IA local para PyMEs · Módulo 0

Fundamentos y vocabulario

Las palabras con las que se habla el resto del curso, definidas por lo que hacen. Y un glosario con símiles para quien llega de cero.

Qué se construye en este módulo

🕮 Glosario con símiles: para entender de qué se habla

Explicar cada término del curso con un símil cotidiano

🕮 Texto plano y Markdown: la base de todo

Justificar por qué todo el sistema se construye sobre texto versionable

🕮 Modelo, pesos, cuantización, contexto y tokens

Definir cada término de forma medible

Qué se construye en este módulo

📖 **Embeddings, herramienta, agente, skill, harness, MCP y multimodal**

Distinguir cada pieza por lo que hace

0.1 · Glosario con símiles: para entender de qué se habla

Glosario con símiles: para entender de qué se habla

Este curso usa unas cuarenta palabras que en las noticias suenan a magia y en la práctica son cosas concretas y medibles. Aquí están todas, cada una con un **símil de la vida diaria** y, al lado, la advertencia de **dónde se rompe el símil** —porque todo símil se rompe, y saber dónde es la mitad de entender el término.

En una tabla

Término	Símil	Dónde se rompe el símil	Cómo se ve en la práctica
Embedding	Convertir un texto en **coordenadas en un mapa**: textos parecidos quedan cerca	En un mapa hay 2 dimensiones; aquí hay cientos, y «cerca» lo decide el modelo de embeddings, no el sentido común	Una lista de 384 a 1.024 números por cada pedazo de texto
Índice vectorial	El archivador donde se guardan esas coordenadas para buscar «lo que se parece»	Busca parecido, **no verdad**: puede traer lo parecido y equivocado	Una base de datos (o una tabla) con los embeddings y sus metadatos
Chunk	Cada ficha en la que se recorta un documento para archivarlo	Recortar mal —por tamaño y no por sentido— **rompe la información**	Pedazos de 200 a 800 tokens con su origen anotado
RAG	Dejarle sobre la mesa	Si las páginas que se	Recuperar → poner en

0.2 · Texto plano y Markdown: la base de todo

Texto plano y Markdown: la base de todo

Todo lo que se construye en este curso —prompts, esquemas, configuraciones, trazas, políticas, documentación de entrega— es **texto**. La decisión más barata y más duradera que se toma al principio es en qué formato vive ese texto. La respuesta es aburrida y correcta: **texto plano**.

En una tabla

Perfil	Este laboratorio
A · sin GPU	Igual. Es texto
B · 8–16 GB	Igual
C · 24 GB+	Igual

Cuándo NO usar esto

Texto plano y Markdown: la base de todo

- Documentos con diagramación exacta y valor legal (un contrato con márgenes, firmas y foliación): se producen en PDF desde una plantilla; el Markdown es el borra
- Datos tabulares grandes: una tabla Markdown de 2.000 filas no sirve para nada. Eso es CSV o una base de datos; Markdown solo para la tabla resumen.
- Formularios que llena gente no técnica: nadie va a escribir `| celda |` a mano. Se les da una interfaz, y la interfaz guarda texto plano por debajo.

Qué se degrada en cada perfil

A · sin GPU

Igual. Es texto

B · 8–16 GB VRAM

Igual

C · 24 GB+

Igual

0.3 · Modelo, pesos, cuantización, contexto y tokens

Modelo, pesos, cuantización, contexto y tokens

Cada término de esta lección se define por **un número que se puede medir en la terminal**. Si alguien los usa sin número, está hablando de otra cosa.

En una tabla

Término	El número	Cómo se mide
Modelo (tamaño)	parámetros: 1B, 8B, 70B…	está en el nombre del archivo
Pesos en disco	gigabytes	`ls -lh`
Cuantización	bits por parámetro: 16, 8, 4…	el sufijo: `F16`, `Q8_0`, `Q4_K_M`
Contexto	tokens que caben en la ventana	lo declara el runtime al cargar
Tokens	cuántos pedazos tiene un texto	un programa tokenizador

El comando, copiable

```
cd ~/labs/el-tornillo && mkdir -p labs/00-03 && cd labs/00-03
python3 -m venv .venv && source .venv/bin/activate
pip install -q tokenizers
# tokenizer.json: descárguelo del repositorio del modelo que vaya a usar (Hugging
    Face) y
# póngalo aquí. Es un archivo de texto; no hacen falta los pesos.
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Modelo, pesos, cuantización, contexto y tokens

- No cuente tokens con el tokenizador de otro modelo para presupuestar contexto: las cifras pueden diferir un 30 %.
- No cuantice a Q2/Q3 para «que quepa» en un equipo que no da: mida primero; casi siempre es mejor un modelo más pequeño en Q8 que uno grande en Q2.
- No compre por parámetros. Un 70B en Q2 puede rendir peor que un 8B en Q8 y costar cuatro veces más de memoria.

Qué se degrada en cada perfil

A · sin GPU

Igual: tokenizar es CPU

B · 8–16 GB VRAM

Igual

C · 24 GB+

Igual

0.4 · Embeddings, herramienta, agente, skill, harness, MCP y multimodal

Embeddings, herramienta, agente, skill, harness, MCP y multimodal

Esta lección cierra el vocabulario con las piezas que **actúan**. Cada una se define por lo que hace y se demuestra con el ejemplo más pequeño que corre. Nada de esto necesita GPU.

El comando, copiable

```
# Un servidor MCP mínimo que expone buscar_factura. [VOLÁTIL — verificado: 2026-09]
pip install -q mcp
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Embeddings, herramienta, agente, skill, harness, MCP y multimodal

- Un agente para una tarea que un flujo determinista resuelve (mover un archivo, renombrar, sumar una columna). Módulo 9: el modelo solo donde hay ambigüedad.
- Embeddings para buscar un número exacto (una cédula, un número de factura). Eso es una búsqueda exacta en base de datos; los embeddings traerán «parecidos».
- MCP para una sola herramienta que usa un solo programa. Es un enchufe: vale la pena cuando hay más de un aparato.

Entregables del módulo

- el `README.md` versionado. De aquí en adelante, cada laboratorio termina con un commit cuyo mensaje empieza por «Capa N».
- `labs/00-03/tokens.py` y una tabla con los tokens de los tres documentos más largos del cliente.

Las palabras con las que se habla el resto del curso, definidas por lo que hacen.