

# Inteligencia Artificial

## Agentes

### Lecture 14



# Contenido

- Fundamentación Teórica
- Ejemplos Prácticos



# Referencias

- Artículos
- Exploring Large Language Model based Intelligent Agents: Definitions, Methods, and Prospects (Cheng et al., 2024).
- Este artículo ofrece una visión detallada de los agentes inteligentes basados en LLMs, abarcando sus definiciones, métodos de investigación y componentes fundamentales, así como sus aplicaciones en sistemas de múltiples agentes.
- The Rise and Potential of Large Language Model Based Agents: A Survey (Xi et al., 2023).
- Este trabajo realiza una encuesta exhaustiva sobre los agentes basados en LLMs, explorando su evolución, aplicaciones en diversos escenarios y el impacto potencial en la inteligencia artificial general.
- Cognitive Architectures for Language Agents (Sumers et al., 2023).
- Este artículo propone arquitecturas cognitivas para agentes lingüísticos, analizando cómo los LLMs pueden integrarse en sistemas más amplios para mejorar sus capacidades de razonamiento y acción.
- 2. Curso
  - Agents from HuggingFace - <https://huggingface.co/learn/agents-course>
- 3. Otros
  - <https://platform.openai.com/docs/guides/agents>
  - <https://github.com/openai/openai-agents-python>



# Contenido

- **Fundamentación Teórica**
- **Ejemplos Prácticos**



# Preguntas

- Entendiendo Agentes
  - Que es un Agente, y como trabaja?
  - Como los agentes toman desiciones usando Razonamiento y Planificación?
- Cual es el papel de los LLM en Agentes?
  - Los LLMs como el cerebro detrás de un agente
  - Como los LLMs estructuran conversaciones via Sistema de Mensajes
- Herramientas y acciones
  - Como los agentes usan herramientas externas para interactuar con el ambiente?
  - Como construir e integrar herramientas dentro de un agente?
- El workflow de un agente?
  - Pensar -> Actuar -> Observar

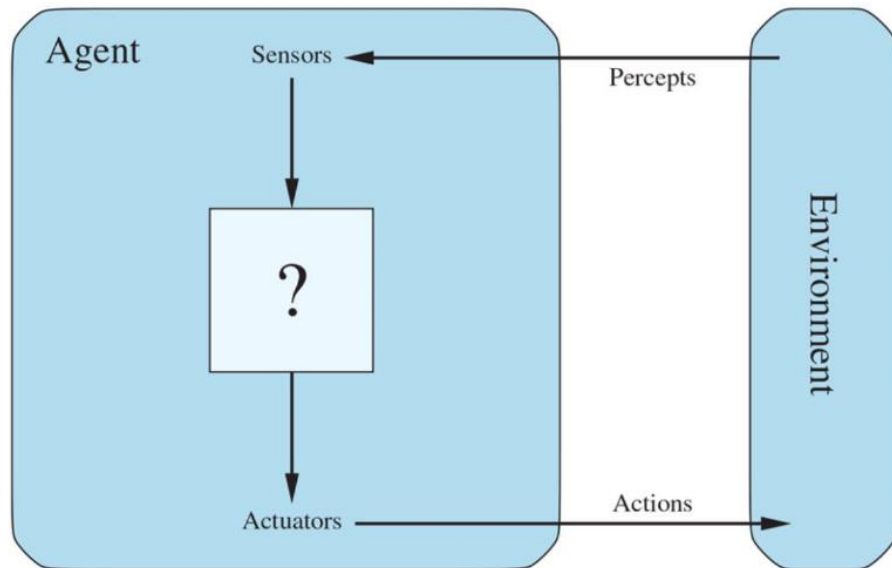
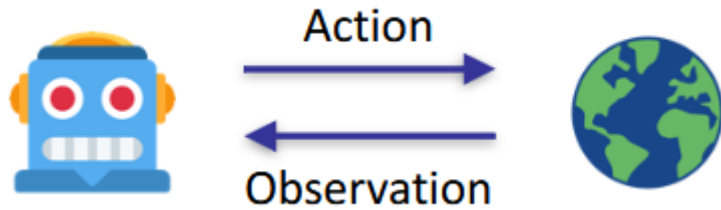


# Qué es un Agente? Un ejemplo

- Llamaremos el agente: Bob
- Imaginemos que Bob recibe un comando:
  - “Bob, Me gustaría un café por favor”
- Bob tendrá varios retos
  - Entendimiento natural del lenguaje
  - Y antes de cumplir la orden, Bob Razonará y Planificará hacia los siguientes pasos:
    - Ir a la cocina
    - Usar la cafetera
    - Prepara el café
    - Llevar el café
  - Una vez tiene el plan, el TIENE que actuar.
    - Para ejecutar el plan necesitará un conjunto de herramientas: energía, cafetera, insumos, etc (Ejecución)
- Finalmente, Bob llevará un café fresco



# Qué es un Agente?

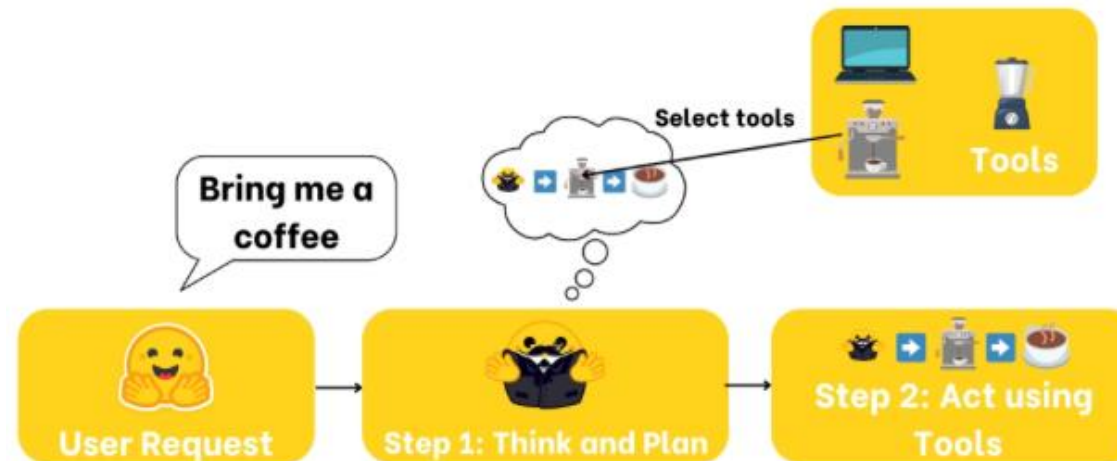


- Un agente es cualquier cosa que puede considerarse como algo que percibe su entorno a través de sensores y actúa sobre ese entorno mediante actuadores.  
-- Russell y Norvig, Inteligencia Artificial: Un Enfoque Moderno
- Un sistema inteligente que interactúa con algún “ambiente”
  - Ambientes físicos: robots, vehículos autónomos, ...
  - Ambientes digitales: Atari, Siri, AlphaGo, ...
  - Humanos como ambientes: chatbot



