**Tutorial básico de Langchain aprenda a desenvolver apps com ChatGPT, LLaMA, e Gemini - Vídeo 0**

**Transcrito por** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell)**.** [**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **para remover esta mensagem.**

Quer ter um tutorial básico sobre LugChain, a biblioteca para se construir LLMs? Fica comigo nesse vídeo. Oi, pessoal, aqui é o Rafa. Hoje um vídeo muito interessante que eu estou trazendo aqui no canal.

É o primeiro vídeo sobre LugChain, uma biblioteca super poderosa para você que está interessado em construir bibliotecas e aplicações de LLMs para resolver problemas da vida real. Se você já tentou se conectar com a OpenAI diretamente com a API da OpenAI, é uma alternativa possível, mas não é a mais viável. E você vai entender nesse vídeo que existe uma alternativa muito melhor chamada LugChain.

Esse passarinho está aparecendo aqui no vídeo para vocês, que é justamente o símbolo dessa biblioteca. Qual é a ideia dessa biblioteca? Esse é um processo open source que foi criado como um framework para ficar entre as aplicações e os modelos LLMs. Agora existem já diversos LLMs.

No começo, há um ano e meio atrás, praticamente a OpenAI reinava ou você a conectava na OpenAI e você não tinha LLM, mas hoje a gente já sabe que a gente já tem alternativas. A gente tem o próprio HuntingFace, a gente tem o Bard, o Gemini, o Yama, o Coherence, uma diversidade de LLMs que estão surgindo, alguns pagos, outros que já são open source, justamente igual a LugChain. E agora eu vou mostrar aqui para vocês, em primeira mão, como é que a gente utiliza a biblioteca LugChain.

Vou dar uma primeira introduzida nos conceitos básicos de como é formada essa biblioteca e nesse primeiro vídeo a gente vai pôr a mão na massa, a gente vai utilizar o código Python justamente para gerar uma primeira interação com a biblioteca. Se vocês gostarem desse modelo de vídeo, vou trazer mais tutoriais desse formato aqui para o canal e a gente vai estudando junto, vai aprendendo junto como se trabalha com a biblioteca LugChain. Então, vamos agora entender a LugChain.

Pessoal, a gente está aqui agora para falar sobre a LugChain, como a gente teve introduzido. Não existe você fazer aplicações profissionais hoje de LLM sem LugChain, justamente pela cola que une diversos componentes. E aqui nesse slide eu vou apresentar para vocês os principais componentes e vou mostrar o que a gente vai fazer no tutorial de hoje.

Esse, óbvio, não é um tutorial expansivo, para ser uma coisa rápida, concisa de introdução à biblioteca, então eu vou me ater apenas a dois pontos principais dele, mas eu vou mostrar aqui para vocês já a ideia de para que serve a biblioteca como um todo. Então a gente pode vir aqui e começar justamente com o que ele tem dentro dele. Ele tem basicamente três grandes pilares.

O primeiro pilar dele são os componentes. Os componentes vão me ajudar a ter mais produtividade na hora de criar aplicações para LLMs. E um dos componentes principais que você vai encontrar aí na literatura, em inglês, é o LLM Wrapper.

O Wrapper, em tradução se fosse literal, é como se fosse uma embalagem. E essas embalagens são uma maneira de você empacotar aplicações LLMs, justamente porque, apesar de a gente fazer o nosso tutorial com a OpenAI, existem já diversos outros LLMs que a gente pode usar. A gente pode usar o Gemini, pode usar a Llama, pode usar o Coherence, pode usar o Running Face.

Tem diversas opções que a gente pode usar, mas ele ajuda a gente a empacotar de maneira abstrata. Até você pensar nos Design Patterns, se você é da área de programação, você pode utilizá-los para criar aplicações mais modularizadas. Além disso, você vai ter aqui os Broken Templates, que a gente vai fazer hoje o uso deles, justamente para facilitar na hora de a gente criar nossas aplicações.

E a gente tem aqui os Index Retrieval. O que são esses Index Retrieval? É para a gente fazer o famoso RAG, o fine-tuning. O que seria isso? É você pegar dados proprietários seus, esses aqui são os seus dados, e você combinar esses seus dados com um LLM para te dar uma outra resposta através de algum prompt.

Então, aqui eu tenho dados proprietários, tenho um LLM, e eu junto essas duas coisas para me dar uma solução. Só que isso aqui, por si só, seria um curso inteiro. Eu não vou falar sobre RAG e Index Retrieval agora.

