

# LLMs, alineamiento y agentes

---

- Modelos causales
- Instruct tuning
- RLHF/DPO/ORPO
- Razonamiento
- Agentes
- Ética

- Modelos causales: tokenización, ventana de contexto y generación.
- Instruction tuning (SFT) sobre modelos preentrenados.
- Alineamiento: RLHF, DPO/ORPO y modelos de preferencia; razonamiento.
- Agentes: herramientas, memoria y planificación.
- Riesgos, sesgos y consideraciones éticas básicas.

# Arquitecturas Transformer (visión general)

- Comparación útil: encoder-only, encoder-decoder y decoder-only.
- LLMs modernos suelen ser decoder-only (autoregresivos).

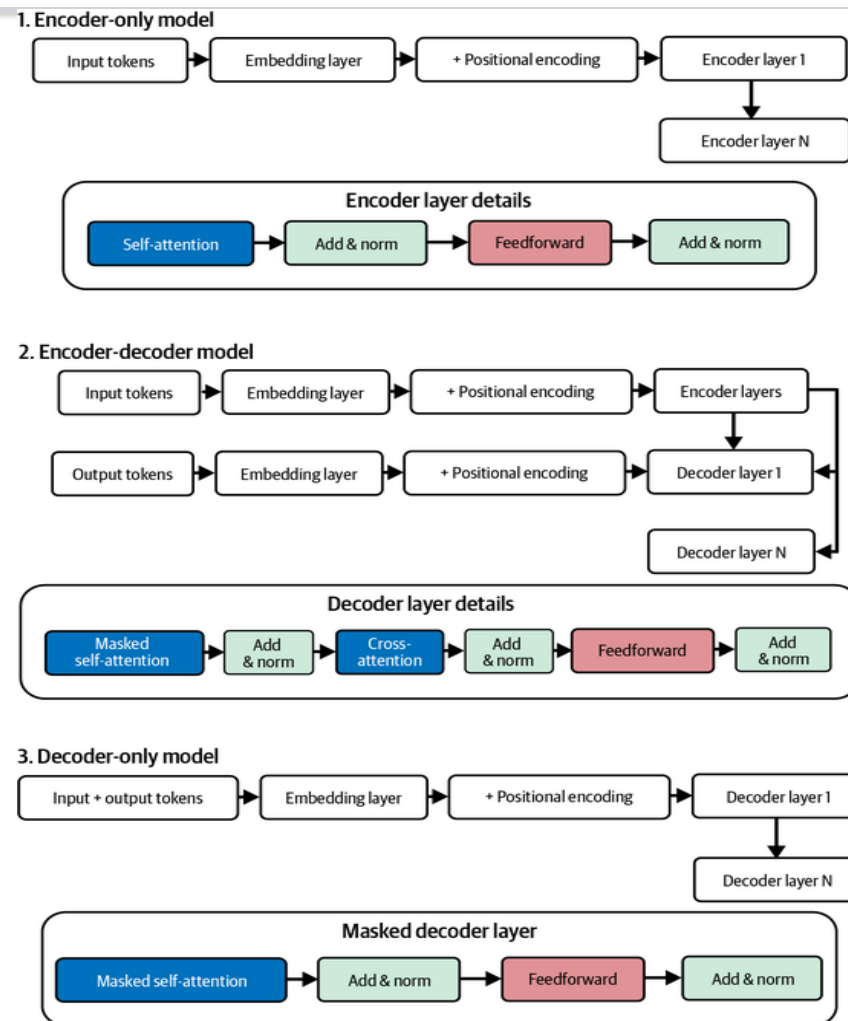


Figura : comparación de arquitecturas Transformer.

# Modelos de Lenguaje Causal (decoder-

- Decoder-only: predice el siguiente token con máscara causal.
- Entrenamiento: next-token prediction; inferencia: generación paso a paso.
- Costo  $\uparrow$  con longitud de contexto y tasa de generación.

