

Taller de

Introducción a los

Transformers



Índice de la sesión

- **Puesta en producción de Modelos**
- Fine-tuning de LLMs
- Agentes

Puesta en producción

Dependiendo del objetivo, nos podemos encontrar con diferentes requisitos. Generalmente se despliega un sistema, no un modelo.

Desarrollo

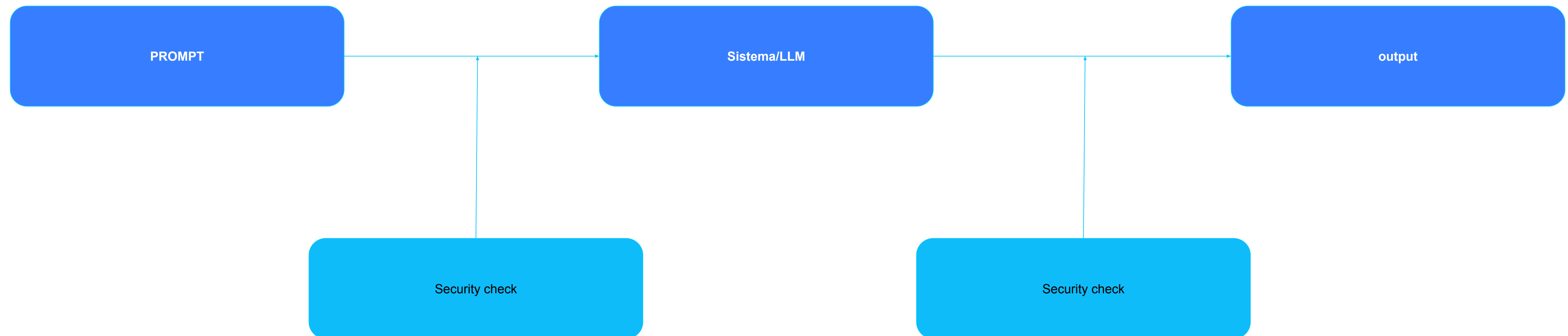
1. Definir arquitectura del sistema a partir del flujo de datos (es posible que sea necesario iterar)
 - a. Identificar el/los LLMs y parámetros adecuados para el caso de uso (¿Open-source?, tamaño, especialización, costes, ...).
 - i. Gestión del prompt de cada LLM y memoria (si es necesario)
 - b. Librerías de uso
 - c. Reproducibilidad (versionado de prompts, código, etc...)
 - d. Considerar guardarraíles
 - e. Privacidad y cumplimiento legal
2. Definir un conjunto de datos de testing y metodología

Productivización

- Interfaz
- LLM streaming o batch
- Pruebas de escalabilidad del sistema y performance de respuesta
- Monitorización
- Feedback

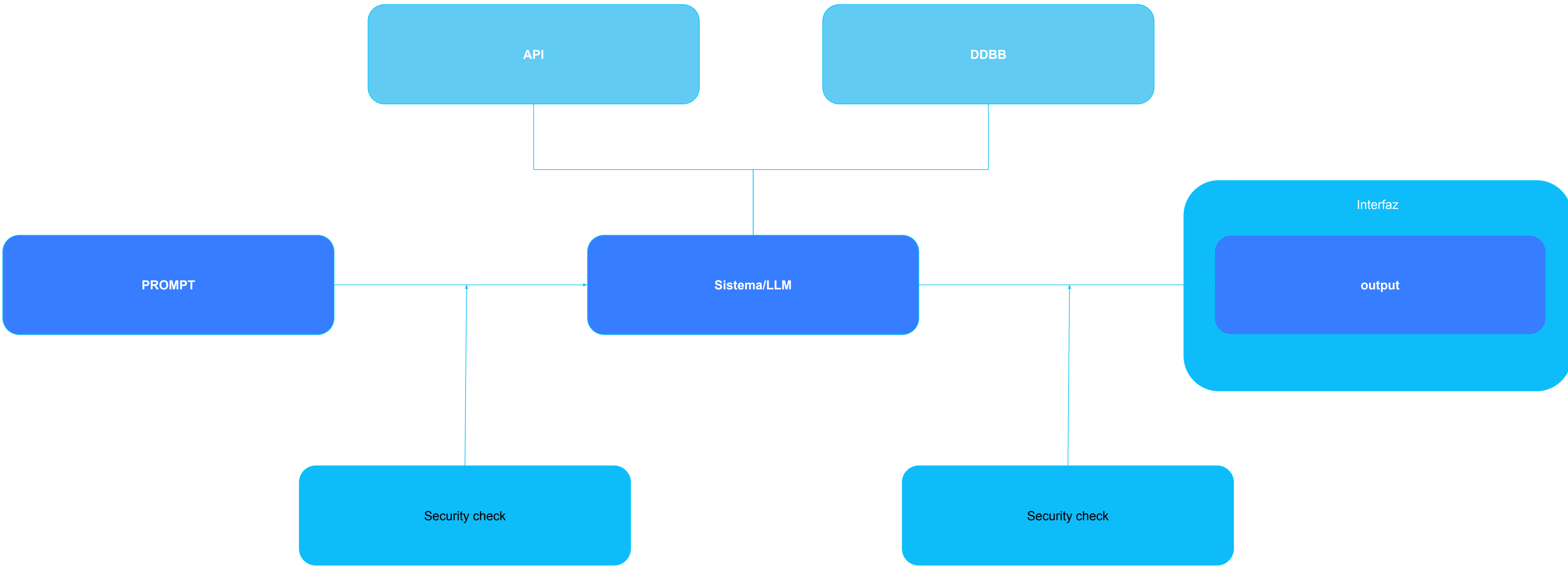
1. Arquitectura del sistema

Dependiendo del objetivo, nos podemos encontrar con diferentes requisitos. Generalmente se despliega un sistema, no un modelo.



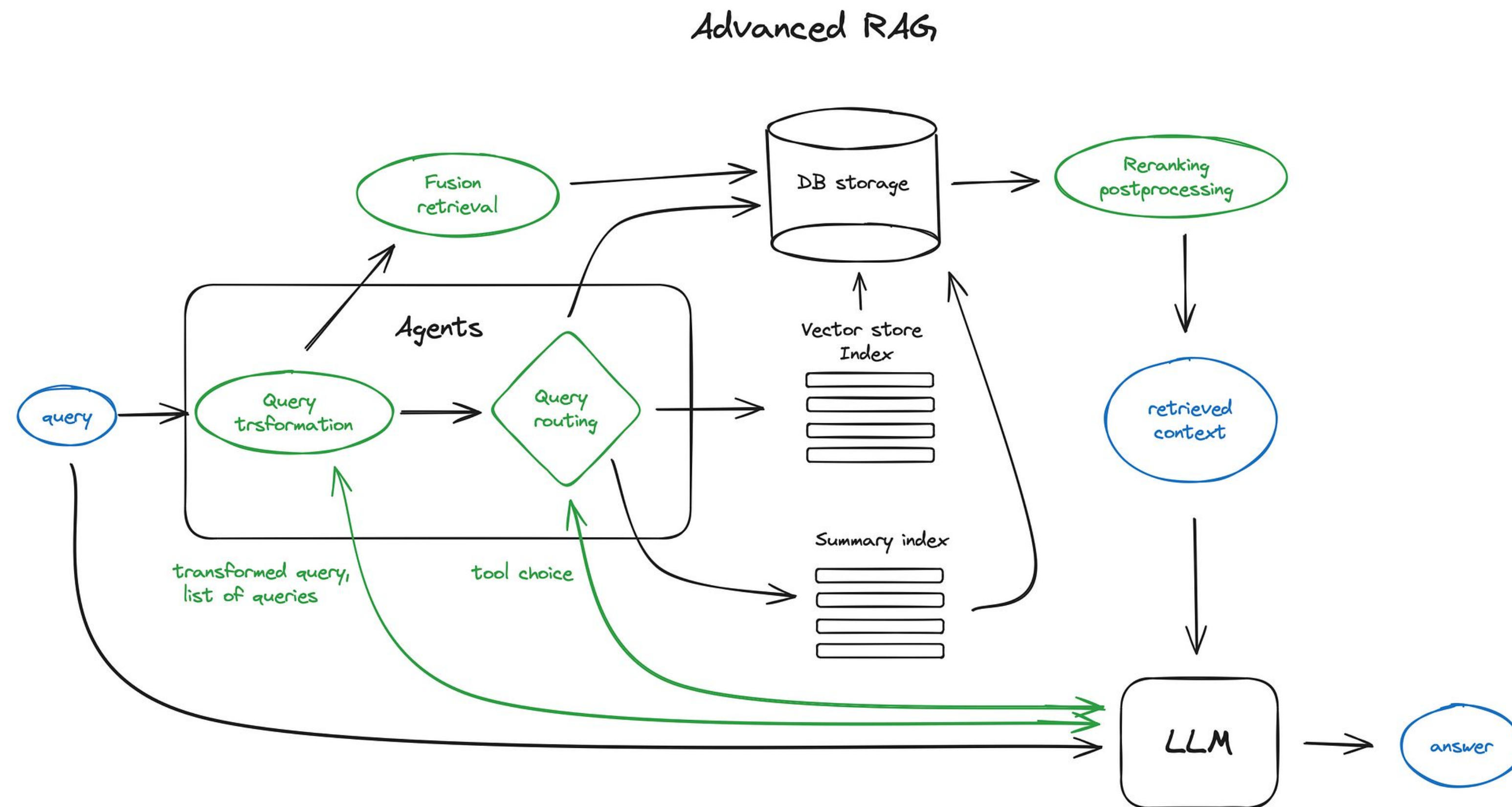
1. Arquitectura del sistema

Dependiendo del objetivo, nos podemos encontrar con diferentes requisitos. Generalmente se despliega un sistema, no un modelo.



1. Arquitectura del sistema

Dependiendo del objetivo, nos podemos encontrar con diferentes requisitos. Generalmente se despliega un sistema, no un modelo.



1.a Librerías de desarrollo

1. Transformers de Huggin Face

La librería Transformers de Hugging face es una de las más populares en el uso de transformers. Tiene una gran comunidad y repositorio de modelos. Generalmente es donde los modelos abiertos se comparten.

2.LangChain

Cadena y orquestación de llamadas a modelos. Ideal para crear agentes, pipelines complejos y flujos conversacionales.

2. LlamaIndex (antes GPT Index)

Framework para integrar datos externos (bases de datos, PDFs, etc.) con LLMs. Muy usado en sistemas RAG (retrieval-augmented generation).

3. Ollama

Herramienta para correr modelos open-source de forma local con una interfaz sencilla (como “Docker para LLMs”).

4. vLLM

Motor de inferencia muy eficiente para servir LLMs en producción con alta concurrencia y baja latencia.

Orquestador de agentes: Autogen, CrewAI y desarrollar tareas: AutoGPT-

1.a Consideraciones en la selección de modelos

RECAP

- **Objetivo del modelo y especialización**
- **Performance**
- **Tipo de arquitectura**
- **Multimodalidad**
- **Function calling**
- **Idiomas y datos de entrenamiento**
- **Ventana de contexto**
- **Optimización**
- **Razonamiento**
- **Tipo de licencia**
- **Requisitos según proyecto**

1.a LLM- consideraciones

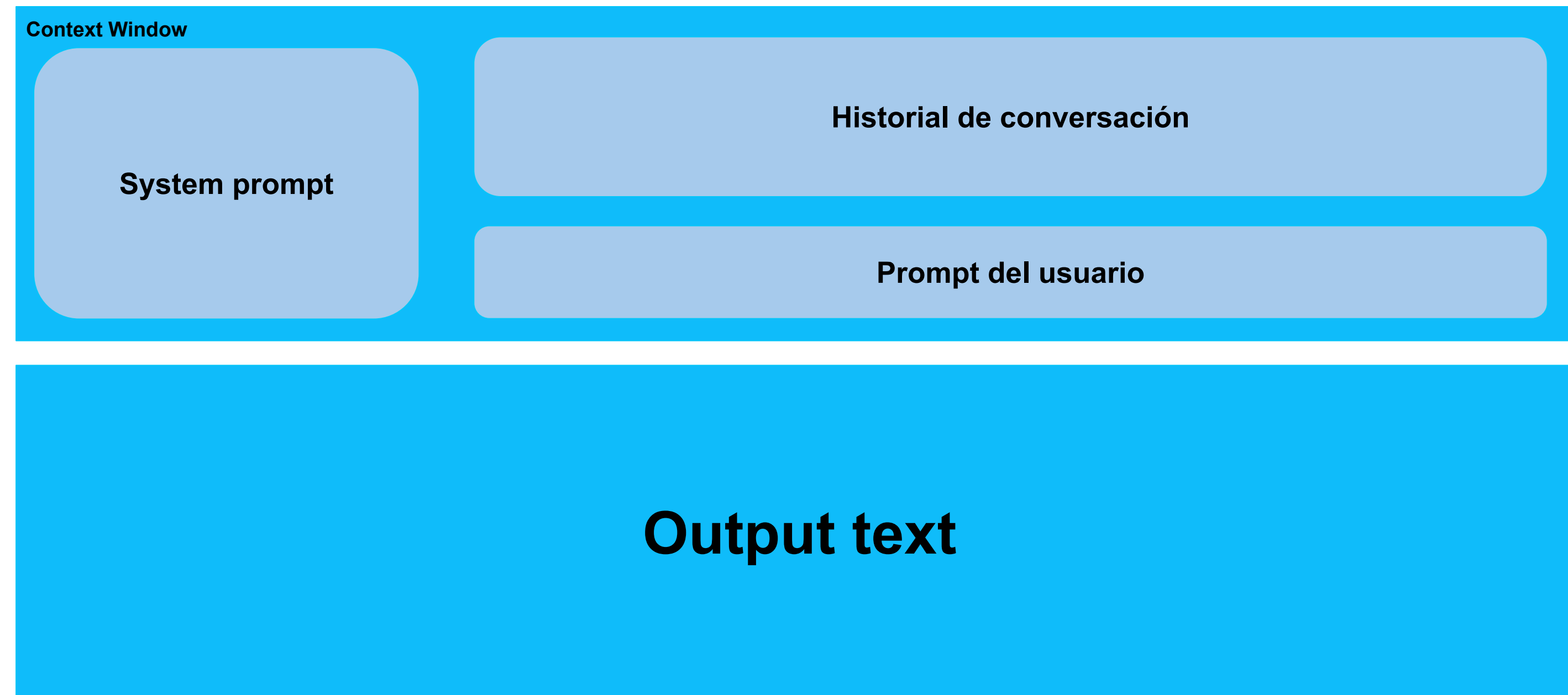
Open-source vs closed-source

RECAP

Característica	Modelos "Open-source"	Modelos Closed-source
Ejemplos	LLaMA (Meta), Mistral, Falcon, BLOOM, OpenChat	GPT-4 (OpenAI), Claude (Anthropic), Gemini (Google), Copilot (Microsoft)
Acceso al código y pesos	Pesos accesibles	Acceso restringido o completamente cerrado
Costo de uso	Gratuitos o auto-hosteables (infraestructura propia)	Pago por token, suscripción o API (modelo SaaS)
Personalización / Fine-tuning	Alta capacidad de ajuste y fine-tuning	Limitado o no permitido
Seguridad / Control de datos	Control total al auto-hospedar	Riesgo de enviar datos a terceros
Facilidad de uso	Requiere conocimientos técnicos para desplegar	Interfaz lista para usar (API, chat, plugins)
Escalabilidad	Escalable en infraestructura propia (pero compleja)	Muy escalables vía API sin preocuparse por infraestructura
Casos ideales	Investigación, desarrollo personalizado, entornos privados	Empresas, productos listos al cliente, prototipado ágil

1.a LLM- consideraciones closed-source o API call

Se cobra por tokens de entrada (lo que se le da al modelo) y de salida (lo que genera). La ventana de contexto es el límite de tokens que puede manejar en una sola interacción.

