

FIAP

NABA



COGNITIVE ENVIRONMENTS

DATA SCIENCE & AI MBA

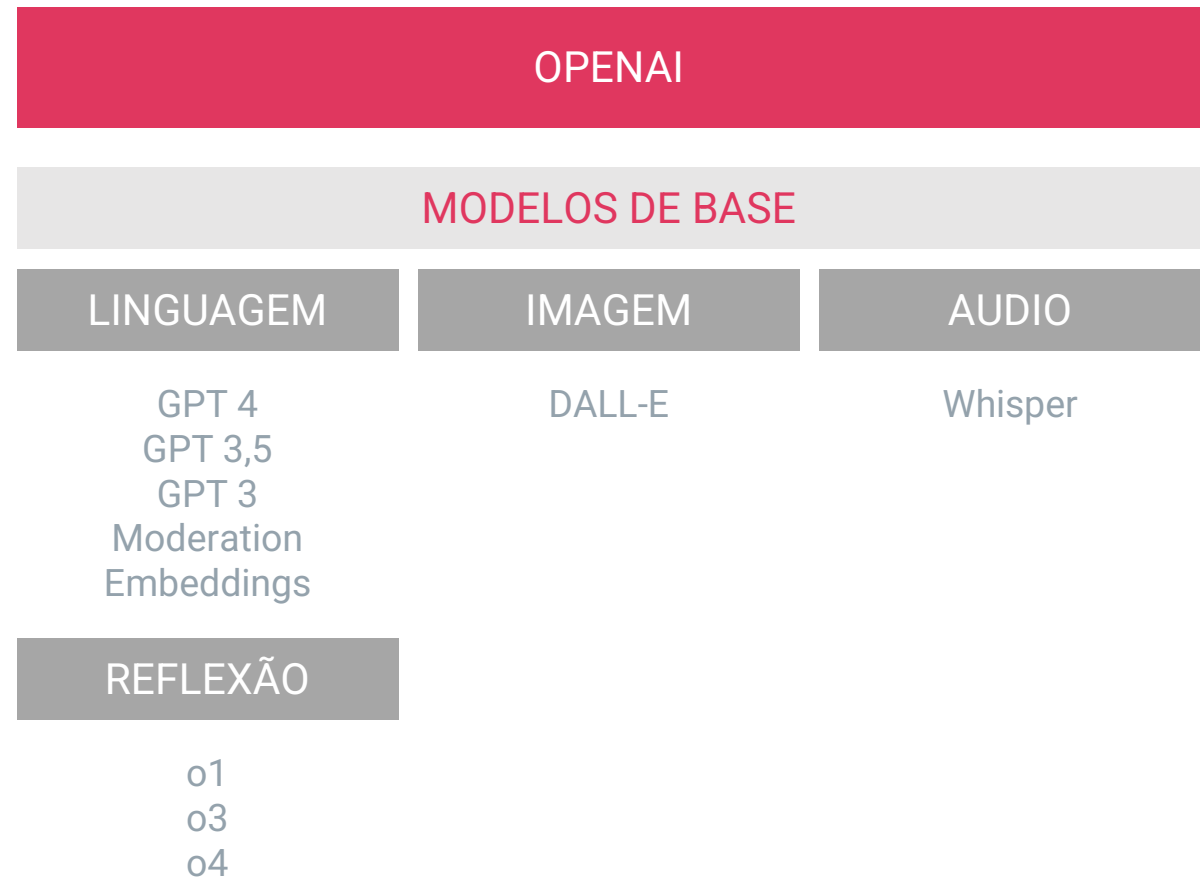


PLATAFORMA OPENAI

Responsável pelo serviço ChatGPT, a OpenAI disponibiliza os modelos de linguagem mais conhecidos da empresa, como o GPT-3.5 e o GPT-4, além de versões mais recentes com capacidades expandidas, como o GPT-4 Turbo, que incorporam mecanismos de raciocínio mais sofisticados, incluindo cadeias de pensamento reflexivas (reflexion models).

Em parceria com a Microsoft, oferece um ambiente de execução privado para empresas através do Azure OpenAI Service, garantindo que os dados corporativos utilizados nas interações não sejam armazenados nem usados para re-treinamento dos modelos.

Além do acesso via interface conversacional, a OpenAI fornece APIs programáticas que permitem a integração de seus modelos em aplicações empresariais, abrangendo não apenas LLMs, mas também modelos multimodais capazes de processar e gerar conteúdo com base em texto, imagem e som.





MODELO GPT

Generative Pre-trained Transformers (GPT) são uma classe de modelos de linguagem em larga escala (Large Language Models ou simplesmente LLMs). Além do GPT da OpenAI, que atualmente inclui versões como GPT-3.5, GPT-4 e GPT-4 Turbo, existem diversos outros modelos relevantes no mercado, incluindo o Gemini (do Google, anteriormente Bard), Claude (Anthropic), Titan (AWS), Llama 3 (Meta) e iniciativas brasileiras como o Maritaca, dentre muitos outros que são lançados frequentemente.

Esses modelos são treinados em grandes volumes de dados, com o GPT-3 contando com cerca de 175 bilhões de parâmetros, enquanto modelos mais recentes como GPT-4 utilizam uma quantidade ainda maior, embora os números exatos não sejam oficialmente divulgados pela OpenAI.

Os dados de treinamento utilizados por esses modelos incluem conteúdos extraídos da internet, como sites, artigos, livros e fontes abertas, como a Wikipedia. Da mesma forma que motores de busca como o Google indexam sites para disponibilizar informações rapidamente aos usuários, os LLMs também coletam e processam grandes volumes de dados online para compor seu treinamento.

O controle sobre quais partes dos sites podem ser acessadas por bots de treinamento de LLMs é feito por meio do arquivo robots.txt, localizado na raiz dos websites. Por exemplo, no caso do GPT da OpenAI, o agente identificado é o GPTBot, enquanto o Google utiliza o agente Google-Extended para seus modelos como o Gemini.

MODELOS DE BASE

Modelo	Descrição	Limite de Tokens	Data de Corte do Treinamento
GPT-3	Terceira geração de modelos de linguagem da OpenAI, com 175 bilhões de parâmetros. Utilizado principalmente para tarefas de completamento de texto.	4.096 tokens	Junho de 2021
GPT-3.5	Versão intermediária entre GPT-3 e GPT-4, com melhorias em compreensão e geração de texto. Inclui o modelo gpt-3.5-turbo.	16.385 tokens	Setembro de 2021
GPT-4	Modelo multimodal com capacidades avançadas de compreensão e geração de texto e imagens.	8.192 tokens	Setembro de 2021
GPT-4 Turbo	Versão otimizada do GPT-4, com maior janela de contexto e custo reduzido.	128.000 tokens	Abril de 2023
GPT-4o	Modelo “omni” com capacidades nativas de entrada e saída de texto, imagem e áudio.	128.000 tokens	Outubro de 2023
GPT-4.1	Última versão lançada, com melhorias significativas em raciocínio e capacidade de lidar com contextos extensos.	1.000.000 tokens	Abril de 2024

MODELOS DE REFLEXÃO AVANÇADA

São variações dos LLMs tradicionais com a capacidade de realizar raciocínio estruturado, avaliando e ajustando suas próprias respostas antes de entregar o resultado final.

Embora não existam modelos especificamente chamados de “modelos de reflexão”, técnicas avançadas como **Chain-of-Thought (CoT)**, **ReAct (Reasoning and Acting)** e **Reflexion** são frequentemente utilizadas em modelos existentes, como **GPT-4** e **GPT-4 Turbo**, ampliando significativamente sua eficiência e precisão em tarefas complexas.

	Descrição	Exemplos de Modelos Compatíveis
Chain-of-Thought (CoT)	Estrutura que força o modelo a gerar explicações passo-a-passo antes de responder, melhorando a capacidade de raciocínio lógico e resolução de problemas.	GPT-3.5 Turbo, GPT-4, GPT-4 Turbo, GPT-4o, GPT-4.1
ReAct	Integra raciocínio e ação, permitindo que o modelo interaja iterativamente com ferramentas externas antes de responder, refletindo sobre suas ações para corrigir trajetórias erradas.	GPT-4, GPT-4 Turbo, GPT-4o, GPT-4.1
Reflexion	Método avançado onde o modelo revisa suas respostas anteriores, refletindo sobre possíveis erros ou inconsistências para aprimorar respostas futuras.	GPT-4, GPT-4 Turbo, GPT-4o, GPT-4.1

O QUE É UM **TOKEN**?

É a granularidade mínima de contagem para determinar tanto a entrada de dados quanto a saída.

Um token não necessariamente representa uma palavra inteira, pois eles podem incluir espaços em branco e também palavras incompletas.

Podem variar para a mesma palavra em diferentes posicionamentos em uma frase.