# Qué es un Agente?

- Un Agente es un Modelo AI capaz de:
  - Razonar
  - Planificar
  - Interactuar con su ambiente. (esta es la característica principal de un agente)





# Definición más formal de Agente

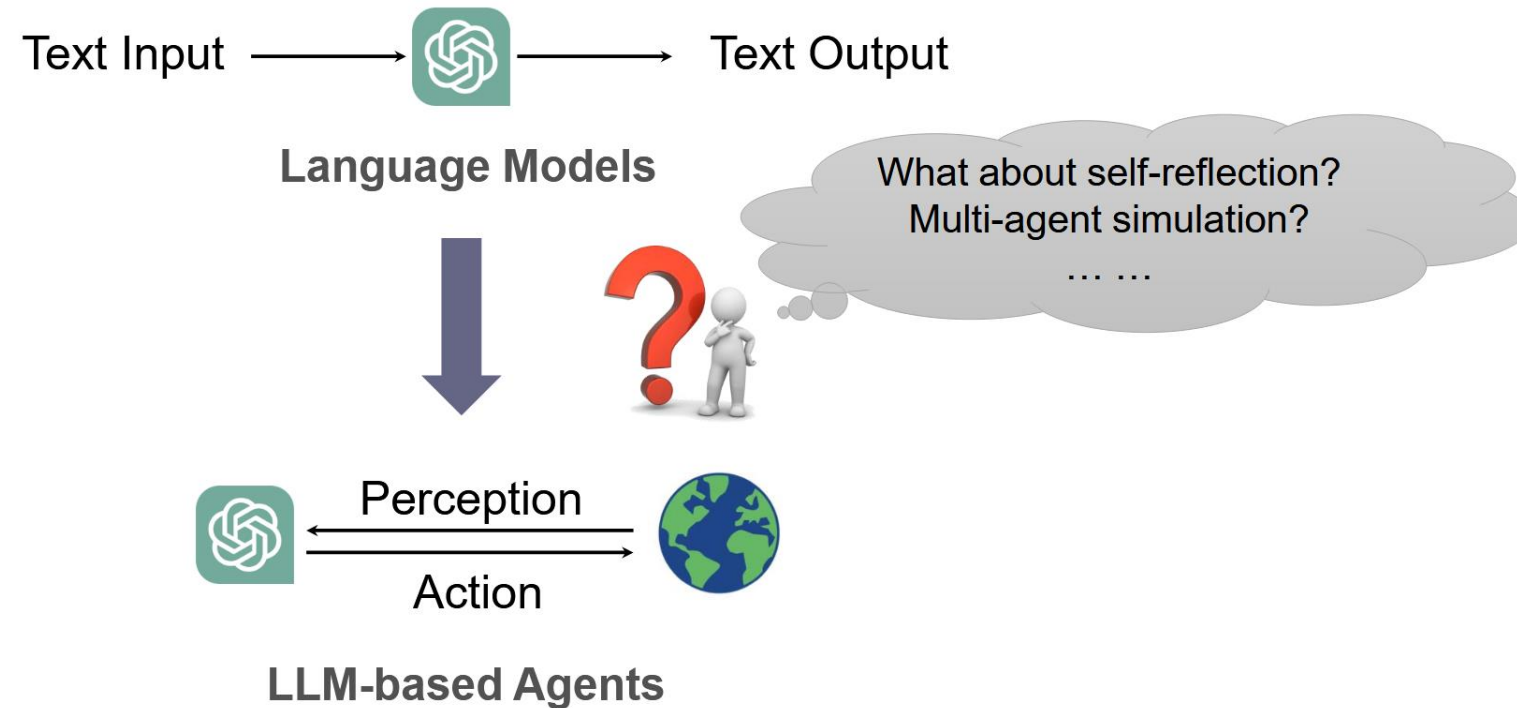
- *“An Agent is a system that leverages an AI model to interact with its environment in order to achieve a user-defined objective. It combines reasoning, planning, and the execution of actions (often via external tools) to fulfill tasks.”*

Compuesto de dos partes principales:

- **El cerebro (Modelo AI)**
  - Donde todo el ‘pensamiento’ o ‘razonamiento’ sucede
  - En general se utiliza un modelo de IA para el RAZONAMIENTO y PLANIFICACIÓN
  - Decide que ACCIONES debe realizar para cumplir la tarea
- **El cuerpo (Capacidades y Herramientas)**
  - Representa todo con lo que está equipado el agente
  - Las posibles acciones depende de con que esté equipado.



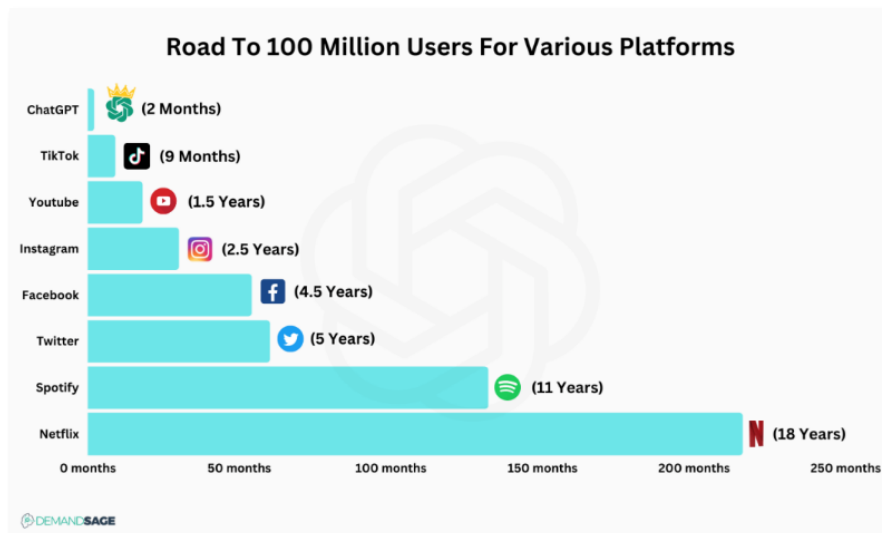
# Agentes modernos: LLM+env



# Diferencias



[https://www.reddit.com/r/ChatGPT/comments/16jvl4x/wait\\_actually\\_yes/](https://www.reddit.com/r/ChatGPT/comments/16jvl4x/wait_actually_yes/)

