Flujos de trabajo con agentes de IA en Python

Rafael Mena Yedra
Manuel Díaz Mendez
Borja Esteve Molner

Índice

01. Introducción

02. Contextualización sobre Modelos Grandes de Lenguaje (LLMs)

03. De LLM a Agente de IA

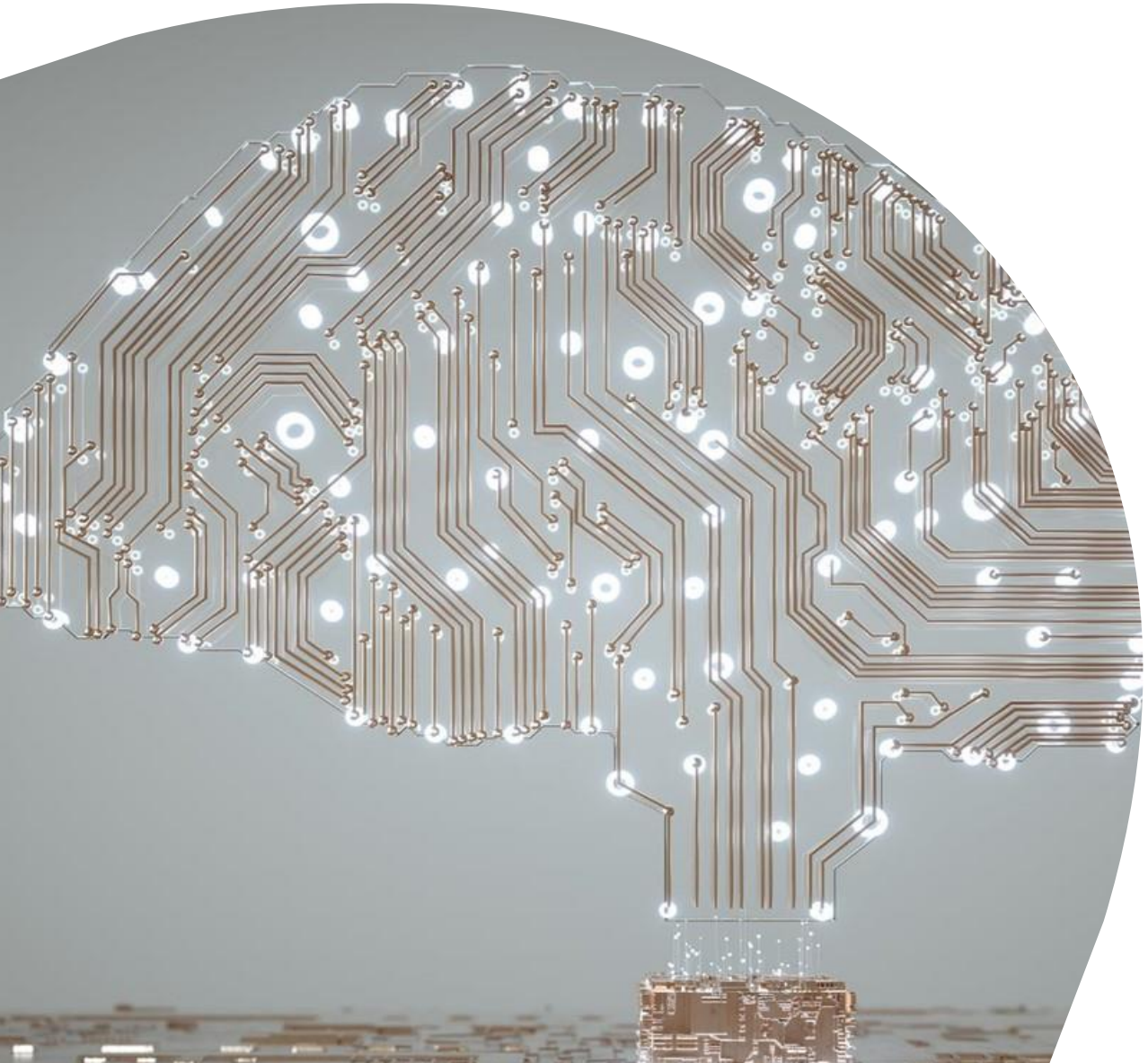
04. Motivaciones y Ventajas de los Agentes de IA