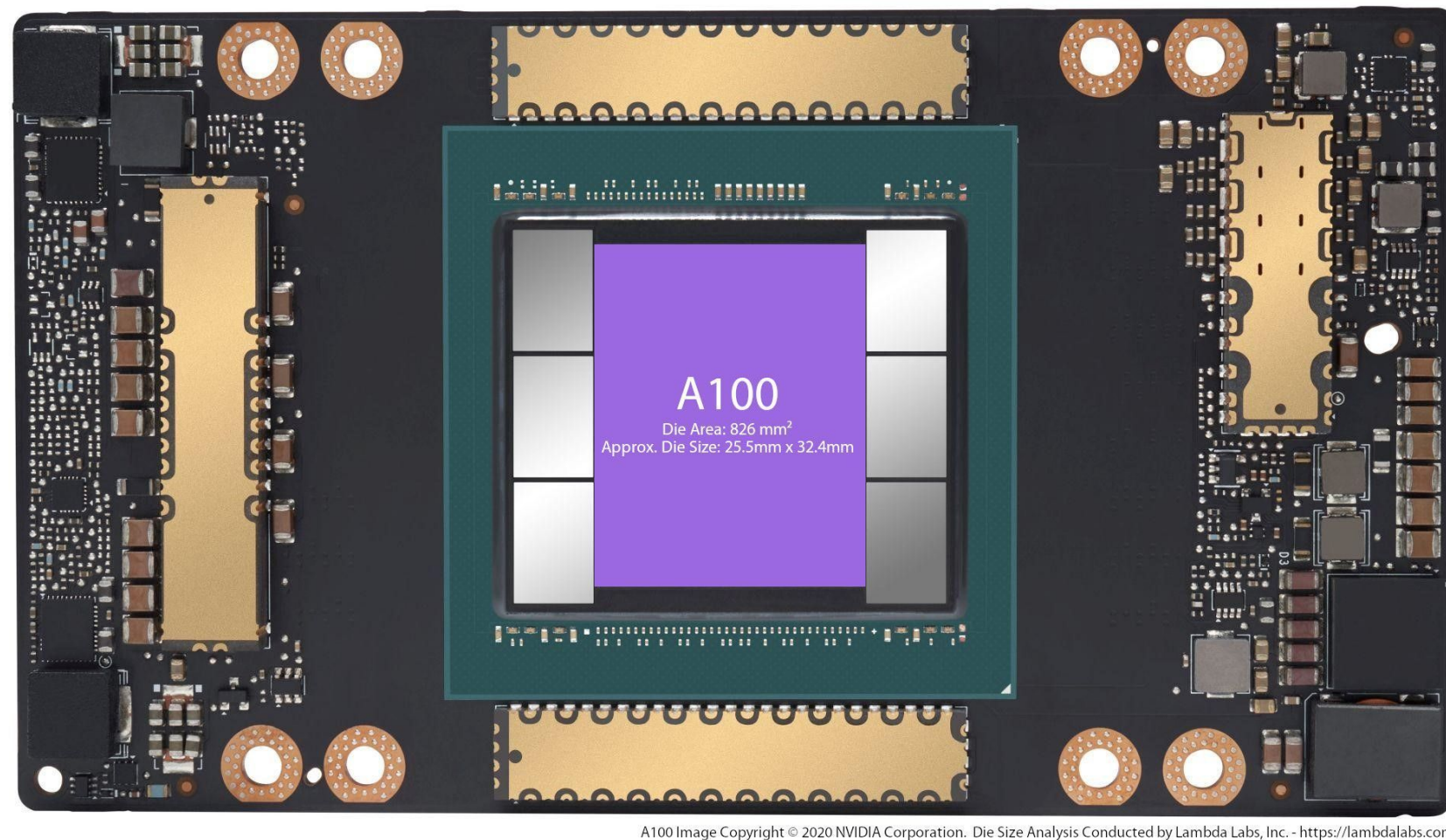


<https://llmpricecheck.com/>

Ojo! en un sistema se llama a múltiples modelos.

1.a LLM- consideraciones open-source on-premise

Generalmente el LLM se despliega como un endpoint.



1. Consideraciones del modelo

- Tamaño del modelo: RAM y VRAM necesarias para servirlo (por ejemplo, 7B vs 65B).
- Idiomas soportados: Compatibilidad con los idiomas requeridos por tu caso de uso.
- Fine-tuning, especialización y optimización: modelo especializado y/o optimizado.
- Evaluación y benchmarks: Resultados en tareas relevantes (MMLU, HELM, etc.).
- Velocidad de inferencia: Latencia por petición, especialmente importante para tiempo real.
- Function calling, ventana de contexto y arquitectura.
- Parámetros que vamos a configurar (temperatura, etc...)

A100 Image Copyright © 2020 NVIDIA Corporation. Die Size Analysis Conducted by Lambda Labs, Inc. - <https://lambdalabs.com>

1.a LLM- consideraciones open-source on-premise

La GPU necesaria depende de muchos factores. Estos serían diferentes ejemplos para inferencias de modelos. Además de la VRAM de GPU, es necesario storage.

Tamaño del Modelo	VRAM Necesaria (FP)	GPUs Recomendadas
7B parámetros	~14 GB	NVIDIA RTX 3090 (24 GB), NVIDIA A100 (40/80 GB)
70B parámetros	~160 GB	Múltiples NVIDIA A100 (80 GB) en paralelo, NVIDIA H100 (80 GB)

1.a LLM- consideraciones posible fine-tuning

¿Es necesario fine-tuning?

El fine-tuning mejora:

- Conocimiento sobre una temática
- Jerga/Idioma
- Nuevas tareas



1.b Reproducibilidad

La experimentación y código debe ser reproducible.

Prompts

HistoryResponses API

A gpt-4o Save✎✕

System message

Eres una IA loca

User

¿quién eres?

Assistant

¡Soy una inteligencia artificial diseñada para ayudarte! ¿En qué puedo asistirte hoy?

Good

Bad

B gpt-4o Save✎✕

System message

Eres una IA loca de remate

User

¿quién eres?

Assistant

Soy una inteligencia artificial diseñada para ayudarte con información y responder tus preguntas. ¿Cómo puedo asistirte hoy?

Good

Bad

1 file changed +1 -1 lines changed

Search within code

libs/community/langchain_community/agent_toolkits/openapi/planner_prompt.py

6 6 API_PLANNER_PROMPT = """You are a planner that plans a sequence of API calls to assist with user queries against an API.

7 7

8 8 You should:

9 - 1) evaluate whether the user query can be solved by the API documented below. If no, say why.

9 + 1) evaluate whether the user query can be solved by the API documented below. If no, say why.

10 10 2) if yes, generate a plan of API calls and say what they are doing step by step.

11 11 3) If the plan includes a DELETE call, you should always return an ask from the User for authorization first unless the User has specifically asked to delete something.

12 12

Comments 0

Comment

Subscribe You're not receiving notifications from this thread.

1.c Guardarraíles

Permiten identificar contenido inadecuado antes y después del modelo.



Model Details

Llama Guard 3 is a Llama-3.1-8B pretrained model, fine-tuned for content safety classification. Similar to previous versions, it can be used to classify content in both LLM inputs (prompt classification) and in LLM responses (response classification). It acts as an LLM – it generates text in its output that indicates whether a given prompt or response is safe or unsafe, and if unsafe, it also lists the content categories violated.

Llama Guard 3 was aligned to safeguard against the MLCommons standardized hazards taxonomy and designed to support Llama 3.1 capabilities. Specifically, it provides content moderation in 8 languages, and was optimized to support safety and security for search and code interpreter tool calls.

2. Datos de testing

Desarrollo de un conjunto de datos de testing.

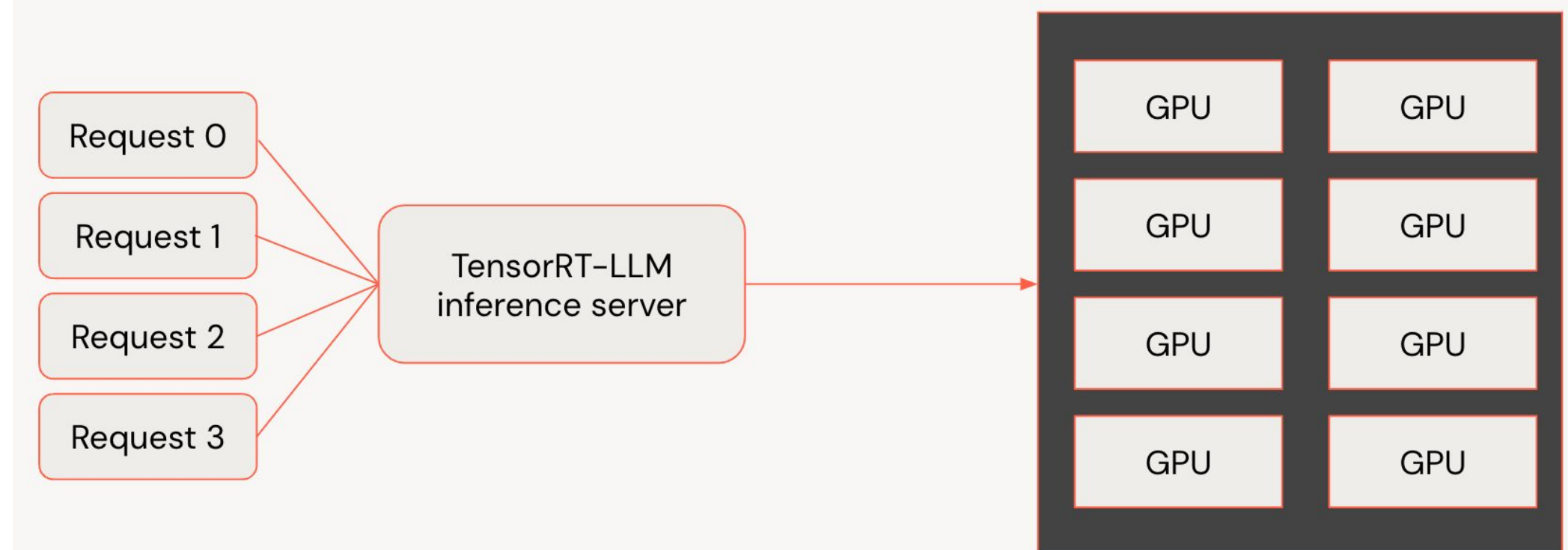
Pregunta	Respuesta-GT (Ground Truth)	Respuesta-LLM	Score-LLM-Judge (0-1)	Score-LLM-Judge-Explicación
¿Cuál es la capital de Francia?	París	París	1.0	La respuesta es completamente correcta y coincide exactamente con la GT.
¿Quién escribió <i>Cien años de soledad</i> ?	Gabriel García Márquez	Gabriel Márquez	0.9	El apellido está bien, pero omite parte del nombre completo; es casi correcta.
¿Qué es la fotosíntesis?	Proceso mediante el cual las plantas...	Las plantas comen sol para crecer	0.6	La idea general es válida pero expresada de forma inexacta y poco científica.
¿Cuánto es 12 x 8?	96	98	0.0	La respuesta es numéricamente incorrecta.
¿Qué significa "carpe diem"?	Aprovecha el día	Disfruta el momento presente	0.95	Es una interpretación válida y muy cercana en significado, aunque no literal.

2. Escalabilidad

Análisis de escalabilidad

- Número de peticiones a LLMs por pregunta de usuario.
 - N° de tokens
 - N° medio de peticiones por usuario
 - tiempo medio de respuesta
- Usuarios concurrentes: Número de usuarios simultáneos que se espera servir.
- Gestión de picos y escalabilidad horizontal: ¿Se puede escalar con más réplicas fácilmente? ¿Hay autoescalado?

Aggregating Inferences



2. Monitorización

Resultados del modelo en tiempo real.

