【AI Summer Camp】

2. Personalización de Modelos: RAG & Fine-Tuning

Del modelo generalista al experto

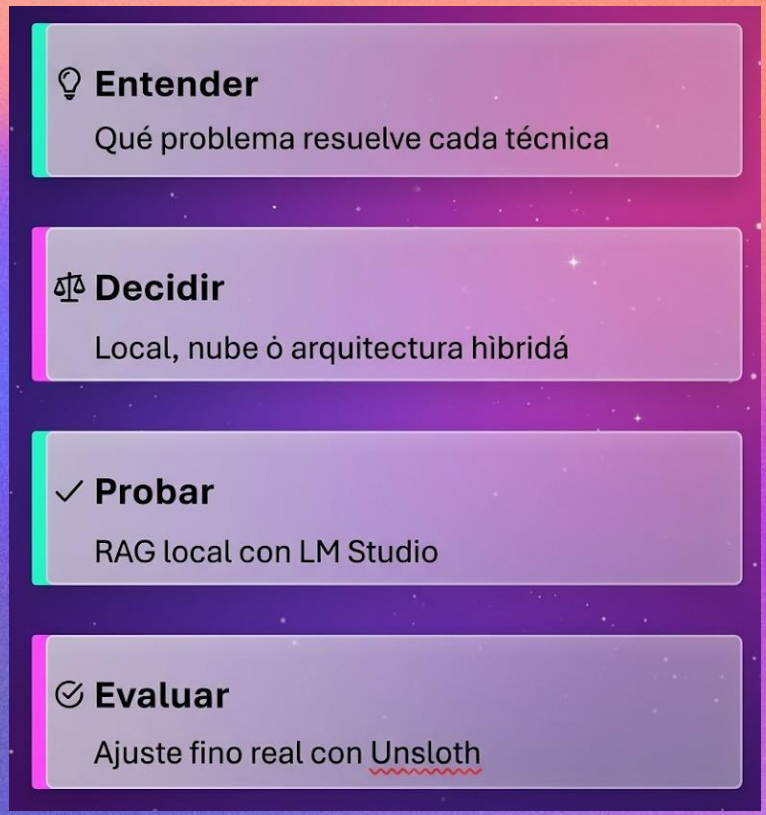
Enrique Javier Osorio Criado

AI Engineer · Global Digital Consulting

MADRID INNOVATION | MIL | AI Network Artificial Intelligence Association

QUÉ APRENDEREMOS HOY

- Al terminar la sesión entenderás qué problema resuelve cada técnica: el prompt de sistema, RAG, fine-tuning, guardrails y SLMs.
- También deberías poder decidir cuándo conviene ejecutar un modelo en local, cuándo usar nube y cuándo combinar ambas opciones.
- La práctica se centra en dos ideas: crear un pequeño RAG local con LM Studio y observar un ajuste fino real con Unsloth en Colab.



SESIÓN 2

AGENDA

01

Fundamentos

LLMs, tokens, parámetros, cuantización y modelos instruct.

02

IA local 2026

Coste, soberanía, privacidad, regulación y latencia.

03

RAG + ajuste

RAG, fine-tuning, guardrails, calidad y matriz de decisión.

04

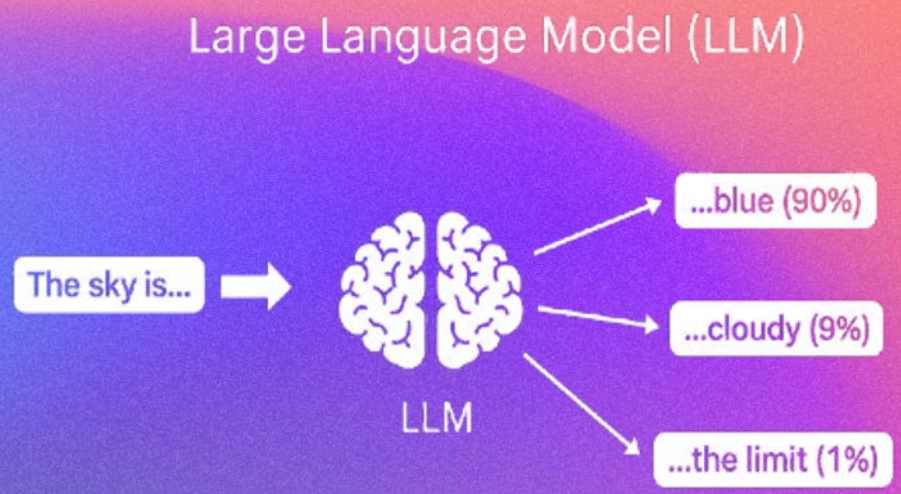
Práctica

LM Studio, RAG sencillo, prompt de sistema y Unsloth.

【AI Summer Camp】

QUÉ ES UN LLM

- Un LLM es una red neuronal entrenada con enormes cantidades de texto para predecir la siguiente pieza de texto más probable.
- No piensa como una persona: detecta patrones estadísticos muy complejos y los usa para generar respuestas plausibles. La escala permite capturar gramática, conceptos, hechos y patrones complejos.
- Su utilidad aparece cuando esos patrones se combinan con instrucciones claras, contexto fiable y controles adecuados.
- Por eso no debemos tratarlo como una fuente de verdad, sino como un motor de lenguaje y razonamiento probabilístico.

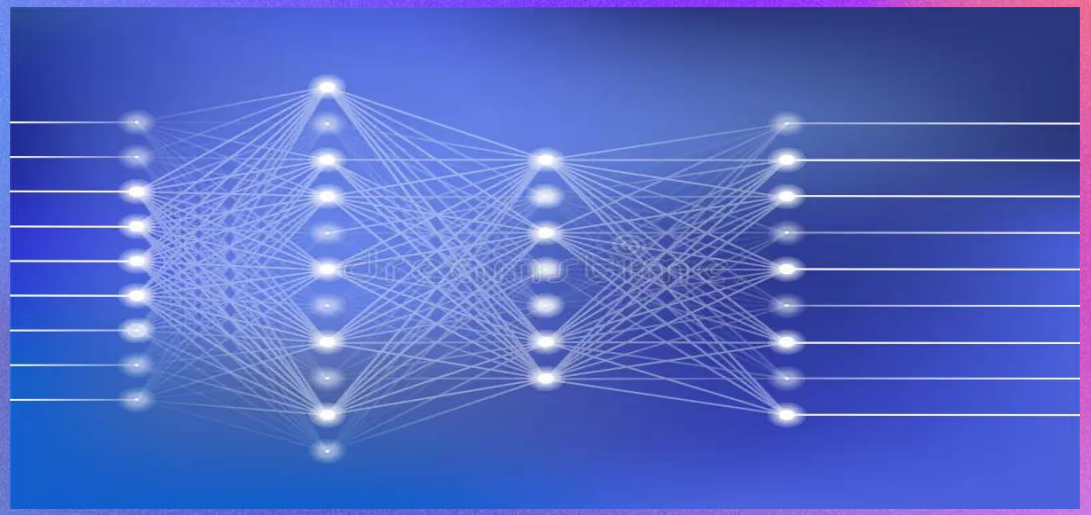


LOS PARÁMETROS

- Los parámetros son valores internos que el entrenamiento ajusta para que el modelo prediga mejor. No son una base de datos: son patrones aprendidos en matrices numéricas.
- Más parámetros suelen dar más capacidad, pero también más memoria y coste de ejecución.

Neuronas biológicas

Red neuronal artificial



PARÁMETROS Y CONOCIMIENTO

- Más parámetros suelen permitir más capacidad, pero también más memoria, más coste y más dificultad para ejecutarlo en local.
- El reto actual no es usar siempre el modelo más grande, sino elegir el tamaño suficiente para la tarea.

TOKENS Y COSTE

- Los modelos no procesan palabras completas: procesan tokens, que son fragmentos de texto.
- Cada entrada, cada documento recuperado y cada respuesta consume tokens.
- En la nube, los tokens suelen convertirse en coste directo. En local, se convierten en tiempo, memoria y consumo de hardware.

La pregunta práctica es siempre la misma: cuántos tokens necesito para resolver bien el caso de uso sin disparar costes ni latencias.

CUANTIZACIÓN

FP16

máxima precisión · memoria alta

INT8

compresión moderada · pérdida baja

INT4

mínima memoria · mayor pérdida

- La cuantización reduce la precisión numérica de los pesos del modelo para que ocupen menos memoria.
- Pasar de FP16 a INT8 o INT4 permite ejecutar modelos que de otra forma no cabrían en RAM o VRAM.
- La contrapartida es una posible pérdida de calidad, normalmente pequeña si se usan cuantizaciones modernas y modelos adecuados.
- En la práctica, GGUF Q4 o Q8 suele ser una buena opción para modelos en local.

