



COMO UTILIZAR PROMPTS E PROMPTTEMPLATES NO LANGCHAIN

Aprenda a criar prompts eficazes para obter as melhores respostas dos seus LLMs.

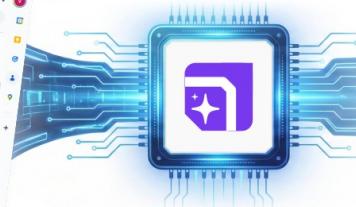
Gere ebooks como este com



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)** Prompts e PromptTemplates: simples, few-shot, output parsers, casos práticos e boas práticas de prompt engineering.

Salve salve Pythonista 

Prompts e **PromptTemplates** são elementos fundamentais no desenvolvimento de aplicações com **Inteligência Artificial** e **Processamento de Linguagem Natural**.

No contexto do **LangChain**, uma poderosa framework em Python para construção de aplicativos de linguagem, entender como utilizar esses conceitos é essencial para criar interações eficientes e precisas com modelos de linguagem.

Neste artigo, exploraremos o conceito de prompts, seus tipos, técnicas avançadas de construção e como implementá-los usando **LangChain**. Também compartilharemos dicas para otimizar seus prompts e exemplos práticos para diferentes tarefas.

Dominar esses conhecimentos é crucial para desenvolver aplicações robustas e inteligentes que aproveitam ao máximo as capacidades dos modelos de linguagem.

Esse artigo faz parte de uma sequência completa sobre o LangChain. Leia também os outros artigos:

- [O que é o LangChain e como funciona](#)
- [Sua primeira aplicação com LLMs](#)

E fique ligado que outros vão aparecer em breve 

O que são Prompts e sua importância

Prompts são instruções ou perguntas fornecidas a modelos de linguagem como o GPT-4 para gerar respostas específicas.

Eles são a base para direcionar o comportamento do modelo, determinando a qualidade e a relevância das respostas obtidas.

A importância dos prompts reside na capacidade de controlar e refinar as respostas dos modelos, tornando-os úteis para uma variedade de aplicações, desde chatbots até geração de conteúdo automatizado.

Tipos de Prompts

Prompts Simples

Prompts simples são instruções diretas e concisas que solicitam uma resposta específica do modelo.

Exemplo:

Explique o conceito de programação orientada a objetos

E a saída será:

Conceito de Programação Orientada a Objetos: A Programação Orientada a Objetos (POO) é um paradigma de programação que utiliza "objetos" para modelar dados e comportamentos através de classes, facilitando a organização e reutilização de código.

Prompts Few-shot

Prompts **few-shot** fornecem ao modelo alguns exemplos de entrada e saída antes da pergunta principal.

Isso ajuda o modelo a entender o contexto, o formato e o tipo de resposta esperado.

Exemplo 1:

Pergunta: O que é herança em POO?

Resposta: Herança é um princípio onde uma classe pode herdar atributos e métodos de outra classe.

Exemplo 2:

Pergunta: O que é encapsulamento?

Resposta: Encapsulamento é o conceito de esconder os detalhes internos de uma classe e expor apenas o necessário.

Pergunta: O que é polimorfismo?

Resposta:

E a saída será:

Polimorfismo é a capacidade de diferentes classes responderem ao mesmo método de maneira específica, permitindo flexibilidade e reutilização de código.

Técnicas Avançadas de Construção de Prompts

Construir prompts eficazes requer entender nuances e aplicar técnicas que maximizem a qualidade das respostas.

Uso de Instruções Claras

Forneça instruções precisas para evitar ambiguidades.

Liste três vantagens da programação funcional em Python.

E a saída será:

1. Imutabilidade dos dados
2. Facilita a depuração e manutenção
3. Suporte aprimorado para processamento paralelo

Contextualização

Adicione contexto relevante para guiar o modelo.

Você é um assistente de programação especializado em Python.

Explique como funciona a função map().

E a saída será:

A função `map()` aplica uma função específica a cada item de um iterável (como listas ou tuplas) e retorna um iterador com os resultados. É útil para realizar operações repetitivas de forma concisa e eficiente.

Limitação de Respostas

Defina limites para a resposta desejada, como o número de itens ou o comprimento.

Resuma o capítulo 1 do livro 'Python para Iniciantes' em até 100 palavras.

E a saída será:

O capítulo 1 de 'Python para Iniciantes' introduz os conceitos básicos da programação, incluindo sintaxe fundamental, tipos de dados e estruturas de controle. Ele estabelece as bases necessárias para o desenvolvimento de habilidades em Python, preparando o leitor para tópicos mais avançados nas próximas seções.