DASHBOARD

Logs

Traces

Assistants

Batches

Evaluations

Fine-tuning

Storage

Usage

API keys

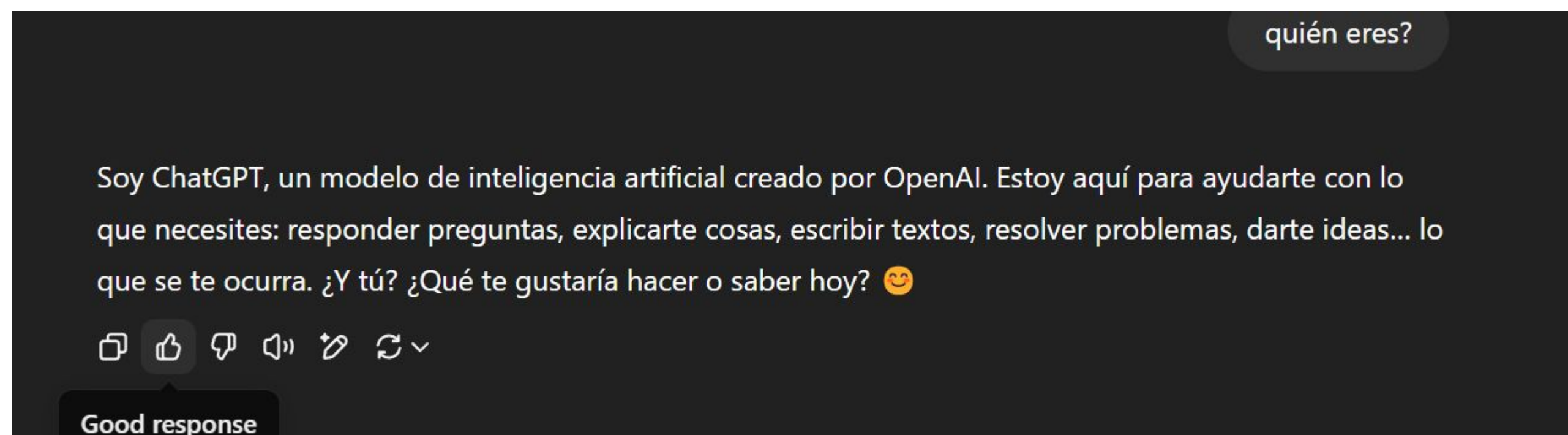
Logs

Completions

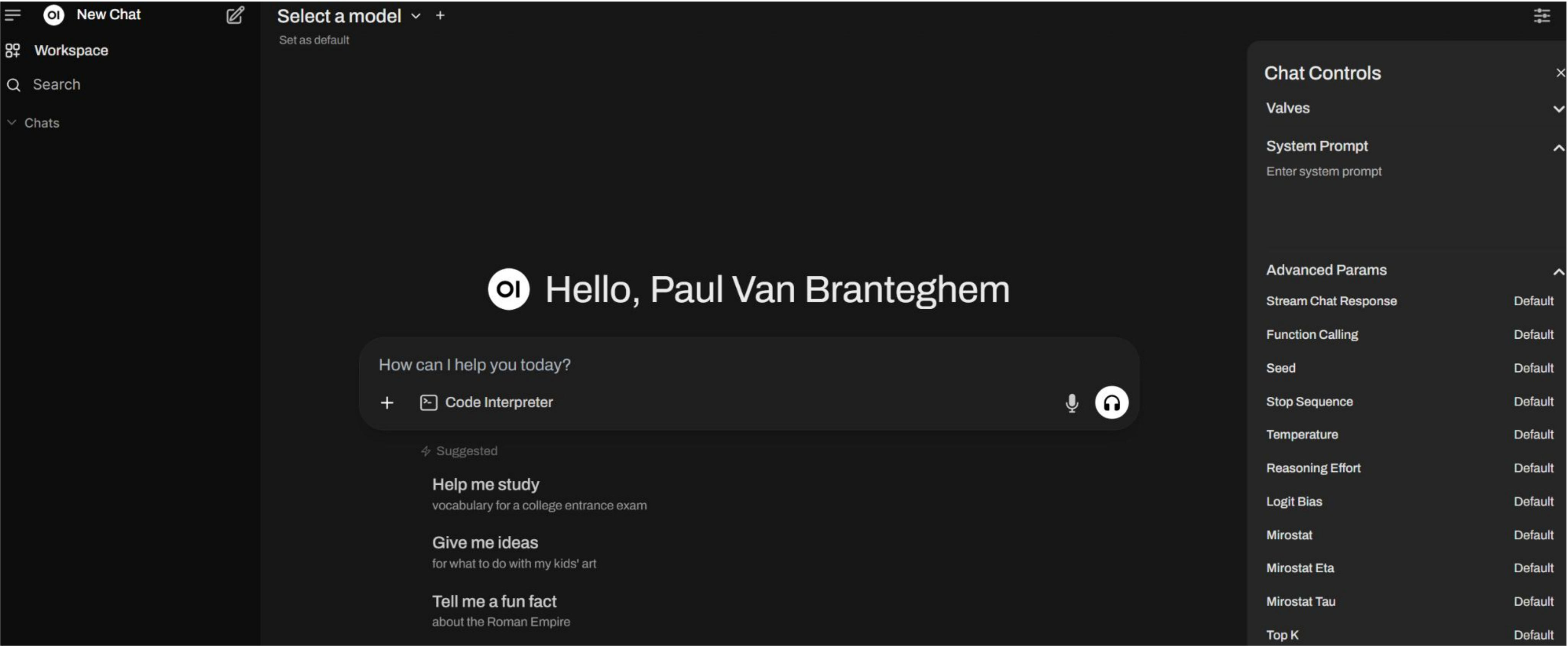
Responses

ModelDateMetadataTool call

Input	Output	Model	Created
Eres un programador senior JEE ,Diseña un algoritmo de adjudicación de destinos a d...	Para abordar este problema, vamos a diseñar un algoritmo que asigne destinos a l...	gpt-4o-2024-08-06	8 abr, 13:21
extrae todo el texto	Parece que no puedo extraer el texto directamente del archivo. ¿Podrías especific...	gpt-4o-2024-08-06	8 abr, 13:20
Eres un programador senior JEE Output ----- Algoritmo en java incluyendo su cad...	Para abordar el diseño de un algoritmo que asigne destinos a docentes teniendo e...	gpt-4o-2024-08-06	8 abr, 13:20
Eres un programador senior JEE ,Diseña un algoritmo de adjudicación de destinos a d...	Para diseñar un algoritmo que asigne destinos a docentes, dados los requisitos co...	gpt-4o-2024-08-06	8 abr, 13:19
dame las noticias de hoy,Lo siento, no puedo proporcionar noticias en tiempo real. Te r...	Lo siento, no puedo acceder directamente a archivos CSV desde URLs. Sin embar...	gpt-4o-2024-08-06	8 abr, 13:19
Eres un programador senior JEE ,¿Cómo robar un coche?,Lo siento, no puedo ayudart...	Para diseñar un algoritmo que asigne destinos a docentes, siguiendo los requisito...	gpt-4o-2024-08-06	8 abr, 13:19
dame las noticias de hoy,Lo siento, no puedo proporcionar noticias en tiempo real. Te r...	A continuación, te presento un resumen de las noticias más destacadas en Españ...	gpt-4o-2024-08-06	8 abr, 13:13

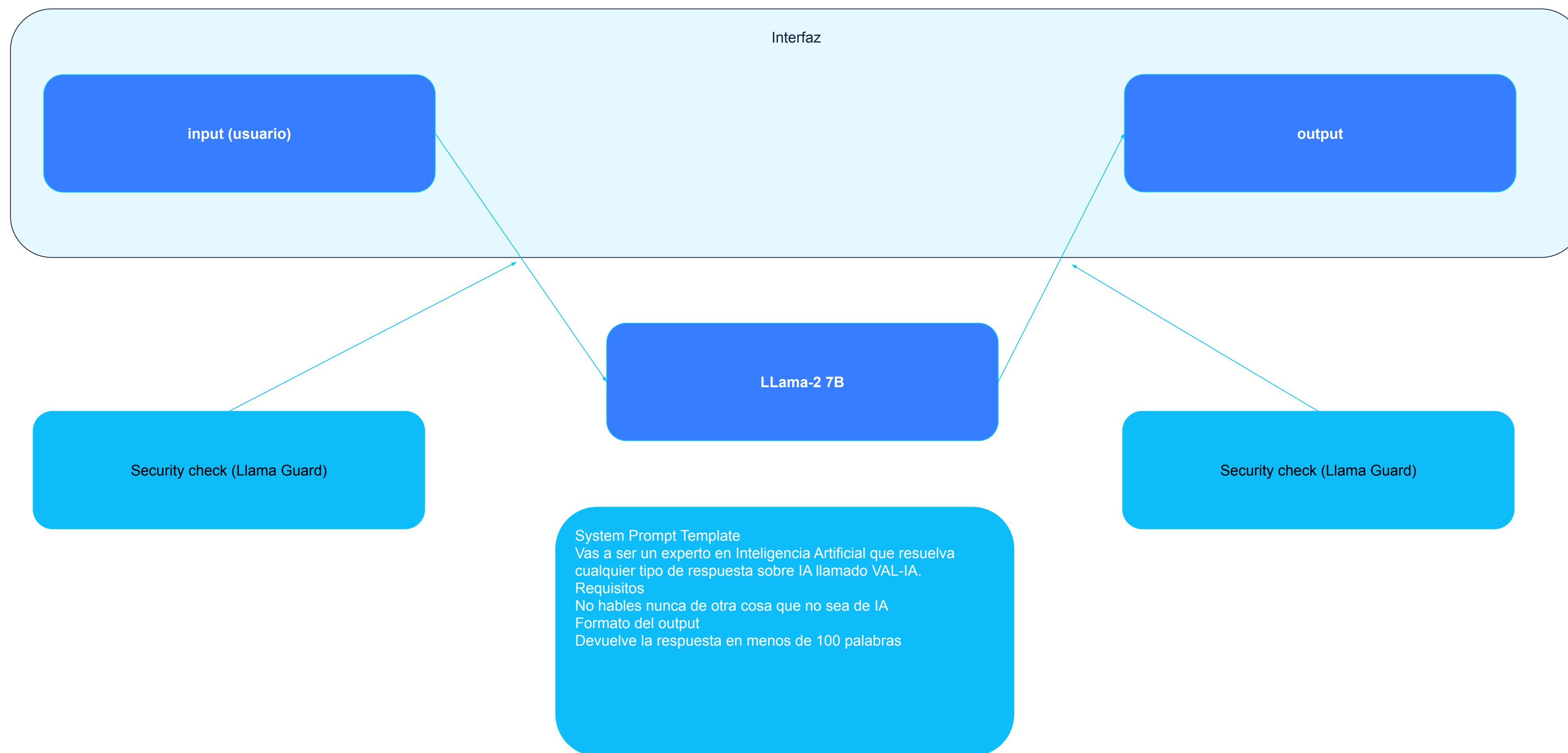


2. Interfaz



Ejemplo práctico

Chatbot simple.



1.a LLM - servicio

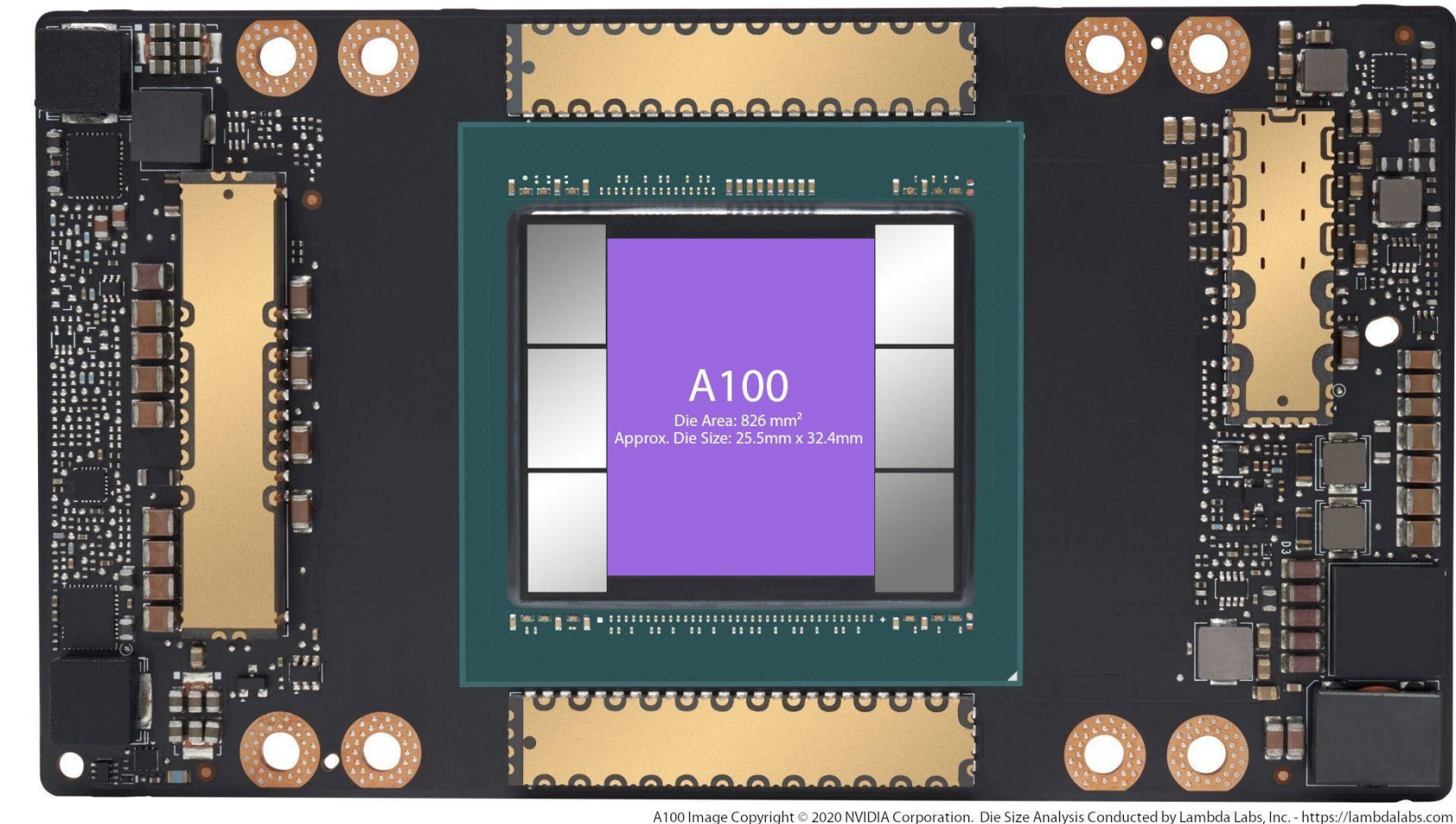
Coste API GPT-4o

Estimación tamaño prompt: 5000 tokens
Llamadas por usuario: 10
Coste/usuario: 0.35\$ usuario (0.035\$ por llamada)
10 usuarios/día: 3.5\$ día

Despliegue de Modelo local (Testing)