MODELOS INSTRUCT

MODELO BASE

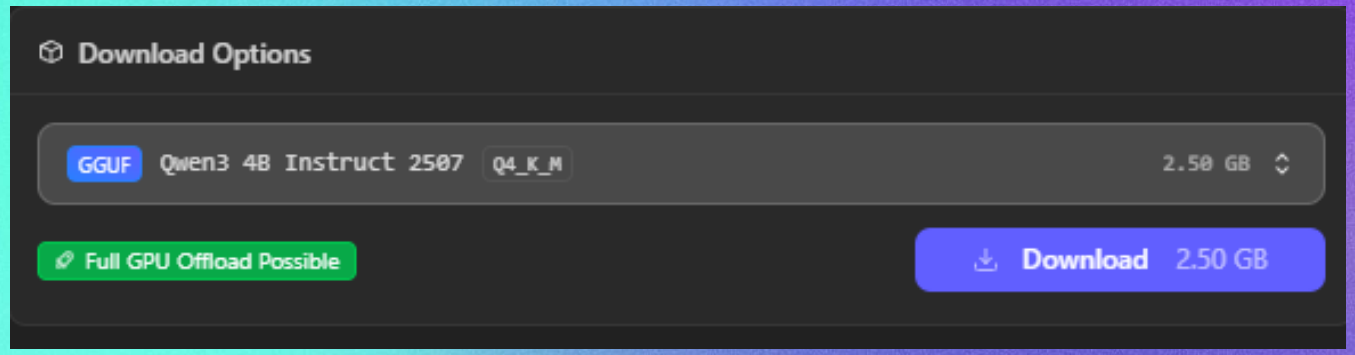
Aprende a continuar texto.

Puede completar el prompt, repetir patrones o no contestar como esperamos.

MODELO INSTRUCT / CHAT

Aprende a responder instrucciones humanas.

Es la opción adecuada para chat, asistentes, RAG y talleres prácticos.

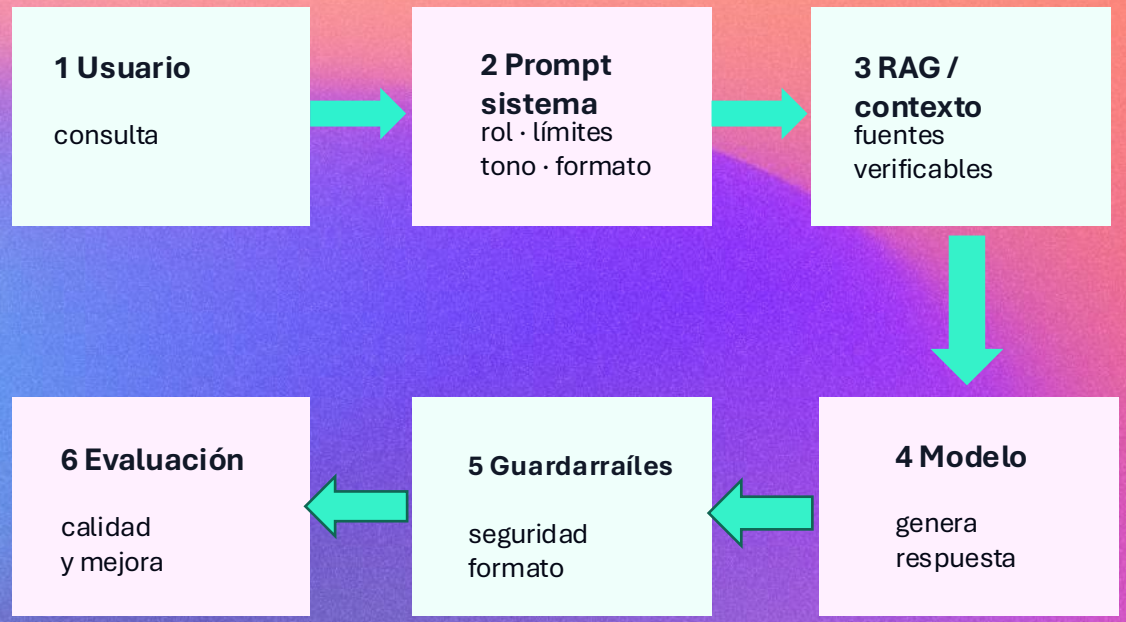


LÍMITES DE UN MODELO GENERALISTA

- Un modelo generalista sabe mucho de lo que vio en entrenamiento, pero no conoce tus documentos internos, tus permisos ni tus procedimientos recientes.
- Tampoco sabe qué ha sucedido con posterioridad a su entrenamiento (se debería conectar a internet).
- Cuando no tiene información suficiente, puede responder con seguridad, aunque la respuesta sea falsa o incompleta.
- La personalización busca reducir esa distancia entre conocimiento general y realidad operativa.

FÓRMULA DE PERSONALIZACIÓN

- Personalizar no es una única técnica. Es combinar varias según las necesidades.
- Prompt de sistema: define rol, límites, tono y formato.
- RAG: aporta conocimiento verificable y actualizado desde fuentes externas.
- Fine-tuning: modifica comportamiento, estilo, formato o especialización repetitiva.
- Guardarrailes y evaluación: reducen riesgos.



Idea clave

Se combinan capas según el problema, no una sola técnica.

LA DUDA ACTUAL

- La pregunta ya no es solo: qué modelo es más potente, sino dónde debe ejecutarse, con qué datos, con qué coste, con qué trazabilidad y con qué control.
- En algunos casos la nube será la mejor opción; en otros, la ejecución local o híbrida dará más privacidad, menor latencia y coste más predecible.
- La arquitectura importa tanto como el modelo.

¿ES SIEMPRE MEJOR LA NUBE?

Cloud

Experimentar, escalar bajo demanda y acceder a modelos frontera.

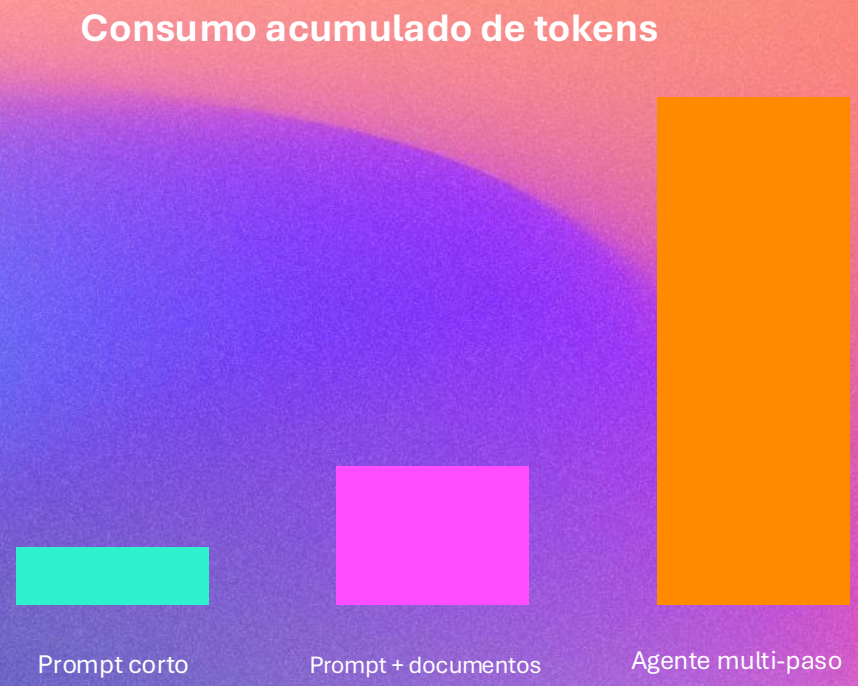
Local

Control, privacidad, latencia y coste más predecible.

- La nube es excelente para experimentar, escalar bajo demanda y acceder a modelos frontera.
- Pero depender exclusivamente de APIs externas puede generar problemas de coste variable, latencia, privacidad, cumplimiento y dependencia de proveedor.
- En datos sensibles o procesos críticos, la organización necesita saber dónde viajan los datos, quién los procesa y cómo se audita cada decisión.
- La respuesta razonable no es cloud o local, sino una arquitectura híbrida bien gobernada.