## 3. Decoder-only model

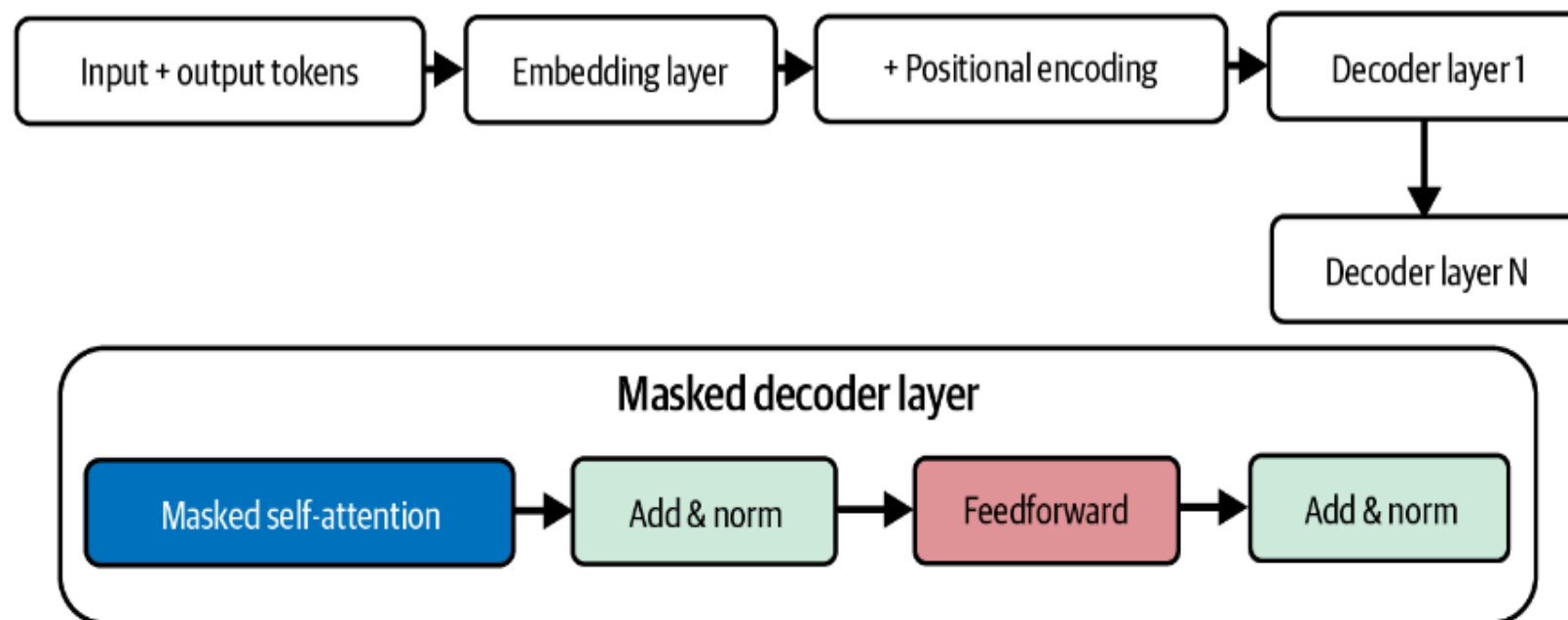


Figura : Modelo decoder-only.

# Tokenización: subwords y merges

- BPE/SentencePiece construyen subwords frecuentes.
- Tokens  $\neq$  palabras: el costo real depende de tokens.
- Medir tokens/ejemplo ayuda a diseñar prompts y datasets.



Figura : ejemplo de tokenización subword (merges).

# Ventana de contexto

- Ventana = tokens máximos (prompt + historial + docs + respuesta).
- Diseño práctico: reservar tokens para la respuesta.
- Estrategias: chunking con solapamiento y RAG.

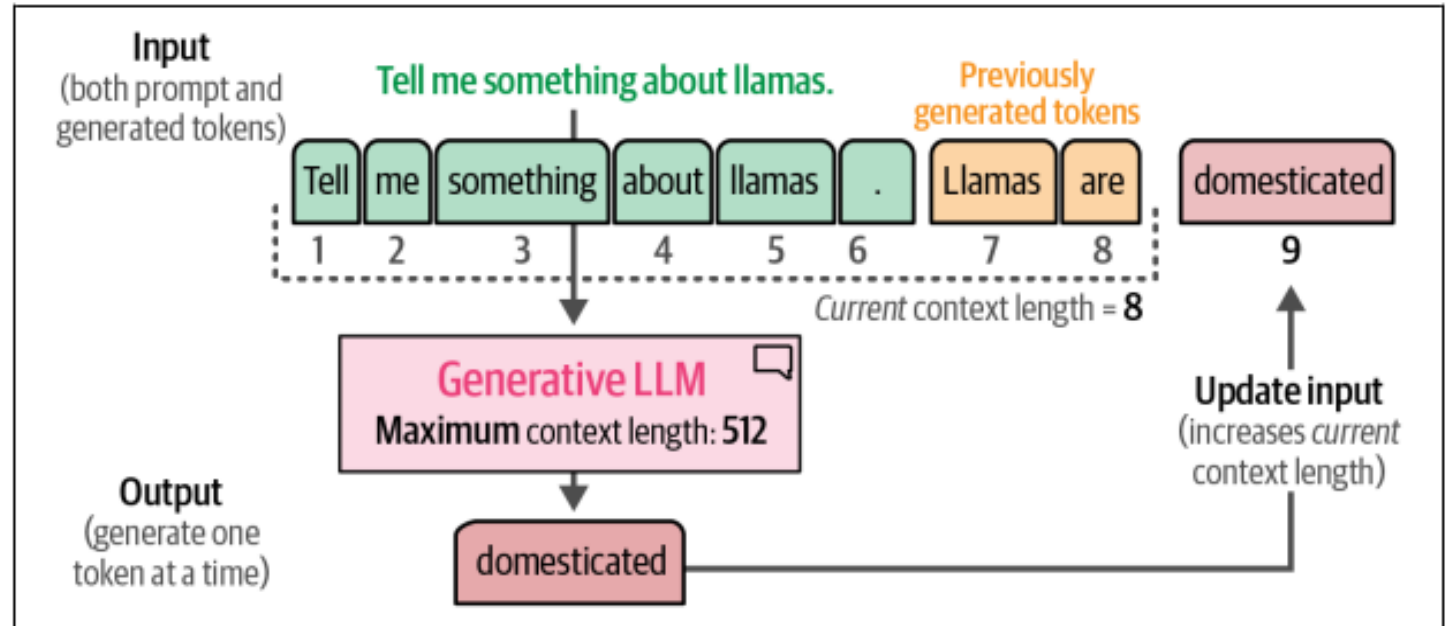


Figura : consumo progresivo del contexto al generar.