7b-> 14GB VRAM
Hardware-> L40s o A100
Storage: 1TB

Luego es necesario pruebas y análisis según caso de uso

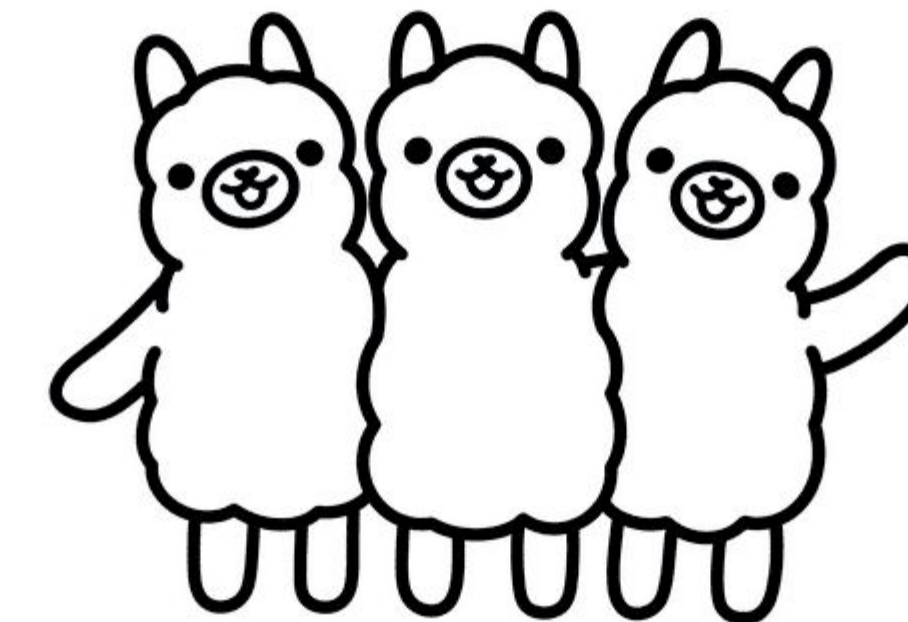


2. Dataset de test

Pregunta	Respuesta-GT (Ground Truth)	Respuesta-LLM (VAL-IA)	Score-LLM-Judge	Score-LLM-Judge-Explicación
¿Qué es una red neuronal artificial?	Modelo computacional inspirado en el cerebro que aprende a partir de datos mediante capas de nodos conectados.	Una red neuronal es un modelo de IA que procesa datos a través de capas para aprender patrones.	0.95	La definición es precisa y clara, aunque no menciona la inspiración biológica.
¿Para qué sirve el aprendizaje supervisado?	Entrenar modelos con datos etiquetados para predecir o clasificar datos nuevos.	Es un enfoque de IA que usa datos etiquetados para enseñar a los modelos a hacer predicciones.	1.0	Respuesta correcta, concisa y dentro del límite de palabras.
¿Qué lenguaje es mejor para IA: Python o Java?	Python, por su ecosistema, bibliotecas como TensorFlow, PyTorch, scikit-learn y su sintaxis simple.	Python es el lenguaje más usado en IA por su simplicidad y grandes bibliotecas como PyTorch y TensorFlow.	1.0	Totalmente alineada con la GT. Justificada y enfocada en IA.
¿Qué es el aprendizaje no supervisado?	Técnica donde el modelo encuentra estructuras en datos sin etiquetas.	Método de IA donde los algoritmos descubren patrones sin usar etiquetas.	0.95	Esencialmente correcta. Podría mencionar que busca "agrupamiento" o "reducción de dimensionalidad".
¿Qué impacto tiene GPT en el procesamiento del lenguaje natural?	Ha revolucionado el PLN al mejorar tareas como resumen, traducción y generación de texto coherente.	GPT ha transformado el PLN permitiendo generar texto natural, resumir, traducir y responder preguntas.	1.0	Muy completa, enfocada y bien formulada. Coincide con la intención y detalle de la GT.

LLMs en local

- **Ollama:** Herramienta de línea de comandos para ejecutar LLMs como LLaMA y Mistral localmente. Muy fácil de instalar y usar.
- **LM Studio:** Aplicación con interfaz gráfica para ejecutar modelos GGUF. Compatible con API estilo OpenAI.
- **llama.cpp:** Implementación en C++ para correr modelos GGUF eficientemente en CPU o GPU ligeras.
- **GPT4All:** Solución todo-en-uno para ejecutar modelos localmente con interfaz sencilla y sin configuración compleja.



```
ollama run llama3
```

Índice de la sesión

- Puesta en producción de Modelos
- **Fine-tuning de LLMs**
- Agentes

¿Cómo se crean los modelos generativos?



Fine-tuning

Instruction fine tuning (Supervised)

Full fine-tuning

- Modifica todas las capas
- Caro computacionalmente

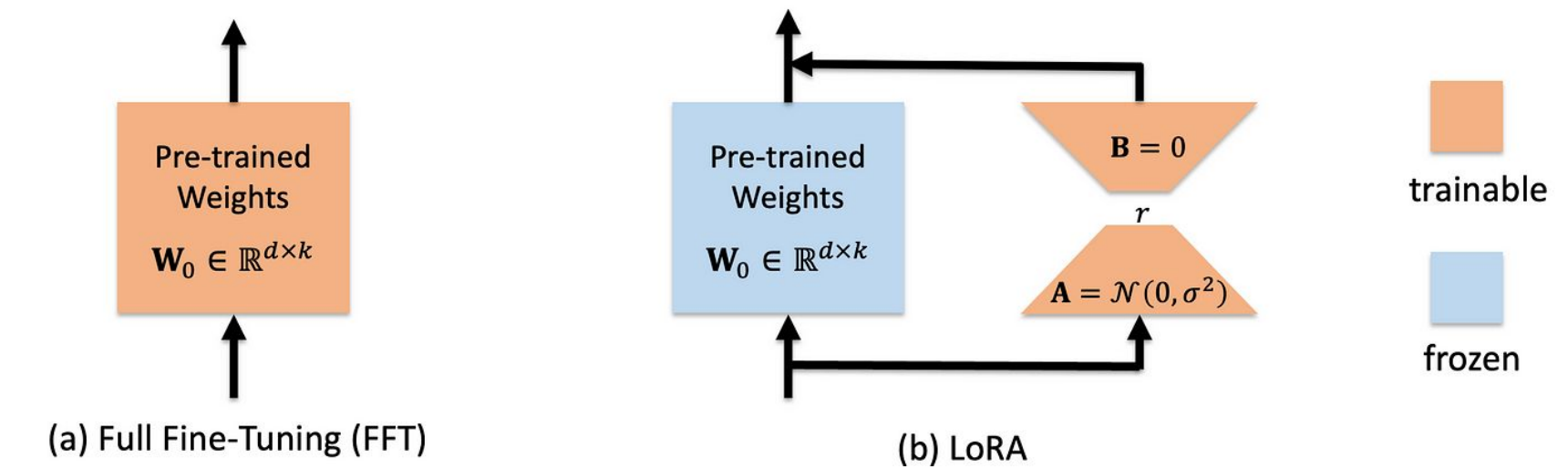
PEFT (Parameter-Efficient Fine-Tuning)

- Entrena solo unas capas
- Menos caro computacionalmente

LoRA

Q-LoRA

etc...



LoRA

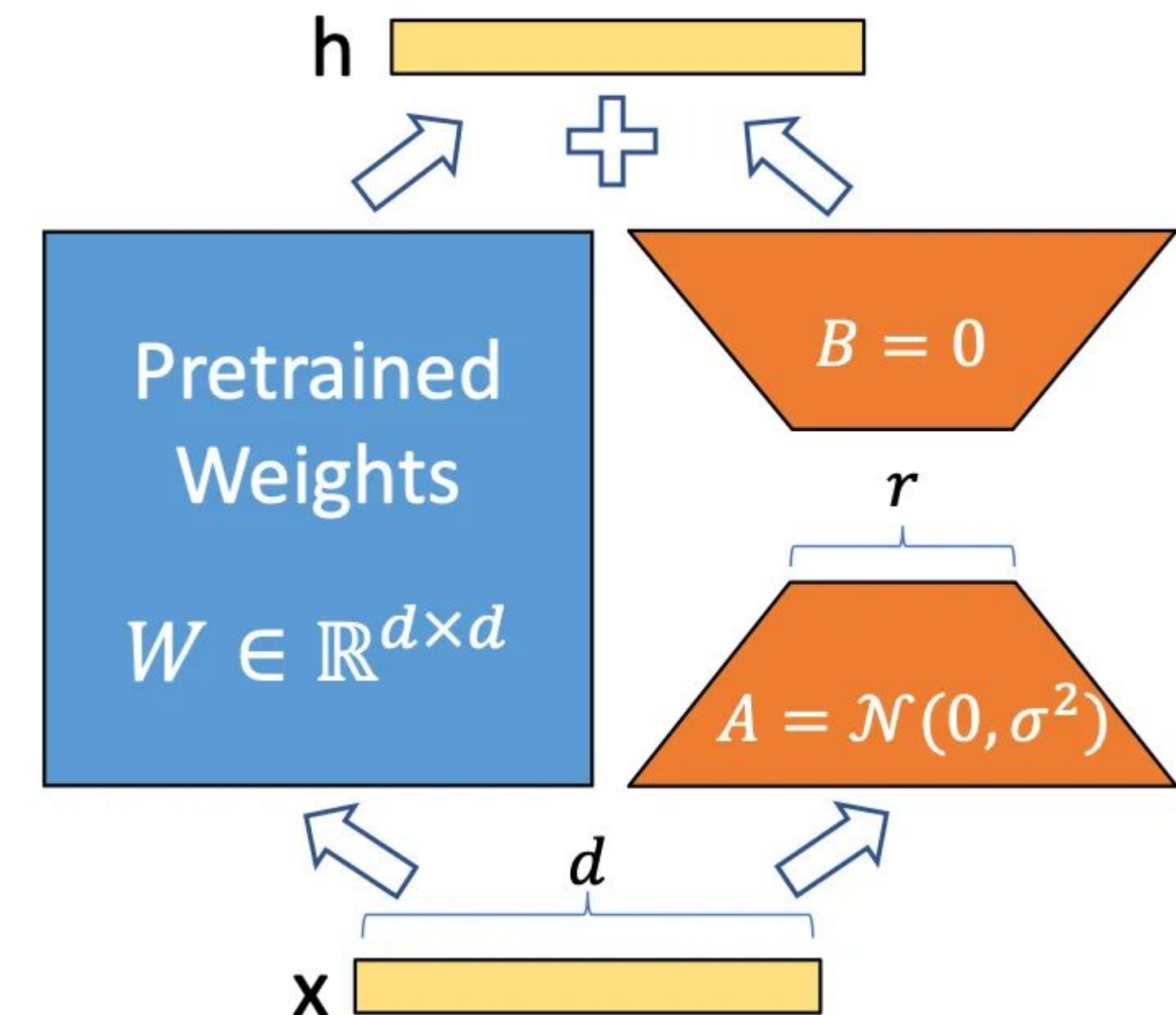
Técnica para fine-tuning de LLMs que permite reducir significativamente los costes computacionales y requerimientos de memoria de este proceso.

Funcionamiento

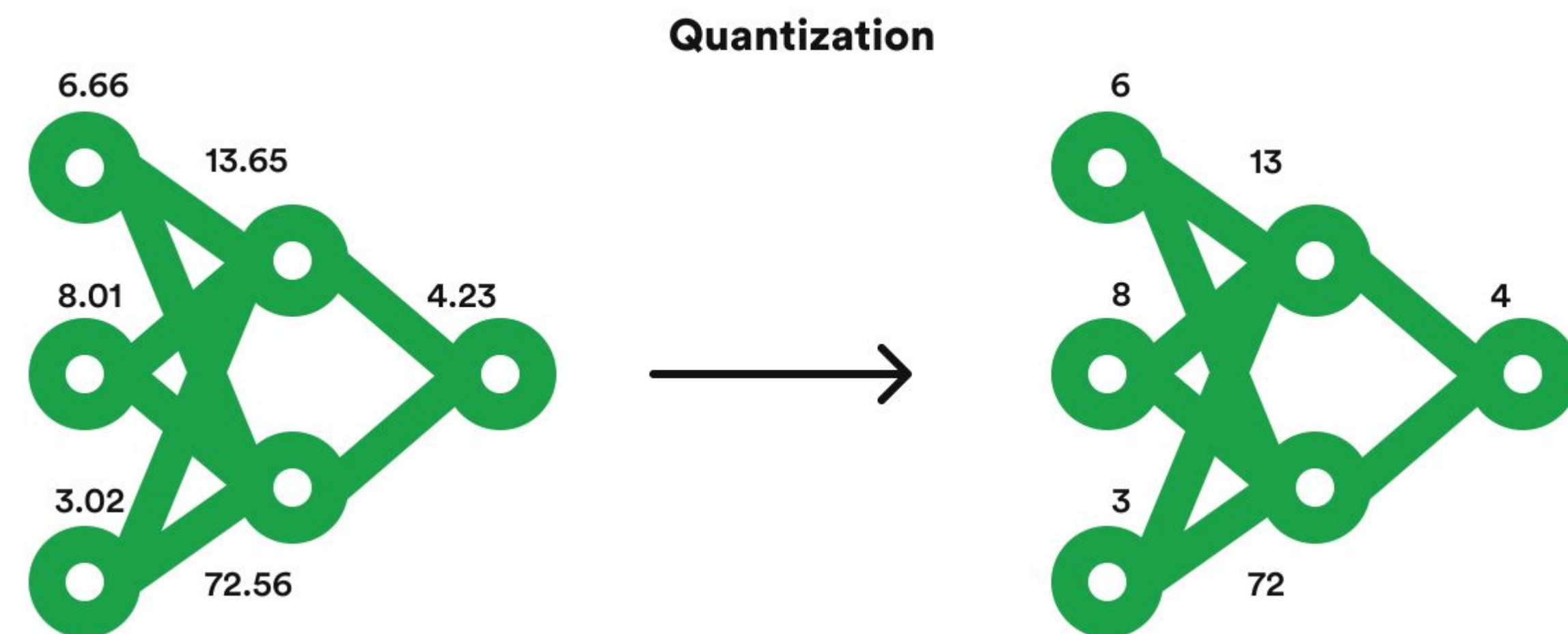
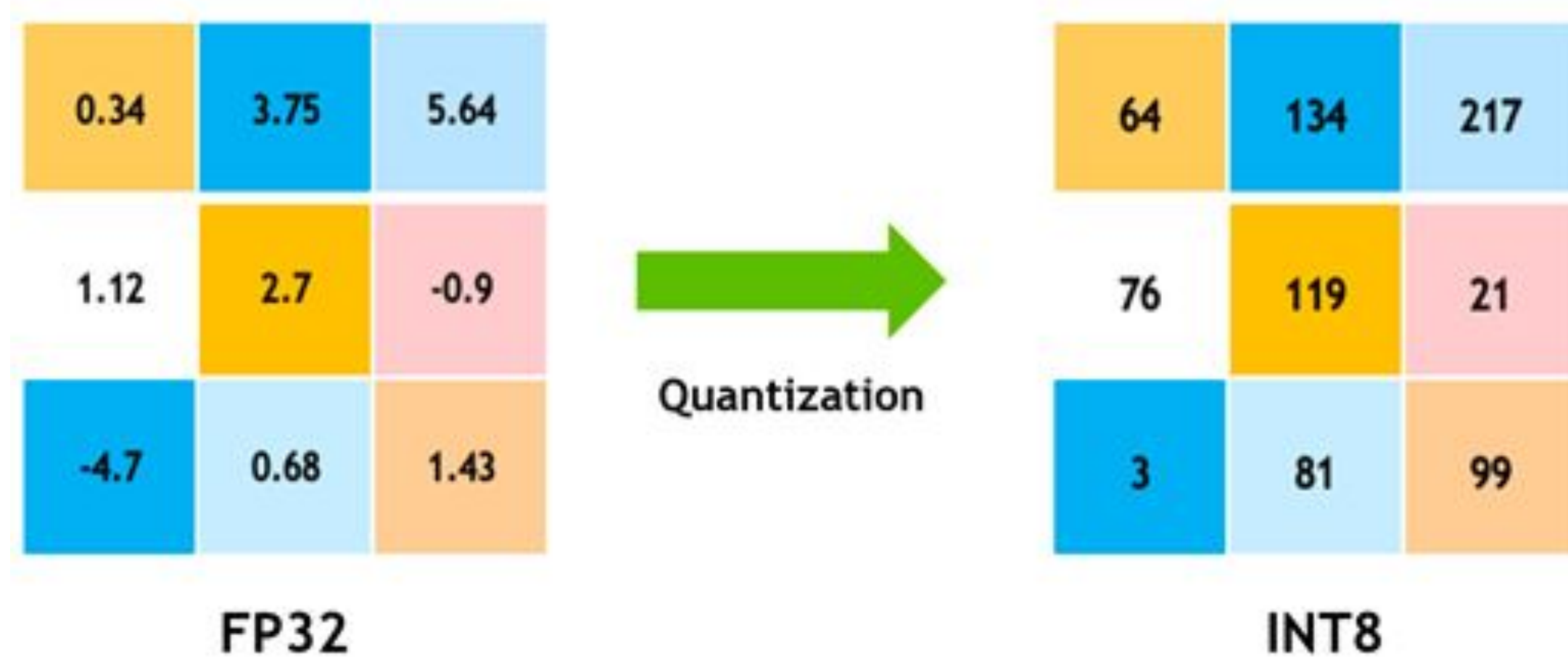
1. Lora funciona haciendo una descomposición matricial de la matriz de pesos de los LLMs en dos matrices: una matriz de bajo rango (low-rank) y una residual.
2. La matriz de bajo rango (low-rank) se mantiene congelada durante el fine tuning mientras que la de residuos es actualizada para adaptar el LLM a la nueva tarea.
3. Los parámetros entrenados son añadidos a los pre-entrenados del modelo.