1 token \sim 4 chars in English
 1 token \sim $\frac{3}{4}$ words
 100 tokens \sim 75 words

1-2 sentence \sim 30 tokens
 1 paragraph \sim 100 tokens
 1,500 words \sim 2048 tokens

GPT-3 Codex

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🍌

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear

Show example

Tokens

64

Characters

252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🍌

Sequences of characters commonly found next to each other may be grouped together: 1234567890

TEXT

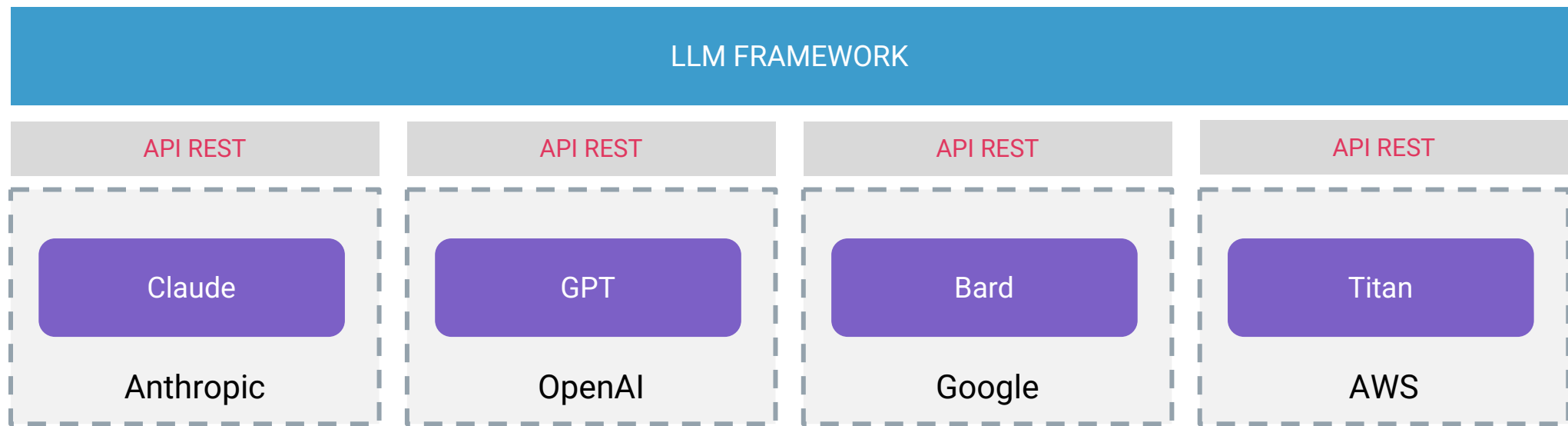
TOKEN IDS



ACESSO PROGRAMÁTICO

Podemos acessar os modelos LLM como GPT de forma direta, pela API da OpenAI ou por um framework especializado.

Utilizando um framework é possível abstrair particularidades de cada modelo de base e realizar testes entre eles mais facilmente.



PROMPT ENGINEERING

Técnica para tirar melhor proveito dos modelos de LLM.

Um dos objetivos é proporcionar a criação de aplicações que utilizem os LLMs de forma mais abstrata, detalhando exatamente o que é necessário.

Apesar de cada demanda pode variar, uma boa prática para construir bons resultados incluem os passos seguintes.

INSTRUÇÃO

Preciso convencer a leitura de livros para um grupo de alunos.

CONTEXTO

Os alunos são crianças de 7 anos que moram em um país que tem o idioma português.

DADOS DE ENTRADA

Considere os 3 livros de maior sucesso no Brasil como referência.

FORMATO DE SAÍDA

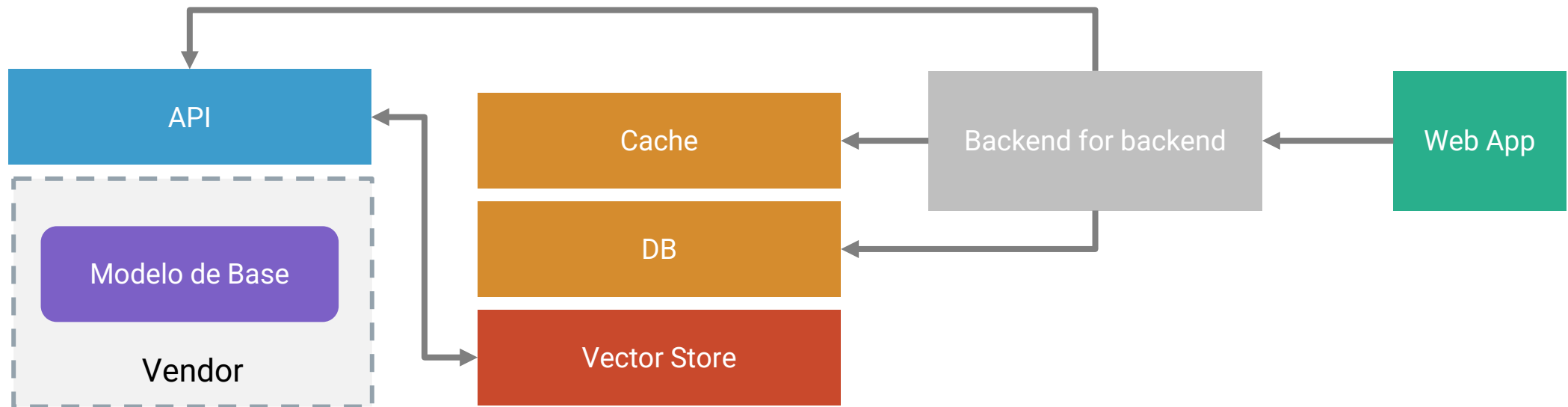
A saída deverá ser em formato JSON, contendo nome do livro e razões.

INTELLIGENT APPS

Técnica para tirar melhor proveito dos modelos de LLM.

Um dos objetivos é proporcionar a criação de aplicações que utilizem os LLMs de forma mais abstrata, detalhando exatamente o que é necessário.

Apesar de cada demanda pode variar, uma boa prática para construir bons resultados incluem os passos seguintes.



UTILIZANDO **API DO OPENAI**

Instalar a biblioteca de acesso a API da OpenAI.

```
!pip install openai
```

Obter uma chave de API da plataforma OpenAI. Novos usuários possuem 30 dias de acesso gratuito.

```
import os  
import openai
```

```
os.environ["OPENAI_API_KEY"] = "chave"
```

```
client = OpenAI(  
    api_key=os.environ.get("OPENAI_API_KEY"),  
)
```

Escolha do modelo.

```
model = "gpt-3.5-turbo"
```

UTILIZANDO API DO OPENAI

Agora vamos definir uma lista de mensagens. Cada mensagem pode ser associada a um papel diferente, como usuário, sistema ou assistente. Esses papéis ajudam a compor as principais partes da engenharia de prompt.

Obter uma chave de API da plataforma OpenAI. Novos usuários possuem 30 dias de acesso gratuito.

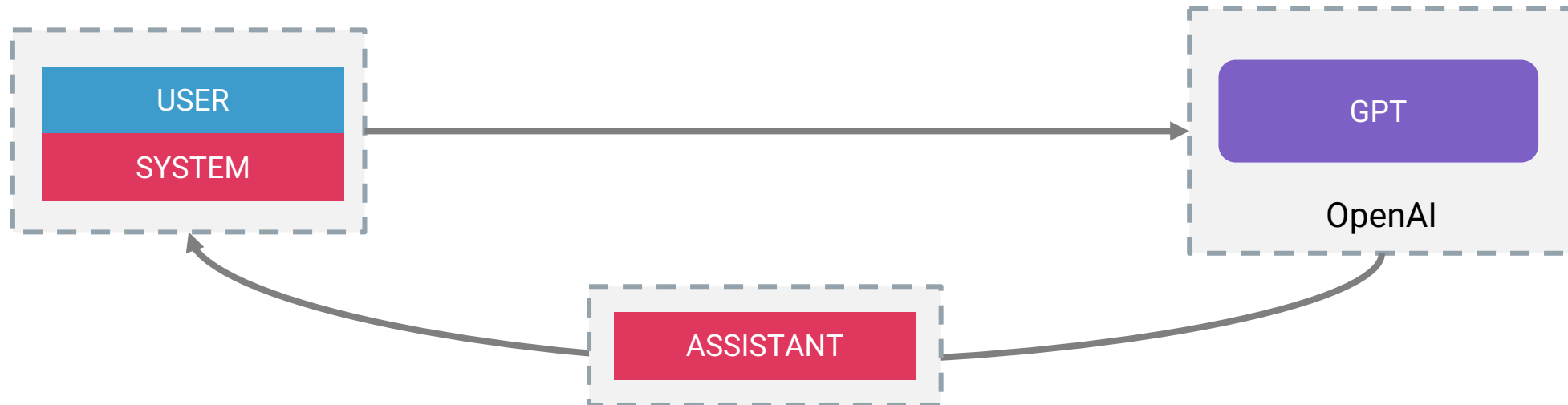
```
messages=[
    {"role": "system", "content": "Você é um professor de visão computacional que leciona para crianças de até 10 anos."},
    {"role": "user", "content": "Como funciona um detector de bordas de Canny, de forma bem resumida?"},
    {"role": "system", "content": "O resultado precisa ser formatado como Markdown."}
]
```

Na execução do prompt, é necessário configurar o modelo, enviar as mensagens (do prompt) e definir o **temperature**. Este último parâmetro está associado a criatividade das respostas, 0 trará os valores com maiores probabilidades (mais determinístico) associados nos embeddings e quanto maior o número mais randômico e diversa será a resposta.

```
response = client.chat.completions.create(
    model=model,
    messages=messages,
    temperature=0,
)
response
```

PAPEIS DA **API DO OPENAI**

- **User**: é o próprio usuário. As interações que partem dele para serem resolvidas (método de entrada).
- **Assistant**: é associada ao GPT da Open AI. Originada pela resposta por meio de um usuário, por exemplo.
- **System**: são complementos que o desenvolvedor pode fornecer para tornar mais assertiva das respostas, como tipo de retorno, limitações na resposta, etc.