Utilizando Prompts no LangChain

O **LangChain** facilita a integração e gestão de prompts em aplicações Python.

Vamos ver como utilizar prompts simples e few-shot com LangChain.

Instalação do LangChain

Primeiro, instale o LangChain usando `pip` - de preferência um ambiente virtual:

```
pip install langchain langchain-community langchain-openai
```

Exemplo de Uso de Prompt Simples

Agora vamos ver como é simples criar

```
import getpass
import os
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate

# Defina e salve a chave de API da OpenAI nas variáveis de ambiente
os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI API key: ")

# Definindo o prompt simples
template = "Explique o conceito de {assunto}."

# Criando o template
prompt = PromptTemplate(
    input_variables=["assunto"],
    template=template,
)

# Inicializando o modelo OpenAI
llm = OpenAI(temperature=0.7)

# Gerando a resposta
resposta = llm.invoke(prompt.format(assunto="programação orientada a
                                    objetos"))
print(resposta)
```

E a saída será:

Conceito de Programação Orientada a Objetos: A Programação Orientada a Objetos (POO) é um paradigma de programação que utiliza "objetos" para modelar dados e comportamentos através de classes, facilitando a organização e reutilização de código.

Explicação do código:

- 1. Importações:** Importamos `OpenAI` e `PromptTemplate` do LangChain.
- 2. Definição da Chave de API da OpenAI:** Inserimos a chave de API da OpenAI nas variáveis de ambiente ([clique aqui para aprender a gerar as suas próprias chaves](#)).
- 3. Definição do Template:** Criamos um template que inclui uma variável `{assunto}`.
- 4. Criação do Prompt:** Passamos as variáveis de entrada e o template para `PromptTemplate`.
- 5. Inicialização do Modelo:** Configuramos o modelo OpenAI com uma temperatura para controle de criatividade.
- 6. Geração da Resposta:** Preenchemos o template com o assunto desejado e invocamos a resposta do modelo.

Utilizando `PromptTemplates` no LangChain

`PromptTemplates` permitem criar prompts parametrizados, tornando-os reutilizáveis e flexíveis para diferentes entradas.

Exemplo de PromptTemplate com Vários Exemplos

```

import os
import getpass
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate

# Defina e salve a chave de API da OpenAI nas variáveis de ambiente
os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI API key: ")

# Definindo o prompt com exemplos few-shot
template = """
Exemplo 1:
Pergunta: O que é herança em POO?
Resposta: Herança é um princípio onde uma classe pode herdar atributos
          e
          métodos de outra classe.

Exemplo 2:
Pergunta: O que é encapsulamento?
Resposta: Encapsulamento é o conceito de esconder os detalhes internos
          de
          uma classe e expor apenas o necessário.

Pergunta: {pergunta}
Resposta:
"""

# Criando o template
prompt = PromptTemplate(
    input_variables=["pergunta"],
    template=template,
)

# Inicializando o modelo OpenAI
llm = OpenAI(temperature=0.7)

# Gerando a resposta
resposta = llm.invoke(prompt.format(pergunta="O que é polimorfismo?"))
print(resposta)

```

E a saída será:

Polimorfismo é a capacidade de diferentes classes responderem ao mesmo método de maneira específica, permitindo flexibilidade e reutilização de código.

Explicação do código:

- 1. Definição do Template:** Incluímos exemplos de perguntas e respostas para guiar o modelo.
- 2. Criação do Prompt:** Usamos `PromptTemplate` com a variável `{pergunta}`.
- 3. Geração da Resposta:** Preenchemos o template com a nova pergunta e obtemos a resposta.

Dicas e Truques para Otimizar seus Prompts

Seja Claro e Conciso

Evite ambiguidades e mantenha o prompt direto ao ponto.

Use Formatação Adequada

Utilize listas, negrito e itálico para destacar informações importantes.

Experimente com Parâmetros

Ajuste parâmetros como `temperature` e `max_tokens` para controlar a criatividade e o tamanho da resposta.

Teste e Itere

Teste diferentes versões dos prompts e itere com base nas respostas obtidas para aprimorar a eficácia.

