Summit País Digital
Hackatón by EY
colabora Microsoft

Introducción a LangChain

Septiembre 2023
Sesión 4

The better the question. The better the answer.
The better the world works.

Calendario de Capacitaciones

El lenguaje de programación base para todas las capacitaciones y esta hackatón es "Python". El calendario de las capacitaciones es el siguiente, este será comunicado a los participantes de la hackatón y puede estar sujeto a cambios por situaciones de fuerza mayor:

- ~~Sesión 1: martes 12 de septiembre de 19:00 a 20:15 horas de Chile - "Design Thinking"~~
- ~~Sesión 2: miércoles 13 de septiembre de 19:00 a 20:15 horas de Chile - "LLM & Prompting"~~
- ~~Sesión 3: jueves 14 de septiembre de 19:00 a 20:15 horas de Chile - "APIS de OpenAI"~~
- Sesión 4: martes 26 de septiembre de 19:00 a 20:15 horas de Chile - "Langchain"
- Sesión 5: miércoles 27 de septiembre de 19:00 a 20:15 horas de Chile - "Casos de Uso / Wireframing"
- IMPORTANTE Sesión 6 Obligatoria (asistencia por lo menos de dos participantes por equipo): jueves 28 de septiembre 2023 de 19:00 a 20:15 horas de Chile – "Pitch Class y Aclaratorias generales"

Todas las sesiones de capacitación quedarán grabadas y estarán disponibles en línea para los participantes del evento.

Agenda

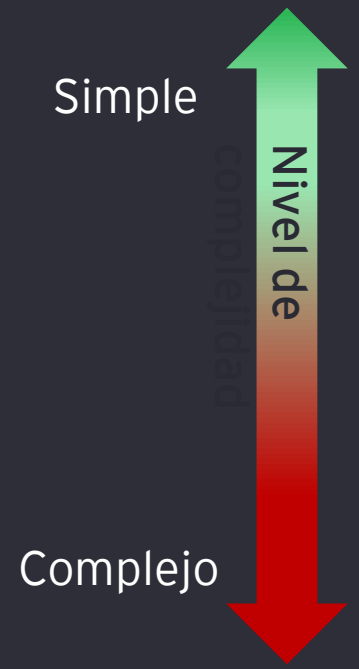
1. Introducción a LangChain y Modelos
2. Retrievals
3. Chains
4. Memoria
5. Agentes

¿Qué es LangChain?

LangChain es un marco para desarrollar aplicaciones impulsadas por modelos de lenguaje. Permite aplicaciones que son conscientes del contexto y pueden razonar. Ofrece componentes para trabajar con modelos de lenguaje y cadenas prefabricadas para tareas de alto nivel. Estas cadenas facilitan el inicio, mientras que los componentes permiten personalización.

