



Introdução ao LangChain

Professor: Elton Sarmanho¹

E-mail: eltonss@ufpa.br



¹Faculdade de Sistemas de Informação - UFPA/CUNTINS

6 de fevereiro de 2025

Roteiro

Planejamento

Fundamentação
Conceitos Fundamentais

Trabalhando com Modelos de Linguagem



Roteiro

Agentes e Memória no LangChain

Prompts e Indexação no LangChain

Python com LangChain



Roteiro

Retrieval Augmented Generation (RAG)

Referências Bibliográficas



Licença

Este trabalho está licenciado sob a licença Creative Commons:



Nesta aula:

- ▶ Compreender o que são Modelos de Linguagem (LLMs).
- ▶ Entender o conceito de Cadeias (Chains) no LangChain.
- ▶ Introduzir a base matemática e computacional por trás dos LLMs.
- ▶ Explorar exemplos práticos de uso de LLMs e Chains.
- ▶ Códigos do Professor estão no github

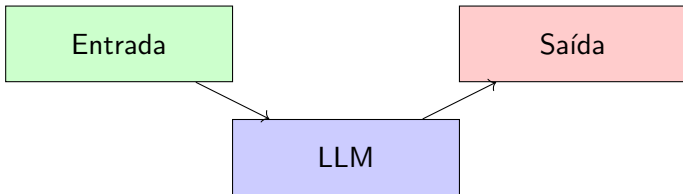


Conceitos Fundamentais



O que são Modelos de Linguagem (LLMs)?

- ▶ Modelos de linguagem são sistemas de inteligência artificial treinados para gerar e entender texto.
- ▶ Baseados em arquiteturas de redes neurais, como Transformers.
- ▶ Exemplos: GPT-3, GPT-4, BERT, etc.



Modelos de Linguagem e Representação Matemática

Um LLM pode ser visto como uma função:

$$f(x) = P(y|x; \theta) \quad (1)$$

Onde:

- ▶ x é a entrada textual;
- ▶ y é a saída gerada;
- ▶ θ são os parâmetros do modelo.

O LangChain encapsula essa função para criar fluxos mais complexos de IA.



Por que LangChain?

- ▶ Grandes Modelos de Linguagem (LLMs) revolucionaram o processamento de texto.
- ▶ LangChain permite criar aplicações avançadas de IA baseadas em LLMs.
- ▶ Foca em pipelines modulares para fácil integração com dados externos.



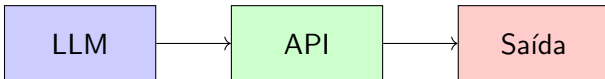
Arquitetura do LangChain

- ▶ **Modelos:** OpenAI, Hugging Face, Sabiá 2.
- ▶ **Cadeias (Chains):** Conectam múltiplas operações de IA.
- ▶ **Agentes (Agents):** Tomam decisões com base em entradas dinâmicas.
- ▶ **Memória:** Armazena contexto de conversas anteriores.
- ▶ **Bases de Dados Vetoriais:** FAISS, Pinecone, ChromaDB.

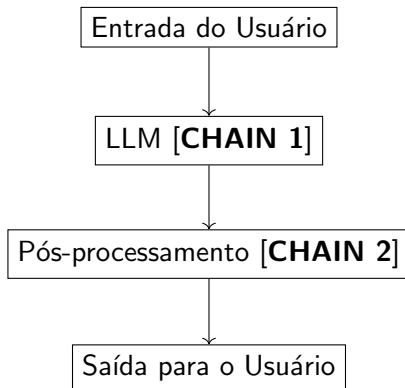


O que são Cadeias (Chains) no LangChain?

- ▶ Cadeias são sequências de chamadas a modelos ou outras funções que são executadas em uma ordem específica.
- ▶ Exemplo: Uma cadeia pode envolver a chamada a um modelo de linguagem, seguida por uma chamada a uma API externa, e então a geração de uma resposta final.



Fluxo de uma Cadeia no LangChain



Exemplo de Código em Python

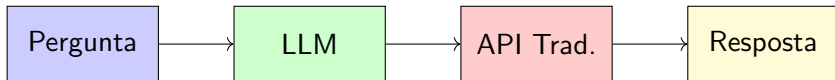
```
from langchain.llms import OpenAI  
llm = OpenAI(model_name="gpt-4")  
resposta = llm.predict("o que e buraco negro")  
print(resposta)
```



Exemplo Prático de uma Cadeia

► Considere uma cadeia que:

1. Recebe uma pergunta do usuário.
2. Chama um modelo de linguagem para gerar uma resposta preliminar.
3. Chama uma API de tradução para traduzir a resposta para outro idioma.
4. Retorna a resposta traduzida ao usuário.