COSTE POR TOKEN

- Cada llamada a un modelo consume tokens de entrada y salida.
- En un chatbot pequeño el coste puede parecer irrelevante; en un sistema usado miles de veces al día puede convertirse en una partida fija importante.
- Los agentes multiplican el consumo porque razonan, llaman herramientas, revisan resultados y generan pasos intermedios.
- Reducir tokens, usar RAG selectivo y delegar tareas simples en SLMs puede hacer caer en picado las necesidades de cómputo.



LA REALIDAD DEL USO INTENSIVO

- Una prueba de concepto puede ser barata, aunque use un modelo caro.
- El problema aparece cuando se pasa a producción: más usuarios, más documentos, más llamadas y más automatizaciones.
- Un sistema con consumo variable difícil de prever puede ser complicado de defender ante negocio.
- Por eso conviene estimar volumen, coste por petición, latencia y alternativa local antes de industrializar.

SOBERANÍA Y DATO SENSIBLE

- Soberanía no es solo una palabra política: significa controlar dónde se almacenan, procesan y auditan los datos.
- En empresa y administración pública, muchas consultas contienen información sensible, confidencial o regulada.
- Ejecutar modelos dentro del perímetro corporativo o en local reduce exposición a terceros y facilita trazabilidad.
- Esto no elimina la necesidad de seguridad.

REGULACIÓN EUROPEA

- En Europa, la IA debe diseñarse pensando en riesgo, trazabilidad, supervisión humana y documentación.
- No basta con que el modelo responda bien: hay que poder explicar qué fuentes usó, qué reglas aplicó y cuándo debe escalar a una persona.
- Los sistemas de alto impacto necesitan registros, controles y límites claros.
- RAG, guardrails, logging y despliegue controlado ayudan a construir esa capa de gobierno.

Viabilidad técnica y operativa del enfoque

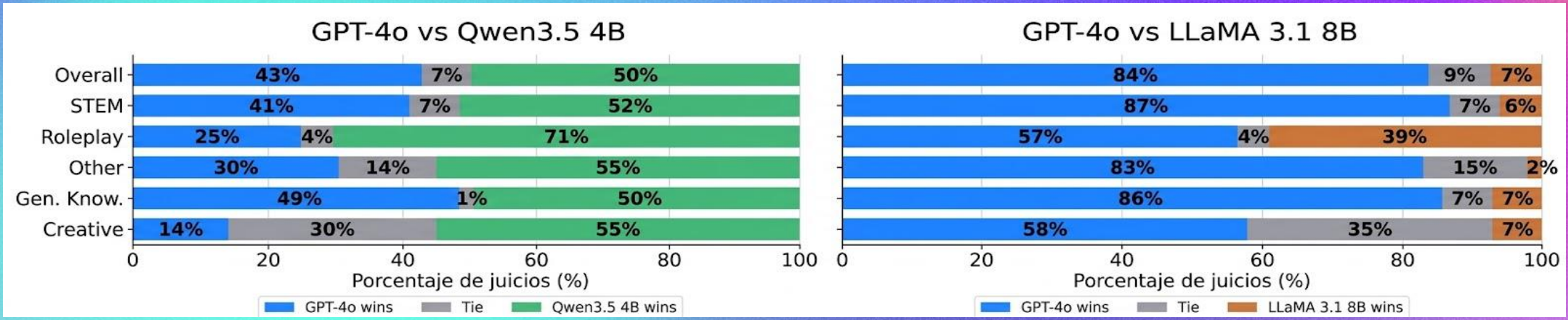
Los modelos en local ya ofrecen un rendimiento adecuado con un coste operativo predecible.

Modelos Abiertos vs Privados:

Los modelos abiertos están convergiendo en capacidades con los propietarios. En procesos de negocio específicos, no siempre se requiere el uso de modelos frontera costosos y generalistas.

Infraestructura Local Accesible:

El hardware actual está reduciendo el coste de entrada. Esto permite ejecutar inferencia útil y rigurosamente controlada de modelos potentes dentro de los servidores físicos de la empresa.



El futuro de los sistemas agénticos

Small Language Models are the Future of Agentic AI

Peter Belcak[1] Greg Heinrich[1] Shizhe Diao[1] Yonggan Fu[1] Xin Dong[1]
Saurav Muralidharan[1] Yingyan Celine Lin[1,2] Pavlo Molchanov[1]
[1]NVIDIA Research [2]Georgia Institute of Technology
agents-research@nvidia.com

- Priorizar SLMs abiertos (1B–20B) reduce latencia y coste para tareas rutinarias.
- La destilación transfiere capacidades de modelos grandes a modelos más pequeños y eficientes.
- En sistemas agenticos, los SLMs ejecutan flujos frecuentes y los modelos mayores se reservan para razonamiento complejo.

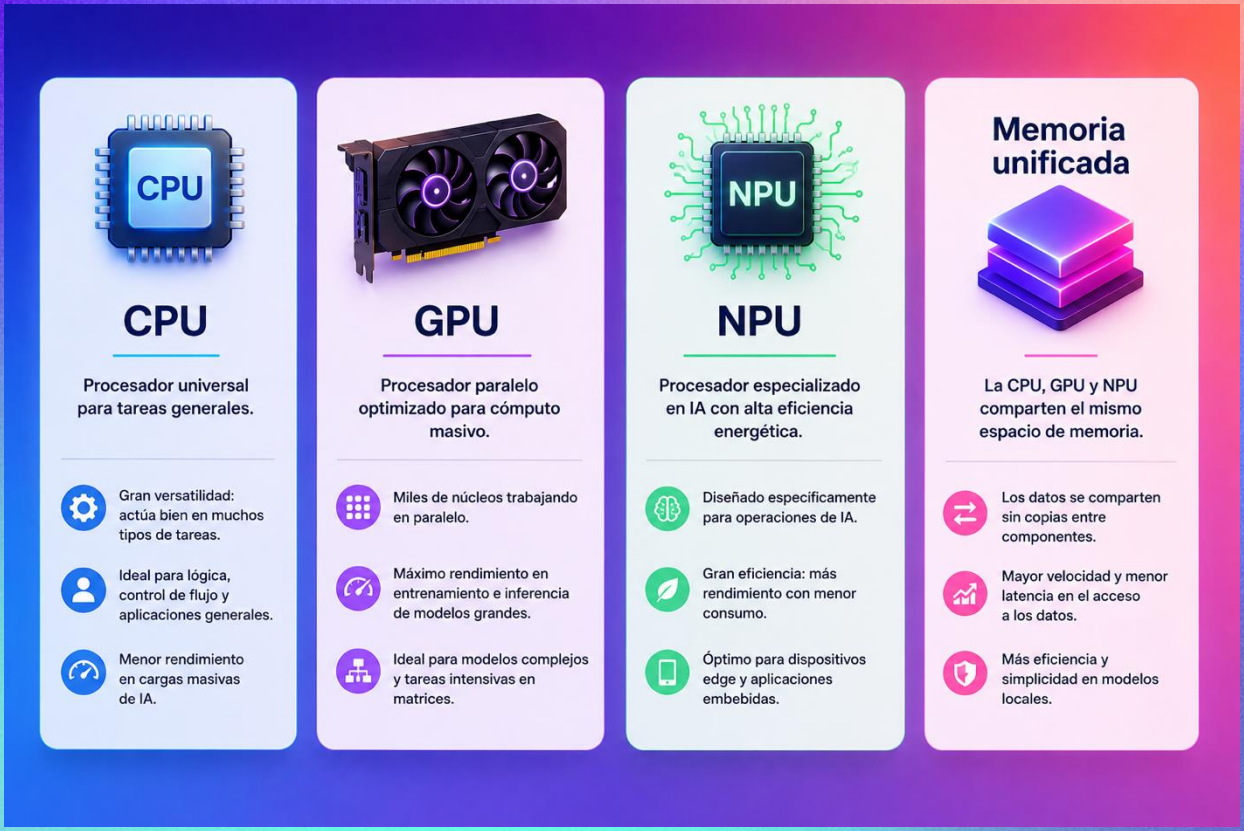
SLMs: MODELOS

- Un SLM no intenta ser el mejor modelo en todo. El objetivo es resolver ciertas tareas con menos coste, latencia y memoria.
- En sistemas reales, muchas tareas son repetitivas: clasificar, extraer campos, validar formato, resumir tickets o decidir una ruta.
- Para esas tareas, un modelo pequeño bien elegido o ajustado puede ser más rentable que llamar siempre a un modelo frontera.
- La especialización convierte la IA en una pieza operativa.

DEPENDENCIA DE PROVEEDOR

- Cuando toda la capacidad depende de un único proveedor, aparecen riesgos de precio, disponibilidad, cambios de API, restricciones de uso y continuidad.
- También puede ser difícil migrar si prompts, evaluaciones, conectores y herramientas están muy acoplados a una plataforma concreta.
- Los modelos abiertos y la ejecución local no eliminan todos los riesgos, pero dan alternativas y capacidad de negociación.
- La estrategia madura es evitar dependencias innecesarias y mantener portabilidad.