# Preentrenamiento vs ajuste fino

- El pretraining aprende capacidades generales
- El fine-tuning especializa el modelo para una tarea concreta con datos curados y menor costo relativo.

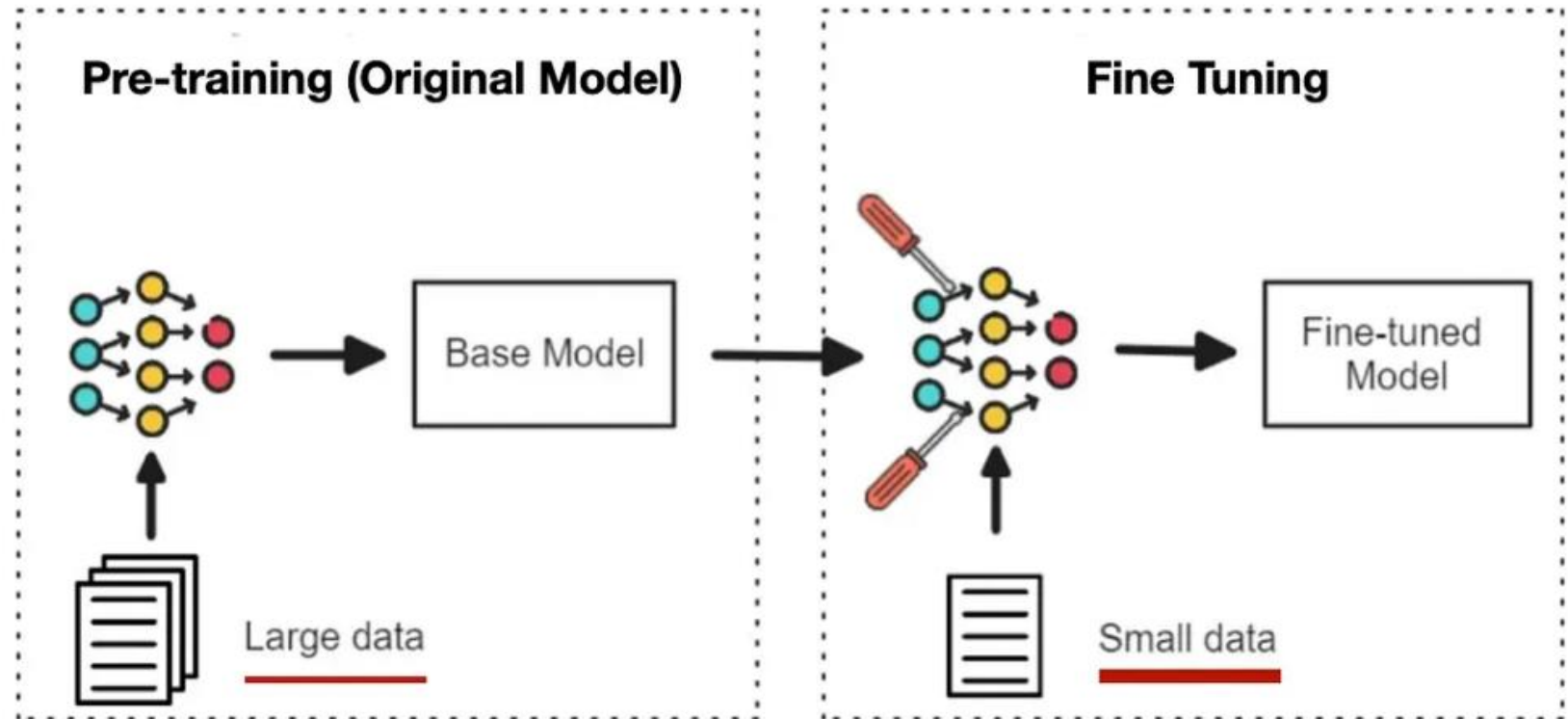


Figura :De modelo base a modelo ajustado: pretraining y fine-tuning.

# Instruction tuning (SFT)

- SFT: entrenar con instrucción->respuesta para seguir instrucciones.
- PEFT (LoRA/QLoRA) reduce costo de ajuste.
- Importa el formato del prompt y la curación de datos.