05. Frameworks y aplicaciones

06. Patrones de Diseño en Flujos de Trabajo con Agentes de IA

07. Caso Práctico: Construcción de un Flujo de Trabajo Multiagente

08. Perspectivas Futuras y Conclusión

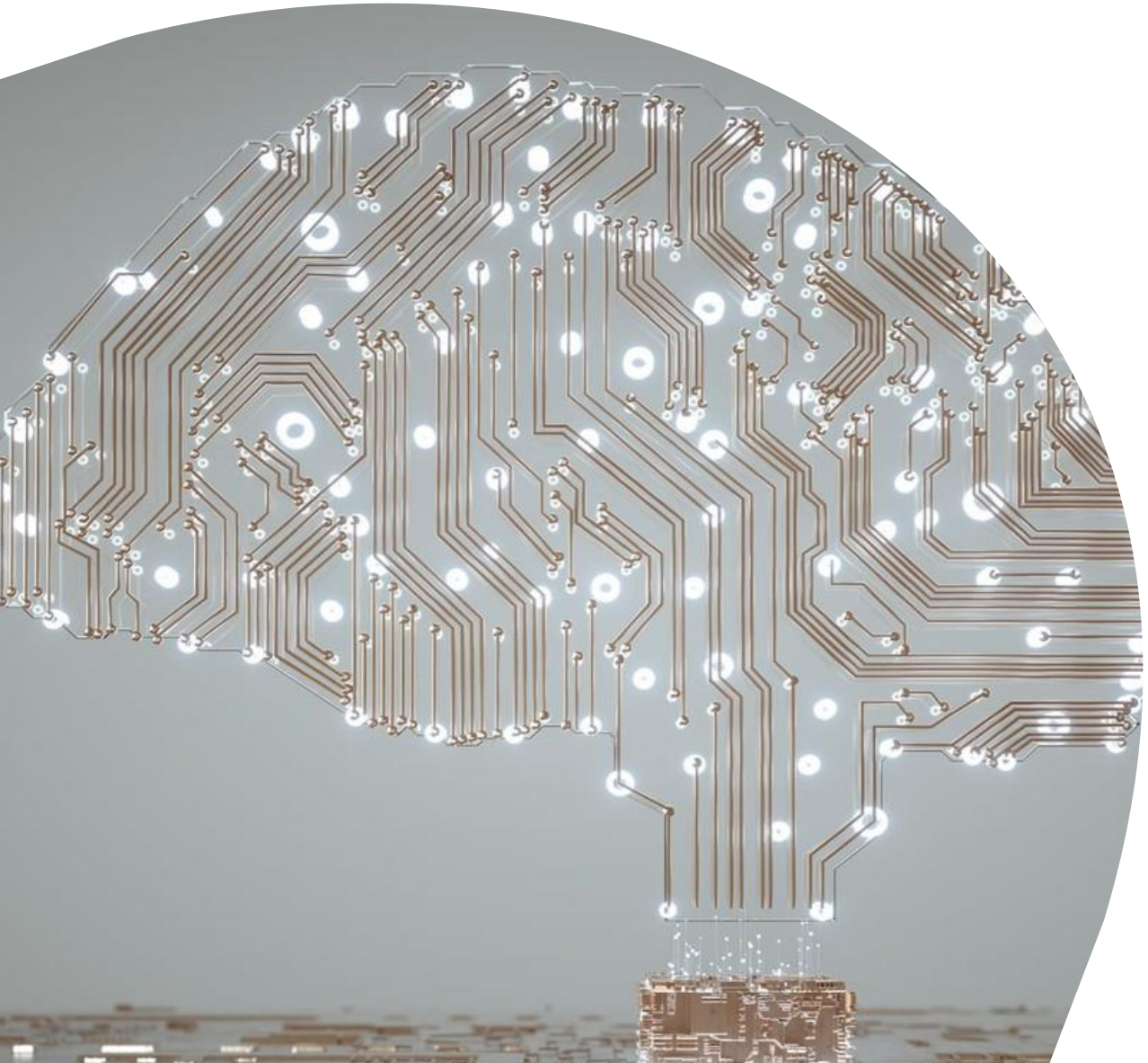


Introducción

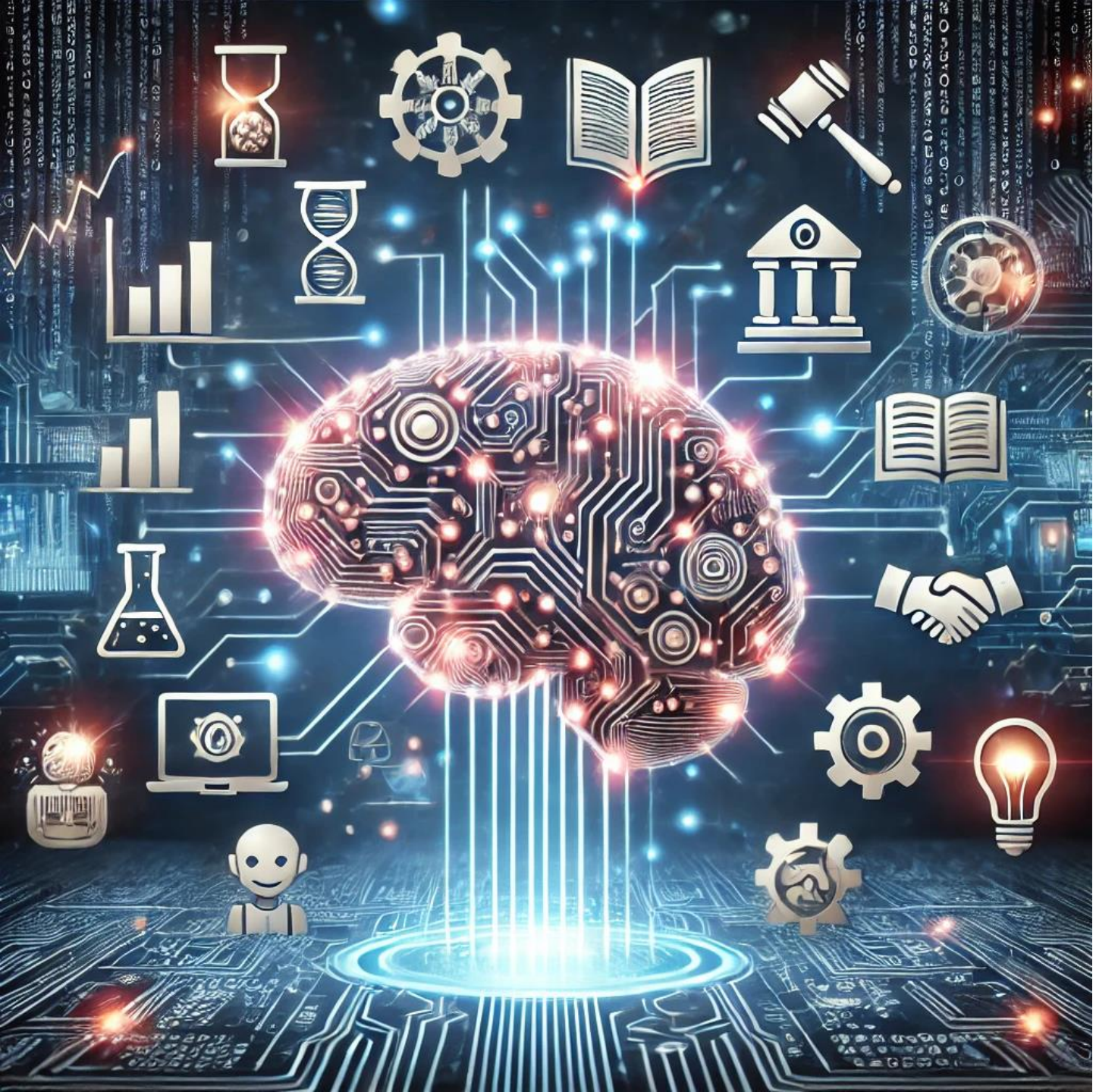
Breve historia de los agentes de IA.

- Existen desde hace décadas.
  - Agentes basados en reglas (MYCIN)
  - Agentes reactivos (Deep Blue)
  - Aprendizaje por refuerzo
  - Aprendizaje por refuerzo profundo (AlphaGo)
- Limitados a tareas específicas.
- Nuevo impulso con LLMs.

AlphaGo



Contextualización de los LLMS



¿Qué es un LLM y por qué son importantes?

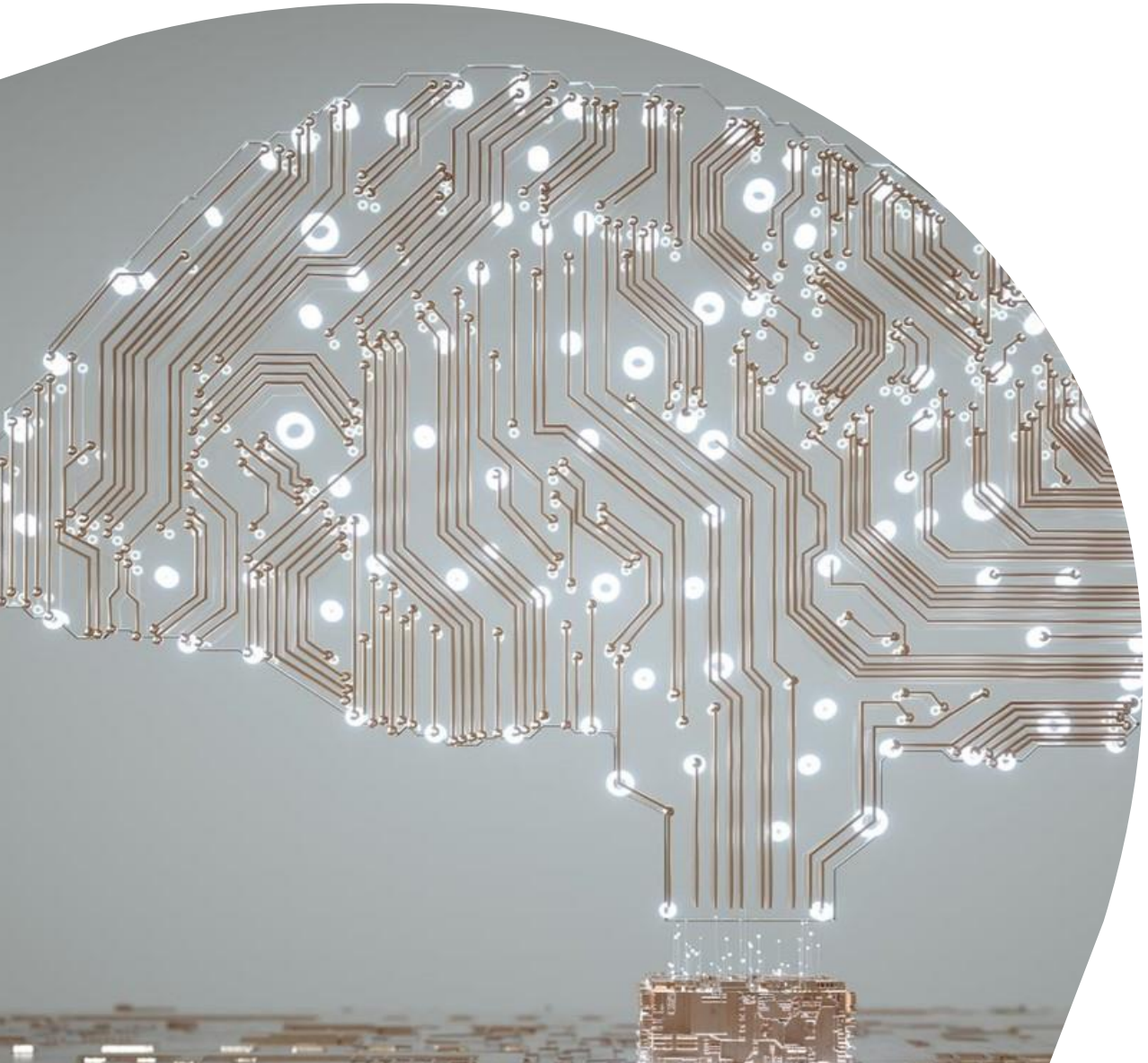
- Los Transformers potenciaron el NLP.
- Aparición de modelos de lenguaje y LLMs.
- Importancia:
  - Generación y comprensión de texto
  - Transferencia de conocimiento
  - Razonamiento complejo