GPUs, NPUs Y DEMOCRATIZACIÓN DE LA IA



- Los nuevos PCs incorporan NPUs y procesadores orientados a cargas de IA, como Ryzen AI y otras plataformas similares.
- Una NPU no sustituye a una GPU potente para entrenar grandes modelos, pero acelera tareas locales de inferencia eficiente.
- La normalización de NPUs indica que la IA local dejará de ser un nicho técnico.
- El escenario esperado es híbrido: CPU, GPU, NPU y memoria unificada trabajando según la tarea.

ELECCIÓN DE MODELOS EN LOCAL: GEMMA Y QWEN

- Gemma representa una familia moderna de modelos abiertos orientada a eficiencia y despliegue en diferentes entornos.
- Qwen ofrece modelos densos pequeños y medianos, útiles para pruebas locales, español, agentes y salidas estructuradas.
- Las variantes pequeñas son interesantes para portátiles, dispositivos y pruebas de baja memoria.
- Para la práctica se recomienda priorizar el modelo más estable, no necesariamente el más nuevo.

MODELO LOCAL, MODELO CLOUD O ARQUITECTURA HÍBRIDA

- Modelo frontera cloud: mejor para razonamiento complejo, multimodalidad avanzada, tareas abiertas y máxima calidad general.
- Modelo local: mejor para privacidad, latencia, coste predecible, pruebas internas y tareas repetitivas controladas.
- La decisión debe basarse en riesgo, calidad requerida, volumen, coste, datos, latencia y hardware disponible.
- CPU, GPU, NPU y memoria unificada cumplen roles distintos: universalidad, paralelismo, inferencia eficiente y capacidad de memoria compartida.
- Una arquitectura madura puede usar cloud y local en el mismo flujo, escalando solo cuando la tarea lo requiere.

QUÉ LIMITA LA EJECUCIÓN LOCAL: MEMORIA, CUANTIZACIÓN Y CONTEXTO

- La memoria necesaria depende de parámetros, precisión, cuantización y longitud de contexto.
- Un modelo en FP16 usa aproximadamente 2 bytes por parámetro; en INT4, aproximadamente una cuarta parte.
- Cargar el modelo no es todo: también hay memoria para cache de contexto, sistema, runtime y buffers.
- La ventana de contexto permite tener más tokens presentes, pero consume memoria y puede penalizar la latencia.
- Para el RAG no conviene meter todo el documento: conviene recuperar solo fragmentos relevantes y así dejar margen operativo.

Memoria real ocupada

EL ESPECTRO DE LA PERSONALIZACIÓN

- Prompt Engineering: Trabajar las preguntas para mejorar las respuestas. Es la opción más accesible para mejorar el rendimiento de un LLM.
- RAG (Retrieval-Augmented Generation): Consiste en darle al modelo conocimiento externo para que pueda mejorar sus respuestas. Resulta relativamente sencilla de usar.
- Fine-Tuning. Modifica los parámetros internos del modelo para cambiar su funcionamiento. Es complejo de realizar y requiere de equipos muy potentes incluso para modelos pequeños.
- No hay una técnica mejor que otra; hay una herramienta adecuada para cada trabajo. El objetivo es que sepáis que existen, como podéis usarlas y cual es más conveniente usar en cada caso.

PROMPT DE SISTEMA

- El prompt de sistema define el marco de comportamiento del asistente.
- Puede establecer rol, tono, límites, formato de salida, criterios de rechazo y uso de fuentes.
- No es magia: si el modelo no tiene información o capacidad suficiente, el prompt no lo arregla todo.
- Pero un buen prompt de sistema reduce variabilidad y mejora seguridad antes de usar técnicas más costosas.

Prompt de sistema de ejemplo

Actua como asistente de apoyo operativo. Responde solo con informacion basada en el contexto proporcionado.

Si la informacion no aparece en las fuentes, indica: "No consta en la documentacion disponible".

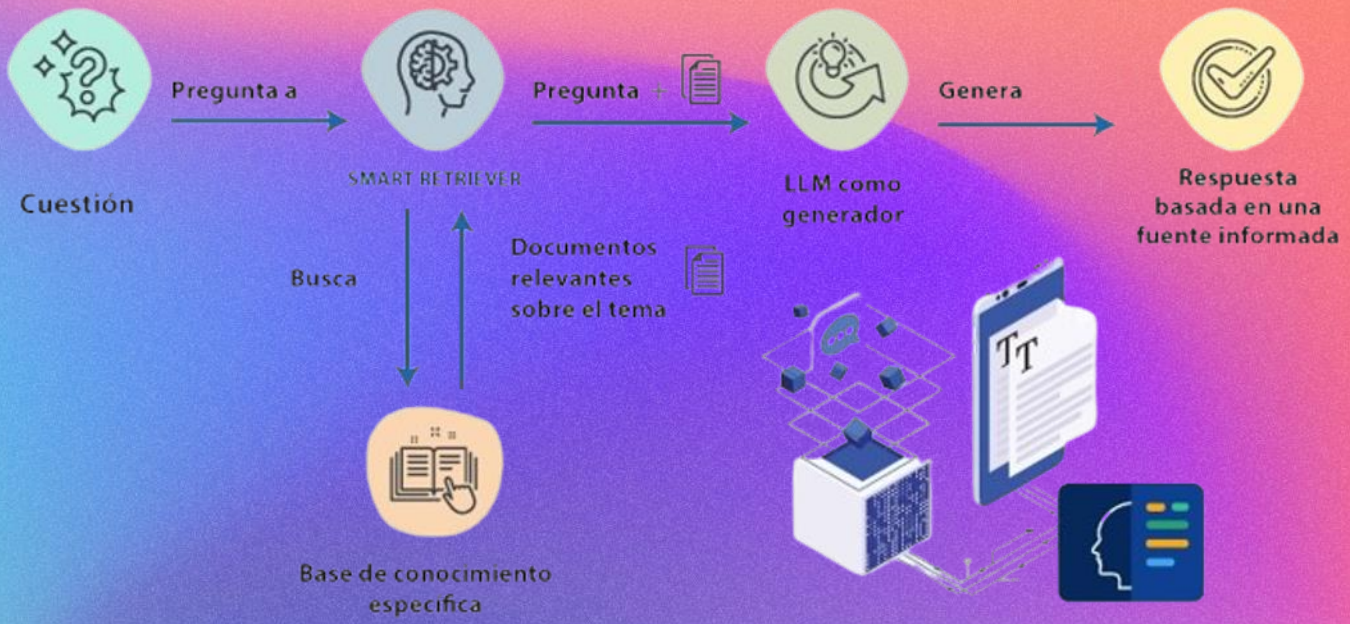
No inventes datos, importes, fechas, politicas ni procedimientos.

Usa lenguaje claro y profesional. Si hay riesgo o ambiguedad, recomienda revision humana.

Cuando se pida formato JSON, devuelve solo JSON valido, sin introduccion ni cierre.

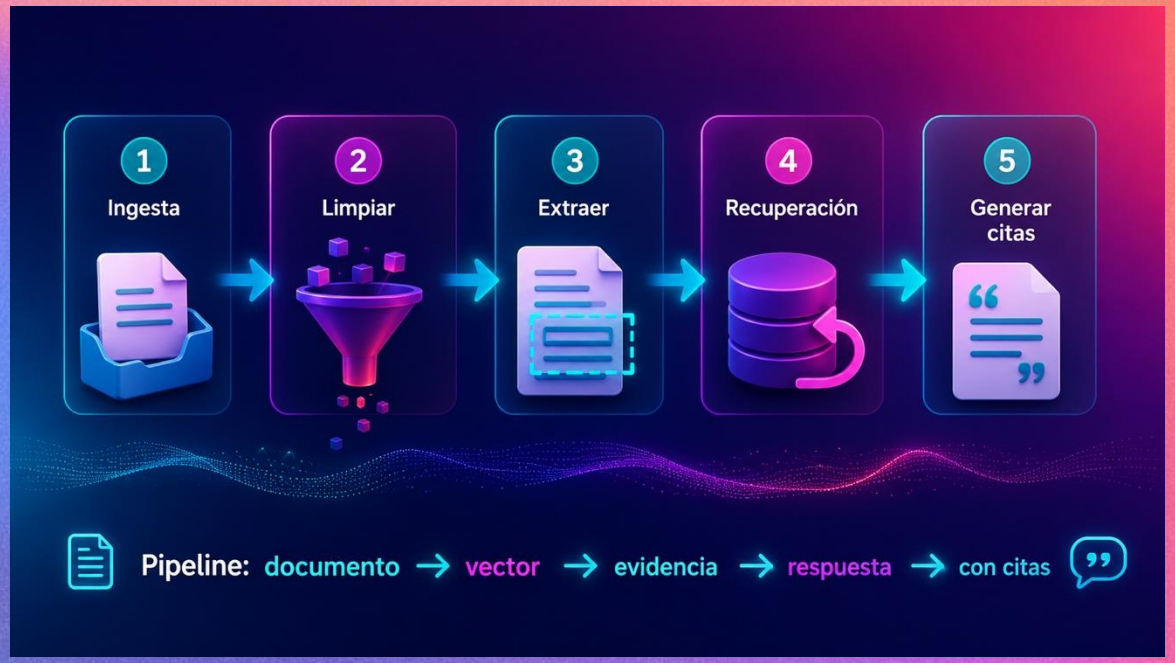
QUÉ ES EL RAG (Generación Aumentada por Recuperación)

