

# LLM

## Base de Datos

La colección `vocabulary` almacenaría objetos JSON que representen palabras y frases de cada idioma. Cada objeto JSON tendría los siguientes campos:

```
{  
  "word": "palabra",  
  "translation": "traducción",  
  "pronunciation": "pronunciación",  
  "use": "uso"  
}
```

La colección `grammar` almacenaría objetos JSON que representen reglas gramaticales de cada idioma. Cada objeto JSON tendría los siguientes campos:

```
{  
  "rule": "regla",  
  "explanation": "explicación",  
  "examples": "ejemplos"  
}
```

La colección `text` almacenaría objetos JSON que representen texto en cada idioma. Cada objeto JSON tendría los siguientes campos:

```
{  
  "text": "texto",  
  "language": "idioma",  
  "category": "categoría"  
}
```

La colección `users` almacenaría objetos JSON que representen información sobre los usuarios del tutor de idiomas. Cada objeto JSON tendría los siguientes campos:

```
{  
  "name": "nombre",  
  "email": "correo electrónico",  
  "preferred_languages": "idiomas preferidos",  
  "skill_level": "nivel de habilidad"  
}
```

Chatbots are one of the central LLM use-cases. The core features of chatbots are that they can have long-running conversations and have access to information that users want to know about.

Aside from basic prompting and LLMs, memory and retrieval are the core components of a chatbot. Memory allows a chatbot to remember past interactions, and retrieval provides a chatbot with up-to-date, domain-specific information.

Here's a quick preview of how we can create chatbot interfaces. First let's install some dependencies and set the required credentials:

```
pip install langchain openai

# Set env var OPENAI_API_KEY or load from a .env file:
# import dotenv
# dotenv.load_dotenv()
```

With a plain chat model, we can get chat completions by passing one or more messages to the model.

The chat model will respond with a message.

```
from langchain.schema import (    AIMessage,    HumanMessage,    SystemMessage ) from
langchain.chat_models import ChatOpenAI chat =
ChatOpenAI() chat([HumanMessage(content="Translate this sentence from English to French: I
love programming.")])
```

```
AIMessage(content="J'adore la programmation.", additional_kwargs={}, example=False)
```

And if we pass in a list of messages:

```
messages = [
    SystemMessage(content="You are a helpful assistant that translates English to French."),
    HumanMessage(content="I love programming.")
]
chat(messages)
```

```
AIMessage(content="J'adore la programmation.", additional_kwargs={}, example=False)
```

We can then wrap our chat model in a `ConversationChain`, which has built-in memory for remembering past user inputs and model outputs.

```
from langchain.chains import ConversationChain

conversation = ConversationChain(llm=chat)
conversation.run("Translate this sentence from English to French: I love programming.")
```

## API Reference:

- [ConversationChain](#)

```
'Je adore la programmation.'
```

```
conversation.run("Translate it to German.")
```

```
'Ich liebe Programmieren.'
```

## Python

```
import langchain

client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client.get_database("my_database")
collection = db.get_collection("my_collection")

# Carga datos de la base de datos en la API de LangChain
data = collection.find()

for doc in data:
    langchain.load_data(doc)
```

- Utiliza la función `load_data()` para cargar datos de tu base de datos en la API de LangChain. Esto te permitirá utilizar los datos existentes en tu base de datos para generar nuevos datos.
- Utiliza la función `save_data()` para guardar datos de la API de LangChain en tu base de datos. Esto te permitirá almacenar los datos generados por la API de LangChain para utilizarlos posteriormente.
- Utiliza la función `generate_data()` para generar nuevos datos en la API de LangChain. Esto te permitirá crear contenido personalizado para tus usuarios.

Este código cargará todos los documentos de la colección `my_collection` en la API de **LangChain**.

También puedes utilizar la API de **LangChain** para escribir datos en una base de datos. Por ejemplo, podrías utilizar la función `generate_data()` para generar datos en la API de **LangChain**, y luego la función `save_data()` para guardar los datos generados en tu base de datos.

Aquí hay un ejemplo de cómo utilizar la API de LangChain para escribir datos en una base de datos de MongoDB:

## Python

```
import langchain

client = MongoClient("mongodb://localhost:27017/")
db = client.get_database("my_database")
collection = db.get_collection("my_collection")

# Genera datos en la API de LangChain
data = langchain.generate_data()

# Guarda los datos generados en la base de datos
collection.insert_many(data)
```

Este código generará datos en la API de **LangChain** y los guardará en la colección `my_collection`.

La forma exacta en que integres la API de **LangChain** con tu base de datos dependerá de tus necesidades específicas. Sin embargo, los ejemplos anteriores te dan una buena idea de cómo funciona el proceso.

## Chat Model

[OpenAI](#) |  [Langchain](#)

Para obtener una licencia de OpenAI, deberás crear una cuenta en el sitio web de OpenAI. Una vez que tengas una cuenta, podrás solicitar una licencia. El proceso de solicitud es sencillo y requiere que proporciones información sobre tu aplicación y tus necesidades.

Una vez que OpenAI apruebe tu solicitud, recibirás un correo electrónico con tus credenciales de licencia. Estas credenciales incluyen una clave API y un secreto API.

Para utilizar el ChatOpenAI con tu licencia, deberás configurar las credenciales de OpenAI en tu aplicación. Puedes hacerlo siguiendo las instrucciones de la documentación de OpenAI.

Aquí hay un ejemplo de cómo configurar las credenciales de OpenAI en Python:

## Python

```
import openai

# Set env var OPENAI_API_KEY or load from a .env file:
# import dotenv
# dotenv.load_dotenv()

# Set API key and secret
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.api_secret = os.getenv("OPENAI_API_SECRET")
```

Una vez que hayas configurado las credenciales de OpenAI, podrás utilizar el ChatOpenAI para crear chatbots.