Aplicaciones LLM

Langchain: Quickstart

LangChain Quickstart

- Repaso de la Quickstart de LangChain a finales de Octubre de 2023.

Contenidos

- Qué es LangChain y para qué sirve.
- Cómo conectar con OpenAI.
- Crear los componentes más habituales de una chain de LangChain:
  - Language Model.
  - Prompt Template.
  - Output Parser.
- Crear una chain.
- Primer contacto con el LangChain Expression Language.

Qué es LangChain y para qué sirve.

- LangChain es una framework para crear aplicaciones basadas en LLMs.

- ¿Para qué sirve? Permite crear aplicaciones que:
  - Conectan el LLM con una fuente externa de datos.
  - Ayudan a los LLM a razonar.

- ¿Qué aporta? Dos elementos principales:
  - Componentes: módulos que trabajan con los LLM.
  - Chains: cadenas de componentes.

Componentes más habituales en una chain

- **Language Model:** el LLM.
- **Prompt Template:** un template del prompt final que será completado con el input del usuario.
- **Output Parser:** una funcionalidad para transformar la respuesta del LLM
  - Texto LLM en datos estructurados (como por ejemplo JSON).
  - ChatMessage en texto (string).
  - Extra info (como OpenAI function invocation) en texto (string).

Tipos de Language Models en Langchain

- LLM: recibe texto (string), responde texto.
- ChatModel: recibe mensaje de chat, responde mensaje de chat.

Un mensaje de chat puede tener 4 orígenes o roles

- **Lo que dice el humano:** HumanMessage.
- **Lo que responde el LLM (le llaman AI):** AIMessage.
- **Las instrucciones que damos al LLM sobre su rol, temperatura, etc:** SystemMessage.
- **Lo que se diga como resultado de ejecutar una función:** FunctionMessage.

Primer paso: conectar con nuestra cuenta de OpenAI

- Evita incluir to API key en el jupyter notebook.
- Pon la API key en un archivo externo y oculto.
- Utiliza los módulos os y python-dotenv para llamar al contenido del archivo externo.
- Incluye los módulos para conectar con el LLM y el Chat de chatGPT3.5

Segundo paso: comunícate con chatGPT3.5

- Crea las instancias de LLM y Chat.

- Utiliza predict y predict_messages para comunicarte con el LLM y el Chat y recibir sus respuestas.

- ¡Ojo! El parquímetro avanzará cada vez que te comuniques con OpenAI.
  - Las operaciones sencillas son muy baratas.
  - Te recomendamos abrir tu gasto de OpenAI para tenerlo controlado en todo momento.

- Observa las diferencias entre las respuestas del LLM y el Chat.
- Observa que OpenAI no responde bien si le preguntamos por la fecha actual.

Tercer paso: experimenta con Prompt Templates

- Crea el PromptTemplate con .from_template()
- Introduce la variable con .format()
- Envía el prompt final al LLM y al Chat.
- Repite el proceso con ChatPromptTemplate.
  - Crea el template con las variables
  - Crea la variable que introduce el humano
  - Crea el ChatPromptTemplate con .from_messages()
  - Crea el prompt final con .format_messages()

Cuarto paso: crea un Output Parser

- Importa el módulo BaseOutputParser
- Crea un Output Parser para convertir un texto con palabras separadas por comas en una lista de python.
- Pruébalo con un texto de ejemplo.

Quinto paso: crea una chain

- Combina los 3 componentes anteriores en una chain
  - LM: el Chat de OpenAI.
  - Prompt Template con instrucciones para el Chat y texto de usuario.
  - Output Parser para convertir lista de palabras en una lista de python.
- Primer contacto con el LangChain Expression Language.
- Invocar la chain con el texto de usuario.