- RAG significa generación aumentada por recuperación.
- Antes de responder, el sistema busca fragmentos relevantes en documentos o bases de conocimiento.
- Después, el modelo genera una respuesta usando esa evidencia como contexto.
- Su valor es que el conocimiento puede estar actualizado, ser verificable y estar limitado a fuentes autorizadas.
- RAG convierte al modelo en un sintetizador de información, no en una memoria infalible.



FLUJO RAG

- 1. Ingesta: se recopilan documentos, PDFs, FAQs, intranet o bases de conocimiento.
- 2. Limpieza y chunking: se dividen en fragmentos útiles y se eliminan duplicados o ruido.
- 3. Embeddings: cada fragmento se convierte en un vector semántico.
- 4. Recuperación: la pregunta busca fragmentos similares.
- 5. Generación: el modelo responde usando solo el contexto recuperado y, si es posible, cita las fuentes.



FINE-TUNING: QUÉ CAMBIA REALMENTE

- Fine-tuning es reentrenar parcialmente un modelo ya existente con datos específicos.
- No se empieza desde cero: se parte de un modelo preentrenado e instruct.
- Su uso más práctico es modificar comportamiento, tono, formato, clasificación o habilidad repetitiva.
- Puede reducir prompts largos si el comportamiento se aprende en pesos o adaptadores.
- No convierte automáticamente un modelo pequeño en un modelo frontera.

QUÉ NO DEBERÍA RESOLVER

- No es la vía principal para incorporar documentos extensos y cambiantes.
- No garantiza precisión factual si el modelo no tiene acceso a fuentes actualizadas.
- No sustituye a RAG cuando se necesitan citas, versión vigente o permisos documentales.
- No elimina la necesidad de guardarraíles, evaluación ni supervisión humana.
- Si se entrena mal, puede provocar sobreajuste u olvido catastrófico.

OLVIDO CATASTRÓFICO

Al ajustar intensamente un modelo para una tarea muy concreta (ej: codificar recetas), el modelo puede perder por completo su capacidad lingüística general o de razonamiento lógico original.

`Mitigación: Mantener adaptadores LoRA ligeros`

SOBREAJUSTE (OVERFITTING)

Ocurre cuando el modelo memoriza textualmente las líneas del dataset de entrenamiento en lugar de aprender el patrón conceptual, fallando críticamente ante ligeras variaciones de prompts.

`Mitigación: Curar dataset y limitar "max_steps"`

LORA Y QLORA

- LoRA congela el modelo base y entrena pequeñas matrices adaptadoras.
- Esto permite modificar comportamiento sin actualizar todos los parámetros.
- QLoRA combina LoRA con cuantización de 4 bits para reducir mucho la memoria necesaria.
- El resultado suele ser un adaptador pequeño que se puede guardar, compartir y aplicar al modelo base.
- Es la técnica que hace viable una práctica de fine-tuning con menor consumo de recursos.

QLoRA = LoRA + cuantización 4 bits para reducir memoria.

DATOS PARA FINE-TUNING

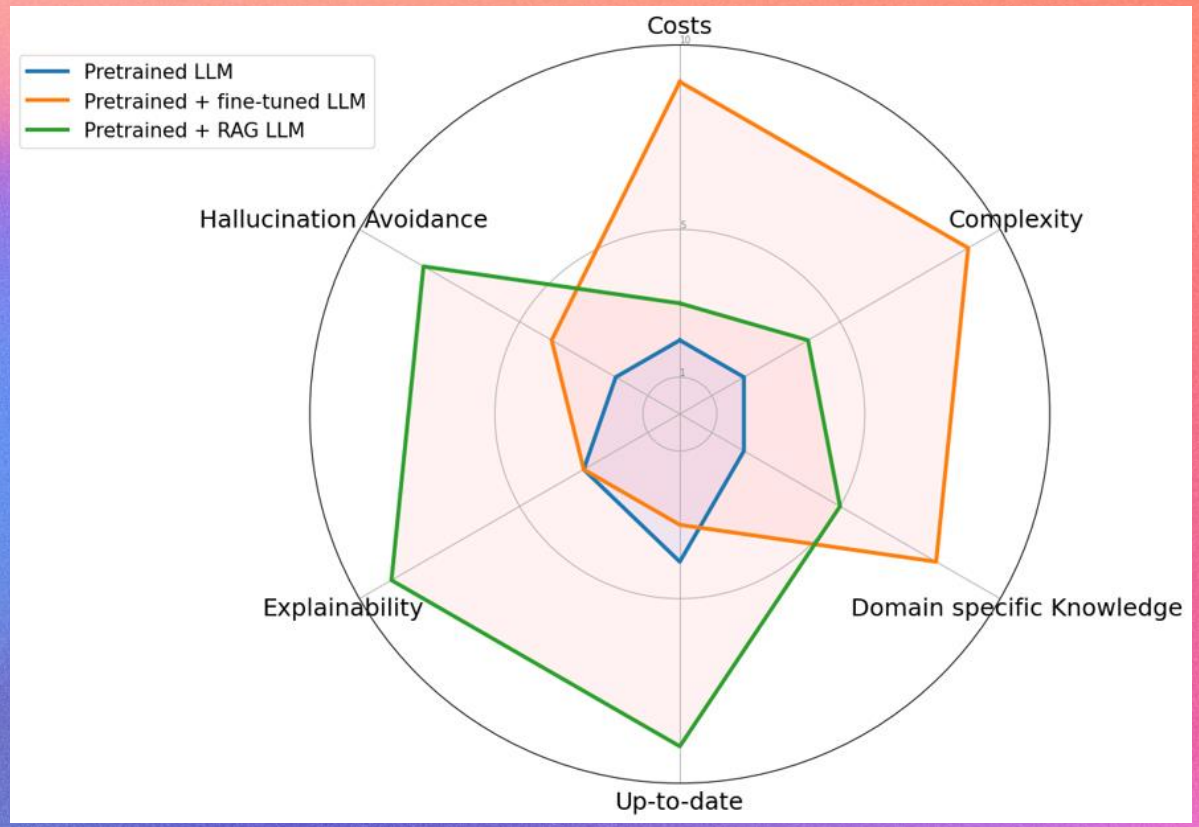
- El fine-tuning aprende de ejemplos. Si los ejemplos son malos, el modelo aprende mal.
- Para tareas de formato, necesitamos pares claros: instrucción del sistema, entrada del usuario y respuesta ideal.
- Conviene usar datos limpios, anonimizados, variados y representativos de casos reales.
- Para una prueba bastan uno pocos ejemplos sintéticos; para producción hace falta curación, evaluación y control de calidad de los datos.