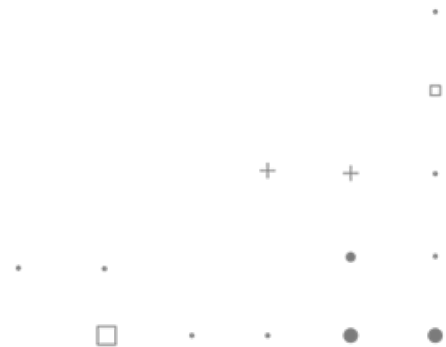
UTILIZANDO API DO OPENAI

Este é o resultado final do prompt enviado.

```
{
  "id": "chatcmpl-7zXMp2J2ni8f1hmnbaim3eRlcYHLp",
  "object": "chat.completion",
  "created": 1694900627,
  "model": "gpt-3.5-turbo-0613",
  "response": {
    "role": "assistant",
    "content": "Um detector de bordas de Canny é um algoritmo usado para identificar as bordas em uma imagem. Ele funciona em várias etapas:\n\n1. Suavização: A imagem é suavizada para reduzir o ruído usando um filtro gaussiano.\n\n2. Gradiente: O gradiente da imagem é calculado para determinar a direção e a magnitude das mudanças de intensidade.\n\n3. Supressão de não-máximos: A supressão de não-máximos é aplicada para afinar as bordas, mantendo apenas os pixels que têm a maior magnitude de gradiente em sua vizinhança.\n\n4. Limiarização: Dois limiares são definidos - um limiar inferior e um limiar superior. Os pixels com magnitude de gradiente acima do limiar superior são considerados bordas fortes, os pixels abaixo do limiar inferior são descartados e os pixels entre os dois limiares são considerados bordas fracas.\n\n5. Conexão de bordas: Os pixels de borda fraca são conectados às bordas fortes para formar bordas contínuas.\n\nO resultado final é uma imagem binária, onde os pixels brancos representam as bordas detectadas."
  },
  "finish_reason": "stop",
  "usage": {
    "prompt_tokens": 64,
    "completion_tokens": 271,
    "total_tokens": 335
  }
}
```

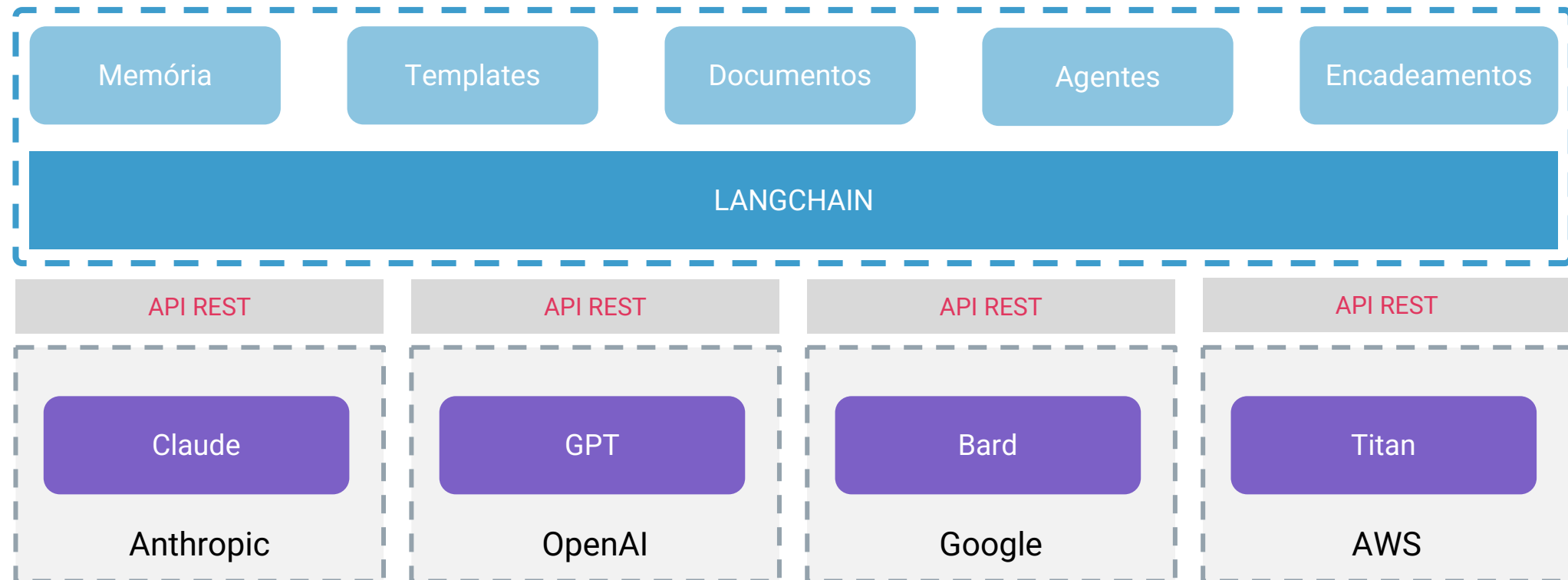
No notebook é possível formatar melhor utilizando o seguinte comando para formatar em Markdown.

```
display_markdown(response.choices[0].message.content)
```



LANGCHAIN: LLM FRAMEWORK

O Langchain é um framework relativamente novo que propõe a abstrair o uso de diferentes modelos de base de LLM ao integrar com cada uma das APIs e permitir que sejam construídos aplicações não dependentes de APIs próprias das plataformas e acrescentar novas camadas neste tipo de uso, tais como memória, encadeamento e agentes.



UTILIZANDO **LANGCHAIN**

O Langchain utiliza a mesma API da OpenAI, por isso, também necessita da chave que utilizamos anteriormente. A forma de obter a chave é por meio da variável de ambiente já definida ("OPENAI_API_KEY").
Instalação das bibliotecas que iremos utilizar.

```
!pip install langchain tiktoken docarray wikipedia xmltodict langchain-openai
```

Para todas as operações do Langchain iremos utilizar o mesmo modelo também definido anteriormente. Em seguida vamos determinar o objeto que representa o modelo de base da OpenAI.

```
openai_chat = ChatOpenAI(temperature=0.0, model=model)  
openai_chat
```

Para executarmos uma simples chamada de prompt basta chamarmos o objeto "openai_chat(mensagem)" onde mensagem é o prompt desejado.

No entanto, vamos utilizar uma forma diferente para tornar o uso dos prompts mais organizado, com o uso de templates.

PROMPT **TEMPLATE**

Template que padroniza o prompt comumente utilizado. A parte dinâmica do prompt é substituído por variáveis, permitindo assim sua reutilização sem ter que escrever o prompt novamente.

```
style = ""  
Tom calmo, sintético e respeitoso, sem incluir nenhuma ofensa ou algo parecido.  
""
```

```
template_string = ""  
Analise a revisão de cliente para que sejam resumidas em no máximo 300 caracteres.  
Utilize o seguinte estilo {style}.  
A revisão é a seguinte: {customer_review}  
""
```

```
from langchain.prompts import ChatPromptTemplate  
  
prompt_template = ChatPromptTemplate.from_template(template_string)
```

```
customer_review = ""  
Comprei um brinquedo que não presta! Utilizei todas as instruções  
mas não funcionada nada! Que droga!  
Se soubesse que fosse assim não teria comprado nessa loja ruim!!!  
""
```



Template

PROMPT **TEMPLATE**

Com o template conseguimos mais flexibilidade ao alterar livremente as variáveis do prompt. No nosso exemplo foram o estilo e a avaliação do cliente.

Também poderíamos ter variações deste template para utilização em traduções, onde a variável poderia ser o idioma desejado.

```
customer_messages = prompt_template.format_messages(  
    style=style,  
    customer_review=customer_review)
```

```
customer_response = openai_chat(customer_messages)  
customer_response.content
```

Comprei um brinquedo que não funcionou mesmo seguindo as instruções. Fiquei decepcionado com a qualidade do produto e não recomendo essa loja.

DESAFIO 1

Revise as avaliações da Amazon utilizando LLM e extraia as seguintes informações: resumo em até 300 caracteres em tom calmo, análise de sentimento (positivo, negativo ou neutro) e uma pontuação de -10 até +10 em relação ao sentimento detectado, sendo -10 mais negativo, 0 neutro e + 10 mais positivo.

Utilize o dataset disponibilizado neste repositório <https://github.com/michelpf/dataset-customer-evaluations>.

