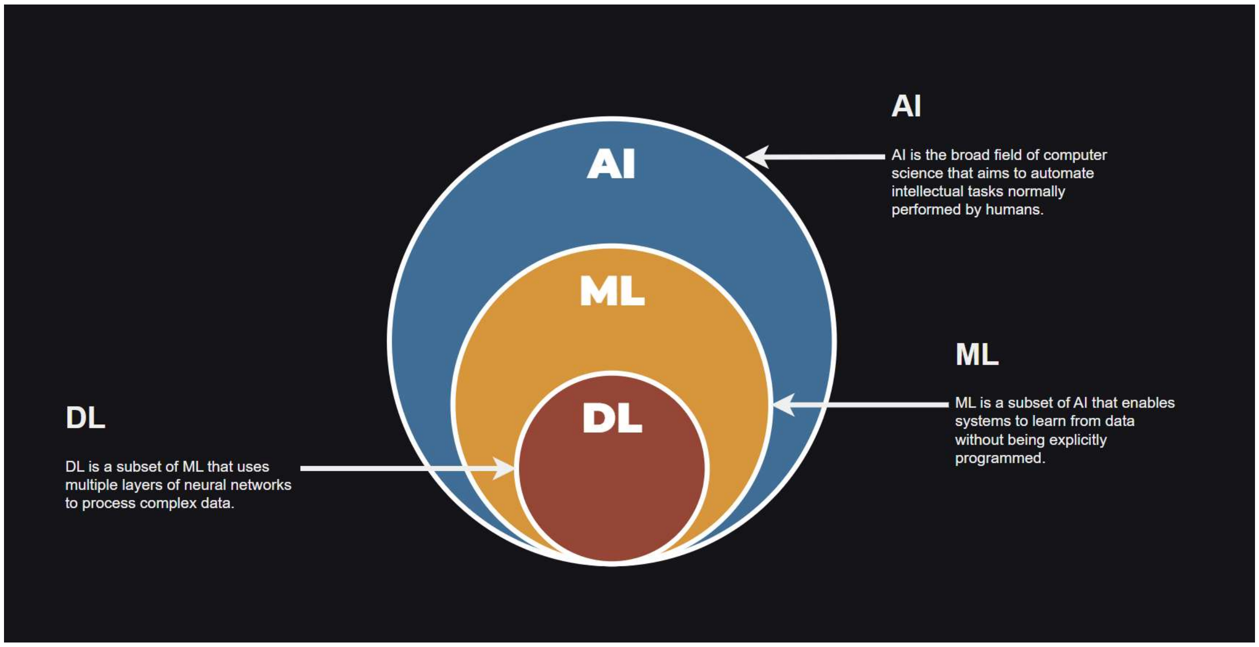
The Mechanics of Generative AI

1. The Paradigm Shift: Deterministic vs. Probabilistic

As Java developers, we are used to **Deterministic Systems**: If `input == A`, then `return B`. Generative AI introduces **Probabilistic Systems**. An LLM (Large Language Model) does not "retrieve" an answer from a database; it "calculates" the answer based on statistical likelihood.

When you ask "What is the capital of France?", the model isn't looking up a fact; it is predicting that the token "Paris" has the highest probability (e.g., 99.8%) of following that sequence. This shift requires a new mental model: we are no longer programming explicit logic, but rather orchestration and context management.


AI
AI is the broad field of computer science that aims to automate intellectual tasks normally performed by humans.
AI
ML
DL
ML
ML is a subset of AI that enables systems to learn from data without being explicitly programmed.
DL
DL is a subset of ML that uses multiple layers of neural networks to process complex data.

