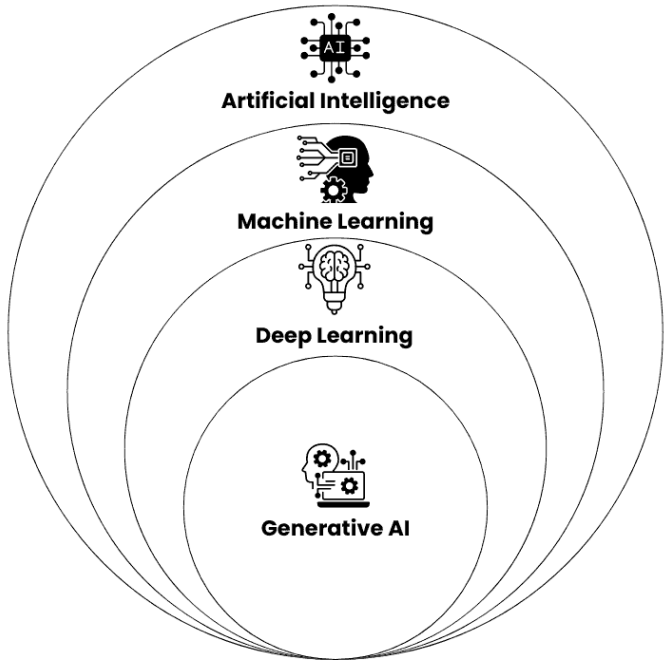
Introducción a Modelos de Lenguaje Grandes (LLMs)

Tipos de IA


AI
Artificial Intelligence
Machine Learning
Deep Learning
Generative AI

¿Qué es un LLM?

Un **Modelo de Lenguaje Grande (LLM)** es un tipo de modelo de inteligencia artificial que ha sido entrenado en una gran cantidad de datos de texto para comprender y generar lenguaje humano.

- **Grande**: Se refiere a la enorme cantidad de parámetros (miles de millones) que tiene el modelo y la cantidad de datos con los que se entrena.
- **Lenguaje**: Se especializa en tareas relacionadas con el lenguaje, como traducción, resumen, respuesta a preguntas y generación de texto.
- **Modelo**: Es una representación matemática de la distribución de probabilidad del lenguaje.

Conceptos Fundamentales: Tokens

Los LLMs procesan texto dividiéndolo en **tokens**. La forma en que se divide el texto (tokenización) es clave para el rendimiento del modelo.

Tipos de Tokenización:

- **Basada en Palabras:** Cada palabra es un token.
  - `El cielo es azul.` -> `["El", "cielo", "es", "azul", "."]`
  - *Problema:* Falla con palabras desconocidas o complejas ("desoxirribonucleico").
- **Basada en Caracteres:** Cada carácter es un token.
  - `El cielo es azul.` -> `["E", "l", " ", "c", "i", "e", "l", "o", ...]`
  - *Ventaja:* Nunca hay "caracteres desconocidos".
  - *Problema:* Secuencias muy largas, pierde el significado semántico de las palabras.

Puedes experimentar con la tokenización en el OpenAI Tokenizer.

Conceptos Fundamentales: Tokens (Continuación)

Tokenización de Subpalabras (Subword Tokenization): El estándar actual

Combina lo mejor de los dos mundos. Las palabras comunes son un solo token, mientras que las palabras raras o complejas se dividen en subpalabras significativas.

- **Algoritmo Común:** Byte Pair Encoding (BPE).

Ejemplo con la palabra `incomprensiblemente`:

- **Tokenización BPE:** `["in", "comprensi", "ble", "mente"]`
- **Ventajas:**
  - Maneja palabras raras sin tratarlas como "desconocidas".
  - Mantiene un vocabulario de tamaño manejable.
  - Captura información morfológica (prefijos, sufijos).

Cada modelo (GPT, Llama, etc.) tiene su propio tokenizador entrenado.

Conceptos Fundamentales: Ventana de Contexto

Es la cantidad máxima de tokens que un modelo puede considerar al generar una respuesta. Una ventana más grande permite al modelo "recordar" más del texto previo.

Ejemplo:

- **Ventana Pequeña (4 tokens):**
  - *Input:* `El perro persigue al gato por el tejado.`
  - *Modelo solo ve:* `por el tejado.`
  - *Pregunta:* ¿Quién persigue al gato? *Respuesta:* No lo sé.
- **Ventana Grande (16 tokens):**
  - *Input:* `El perro persigue al gato por el tejado.`
  - *Modelo ve:* `El perro persigue al gato por el tejado.`
  - *Pregunta:* ¿Quién persigue al gato? *Respuesta:* El perro.