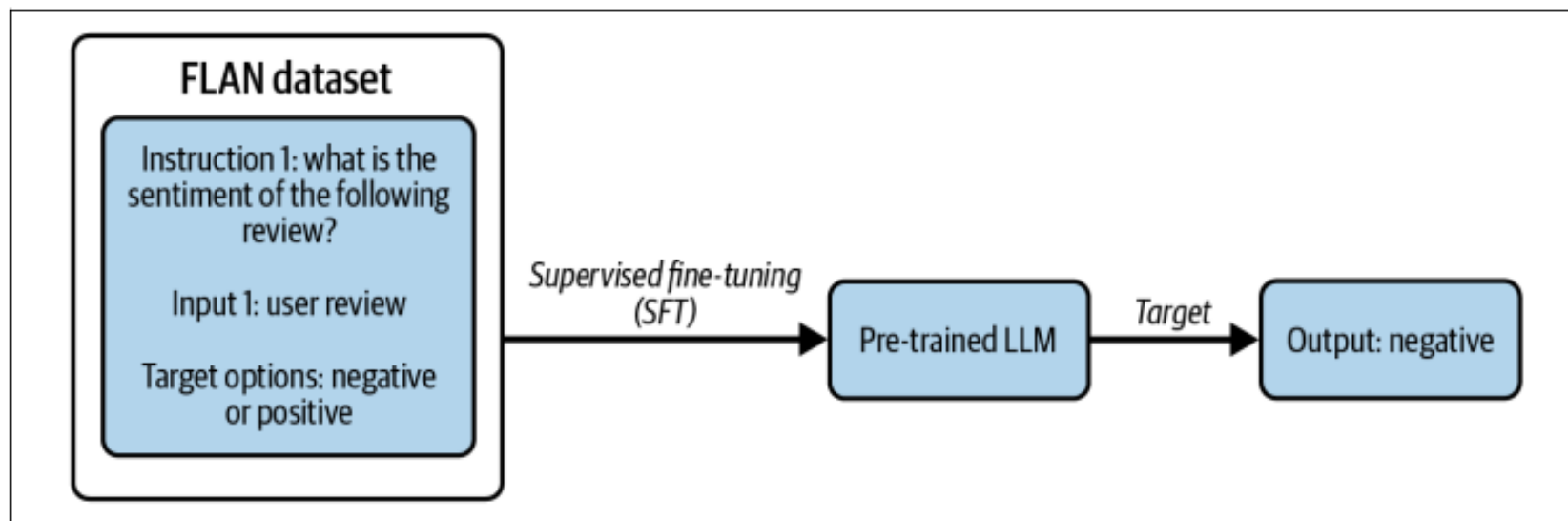


Figura : flujo de SFT sobre un LLM preentrenado.

# RLHF y modelo de recompensa

- Preferencias humanas -> reward model -> política alineada.
- Potente, pero más costoso y complejo que DPO/ORPO.

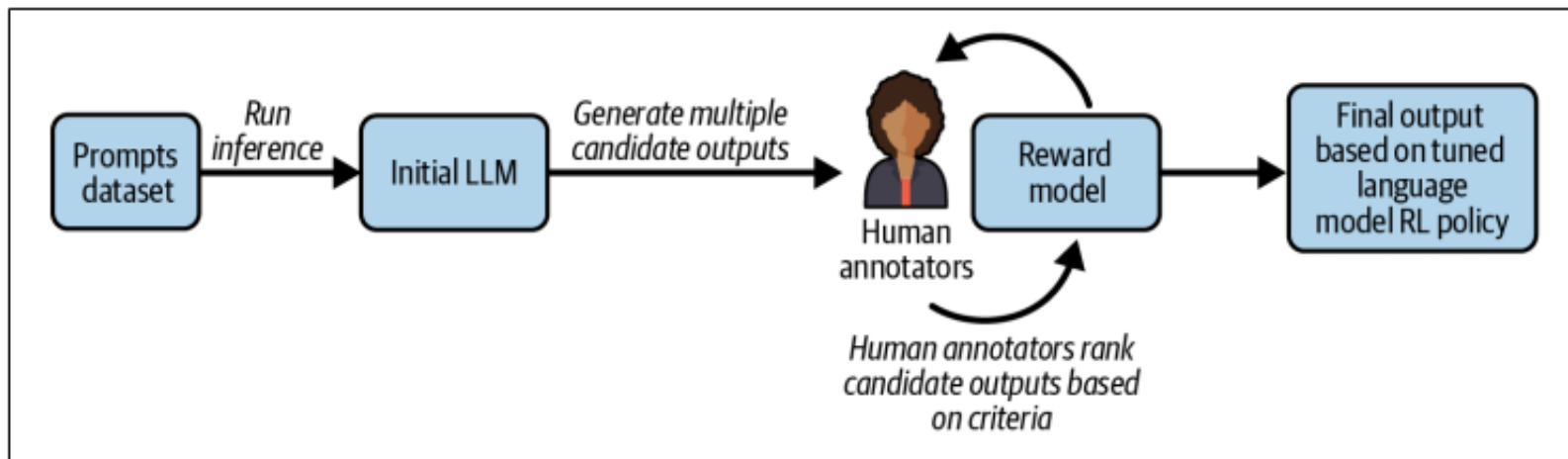


Figura : pipeline RLHF (anotadores -> reward model -> política).

# DPO: preferencia sin PPO clásico

- Usa pares aceptada/rechazada frente a un modelo referencia.
- Aumenta probabilidad relativa de la respuesta aceptada.

