

Promptior Chatbot

El objetivo es hacer un chatbot con Langchain que implemente una arquitectura RAG, para retornar informacion que se encuentra en promptior.ai y que sea capaz de responder preguntas acerca de Promptior, como por ejemplo "Que servicios ofrece Promptior?" y "Cuando fue fundada la empresa?". Y que ese mismo chatbot se pueda interactuar desde Langserve Playground.

Las Tecnologias utilizadas fueron:

- Python
- Langchain
- LangSmith (Evaluador)
- LangServe
- Redis (Chat History)
- Pinecone (Vector Store)
- Replit (IDE y Cloud provider)

A continuacion te presentare las etapas de la implementacion, anexandote imagenes, codigo y comentarios de los problemas que tuve en el camino y como los resolvi.

Preprocessing

1. **Webscraping:** Obtener el HTML del website, así como sus links asociados.
Uno de los principales razones por la que no use la clase de langchain para hacer Scraping es porque no me retornaba el HTML completo. Decidi construir funciones que retornara correctamente el html de la pagina principal y de la de los links asociados, ignorando los links que retornaban a la misma pagina y los que eran links externos.

Resolucion de Problema: El primer problema fue que no me cargaba bien el sitio y le agregue una condicion para que esperara a que un Xpath estuviera presente, despues de eso que me retornara el HTML. El segundo fue que cuando queria que retornara el HTML de los links asociados, excepto los externos y los de la pagina principal.

```
class WebScraper:
    def __init__(self, url):
        self.main_url = url
        self.browser = None
        self.page = None

    def setup_browser(self, playwright):
        self.browser = playwright.chromium.launch(headless=True)
        self.page = self.browser.new_page()

    def navigate_and_extract(self):
        self.page.goto(self.main_url, wait_until="networkidle")
        self.page.wait_for_selector("xpath=//div[@id='root']")
        html_content = self.page.content()
        title = self.page.title()

        links = self.page.query_selector_all("a")
        hrefs = [self.page.evaluate("el => el.getAttribute('href')") for el in links]

        main_domain = urlparse(self.main_url).netloc

        for href in hrefs:
            if href.startswith("/") and not href.startswith("/"):
                href = self.main_url.rstrip("/") + href
            if urlparse(href).netloc == main_domain:
                print(f"Navegando a {href}")
                try:
                    self.page.goto(href, wait_until="networkidle")
                    page_content = self.page.content()
```

```

        except Exception as e:
            print(f"Error al navegar a {href}: {e}")
        finally:
            self.page.goto(self.main_url, wait_until='load')
    else:
        print(f"Enlace externo o sección omitida: {href}")

    return html_content, page_content, title

def run(self):
    with sync_playwright() as playwright:
        self.setup_browser(playwright)
        html_content, page_content, title = self.navigate()
        self.browser.close()
        return html_content, page_content, title

```

Retrieval System

1. Text Processing: Despues de obtener el HTML, se proceso a texto con `HTML2TextTransform` y al ver que el Texto me lo entrego con los titulos y los subitulos con el formato markdown (`#` , `##`, `###`).
2. Split in Chunks: Se uso `MarkdownHeaderTextSplitter` para obtener los chunks, y que cada chunk correspondiera al tema que le corresponde. En este caso como el contexto de la informacion no era muy grande, elegi por dividirlo solo por el header `##`, por ejemplo:

Who are we?

At Promptior, we're more than consultants, we're your partners in harnessing the transformative power of AI. We believe the real magic happens not just through technology implementation, but when it's fully embraced within every corner of your organization. We're committed to fostering an ecosystem where innovation isn't merely applied, but seamlessly integrated, becoming an essential part of your daily processes, strategic decisions, and operational framework. We're here to guide you every step of the way on this transformative journey, ensuring it's not just smooth, but truly empowering. With Promptior by your side you're stepping confidently into a future full of unprecedented possibilities.

```
def transform_and_split(self, html_content, page_content, title):
    doc_1 = Document(page_content=html_content)
    doc_2 = Document(page_content=page_content)

    html2text_transformer = Html2TextTransformer()
    transformed_docs = html2text_transformer.transform_documents([doc_1, doc_2])

    promptior_content = f"## {title} \n {transformed_docs}"

    headers_to_split_on = [
        ("##", "title")
        # ("###", "sub_title"),
    ]

    markdown_splitter = MarkdownHeaderTextSplitter(headers_to_split_on)
    docs = markdown_splitter.split_text(promptior_content)

    for doc in docs:
        if 'title' in doc.metadata:
            doc.page_content = f"Title: {doc.metadata['title']} \n {doc.page_content}"

    return docs
```

De igual forma a los metadata se le agrego el titulo del header que el corresponde.