Esse curso aqui, eu vou focar basicamente, curso não, tutorial, LLM Wrapper e Prompts Templates. São esses dois pontos aqui que a gente vai tratar aqui. A LangChain, além dos components, ela tem outras duas grandes tipos de base para a gente trabalhar, que são as Chains, que a gente vai utilizar hoje, e a gente tem os Agents.

O que são as Chains? As Chains ajudam a gente a conectar componentes. Então, conectar os componentes. Esses componentes que a gente criou aqui.

Como se fosse a cola que juntasse toda essa galera aqui para formar uma aplicação, para a gente conseguir chamá-la de maneira programática. E os Agents, eles permitem a gente fazer interações. Interações no sentido de, por exemplo, a gente pode, inclusive, utilizar os mesmos componentes.

A gente pode utilizar o Wrapper, pode utilizar um prompt, e fazer ele, a partir do resultado do LLM, ele chamar um serviço de e-mail. Eu posso fazer ele acessar uma API minha, interna. Quando eu quero fazer interações com o mail, aí eu vou utilizar um Agente.

Se eu quiser só prompts, LLM puro, eu vou utilizar aqui o LangChain. Essa, basicamente, é a nossa estrutura que a gente vai usar. A gente está falando aqui de, basicamente, temos essas três grandes aplicações aqui.

LangChain, ele conecta componentes à Chain. Os Agentes, eles fazem interações. Nesse tutorial, que é um primeiro tutorial, o que a gente vai fazer? A gente vai criar um LLM Wrapper, para o OpenAI mesmo.

A gente vai introduzir um prompt template, que vai conectar esses dois caras aqui, justamente, a LangChain. Então, a gente está trabalhando esse item, esse item e esse item aqui, da base do LangChain. Então, vamos agora colocar a mão no código.

Pessoal, agora a gente vai começar a botar a mão na massa sobre a biblioteca LangChain. Aqui eu vou criar um ambiente para a gente fazer as primeiras interações, chamar o chadpt via a biblioteca LangChain, pelo primeiro bloco que eu mostrei para vocês, que é o bloco de componentes. Basicamente, é um LLM Wrapper, como se fosse um embrulho, um pacote.

É uma maneira de você empacotar na LLM uma classe do LangChain e ter acesso programático a ela. E o primeiro passo para fazer isso, a gente tem que gerar uma chave de API na própria OpenAI. Vou fazer esse passo a passo aqui com vocês.

Estou aqui no site da OpenAI. Vou vir aqui em Login, sem segredo para ninguém. Loga com sua conta Google ou qualquer outra conta.

Minha aqui já estava logada. Vai escolher aqui API, se você quer um acesso programático. Aqui do lado, você tem as opções, Playgrounds, que você pode testar algumas aplicações no próprio chat da OpenAI sem custo.

Aqui eles oferecem alguma série de serviços, criar os próprios assistentes, fazer fine tuning, mas o que interessa aqui para a gente é a parte de API Keys. Aqui eu já tenho uma chave, mas eu vou até criar uma nova aqui para mostrar para vocês. É só clicar aqui no botão Create a Secret Key.

Claro que depois eu vou destruir essa Secret Key aqui, justamente para que as pessoas não copiaram e tudo mais. Mas é só dar um nome para ela aqui, test2. Vou criar ela aqui.

Já posso copiar essa minha Secret Key aqui. E a partir dessa Secret Key, o que eu vou fazer? Eu vou vir aqui agora. Aqui já tem um arquivo chamado .env e dentro dele já tem a minha Secret Key.

Eu não vou abrir esse arquivo justamente para não ficar gravado aqui. É uma coisa que é privada. Então vou criar um arquivo .env key temp, só para mostrar para vocês como seria esse arquivo, onde eu tenho que colar isso.

E vou colocar aqui uma Open AI underline API underline key igual. Aí eu coloco aqui aquela chave que eu copiei. Lembrando, não adianta copiar essa chave aqui, porque assim, ao acabar o vídeo, eu vou deletar essa chave.

Ela não vai ter nenhuma utilidade mais. Mas é só um detalhe. Vocês vão colar exatamente esse código aqui dentro do arquivo .env Não tem esse temp aqui, justamente porque a gente quer expandir essa variável de ambiente problematicamente, usando o meu Python para isso.

Vamos falar em Python. Boas práticas. A gente sempre cria um novo terminal aqui.

A gente vai criar um ambiente virtual. A ideia do ambiente virtual, se vocês já dominam Python, vocês já sabem, a ideia é a gente isolar a nossa aplicação, as dependências dessa aplicação. A gente rodar uma versão de Python que a gente conhece e saber todas as aplicações que tem nela, nas nossas bibliotecas, as versões que a gente está usando.

