Name	Bidirectional Encoder Representations from Transformers	Generative Pre-trained Transformer
Developed By	Google AI	OpenAI
Model Type	Encoder-only Transformer	Decoder-only Transformer
Training Objective	**Masked Language Modeling (MLM)**: Predict missing words in a sentence	**Causal Language Modeling (CLM)**: Predict next word in a sequence
Directionality	**Bidirectional**: reads context from both left and right	**Unidirectional** (left-to-right)
Purpose	Understanding tasks: classification, QA, NER, sentiment analysis	Generation tasks: text completion, summarization, dialogue, story writing
Input Processing	Tokens fed through encoder to produce embeddings representing context	Tokens fed through decoder to generate next token probabilities

2. Tokenization

- **BERT:** Uses **WordPiece tokenizer**, which splits rare words into subwords. Special tokens include [CLS] (classification start) and [SEP] (sentence separator).
- **GPT:** Uses **Byte-Pair Encoding (BPE)**. Adds special handling like Ġ to indicate a space, supporting smooth text generation.

Example: “Have the bards”

- BERT tokens: [CLS], have, the, bards, [SEP]
- GPT tokens: ['Have', 'Ġthe', 'Ġbards']

3. Use Cases

BERT (Understanding):

- Sentiment Analysis
- Question Answering (SQuAD)
- Named Entity Recognition (NER)
- Text Classification

GPT (Generation):

- Text completion
- Story/essay generation
- Dialogue/chatbots
- Summarization and translation

4. Key Differences in Training & Architecture

1. **Contextual Understanding**
   - BERT: Bidirectional → understands meaning from **both left and right context**.
   - GPT: Left-to-right → generates coherent text **one word at a time**.
2. **Fine-Tuning**
   - BERT: Fine-tuned on downstream tasks with task-specific layers.
   - GPT: Can be fine-tuned, but often used in **few-shot learning** or **prompt-based generation**.
3. **Output**
   - BERT: Produces embeddings for tokens (good for classification or comprehension).
   - GPT: Produces probabilities for **next token prediction** (good for generation).

Summary

- **BERT = “Reader”** → excels at understanding and analyzing text.
- **GPT = “Writer”** → excels at generating coherent and fluent text.