MEMÓRIA E CONTEXTO

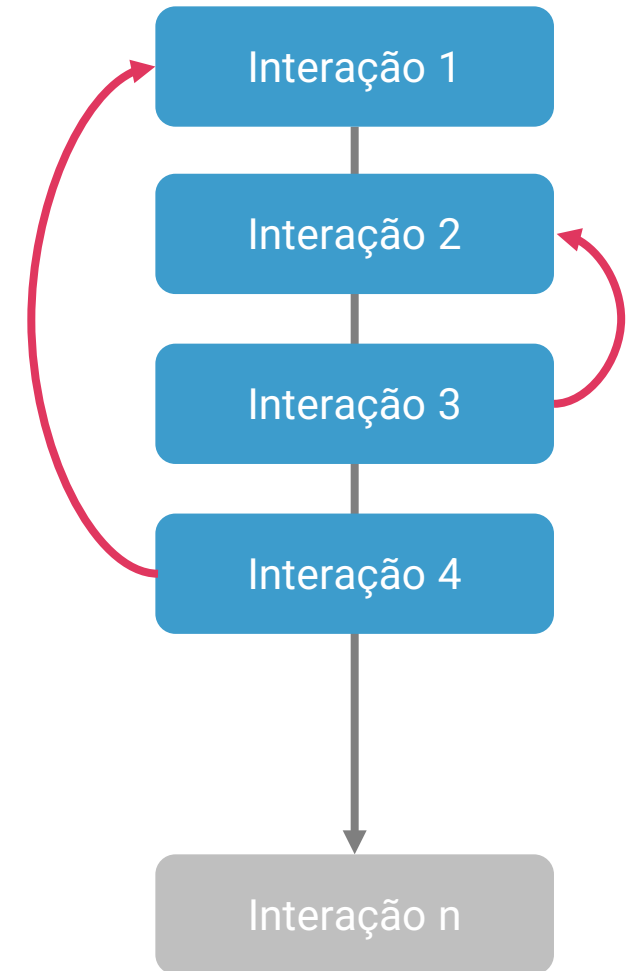
Adicionar memória torna as interações com os modelos LLM contextualizada por certos agrupamentos.

No ChatGPT este contexto é referido a cada conversa iniciada.

Utilizando o Langchain conseguimos obter o mesmo efeito com algumas técnicas diferentes, visando diminuir o número de tokens carregados e garantindo o adequado contexto.

Os tipos mais comuns de memória são:

- Limitado por entrada
- Limitado por tokens
- Limitado por tokens e sumarizado



BUFFER DE CONVERSA

A memória do tipo buffer conversacional não implica em limites controlados pelo Langchain. Todo o contexto armazenado não será excluído, sendo a forma mais simples de configurar memória nas interações.

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

```
chat_openai = ChatOpenAI(temperature=0.0, model=model)
```

```
memory = ConversationBufferMemory()
```

```
conversation = ConversationChain(llm = chat_openai, memory = memory, verbose=True)
```

```
conversation.predict(input="Olá, meu nome é Michel e sou o professor na FIAP")
```

As interações são
armazenadas
automaticamente,

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Olá, meu nome é Michel e sou o professor na FIAP

AI:

> Finished chain.

Olá, Michel! É um prazer conhecê-lo. Como posso ajudá-lo hoje?

BUFFER DE CONVERSA

Note que após as primeiras interações, a memória indica o contexto para a terceira interação levando em consideração as anteriores.

```
conversation.predict(input="Você conhece a FIAP?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.


Current conversation:

Human: Olá, meu nome é Michel e sou o professor na FIAP

AI: Olá, Michel! É um prazer conhecê-lo. Como posso ajudá-lo hoje?

Human: Você conhece a FIAP?

AI:



Memória sendo
construída

> Finished chain.

Sim, eu conheço a FIAP. A FIAP é uma instituição de ensino superior localizada em São Paulo, Brasil. Ela oferece cursos de graduação e pós-graduação nas áreas de tecnologia da informação, negócios e engenharia. A FIAP também é conhecida por suas parcerias com empresas de tecnologia e por promover eventos e competições relacionadas à área.

BUFFER DE CONVERSA

Finalmente a dedução conforme o contexto armazenado,

```
conversation.predict(input="Qual é o meu nome e onde eu trabalho?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Olá, meu nome é Michel e sou o professor na FIAP

AI: Olá, Michel! É um prazer conhecê-lo. Como posso ajudá-lo hoje?

Human: Você conhece a FIAP?

AI: Sim, eu conheço a FIAP. A FIAP é uma instituição de ensino superior localizada em São Paulo, Brasil. Ela oferece cursos de graduação e pós-graduação nas áreas de tecnologia da informação, negócios e engenharia. A FIAP também é conhecida por suas parcerias com empresas de tecnologia e por promover eventos e competições relacionadas à área.

Human: Qual é o meu nome e onde eu trabalho?

AI:

> Finished chain.

Seu nome é Michel e você trabalha na FIAP como professor.

BUFFER DE CONVERSA

Adicionando contextos específicos.

```
memory.save_context({"input": "Eu tenho dois filhos, uma de 10 anos e um de 6 anos."},  
                    {"output": "Legal, então você tem um casal."})
```

```
conversation.predict(input="Em que ano meus filhos nasceram, sabendo que estamos em 2023?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Olá, meu nome é Michel e sou o professor na FIAP

AI: Olá, Michel! É um prazer conhecê-lo. Como posso ajudá-lo hoje?

Human: Você conhece a FIAP?

AI: Sim, eu conheço a FIAP. A FIAP é uma instituição de ensino superior localizada em São Paulo, Brasil. Ela oferece cursos de graduação e pós-graduação nas áreas de tecnologia da informação, negócios e engenharia. A FIAP também é conhecida por suas parcerias com empresas de tecnologia e por promover eventos e competições relacionadas à área.

Human: Qual é o meu nome e onde eu trabalho?

AI: Seu nome é Michel e você trabalha na FIAP como professor.

Human: Eu tenho dois filhos, uma de 10 anos e um de 6 anos.

AI: Legal, então você tem um casal.

Human: Em que ano meus filhos nasceram, sabendo que estamos em 2023?

AI:

> Finished chain.

Se estamos em 2023, sua filha de 10 anos nasceu em 2013 e seu filho de 6 anos nasceu em 2017.

BUFFER DE CONVERSA COM JANELA

O buffer de conversa com janela adiciona uma limitação representado por uma janela de passagem. Essa janela é determinado pela variável k , ou seja, para $k = 1$ somente será persistido apenas 1 único contexto.

```
from langchain.memory import ConversationBufferWindowMemory
```

```
memory_limited = ConversationBufferWindowMemory(k=1)
```

```
memory_limited.save_context({"input": "Estudei Engenharia Elétrica na FEI em São Bernardo do Campo."},  
                             {"output": "Muito bom! É uma faculdade reconhecida nesta área."})
```

```
memory_limited.save_context({"input": "Eu nasci na cidade de São Paulo."},  
                             {"output": "Legal!"})
```

```
memory_limited.load_memory_variables({})
```

```
'history': 'Human: Eu nasci na cidade de São Paulo.\nAI: Legal!'
```

BUFFER DE CONVERSA COM JANELA

Ao interagir solicitando informações fora da janela de memória, o modelo não é capaz de associar o contexto respectivo.

```
conversation = ConversationChain(llm = chat_openai, memory = memory_limited, verbose=True)
```

```
conversation.predict(input="Onde eu me estudei?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Eu nasci na cidade de São Paulo.

AI: Legal!

Human: Onde eu me estudei?

AI:

> Finished chain.

Desculpe, mas eu não tenho informações sobre onde você estudou.

BUFFER DE CONVERSA COM JANELA

Mas quando a informação está presente na janela, a informação é obtida normalmente.

```
conversation.predict(input="Onde eu nasci?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Onde eu me estudei?

AI: Desculpe, mas eu não tenho informações sobre onde você estudou.

Human: Onde eu nasci?

AI:

> Finished chain.

Você nasceu em São Paulo, Brasil.

BUFFER DE CONVERSA COM **TOKENS**

O buffer de memória baseada em tokens utiliza a mesma ideia de limitação por memória. Neste caso a limitação é por token.

```
from langchain.memory import ConversationTokenBufferMemory
```

```
memory_limited = ConversationTokenBufferMemory(llm=chat_openai, max_token_limit=10)
```