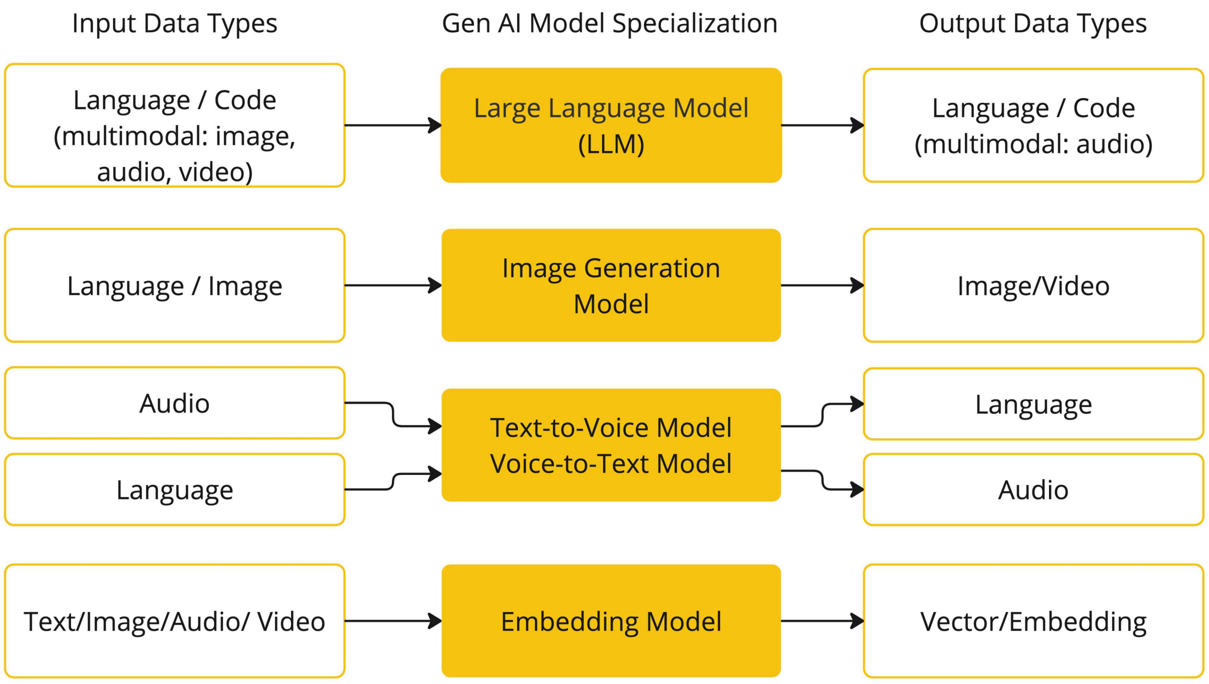
2. LLMs & The Transformer Architecture

Modern LLMs (GPT-4, Llama 3, Claude) are based on the **Transformer Architecture** (introduced by Google, 2017).

A good resource: https://poloclub.github.io/transformer-explainer/

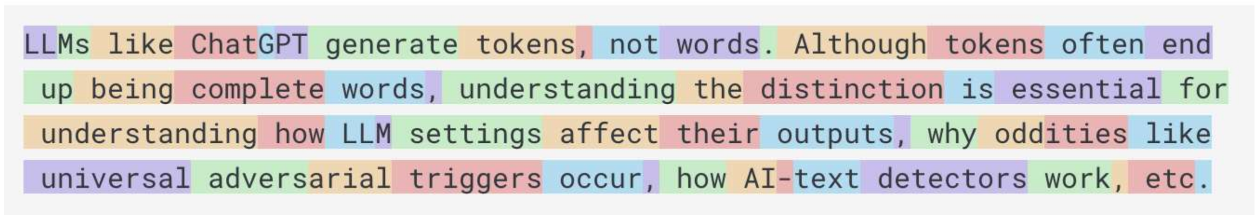
- **The Mechanism:** Unlike older RNNs (Recurrent Neural Networks) that processed text sequentially, Transformers process the entire input at once using an **"Attention Mechanism."** This allows the model to understand that in the sentence *"The animal didn't cross the street because it was too tired,"* the word *"it"* refers to the *"animal,"* but in *"The animal didn't cross the street because it was too wide,"* the word *"it"* refers to the *"street."*

- **Parameters:** When you hear "Llama-3-8B," the **8B** refers to 8 Billion parameters. Parameters are the internal weights (floating-point numbers) inside the neural network that represent the model's learned knowledge. More parameters generally mean higher reasoning capability but require more VRAM (Video RAM) to run.