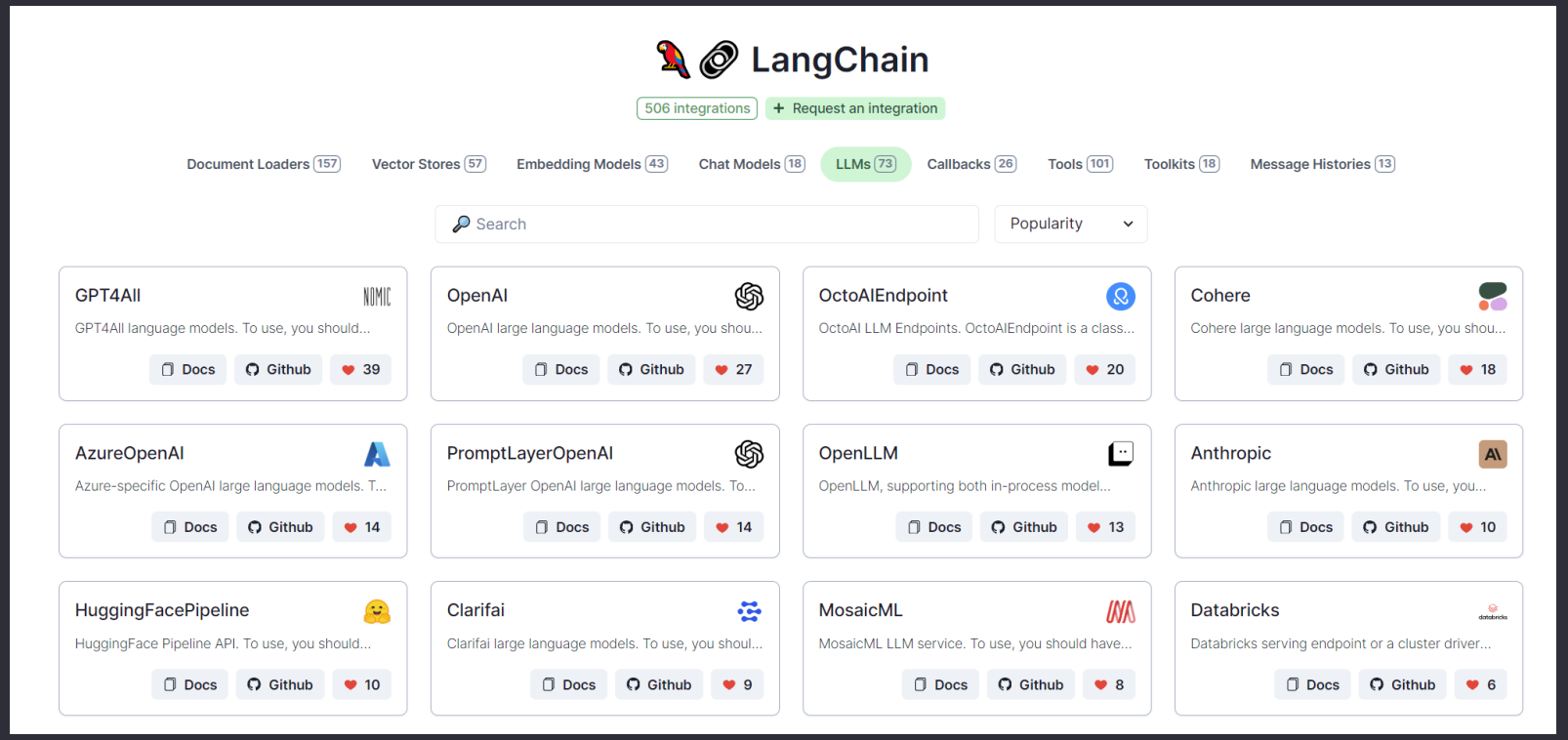
Principales conceptos que revisaremos:

- **Model I/O**: *Interfaz con LLMs*
- **Retrieval**: *Interfaz para interactuar con datos específicos de la aplicación*
- **Chains**: *Construcción de secuencia de llamadas*
- **Memory**: *Persistencia en el estado de la aplicación entre ejecuciones de una cadena*
- **Agents**: *Permite que las cadenas elijan qué herramientas usar dadas directrices de alto nivel*



Modelos en LangChain

La oferta de modelos actual es amplia, y según sea el requerimiento a realizar es la importancia de elegir uno sobre otro. Existen modelos capaces de sintetizar información, de generar conversaciones, corregir código entre múltiples otras actividades. LangChain ofrece una amplia gama de posibilidades para conectarse a ellos:

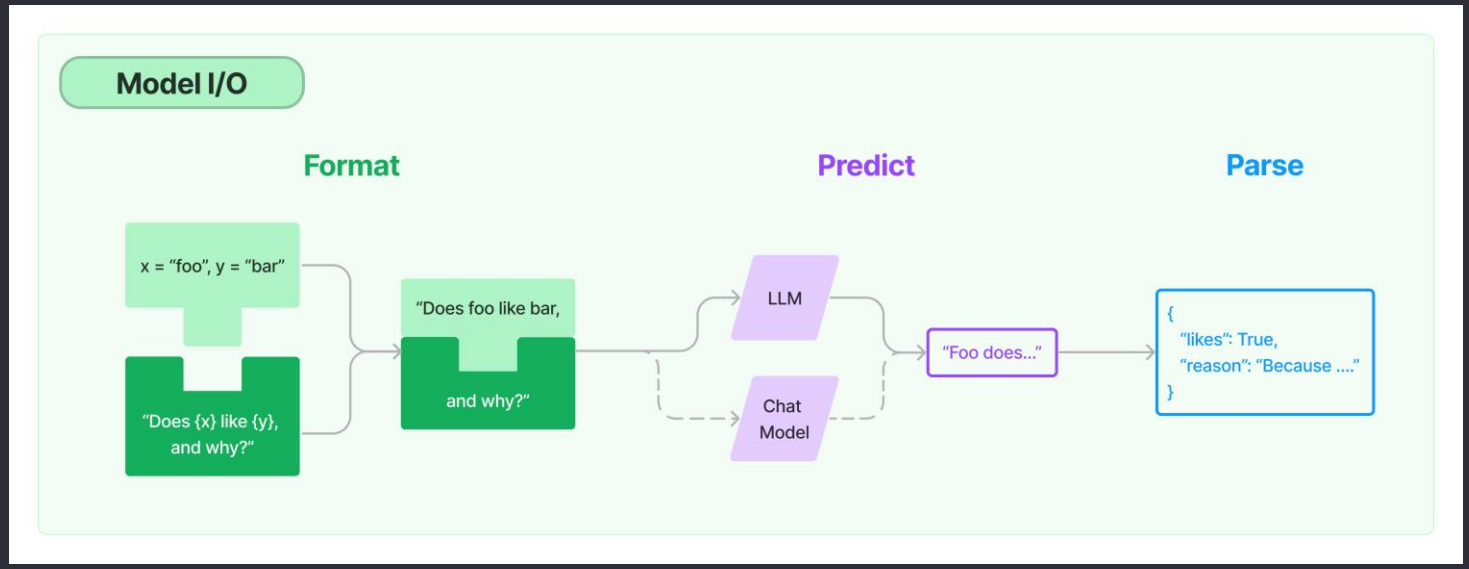


Listado completo de integraciones: https://integrations.langchain.com/

Modelos en LangChain

Pero hay algo en común al momento de crear soluciones que interactuar con estos modelos, y es que hay 3 piezas que convergen en la ecuación:

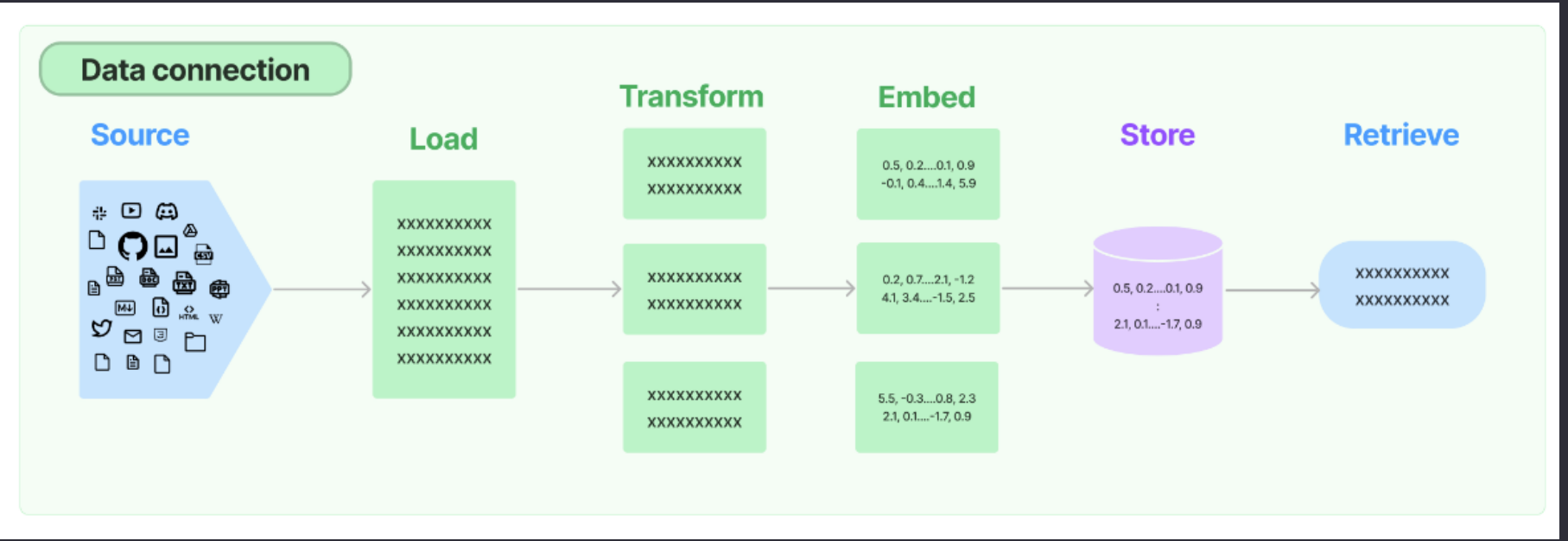
- Prompts: Caracterizar la forma de ingreso de información hacia el modelo
- Language models: Comunicación con el modelo de lenguaje mediante alguna interfaz
- Output parsers: La obtención y manipulación de información como respuesta del modelo