```
memory_limited.save_context({"input": "Estudei Engenharia Elétrica na FEI em São Bernardo do Campo."},  
                             {"output": "Muito bom! É uma faculdade reconhecida nesta área."})
```

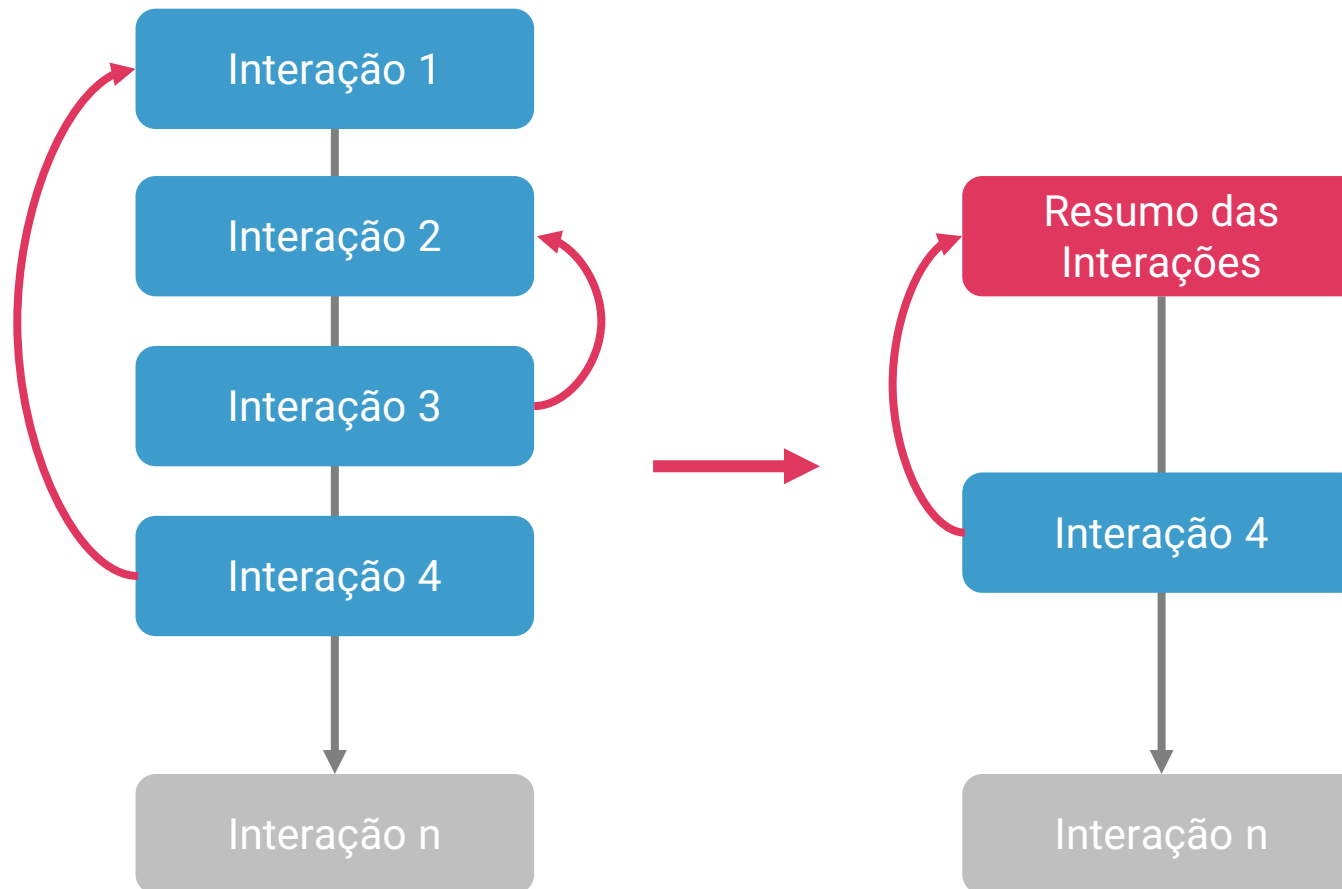
```
memory_limited.save_context({"input": "Eu nasci na cidade de São Paulo."},  
                             {"output": "Legal!"})
```

```
memory_limited.load_memory_variables({})
```

```
{'history': 'AI: Legal!'}
```

BUFFER DE CONVERSA COM **SUMMARY**

Este tipo de buffer combina a limitação por token junto com a sumarização de todo o contexto de conversa existente. Assim o próprio modelo LLM realiza a sumarização do contexto que precisa ter em memória.



BUFFER DE CONVERSA COM SUMMARY

Vamos adicionar algumas informações densas para entendermos como o Langchain se comunicará com o modelo para realizar o resumo das informações de contexto.

```
from langchain.memory import ConversationSummaryBufferMemory
```

```
memory_limited = ConversationSummaryBufferMemory(llm=chat_openai, max_token_limit=100)
```

```
memory_limited.save_context(  
    {"input": "Estudei Engenharia Elétrica na FEI em São Bernardo do Campo, referência na área e que formou muitos profissionais."},  
    {"output": "Muito bom! É uma faculdade reconhecida nesta área."})
```

```
memory_limited.save_context(  
    {"input": "Eu nasci na cidade de São Paulo, capital do Estado. Esta cidade é uma das maiores do Brasil.\nNesta cidade as pessoas costumam ir ao trabalho utilizando transporte público.\nO transporte público mais preferido é o metrô, seguido do trem e do ônibus.\nMuitas pessoas optam por ir de carro, mas as restrições como o rodízio de veículos tornam a necessidade \nde ter mais de um carro ou ir de transporte público em algum dia da semana."},  
    {"output": "Legal!"})
```


BUFFER DE CONVERSA COM SUMMARY

Note que o conteúdo da memória de contexto foi alterada para um resumo de todas as informações lá contidas. Também haverá o limite de contexto a contar das novas interações. Este limite não se aplica ao resumo.

```
memory_limited.load_memory_variables({})
```

```
{'history': 'System: The human mentions studying Electrical Engineering at FEI in São Bernardo do Campo, a renowned institution that has produced many professionals in the field. The AI responds positively, acknowledging that it is a recognized college in this area. The human then shares that they were born in São Paulo, the capital city of the state, which is one of the largest cities in Brazil. In São Paulo, people usually commute to work using public transportation, with the subway being the most preferred mode, followed by trains and buses. While some people choose to drive, restrictions like vehicle rotation make it necessary to either have multiple cars or use public transportation on certain days of the week.\nAI: Legal!'}
```

BUFFER DE CONVERSA COM SUMMARY

A memória utilizada pode também ser visualizada pelo atributo “chat_memory.messages”. Nesta será apresentada as mensagens até o limite de tokens definido.

```
conversation = ConversationChain(llm=chat_openai, memory = memory_limited, verbose=True)
conversation.predict(input="Onde eu me formei?")
```

> Entering new ConversationChain chain...

Prompt after formatting:

The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.

Current conversation:

System: The human mentions studying Electrical Engineering at FEI in São Bernardo do Campo, a renowned institution in the field. The AI acknowledges the recognition of the college. The human then shares that they were born in São Paulo, the capital of the state, which is one of the largest cities in Brazil. They mention that people in the city usually commute using public transportation, with the subway being the most preferred mode, followed by trains and buses. They also mention that many people choose to drive, but restrictions like vehicle rotation make it necessary to have more than one car or use public transportation on certain days of the week.

AI: Legal!

Human: Onde eu me formei?

AI:

> Finished chain.

Você se formou no FEI em São Bernardo do Campo.

```
memory_limited.chat_memory.messages
```

```
[AIMessage(content='Legal!'), HumanMessage(content='Onde eu me formei?'), AIMessage(content='Você se formou no FEI em São Bernardo do Campo.')]
```

UTILIZANDO **DADOS EXTERNOS**

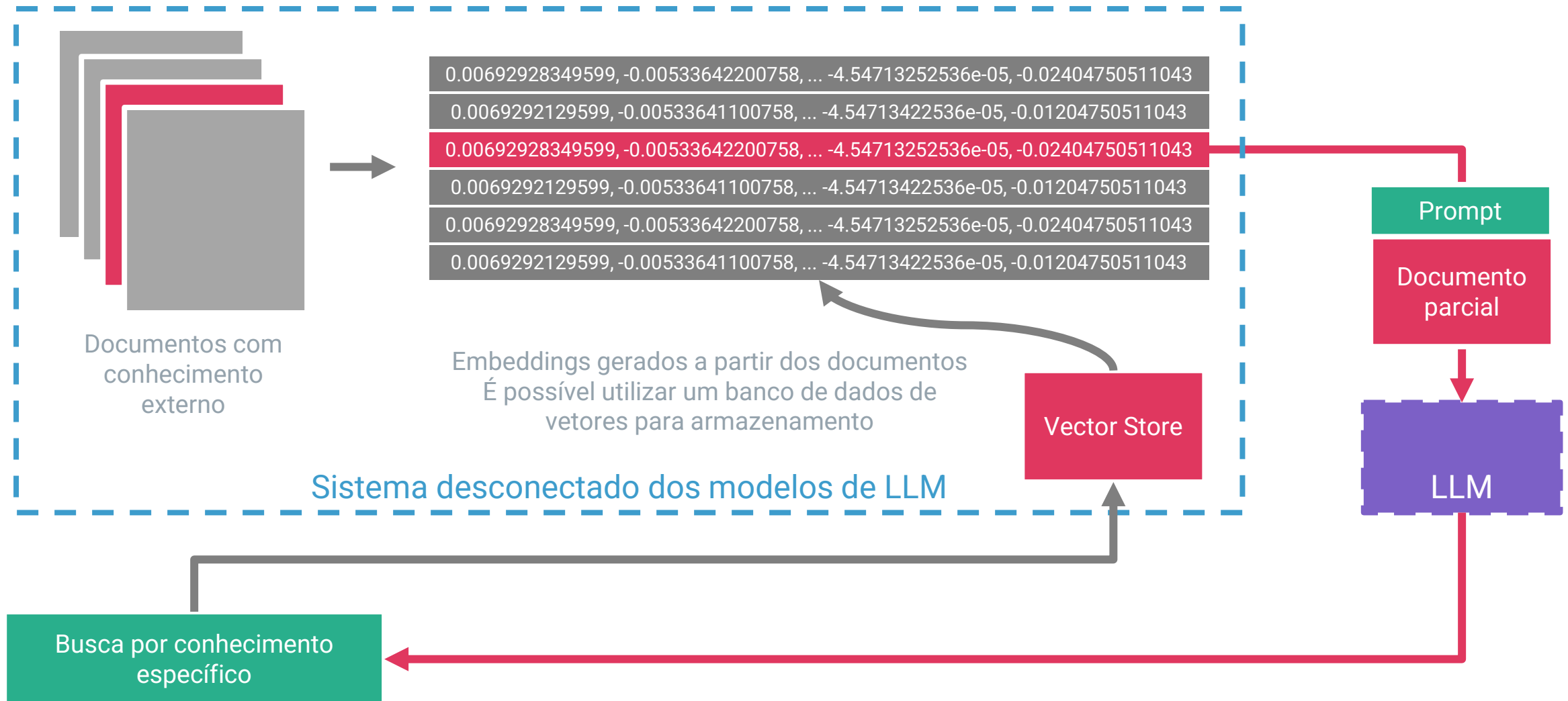
Adicionar memória externa aos modelos LLMs trazem conhecimento especializado da qual não foi originalmente treinado.

Podemos adicionar novos conhecimento basicamente em 2 formas principais. A primeira é treinar mais exemplos de prompt e resposta para gerar um modelo customizado, o que chamamos de fine tuning. Esta forma pode ser custosa, pois depende de muitos exemplos e demanda novo treinamento.

Por outro lado há uma outra estratégia promissora, com uso cada vez mais difundido atualmente que é mapear diferentes formas de representação fora dos modelos para uma pesquisa inicial para então levar aos LLMs contexto relevante para uma resposta ser elaborada.

Ou seja, inicialmente uma pergunta é analisada por algum banco de conhecimento que buscará por informações associadas. Geralmente esses bancos de conhecimento utilizam vetores de representação por meio de embeddings, os mesmos que são utilizados para criar os modelos GPT.

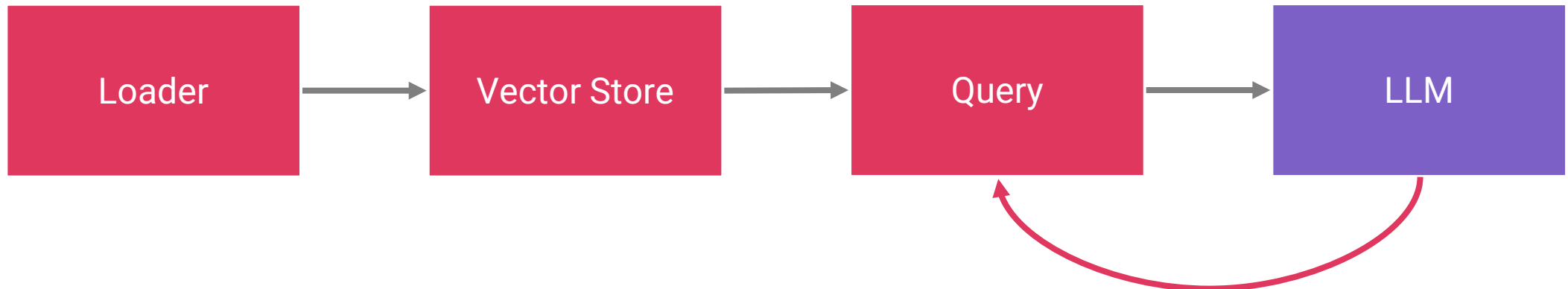
MEMÓRIA COM EMBEDDINGS



CARGA DE DOCUMENTOS

O Langchain vem integrado com uma série de *document loaders*. Os documentos variam de formato, tais como CSV, PDF, JSON, HTML, etc.

Cada tipo de documento é convertido para um embedding, posteriormente salvo em um *vector store* para permitir buscas antes de enviar as partes mais relevantes para o LLM modelar a resposta.



CARGA DE DOCUMENTO **CSV**

Para este exemplo vamos utilizar o arquivo de avaliações do Mercado Livre, disponível neste repositório (<https://github.com/michelpf/dataset-customer-evaluations>).

Para não ficar muito extenso nosso *vector store* vamos filtrar apenas os dados de smartphones.