Modelos de Linguagem



Modelos de Linguagem

- ▶ Como LangChain se conecta a diferentes modelos de LLMs.
- ▶ Importância do ajuste fino dos prompts.
- ▶ Estratégias de otimização de desempenho.



Escolha do Modelo Adequado

- ▶ Modelos open-source vs. modelos proprietários.
- ▶ Custo-benefício na utilização de LLMs.
- ▶ Comparação de latência e acurácia.



Embeddings e Representação Vetorial

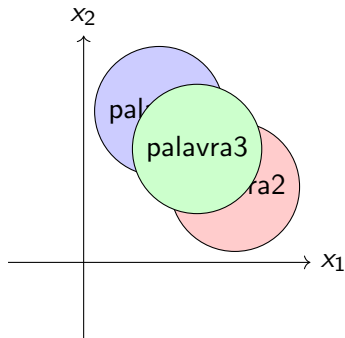
Definição: Representação numérica de palavras/sentenças para cálculos matemáticos.

$$E(w) = \mathbf{v} \in \mathbb{R}^d \quad (2)$$

Onde d é a dimensão do espaço vetorial e \mathbf{v} é o vetor associado à palavra w .



Espaço Vetorial de Embeddings



Agentes Memória no LangChain



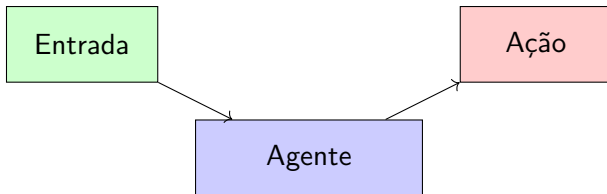
Objetivos

- ▶ Compreender o conceito de Agentes no LangChain.
- ▶ Entender como a Memória é usada para manter o contexto.
- ▶ Explorar a base matemática e computacional por trás dos Agentes.
- ▶ Ver exemplos práticos de Agentes e Memória em ação.



O que são Agentes no LangChain?

- ▶ Agentes são componentes que tomam decisões sobre quais ações executar com base em entradas e contextos específicos.
- ▶ Eles podem decidir qual modelo chamar, qual API usar, ou até mesmo qual cadeia de ações executar.
- ▶ Exemplos: ZeroShotAgent, ReActAgent.



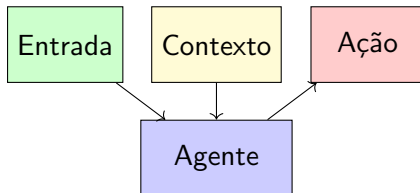
Base Computacional dos Agentes

- ▶ Agentes usam técnicas de **planejamento** e **tomada de decisão** para escolher a melhor ação.
- ▶ Podem ser modelados como um problema de **otimização**:

$$Ação^* = \arg \max_{a \in A} P(a \mid \text{Entrada}, \text{Contexto})$$

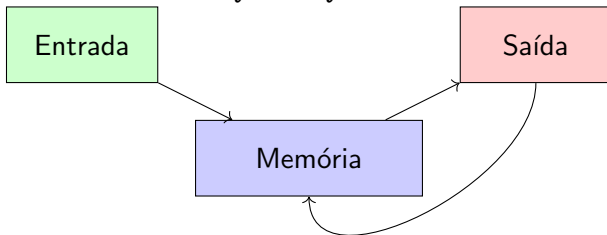
onde:

- ▶ A: Conjunto de ações possíveis.
- ▶ P: Função de probabilidade que avalia a adequação da ação.



O que é Memória no LangChain?

- ▶ Memória é a capacidade de armazenar e recuperar informações entre diferentes interações ou chamadas.
- ▶ Mantém o contexto de uma conversa ou de um fluxo de trabalho.
- ▶ Exemplos: `ConversationBufferMemory`, `ConversationSummaryMemory`.