Referencia: https://python.langchain.com/docs/modules/model_io/

Retrievals

Cuando diseñamos una solución basada en LLM y requerimos del uso de datos que no han sido parte de la fase de entrenamiento del modelo, es que el concepto de retrieval toma importancia. En particular, mediante RAG (Retrieval Augmented Generation), el proceso "recupera" datos externos para que sean provistos al modelo en la etapa de generación.
Las principales fases a revisar son: Load, Transform, Embed, Store y Retrieve.



Referencia: https://python.langchain.com/docs/modules/data_connection/

Retrievals

Document Loaders

LangChain ofrece una cantidad importante de opciones según el formato que se esté manipulando. Lo importante de esta etapa es que la información que se está cargando en LangChain se transforma en un objeto de tipo "Documento". Este paso incorpora metadatos a la pieza de texto que se está manipulando

Import langchain.document_loaders

Document Transformers

Luego de la carga del documento, se debe evaluar alguna forma de "particionar" la pieza de información en segmentos de información, de tal manera que, tanto el modelo como el proceso en sí, pueda ser ejecutado correctamente.
Una forma es mediante "splitters", cuyos parámetros como chunk size, chunk overlap, length fun se hacen esenciales

Import langchain.text_splitter

Text Embeddings

De forma simplificada, Embeddings es una forma de representar texto en vectores numéricos. Esto facilita, por ejemplo, la búsqueda semántica puesto que la similitud de textos se realiza en un espacio vectorial multidimensional

Import langchain.embeddings

Vector Store

Almacenamiento indexado de aquellas representaciones vectoriales de piezas de texto

Import langchain.vectorstores

Chains

Una Cadena en LLM es una estrategia que agrega funcionalidades entorno a los modelos de lenguaje. Esto da versatilidad a la hora de crear flujos de razonamiento (creación de agentes)

Asimismo, al momento de construir una aplicación basada en LLM, la presencia de Cadenas facilita el proceso de debugging y mantención de la solución

En simple, una Cadena consta de un Prompt Template y un modelo de lenguaje (que puede ser LLM o un modelo de chat). El Prompt Template se forma mediante el ingreso de inputs (y, eventualmente, una componente de memoria), pasando el string formateado al LLM y devolviendo el output desde el propio LLM.