```
!git clone https://github.com/michelpf/dataset-customer-evaluations
df = pd.read_csv("dataset-customer-evaluations/dataset/ml_scrape_final.csv")
df.head()
```

Pesquisa	Título	Link	Comentario
smartphone	Smartphone Samsung Galaxy A14 Dual 6.6 128gb P...	https://produto.mercadolivre.com.br/MLB-331518...	A foto fica amarelada quando eu vou fotografar...
smartphone	Smartphone Samsung Galaxy A14 Dual 6.6 128gb P...	https://produto.mercadolivre.com.br/MLB-331518...	👏👏👏👏👏.
smartphone	Smartphone Samsung Galaxy A14 Dual 6.6 128gb P...	https://produto.mercadolivre.com.br/MLB-331518...	Muito bom.
smartphone	Smartphone Samsung Galaxy A14 Dual 6.6 128gb P...	https://produto.mercadolivre.com.br/MLB-331518...	Produto muito bom dei de presente pra meu filh...
smartphone	Smartphone Samsung Galaxy A14 Dual 6.6 128gb P...	https://produto.mercadolivre.com.br/MLB-331518...	Recomendo.

```
df.query("Pesquisa == 'smartphone']").to_csv("smartphone_review.csv")
```

CARGA DE DOCUMENTO **CSV**

Como arquivo gerado, vamos agora incorporar os passos para adição de conhecimento externo ao LLM.

```
from langchain.chains import RetrievalQA
from langchain_openai import ChatOpenAI
from langchain.document_loaders import CSVLoader
from langchain.vectorstores import DocArrayInMemorySearch
from langchain.indexes import VectorstoreIndexCreator
from langchain_community.vectorstores import Chroma
```

```
loader = CSVLoader(file_path="smartphone_review.csv")
```

Loader

```
index = VectorstoreIndexCreator(vectorstore_cls=Chroma).from_loaders([loader])
```

Vector Store

```
query = "Liste as 3 piores revisões de clientes, listando com o modelo e o link do produto."
```

Query

```
response = index.query(query, verbose=True)
display_markdown(response)
```

LLM

1. Smartphone Motorola Moto G22 Dual 6,5 128gb 4gb Ram Azul (https://produto.mercadolivre.com.br/MLB-2660980219-smartphone-motorola-moto-g22-dual-65-128gb-4gb-ram-azul-_JM#position=48&search_layout=stack&type=item&tracking_id=4703b69b-1b0f-4cc2-b281-a641f81b3281): "O smartphone é muito lento, realmente se alguém tivesse me dito isso antes não teria comprado.
2. "Smartphone Moto E13 32gb Tela 6.5" 2gb Ram Grafite Motorola (https://produto.mercadolivre.com.br/MLB-3140442935-smartphone-moto-e13-32gb-tela-65-2gb-ram-grafite-motorola-_JM#position=54&search_layout=stack&type=item&tracking_

CARGA DE DOCUMENTO PDF

O processo é muito similar. Desta vez vamos utilizar uma indexação diferente, chamada FAISS, criado pelo Facebook para buscar similaridade em textos.

Precisamos instalar as bibliotecas de leitura de PDF e do FAISS.

```
!pip install pypdf faiss-cpu
```

```
from langchain.vectorstores import FAISS
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.document_loaders import PyPDFLoader
```

```
loader = PyPDFLoader("SM-A146M_Emb_BR_Rev.1.2.pdf")
pages = loader.load_and_split()

faiss_index = FAISS.from_documents(pages, OpenAIEmbeddings())
```

Os embeddings neste caso serão gerados pela OpenAI

```
docs = faiss_index.similarity_search("O que fazer se o aparelho não ligar??", k=2)
for doc in docs:
    print(str(doc.metadata["page"]) + ":", doc.page_content[:300])
```


CARGA DE DOCUMENTO PDF

Somente com os embeddings e a busca por similaridade do FAISS, o resultado foi este:

12: Primeiros passos

13

Ligar ou desligar seu aparelho

Siga todos os avisos e instruções recomendadas pelo pessoal autorizado em áreas onde aparelhos sem fio são proibidos, tais como aviões e hospitais.

Tecla Lateral

Ligar o aparelho

Mantenha pressionada a Tecla Lateral por alguns segundos para ligar o aparelho.

Desligar o aparelho

1 Para desligar o aparelho, mantenha as teclas Lateral e Diminuir volume pressionadas simultaneamente. Como alternativa, abra o painel de notificações e toque em

.

2 Toque em Desligar .

Para reiniciar o aparelho, toque em Reiniciar .

Forçar reinício

Se o seu aparelho estiver travado e sem operação, mantenha as teclas Lateral e Diminuir volume pressionadas simultaneamente por aproximadamente 7 segundos para reiniciá-lo.

Configuração inicial

Ao ligar pela primeira vez ou após executar uma restauração de dados, siga as instruções na tela para configurá-lo.

Se você não se conectar a uma rede Wi-Fi, você não conseguirá definir algumas funções do aparelho durante a configuração inicial....

CARGA DE DOCUMENTO PDF

Agora vamos utilizar o LLM para gerar a resposta.