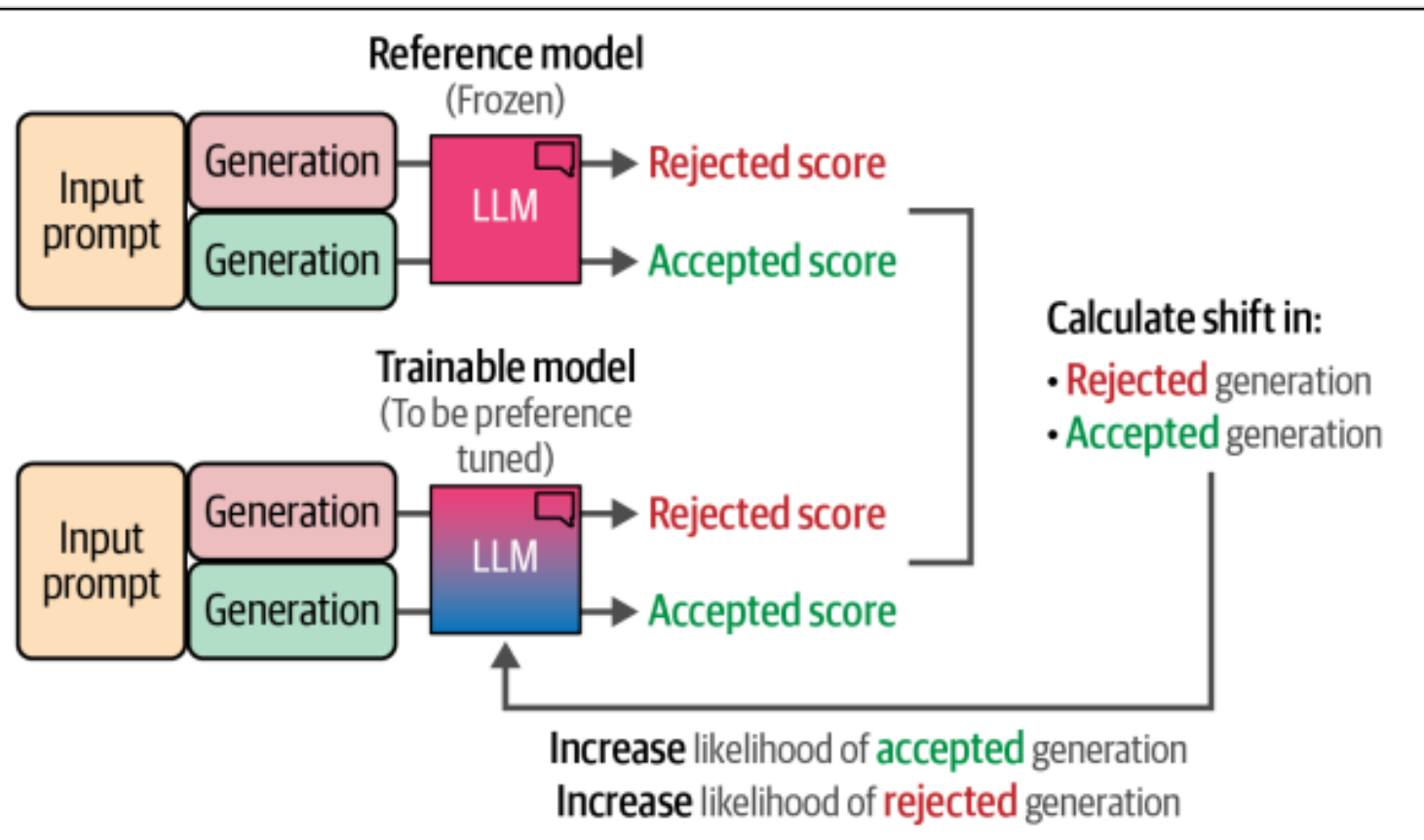


Figura : intuición de DPO con modelo de referencia.

# ORPO + panorama (RLHF/DPO/ORPO)

- Resumen visual de RLHF, DPO y ORPO.
- ORPO integra preferencia y objetivo de lenguaje.