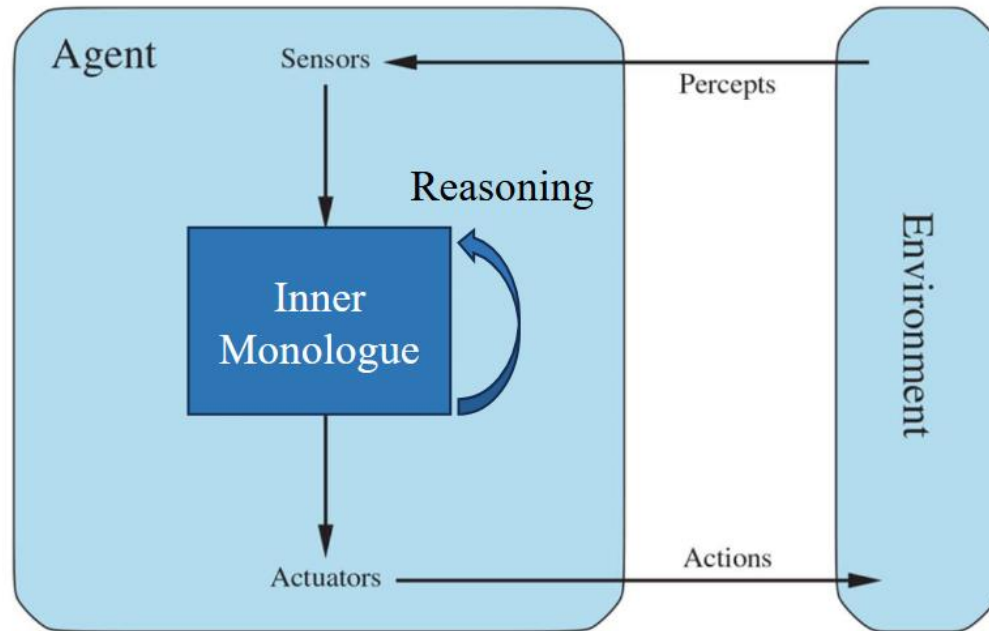


<https://www.demandsage.com/chatgpt-statistics/>

- Utilizan el lenguaje como medio para razonar y comunicarse.
- Pueden seguir instrucciones, aprender en contexto y personalizar salidas.
- Aplican razonamiento para mejorar sus acciones (inferencias, autorreflexión, replanificación, entre otros).



# Agentes modernos: LLM+env



- Es un nuevo tipo de acción (distinto a las acciones en entornos externos).
- Ocurre en un entorno interno, como un monólogo interno.
- La autorreflexión es una acción de razonamiento 'meta' (razonamiento sobre el propio razonamiento), similar a funciones metacognitivas.
- El razonamiento busca mejorar la acción, mediante inferencias sobre el entorno, retrospección, etc.
- Los espacios de percepción y acción externa se amplían significativamente gracias al uso del lenguaje y la percepción multimodal.



# ¿Qué es un Agente en Inteligencia Artificial?

- Un **agente** en IA es un sistema autónomo que percibe su entorno a través de sensores, procesa información y toma decisiones para realizar acciones que optimicen un objetivo. En el contexto de la IA, los agentes pueden ser reactivos o deliberativos.
- Los agentes inteligentes basados en modelos de lenguaje de gran escala (LLMs) representan un avance significativo en el camino hacia la inteligencia artificial general (AGI).
- Estos agentes utilizan el lenguaje natural como interfaz y exhiben capacidades de generalización robustas en diversas aplicaciones, desde asistentes de tareas hasta sistemas de múltiples agentes.



# Que tipos de tareas puede hacer un agente

- Cualquier tarea que pueda implementar vía Herramientas para completar las Acciones.
- Por ejemplo, un Agente como asistente personal en mi computador (como Siri), el cual le pueda indicar cosas como:
  - “enviar un email a mi jefe reportando que me demoro en entrar a la reunión de hoy”
  - Debo contar con algun código para mandar email, algo así:

```
def send_email(recipient, message):
```

```
...
```

```
...
```

```
send_email('jefearea@eafit.edu.co', 'me demoro 15 minutos en entrar a la reunión')
```

Parte de la clave del éxito de los agentes, es la disponibilidad de herramientas, desde genéricas (web\_search), hasta muy específicas.



# Ejemplos

- Asistentes virtuales personales
  - Siri, Alexa, etc, trabajan como agentes en nombre de usuarios usando un ambiente digital.
  - Recibe las consultas. Analiza el contexto, recupera info de bases de datos y provee una respuesta o inicia una acción (colocar una alarma, enviar un mensaje, controlar algo IoT, colocar una canción, etc)
- Chatbots de Atención a cliente
  - Interacción en lenguaje natural
  - Responde preguntas (de un FAQ)
  - Completa transacción (actualización de un plan, asignación de una cita médica)



# Question Answering

Q: what is  $1 + 2$ ?



A: 3

Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for \$2 per egg. How much does she make every day?



Requires reasoning

Q: who is the latest UK PM?



Requires knowledge

Q: what is the prime factorization of 34324329?



Requires computation





# Code augmentation

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is  $0+1=1$ . The fourth number is  $1+1=2$ . The fifth number is  $1+2=3$ . The sixth number is  $2+3=5$ . The seventh number is  $3+5=8$ . The eighth number is  $5+8=13$ .  
..... (Skip 1000 tokens)  
The 50th number is 32,432,268,459.

