**How to Build a Local AI Agent With Python (Ollama, LangChain & RAG)**

**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.**

Neste vídeo, eu vou mostrar como construir um agente AI local em apenas alguns minutos usando o Python. Vamos usar o Llama, o LangChain e algo chamado ChromaDB para atuar como nossa database de pesquisas vectores porque eu vou mostrar como adicionar geração de retribo de aumento para esta aplicação. Isso significa que podemos retirar informação relevante de algo como um arquivo CSV ou um PDF e trazê-lo para o nosso modelo.

Tudo isso é totalmente grátis, você não precisa de um acounto de OpenAI, você não precisa de um aconte Cloud ou algo assim, você pode fazer tudo isso do seu computador local. Então, eu vou te mostrar como setá-lo tudo. Então, eu vou te mostrar uma demonstração rápida do produto terminado e depois vamos para o tutorial.

Agora, você pode ver na direita da minha tela aqui que eu apenas abri um arquivo CSV. Este arquivo CSV só contém algumas revistas falsas para um restaurante de pizza aleatório. Então, nós temos título, dia, rating e revista.

E você pode ver algo como a melhor pizza da cidade. Aqui está o dia, aqui está o ID, ou, desculpe, o rating da revista de 5. E aí você tem o que é a revista real e tem algumas informações. Agora, o que eu vou te mostrar é como construir um agente de AI aqui que pode ir e procurar revistas relevantes deste documento para responder perguntas sobre o restaurante.

Eu não sei sobre vocês, mas sempre que eu for para um lugar novo, eu sempre procuro as revistas e, tipo, eu estou procurando uma resposta para a minha pergunta particular. Então, isto pode fazer isso para você. Então, por exemplo, talvez eu queira saber, sabe, qual é a qualidade da pizza.

Ok? Bem, o que ele pode fazer é ir para este documento, encontrar as revistas relevantes, que você pode ver, ele tipo, pula aqui e analisa, e aí me dá uma conclusão. No geral, sem mais dados ou contextos, é desafiante dar um score definitivo na pizza baseado somente nas revistas. No entanto, eles sugerem um restaurante com potencialmente espaço para melhoria na apresentação e consistência geral.

Então, aí está, certo? Eu poderia perguntar algo como, são opções veganas? Vamos ver o que isso nos dá. E você pode ver aqui, em conclusão, baseado nas revistas, há pelo menos uma pizza vegana e possivelmente mais opções veganas disponíveis. Ok, legal.

Então, é isso que vamos construir. Isto não vai ser super complicado, vai ser bem rápido. Então, segure-se e deixe-me mostrar-lhe como fazemos isso.

Certo, então temos alguns passos rápidos aqui e então podemos entrar direto no código. Agora, a primeira coisa que vamos precisar é, obviamente, algum tipo de arquivo CSV. Agora, você pode usar qualquer coisa que você quiser e eu vou mostrar-lhe como ajustar este código para o seu próprio exemplo, mas se você quiser acessar este arquivo CSV que eu estou usando, eu vou deixar um link para ele na descrição e, de fato, todo o código vai estar disponível da GitHub.

Então, você pode ir para a GitHub e você pode descarregar este arquivo CSV e simplesmente trazê-lo para um novo arquivo em VSCode. Então, para começar, abra algum tipo de editor de código. Eu estou usando VSCode.

Crie um novo arquivo. Você pode ver que eu tenho um chamado Local AI Agent. Traz o arquivo CSV e então eu também criei este arquivo Requirements.txt que só tem as três coisas que vamos precisar instalar no Python.

Então, vamos começar com isso, instalando nossas dependências do Python e então eu vou mostrar-lhe os próximos passos. Então, o que precisamos fazer é abrir nosso terminal. De novo, eu estou dentro do diretório que eu quero escrever o código para este vídeo e o que eu vou fazer é criar um ambiente virtual.

Então, para fazer isso, eu vou escrever python-mvenv-venv. Se você está no Mac ou Linux, você pode mudar isso para Python 3 e o que isso vai fazer é criar um novo ambiente isolado que podemos instalar várias dependências. Se você não sabe nada sobre ambientes virtuais e quer aprender mais, eu vou deixar um vídeo na tela.

Agora que o ambiente virtual foi criado, precisamos ativá-lo. Para ativá-lo, se você está no Windows, vai ser .slash, o nome do ambiente virtual, slash e então scripts com um capital S e então slash ativar. Quando você escrever isso, você deve ver o nome do ambiente virtual como um prefixo antes da sua linha de comando.