```
index = VectorstoreIndexCreator(vectorstore_cls=FAISS).from_loaders([loader])
```

```
query = "O que fazer se o aparelho não ligar?"  
response = index.query(query)  
display_markdown(response)
```

Mantenha as teclas Lateral e Diminuir volume pressionadas por mais de 7 segundos para reiniciá-lo.

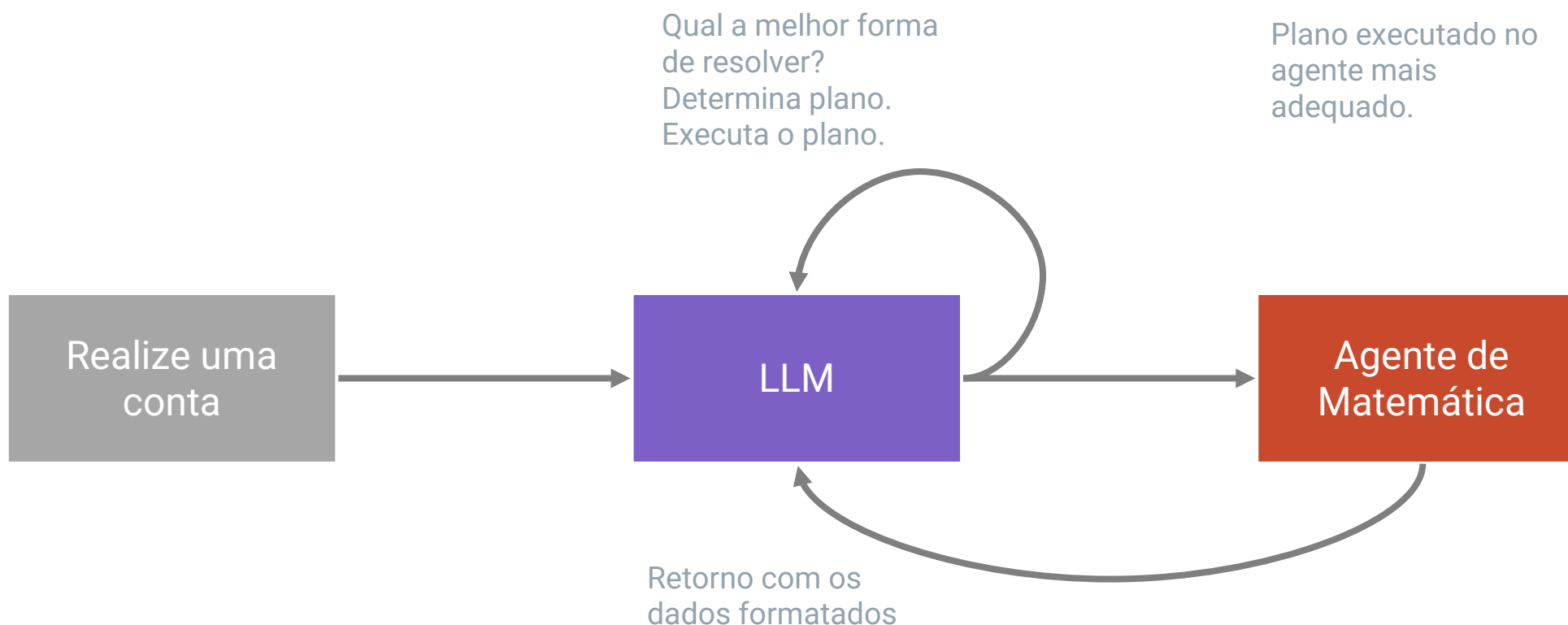
Adicionando as fontes, se for necessário. Assim temos certeza da origem do documento quando for realizado alguma resposta.

```
response = index.query_with_sources(query)
```

```
{'question': 'O que fazer se o aparelho não ligar?', 'answer': ' Se o aparelho não ligar, mantenha as teclas Lateral e Diminuir volume pressionadas por mais de 7 segundos para reiniciá-lo.\n', 'sources': 'SM-A146M_Emb_BR_Rev.1.2.pdf'}
```

O Langchain permite a criação de agentes utilizando os LLM como formas de criar uma planejamento, escolha de ação e refino das respostas obtidas.

Por exemplo, tarefas em que os modelos não são bons ou não adequados, um agente pode suprir potenciais deficiências ou conectar com mais sistemas.



A documentação do Langchain disponibiliza uma série de agentes prontos para interagir com diferentes sistemas e aplicações, tais como:

- Funções customizadas em Python
- Funções math em Python
- Wikipedia
- Busca no Google (via SerAPI)
- PubMed
- Youtube

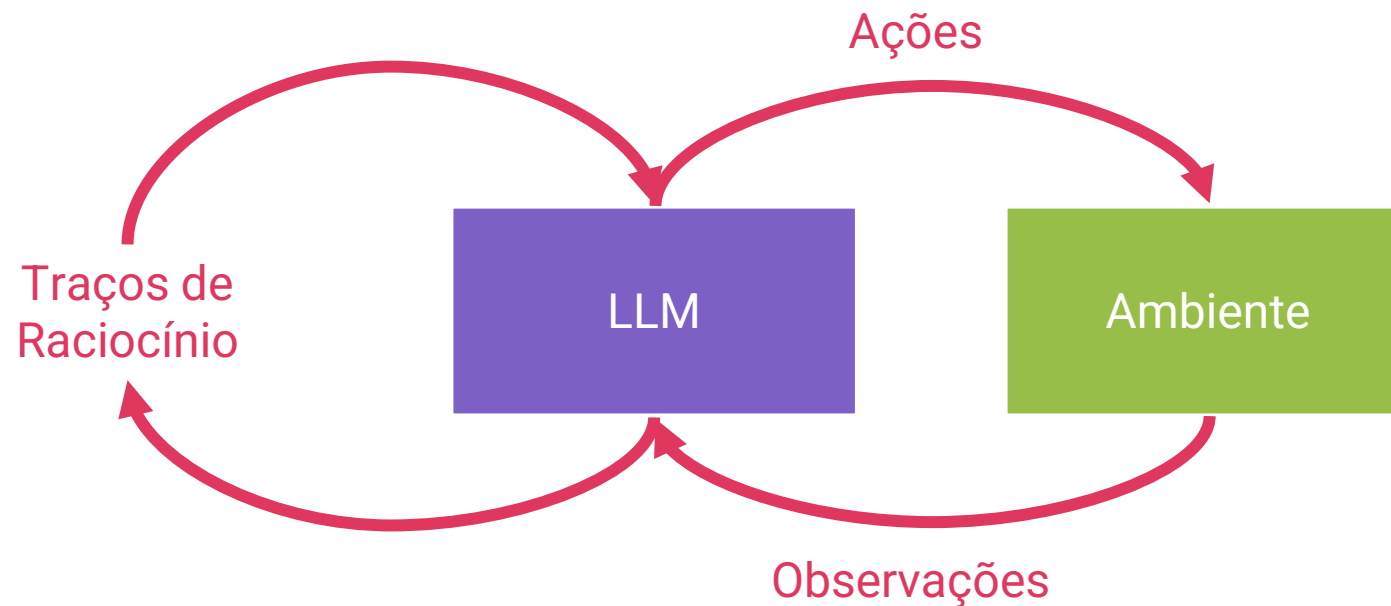
Para interagir com APIs externas basta utilizar um agente baseado em função customizada e realizar as chamadas via biblioteca Requests, por exemplo.

Não é necessário ter uma integração oficial do Langchain para criar seu próprio agente.

UTILIZANDO AGENTES

Os agentes existentes vem pré-configurados com informações que os acionam. Tais informações são descrições bem definidas sobre sua utilização.

A forma de como acionar o agente se baseia em um método chamado Zero Shot React, embora outros também podem ser utilizados.



UTILIZANDO AGENTES

"Who is Olivia Wilde's boyfriend? What is his current age raised to the 0.23 power?"

I need to find out who Olivia Wilde's boyfriend is and then calculate his age raised to the 0.23 power.

Action: Search

Action Input: "Olivia Wilde boyfriend"

Observation: Olivia Wilde started dating Harry Styles after ending her years-long engagement to Jason Sudeikis — see their relationship timeline.

Thought: I need to find out Harry Styles' age.

Action: Search

Action Input: "Harry Styles age"

Observation: 29 years

Thought: I need to calculate 29 raised to the 0.23 power.

Action: Calculator

Action Input: $29^{0.23}$ Observation:

Answer: 2.169459462491557

Thought: I now know the final answer.

Final Answer: Harry Styles, Olivia Wilde's boyfriend, is 29 years old and his age raised to the 0.23 power is 2.169459462491557.

TIPO DE AGENTES

Existem diferentes tipos de agentes que são conectados com seus respectivos prompts. A Langchain criou o LangchainHub para facilitar o reuso de diferentes prompts em aplicações com agentes.

O que diferencia um agente de outro são o uso de “tools”, plug-ins para se conectar com outros serviços, “funções exclusivas”, memória, etc.

Os mais comuns são:

- **ReAct**: <https://smith.langchain.com/hub/hwchase17/react?organizationId=64bbdd11-ef14-5a51-9794-8a07e5df2b5d>
- **JSON Chat** (utiliza o mesmo conceito do ReAct, porém com retorno estruturado em JSON):
<https://smith.langchain.com/hub/hwchase17/react-json?organizationId=64bbdd11-ef14-5a51-9794-8a07e5df2b5d>

UTILIZANDO AGENTE DE MATEMÁTICA

Importação dos módulos para acionar agentes e a instalação do langchain hub para obter prompts para cada tipo de conexão com os agentes. módulo de agente pode ser incluído na função “load-tools”, como por exemplo “wikipedia”.

```
!pip install langchainhub
```

```
from langchain.agents import load_tools
from langchain.agents import AgentType, create_json_chat_agent, create_react_agent, AgentExecutor
from langchain_openai import ChatOpenAI
```

```
from langchain import hub

prompt = hub.pull("hwchase17/react-chat-json")
chat_openai = ChatOpenAI(temperature=0, model=model)
tools = load_tools(["llm-math"], llm=chat_openai)
agent = create_json_chat_agent(
    chat_openai,
    tools,
    prompt)
agent_executor = AgentExecutor(
    agent=agent, tools=tools, verbose=True, handle_parsing_errors=True
)
agent_executor.invoke({"input": "Quanto é raiz quadrada de 5 menos 0,75?"})
```


UTILIZANDO AGENTE DE MATEMÁTICA

A execução do agente.

Entering new AgentExecutor chain...