CoT

32,432,268,459



```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
For i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
    fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

PoT



12,586,269,025



Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks



# RAG for knowledge

- Answer knowledge-intensive questions with
  - Extra corpora
  - A retriever (e.g., BM25, DPR, etc.)
- What if there's no corpora? (e.g. who's the latest PM?)

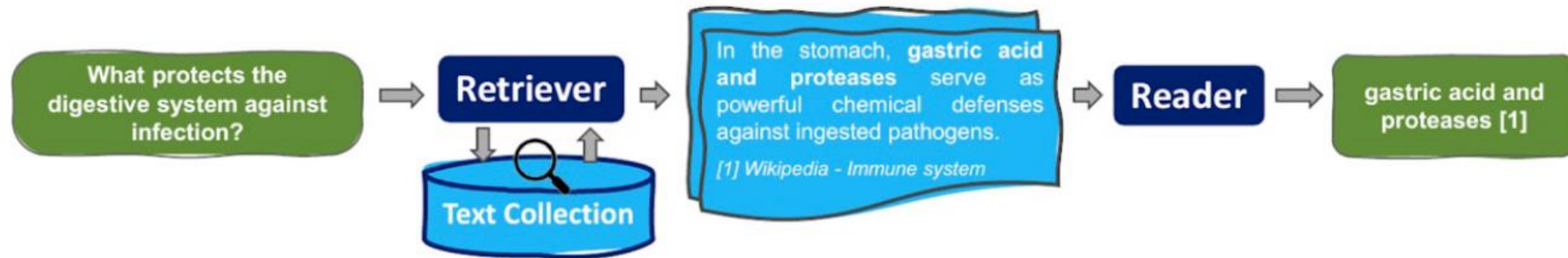


Image: <http://ai.stanford.edu/blog/retrieval-based-NLP/>



# Tool use

- Special tokens to invoke tool calls for
  - Search engine, calculator, etc.
  - Task-specific models (translation)
  - APIs
- Unnatural format requires task/tool-specific fine-tuning
- Multiple tool calls?

## A weather task:

how hot will it get in NYC today? |*weather* lookup re-  
gion=NYC |*result* precipitation chance: 10, high temp:  
20c, low-temp: 12c |*output* today's high will be 20C

TALM: Tool Augmented Language Models.

Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

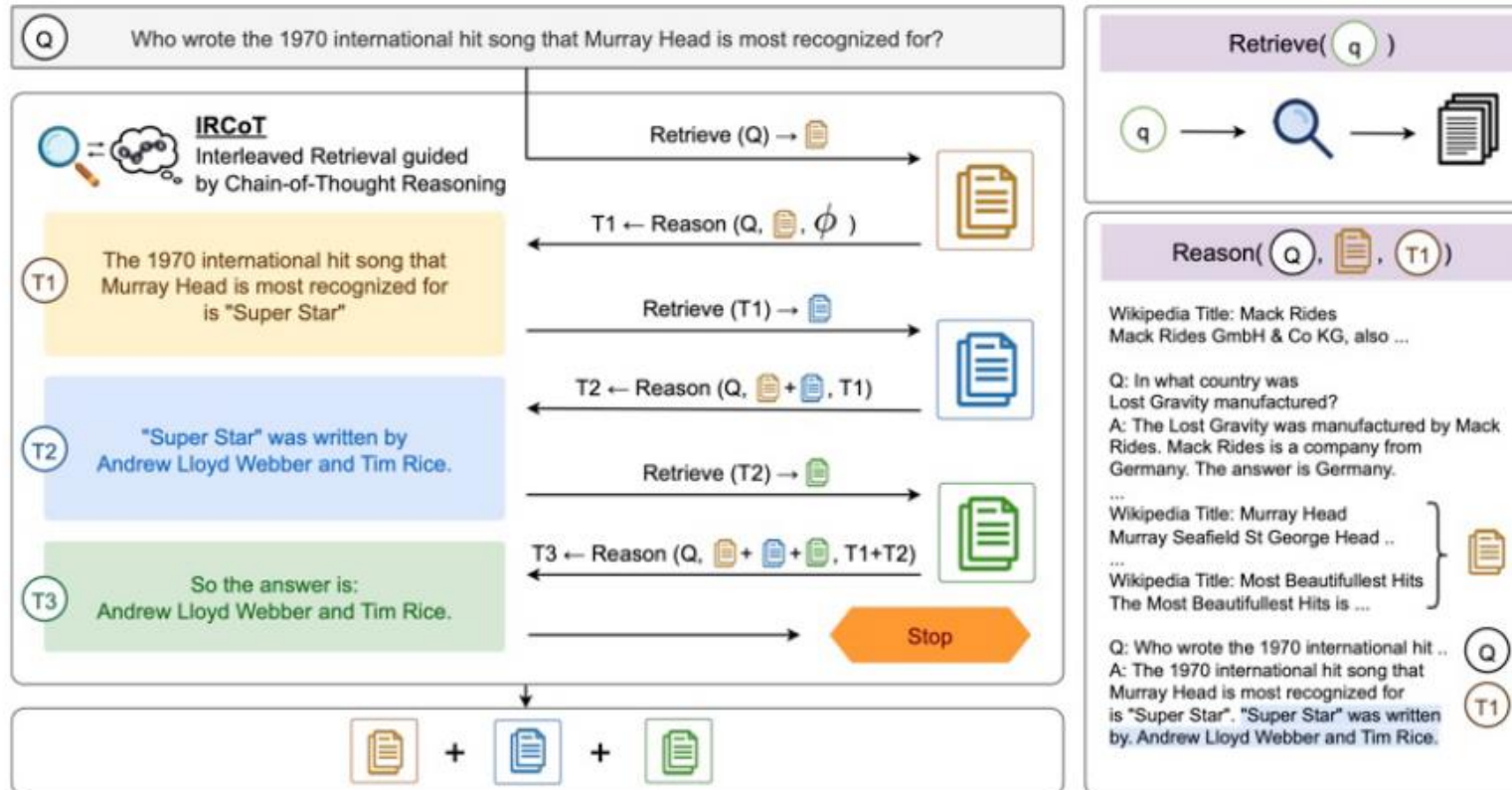
The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Toolformer: Language Models Can Teach Themselves to Use Tools

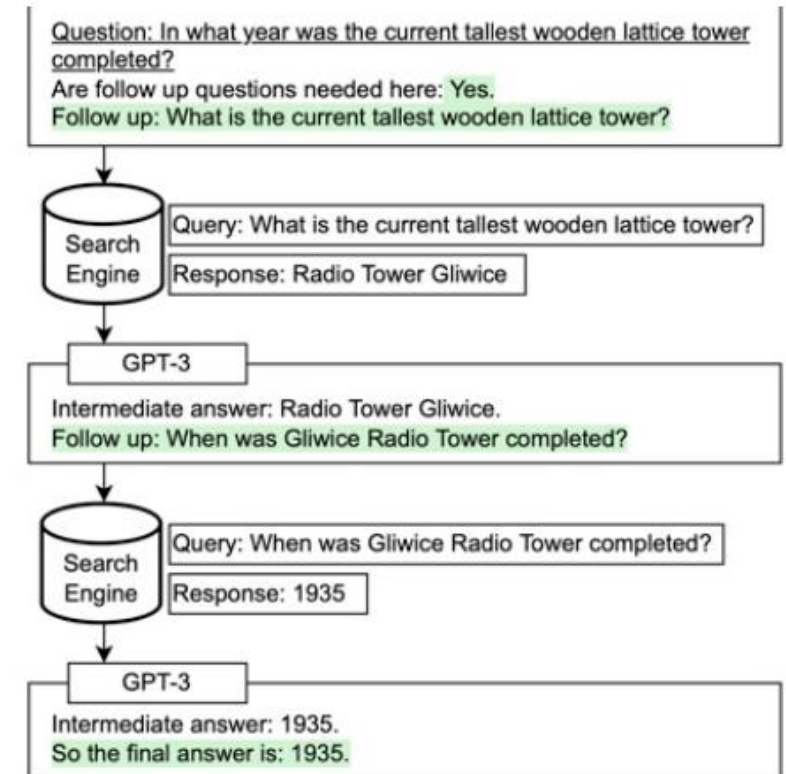




# Knowledge + reasoning



Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions



Measuring and Narrowing the Compositionality Gap in Language Models.

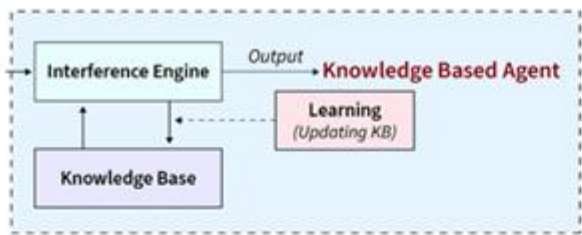


# Desarrollos previos

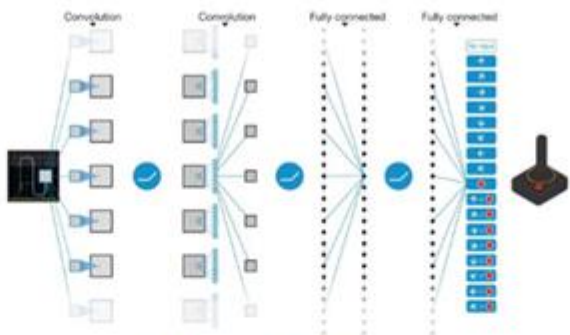
- **Agentes simbólicos:** Basados en reglas lógicas, utilizados en sistemas expertos, pero con dificultades para adaptarse a entornos inciertos.
- **Agentes basados en aprendizaje por refuerzo:** Mejoran su desempeño mediante experiencia, como en los sistemas AlphaGo.



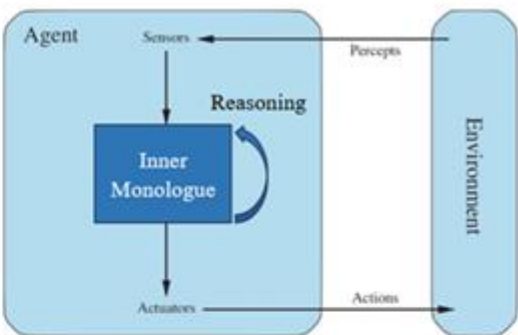
# Avances de los agentes



**Logical Agent**



**Neural Agent**



**Language Agent**

<b>Expressiveness</b>	Low bounded by the logical language	Medium anything a (small-ish) NN can encode	High almost anything, esp. verbalizable parts of the world
<b>Reasoning</b>	Logical inferences sound, explicit, rigid	Parametric inferences stochastic, implicit, rigid	Language-based inferences fuzzy, semi-explicit, flexible
<b>Adaptivity</b>	Low bounded by knowledge curation	Medium data-driven but sample inefficient	High strong prior from LLMs + language use

Image sources: <https://www.scaler.com/topics/artificial-intelligence-tutorial/knowledge-based-agent/>,  
Mnih et al., “Human-level control through deep reinforcement learning.” Nature (2015)



# Tipos de Agentes en IA

1. **Agentes Reactivos:** No tienen memoria ni modelan el entorno, toman decisiones basadas en reglas predefinidas (Ejemplo: termostatos inteligentes).
