



LANGCHAIN: SEU 1º APP COM LLMS

Use o poder do LangChain e crie sua primeira aplicação com modelos de linguagem de grande porte (LLM).

PYTHONACADEMY.COM.BR

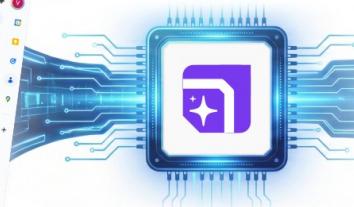
Gere ebooks como este com



Ebookr

em <https://ebookr.ai>

Crie ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!



Capas gerados por IA



Infográficos feitos por IA



Edite em Markdown em Tempo Real



Adicione Banners Promocionais

TESTE AGORA



PRIMEIRO CAPÍTULO 100% GRÁTIS

 **Atualizado para LangChain 0.1+** (Março 2025) Seu primeiro app com LLMs: OpenAI GPT-4, Anthropic Claude, modelos locais, casos práticos e boas práticas.

Salve salve Pythonista 

Desenvolver aplicações que utilizam **Modelos de Linguagem de Grande Porte (LLMs)** está se tornando cada vez mais popular.

Neste contexto, o **LangChain** surge como uma poderosa ferramenta para facilitar a criação de aplicações com LLMs usando Python.

Neste artigo, você aprenderá:

- O que é o LangChain
- Como instalá-lo e configurá-lo
- como conectar-se a diferentes provedores de LLMs como **OpenAI** e **Hugging Face**,
- Criar seu primeiro prompt simples
- Executar seu primeiro modelo de linguagem e, por fim
- Desenvolver um exemplo prático de um gerador de texto simples.

Se é novo e quer primeiro ler um artigo introdutório sobre o assunto, [clique aqui e leia este outro artigo que escrevi sobre o LangChain](#) (e depois volte pra cá 😊)

O que é LangChain



LangChain é uma biblioteca Python que facilita a integração e o gerenciamento de LLMs em aplicações.

Com o crescente uso de modelos de linguagem como GPT-4, LangChain oferece abstrações que simplificam tarefas complexas, como gestão de prompts, manipulação de respostas e integração com diversas fontes de dados.

Utilizar LangChain com Python é vantajoso devido à flexibilidade da linguagem e à vasta comunidade de desenvolvedores que contribuem para seu ecossistema.

Além disso, LangChain é altamente compatível com outras bibliotecas populares de Python, tornando-o uma escolha ideal para desenvolvedores que buscam construir soluções robustas e escaláveis.

Instalação e configuração do ambiente LangChain

Para começar a usar o LangChain, primeiro **instale-o** no seu ambiente Python.

É recomendado utilizar um ambiente virtual para gerenciar as dependências do seu projeto.

Se não souber configurar um ambiente virtual, [veja esse artigo completo sobre virtualenv!](#)

```
pip install langchain openai
```

Após a instalação, você precisa **configurar as credenciais** dos provedores de LLM que pretende utilizar.

Por exemplo, para usar a OpenAI, defina a variável de ambiente `OPENAI_API_KEY` com sua chave de API.

```
export OPENAI_API_KEY='sua-chave-api'
```

Isso garante que suas credenciais estejam seguras e acessíveis para o LangChain.

E adivinha só: também temos um artigo completo ensinando a integrar o ChatGPT ao seu código Python, é só [clicar aqui](#) e depois voltar pra cá 😊

Conectando a um LLM: OpenAI, Hugging Face e mais.

LangChain suporta diversos provedores de LLMs, como **OpenAI**, **Hugging Face** e outros.

Vamos ver como se conectar a alguns deles.

Conectando-se à OpenAI

Para conectar-se à OpenAI, certifique-se de que sua chave de API está configurada.

Em seguida, utilize o seguinte código:

```
from langchain import OpenAI

modelo = OpenAI(api_key='sua-chave-api')
resposta = modelo("Qual é a capital da França?")
print(resposta)
```

E a saída será:

```
Paris
```

Conectando-se à Hugging Face

Para usar modelos da Hugging Face, instale a biblioteca necessária e configure a chave de API.

```
pip install huggingface_hub
```

```
from langchain import HuggingFaceHub

modelo = HuggingFaceHub(repo_id="gpt-neo-2.7B", api_key='sua-chave-
api')
resposta = modelo("Explique a teoria da relatividade.")
print(resposta)
```

E a saída será algo similar à:

A teoria da relatividade, desenvolvida por Albert Einstein, descreve a interação entre espaço e tempo e como a gravidade afeta essa interação.

