Exercícios Progressivos de LangChain Chains

Pré-requisitos

- Python 3.8+
- LangChain instalado: `pip install langchain langchain-openai`
- Chave de API da OpenAI configurada

Exercícios

Exercício 1: Chain Simples com PromptTemplate

Crie uma chain que recebe um nome de animal e gera uma descrição básica usando um PromptTemplate.

Tarefa:

- Use `PromptTemplate` para criar um template que pergunta sobre um animal
- Conecte com um LLM usando o operador `|`
- Teste com diferentes animais

Exercício 2: Chain com OutputParser

Modifique o exercício anterior para incluir um parser de saída que retorne apenas texto limpo.

Tarefa:

- Use `StrOutputParser()` para processar a saída
- Crie uma chain: `prompt | llm | parser`

Exercício 3: Chain com Múltiplas Variáveis

Crie uma chain que recebe nome, idade e profissão de uma pessoa e gera uma biografia.

Tarefa:

- Template deve ter múltiplos placeholders: `{nome}`, `{idade}`, `{profissao}`
- Use `.invoke()` com dicionário de variáveis

Exercício 4: Chain Sequencial Básica

Crie duas chains: uma que gera uma história curta e outra que a resume.

Tarefa:

- Primeira chain: gera história a partir de um tema
- Segunda chain: resume a história gerada
- Execute sequencialmente passando o resultado da primeira para a segunda

---

Exercício 5: Chain com RunnableLambda

Crie uma chain que processa texto com uma função personalizada entre o prompt e o LLM.

Tarefa:

- Use `RunnableLambda` para criar uma função que capitaliza o input
- Chain: `prompt | lambda_function | llm | parser`

Exercício 6: Chain com RunnablePassthrough

Crie uma chain que mantém o input original disponível na saída final.

Tarefa:

- Use `RunnablePassthrough()` para preservar o input
- Combine com `RunnableParallel` para ter input e output juntos

Exercício 7: Chain com Branches Paralelas

Crie uma chain que gera duas análises diferentes do mesmo texto em paralelo.

Tarefa:

- Use `RunnableParallel` para criar duas análises: "positiva" e "crítica"
- Combine os resultados em um dicionário

Exercício 8: Chain Condicional Simples

Crie uma chain que escolhe entre dois templates baseado no comprimento do input.

Tarefa:

- Use `RunnableBranch` para decidir entre template "curto" ou "longo"
- Condição: se len(input) < 50, usar template curto

Exercício 9: Chain com Map-Reduce

Processe uma lista de tópicos gerando uma descrição para cada um e depois um resumo geral.

Tarefa:

- Use `RunnableMap` para processar múltiplos tópicos
- Combine todos os resultados em um resumo final

Exercício 10: Chain com Retry e Error Handling

Crie uma chain com tratamento básico de erro usando RunnableLambda.

Tarefa:

- Implemente uma função que tenta executar o LLM e retorna erro amigável se falhar
- Use try/except dentro de um RunnableLambda

Exercício 11: Chain de Tradução Sequencial

Crie uma chain que traduz texto por múltiplos idiomas sequencialmente.

Tarefa:

- Português → Inglês → Francês → Português
- Cada etapa deve ser uma chain separada conectada

Exercício 12: Chain com Routing Complexo

Crie um sistema que roteia perguntas para chains especializados baseado na categoria.

Tarefa:

- Detecte se é pergunta sobre: ciência, história ou tecnologia
- Tenha 3 chains especializados diferentes
- Use RunnableBranch com múltiplas condições

Exercício 13: Chain com Memória Simples

Implemente uma chain que mantém contexto das últimas 3 interações.

Tarefa:

- Use uma lista simples para armazenar histórico
- Inclua contexto no prompt atual
- Atualize o histórico a cada execução

Exercício 14: Chain de Análise Multi-Step

Crie uma análise de texto em 3 etapas: extração de temas, análise de sentimento, e conclusão.

Tarefa:

- Cada etapa usa o resultado da anterior
- Combine todos os resultados em um relatório final
- Use RunnableParallel onde apropriado

Exercício 15: Chain Master com Combinação de Técnicas

Crie uma chain complexa que combina várias técnicas aprendidas nos exercícios anteriores.

Tarefa:

- Receba um texto longo
- Processe em paralelo: resumo, análise de sentimento, e extração de palavras-chave
- Use branching para diferentes tipos de output baseado no tamanho
- Inclua error handling
- Retorne resultado estruturado

Estrutura Recomendada para Cada Exercício

```python
# Template básico para cada exercício
from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser

# 1. Configuração
llm = ChatOpenAI(temperature=0.7)

# 2. Criação do prompt

# 3. Criação da chain

# 4. Teste da chain

# 5. Print do resultado
```

Próximos Passos

Após completar estes exercícios, você estará pronto para:

- Trabalhar com chains mais complexas
- Implementar agentes

- Usar ferramentas (tools)
- Criar aplicações completas com classes
- Integrar com bases de dados vetoriais