Conceptos Fundamentales: Temperatura

Controla la "creatividad" o aleatoriedad de las respuestas del modelo.

- **Baja temperatura (~0.2):** Respuestas más predecibles y enfocadas.
- **Alta temperatura (~0.9):** Respuestas más creativas y diversas.

Ejemplo: Completar la frase "El sol brilla y..."

- **Temp. Baja:** "…el cielo está despejado." (Una respuesta muy común y esperada).
- **Temp. Media:** "…los pájaros cantan en los árboles." (Creativa pero coherente).
- **Temp. Alta:** "…los sueños de los dragones pintan el universo." (Muy creativa, quizás sin sentido).

Conceptos Fundamentales: Top-P (Nucleus Sampling)

Controla la selección de palabras de un vocabulario de forma dinámica. En lugar de tomar las `k` más probables (Top-K), Top-P selecciona el conjunto más pequeño de palabras cuya probabilidad acumulada es mayor que `p`.

- **Bajo Top-P (~0.1):** Muy conservador. Solo considera las palabras más probables.
- **Alto Top-P (~0.9):** Más diverso. Considera un rango más amplio de palabras posibles, incluso algunas menos probables.

Ejemplo: Completar la frase "El sol brilla y..." con P=0.75

1. El modelo calcula las probabilidades de la siguiente palabra:
   - `el` : 0.4
   - `los` : 0.2
   - `las` : 0.15
   - `su` : 0.1

Práctica: PartyRock de Amazon

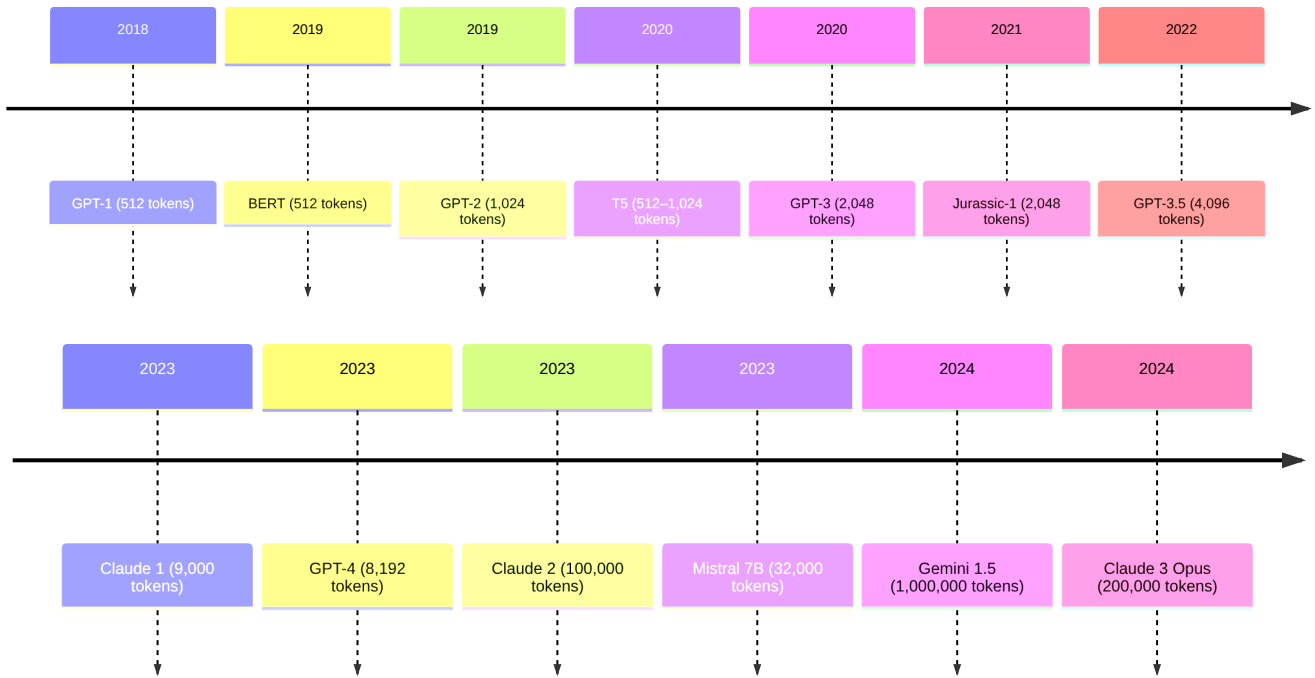
Para experimentar con los conceptos de **Temperatura** y **Top-P** de forma interactiva, utilizaremos **PartyRock**, un playground de IA generativa de Amazon Bedrock.