Outros Provedores

Além de OpenAI e Hugging Face, LangChain suporta outros provedores como **Cohere**, **AI21**, entre outros.

A conexão segue um padrão similar, onde você inicializa o modelo com as credenciais apropriadas e utiliza os métodos disponíveis para interagir com ele.



Quer ver o LangChain em ação? O [Ebookr.ai](#) é um exemplo real de aplicação construída com LLMs e LangChain. Ele gera ebooks profissionais sobre qualquer tema, com capa gerada por IA, infográficos e revisão automática — tudo usando as mesmas técnicas que você está aprendendo aqui. Confere!

Crie Ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

... e deixe que nossa IA faça o trabalho pesado!

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

 Capas gerados por IA

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

Criando seu primeiro prompt simples com LangChain

Agora que conectamos ao LLM, vamos criar um **prompt simples** para obter uma resposta.

```
from langchain import OpenAI

modelo = OpenAI(api_key='sua-chave-api')
prompt_simples = "Escreva uma breve introdução sobre Python."
resposta = modelo(prompt_simples)
print(resposta)
```

E a saída será:

Python é uma linguagem de programação versátil e poderosa, amplamente utilizada para desenvolvimento web, análise de dados, automação e inteligência artificial.

Explicação do código:

- 1. Importação:** Importamos a classe `OpenAI` do LangChain.
- 2. Inicialização do Modelo:** Criamos uma instância do modelo com a chave de API.
- 3. Definição do Prompt:** Especificamos o prompt que queremos enviar ao modelo.
- 4. Obtenção da Resposta:** Chamamos o modelo com o prompt e recebemos a resposta.
- 5. Exibição da Resposta:** Imprimimos a resposta gerada pelo modelo.

Exemplo prático: um gerador de texto simples

Vamos aplicar o que aprendemos para **criar um gerador de texto simples** que cria descrições de produtos.

```

from langchain import OpenAI

def gerar_descricao(produto, características):
    modelo = OpenAI(api_key='sua-chave-api')
    prompt = f"Crie uma descrição atraente para um produto chamado '{produto}' com as seguintes características: {características}."
    descricao = modelo(prompt)
    return descricao

# Exemplo de uso
produto = "Smartwatch XYZ"
características = "monitor de frequência cardíaca, GPS integrado, bateria com duração de 7 dias"
descricao = gerar_descricao(produto, características)
print(descricao)

```

E a saída será:

O Smartwatch XYZ combina estilo e funcionalidade, oferecendo monitoramento preciso da frequência cardíaca, GPS integrado para rastreamento fácil de suas atividades e uma bateria que dura até 7 dias, garantindo que você esteja sempre conectado.

Explicação do código:

- 1. Definição da Função:** Criamos uma função `gerar_descricao` que recebe o nome do produto e suas características.
- 2. Inicialização do Modelo:** Dentro da função, inicializamos o modelo OpenAI com a chave de API.
- 3. Criação do Prompt:** Montamos um prompt que solicita a criação de uma descrição baseada nas características fornecidas.
- 4. Execução do Modelo:** Passamos o prompt para o modelo e recebemos a descrição.

5. **Retorno da Descrição:** A função retorna a descrição gerada.
6. **Uso da Função:** Demonstramos como usar a função com um exemplo de produto.

A descrição gerada será uma narrativa atraente destacando as características do produto, pronta para uso em estratégias de marketing.

Mais Casos Práticos

1. Chatbot com Memória

```
from langchain.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationChain

def criar_chatbot():
    """Chatbot que lembra do contexto da conversa"""
    llm = ChatOpenAI(temperature=0.7)
    memory = ConversationBufferMemory()
    conversation = ConversationChain(
        llm=llm,
        memory=memory,
        verbose=True
    )
    return conversation

# Uso
bot = criar_chatbot()
print(bot.predict(input="Olá, meu nome é Maria"))
print(bot.predict(input="Qual é meu nome?")) # Lembra que é Maria!
```

2. Sumarizador de Textos Longos

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains.summarize import load_summarize_chain
from langchain.chat_models import ChatOpenAI
from langchain.docstore.document import Document

def sumarizar_texto_longo(texto):
    """Sumariza textos que excedem limite de tokens"""
    # Dividir texto em chunks
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=2000,
        chunk_overlap=200
    )

    docs = [Document(page_content=texto)]
    split_docs = text_splitter.split_documents(docs)