Wei, J., et al. (2022). *Chain of Thought Prompting Elicits Reasoning in Large Language Models.* arXiv:2201.11903

Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Brown, T., et al. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165

Generative AI exists because of the transformer. (2023, 12 septiembre). *Financial Times.* https://ig.ft.com/generative-ai/



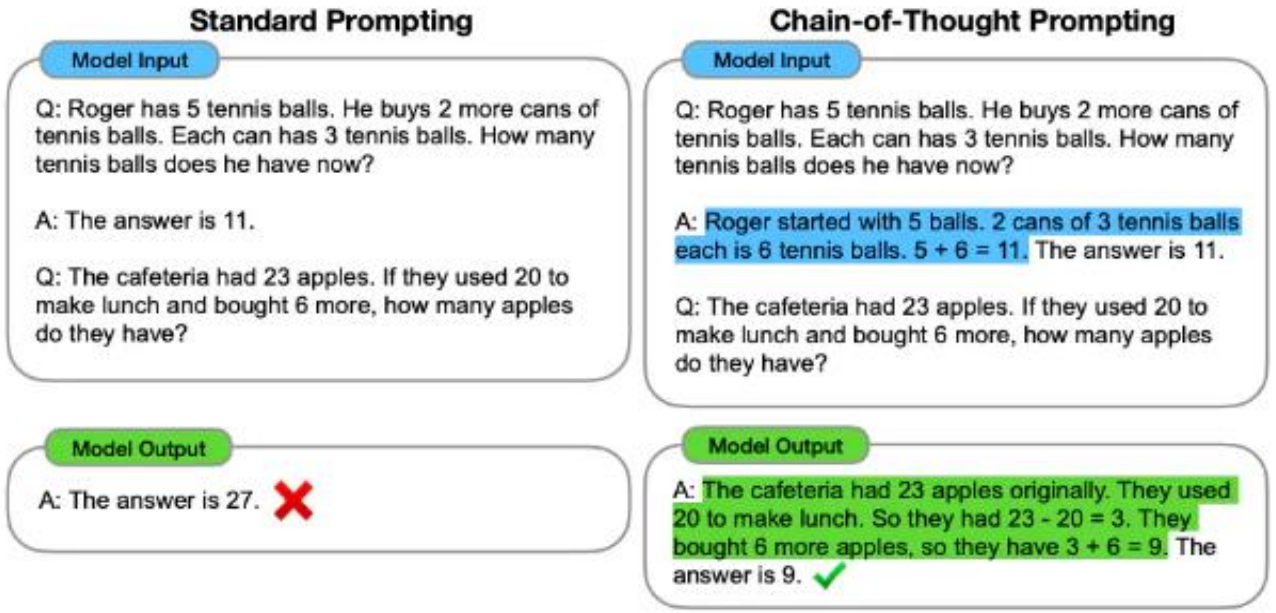
De LLM a Agente de IA

De LLM a Agente de IA

- **Chain of Thought** Wei, J., et al. (2022). *Chain of Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903
- **ReAct: Synergizing Reasoning and Acting** Yao, S., et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629
- **Toolformer** Schick, T., et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761

Chain of Thought

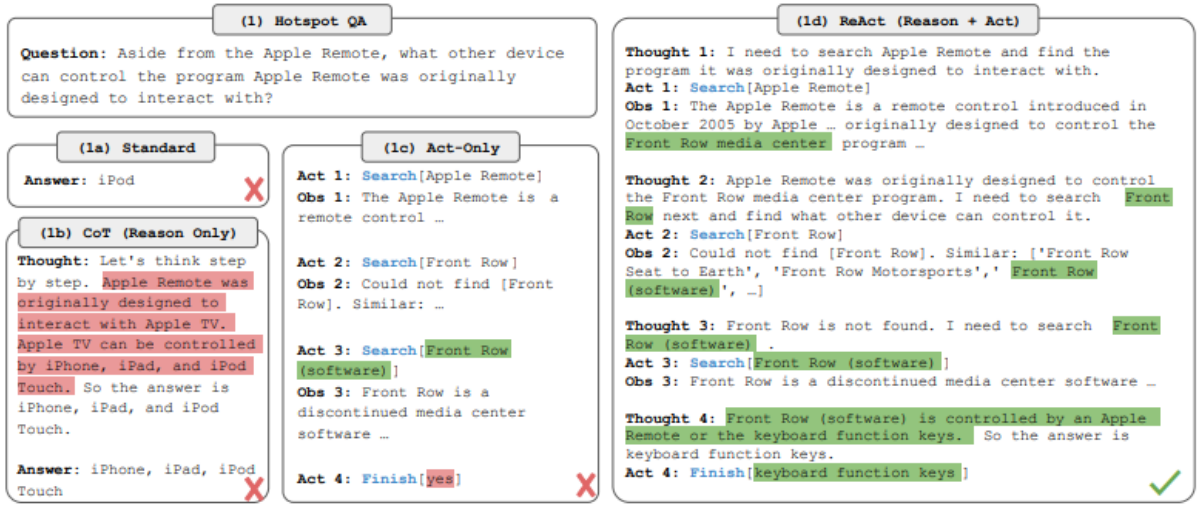
- **Chain of Thought**
  - Introduce la técnica de chain-of-thought prompting, permitiendo a los LLMs descomponer problemas complejos en pasos de razonamiento más simples.



Wei, J., et al. (2022). *Chain of Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903

ReAct

- **ReAct: Synergizing Reasoning and Acting**. Establece el ciclo:
  - **Actuar:** El agente llama a herramientas o toma acciones.
  - **Observar:** Evalúa los resultados de esas acciones.
  - **Razonar:** Decide si realizar una nueva acción o generar una respuesta final.



Yao, S., et al. (2022). *ReAct: Synergizing Reasoning and Acting in Language Models*. arXiv:2210.03629.

Toolformer

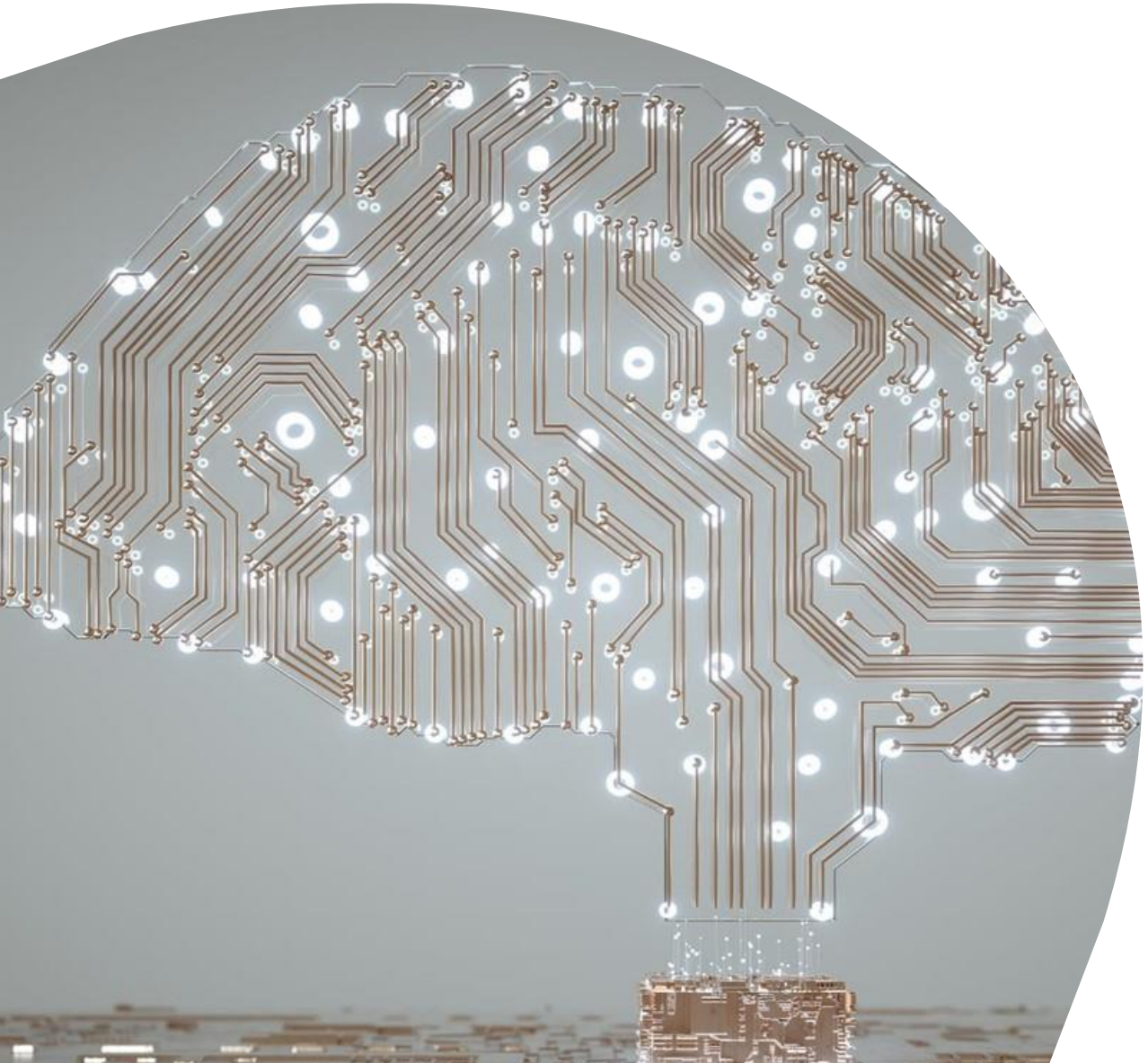
- **Toolformer**
  - Introduce el concepto de LLMs que deciden autónomamente cuándo y cómo usar herramientas externas para completar tareas, una característica clave en los sistemas de agentes modernos.