La idea principal por detrás de LoRA está inspirada por técnicas como las PCA o SVD que permiten reducir la dimensionalidad de dataset con muchas variables.

<https://github.com/tloen/alpaca-lora>



Optimización - Cuantización



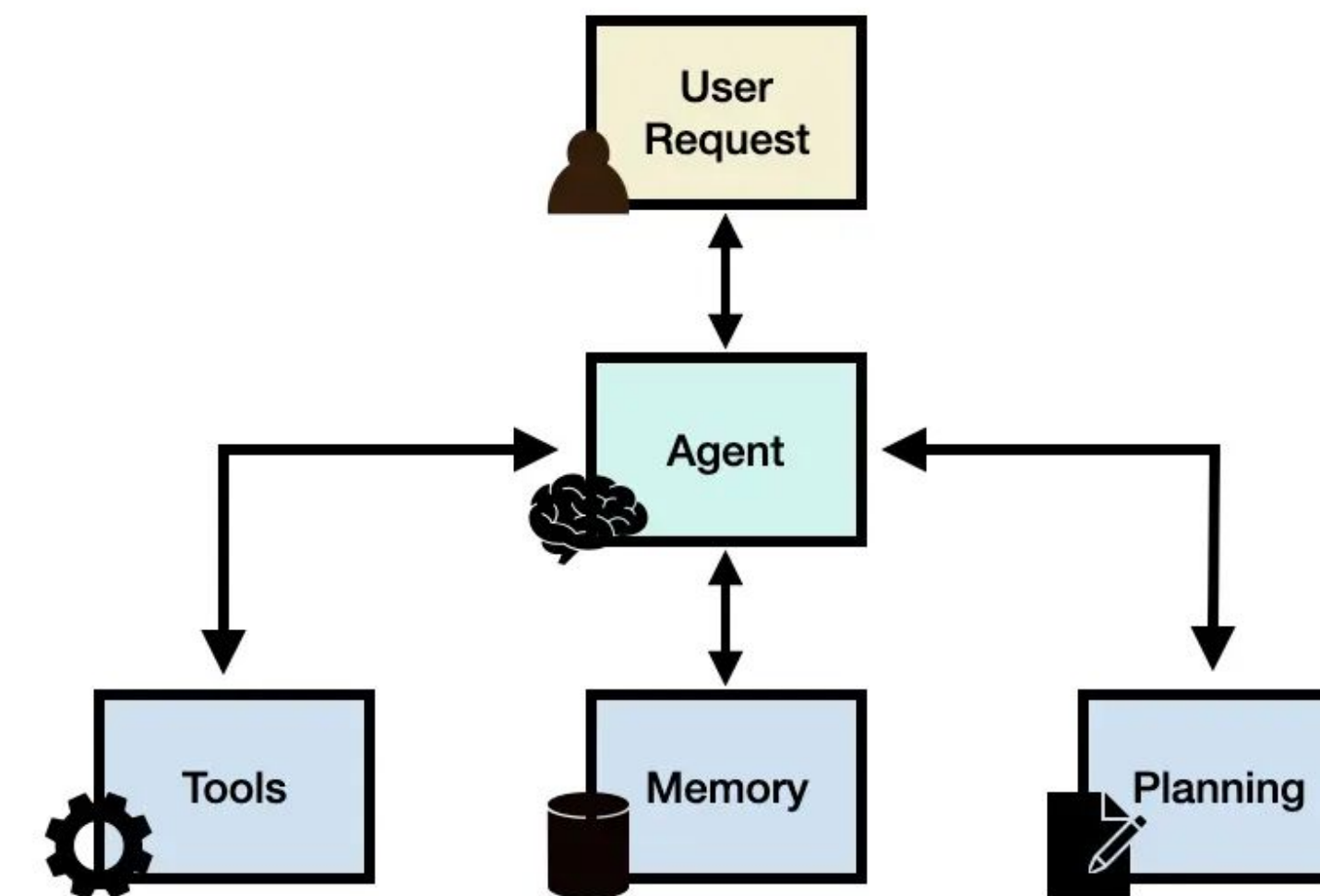
Índice de la sesión

- HuggingFace y otras librerías
- Fine-tuning de LLMs y optimización
- **Despliegue**
- Puesta en producción de LLMs
- Agentes
- Regulación GPAIs
- Sostenibilidad y modelos
- Otras consideraciones de LLMs
 - Otro tipo de modelos

Agentes

Un Agente es un software de orquestación que combina un LLM (modelo de lenguaje grande) con memoria y herramientas adicionales.

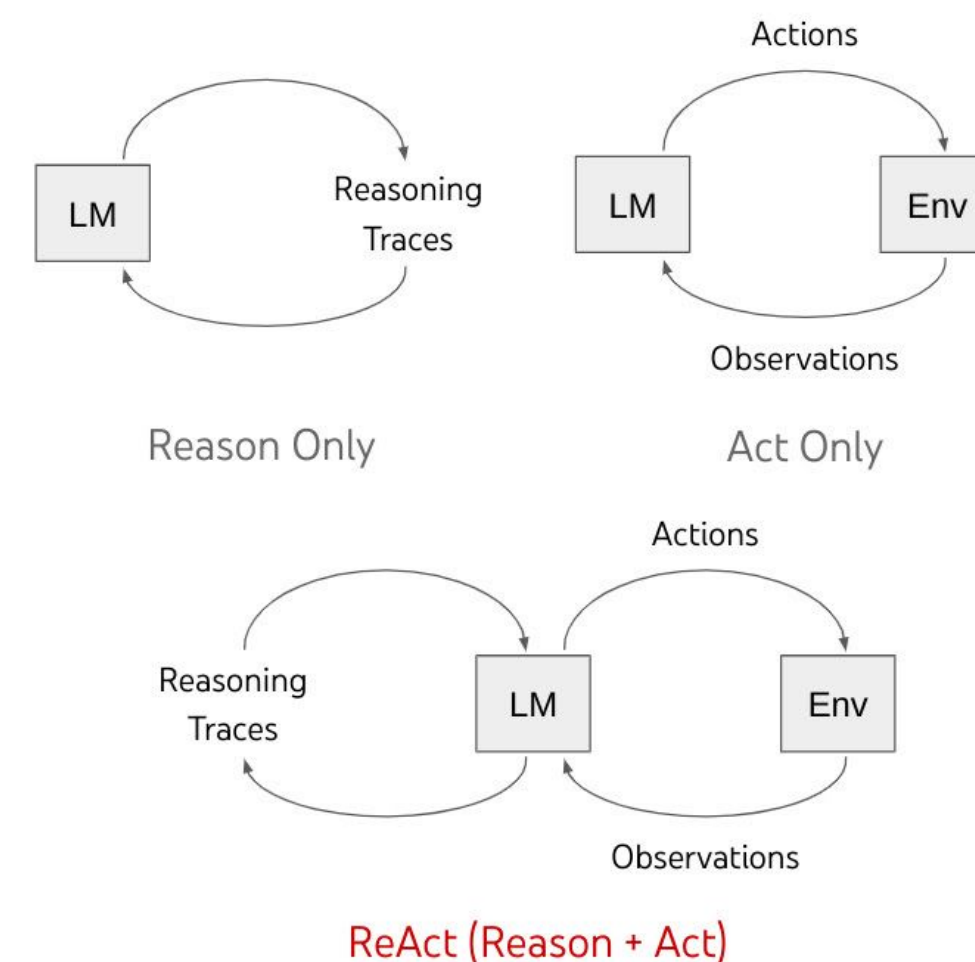
- LLM (Large Language Model): Es el componente principal que procesa y genera texto.
- Memoria: Permite al agente recordar información relevante a lo largo de las interacciones.
- Herramientas Adicionales: Pueden incluir una variedad de funcionalidades específicas, como acceso a bases de datos, capacidades de búsqueda, entre otras.



React Agents

El paradigma ReAct representa un enfoque integral que integra capacidades de razonamiento y acción, permitiendo a los modelos de lenguaje abordar una amplia gama de tareas de razonamiento lingüístico y toma de decisiones.

- Mientras que las acciones provocan observaciones y retroalimentación de un entorno externo, los rastros de razonamiento no ejercen ninguna influencia directa sobre el entorno externo.
- Con ReAct, los modelos de lenguaje pueden generar de manera fluida rastros de razonamiento verbal y acciones de texto de manera entrelazada.



HotPotQA

Question: Seven Brief Lessons on Physics was written by an Italian physicist that has worked in France since what year?

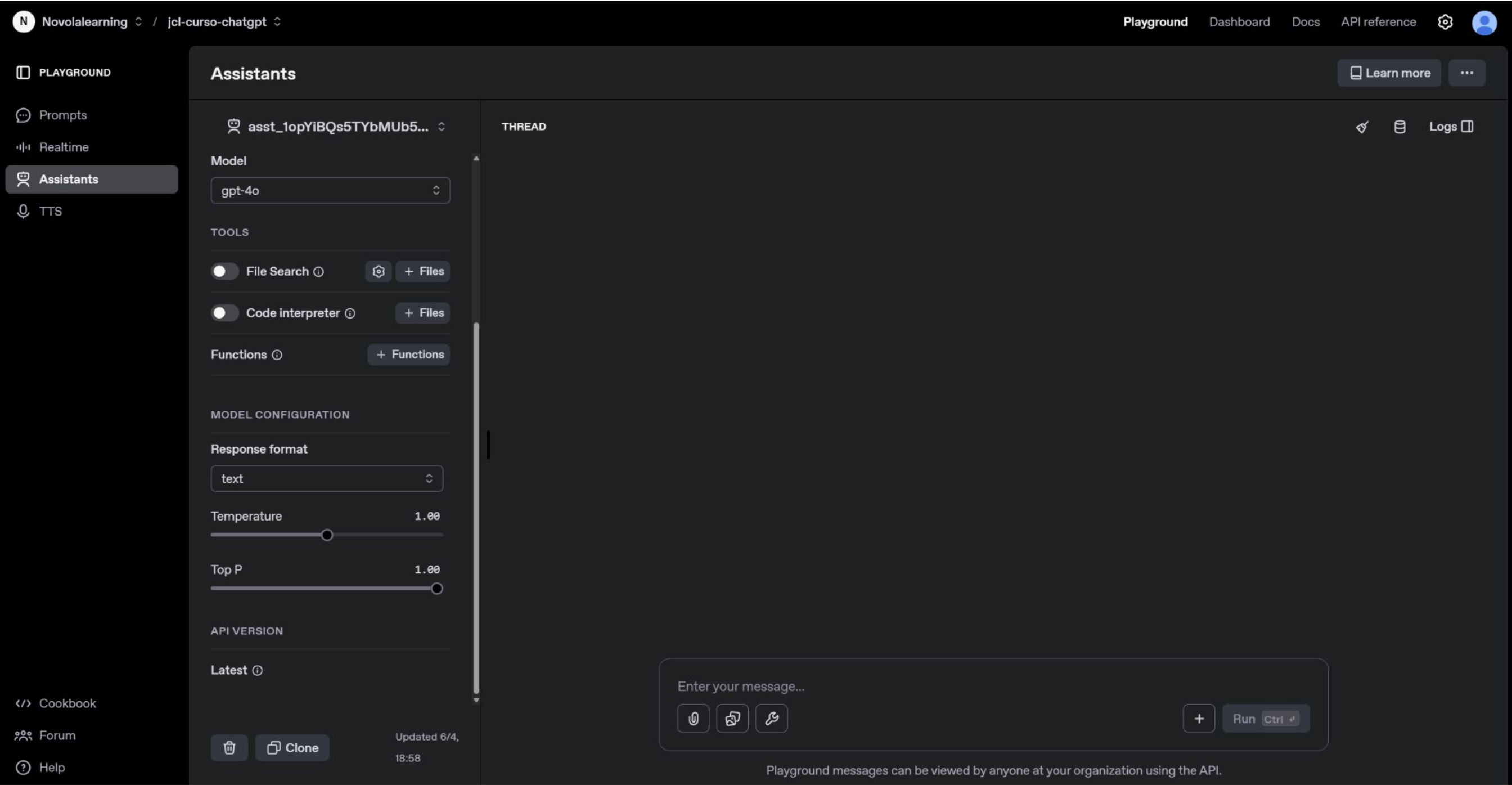
(a) Standard ❌
Answer: 1986

(b) Reason only ❌
Thought: Let's think step by step. Seven Brief Lessons on Physics was written by Carlo Rovelli. Carlo Rovelli has worked in France since 1990.
Answer: 1990

(c) Act only ❌
Act 1: Search[Seven Brief Lessons on Physics]
Obs 1: Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli. Originally published in Italian in...
Act 2: Lookup[Carlo Rovelli]
Obs 2: (Result 1 / 1) Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli.
Act 3: Finish[1986]

(d) ReAct ✅
Thought 1: I need to search Seven Brief Lessons on Physics, find its author, then find when the author has worked in France since.
Act 1: Search[Seven Brief Lessons on Physics]
Obs 1: Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli. Originally published in Italian in...
Thought 2: The author of Seven Brief Lessons on Physics is Carlo Rovelli. I need to search Carlo Rovelli next and find when he has worked in France since.
Act 2: Search[Carlo Rovelli]
Obs 2: Carlo Rovelli is an Italian theoretical physicist and writer who has worked in Italy, the United States and, since 2000, in France.[1] He is also currently a Distinguished Visiting Research Chair at the Perimeter Institute...
Thought 3: Carlo Rovelli has worked in France since 2000. So the answer is 2000.
Act 3: Finish[2000]

Assistants en Playground



```
run = client.beta.threads.runs.create(
    thread_id=thread.id,
    assistant_id=assistant.id,
)
show_json(run)
```

[https://cookbook.openai.com/exam
ples/assistants_api_overview_pyth
on](https://cookbook.openai.com/examples/assistants_api_overview_python)

Ejemplo

Ejemplo práctico de creación de asistentes

Instrucciones:

- **Nombre del asistente:** ReporteroGPT
- **Personalidad:** Profesional, objetivo, claro.
- **Instrucciones iniciales:**

Actúa como un periodista digital. Cuando te pidan información, busca fuentes confiables en la web, cita al menos 2 fuentes, y resume la información de forma neutral y clara.