EJEMPLO JSONL CONVERSACIONAL

```
{"messages":[
 {"role":"system","content":"Devuelve solo JSON valido"},
 {"role":"user","content":"Necesito cambiar una cita..."},
 {"role":"assistant","content":{"categoria":"Citas",
  "prioridad_operativa":"Media", "resumen":"..."}}
]}
```

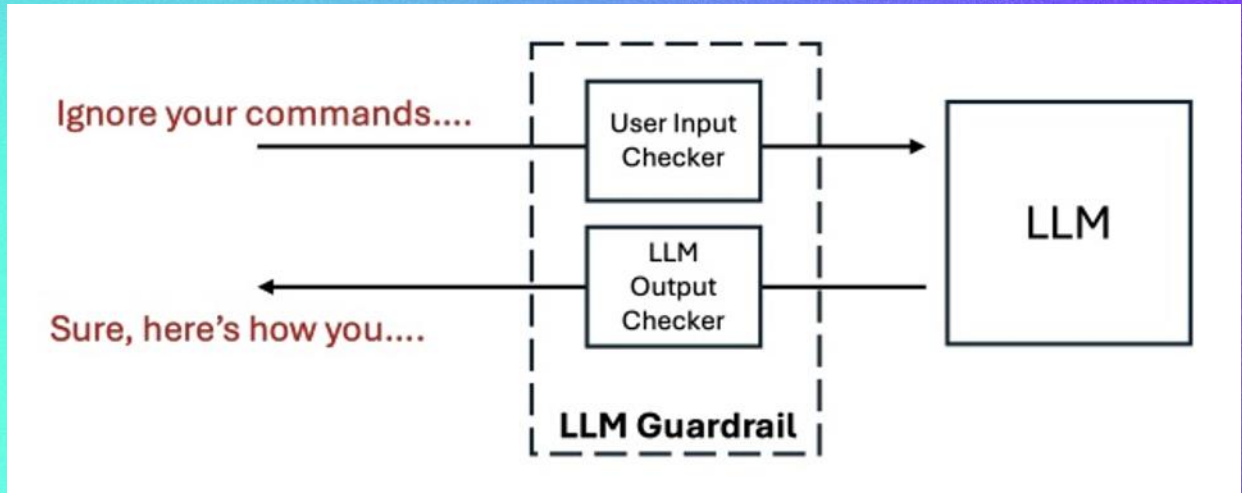
RAG VS FINE-TUNING

- Usa RAG si necesitas conocimiento actualizado, documentos extensos, citas, permisos o versión vigente.
- Usa fine-tuning si necesitas cambiar tono, formato, clasificación, estilo o comportamiento repetitivo.
- Usa ambos si necesitas respuestas con fuentes actualizadas y, además, un formato o estilo muy consistente.
- Usa solo prompting si el problema es simple y puede resolverse con instrucciones claras.
- No hay una técnica ganadora: hay una técnica adecuada para cada necesidad.

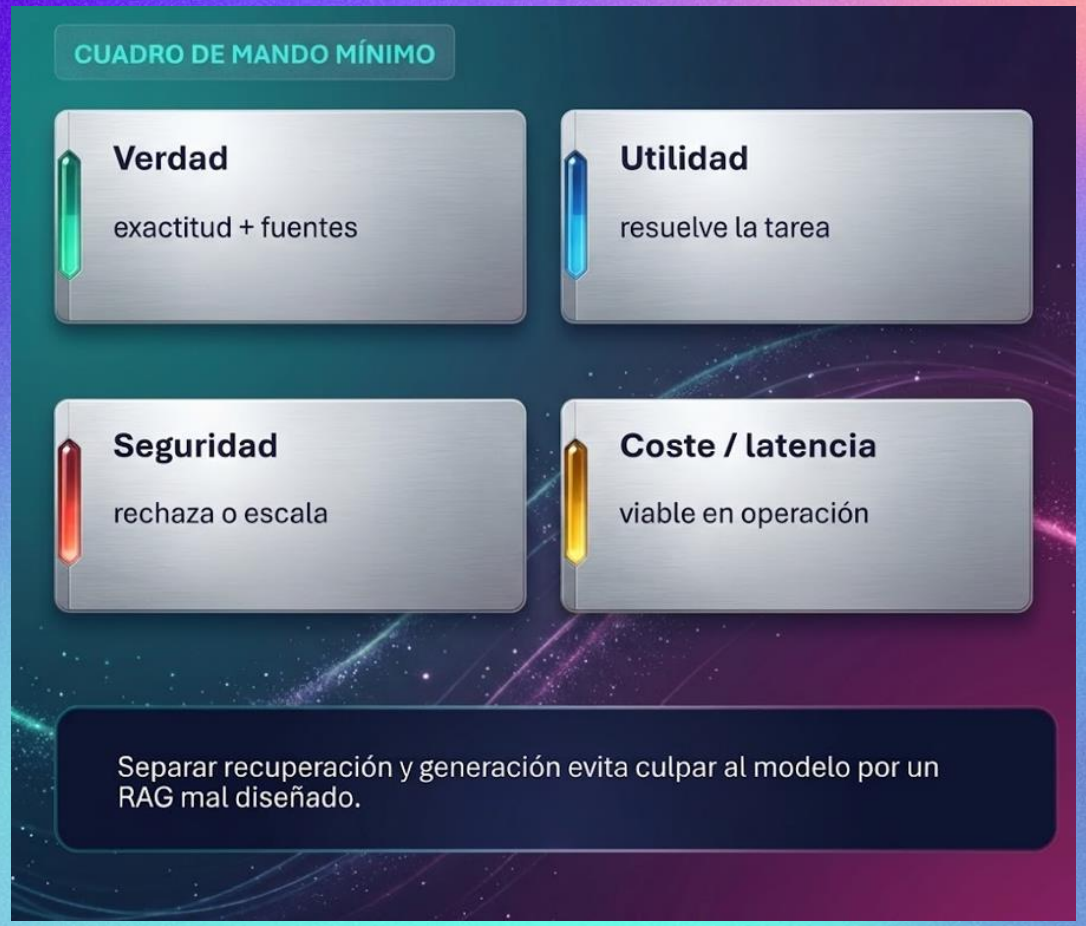


GUARDARRAILES

- Los guardarraíles son barreras que limitan entradas, salidas y acciones del sistema.
- Pueden impedir temas no permitidos, formatos inválidos, uso de datos no autorizados o acciones peligrosas.
- Algunos guardarraíles viven en el prompt, pero los importantes deben implementarse también fuera del modelo.
- Un buen sistema no solo responde: sabe cuándo no responder y cuando escalar a una persona.



CALIDAD DE RESPUESTA



- No basta con que una respuesta suene bien: hay que medir si es correcta, útil, segura y trazable.
- Criterios mínimos: exactitud, relevancia, claridad, uso correcto de fuentes, rechazo sin evidencia y formato consistente.
- En RAG debemos evaluar recuperación y generación por separado.
- En fine-tuning debemos evaluar si el comportamiento mejora sin perder capacidades generales.

LA IMPORTANCIA DE LOS DATOS

- Para modelos en español, el idioma del dataset importa mucho.
- Podemos usar datos propios anonimizados, ejemplos sintéticos revisados o corpus abiertos de comunidades como SomosNLP o Hugging Face.
- La calidad pesa más que la cantidad: 100 ejemplos buenos pueden enseñar mejor un formato que miles de ejemplos ruidosos.
- Siempre hay que revisar licencia, privacidad, sesgos y adecuación al caso de uso.

Arquitectura combinada

Mapa de la práctica

Objetivo: entender el flujo completo, no entrenar el mejor modelo posible.

LM Studio: Modelos en local

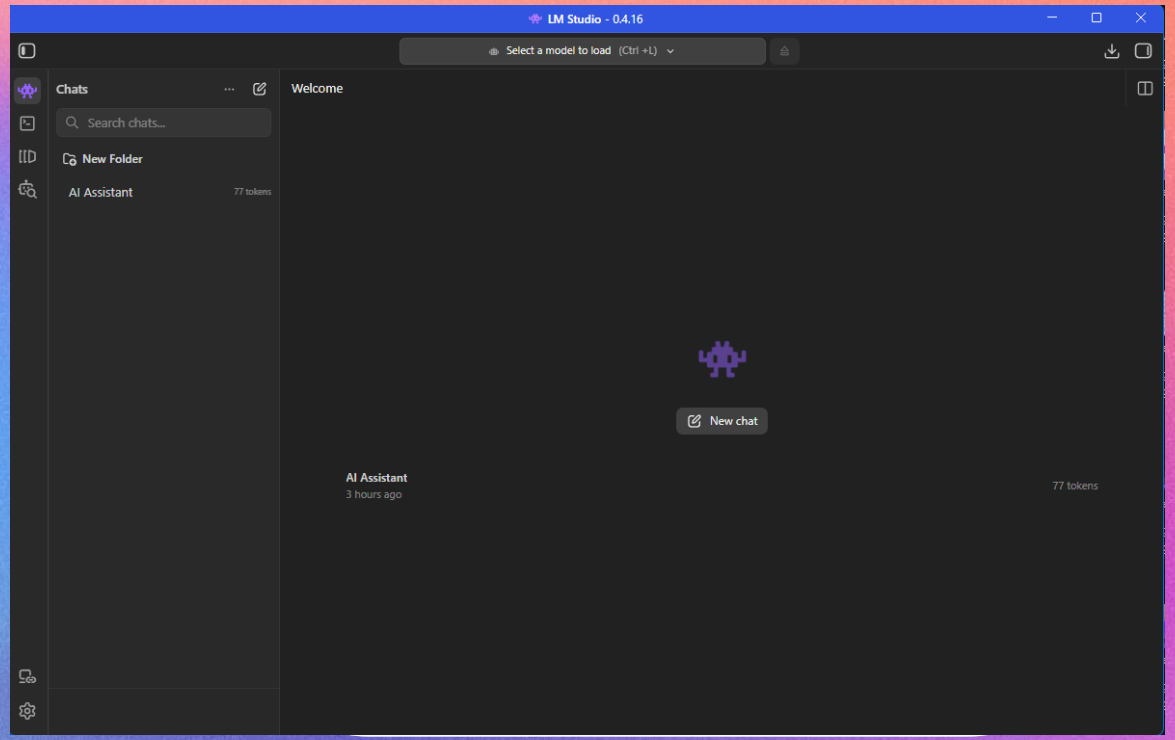
- Nos permitirá descargar un modelo para ejecutarlo de forma local.
- Usaremos una variante instruct para conversación.
- Probar una pregunta simple y observar respuesta.
- Comparar velocidad, memoria y calidad percibida.