3. The Unit of Computation: Tokens

LLMs do not see words; they see integers known as **Tokens**.



```
LLMs like ChatGPT generate tokens, not words. Although tokens often end
 up being complete words, understanding the distinction is essential for
 understanding how LLM settings affect their outputs, why oddities like
 universal adversarial triggers occur, how AI-text detectors work, etc.
```

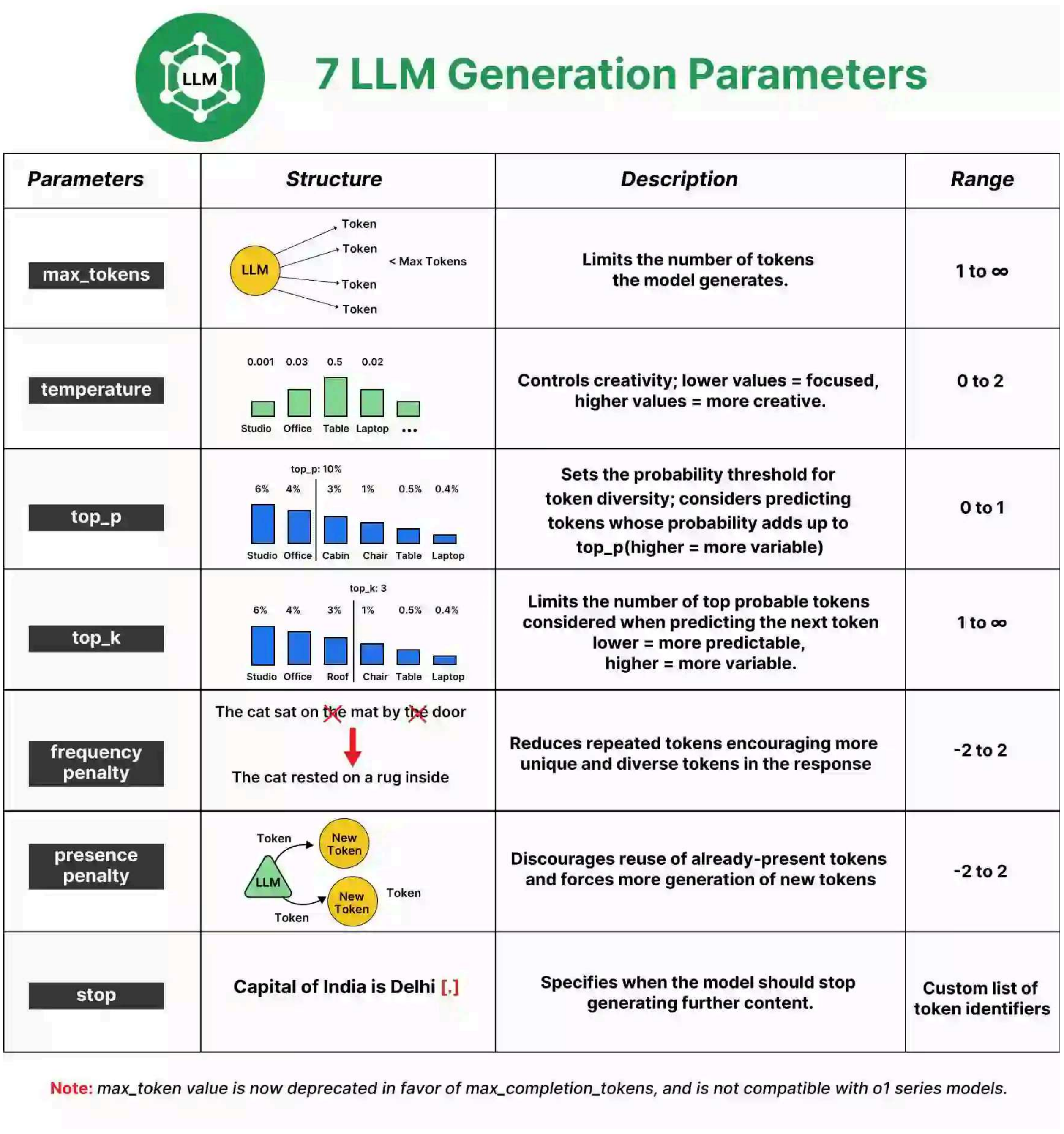
- **Tokenization:** Before text hits the model, a "Tokenizer" breaks text into chunks. Common words map to single tokens (`" apple"`), while complex words map to multiple (`"implementation"` might be `["imp", "le", "men", "ta", "tion"]`).
- **The Math:** roughly **1,000 tokens are approximately 750 words**.
- **Why this matters for Devs:**
  1. **Cost:** OpenAI/Azure charge per 1M tokens (Input tokens are usually cheaper than Output tokens).
  2. **Context Window:** Every model has a "Context Window" (e.g., 8k, 128k). This is the maximum buffer size. If your prompt + previous chat history + the generated answer exceeds this limit, the model crashes or truncates the conversation.
  3. **Latency:** Output generation is sequential. A model generating 100 tokens takes roughly 10x longer than a model generating 10 tokens.