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick **Jane Dwivedi-Yu** **Roberto Dessì**[†] **Roberta Raileanu**
Maria Lomeli **Luke Zettlemoyer** **Nicola Cancedda** **Thomas Scialom**

Meta AI Research [†]Universitat Pompeu Fabra

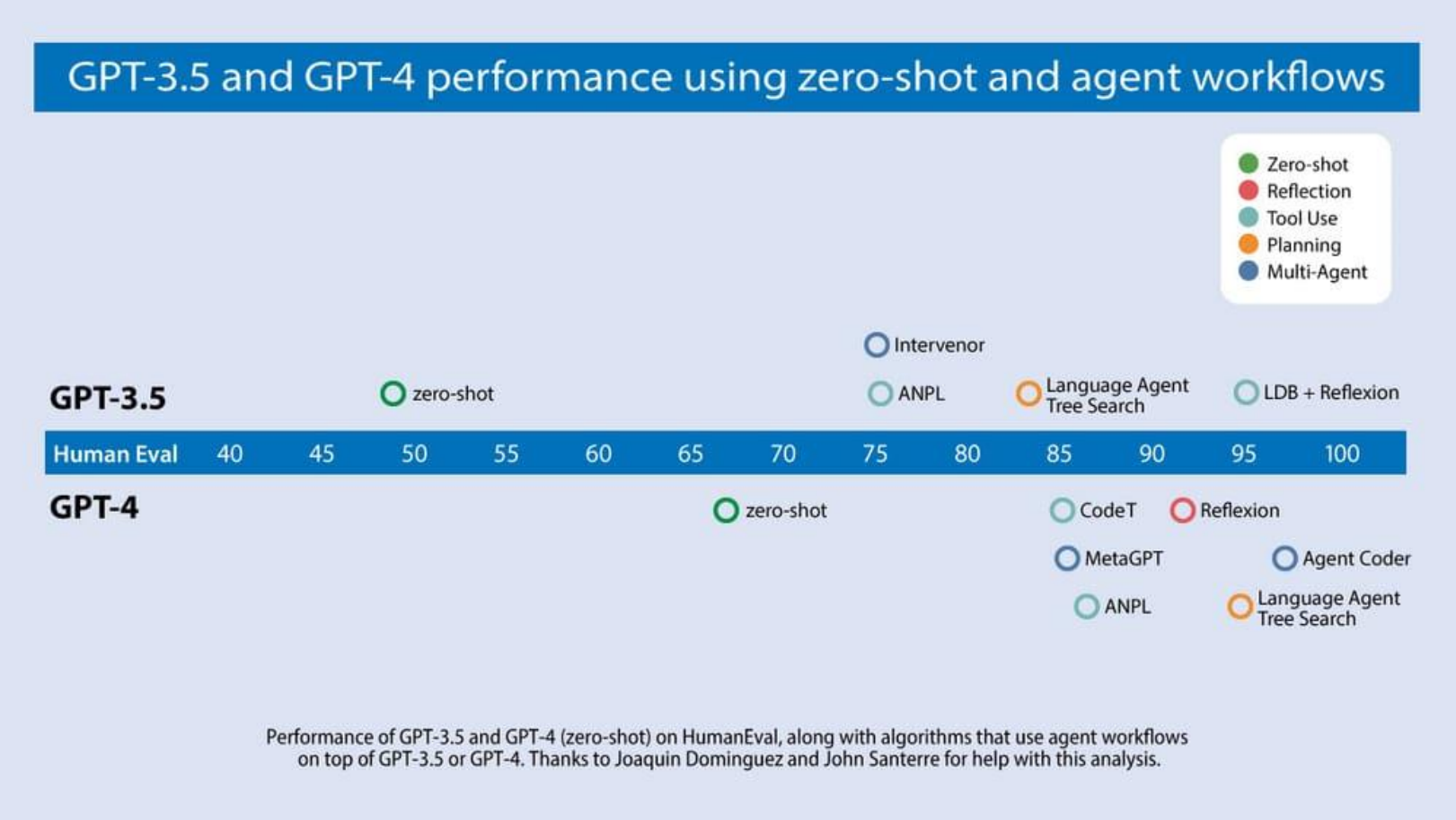
Schick, T., et al. (2023). *Toolformer: Language Models Can Teach Themselves to Use Tools*. arXiv:2302.04761.



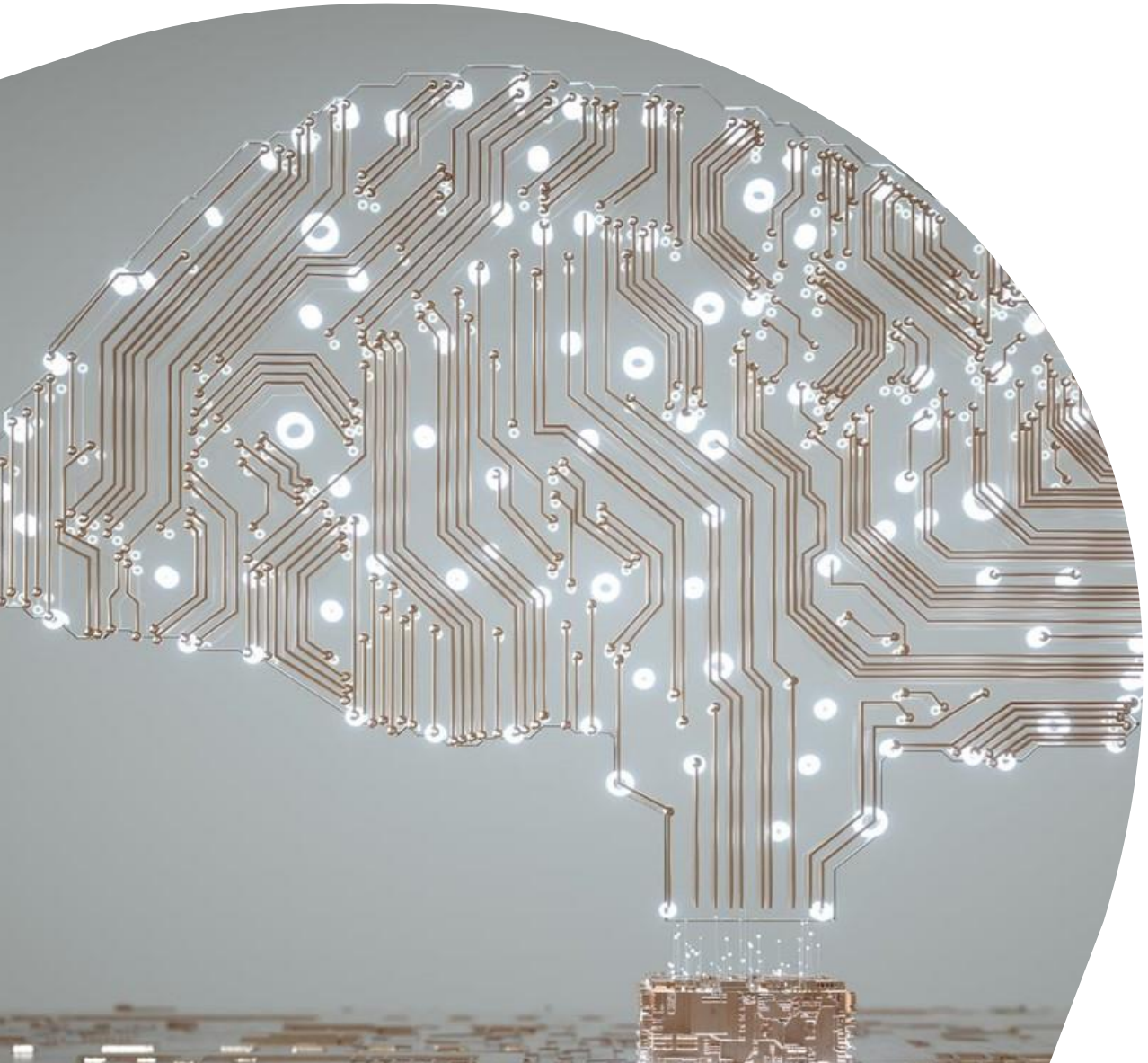
Motivación y ventajas de los agentes de IA