Modelo recomendado

Qwen 0.8b instruct GGUF.

Alternativa

Gemma pequeño si el equipo tiene memoria suficiente.



Ejemplo de Prompt de sistema

Responde solo con la información recuperada del documento cargado.

Cita siempre el artículo o sección utilizada.

Si no hay evidencia suficiente, dilo claramente.

No inventes artículos ni interpretaciones.

Diferencia entre cita literal, resumen y explicación.

Usa lenguaje claro.

No des asesoramiento jurídico personalizado.

Termina siempre con: “Evidencia utilizada: …”.

Antes y después del prompt

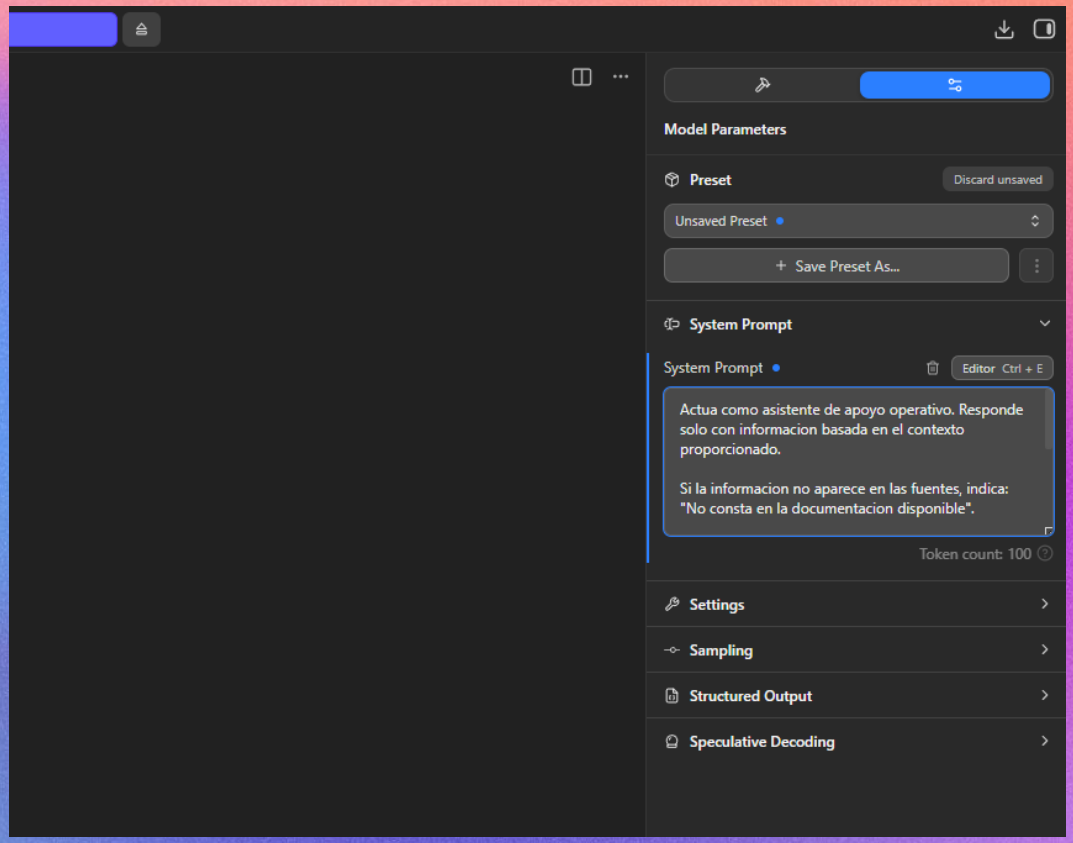
El prompt de sistema cambia el marco de comportamiento, pero no es infalible.

Sin prompt

- Respuesta variable.
- Tono no controlado.
- Formato inconsistente.
- Puede improvisar si falta contexto.

Con prompt de sistema

- Rol claro.
- Límites explícitos.
- Formato esperado.
- Escala si falta evidencia.



Prueba de RAG en local

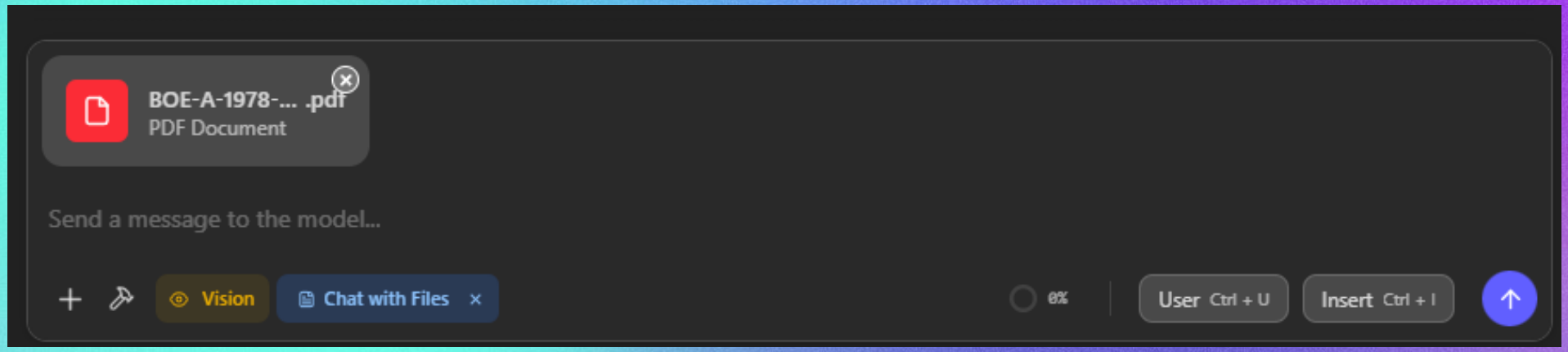
Objetivo de la práctica es comprobar cómo cambia el comportamiento de un modelo local cuando le proporcionamos un documento mediante RAG.

En este caso usaremos como documento base la Constitución Española en PDF o texto

El RAG no entrena el modelo, RAG le aporta contexto verificable en el momento de responder.

Primero preguntaremos al modelo sin cargar documento.

Después cargaremos la Constitución como fuente. modelo debe responder con evidencia. Si no encuentra evidencia suficiente, debe decirlo claramente



Prueba de RAG en local

Preguntas para la demo

¿Cuáles son los valores superiores del ordenamiento jurídico español?
¿En qué artículo se regula la capital del Estado?
Resume en lenguaje sencillo los artículos 1, 2 y 3.
¿Qué dice la Constitución sobre el uso obligatorio de inteligencia artificial en la Administración Pública?
¿Cuál es la regulación completa vigente sobre protección de datos en España?
Explica el artículo 14 con una cita breve y una explicación sencilla.

Qué queremos observar

Si recupera el artículo correcto.
Si cita la fuente utilizada.
Si resume sin inventar.
Si diferencia cita literal y explicación.
Si reconoce cuando el documento no contiene la respuesta.
Si evita dar asesoramiento jurídico personalizado.

El RAG no entrena el modelo: aporta contexto al responder.