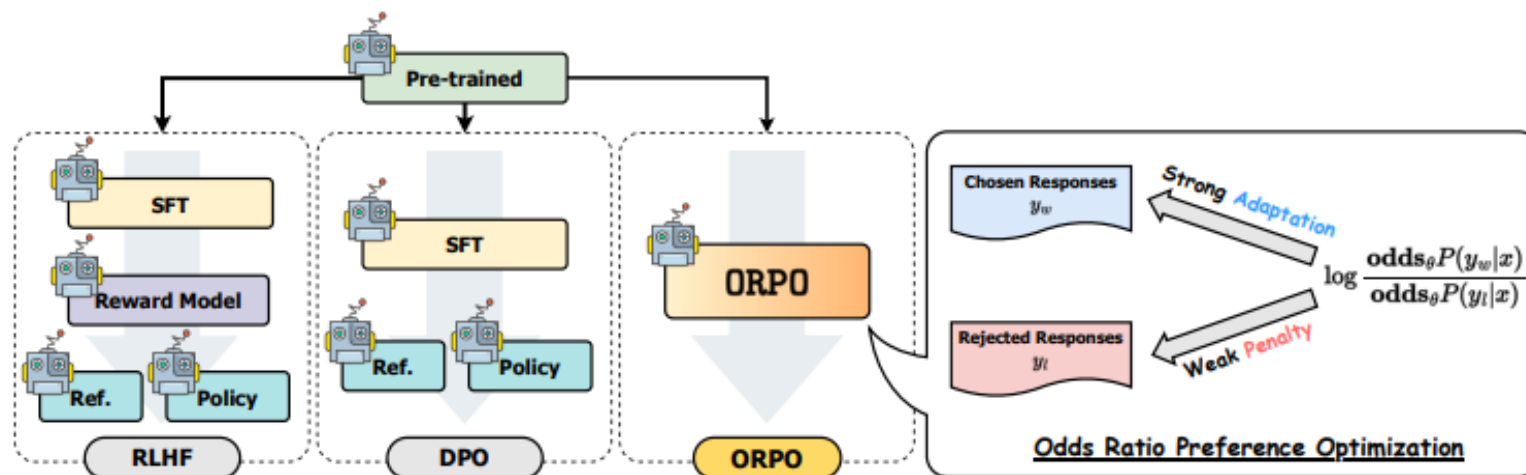


Figura: comparación RLHF/DPO/ORPO + ORPO.

# Agentes y uso de herramientas

- Herramientas convierten preguntas en acciones verificables.
- Observación de herramienta vuelve al contexto del LLM.

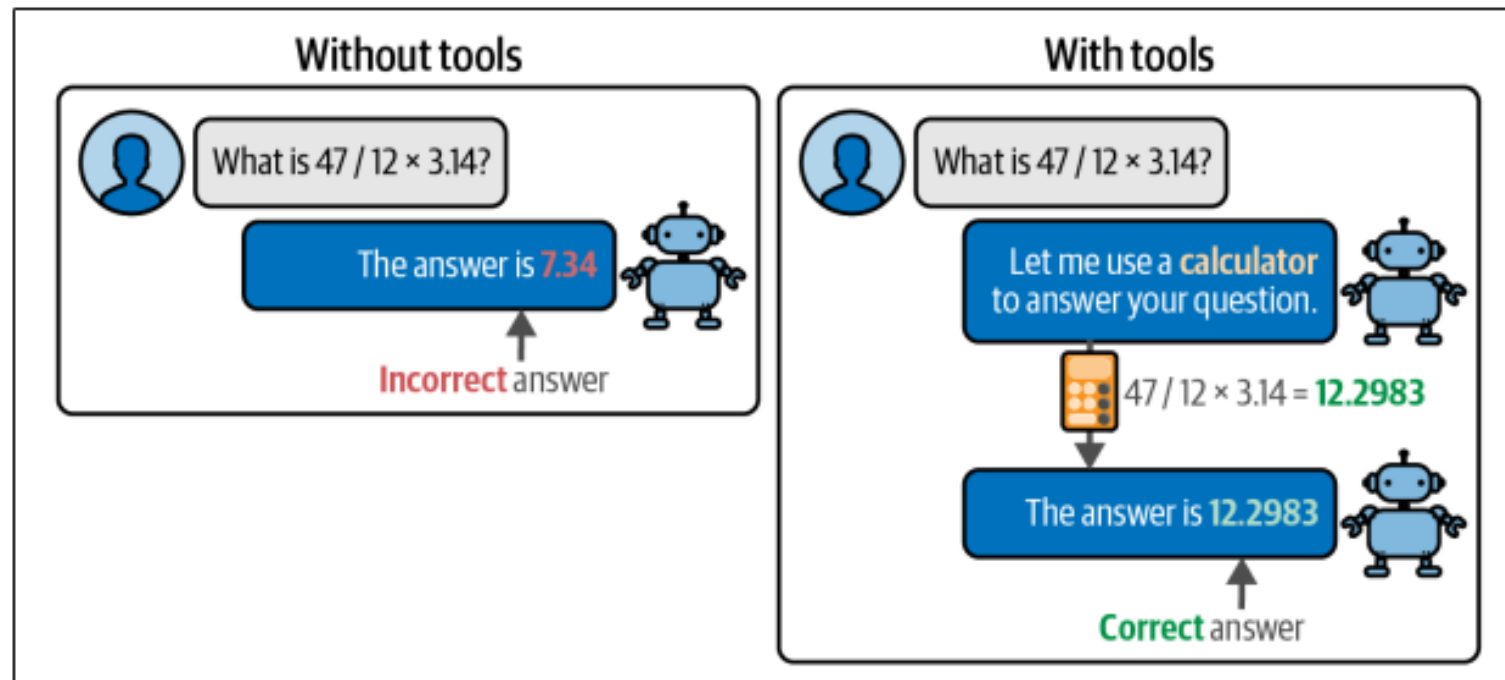


Figura : sin tools vs con tools (calculadora).

# Diagrama conceptual de agente

- Loop: planificar -> actuar -> observar -> actualizar memoria -> decidir siguiente acción.
- Separar: política (LLM) vs herramientas (acciones) vs memoria (estado).