Motivación

Automatización y mejora en la calidad de la respuesta.



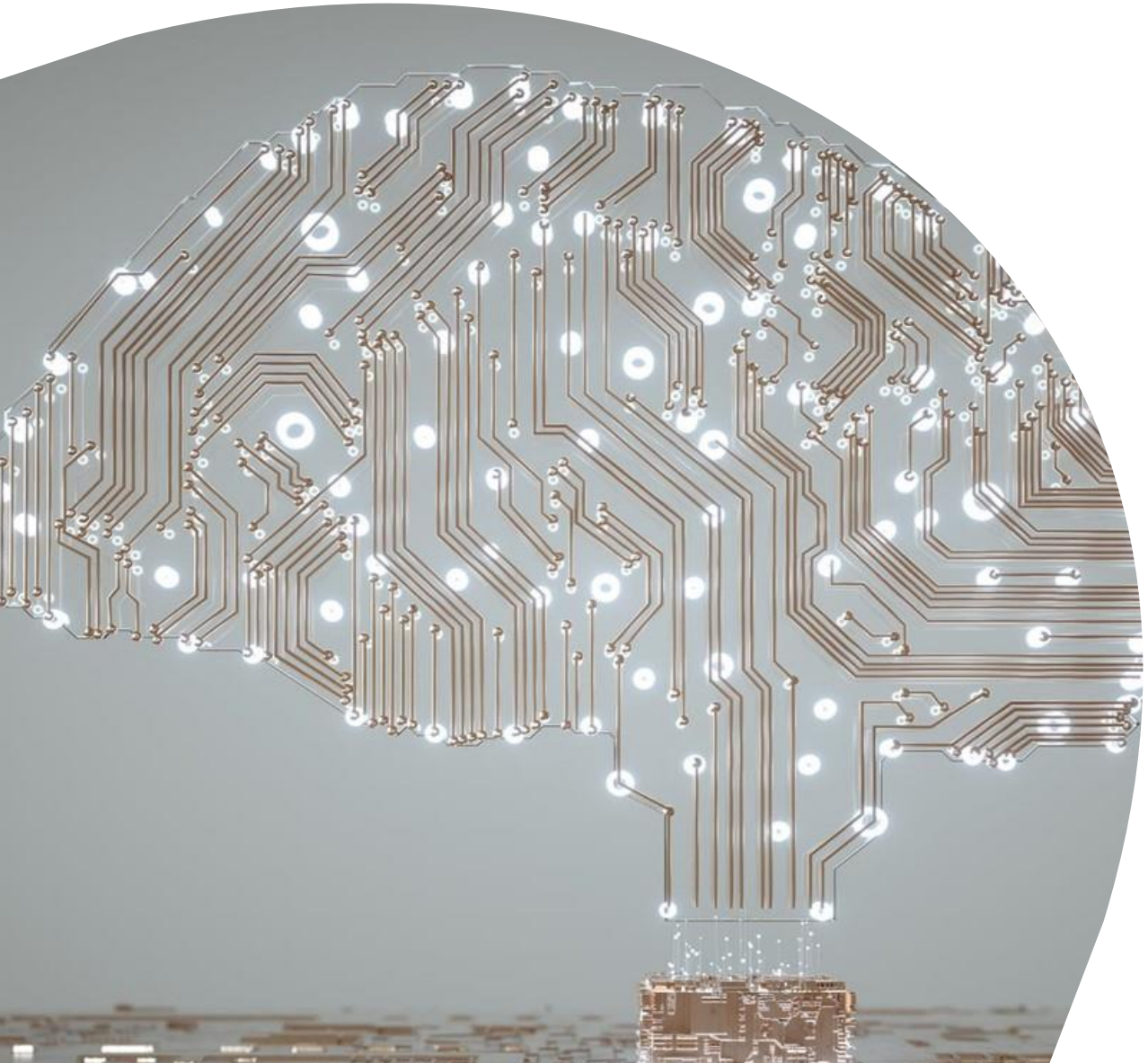
Ng, Andrew. (2023). *Agentic Workflows and Large Language Models: Performance Comparisons on HumanEval Benchmarks.* DeepLearning.AI.



Aplicaciones con agentes IA

Aplicaciones con agentes IA

- Ingeniería de software
  - Devin: https://devin.ai/
  - Devika: https://github.com/stitionai/devika
  - OpenHands: https://github.com/All-Hands-AI/OpenHands
- Investigación
  - The AI Scientist: https://github.com/SakanaAI/AI-Scientist
- Propósito general
  - Ailice: https://github.com/myshell-ai/AIlice



Patrones de diseño en flujos de trabajo con agentes de IA

Conceptos fundamentales del diseño basado en agentes

Reflexión

- El modelo critica y mejora sus propias salidas, iterando para refinar el resultado. Ejemplo: revisión y optimización de código generado.

Liu, Y., Lo, S. K., Lu, Q., et al. (2024). *Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents*. arXiv: 2405.10467

Conceptos fundamentales del diseño basado en agentes

Uso de herramientas

- El modelo utiliza recursos externos como APIs o herramientas especializadas (calculadoras, búsquedas web) para resolver tareas más complejas.

Liu, Y., Lo, S. K., Lu, Q., et al. (2024). Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents. arXiv: 2405.10467

Conceptos fundamentales del diseño basado en agentes

Planificación

- Descompone tareas en pasos pequeños y ejecutables, permitiendo una mejor organización y coordinación en tareas complejas.

Liu, Y., Lo, S. K., Lu, Q., et al. (2024). Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents. arXiv: 2405.10467