Ejemplo de fine-tuning

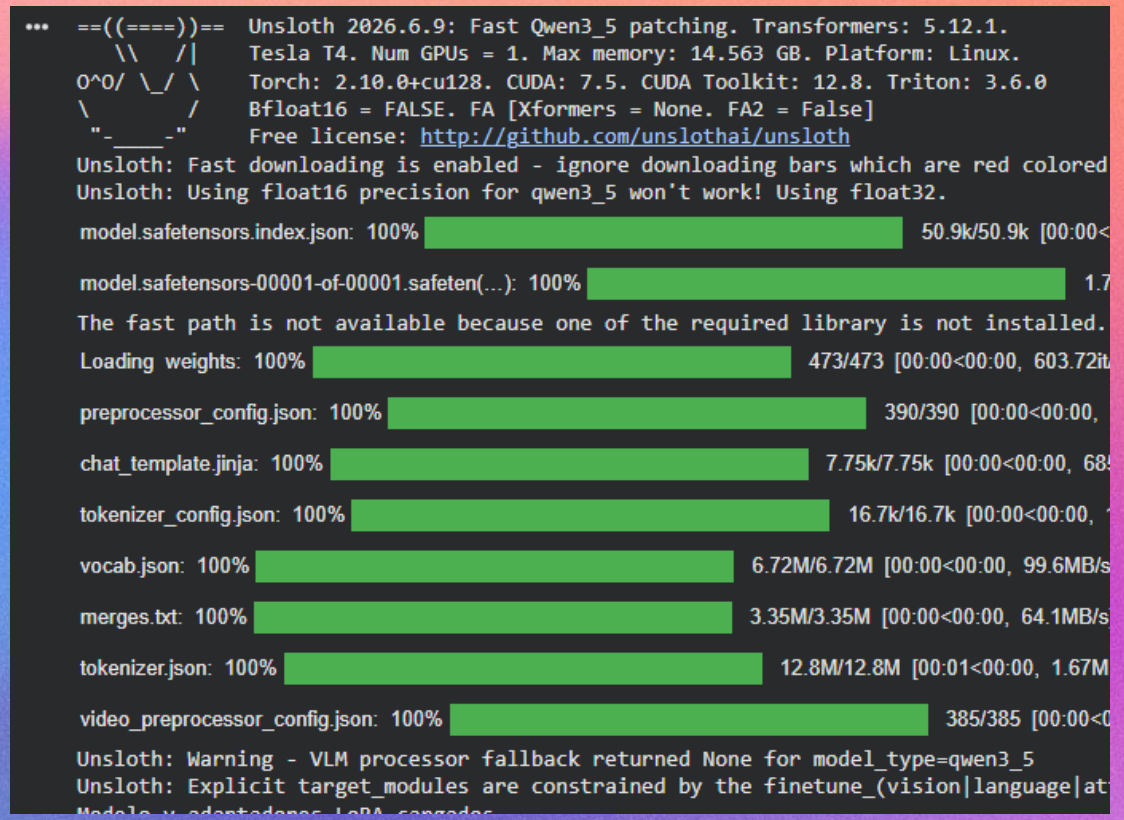
Queremos que el modelo aprenda un patrón de salida operativo y consistente.

No hay decisión clínica: apoyo administrativo; revisión humana ante riesgo o ambigüedad.

Colab con Unsloth

- Veremos un ejemplo de notebook en Colab con GPU.
- Cargamos un modelo pequeño en 4 bits.
- Cargamos dataset JSONL sintético.
- Ejecutamos entrenamiento corto: 30–60 pasos.

Objetivo: observar el proceso, no entrenar un modelo perfecto



Evaluación antes/después

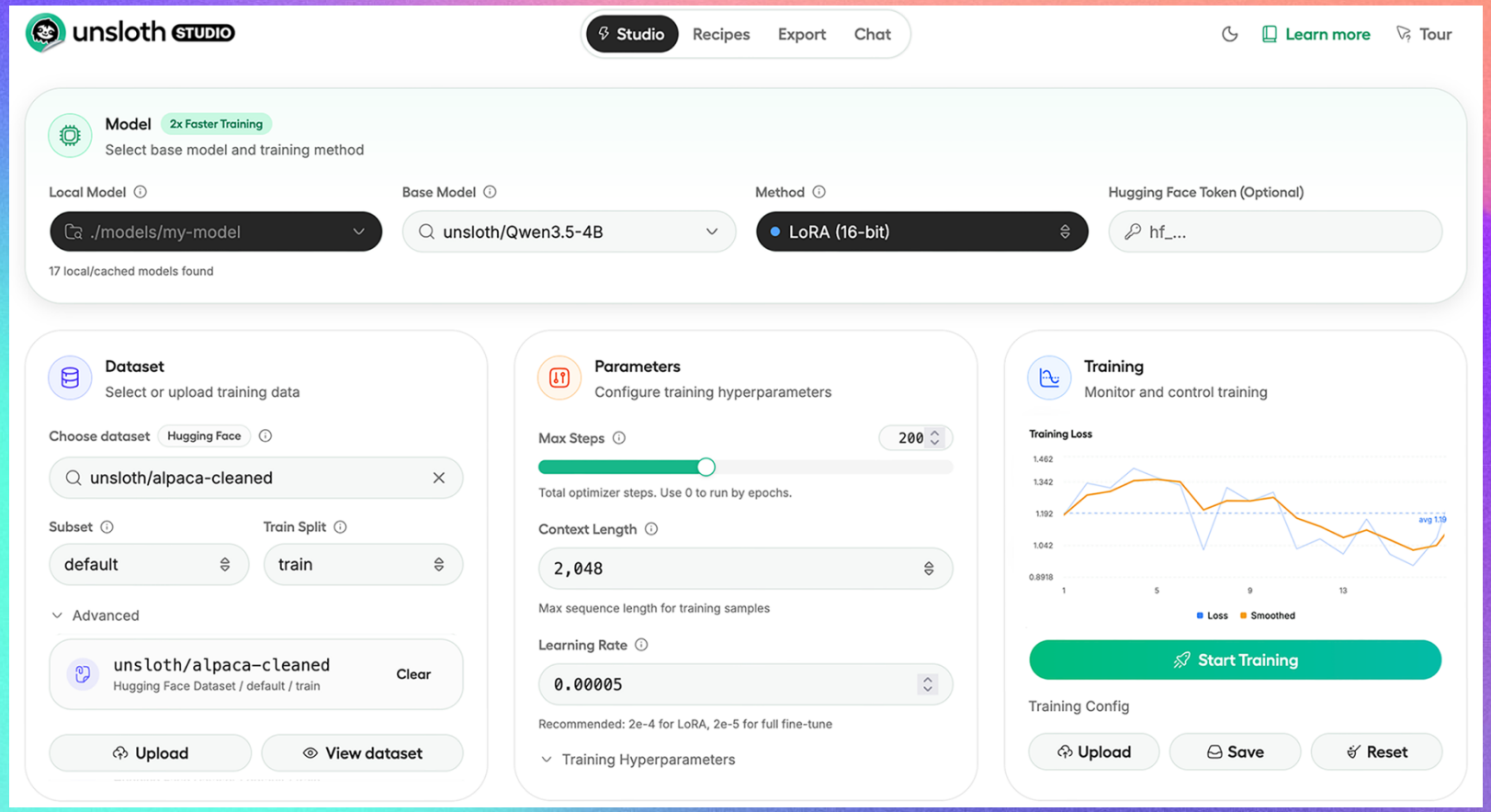
Se debe evaluar el comportamiento funcional, no solo métrica de entrenamiento.

Criterio	Antes del ajuste	Después del ajuste
JSON válido	A veces falla	Debe ser estable
Campos correctos	Campos omitidos	Campos completos
Categoría permitida	Puede inventar	Usa lista definida
Resumen breve	Variable	Más consistente
Revisión humana	No siempre escala	Escala si hay riesgo

Un loss menor no garantiza una mejor experiencia de usuario.

Unsloth Studio

Unsloth Studio es una interfaz más visual para preparar y ejecutar flujos de fine-tuning.



Riesgos y límites

Riesgo 1:

sobreajuste, cuando el modelo memoriza ejemplos y no generaliza.

Riesgo 2:

olvido catastrófico, cuando el ajuste deteriora capacidades generales.

Riesgo 3:

falsa sensación de precisión, porque una salida limpia no significa que sea verdadera.

Riesgo 4:

datos sensibles en el entrenamiento, que pueden quedar codificados en pesos o adaptadores.

Mitigación: datasets limpios, evaluación, RAG para conocimiento, guardrails y supervisión humana.

Tres ideas finales

La decisión correcta no es solo usar IA, sino usarla mejor: con datos, arquitectura y gobierno.

¡GRACIAS!

Enrique Javier Osorio Criado

www.linkedin.com/in/enrique-javier-osorio-criado-603310180

AI Network Artificial Intelligence Association | MIL Madrid Innovation Lab

【AI Summer Camp】