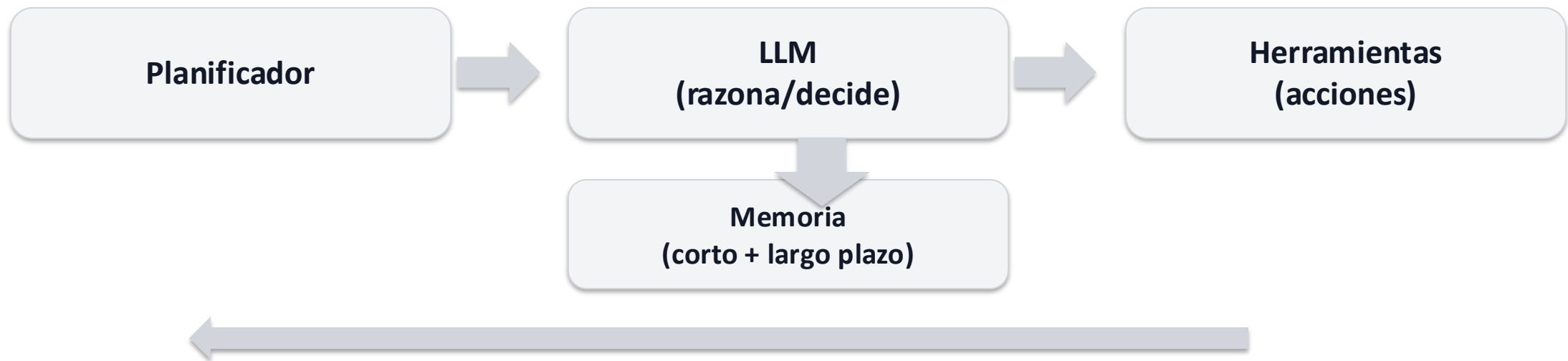


Diagrama: componentes mínimos y flujo de control de un agente LLM.

# ReAct: pensamiento-acción-observación

- Estructura el control del agente y hace trazable el proceso.
- Se combina con herramientas y RAG.

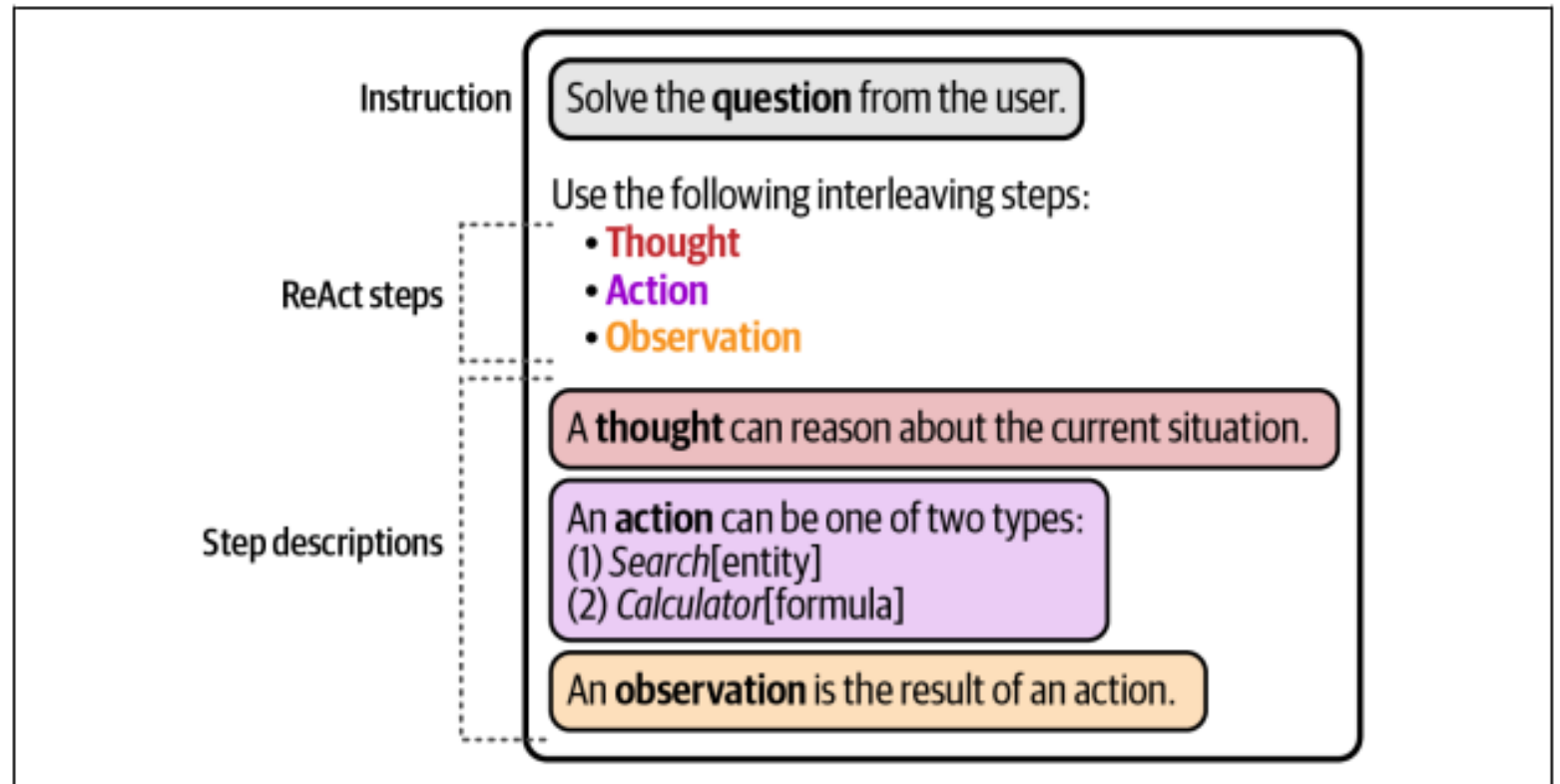


Figura: plantilla ReAct.