E o comando é muito simples para isso. python-env vai criar aqui meu ambiente virtual. Então, quando eu ativar esse ambiente virtual, o Python não vai estar rodando o Python que está aqui no meu SO, ele vai rodar diretamente o Python que eu criei aqui.

Você viu que eu criei aqui essa .env, aqui está todos os arquivos binários, tudo que a gente precisa para fazer o Python rodar aqui na máquina. Então a gente está aqui e o comando para ativar é source venv, que é o nome da pasta, bin, activate. Outra coisa de curiosidade, se vocês abrirem a pasta venv aqui, dentro de bin tem o activate.

Eu estou executando esse comando. O source é para eu não criar o meu processo, para eu manter o processo nesse terminal. E a parte de agora, vocês podem ver que aqui no meu terminal aparece o venv aqui.

O que significa que eu estou rodando o Python aqui dentro. Vou fechar essa pasta aqui, vou até deletar ela, porque a gente não precisa mais. Vamos imaginar que essa venv está aqui dentro, não é ele que eu criei.

E agora para a minha aplicação, o que a gente precisa fazer é criar um arquivo main main.py Vou tirar minha tela daqui para não atrapalhar. Está bem. Sumiu, fica mais fácil da gente fazer o código.

Esquecei, primeiro de tudo a gente tem que instalar as bibliotecas. Então vamos lá. pip install lang chain Vai instalar o lang chain aqui para a gente.

Pode ser que demore um pouquinho. Lembrando que quando eu instalo o lang chain assim, eu estou instalando dentro do meu venv. Ele vai aparecer aqui.

Estou instalando nessa biblioteca aqui. É aqui que ele vai aparecer. Vamos aguardar ele terminar.

Legal, a gente terminou de instalar a lang chain. Agora vamos instalar o OpenAI, já que a gente vai usar o LLM da OpenAI aqui, temos que instalar ela. E por último vamos instalar um pacote chamado python .env justamente para a gente lidar com essas variáveis de ambiente.

A gente coloca essa variável de ambiente aqui no arquivo .env e a gente tem que de alguma maneira falar para o sistema, pegue essa variável que está aí e coloque no sistema com a variável de ambiente. Beleza, já temos todas as nossas bibliotecas. Agora o que a gente vai fazer, simplesmente vai aqui para o nosso arquivo .men e vai escrever a nossa primeira contato, como falei a gente vai fazer um wrapper, a gente vai simplesmente pegar o chat EPT da OpenAI gerar um prompt e fazer ele resolver de maneira problemática o primeiro passo é usar a lang chain e ao longo desse parte desse exemplo a gente vai fazer coisas mais complexas, vou mostrar para vocês os componentes que ajudam a gente no desenvolvimento.