    # Chain de sumarização
    llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo")
    chain = load_summarize_chain(llm, chain_type="map_reduce")

    summary = chain.run(split_docs)
    return summary

# Uso
artigo_longo = """... texto muito longo ..."""
resumo = sumarizar_texto_longo(artigo_longo)
print(resumo)
```

3. Validador de Dados com LLM

```
from langchain.chat_models import ChatOpenAI
from langchain.output_parsers import PydanticOutputParser
from pydantic import BaseModel, Field, validator

class Endereco(BaseModel):
    rua: str
    numero: int
    cidade: str
    estado: str
    cep: str

    @validator('cep')
    def validar_cep(cls, v):
        if len(v) != 8:
            raise ValueError('CEP deve ter 8 dígitos')
        return v

def extrair_endereco(texto_livre):
    """Extrai endereço estruturado de texto livre"""
    parser = PydanticOutputParser(pydantic_object=Endereco)

    llm = ChatOpenAI(temperature=0)
    prompt = f"""Extraia o endereço deste texto:
{texto_livre}

{parser.get_format_instructions()}"""

    result = llm.predict(prompt)
    endereco = parser.parse(result)
    return endereco

# Uso
texto = "Moro na Rua das Flores, número 123, São Paulo, SP, CEP
01234567"
endereco = extrair_endereco(texto)
print(f"Cidade: {endereco.cidade}, Estado: {endereco.estado}")
```

Boas Práticas

1. Controle de Custos

```
from langchain.callbacks import get_openai_callback

with get_openai_callback() as cb:
    result = chain.run("Sua pergunta")
    print(f"Custo: ${cb.total_cost:.4f}")
    print(f"Tokens: {cb.total_tokens}")
```

2. Tratamento de Erros

```
import openai
from tenacity import retry, stop_after_attempt, wait_exponential

@retry(
    stop=stop_after_attempt(3),
    wait=wait_exponential(multiplier=1, min=4, max=10)
)
def call_llm_with_retry(prompt):
    try:
        llm = ChatOpenAI()
        return llm.predict(prompt)
    except openai.error.RateLimitError:
        print("⚠️ Rate limit atingido, tentando novamente...")
        raise
    except Exception as e:
        print(f"🔴 Erro: {e}")
        raise
```

3. Usar Streaming para Respostas Longas

```
from langchain.chat_models import ChatOpenAI
from langchain.callbacks.streaming_stdout import
    StreamingStdOutCallbackHandler

# Exibe resposta em tempo real
llm = ChatOpenAI(
    streaming=True,
    callbacks=[StreamingStdOutCallbackHandler()],
    temperature=0.7
)

response = llm.predict("Escreva um artigo sobre IA")
# Palavras aparecem uma a uma na tela!
```

Conclusão

Neste guia sobre **Seu Primeiro App com LLMs**, você aprendeu:

✓ **Setup básico** - Instalação e configuração LangChain ✓ **Primeiro prompt** - OpenAI e Hugging Face ✓ **Casos práticos** - Chatbot, sumarizador, validador ✓ **Boas práticas** - Custos, erros, streaming ✓ **Gerador de texto** - App completo passo a passo ✓ **Memória** - Chatbot que lembra contexto

Principais lições:

- LangChain **simplifica** integração com LLMs
- **Monitore custos** com callbacks
- Use **retry** para lidar com rate limits
- **Streaming** melhora UX em respostas longas
- **Pydantic** para estruturar dados

Próximos passos:

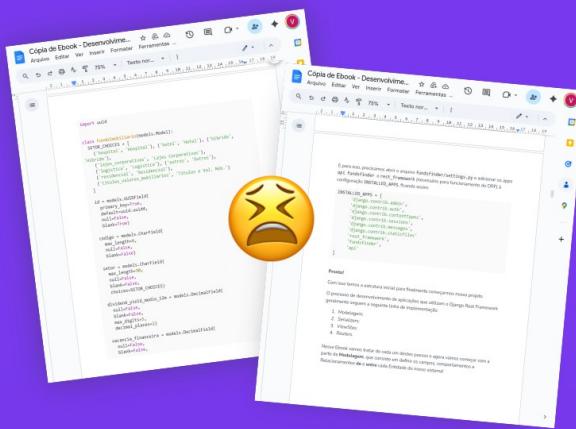
- Explore [LangChain](#) conceitos
- Aprenda [Prompts](#) avançados
- Integre [ChatGPT](#) em apps
- Estude sobre RAG e embeddings

Não se esqueça de conferir!



Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Capas gerados por IA



Infográficos feitos



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

TESTE AGORA



 PRIMEIRO CAPÍTULO 100% GRÁTIS