Conceptos fundamentales del diseño basado en agentes

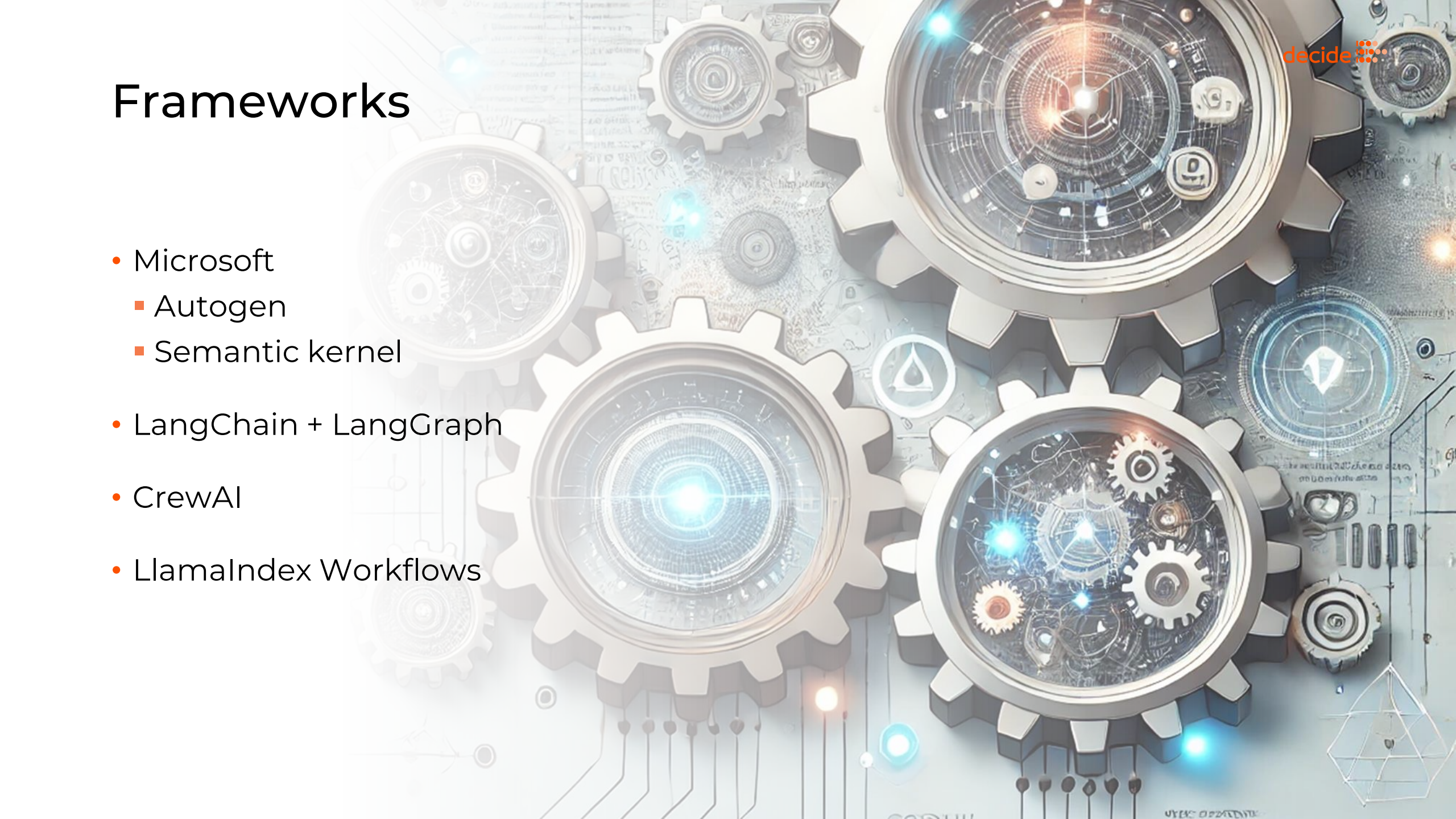
Colaboración entre múltiples agentes

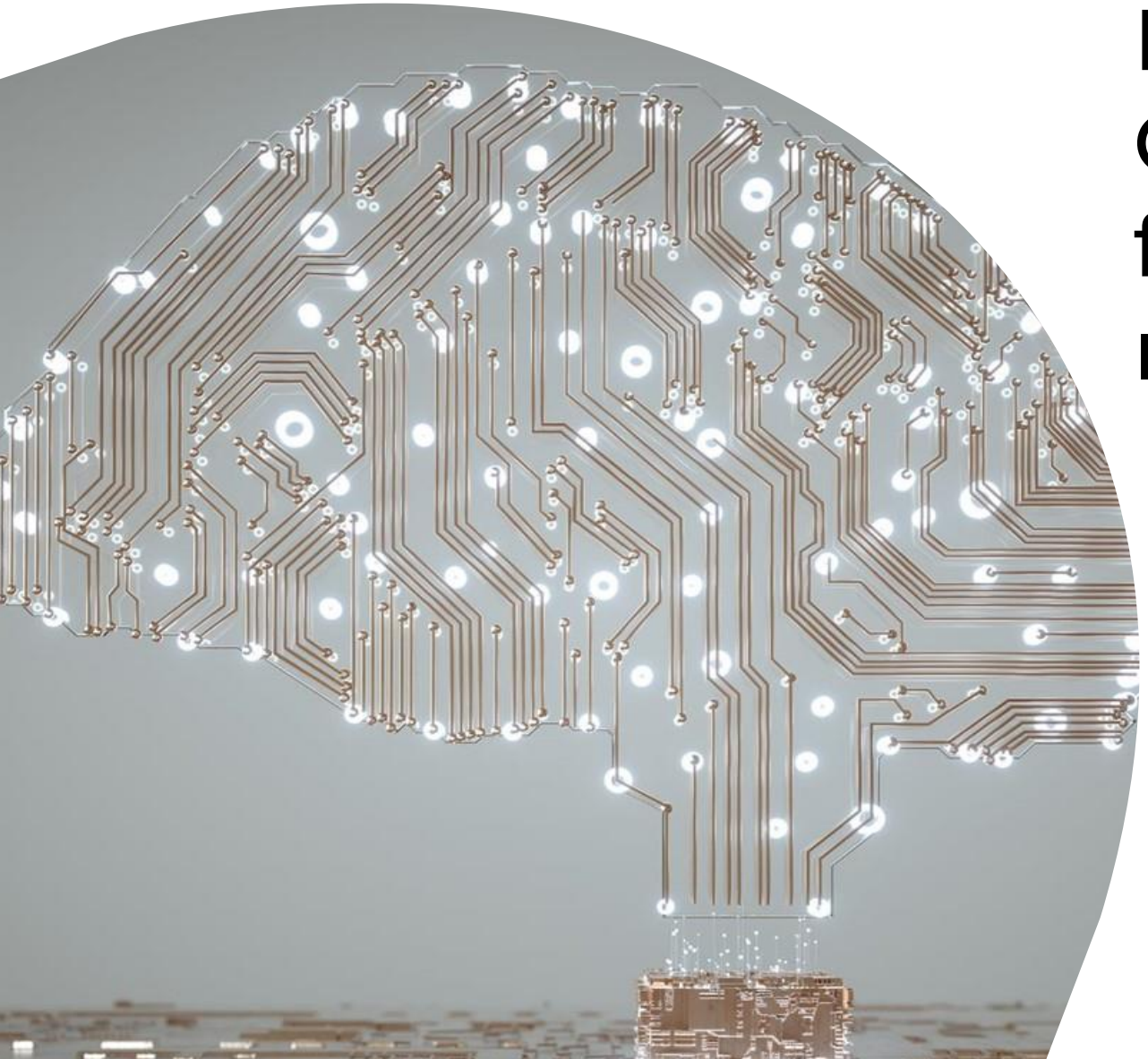
- Varios agentes trabajan juntos, con roles especializados, comunicándose y cooperando para completar tareas más eficientemente.

Liu, Y., Lo, S. K., Lu, Q., et al. (2024). *Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents.* arXiv: 2405.10467

Frameworks

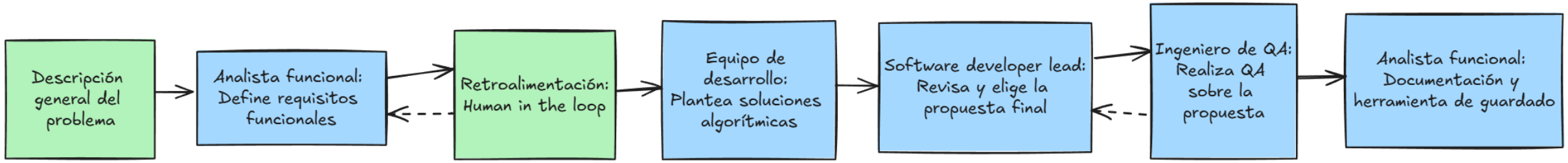
- Microsoft
  - Autogen
  - Semantic kernel
- LangChain + LangGraph
- CrewAI
- LlamaIndex Workflows