Vamos lá, pronto lang chain LLM import OpenAI OpenAI transcurr from .env import load .env aqui esse .env tem a missão justamente de pegar tudo que está no arquivo .env e mandar ele como variável de ambiente, é isso que ele está fazendo aqui e o OpenAI é simplesmente o LLM aqui nessa primeira exercício a gente vai replicar um comportamento que a gente já tem junto com um chat EPT vou criar uma funçãozinha básica e como primeiro projeto vamos fazer uma coisa bem simples, eu quero que ele me ajude a gerar o nome de gatos vamos supor que a gente vai criar um gatinho novo eu quero que ele me ajude a gerar o nome do gatinho generate underlying cat name, por exemplo e o que essa função vai fazer? a primeira coisa que eu tenho que fazer é instanciar um LLM, tem esse objeto aqui LLM vai ser OpenAI e aqui eu tenho vários parâmetros que eu posso passar, mas o mais importante é a temperatura essa temperatura é o quão aleatório eu quero que o chat EPT seja quando a gente usa ele na interface a gente não consegue dominar esse parâmetro mas se o parâmetro vai de 0 a 1, quanto mais próximo de 0 ele é mais determinístico ele é menos criativo entre aspas e quanto mais próximo de 1 ele é mais aleatório mais caótico e a gente definiu esse aqui justamente para que ele tenha uma variabilidade toda vez que ele chamar ele vai ter uma resposta diferente e aqui, depois que a gente cria o chat EPT a gente pode usar ele como um prompt normal vamos chamar aqui de names LLM e aqui eu posso passar o meu prompt pra ele se eu pensar, você tem um gatinho filhote novo e gostaria de dar um nome legal para ele e de uma lista de cinco possíveis nomes aqui um prompt super simples tarefa só pra gente ver como é que funciona e eu vou dar um return nome e names espero que ele me dê uma lista de nomes aqui só de novo para ele ser das boas práticas do Python if name equal main vamos executar o arquivo direto vamos chamar vamos já dar um print direto mas só vamos chamar a função aqui acho que a gente pode fazer assim direto se concorda um simples, a ideia de novo aqui agora é só replicar o chat EPT pra gente chamar de uma maneira problemática e aí eu vou acrescentando algumas camadas nessa aplicação simplesmente aqui, Python deixa eu limpar aqui pra ficar mais tranquilo estou comandando Python main.py e eu espero que ele me dê a lista dos cinco gatos se não tiver nenhum erro aqui aqui ele me deu algumas informações sobre a biblioteca, ele está falando que essa versão de import ela vai ser depreciada, então eu tenho que dar uma olhada na comunidade quando ela vai mudar, por exemplo aqui agora eu vou ter que usar langchain community o langchain é uma biblioteca que está em consenso de desenvolvimento então é muito comum esses warnings aqui avisando que alguma coisa vai mudar na linguagem então você tem que alterar seu código e pra você instalar o langchain community ele até falou, instala sim e está ao menos um langchain community depois a gente pode fazer as atualizações se der pra parar de mandar esses warnings mas ele finalmente respondeu Simba, Oliver, Luna, Thor e Mia vamos fazer essas pequenas atualizações e a gente não se preocupa em problemas do futuro para de fazer esses warnings mas a nossa primeira missão que é rodar o chat de apt de maneira programática vamos rodar ele mais uma vez só pra gente ver se a gente conseguiu se você gosta das atualizações ele tem mais algum update que a gente precisa fazer aqui ah, recebi o install mas você viu que ele conseguiu responder, ele manteve vamos de novo fazer os updates que ele precisa provavelmente vocês vão precisar fazer vários updates se você já vem nesse vídeo um pouco no futuro como eu falei, o langchain está em desenvolvimento todo dia literalmente ele muda alguma coisa bom, nem vamos mais nos preocupar com esses parâmetros com esses warnings, ele está aqui trabalhando bem e essa é a primeira maneira de usar ele programaticamente lembrando que quando eu estou usando a OpenAI eu tive que colocar a minha chave aqui no .env e cada vez que eu rodo isso ele vai me cobrar alguns centavos de dólar pra isso, mais pra frente eu posso entrar na especificação pessoal, agora o que a gente vai fazer a gente vai dar um outro passo que é justamente começar a usar a tal das chains que dá o nome a biblioteca, a langchain como é que a gente faz para conectar o nosso prompt e ter esses componentes que a gente falou logo na introdução a primeira coisa que a gente vai fazer aqui é importar um objeto para a gente criar alguns templates, como é que a gente vai fazer isso from langchain .prompts import prompt template esse objeto aqui é o que vai deixar a gente criar os nossos primeiros templates justamente pra a gente ao invés de passar algo físico e estático aqui, a gente poder passar variáveis, por exemplo, pra isso daqui ao invés do animal eu posso colocar o tipo de animal eu posso colocar a cor do animal colocar outras características vou estanciar ele aqui vou chamar ele de prompt animalName igual a esse objeto e aqui a primeira coisa que a gente tem que passar pra ele são as variáveis inputs, variables o que eu vou chamar de variável aqui pra ele? eu vou chamar justamente o tipo do animal então eu vou chamar animal type hoje está difícil eu colocaria entre aspas aqui beleza, essa é a primeira variável que eu vou passar pra ele e agora aqui eu vou passar de fato o meu template o meu template pode ser exatamente isso daqui só que agora eu vou colocar as variáveis nele eu vou ter ao invés de gatinho, eu tenho um aqui