2. **Agentes Basados en Modelos:** Poseen una representación interna del entorno y pueden razonar sobre estados futuros (Ejemplo: asistentes personales).
3. **Agentes con Planificación:** Toman decisiones considerando una serie de pasos y objetivos a largo plazo.
4. **Agentes Basados en Aprendizaje:** Aprenden de la experiencia y mejoran su desempeño con el tiempo.
5. **Agentes Autónomos Inteligentes:** Pueden adaptarse a entornos complejos y evolucionar sus estrategias sin intervención humana.
6. **Agentes Basados en LLM:** Utilizan modelos de lenguaje como núcleo cognitivo, combinando razonamiento y generación de texto.



# ¿Qué es un Agente Basado en LLM?

- Un agente basado en LLM (Large Language Model) es una extensión avanzada de los agentes tradicionales, donde un modelo de lenguaje grande (LLM) se usa como el núcleo de razonamiento y procesamiento del agente.
- Estos agentes pueden interpretar texto, generar respuestas coherentes, planificar tareas y ejecutar acciones basadas en el conocimiento almacenado en sus modelos.





# Diferencia entre un chatbot y un agente autónomo

Característica	Chatbot Tradicional	Agente Basado en LLM
Memoria	Limitada o inexistente	Persistente y evolutiva
Razonamiento	Responde con patrones predefinidos	Puede planificar y descomponer problemas
Adaptabilidad	Limitada	Se ajusta al contexto dinámico
Integración con herramientas	Básica	Se conecta con APIs, bases de datos, búsqueda web



# Arquitectura – Agentes LLM

Tres módulos principales:

- 1.Cerebro:** Un LLM que actúa como el núcleo del agente, encargándose del almacenamiento de conocimiento, memoria, razonamiento y planificación.
- 2.Percepción:** Entrada de datos en diversos formatos, incluyendo texto, imágenes, sonido y sensores externos.
- 3.Acción:** La capacidad de interactuar con el entorno mediante generación de texto, herramientas externas o manipulación de objetos en entornos físicos o virtuales.



# Arquitectura extendida – Agentes LLM

- 1. Percepción:** Captan datos del entorno a través de APIs, sensores o entradas textuales.
- 2. Memoria:** Utilizan bases de datos vectoriales como FAISS o Pinecone para retención de información.
- 3. Planificación:** Descomponen problemas en subtarefas utilizando técnicas como Chain of Thought (CoT) o Tree of Thought (ToT).
- 4. Ejecución de Acciones:** Interactúan con el entorno a través de herramientas y generación de texto.
- 5. Aprendizaje y Adaptación:** Utilizan memoria y realimentación para mejorar decisiones futuras.



# Sistemas de Agentes Individuales vs Sistemas Multiagente

- **Sistemas de Agentes Individuales:** Un solo agente con autonomía para ejecutar tareas complejas, útil en aplicaciones como asistentes virtuales y generación de código.