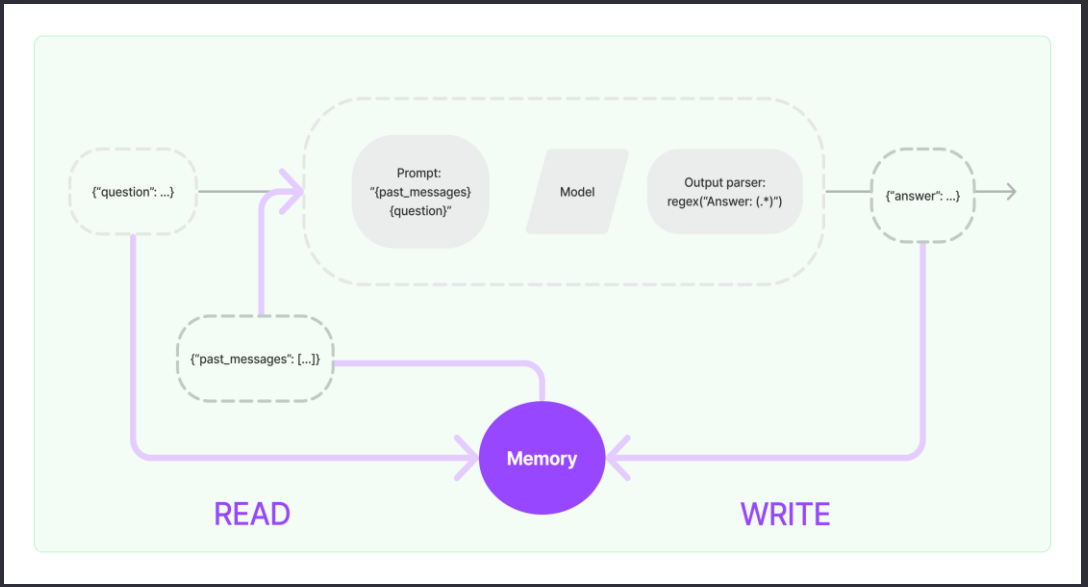
Memoria

Para LangChain, el concepto de "Memoria" hace referencia a la posibilidad de acceso al historial de conversaciones del pasado, durante el proceso de interacción del usuario con la aplicación basada en LLM.

Para estos efectos, existen múltiples mecanismos de manipulación de esta "memoria", en términos de qué y cuánto es el almacenamiento óptimo de historial, y el cómo se almacena ese historial para su manipulación posterior y entrega al modelo a modo de contexto.

Un sistema de memoria necesita de dos acciones básicas: leer y escribir. Recuerdar que cada cadena define alguna lógica de ejecución central que espera ciertas entradas. Algunas de estas entradas provienen directamente del usuario, pero algunas de estas entradas pueden provenir de la memoria. Una cadena interactuará con su sistema de memoria dos veces en una ejecución determinada:

- DESPUÉS de recibir las entradas iniciales del usuario pero ANTES de ejecutar la lógica central, una cadena LEERÁ desde su sistema de memoria y aumentará las entradas del usuario
- DESPUÉS de ejecutar la lógica central pero ANTES de devolver la respuesta, una cadena ESCRIBIRÁ las entradas y salidas de la ejecución actual en la memoria, para que se pueda hacer referencia a ellas en ejecuciones futuras