a gente pode usar a notação de fString, que não é filhote eu gostaria de dar um nome legal pra ele, de mais 5 nomes possíveis beleza tenho aqui agora o meu objeto já criei o meu prompt e agora eu tenho que conectar justamente o meu prompt com o meu LLM, fazer essa conexão e essa conexão é feita através da chain, da cadeia lembra chain? elo de uma corrente lang chain import LLM LLM aí eu vou pegar esse objeto aqui e agora eu vou juntar essas duas coisas não preciso mais desse nome aqui só que eu preciso dar o nome da minha chain animal name chain e vou instanciar esse objeto aqui LLM chain a partir dele o que eu tenho que passar pra ele? o meu LLM, porque eu posso usar outros, aqui a gente está usando a OpenAI eu posso usar o da Llama Gemini, qualquer outro que ela me dê suporte, acredito que o objetivo do projeto é dar suporte para todos os LLMs o prompt aqui, eu vou passar esse prompt que eu falei então a partir desse momento eu juntei o meu LLM com o meu prompt então nota que a partir desse mesmo prompt aqui, eu poderia muito bem criar vários LLMs pra combinar as respostas, utilizar uma análise de custo pra ver se vale a pena usar a versão open source e aqui agora eu tenho que fazer só uma pequena alteração, que eu preciso chamar o meu prompt e passar essa variável aqui pra ele, eu vou chamar aqui de response e vou passar minha LML chain aqui pra ele, e agora eu passo as minhas variáveis através de um dicionário, modelo chave valor, chave animal type e dois pontos, posso passar o valor eu vou fazer o seguinte vou começar uma função, vou colocar aqui na função o parâmetro e vou pegar ela aqui, assim aqui em baixo a gente pode passar o animal que a gente quiser e vamos só fazer o nosso teste de sanidade vamos continuar pedindo um gatinho, e vamos ver o que ele dá aqui de resultado pra manter o mesmo resultado se for tudo certo a única coisa que a gente tem que colocar aqui é como resposta, se não vai dar errado então vou chamar aqui man.py novamente, e vamos ver o que ele vai nos trazer aqui como resposta ele já me devolve aqui agora, como disse só um parciário, tá vendo? A variável da minha foi gatinho, e de novo simba, oliver, luna, nala e tel, nome de gatinho, agora vamos pegar o nome de um... um chupapá da barraca, um avestruz vamos ver o que ele vai fazer pra gente luma, pequeno, pena, nino, casper, nome de avestruz e aqui você pode perceber que eu já tenho mais modularidade, né? não preciso passar só um parâmetro, eu posso passar vários posso passar por exemplo a cor do animal e como é que eu passo aqui? nas minhas input variables eu coloco a cor, e aqui no meu prompt, eu vou ter que passar o parâmetro cor pra ele vou pegar a color que eu passei vamos colocar aqui agora uma arara não sei se tem, sei lá arara marrom vamos ver o que ele dá pra gente eu passei uma arara marrom e aqui ele tá os nomes, kio e tico lula, bento, maya é um prompt bem bobinho mas o que tem que ficar aqui nessa tutorial aqui pra vocês é essa modularidade, se eu quiser criar aplicações realmente com LLMs não dá pra depender da API do OpenAI, ficar batendo a API e chamando lá, isso é como um amador faria isso se você quer usar essa API de maneira profissional, tem que usar o LungChanger é praticamente impossível você desenvolver uma solução de alto nível hoje em dia sem utilizar a LungChanger e ela é justamente a maneira de você conectar tanto os diversos LLMs que existem, quanto criar prompts de maneiras variáveis ah, aqui teve um erro, né? eu coloquei a variável aqui no prompt e coloquei o color animal do tipo type, filhote com a cor vou passar aqui a cola um pouquinho não usei, né? vamos ver como é que fica a diferença aí eu tenho mais um prompt, arara marrom marronzinho amendoim, chocolate, nino você vê que ele conseguiu utilizar essas duas variáveis, isso aqui é uma maravilha, né? pra gente parciar, pegar a resposta, colocar no banco de dados fazer uma aplicação corriqueira, normal sem a LungChanger, isso aqui daria muito trabalho fazer toda essa parte pessoal, esse aqui foi um tutorial bem simples, a minha ideia não é fazer um prompt milagroso, não é fazer uma solução completa, é apresentar a biblioteca pra vocês eu acredito que a maioria de vocês não conhecia ainda a biblioteca, acredito que infelizmente ou felizmente boa parte de vocês ainda não tiveram oportunidade de implementar um LLM em produção, criar uma aplicação com esse, e se você quiser chegar nesse nível, LungChanger é essencial dominar essa biblioteca, trouxe aqui pra vocês essa primeira parte, onde a gente viu o wrapper do OpenAI mostrei pra vocês como se conecta com ela mostrei pra vocês como você cria um prompt, mostrei programaticamente como a gente junta tudo isso numa chain numa cadeia, que é as duas primeiras bases ali da biblioteca LungChanger, o que faltou nesse vídeo? faltou mostrar os agentes, que seriam praticamente uma série a parte, e também o HAG, o Index Retriever que é você utilizar os seus dados e dar contexto pros seus LLMs utilizando os seus dados privados, pra responder perguntas que você tenha interesse espero que tenha ficado claro esse vídeo a ideia dele não é que você saia um expert, é só dar uma pequena introdução de novo, curte, comente e compartilha se inscreve no canal, se vocês gostaram desse vídeo, vou trazer mais tutoriais da MeeChain e até a próxima, tchau tchau

**Transcrito por** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell)**.** [**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **para remover esta mensagem.**