  - Capaces de ejecutar tareas específicas, como generación de código, automatización de procesos y asistencia en investigación científica.
- **Sistemas Multiagente:** Redes de agentes que colaboran entre sí, aplicadas en simulaciones económicas, juegos y sistemas de colaboración cognitiva.
  - Interacción entre múltiples agentes para colaboración o competencia en entornos simulados y del mundo real.
- **Cooperación humano-agente:** Modelos de colaboración en donde los agentes pueden actuar como asistentes o socios estratégicos.

En los sistemas multiagente, se enfatiza la comunicación eficiente mediante técnicas como el paso de mensajes, la coordinación de roles y el aprendizaje colectivo.



# Retos

- **Evaluación del desempeño:** No existen métricas estándar para medir la efectividad de estos agentes en escenarios dinámicos y complejos.
- **Seguridad y confianza:** Los LLMs pueden generar contenido incorrecto o sesgado, lo que plantea riesgos en aplicaciones críticas.
- **Escalabilidad:** Integrar múltiples agentes en un ecosistema requiere nuevas arquitecturas para coordinación eficiente.
- **Alineación con valores humanos:** Se deben desarrollar técnicas que permitan garantizar que las decisiones de los agentes estén alineadas con normas éticas y sociales.



# Evaluación del Desempeño de los Agentes

Para medir la efectividad de estos agentes, se utilizan benchmarks y conjuntos de datos específicos. Entre los más populares están:

- **BIG-Bench:** Evaluación de habilidades cognitivas de los modelos de lenguaje.
- **AgentBench:** Conjunto de pruebas para agentes autónomos con tareas específicas.

Una plataforma de evaluación interesante es la de LLamaIndex  
LLamaTrace –

*Arize Phoenix hace trazabilidad y evaluación de prompts hacia LLM*

<https://llamatrace.com>



# Marcos y Herramientas para Agentes con LLM

- **LangChain:** Framework para construir agentes basados en LLMs.
- **LlamaIndex:** Integración con fuentes de datos estructurados y no estructurados.
- **AutoGPT y BabyAGI:** Agentes autónomos con objetivos predefinidos.
- **Integración con APIs externas:** OpenAI API, Pinecone, FAISS para memoria.



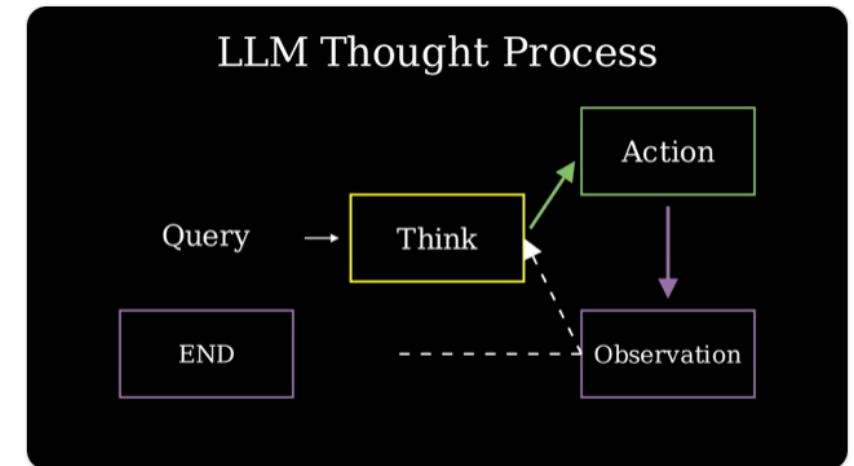
# Los agentes de IA a través del ciclo de pensamiento-acción-observación

Los Agentes IA son sistemas que pueden

- Razonar -> planear -> interactuar con su ambiente.

Pero cual es el ciclo completo?

- 1.Razonar:** La parte de LLM del agente decide cuál debe ser el siguiente paso.
- 2.Acción:** El agente realiza una acción llamando a las herramientas con los argumentos asociados.
- 3.Observación:** El modelo refleja la respuesta de la herramienta.





# Ciclo de pensamiento-acción-observación

En muchos frameworks, las REGLAS y las GUIDELINES son embebidas directamente en el sistema de prompt, ejemplo:

```
system_message="""You are an AI assistant designed to help users efficiently and accurately. Your
primary goal is to provide helpful, precise, and clear responses.

You have access to the following tools:
Tool Name: calculator, Description: Multiply two integers., Arguments: a: int, b: int, Outputs: int

You should think step by step in order to fulfill the objective with a reasoning divided in
Thought/Action/Observation that can repeat multiple times if needed.

You should first reflect with 'Thought: {your_thoughts}' on the current situation,
then (if necessary ), call a tool with the proper JSON formatting 'Action: {JSON_BLOB}', or your print
your final answer starting with the prefix 'Final Answer:'
"""
```



# Ejemplo

- Bob, cual es la temperatura en Medellin?
  - Bob contestará esta pregunta usando weather API tools.
  - Como se desarrolla:
  - **Pensar:**
    - Razonamiento interno: se recibe la consulta. Bob podría interpretar:
      - “el usuario necesita la info de clima de medellin. Tengo acceso a una herramienta que recupera datos de clima, debo llamar esa API para tener datos actualizados”.
  - **Acción:**
    - Usar la Herramienta (tool)
    - Basado en el razonamiento, Bob sabe acerca de la herramienta get\_weather, prepara un comando JSON para llamar la API. Ejemplo:

```
{  "action": "get_weather",  "action_input": {    "location": "Medellín"  }}
```
  - **Observación**, feedback del ambiente
    - Respuesta de la API
    - “Current weather in Medellin: partly cloudy, 15°C, 60% humidity.”
  - **Actualizar** el razonamiento
    - Recibida la observación, Bob actualiza su razonamiento interno
    - “ahora tengo el clima de Medellin, puedo preparar una respuesta al usuario”
  - **Acción final**
    - Bob genera una respuesta final
    - Razonamiento: “tengo los datos del clima ahora, el clima actual de medellín es ,,,,”
    - Respuesta final: “el clima actual de medellin es parcialmente nublado con una temperatura de 15 grados y una humedad relativa del 60 porciento” (observe que hubo traducción de inglés a español)



# Razonamiento Interno de un Agente IA

- Un agente usa la capacidad de un LLM para analizar información cuando es presentado en su prompt
- El razonamiento de un agente es responsable de acceder observaciones actuales y decidir cual será la próxima acción
- El agente puede dividir tareas complejas en más pequeñas, pasos, reflejando su experiencia pasada y continuamente ajustando el plan con nueva info



# Razonamiento Interno de un Agente IA

Tipos de razonamiento mediante prompt tools:

Type of Thought	Example
Planning	"I need to break this task into three steps: 1) gather data, 2) analyze trends, 3) generate report"
Analysis	"Based on the error message, the issue appears to be with the database connection parameters"
Decision Making	"Given the user's budget constraints, I should recommend the mid-tier option"
Problem Solving	"To optimize this code, I should first profile it to identify bottlenecks"
Memory Integration	"The user mentioned their preference for Python earlier, so I'll provide examples in Python"
Self-Reflection	"My last approach didn't work well, I should try a different strategy"
Goal Setting	"To complete this task, I need to first establish the acceptance criteria"
Prioritization	"The security vulnerability should be addressed before adding new features"



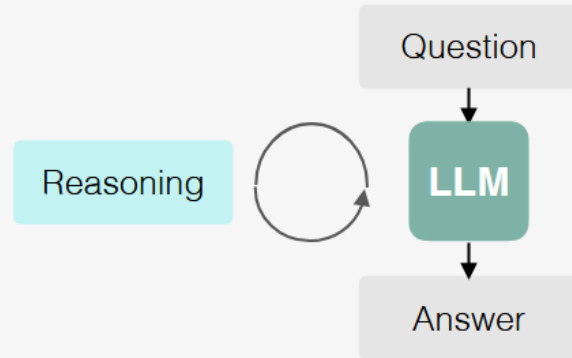
# Acercamiento Re-Act

- Re-Act es la concatenación de “Reasoning” (think) con “Acting” (Act).
- ReAct es una técnica simple de prompting que adiciona “Let’s think step by step” antes de permitir al LLM decodificar los próximos tokens.
  - Esto conduce a generar un plan, más que una solución final.

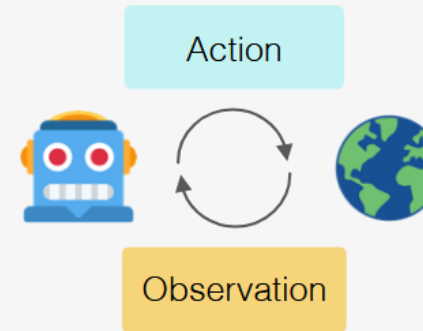


# ReAct

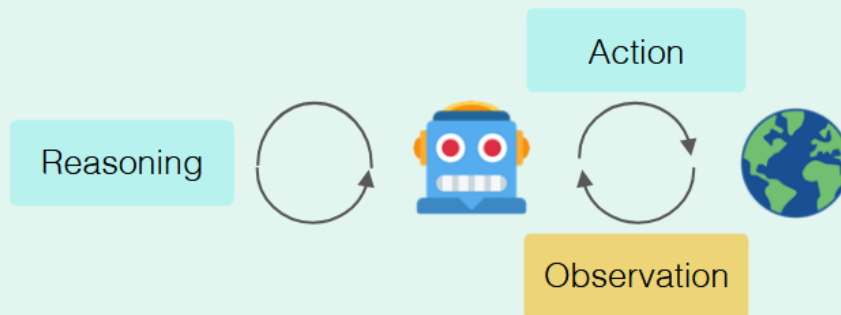
**Reasoning** (update internal belief)



**Acting** (obtain external feedback)



**ReAct**: a new paradigm of agents that **reason and act**



- **Synergy** of reasoning and acting
- **Simple** and intuitive to use
- **General** across domains



## (a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

## (c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

## (b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓

## (d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓





# Acciones

- Las acciones son los pasos concretos del Agente para interactuar con el ambiente, ejemplos:
  - Navegar la web
  - Controlar un dispositivo
  - Acceder un API





# Contenido

1. Fundamentación Teórica

**2. Ejemplos Prácticos**

Lecture08/notebooks



# Ejemplo 1 - Creación de un Agente LLM en python

- Notebook class14.ipynb
- Utilizar langchain, openai y faiss
- Agente de búsqueda en Wikipedia
- Se utiliza la API KEY de OpenAI, tener una cuenta.
- Se utiliza la API KEY de SerpAPI, tener una cuenta.
  - <https://serpapi.com/>
  - En la versión gratuita permite 100 consultas al mes.



# Ejemplo 2 - Agente con Razonamiento y Planificación

- Notebook class14.ipynb
- Utilizar langchain, openai y faiss
- Agente de búsqueda en Wikipedia
- Se utiliza la API KEY de OpenAI, tener una cuenta.
- Se utiliza la API KEY de SerpAPI, tener una cuenta.
  - <https://serpapi.com/>
  - En la versión gratuita permite 100 consultas al mes.



# Framework LlamaIndex



# Introducción a LlamaIndex

- LlamaIndex es un toolkit completo para crear agentes habilitados por LLM sobre tus datos usando índices y workflows.
- Bloques
  - **Componentes:**
    - prompts, modelos y bases de datos
    - Conecta LlamaIndex con otras herramientas y librerías
  - **Agentes**
    - Componentes autónomos que pueden usar herramientas y tomar decisiones.
    - Coordina herramientas para lograr metas complejas
  - **Tools**
    - Provee capacidades específicas como búsquedas, cálculos, acceso a servicios externos.
    - Habilita agentes a ejecutar tareas.
  - **Workflows**
    - Procesos paso a paso que procesa lógica conjunta.



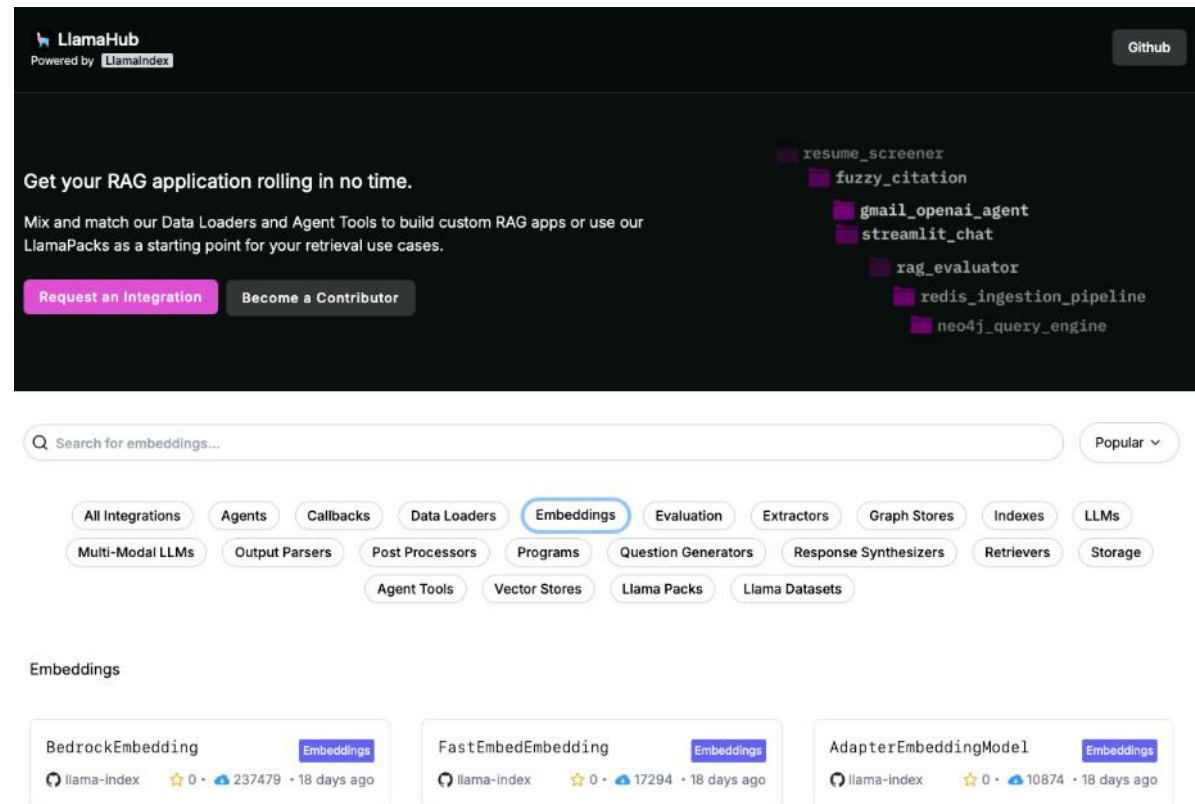
# Cual es el diferencial de LlamaIndex

- Sistema de Workflow
- Parseo avanzado de documentos con LlamaParse
- Muchos componentes listos para usar
- LlamaHub



# LlamaHub

- Registro de cientos de Integraciones, Agentes y Herramientas que puede usar dentro de LlamaIndex



# Installation

- `pip install llama-index-{component-type}-{framework-name}`
- `pip install llama-index-llms-huggingface-api llama-index-embeddings-huggingface`