Estrutura da Memória

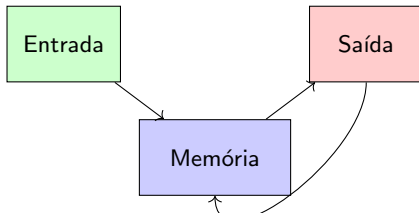
- ▶ A memória pode ser modelada como um **estado** S que evolui ao longo do tempo:

$$S_{t+1} = f(S_t, Entrada_t)$$

onde:

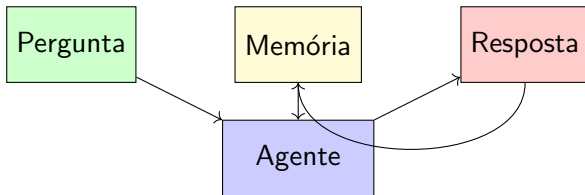
- ▶ S_t : Estado atual da memória.
- ▶ $Entrada_t$: Nova entrada no tempo t .
- ▶ f : Função de atualização do estado.
- ▶ A saída é gerada com base no estado atual:

$$Saída_t = g(S_t)$$



Exemplo Prático de Agente com Memória

- Considere um agente que:
1. Recebe uma pergunta do usuário.
 2. Usa a memória para manter o contexto da conversa.
 3. Decide qual ação tomar (ex.: chamar um modelo de linguagem ou uma API).
 4. Retorna a resposta ao usuário e atualiza a memória.



Prompts Indexação no LangChain



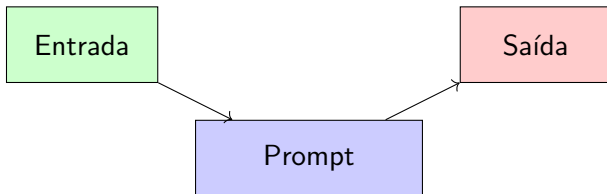
Objetivos do Módulo

- ▶ Compreender o conceito de Prompts no LangChain.
- ▶ Entender como a Indexação e Recuperação de informações funcionam.
- ▶ Explorar a base matemática e computacional por trás dos Prompts.
- ▶ Ver exemplos práticos de Prompts e Indexação em ação.



O que são Prompts no LangChain?

- ▶ Prompts são textos ou instruções que são fornecidos ao modelo de linguagem para guiar sua geração de texto.
- ▶ Podem ser estáticos ou dinâmicos, dependendo do contexto.
- ▶ Exemplos: PromptTemplate, FewShotPromptTemplate.



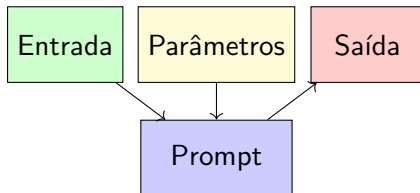
Base dos Prompts

- ▶ Prompts podem ser vistos como uma função f que mapeia uma entrada x para uma saída y :

$$y = f(x, \theta)$$

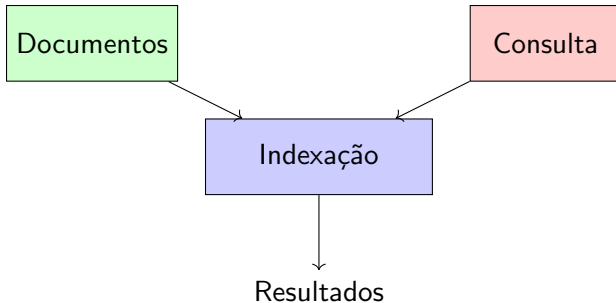
onde:

- ▶ x : Entrada do usuário.
- ▶ θ : Parâmetros do prompt (ex.: instruções, exemplos).
- ▶ y : Saída gerada pelo modelo.
- ▶ Em prompts dinâmicos, θ pode variar com base no contexto.



Indexação e Recuperação no LangChain

- ▶ Indexação é o processo de organizar documentos ou dados para facilitar a recuperação.
- ▶ Recuperação é o processo de buscar informações relevantes com base em uma consulta.
- ▶ Exemplos: VectorStore, Retriever.



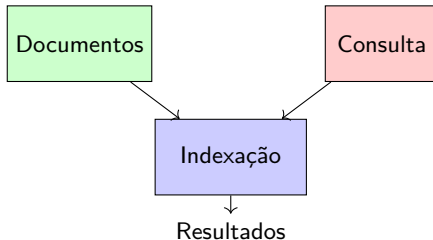
Estrutura da Indexação e Recuperação

- ▶ A indexação pode ser modelada como um problema de similaridade:

$$\text{Similaridade}(q, d) = \frac{q \cdot d}{\|q\| \|d\|}$$

onde:

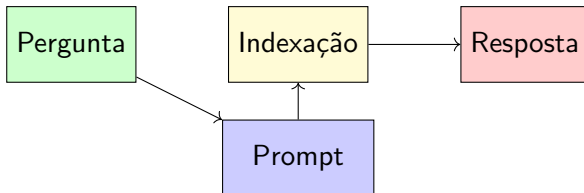
- ▶ q : Vetor de consulta.
- ▶ d : Vetor de documento.
- ▶ A recuperação busca maximizar a similaridade entre a consulta e os documentos.