3. Embed Chunks: Se uso el modelo de OpenAI `text-embedding-3-small` con dimension de 1536.
4. Vector Store: Use Pinecone (`PineconeVectorStore`), por sus diferentes formas de hacer busqueda y tambien por calidad y precio en su mejora de Serverless.

Serverless Indexes

\$99.38 serverless credits

AC

BROWSER

METRICS

NAMESPACES (1)

Matches: 1-4 of 4

+ Upsert Record

1

ID

VALUES

0c4b4442-25c...

...

SCORE

-0.0444

METADATA

text: "# Promptior\n≡ \nEN \n/ \nES \n* About\n* Services\n* Contact \nCreate **unprecedented** opportunities in an ever-evolving world. \nUnleash the potential o...

2

ID

VALUES

33268151-527f...

...

SCORE

-0.0542

METADATA

text: "Title: Who are we? \n\n ## Who are we? \nAt Promptior, we're more than consultants, we're your partners in harnessing\nthe transformative power of AI. We bel...
title: "Who are we?"

Resolucion de Problemas: En esta etapa tuve problemas con definir de que manera era mas eficiente hacer splits de la informacion, ya que que al implementar el RAG, hacia busqueda por similaridad y me retornaba chunks que no deberia (Los retornaba porque en ciertos chunks contenia la palabra "Services", que era parte del header o footer del html). Eso pasaba cuando realizaba la pregunta "Que servicios ofrece la empresa?". Por lo tanto al estar analizando bien la situacion, no era necesario que en mi retrival se retornaran 2 documentos de contexto, y por el caso de es muy poca la informacion porcedi con el aumento del top_k=4. Tambien aplique Rerank con Cohere (Para hacer un filtrado de la informacion y en ves de 4 chunks, fueran 2 o 3), pero vi que no era necesario el Rerank y lo deje en top_k = 4.

Agent

Se creo un Agente con Chat History, para mejorar la conversacion entre Usuario-Agente, tambien agregandole capacidades para ejecutar Tools en el momento que sea requerida.

```

def create_agent(self):
    prompt = ChatPromptTemplate.from_messages(
        [
            (
                "system",
                "You are very powerful assistant."
                "Talk with the user as normal. "
            ),
            MessagesPlaceholder(variable_name="chat_history"),
            ("user", "{input}"),
            MessagesPlaceholder(variable_name="agent_scratchpad"),
        ]
    )

    llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo-0125")
    agent = create_openai_functions_agent(llm, self.tools, prompt)
    agent_executor = AgentExecutor(agent=agent, tools=self.tools,
    return agent_executor

def run(self):
    agent_with_chat_history = RunnableWithMessageHistory(
        self.agent,
        self.get_message_history,
        input_messages_key="input",
        history_messages_key="chat_history",
    ).with_types(input_type=Input, output_type=Output).with_config(

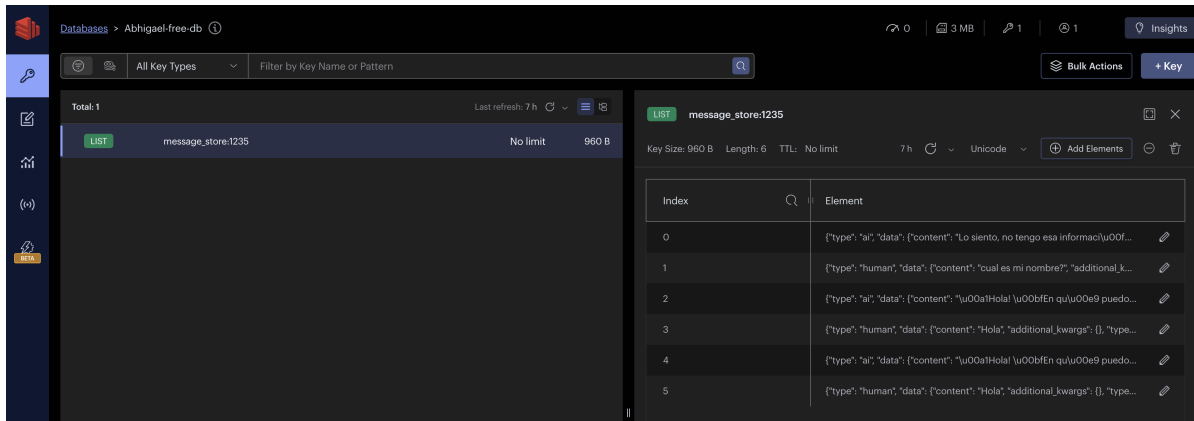
    return agent_with_chat_history

```

1. Tools \leftrightarrow `search_promptior` \rightarrow Retrieval(\rightarrow Input Query \rightarrow List[Documents]): Se creo para que cuando el usuario realice una pregunta acerca de Promptior, haga busqueda por similitud y retorne una lista de documentos para el contexto.
2. Model \rightarrow gpt-3.5-turbo-0125 : Se uso este modelo por fines de calidad-precio.