Agora, se você está no Mac ou Linux, então o comando vai ser slash venv e então isso vai ser bin slash ativar, ok? Então é diferente se você está no Windows, é este aqui e se você está no Mac ou Linux, vai ser este aqui e, de novo, eu vou deixar um vídeo na tela que vamos passar mais em detalhe sobre isso. Agora que temos o ambiente virtual ativado, o que vamos fazer é instalar as várias dependências dentro disso. Agora, se você tem este arquivo requirements.txt, então você pode dizer pip install-r e então você pode fazer requirements.txt e isso vai instalar todos os requerimentos para o nosso ambiente virtual.

Porém, se você não tiver o arquivo requirements.txt, você pode apenas escrever eles, então você pode apenas instalar lang-chain, você pode instalar lang-chain-olama e você pode instalar lang-chain-chroma, assim. Ok? Então, nós só precisamos instalar essas dependências para poder usar isso no Python. Então, isso vai levar um segundo, instalar todas essas dependências para nós e, uma vez isso terminado, eu estarei de volta.

Ok, então essas estão instaladas e a próxima coisa que vamos precisar de é algo chamado olama. Agora, olama nos permite operar modelos localmente no nosso próprio computador usando nosso próprio hardware. Então, é por isso que nós podemos fazer tudo localmente aqui, ao invés de ter que usar algo como uma chave API OpenAI.

Então, por favor, vá para esta página, apenas olama.com, se você não já tem este software e simplesmente baixe-o. Uma vez que você baixou, o que você deveria poder fazer é apenas abrir algum tipo de terminal ou pronto de comando e então escreva o comando olama. Se você tiver algum problema com isso, de novo, eu vou colocar outro vídeo na tela que passa através do olama em profundidade e eu vou mostrar para você como instalar isso.

Mas uma vez que nós temos o olama instalado no nosso computador, o que nós vamos fazer é instalar um modelo olama. Agora, olama, de novo, é apenas um software de abertura e nos permite puxar vários modelos para o nosso próprio computador e então os ruar usando nosso próprio hardware. Agora, dependendo do tipo de hardware que você tenha, isso vai dictar os modelos que você vai poder ruar.

Por exemplo, você provavelmente não pode ruar um modelo de 200 GB se você não tiver um cartão gráfico no seu computador. Então, eu vou te mostrar alguns modelos que devem funcionar em mais máquinas se você tiver um cartão gráfico. Se você não tem um cartão gráfico e você só tem um CPU, há alguns modelos muito pequenos que você pode baixar e usar, mas obviamente a performança não será tão boa.

Então, o que você pode fazer é ir na biblioteca olama. Eu vou deixar esse link na descrição e você pode ver que há vários modelos diferentes e isso te mostra todas as opções que eles têm. Agora, nós vamos puxar dois modelos para o nosso computador.

Nós vamos puxar o olama 3.2. Então, este é um modelo mais pequeno que podemos usar e que funciona bem. E então, nós vamos puxar um modelo de embedição e eu vou te mostrar o nome disso em um segundo, que nós vamos usar para embedir os documentos que nós adicionamos para a nossa loja de vetores. Se isso não significa nada para você, não se preocupe, apenas siga os próximos passos.

Ok, então nós vamos entrar no nosso terminal e, de novo, nós vamos ter certeza que o comando olama funciona. E então, nós vamos escrever olama puxar e nós vamos começar puxando o modelo olama 3.2. Agora, você pode puxar qualquer modelo que você quiser. Você pode escolher, você pode dar uma olhada na direção, mas eu vou puxar o 3.2. Uma vez isso terminado, ok, você pode ver que está aqui porque eu já tinha baixado.

Então, nós podemos ir para o próximo. Agora, o próximo modelo que nós vamos puxar vai ser um modelo de embedição. Agora, este modelo de embedição vai ser MXBAI e então este vai ser "-embed-large".

Existem vários outros modelos de embedição que você pode usar, mas este é o que nós vamos usar para este vídeo. Ok, então nós vamos ir e clicar em entrar e, de novo, baixar para o nosso computador. Estes não são muito grandes, então você deveria poder funcionar eles no seu computador se você tiver qualquer tipo de GPU.

Ok, então agora que nós temos estes modelos, estamos prontos para começar a escrever um pouco de código. Então, eu vou voltar para o VS Code, eu vou fazer um arquivo chamado main.py e neste arquivo eu vou começar a escrever um pouco de código. Agora, você vai notar que eu realmente recebo este autocompleto aqui.

Isto vem do GitHub Copilot. Você sabe, aquele assistente muito legal que replaça muito do seu trabalho de escritura manual. Eles na verdade forneceram este vídeo.

E falando de Microsoft's GitHub Copilot, eu fui fortunado o suficiente para ter eles fornecer um vídeo há algumas semanas atrás sobre agentes de AI e este vídeo em que eu prometo destacar algumas das maneiras em que os desenvolvedores estão usando o GitHub Copilot que vocês publicaram com o hashtag CodingWithCopilot. Então, vamos lá. Veja estes exemplos de como os desenvolvedores estão usando o GitHub Copilot.