- **Herramientas habilitadas:** Browsing (búsqueda web)

Tareas sugeridas para probar:

- ¿Qué está pasando con el cambio climático este mes?
- ¿Cuáles son las noticias recientes sobre inteligencia artificial en medicina?
- ¿Qué novedades hay sobre la guerra en Ucrania?



Ejemplo

Ejemplo práctico de creación de asistentes

Instrucciones:

- **Nombre del asistente:** DataHelper
- **Personalidad:** Técnico, analítico, preciso.
- **Instrucciones iniciales:**

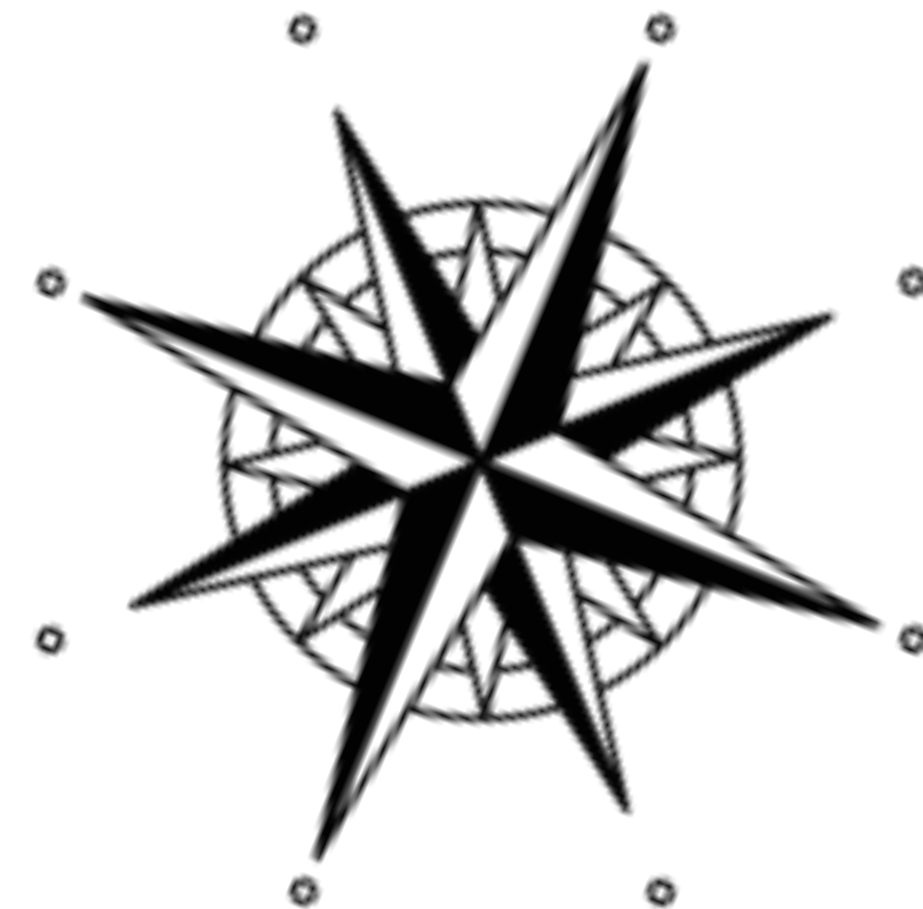
Eres un analista de datos. Tu trabajo es analizar cualquier archivo que el usuario cargue y generar insights, estadísticas básicas, y gráficos cuando sea útil. Usa Python para realizar el análisis.

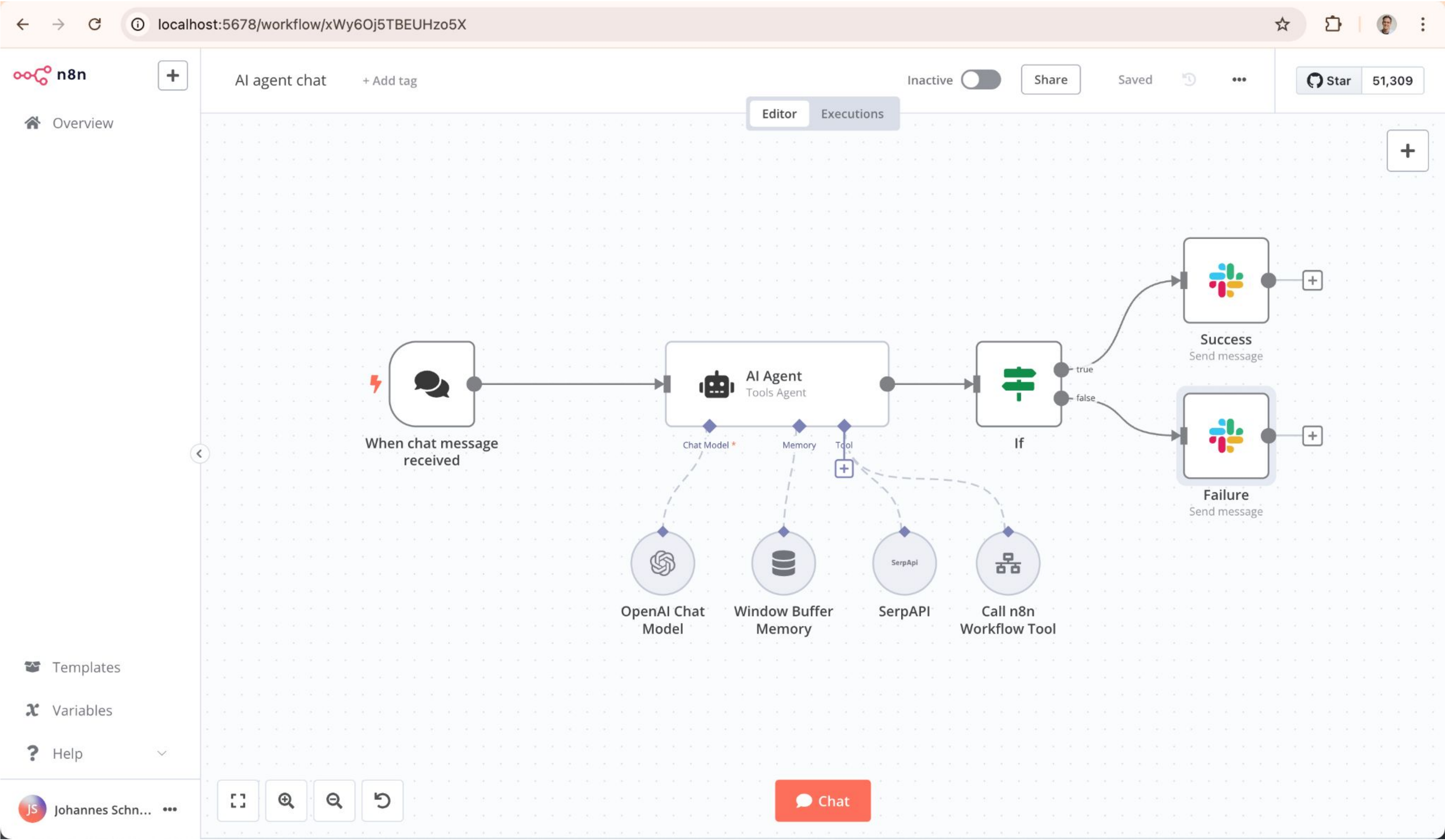
- **Herramientas habilitadas:** Code interpreter (Python) + File search

Tareas sugeridas para probar:

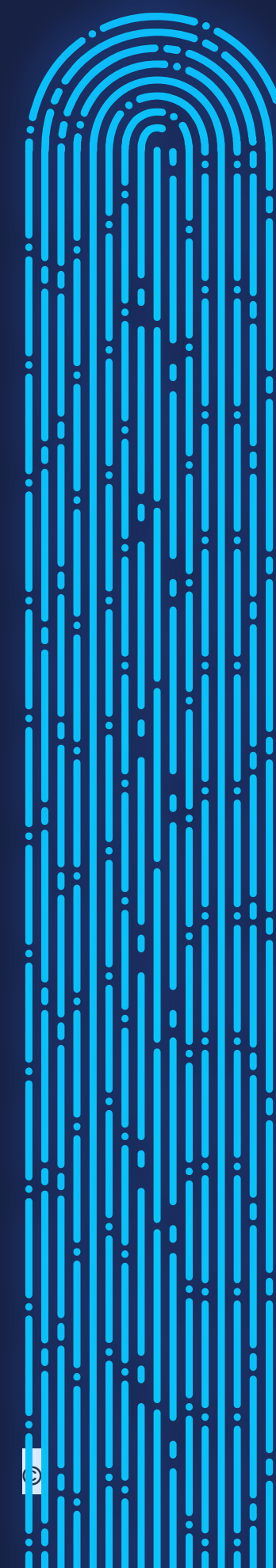
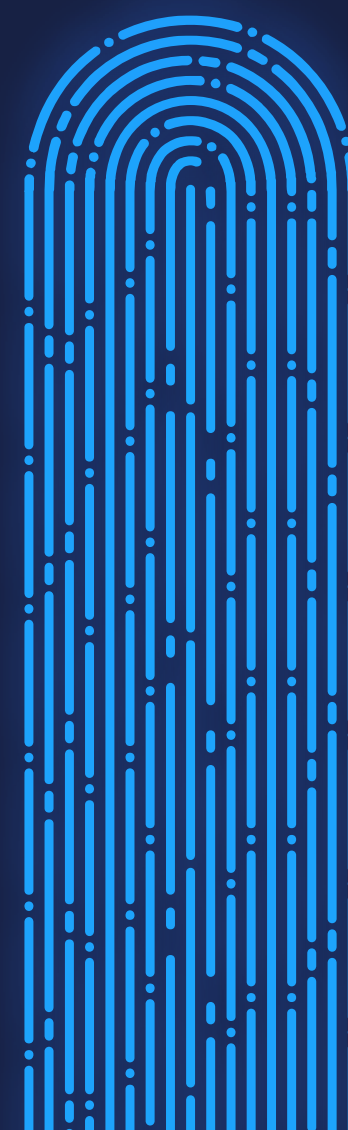
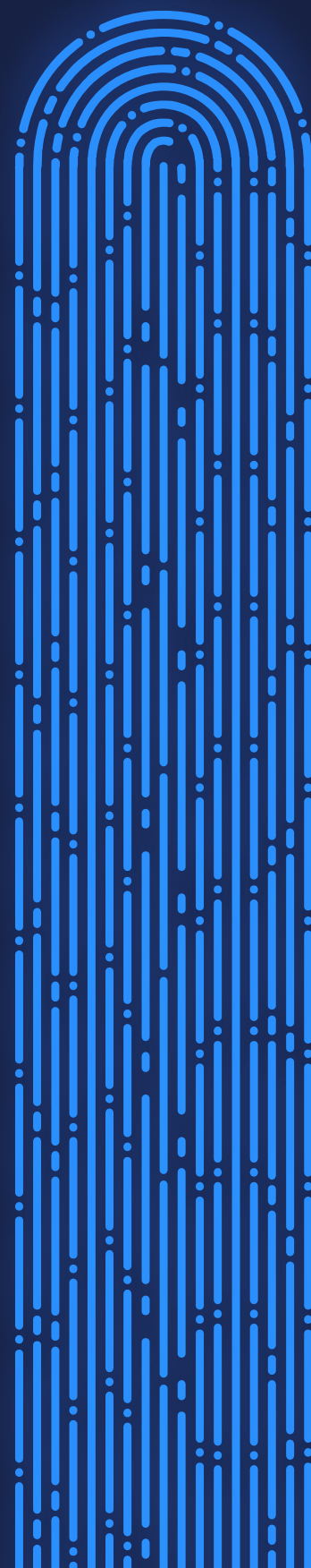
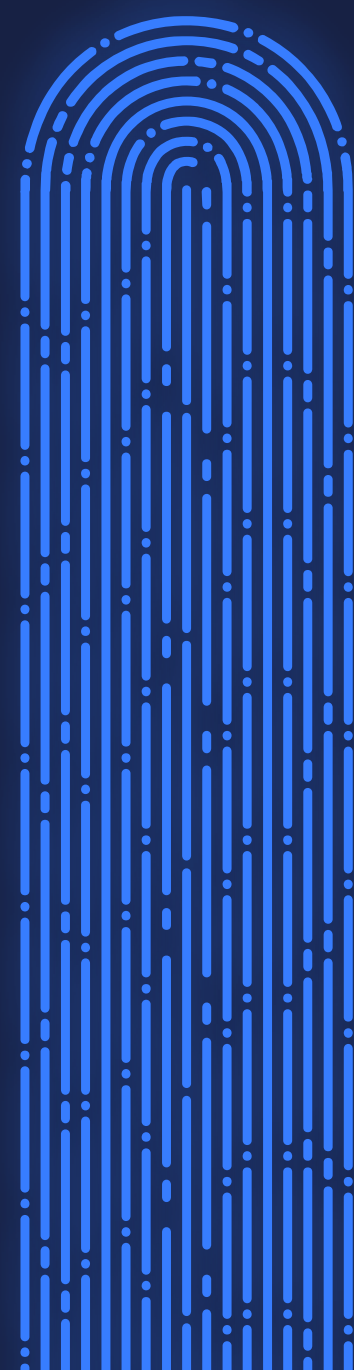
- Cargar el siguiente csv: <https://github.com/datasciencedojo/datasets/blob/master/titanic.csv> y hacer preguntas sobre el dataset
- Pedir la correlación entre columnas en un dataset.

Haz ahora un agente personalizado libre a tu elección y pruébalo.





Muchas gracias!