4. Hyperparameters: Controlling the "Brain"

When calling an AI model via API (Spring AI), you pass configuration parameters that alter behavior:

- **Temperature (0.0 to 1.0):** Controls randomness in the output layer (Softmax).
  - `0.0`: **Greedy Decoding.** The model always picks the single most likely next token. Result: Deterministic, repetitive, factual. *Use case: JSON extraction, code generation.*
  - `0.8+`: **Sampling.** The model might pick the 2nd or 3rd most likely token. Result: Creative, varied. *Use case: Storytelling, brainstorming.*

- **Top-P (Nucleus Sampling):** An alternative to temperature. It restricts the token pool to the top X% of probability mass. Usually, set Temperature or Top-P, not both.
- **More here**: https://www.promptingguide.ai/introduction/settings



7 LLM Generation Parameters

Parameters	***Structure***	***Description***	***Range***
max_tokens	LLM Token Token Token Token < Max Tokens	**Limits the number of tokens the model generates.**	**1 to ∞**
temperature	0.001 Studio; 0.03 Office; 0.5 Table; 0.02 Laptop; …	**Controls creativity; lower values = focused, higher values = more creative.**	**0 to 2**
top_p	top_p: 10%; 6% Studio; 4% Office; 3% Cabin; 1% Chair; 0.5% Table; 0.4% Laptop	**Sets the probability threshold for token diversity; considers predicting tokens whose probability adds up to top_p(higher = more variable)**	**0 to 1**
top_k	top_k: 3; 6% Studio; 4% Office; 3% Roof; 1% Chair; 0.5% Table; 0.4% Laptop	**Limits the number of top probable tokens considered when predicting the next token lower = more predictable, higher = more variable.**	**1 to ∞**
frequency penalty	The cat sat on ~~the~~ mat by ~~the~~ door $\downarrow$ The cat rested on a rug inside	**Reduces repeated tokens encouraging more unique and diverse tokens in the response**	**-2 to 2**
presence penalty	Token New Token LLM Token New Token Token	**Discourages reuse of already-present tokens and forces more generation of new tokens**	**-2 to 2**
stop	**Capital of India is Delhi [.]**	**Specifies when the model should stop generating further content.**	**Custom list of token identifiers**

Note: *max_token value is now deprecated in favor of max_completion_tokens, and is not compatible with o1 series models.*

5. Prompt Engineering Patterns

Prompt Engineering is the syntax of GenAI. It is not just "asking nicely"; it is structuring the input to constrain the probabilistic output.

A good resource: https://www.promptingguide.ai/techniques

- **Zero-Shot:** Providing a task without examples.
    - *Input:* `Classify this email sentiment: "I hate this service."`
- **Few-Shot (In-Context Learning):** Providing examples within the prompt to teach the model a specific format.Plaintext

```
// Prompt
Convert natural language to SQL.
Input: "Get all users from London" → Output: SELECT * FROM users WHE
RE city = 'London';
Input: "Count orders in 2023" → Output: SELECT count(*) FROM orders W
HERE year = 2023;
Input: "Show top 5 products" → Output:
```

- **Chain of Thought (CoT):** Forcing the model to "show its work." This significantly increases accuracy in math and logic problems.
    - *Technique:* Append the phrase *"Let's think step by step"* to your prompt. This forces the model to generate reasoning tokens *before* the final answer token, which grounds the logic.

6. Running Locally: Ollama

For development, we avoid cloud costs and privacy risks by running models locally. **Ollama** is the industry standard runtime for this. It wraps `llama.cpp` (a C++ library for inference) in a user-friendly API.

- **Installation:** Download from ollama.com.

- **Pulling a Model:** Models are stored in `~/.ollama`. To pull Llama 3 (Meta's open model):

  ```
  //pull model
  ollama pull llama3

  //run model (interactive + verbose)
  ollama run llama3 --verbose
  ```

- **The API:** Ollama exposes a REST API on port `11434`. This is what Spring AI connects to. You can test it with `curl`

  ```
  curl http://localhost:11434/api/generate -d '{
    "model": "llama3",
    "prompt": "Write a Hello World in Java",
    "stream": false
  }'
  ```