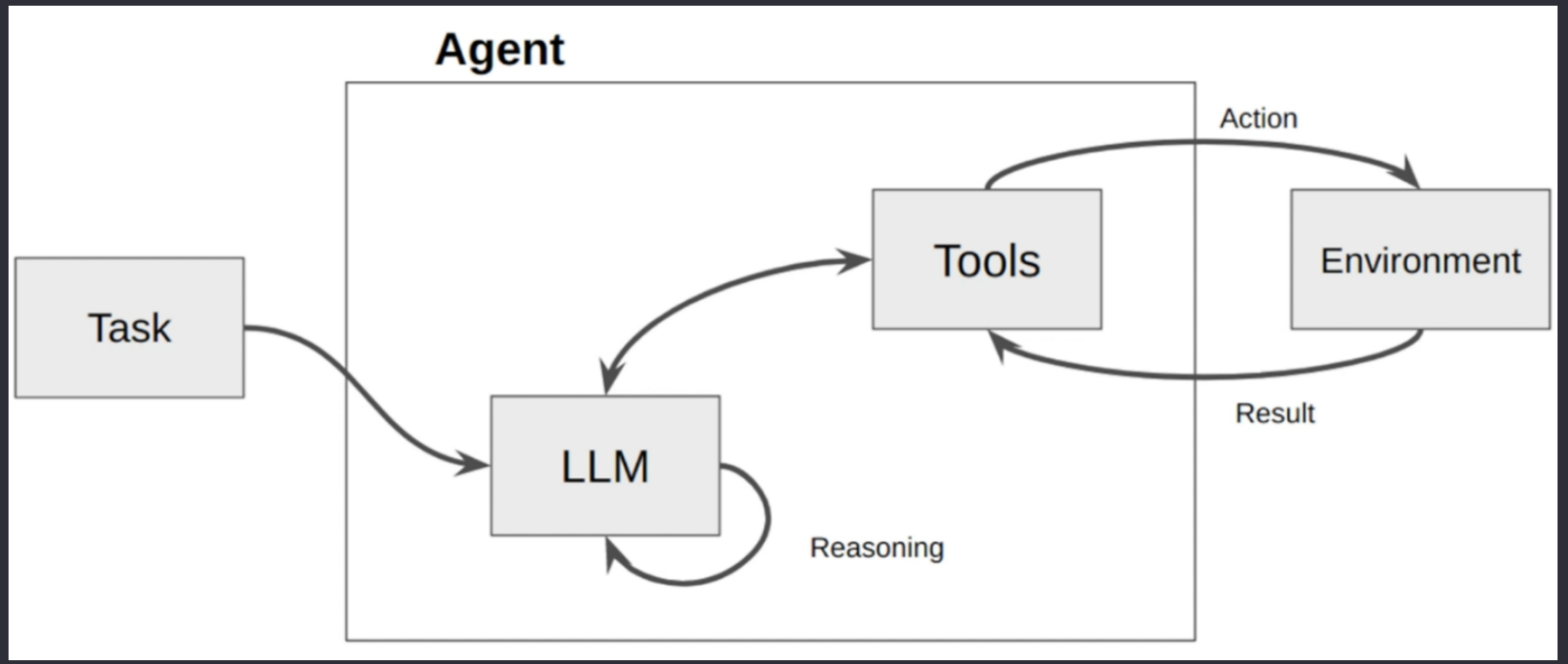


Referencia: https://python.langchain.com/docs/modules/memory/

Agentes – El objetivo a alcanzar

La idea central de los Agentes es utilizar un LLM para elegir una secuencia de acciones a realizar. En las Cadenas, una secuencia de acciones está preestablecida en el código. En los agentes, se utiliza un modelo de lenguaje como motor de razonamiento para determinar qué acciones realizar y en qué orden.

Es en esta parte donde la aplicación de LLM se extiende fuera de su ámbito de acción, e interactúa con el entorno (fuentes externas)



Preguntas o Consultas

Avisos y fechas importantes

Fechas de sesiones

- ~~Sesión 1: martes 12 de septiembre - “Design Thinking”~~
- ~~Sesión 2: miércoles 13 de septiembre - “LLM & Prompting”~~
- ~~Sesión 3: jueves 14 de septiembre - “APIS de OpenAI”~~
- Sesión 4: martes 26 de septiembre - “Langchain”
- Sesión 5: miércoles 27 de septiembre - “Casos de Uso / Wireframing”
- Sesión 6: jueves 28 de septiembre 2023 – “Pitch Class y Aclaratorias generales”

  (Sesión 6 es OBLIGATORIA, deben participar a lo menos 2 integrantes del equipo)


HACKATÓN
XI
2023
SUMMIT
PAÍS
DIGITAL
FUNDACIÓN
país
digital
By:
EY
Building a better working world
Colabora:
Microsoft