Como Emery, que criou uma aplicação móvel para o Flutter inteiro. Tugdool, que criou um script de Python para resizar e salvar imagens. Adrian, que usou o Copilot como um iniciante quando ele estava trabalhando no Jupyter para aprender melhores maneiras de escrever funções.

E Yusif, que usa o Copilot para evitar escrever documentação tediosa manualmente. Agora, pessoalmente, eu uso o GitHub Copilot a cada vez que eu abro o VS Code. E é louco o quão bem ele pode predigir o que eu quero fazer a seguir e me salvar muitas horas de escritura manual.

É literalmente como se ele pudesse ler a minha mente. Agora, eu tenho certeza de que vocês têm mais histórias sobre como vocês estão usando o GitHub Copilot. Então, por favor, compartilhem com mim usando o hashtag CodingWithCopilot, porque estou emocionado para ver eles.

Agora, com isso, vamos voltar ao vídeo. Tudo bem, então, voltando para o editor de código aqui, vamos lá e começar. Agora, vamos começar apenas importando algumas coisas.

Então, vamos dizer, de langchain underscore olama.llms, importe o olama.llm. Então, vamos dizer, de langchain underscore core.prompts, importe o chat.prompt.template. Ok, agora, se você está desconhecido com o LangChain, este é um framework que só faz isso muito mais fácil para a gente trabalhar com LLMs. É muito popular no Python e tem todas essas extensões, como a extensão olama, que nos permite usar diretamente nossos modelos olama. E, a propósito, o que vai acontecer é que o olama deveria estar funcionando no fundo do seu computador, e vai expor um servidor, ou tipo um HttpRest API, que vamos poder comunicar com o nosso programa.

Então, quando você pula esses modelos, eles estão realmente funcionando no seu próprio computador, e podemos ativar o olama para utilizar esses modelos de código no Python, ou podemos fazer isso diretamente da linha de comando. Então, tudo que eu estou mostrando aqui vai funcionar 100% localmente no seu próprio computador, mesmo que isso não necessariamente pareça assim, mas também significa que vai ser bem rápido. Ok, então, depois disso, vamos especificar o nosso modelo.

Agora, eu vou mostrar para você, neste snippet de código, como utilizar um modelo olama, tipo, bem rapidamente, e então vamos começar a conectar mais complexidade a ele com o database de vetores, e eu vou falar sobre o que isso significa. Então, eu vou dizer que o modelo é igual a olama, e então, dentro disso, eu preciso especificar o modelo específico do olama que eu quero usar. Agora, se você está confuso sobre o que colocar aqui, você pode abrir seu código de comando, você pode escrever uma lista olama, como esta, e ela vai mostrar os modelos que você tem disponível.

Então, você pode ver que eu tenho este modelo de embedição, eu tenho loma 3.2, eu tenho mistral, eu tenho loma 2, então qualquer um destes modelos eu posso usar. Então, o que eu vou fazer é apenas copiar loma 3.2. Você não precisa da parte mais recente, você pode fazer apenas o nome original, e você pode colocá-lo aqui. Ok? Então, eu vou usar o modelo olama, modelo igual a loma 3.2, e agora eu posso começar a utilizar este modelo e, tipo, invocar-o.

Então, o que nós vamos fazer agora é criar um template, e este template vai ser apenas uma linha, e dentro desta linha, nós vamos especificar o que nós queremos que o modelo faça. Então, nós vamos dizer algo como, você é um experto em responder perguntas sobre um restaurante de pizza. Aqui estão algumas revistas relevantes, e então nós vamos colocar dentro de uma variável aqui, revistas, e vamos dizer que aqui é a pergunta para ser respondida.

Ok? E então nós vamos colocar uma pergunta. Perfeito. Então, o que nós vamos fazer é dizer que o nosso prompt é igual a um chat prompt template.

Nós vamos passar o nosso template e, na verdade, nós não precisamos passar o modelo, eu não sei porque ele está fazendo isso, e agora nós criamos um chat prompt template, onde nós vamos poder passar uma variável de revistas e uma variável de perguntas, e então o modelo pode responder a isso. Ok? Então, nós vamos criar uma linha. Então, com a linha, nós podemos dizer prompt, e então nós podemos colocar uma pipa, e então nós podemos colocar modelo.