Ejemplo práctico: Construcción de un flujo de trabajo multiagente en Python

Ejemplo: Workflow colaborativo de desarrollo de software

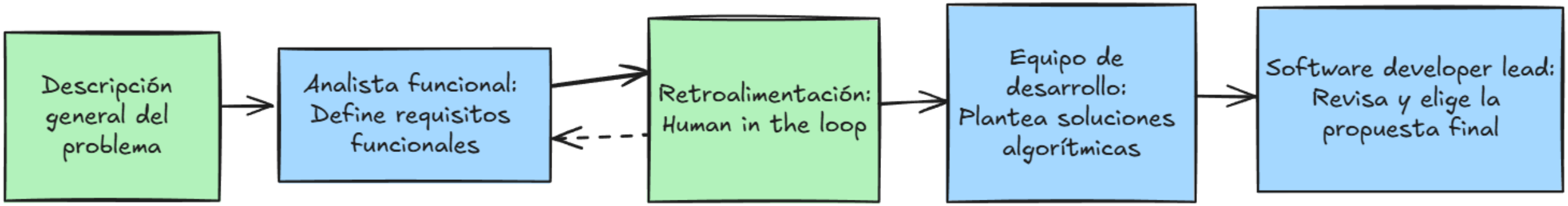


LangGraph

Workflow colaborativo de desarrollo de software (I)

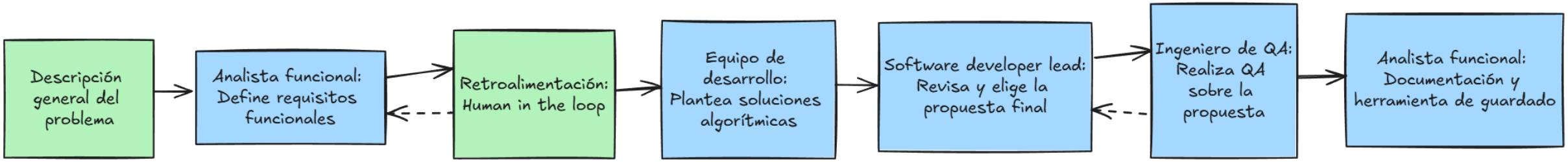
1. El proceso arranca con la Analista Funcional, quien traduce la descripción del problema del usuario en requisitos funcionales claros.
2. A partir de aquí, se abre un ciclo de retroalimentación continua (human-in-the-loop), donde los involucrados colaboran activamente para asegurar una alineación precisa.

Workflow colaborativo de desarrollo de software (II)



3. El equipo de desarrollo propone distintas soluciones algorítmicas, buscando diversidad en sus soluciones a la par que se cumplen los requisitos definidos.
4. Las propuestas son revisadas por el Software Developer Lead, quien selecciona, ajusta o combina las propuestas para consolidar una solución final.

Workflow colaborativo de desarrollo de software (III)



4. El Ingeniero de QA realiza una exhaustiva validación, verificando que la propuesta cumpla con los requisitos y asegure la calidad del código. Este proceso de mejora se itera con el agente Software Developer Lead hasta alcanzar la versión óptima.
5. Finalmente, la Analista Funcional documenta la solución y asegura su correcto guardado a través de herramientas.

Detalles del modelado con LangGraph: nodos, aristas y estado.

LangGraph modela los flujos de trabajo de los agentes como grafos, utilizando tres componentes clave para definir el comportamiento de los agentes:

- **Estado**: Estructura de datos compartida que representa el estado actual de la aplicación.
- **Nodos**: Funciones de Python que procesan el estado y actualizan el flujo.
- **Aristas**: Determinan qué nodo ejecutar a continuación, según el estado actual.

Estado del grafo y memoria

```python
from langgraph.graph import MessagesState

class MessagesState(TypedDict):
    messages: Annotated[list[AnyMessage], add_messages]

class CodingWorkflowState(MessagesState):
    general_description: str
    functional_requirements: str
    human_functional_feedback: str
    num_software_developers: int
    algorithmic_proposals: str
    algorithmic_proposal: str
    qa_suggestions: str
    final_solution_with_doc: str
```

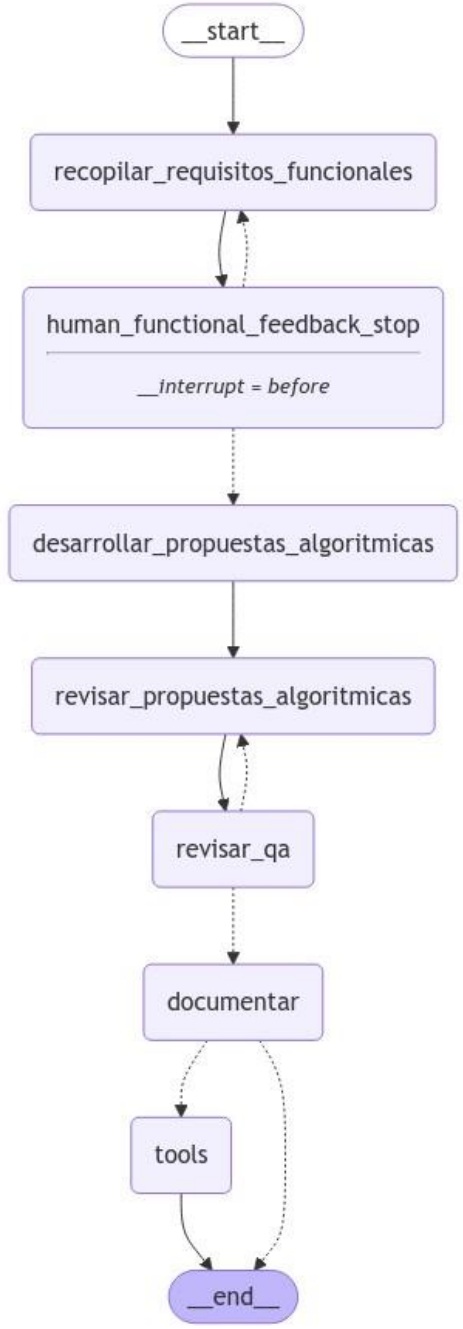
- El estado guarda el contexto de la conversación en `messages` (una lista).
- `Checkpointers` permiten persistir la ejecución del grafo y retomarla más tarde.
- La persistencia puede hacerse con bases de datos como SQLite o Postgres.

Creación del grafo

```python
builder = StateGraph(CodingWorkflowState)
builder.add_node("recopilar_requisitos_funcionales", recopilar_requisitos_funcionales)
builder.add_node("human_functional_feedback_stop", human_functional_feedback_stop)
builder.add_node(
    "desarrollar_propuestas_algoritmicas", desarrollar_propuestas_algoritmicas
)
builder.add_node("revisar_propuestas_algoritmicas", revisar_propuestas_algoritmicas)
builder.add_node("revisar_qa", revisar_qa)
builder.add_node("documentar", documentar)
builder.add_node("tools", ToolNode(tools))

builder.add_edge(START, "recopilar_requisitos_funcionales")
builder.add_edge("recopilar_requisitos_funcionales", "human_functional_feedback_stop")
builder.add_conditional_edges(
    "human_functional_feedback_stop",
    validar_requisitos_funcionales,
    ["recopilar_requisitos_funcionales", "desarrollar_propuestas_algoritmicas"],
)
builder.add_edge(
    "desarrollar_propuestas_algoritmicas", "revisar_propuestas_algoritmicas"
)
builder.add_edge("revisar_propuestas_algoritmicas", "revisar_qa")
builder.add_conditional_edges(
    "revisar_qa",
    validar_qa,
    ["revisar_propuestas_algoritmicas", "documentar"],
)
builder.add_conditional_edges(
    "documentar",
    deberia_usar_herramienta,
)
builder.add_edge("tools", END)

# Compile
memory = MemorySaver()
graph = builder.compile(
    interrupt_before=["human_functional_feedback_stop"], checkpointer=memory
)
```