```
from llama_index.llms.huggingface_api import HuggingFaceInferenceAPI
```

```
llm = HuggingFaceInferenceAPI(  
    model_name="Qwen/Qwen2.5-Coder-32B-Instruct",  
    temperature=0.7,  
    max_tokens=100,  
    token="hf_xxx",  
)
```

```
llm.complete("Hello, how are you?")  
# I am good, how can I help you today?
```

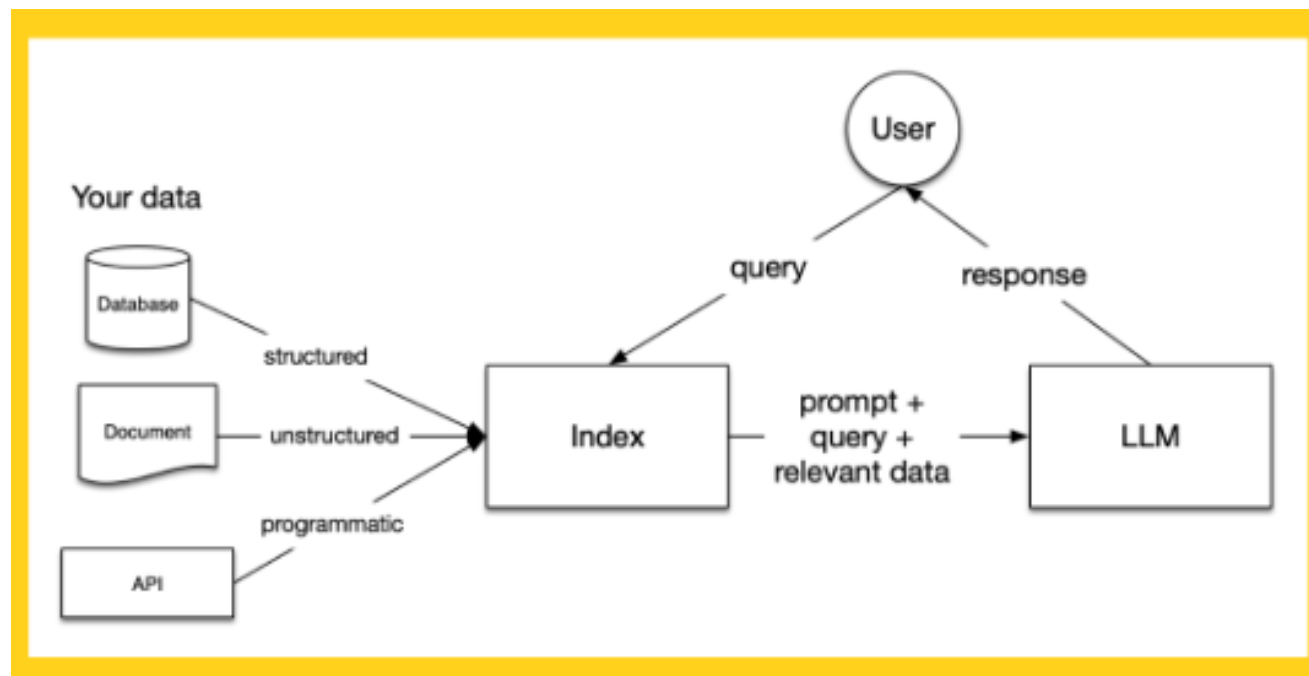
Ejemplo 3 en: `class14.ipynb`





# Componentes en LLamaIndex

- Recuerda: Bob necesita entender los requerimientos y
  - Preparar, encontrar y usar información relevante para completar la tarea
- Un componente muy importante es **QueryEngine**
  - En Conjunto con RAG potencia mucho los agentes.



# Ej: planificar una cena

- 1.Ud pregunta a Bob que le ayude a planificar la cena
- 2.Bob necesita chequear su calendario, preferencias de dieta, menus pasados exitosos.
- 3.QueryEngine ayuda a Bob a encontrar información y usarla para planear la cena.

Como desarrollar este agente?



# Crear el RAG

- 1.Loading (cargar datos)
- 2.Indexing (indezar)
- 3.Storing (almacenar los embeddings)
- 4.Querying (consultar)
- 5.Evaluation (evaluar)

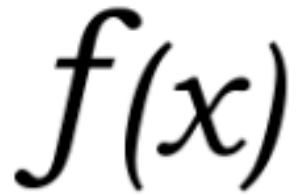
Ejecutar notebook: *components.ipynb*

- Ejemplo adaptado del curso de Agentes de HuggingFace

<https://huggingface.co/learn/agents-course/unit0/introduction>



# Tools en LlamaIndex



## FunctionTool

Convert any Python function into a tool that an agent can use.



## Search Engine

A tool that lets agents use query engines.



## Toolspecs

Set of tools created by the community.



## Utility Tools

Special tools that help handle large amounts of data from other tools.



# Ejemplos de Tools

Ejecutar notebook: *tools.ipynb*

- Ejemplo adaptado del curso de Agentes de HuggingFace

<https://huggingface.co/learn/agents-course/unit0/introduction>



# Otros ejemplos en `Lecture14/notebooks`



# Customer Service

Ejecutar notebook: *customerservice.ipynb*

- Ejemplo adaptado de OpenAI

<https://platform.openai.com/docs/guides/agents>

<https://github.com/openai/openai-agents-python>

[https://github.com/openai/openai-agents-python/tree/main/examples/customer\\_service/](https://github.com/openai/openai-agents-python/tree/main/examples/customer_service/)

