DIGISOFT

Entendiendo las herramientas y su metodo de implementacion

Miguel Cotrina

Miguel Cotrina

- **Perfil Académico**
  - Ingeniero de software por la Universidad Tecnológica del Perú.
  - Maestría en ciencia de datos por la Universidad Ricardo Palma.
- **Perfil Profesional**
  - Arquitecto de datos región latam en Indra
  - Consultor e Instructor de Big Data, cloud, IA e IA Gen en empresas privadas y públicas

Agenda

- Qué son las herramientas y cuál es su función en los agentes
- Herramienta basada en @tool
  - Creación con @tool + anotaciones
  - Personalizar @tool el nombre y argumentos json mediante el decorador
- Herramienta basada en runnables
- Herramienta basada en la subclase BaseTool
- Laboratorio practico

¿Qué son las herramientas y cuál es su función en los agentes?

Miguel Cotrina

DIGISOFT

Definición de Herramientas

En LangChain, las herramientas son funciones encapsuladas que los agentes pueden invocar para realizar tareas específicas. Actúan como interfaces entre el agente y el entorno externo, permitiéndole acceder a APIs, bases de datos, sistemas externos, o incluso realizar cálculos específicos.

Función de las Herramientas en los Agentes

Los agentes de LangChain utilizan herramientas para mejorar sus capacidades, delegando tareas especializadas a estas funciones. Algunas aplicaciones incluyen:

- Consultar bases de datos o APIs externas.
- Ejecutar cálculos matemáticos.
- Llamar a servicios como Google Search, Weather APIs, etc.
- Procesar documentos y extraer información relevante.

Categorías de Herramientas en LangChain

Existen múltiples maneras de definir herramientas en LangChain. En esta clase abordaremos tres enfoques principales:

- Herramientas con @tool (decoradores y anotaciones).
- Herramientas basadas en Runnable (pipeline flexible y reutilizable).
- Herramientas basadas en la subclase BaseTool (máxima personalización).

Herramienta basada en @tool

Miguel Cotrina

Creación con @tool + Anotaciones

El decorador @tool permite definir funciones como herramientas de manera sencilla. Se usa comúnmente con anotaciones de tipo para que LangChain pueda interpretar correctamente los parámetros y devolver respuestas estructuradas.

```python
@tool
def multiply(
    a: Annotated[int, "Primero operador de multiplicacion"],
    b: Annotated[int, "Segundo operador de multiplicacion"],) -> int:
    """Multiplicar dos numeros"""
    return a * b
```

Este enfoque facilita la integración rápida de funciones personalizadas sin necesidad de definir clases.

Personalizar @tool: Nombre y Argumentos JSON mediante el Decorador

Podemos personalizar nuestras herramientas usando parámetros en el decorador @tool, especificando el nombre y los argumentos en formato JSON.

```python
class SubInput(BaseModel):
    a: int = Field(description="Primer numero")
    b: int = Field(description="Segundo numero")

@tool("sub-tool", args_schema=SubInput, return_direct=True)
def sub(a: int, b: int) -> int:
    """resta dos numeros"""
    return a - b
```

Herramienta basada en Runnable

Miguel Cotrina

¿Qué es Runnable y por qué usarlo?

El framework Runnable en LangChain permite construir pipelines reutilizables de procesamiento de datos. A diferencia de @tool, que define funciones estáticas, Runnable permite una mayor flexibilidad y composición.

```
binarizador_tool = chain.as_tool(
    name="Binarizador", description="Convierte cualquier numero en binario",
    args_schema=BinarizerInput
)
```

Ventajas de Runnable:

- Permite componer herramientas en secuencias de procesamiento.
- Se integra con flujos de trabajo asíncronos.
- Es altamente modular y reutilizable.

Herramienta basada en la subclase BaseTool

Miguel Cotrina

¿Por qué usar BaseTool?

La subclase BaseTool permite una personalización total de la herramienta, ideal cuando se necesita controlar el procesamiento interno, manejar estados o definir métodos auxiliares.

Comparación de enfoques

Método	Ventajas	Desventajas
`@tool`	Fácil de usar, rápido de implementar	Menos flexible
`Runnable`	Modular, encadenamiento de procesos	Puede ser más complejo
`BaseTool`	Máximo control y personalización	Mayor complejidad

Conclusion

Las herramientas en LangChain permiten extender la capacidad de los agentes, integrando funciones externas de manera eficiente. Dependiendo del nivel de personalización requerido, se pueden implementar usando @tool, Runnable o BaseTool. Comprender sus diferencias y casos de uso ayudará a construir soluciones más flexibles y escalables

Laboratorio aplicado

Codigo compartido

- Despliegue de ejemplo de herramientas
  - @tool
  - Runnables
  - BaseTool

Enlace de materiales:

https://github.com/macespinoza/programa7genai/tree/main/Clase%2005

Tarea actividad 05

Actividad 01: Desplegar ejemplo de las 3 diferentes tipos de implementación de herramientas

Miguel Cotrina

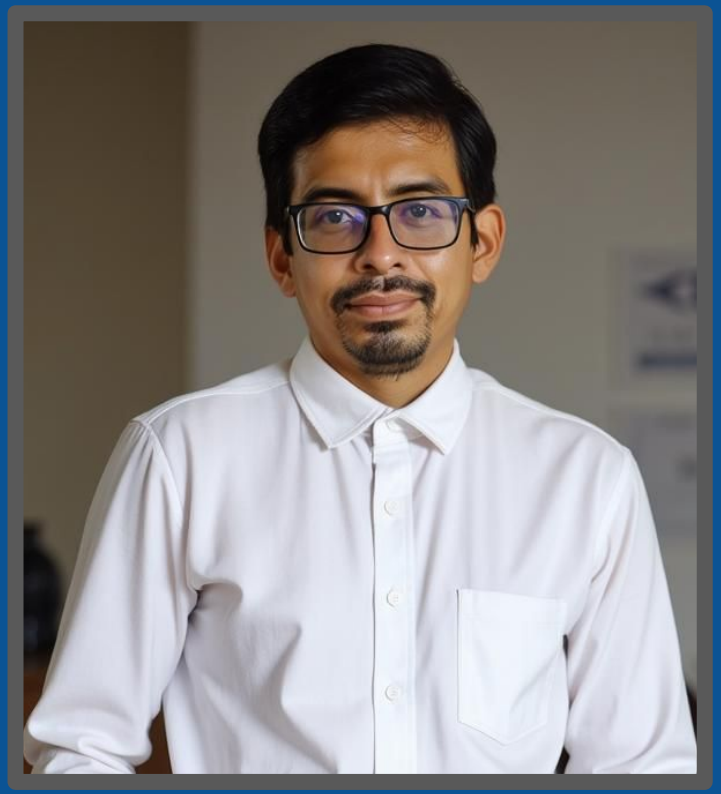
Agradecimiento y preguntas

Muchas gracias a todas las personas que están interesadas en aprender sobre estas nuevas tecnologías, el camino comienza pero el destino aún es desconocido.

Todas sus preguntas consultas o feedback son bienvenidos y lo pueden dejar en los comentarios del video de cada clase

Redes sociales:

- https://www.linkedin.com/in/mcotrina/
- https://www.youtube.com/@macespinozaonline
- https://github.com/macespinoza/



Programa de Introducción a la IA Generativa con Modelos de Gran Tamaño de 7 clases

Miguel Cotrina