Reflexión, uso de herramientas y human-in-the-loop

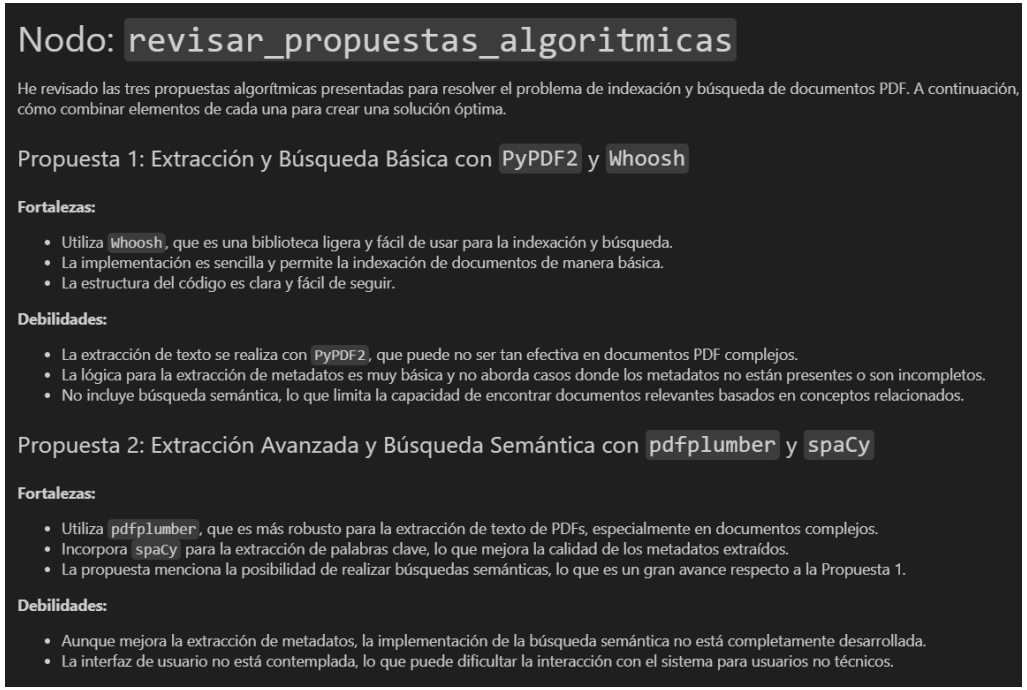
- **Reflexión**: en los agentes Lead y el ingeniero QA

- **Uso de herramientas**: Los agentes interactúan con sistemas externos usando APIs o herramientas especializadas.

- **Human-in-the-loop**: Los humanos intervienen en puntos clave para supervisar, ajustar o validar decisiones complejas.
  - Aprobar acciones específicas.
  - Proporcionar retroalimentación para actualizar el estado del agente.
  - Ofrecer orientación en procesos de toma de decisiones complejos.

Resultado

- https://github.com/rael-my/pycon24_agents
- Output: código en un .py y documentación en markdown.
- LLM utilizado: gpt4o-mini (necesario definir OPENAI_API_KEY en un .env para ejecutar el código).



Nodo: `revisar_propuestas_algoritmicas`

He revisado las tres propuestas algorítmicas presentadas para resolver el problema de indexación y búsqueda de documentos PDF. A continuación, cómo combinar elementos de cada una para crear una solución óptima.

Propuesta 1: Extracción y Búsqueda Básica con `PyPDF2` y `Whoosh`

Fortalezas:

- Utiliza `Whoosh`, que es una biblioteca ligera y fácil de usar para la indexación y búsqueda.
- La implementación es sencilla y permite la indexación de documentos de manera básica.
- La estructura del código es clara y fácil de seguir.

Debilidades:

- La extracción de texto se realiza con `PyPDF2`, que puede no ser tan efectiva en documentos PDF complejos.
- La lógica para la extracción de metadatos es muy básica y no aborda casos donde los metadatos no están presentes o son incompletos.
- No incluye búsqueda semántica, lo que limita la capacidad de encontrar documentos relevantes basados en conceptos relacionados.

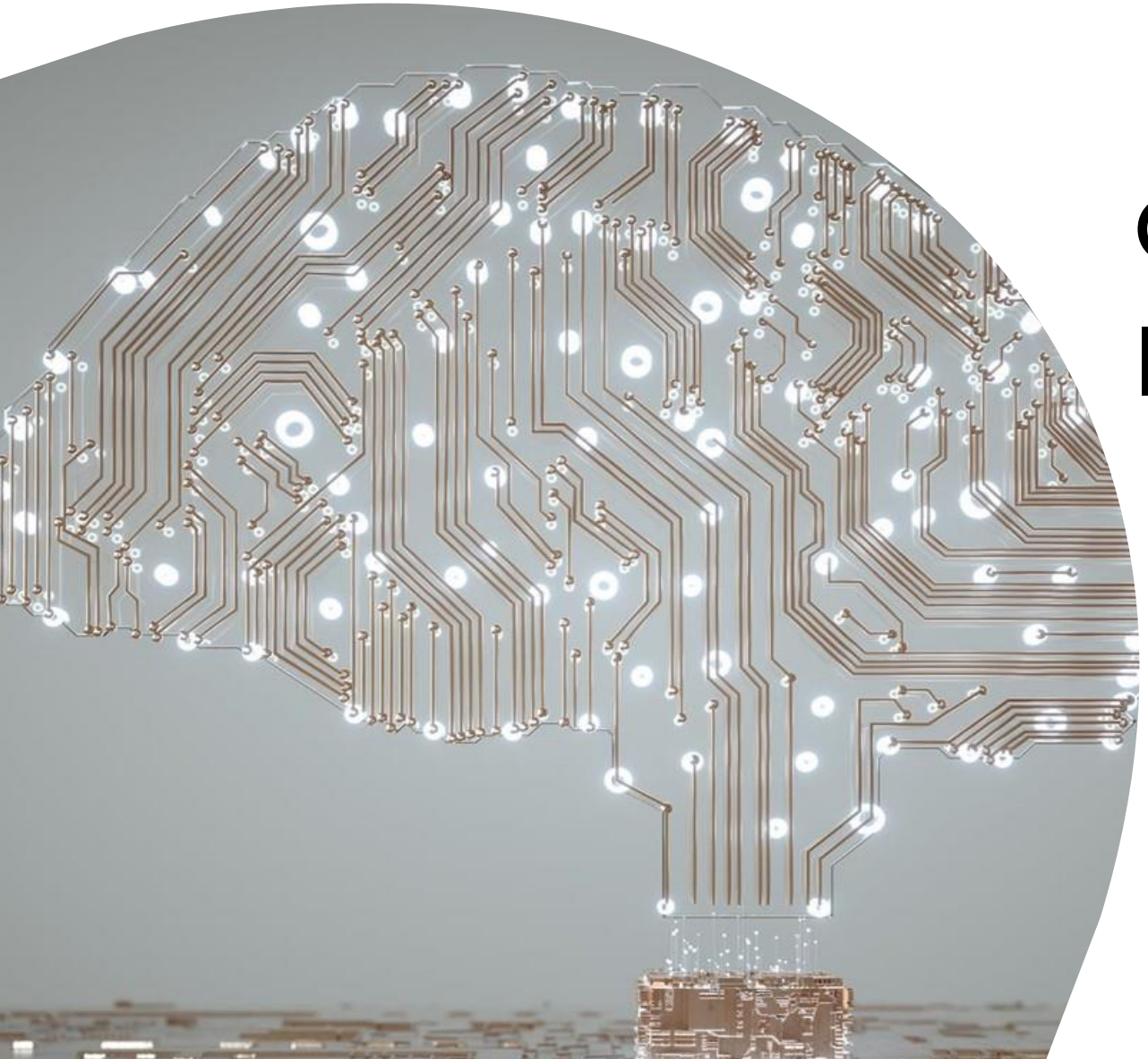
Propuesta 2: Extracción Avanzada y Búsqueda Semántica con `pdfplumber` y `spaCy`

Fortalezas:

- Utiliza `pdfplumber`, que es más robusto para la extracción de texto de PDFs, especialmente en documentos complejos.
- Incorpora `spaCy` para la extracción de palabras clave, lo que mejora la calidad de los metadatos extraídos.
- La propuesta menciona la posibilidad de realizar búsquedas semánticas, lo que es un gran avance respecto a la Propuesta 1.

Debilidades:

- Aunque mejora la extracción de metadatos, la implementación de la búsqueda semántica no está completamente desarrollada.
- La interfaz de usuario no está contemplada, lo que puede dificultar la interacción con el sistema para usuarios no técnicos.



Conclusión y perspectivas futuras

Conclusión y perspectivas futuras

- Optimización y automatización de procesos.

- Perspectivas Futuras
  - Automatización avanzada de tareas complejas.
  - Colaboración fluida entre agentes y humanos.
  - Mayor personalización de los agentes a distintos entornos.
  - Ética y privacidad como desafíos clave.
  - Nuevas oportunidades laborales en la supervisión y desarrollo de agentes.

- A pesar del gran avance, aún hay incertidumbre sobre la integración robusta y el despliegue de agentes en sistemas de producción, especialmente en términos de escalabilidad y fiabilidad a largo plazo.

¡Muchas gracias!

Info@decidesoluciones.es
www.decidesoluciones.es