3. Chat history: Aqui implemente *Redis* para almacenar y consultar el historial de las conversaciones, dependiendo del "session_id" (Para identificar a quien pertenece).

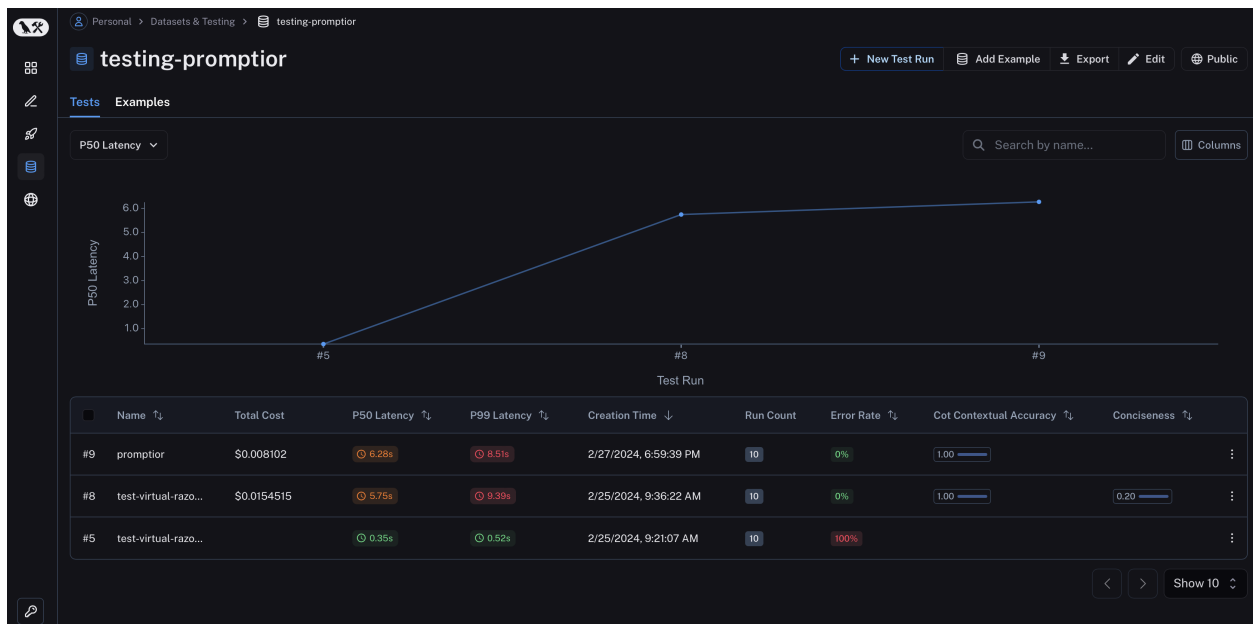
```
def get_message_history(self, session_id: str):  
    return RedisChatMessageHistory(session_id, url=self....
```



Resolucion de Problemas: Aqui el porblema que tuve fue al principio la integracion con Redis, me sucedian problemas de Autenticacion por la URL ingresada, pero ya al estar leyendo documentacion tenia que agregarles mas parametros como la password dentro de la URL, despues de eso el historial de las conversaciones lo registra correctamente dependiendo del "session_id" que se asigne.

Evaluation Benchmark

Cree una evaluacion para evaluar la capacidad de pregunta y respuesta del Chatbot con LangSmith.



Genere una lista de preguntas y respuestas esperadas. Despues con `RunEvalConfig` use la metrica de "Cot Contextual Accuracy", para evaluar que lo que responda el chatbot este dentro del contesto de la respuesta correcta.

```
def evaluation_agent(self, dataset_name="testing-promptior", pro
    eval_config = RunEvalConfig(
        evaluators=[
            "cot_qa"
        ],
        custom_evaluators=[],
        eval_llm=ChatOpenAI(model="gpt-3.5-turbo-0125", temper
    )

    client = langsmith.Client()
    chain_results = client.run_on_dataset(
        dataset_name=dataset_name,
        llm_or_chain_factory=self.agent,
        evaluation=eval_config,
        project_name=project_name,
        concurrency_level=5,
        verbose=True,
```



```
)
return chain_results
```


Con gpt-3.5-turbo-0125, evalúe si la respuesta generada estaba bien o mal.