Exemplos de Prompts para Diferentes Tarefas

Geração de Resumo

```

import os
import getpass
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate

os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI API key: ")

template = "Resuma o seguinte texto em uma frase: {texto}"

prompt = PromptTemplate(
    input_variables=["texto"],
    template=template,
)

llm = OpenAI(temperature=0.5)

resumo = llm.invoke(
    prompt.format(
        texto="""
        O LangChain é uma biblioteca em Python projetada para facilitar
        a criação
        de aplicações que utilizam modelos de linguagem de grande porte
        (LLMs),
        como o GPT-4, o DeepSeek ou o Grok.

        Sua importância reside na capacidade de integrar e orquestrar
        diferentes
        componentes necessários para desenvolver funcionalidades
        avançadas, como
        chatbots inteligentes, sistemas de perguntas e respostas (Q&A)
        e
        ferramentas de sumarização de texto.
        """
    )
)
print(resumo)

```

E a saída será:

LangChain é uma framework poderosa que utiliza modelos de linguagem avançados para criar aplicações de linguagem eficientes.

Tradução de Texto

```
import os
import getpass
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate

os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI API key: ")

template = "Traduza o seguinte texto para inglês: {texto}"

prompt = PromptTemplate(
    input_variables=["texto"],
    template=template,
)

llm = OpenAI(temperature=0.3)

traducao = llm.invoke(prompt.format(texto="Olá, como você está?"))
print(traducao)
```

E a saída será:

Hello, how are you?

Geração de Código

```
import os
import getpass
from langchain_openai import OpenAI
from langchain.prompts import PromptTemplate

os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI API key: ")

template = "Crie uma função em Python que {descricao}."

prompt = PromptTemplate(
    input_variables=["descricao"],
    template=template,
)

llm = OpenAI(temperature=0.2)

codigo = llm.invoke(prompt.format(descricao="verifique se um número é
    primo"))
print(codigo)
```

E a saída será:

```
def eh_primo(numero):
    if numero < 2:
        return False
    for i in range(2, int(numero ** 0.5) + 1):
        if numero % i == 0:
            return False
    return True
```

Explicação do código:

- 1. Resumir Texto:** Cria um prompt para resumir um texto específico.
- 2. Tradução:** Define um prompt para traduzir texto do português para o inglês.

3. Geração de Código: Configura um prompt para gerar uma função Python com base em uma descrição.

 Estou desenvolvendo o **Ebookr.ai**, uma plataforma que transforma suas ideias em ebooks profissionais usando IA — com geração de capa, infográficos e exportação em PDF. Te convido a conhecer!

 **Ebookr**

Crie Ebooks profissionais incríveis em minutos com IA



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

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

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

 Capas gerados por IA  Adicione Banners Promocionais  Edite em Markdown em Tempo Real  Infográficos feitos por IA

Conclusão

Neste guia sobre **Prompts no LangChain**, você aprendeu:

 **Conceitos fundamentais** - O que são prompts e por que importam 
PromptTemplates - Templates reutilizáveis com variáveis  **Few-shot prompts** - Ensinar LLMs por exemplos  **Output parsers** - Estruturar respostas com Pydantic  **Casos práticos** - Resumo, tradução, geração de código  **Boas práticas** - Clareza, contexto, exemplos, iteração

Principais lições:

- **Prompts claros** produzem melhores resultados
- **Few-shot learning** é poderoso para tarefas específicas
- **PromptTemplates** facilitam reutilização
- **Output parsers** garantem estrutura consistente
- **Iterar e testar** é essencial para otimizar

Próximos passos:

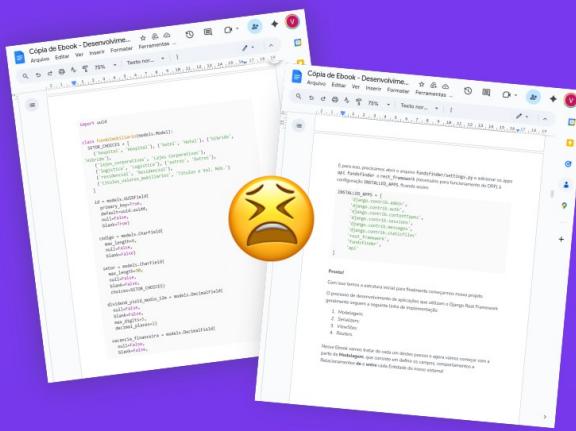
- Explore [LangChain](#) conceitos básicos
- Pratique com seu primeiro app LLM
- Estude sobre Chains e Agents
- Experimente diferentes modelos (GPT-4, Claude, Llama)

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 para...



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

TESTE AGORA



 PRIMEIRO CAPÍTULO 100% GRÁTIS