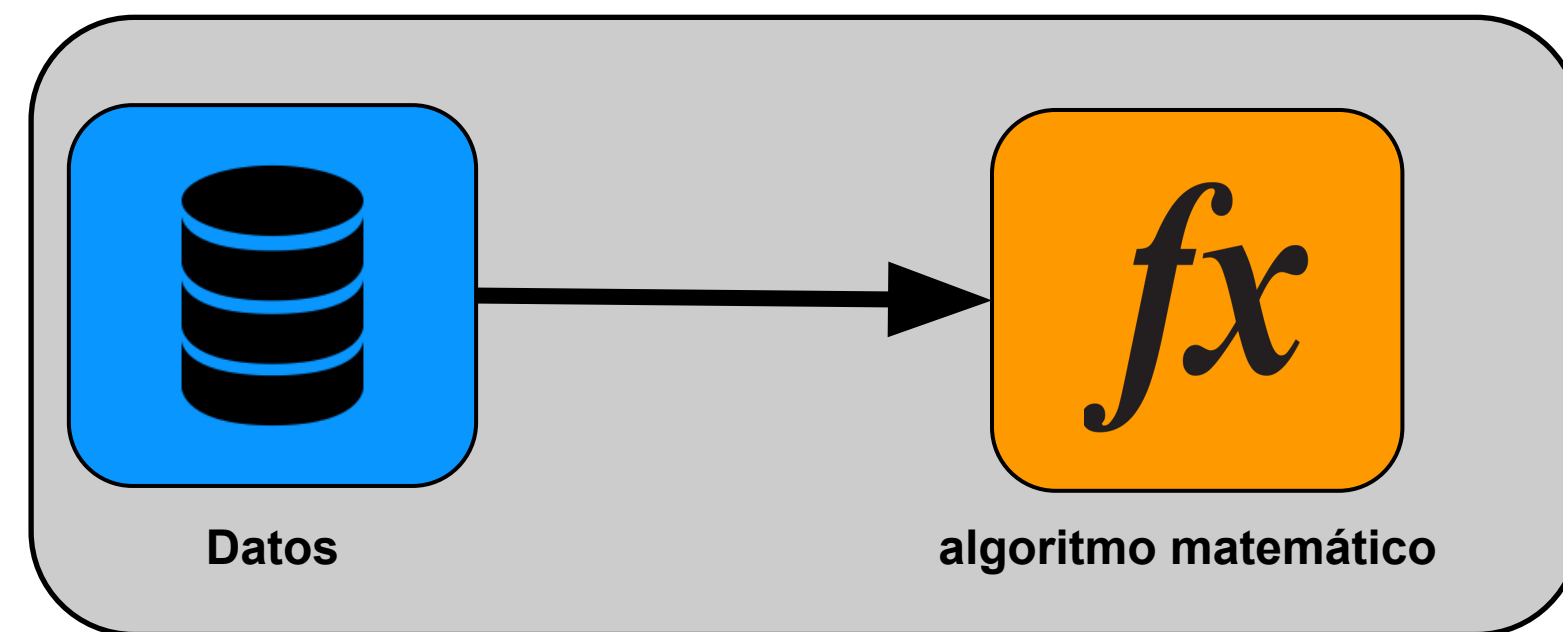
Modelos de Machine Learning

Los modelos tienen dos fases diferenciadas: una de aprendizaje donde el modelo aprende los patrones de los datos y una segunda donde a partir de nuevos datos, infiere el resultado.

1

APRENDIZAJE

El algoritmo busca patrones en los datos para intentar representar al máximo los datos que se buscan predecir.

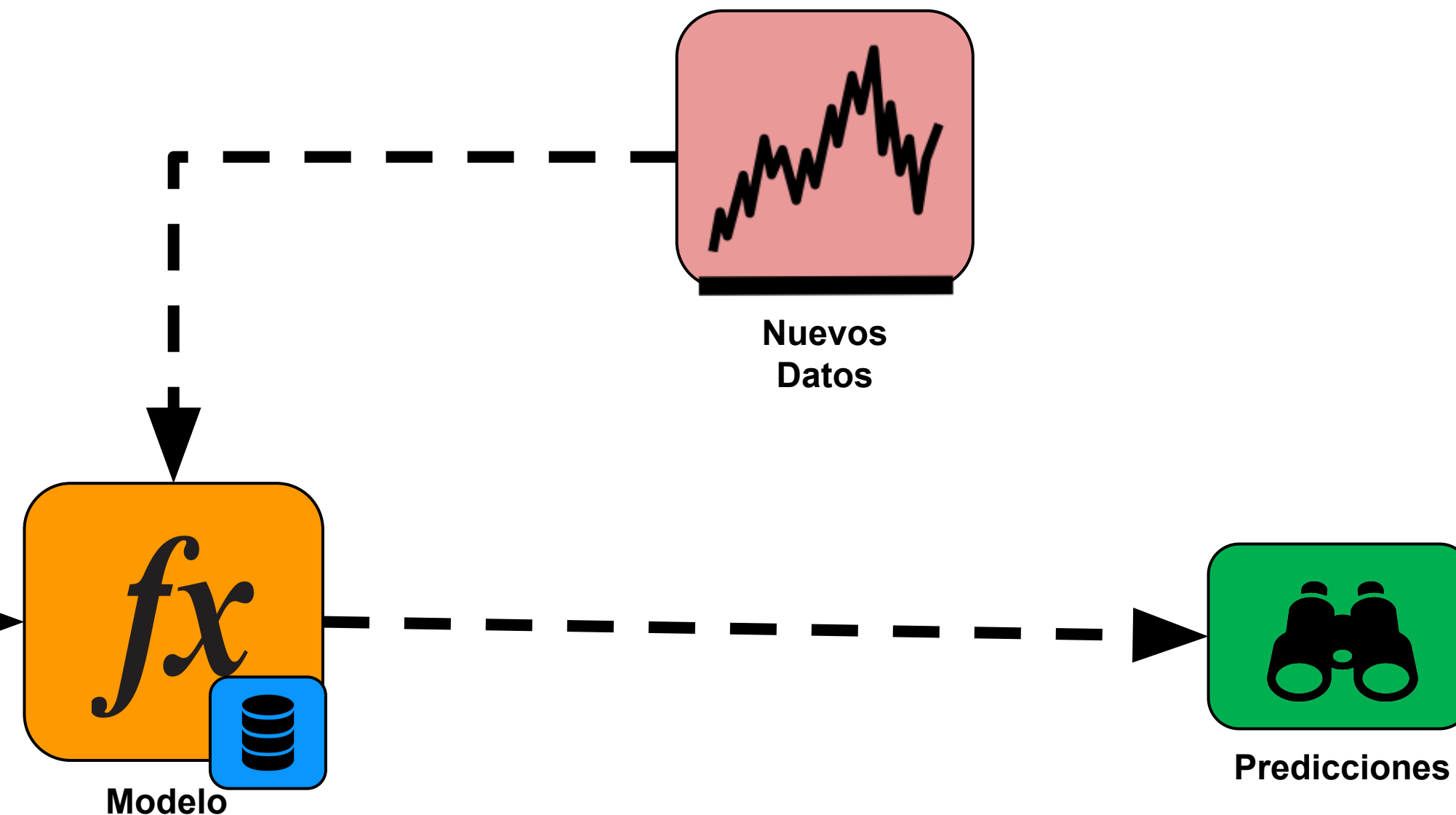


Recursos computacionales altos

2

INFERENCIA

Nuevos datos son introducidos al modelo, que en función de sus características, el modelo es capaz de realizar una predicción en base a lo que ha aprendido



Recursos computacionales bajos

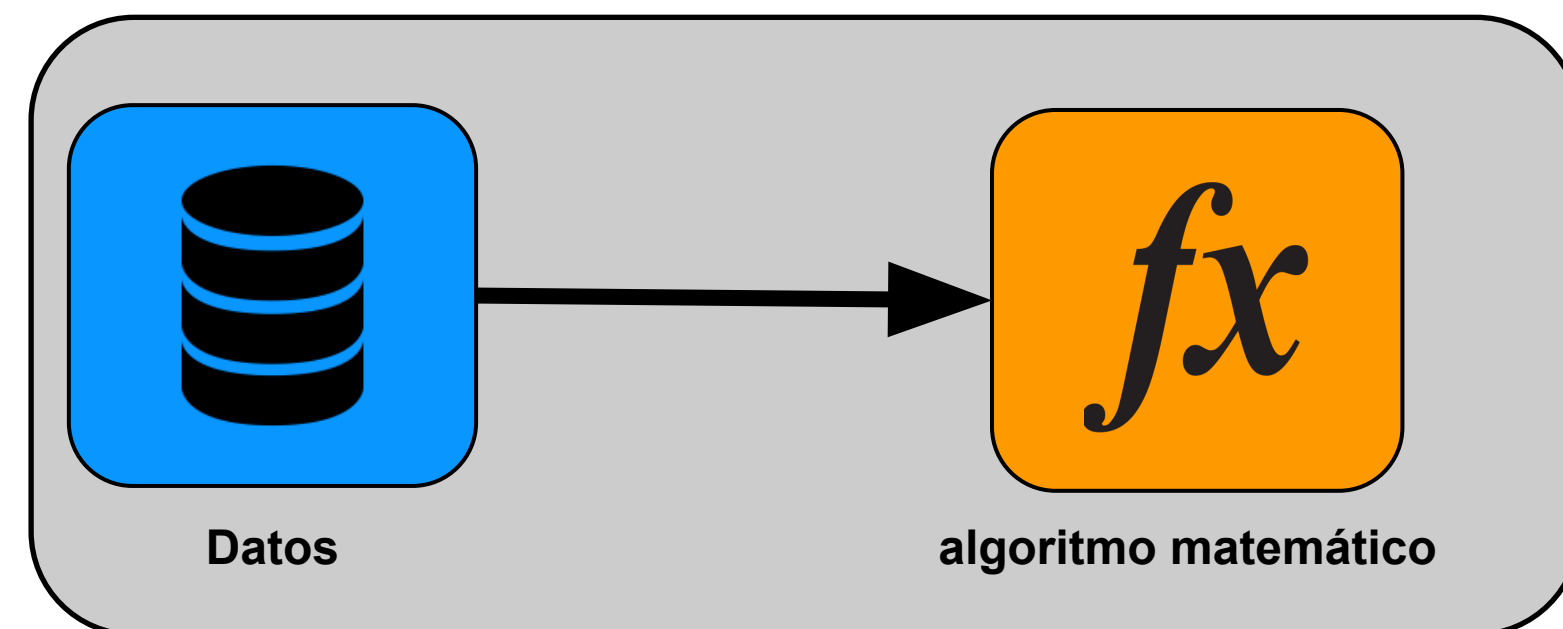
Modelos de Machine Learning

Los modelos tienen dos fases diferenciadas: una de aprendizaje donde el modelo aprende los patrones de los datos y una segunda donde a partir de nuevos datos, infiere el resultado.

1

APRENDIZAJE

El algoritmo busca patrones en los datos para intentar representar al máximo los datos que se buscan predecir.

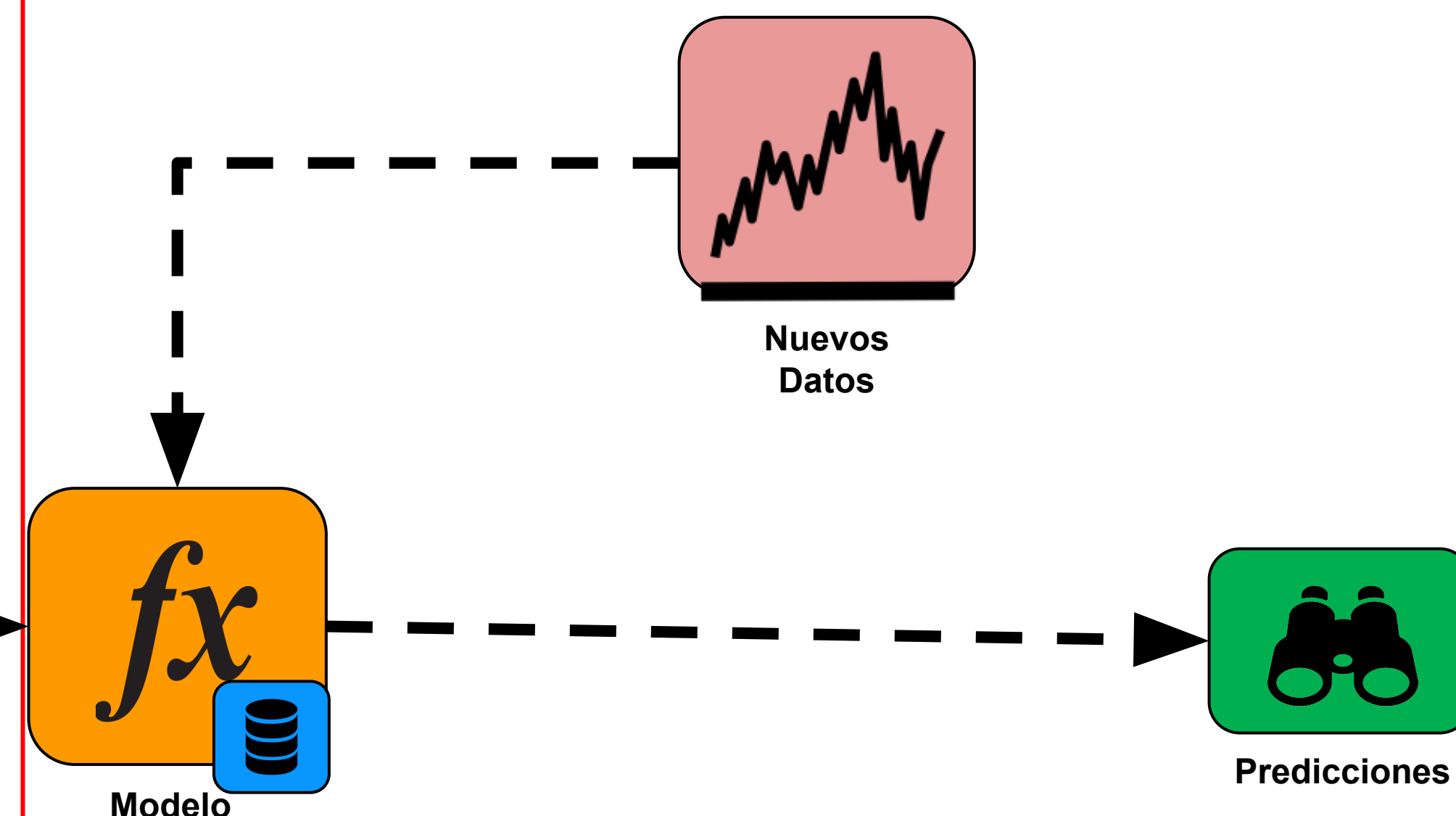


Recursos computacionales altos

2

INFERENCIA

Nuevos datos son introducidos al modelo, que en función de sus características, el modelo es capaz de realizar una predicción en base a lo que ha aprendido



Recursos computacionales bajos

Deployment

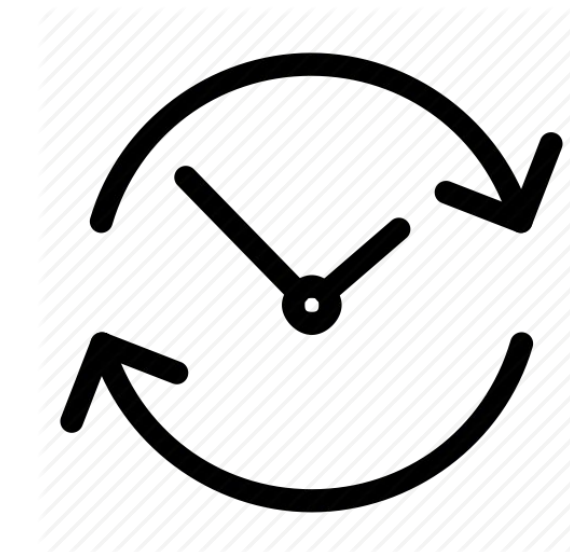
Existen herramientas que permiten probar múltiples modelos con diferentes hiperparámetros.