```
{  
  "action": "Calculator",  
  "action_input": "sqrt(5) - 0.75"  
}Answer: 1.4860679774997898{  
  "action": "Final Answer",  
  "action_input": "The answer is approximately 1.4860679774997898."  
}
```

> Finished chain.

```
{'input': 'Quanto é raiz quadrada de 5 menos 0,75?',  
'output': 'The answer is approximately 1.4860679774997898.'}
```

UTILIZANDO AGENTE DA **WIKIPEDIA**

Utilizar outros módulos seguem a mesma forma. É importante verificar na documentação (em Tools) a forma de implementá-los, pois variam de agente para agente.

Vamos utilizar o ReAct neste exemplo sem o retorno por JSON.

```
tools = load_tools(["wikipedia"], llm=chat_openai)

prompt = hub.pull("hwchase17/react")

agent = create_react_agent(
    chat_openai,
    tools,
    prompt)
agent_executor = AgentExecutor(
    agent=agent, tools=tools, verbose=True, handle_parsing_errors=True
)

agent_executor.invoke({"input": "Quem é o fundador da Gurgel?"})
```

UTILIZANDO AGENTE DA WIKIPEDIA

A execução do agente.

> Entering new AgentExecutor chain...

I need to find out who the founder of Gurgel is.

Action: Wikipedia

Action Input: GurgelPage: Gurgel

Summary: Gurgel Motores (Portuguese pronunciation: [ɡuʁˈʒɛw]) was a Brazilian automobile manufacturer, named after its founder João do Amaral Gurgel. The company was founded in 1969 and first specialised in buggies and off-road vehicles. Early models were fiberglass bodies installed on Volkswagen Beetle chassis and machinery, but VW bodies and chassis were later replaced by a unique solution made of Plasteel, which consists of fiberglass and steel joined, a system patented by Gurgel. Gurgel also introduced Brazil's first fully domestically designed and manufactured car, the BR-800.

Page: Fábio Gurgel

Summary: Fábio Duca Gurgel do Amaral (born 18 January 1970 in Rio de Janeiro, Brazil) is a former Mixed martial arts fighter and 7th degree coral belt Brazilian jiu-jitsu practitioner and coach. One of the best jiu-jitsu competitors of his generation, Gurgel is a 4-time World Jiu-Jitsu Champion (1996–2001), Brazilian National champion and European Open champion. Gurgel is the co-founder of the Alliance Jiu Jitsu team, president of the Professional League of Jiu-Jitsu, and is regarded as one of the top coaches in Brazilian jiu-jitsu.

Page: Rodrigo Gurgel

Summary: Rodrigo Gurgel (born 1955) is a Brazilian literary critic and professor of literature.

Gurgel was the controversial critic of Jabuti Award 2012 who gave very low notes to books released by personalities that year, for which he was called the Jurado "C" ("C" judge). His notes were criticized by Leftist journalists. For 13 years, Gurgel was a pupil of the Brazilian writer Olavo de Carvalho (1947-2022), who taught philosophy at an online course started in 2009. When Olavo died, on January 25, 2022, Gurgel wrote a eulogy to him. Gurgel is among the most important Brazilian Conservative literary critics. The founder of Gurgel is João do Amaral Gurgel.

Final Answer: João do Amaral Gurgel

> Finished chain.

{'input': 'Quem é o fundador da Gurgel?', 'output': 'João do Amaral Gurgel'}

DESAFIO 2

Implemente o agente do PubMed e utilize o prompt para trazer informações e/ou artigos médicos sobre uma doença rara conhecida como “Febre Familiar do Mediterrâneo”.

UTILIZANDO AGENTE **CUSTOMIZADO**

Um agente customizado pode realizar qualquer ação definida por uma função. Esta função precisa ter no mínimo um parâmetro de entrada e retornar algum valor.

Neste exemplo vamos consumir APIs de teste (ou *dummy*) que retornará:

1. Lista de produtos: <https://dummyjson.com/products>
2. Um único produto baseado no seu identificador <https://dummyjson.com/products/1>

```
{'id': 1, 'title': 'iPhone 9', 'description': 'An apple mobile which is nothing like apple', 'price': 549,
'discountPercentage': 12.96, 'rating': 4.69, 'stock': 94, 'brand': 'Apple', 'category': 'smartphones',
'thumbnail': 'https://i.dummyjson.com/data/products/1/thumbnail.jpg', 'images':
['https://i.dummyjson.com/data/products/1/1.jpg', 'https://i.dummyjson.com/data/products/1/2.jpg',
'https://i.dummyjson.com/data/products/1/3.jpg', 'https://i.dummyjson.com/data/products/1/4.jpg',
'https://i.dummyjson.com/data/products/1/thumbnail.jpg']}
```

Formato padrão do retorno de um produto. No caso de lista de produtos é literalmente uma lista de itens como este.

UTILIZANDO AGENTE **CUSTOMIZADO**

Vamos utilizar a biblioteca *requests* para obter os dados da API.

A função deve ter a *decorator* “@tool” (que modifica o comportamento da função para agir como um agente), o parâmetro de entrada com seu tipo como também o tipo de retorno. Após declarar o nome da função, devemos incluir um comentário explicando o que a função faz, semelhante ao *docstring*, este texto ajudará o LLM a identificar quando utilizar este agente.

```
import requests

@tool
def products(text: str) -> str:
    """Retorna lista de produtos de uma loja de
    comércio eletrônico chamada ShopAI.
    Não requer parâmetro de entrada, se necessário informe vazio.
    O resultado será na forma JSON ou dicionário."""

    retorno = requests.get("https://dummyjson.com/products")
    items = json.loads(retorno.content)
    items = items["products"][:5]

    return items
```

UTILIZANDO AGENTE **CUSTOMIZADO**

A mesma ideia com o endpoint da API que retorna apenas um único item baseado no seu identificador.

```
@tool
def product(text: str) -> str:
    """Retorna o produto baseado na sua identificação.
    Ela é numérica, mas deve ser enviado no formato texto e é obrigatória ser passado.
    Esse é um produto do comércio eletrônico chamada ShopAI.
    O resultado será na forma JSON ou dicionário."""

    retorno = requests.get("https://dummyjson.com/products/"+text)
    item = json.loads(retorno.content)

    return item
```

```
prompt = hub.pull("hwchase17/react")
tools = [products, product]
agent = create_react_agent(
    chat_openai,
    tools,
    prompt)
agent_executor = AgentExecutor(
    agent=agent, tools=tools, verbose=True, handle_parsing_errors=True
)
```

UTILIZANDO AGENTE CUSTOMIZADO

```
agent_executor.invoke({"input": "Traga a lista de produtos da loja ShopAI."})
```

Vou usar a função `products` para obter a lista de produtos da loja ShopAI.

```
{  
  "action": "products",  
  "action_input": ""  
}
```

Observation: [{ 'id': 1, 'title': 'iPhone 9', ...

Thought: Could not parse LLM output: I have obtained the list of products from the ShopAI store.

Observation: Invalid or incomplete response

Thought: I will use the `products` function to get the list of products from the ShopAI store.

```
{  
  "action": "products",  
  "action_input": ""  
}
```

Observation: [{ 'id': 1, 'title': 'iPhone 9', ...

Thought: I have obtained the list of products from the ShopAI store. The products are as follows:

1. iPhone 9 - \$549
2. iPhone X - \$899
3. Samsung Universe 9 - \$1249

Final Answer: The list of products from the ShopAI store is iPhone 9, iPhone X, and Samsung Universe 9.

> Finished chain.

```
{ 'input': 'Traga a lista de produtos da loja ShopAI.', 'output': 'The list of products from the ShopAI store is iPhone 9, iPhone X, and Samsung Universe 9.' }
```


UTILIZANDO AGENTE CUSTOMIZADO

```
agent_executor.invoke({"input": "Traga o produto com identificação 1 da loja ShopAI."})
```

> Entering new AgentExecutor chain...

Vou usar a função `product` para obter o produto com identificação 1 da loja ShopAI.

```
{  
  "action": "product",  
  "action_input": "1"  
}
```

Observation: {'id': 1, 'title': 'iPhone 9...'}

Thought: O produto com identificação 1 da loja ShopAI é o iPhone 9. Ele possui as seguintes informações:

- Título: iPhone 9
- Descrição: An apple mobile which is nothing like apple
- Preço: \$549
- Desconto: 12.96%
- Avaliação: 4.69
- Estoque: 94
- Marca: Apple
- Categoria: smartphones
- Imagem: [Link da imagem](https://i.dummyjson.com/data/products/1/thumbnail.jpg)

Final Answer: O produto com identificação 1 da loja ShopAI é o iPhone 9.

> Finished chain.

```
{  
  'input': 'Traga o produto com identificação 1 da loja ShopAI.',  
  'output': 'O produto com identificação 1 da loja ShopAI é o iPhone 9.'  
}
```

PARA SABER **MAIS**

<https://learn.deeplearning.ai/langchain>, Curso sobre Langchain desenvolvido pelo próprio criador

<https://cursos.alura.com.br/ai-generativa-michelpf-1696298768913-p666380>, Trilha de cursos sobre AI Generativa

<https://arxiv.org/pdf/2306.06624>, RestGPT: Connecting Large Language Models with Real-World RESTful APIs

FIAP