# ReAct en ciclos con herramientas

- Ciclos cortos de pensar-actuar-observar.
- Evaluar: éxito, pasos, latencia/costo, robustez y seguridad.

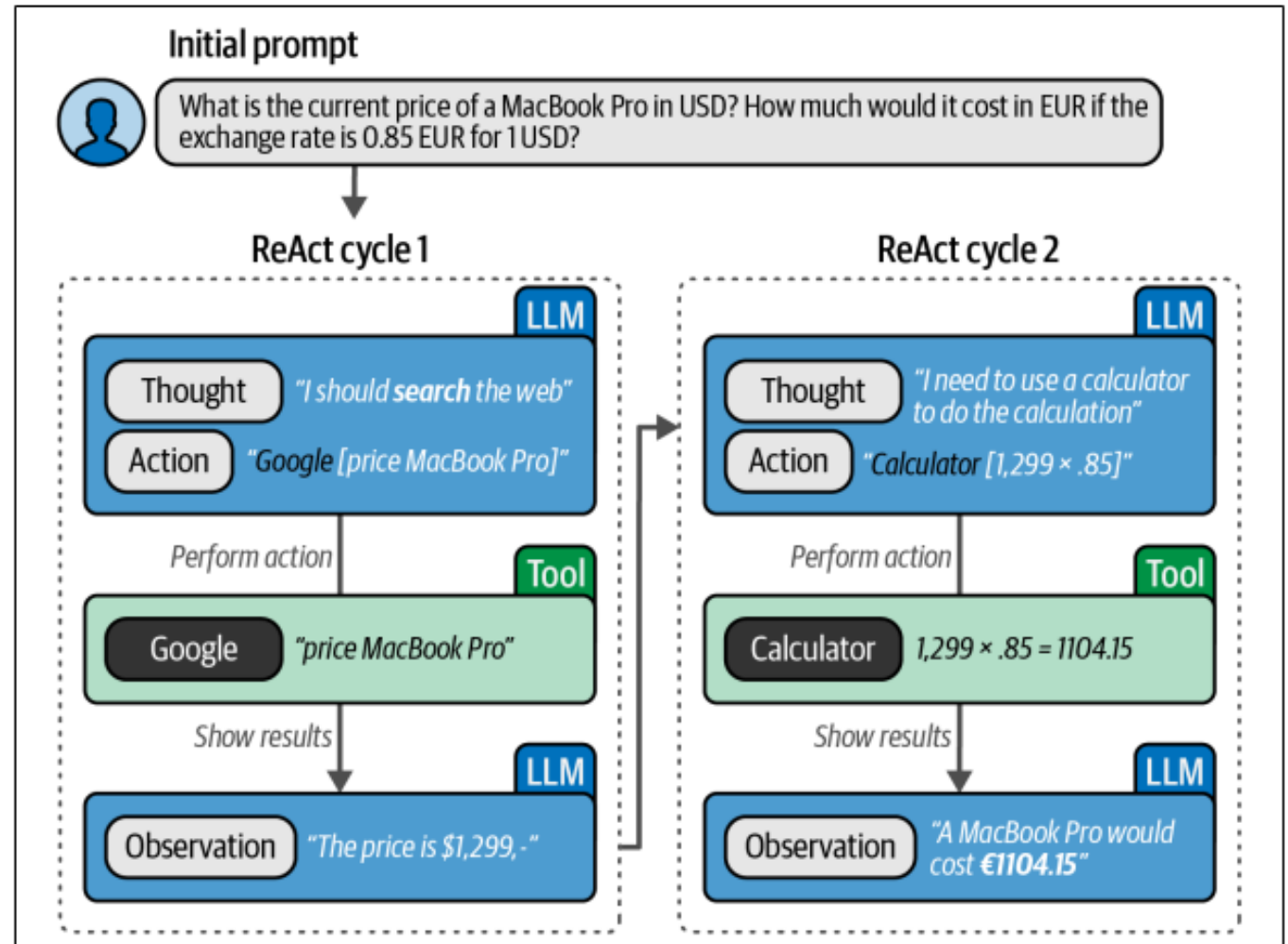


Figura: dos ciclos ReAct (búsqueda + cálculo).

# Riesgos, sesgos y ética básica

## Riesgos frecuentes

- Alucinaciones y sobreconfianza.
- Sesgos por datos/prompts/criterios de evaluación.
- Privacidad: PII y fuga de información.
- Uso indebido: fraude, spam, desinformación.

## Prácticas mínimas

- Gobernanza de datos (origen, licencias, limpieza).
- Evaluación continua + red teaming.
- Permisos mínimos para tools y logging.
- Human-in-the-loop en tareas sensibles.