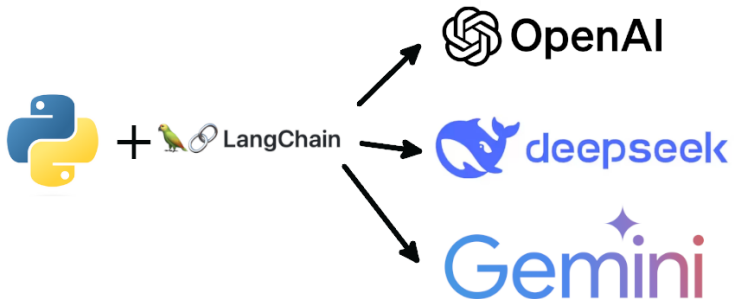


Exemplo Prático de Prompts e Indexação

► Considere um sistema que:

1. Recebe uma pergunta do usuário.
2. Usa um prompt para gerar uma consulta.
3. Recupera documentos relevantes de um índice.
4. Gera uma resposta com base nos documentos recuperados.





Python com LangChain



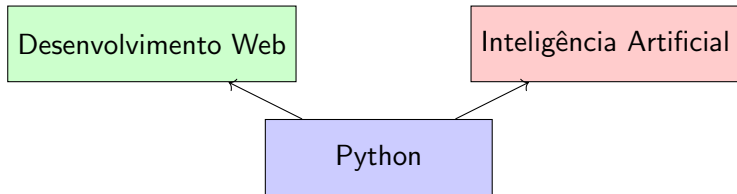
Objetivos

- ▶ Explorar a integração de Python com ferramentas avançadas como OpenAI, LangChain, DeepSeek e Gemini.
- ▶ Fornecer exemplos práticos de uso de Python.



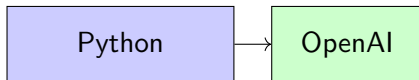
O que é Python?

- ▶ Python é uma linguagem de programação de alto nível, interpretada e de propósito geral.
- ▶ Conhecida por sua sintaxe simples e legibilidade.
- ▶ Amplamente utilizada em áreas como desenvolvimento web, ciência de dados, automação e inteligência artificial.



Integração com OpenAI

- ▶ OpenAI fornece modelos de linguagem avançados como GPT-3 e GPT-4.
- ▶ Python pode ser usado para integrar esses modelos em aplicações.
- ▶ Exemplo: Uso da API da OpenAI para gerar texto.



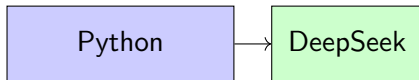
Integração com LangChain

- ▶ LangChain é uma estrutura para construir aplicações com modelos de linguagem.
- ▶ Python é a linguagem principal para trabalhar com LangChain.
- ▶ Exemplo: Criação de cadeias de processamento de linguagem natural.



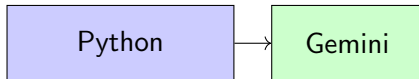
Integração com DeepSeek

- ▶ DeepSeek é uma plataforma para análise de dados e machine learning.
- ▶ Python pode ser usado para integrar DeepSeek em pipelines de dados.
- ▶ Exemplo: Análise de grandes volumes de dados com DeepSeek.



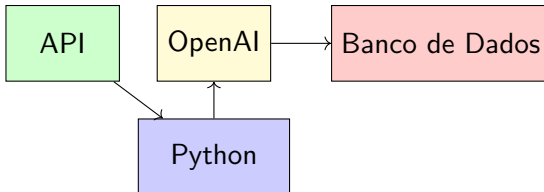
Integração com Gemini

- ▶ Gemini é uma plataforma para desenvolvimento de aplicações blockchain.
- ▶ Python pode ser usado para interagir com a blockchain via Gemini.
- ▶ Exemplo: Criação de smart contracts e transações.



Exemplo Prático de Uso de Python

- ▶ Considere um projeto que:
 1. Coleta dados de uma API.
 2. Processa os dados usando Python.
 3. Integra com OpenAI para análise de texto.
 4. Armazena os resultados em um banco de dados.



RAG



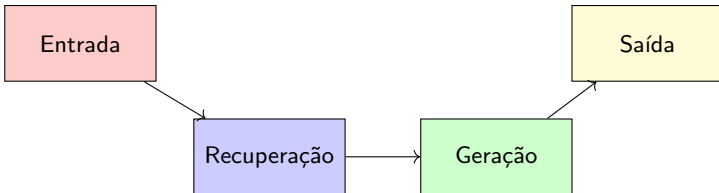
Objetivos do Módulo

- ▶ Compreender o conceito de Retrieval Augmented Generation (RAG).
- ▶ Aprender a implementar RAG usando LangChain.
- ▶ Ver exemplos práticos de RAG em ação.