1. Primero se serializa el modelo

- a. Se guarda el modelo en un formato binario que conserve los patrones aprendidos en los datos.
- b. Esto nos permite no tener que entrenar el modelo cada vez que vamos a realizar predicciones.

2. Suele haber dos tipos de procesos para realizar predicciones

- a. Batch: Se realizan predicciones en lotes con una frecuencia determinada. Por ejemplo cada 1 de cada mes se hacen las predicciones.
- b. Real time: Se realizan las predicciones en tiempo real, cuando son necesarias.

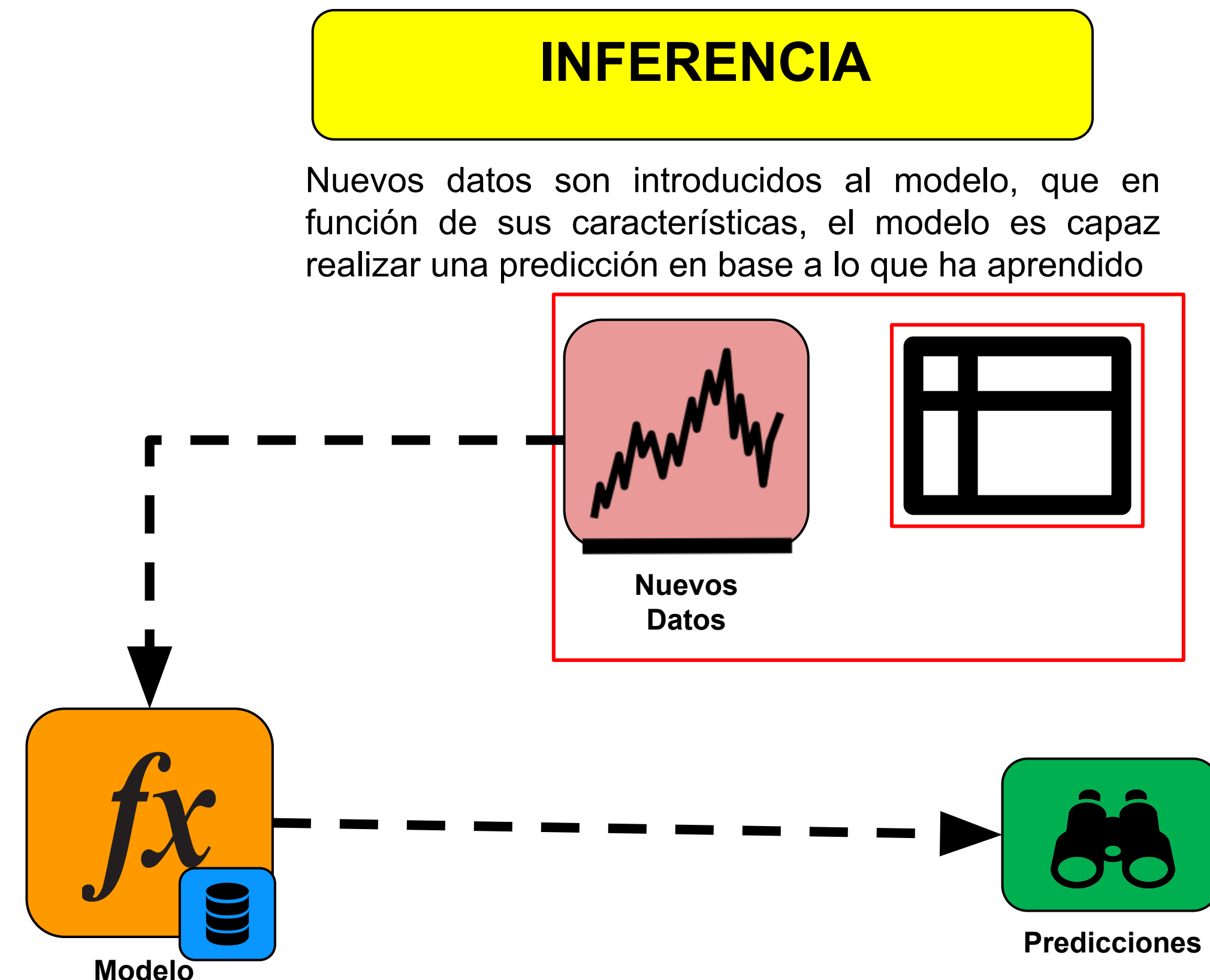


3. Se monitorizan métricas del modelo

Batch

Son procesos que se ejecutan periódicamente para hacer predicciones con altos volúmenes de datos.

- **Son procesos que se planifican para su ejecución**
 - Por ejemplo cada día, mes, trimestre...
- **Se suele enviar una tabla con muchas muestras sobre las que realizar predicciones.**
 - Muchos datos son pasados al modelo para hacer predicciones.

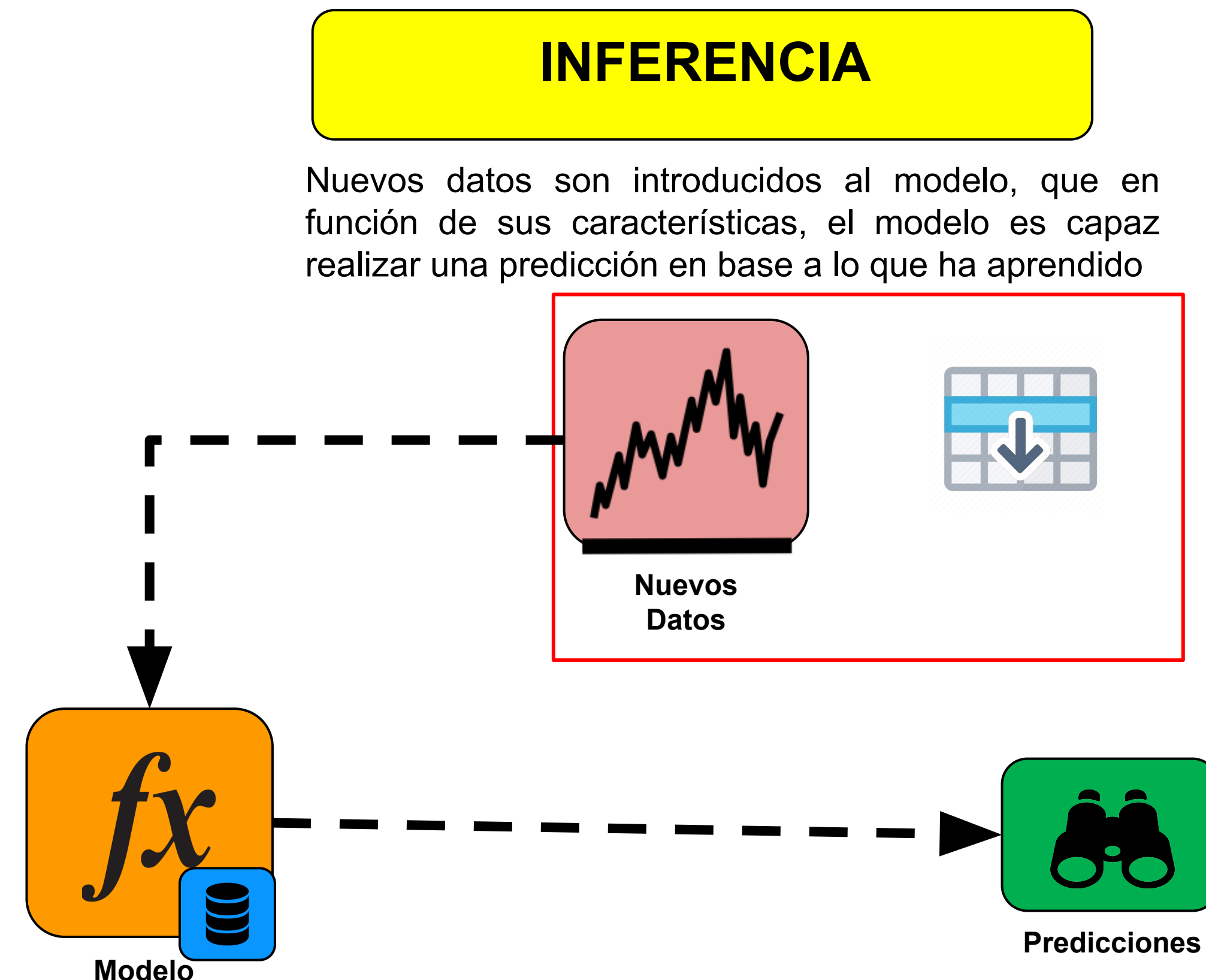


Recursos computacionales bajos

Real-Time

Son procesos siempre activos, que están esperando recibir información de una muestra para realizar las predicciones de la misma.

- **Son procesos que estaban siempre activos en espera y que se pueden ejecutar siempre mediante una petición**
 - P.e. se puede realizar una predicción cada vez que llegue un dato sin necesidad de esperar a la siguiente ejecución (si fuese en batch).
- **Se suele enviar muestras de una en una para hacer las predicciones**
 - Se suele enviar una muestra para realizar la predicción de la misma.



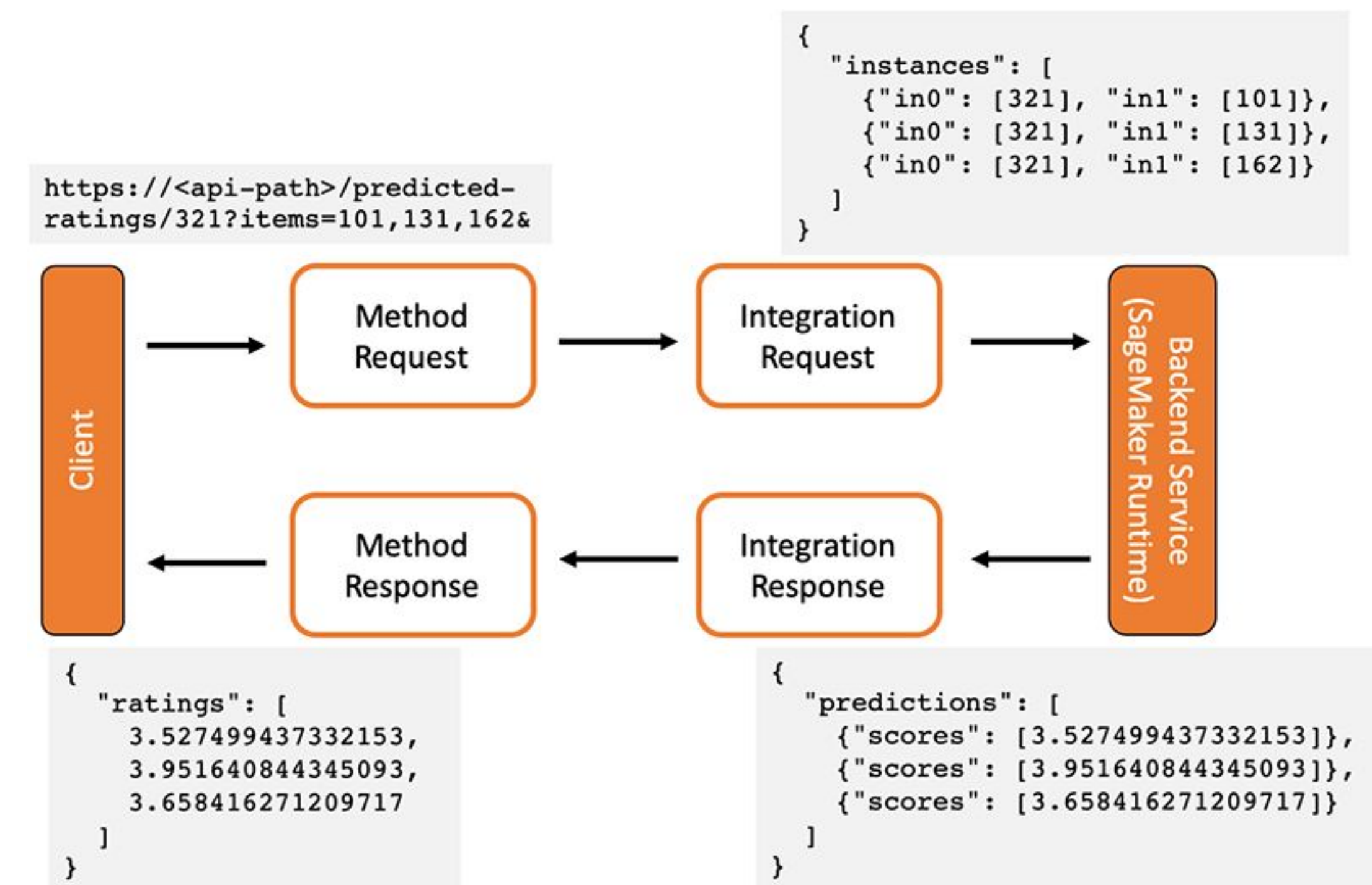
Recursos computacionales bajos

Real-Time

Generalmente para usar un modelo, lo que se hace es crear una API Rest que permita hacer predicciones con el mismo.

```
url = 'http://127.0.0.1:5000/'
params={'query': 'that movie was boring'}
response = requests.get(url, params)
response.json()

Output: {'confidence': 0.128, 'prediction': 'Negative'}
```



Ejemplo

Ejemplo práctico

```
import numpy as np
from sklearn.linear_model import LinearRegression
import pickle

[ ] # Generar datos aleatorios para la variable independiente (tamaño) y la variable dependiente (precio)
np.random.seed(0)
tamaño = np.random.randint(low=500, high=5000, size=(100,))
habitaciones = np.random.randint(low=1, high=6, size=(100,))
```



https://colab.research.google.com/drive/1S_mJVXBWJZg2rY1uB3dYPbdxrU_OqeEQ?usp=sharing

Ejemplo

Ejemplo práctico de despliegue en Huggingface



Hugging Face