Input	Reference Output	#9 promptior →
input: ¿Qué es Promptior?	output: Promptior es una empresa que se especializa en aprovechar el poder transformador de la inteligencia artificial para crear oportunidades sin precedentes en un mundo en constante evolución. Actúan como socios en la implementación de IA, integrándola en todos los aspectos de una organización.	output: Promptior es una empresa fundada en marzo de 2023 con la misión de democratizar y facilitar el acceso a la Inteligencia Artificial para personas y organizaciones en todo el mundo. Su equipo está compuesto por profesionales con amplia experiencia en tecnología e inteligencia artificial, comprometidos a proporcionar las herramientas, recursos y conocimientos necesarios para ayudar a sus clientes a adoptar rápidamente estos beneficios. Promptior se dedica al éxito de sus clientes y al avance de la IA a nivel mundial, trabajando para construir un futuro más inteligente y conectado para todos. Ofrecen servicios como capacitación personalizada, soluciones de IA personalizadas, evaluación de cartera y empoderamiento de equipos de desarrollo de software con IA. COT CONTEXTUAL ACCURACY: 1 → 6.18% SUCCESS 1.278 Open Run
input: ¿Cuál es la misión de Promptior?	output: La misión de Promptior es democratizar y facilitar el acceso a la inteligencia artificial, impulsando la innovación, mejorando la eficiencia y transformando la forma en que las personas trabajan, aprenden y colaboran.	output: La misión de Promptior es democratizar y facilitar el acceso a la Inteligencia Artificial para personas y organizaciones en todo el mundo. Su objetivo es impulsar la innovación, mejorar la eficiencia y transformar radicalmente la forma en que las personas trabajan, aprenden y colaboran. Promptior se compromete a proporcionar las herramientas, recursos y conocimientos necesarios para ayudar a sus clientes a adoptar rápidamente estos beneficios y a construir un futuro más inteligente y conectado para todos. COT CONTEXTUAL ACCURACY: 1 → 5.20% SUCCESS 1.237 Open Run
input: ¿Qué servicios ofrece Promptior?	output: Promptior ofrece servicios como capacitación personalizada para todos los niveles organizativos, soluciones personalizadas de IA y estrategia de producto, evaluación de cartera para identificar disrupciones y oportunidades con IA, y empoderamiento de equipos de desarrollo de software con formación en IA.	output: [-] Promptior ofrece los siguientes servicios: 1. Capacitación y formación personalizada. 2. Soluciones personalizadas de IA y estrategia de productos. 3. Evaluación de cartera. 4. Empoderamiento de equipos de software con IA. Estos servicios están diseñados para ayudar a las organizaciones a aprovechar el poder de la inteligencia artificial de manera efectiva en sus operaciones y estrategias comerciales. COT CONTEXTUAL ACCURACY: 1 →

Link de la prueba: <https://smith.langchain.com/public/4cf46e18-3b0e-4a46-8ef6-601413d9af80/d>

Deployment

- Langserve:
- UI → Langserve Playground: Se ingresa el ID para la conversación que quieren almacenar y el input.

 **LangServe Playground**

Configure

Configurable

SESSION ID

123

Try it

Inputs

Reset

INPUT*

Hola

Output

```
{
  "output": "¡Hola! ¿En qué puedo ayudarte hoy?",
  "messages": [
    {
      "content": "¡Hola! ¿En qué puedo ayudarte hoy?",
      "additional_kwargs": {},
      "type": "ai",
      "name": null,
      "id": null,
      "example": false
    }
  ]
}
```

Intermediate steps 11

>

- Cloud provider → Replit

Uno de los problemas que tuve al principio fue la integración del agente que cree a la app de langserve, para que las dependencias no tuvieran conflicto. Pasando un momento de respiro me puse a leer la documentación y langserve pudo correr perfectamente. También el otro fue al querer hacer el deploy con AWS, intente

realizar la configuracion y obtuve unos errores, pero me puse a buscar y vi que se podia usar LangServe en Replit (Es donde normalmente hago el deploy de mis aplicaciones), y el despliegue fue muy rapido.