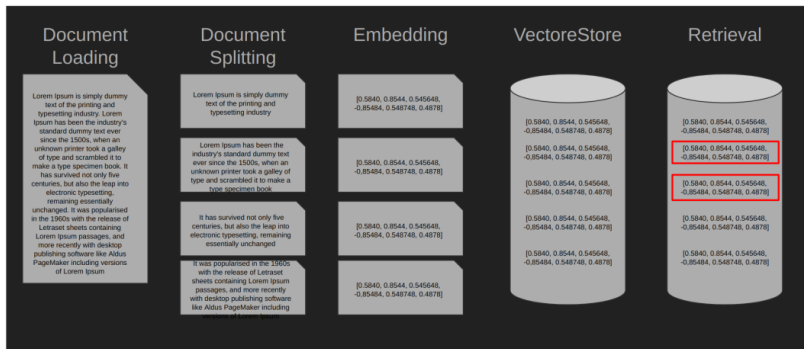
O que é Retrieval Augmented Generation (RAG)?

- ▶ RAG combina recuperação de informações com geração de texto.
- ▶ Usa um sistema de recuperação para buscar documentos (partes) relevantes.
- ▶ Gera respostas com base nos documentos recuperados.



Retrieval Augmented Generation (RAG)

Processo do RAG



Estrutura Matemática do RAG

- ▶ RAG pode ser modelado como:

$$P(y \mid x) = \sum_{d \in D} P(d \mid x) \cdot P(y \mid x, d)$$

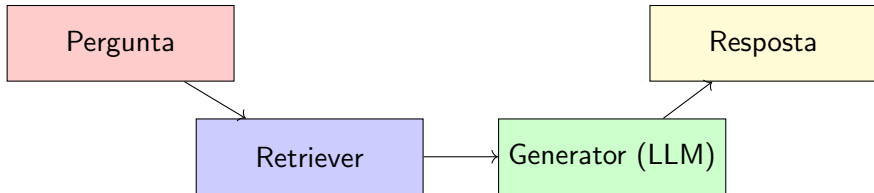
onde:

- ▶ x : Entrada (pergunta).
 - ▶ y : Saída (resposta).
 - ▶ d : Documento recuperado.
 - ▶ D : Conjunto de documentos.
- ▶ $P(d \mid x)$: Probabilidade de recuperar o documento d dado x .
 - ▶ $P(y \mid x, d)$: Probabilidade de gerar a resposta y dado x e d .



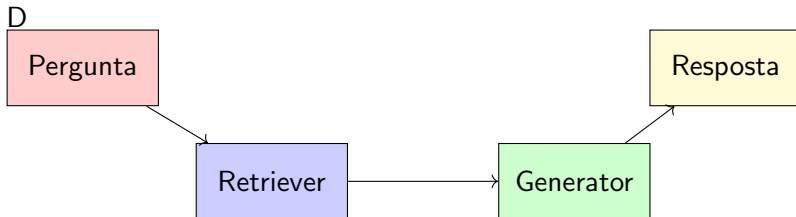
Implementação do RAG no LangChain

- ▶ Usamos um 'Retriever' para buscar documentos relevantes.
- ▶ Um 'Generator' (LLM) gera a resposta com base nos documentos.







Exemplo Prático de RAG

- ▶ Considere um sistema que:
 1. Recebe uma pergunta do usuário.
 2. Recupera documentos relevantes de um índice.
 3. Gera uma resposta com base nos documentos.






Referências I

-  Chollet, F. (2021). *Deep Learning with Python*, Second Edition. Shelter Island, NY: Manning Publications.
-  Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly Media.
-  Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4^a ed.). Hoboken, NJ: Pearson.
-  Moroney, L. (2020). *AI and Machine Learning for Coders*. Sebastopol, CA: O'Reilly Media.



Referências II

-  LangChain Documentation. (2023). *LangChain: Building applications with LLMs*. Disponível em: <https://www.langchain.com/>
-  Vaswani, A., et al. (2017). *Attention Is All You Need*. Disponível em: <https://arxiv.org/abs/1706.03762>.
-  Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.





Introdução ao LangChain

Professor: Elton Sarmanho¹

E-mail: eltonss@ufpa.br



¹Faculdade de Sistemas de Informação - UFPA/CUNTINS

6 de fevereiro de 2025