Esto nos permitirá ver en tiempo real cómo estos parámetros afectan la creatividad y coherencia de las respuestas de un LLM.

PartyRock

Evolución de la Ventana de Contexto

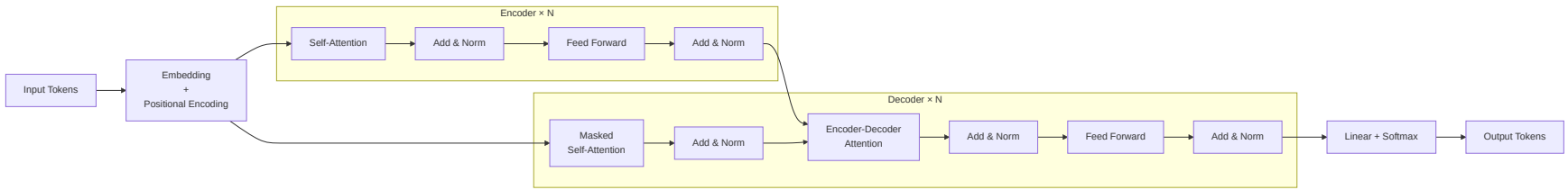
La ventana de contexto de los LLMs ha crecido exponencialmente, permitiendo procesar documentos cada vez más largos y mantener conversaciones más coherentes.



Arquitectura Transformer

La mayoría de los LLMs modernos se basan en la arquitectura **Transformer**, introducida en 2017.

Su innovación clave es el mecanismo de **atención**, que permite al modelo ponderar la importancia de diferentes tokens al procesar una secuencia.



Repaso de Spring Boot

1. Estructura Básica y `pom.xml`

Spring Boot simplifica el desarrollo de aplicaciones Spring. Un proyecto típico incluye:

- **Clase Principal:** Anotada con `@SpringBootApplication`.
- **Controladores (`@RestController`):** Manejan las peticiones HTTP.
- **Servicios (`@Service`):** Contienen la lógica de negocio.
- **Repositorios (`@Repository`):** Interactúan con la base de datos.

Ejemplo de `pom.xml` (simplificado)

Parte 1:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
</project>
```

Ejemplo de `pom.xml` (simplificado)

Parte 2:

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- Otras dependencias como spring-boot-starter-data-jpa, etc. -->
    </dependencies>
</project>
```

2. Controlador (`@RestController`)

Maneja las peticiones HTTP y devuelve respuestas. Utiliza anotaciones para mapear URLs y métodos HTTP.

```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/saludos")
public class SaludoController {

    @GetMapping("/hola")
    public String saludar() {
        return "¡Hola desde Spring Boot!";
    }

    @GetMapping("/personalizado/{nombre}")
    public String saludarPersonalizado(@PathVariable String nombre) {
        return "¡Hola, " + nombre + "!";
    }
}
```

3. Servicio (`@Service`)

Contiene la lógica de negocio de la aplicación. Los controladores delegan tareas a los servicios.

```java
import org.springframework.stereotype.Service;

@Service
public class SaludoService {

    public String obtenerSaludoPorDefecto() {
        return "Un saludo cordial.";
    }

    public String generarSaludoComplejo(String nombre, int edad) {
        if (edad < 18) {
            return "Hola joven " + nombre + ".";
        } else {
            return "Saludos, Sr./Sra. " + nombre + ".";
        }
    }
}
```

4. Lectura de Configuración (`application.properties`)

Spring Boot permite configurar la aplicación usando `application.properties` o `application.yml`.

`src/main/resources/application.properties`:

```
app.mensaje.bienvenida=Bienvenido a mi aplicación Spring Boot
app.version=1.0.0
```

```
@Component
public class AppConfig {

    @Value("${app.mensaje.bienvenida}")
    private String mensajeBienvenida;
    @Value("${app.version}")
    private String appVersion;

    public void mostrarConfiguracion() {
        System.out.println("Mensaje: " + mensajeBienvenida);
        System.out.println("Versión: " + appVersion);
    }
}
```