Agora, o que isso nos permite fazer é, essencialmente, invocar essa linha inteira, que pode combinar múltiplas coisas juntas para gerar o nosso LLM. Então, primeiro, o que nós vamos fazer é passar revistas e perguntas de variáveis nesse prompt, esse chat prompt template que nós criamos, e então isso será automaticamente passado para o nosso modelo, porque nós colocamos ele dentro dessa linha, e então ele vai nos retornar o que for a resposta. Então, se nós queremos testar isso muito rapidamente, porque isso é literalmente tudo que precisamos para fazer isso, nós podemos dizer chain.invoke, e então, dentro de um dicionário Python, nós precisamos especificar as duas variáveis que nós tivemos dentro desse prompt.

Então, nós vamos ter revistas, e então perguntas. Ok? Então, nós vamos começar com revistas, e, por enquanto, nós podemos fazer isso em uma lista vazia, e então nós podemos dizer perguntas, e algo como, qual é o melhor lugar de pizza na cidade? Isso não pode necessariamente fazer sentido, porque isso é só sobre um lugar de pizza, mas eu só quero mostrar uma demonstração rápida. Então, nós vamos dizer result é igual a isso, e então nós vamos descer, e nós vamos dizer print result.

Ok? Então, nós podemos testar isso e ter certeza de que está funcionando, e isso deveria ir em frente e invocar o nosso llama.llm e nos dar algum tipo de resposta. Então, vamos aqui e lançar isso. Nós podemos fazer isso através de escrever Python, o nome do nosso arquivo, que é main.py, ou python3 main.py. Então, eu vou clicar em entrar, dar um segundo para lançar, e temos um erro, algum tipo de problema de formato, então vamos ver qual é o problema.

Ok, então um erro estúpido aqui, mas nós precisamos dizer chat.prompttemplate.from.template. Eu esqueci de especificar esse método, então, é claro, isso está nos dando um problema. Então, vamos voltar aqui e fixar isso rapidamente, python main.py, e devemos ver que isso funciona agora, dar um segundo, e você pode ver que isso é baseado no nosso feedback de clientes, e ratings, eu recomendo muito isso, o lugar de pizza de top rating, um revisor mencionou, blá, blá, blá, blá. Na verdade, nosso próprio time sampleou sua pizza.

Então, eu só cheguei com algo rápido aqui, porque eu não dei nenhum revisor, então está meio que alucinando a resposta, mas você tem a ideia, ok? Isso realmente funcionou, nós conseguimos usar o Lama, e nós recebemos uma resposta do modelo, o que é realmente apenas o ponto do que nós estávamos testando aqui. Ok, então agora o que nós vamos fazer é colocar isso dentro de um loop while, então, essencialmente, nós podemos continuar perguntando perguntas, e então nós vamos setar o vector search, para que nós possamos ter uma resposta relevante. Então vamos setar um simples loop aqui, nós vamos dizer while true, então nós vamos perguntar, então nós vamos dizer que a pergunta é igual a input, então nós podemos receber algum input do usuário, e nós vamos dizer, sabe, pergunte a sua pergunta, e então nós vamos colocar um set de parênteses aqui e dizer que queira desistir, então se eles escreveram queira, então nós podemos desistir.

Nós vamos dizer, se a pergunta é igual a queira, então nós vamos interromper, senão nós podemos invocar essa linha. Então nós vamos dizer, resultado é igual a chain.invoke, ok? E para a pergunta, nós vamos colocar a pergunta que o usuário pediu, então nós vamos substituir isso com a pergunta, e então nós podemos imprimir o resultado. E nós também podemos ter alguns tipos de variáveis de formato aqui, então eu vou dizer imprimir, e eu vou imprimir uma grande linha com alguns caracteres backslash-n, e então a mesma coisa aqui, eu vou imprimir alguns backslash-n, para que nós possamos ler o que está acontecendo.

Ok. Então nós não precisamos testar isso, mas isso nos permitirá continuar perguntando até escrevermos um Q. Agora o que eu quero fazer é mostrar para você como instalar o VectorSearch. Ok, então nós vamos criar um novo arquivo aqui chamado vector.py, você pode chamar isso de qualquer coisa que você quiser, e aqui é onde nós vamos escrever a lógica para realmente embedir nossos documentos e depois olhá-los, ou vectorizar nossos documentos.

Agora, caso você esteja desconhecido com o VectorSearch, essencialmente isso vai ser um database, vai ser hostado localmente no nosso próprio computador, usando algo chamado ChromaDB, que instalamos antes, e isso vai nos permitir realmente rapidamente olhar informação relevante, que nós podemos passar para o nosso modelo, e então o nosso modelo pode usar esse dado para nos dar respostas mais contextuais. Então, obviamente, LLMs são muito boas em sintetizar texto e nos dar respostas, mas geralmente eles não têm dados corretos. Então, neste caso, o que nós vamos fazer é que nós vamos pegar este todo arquivo CSV, nós vamos colocá-lo dentro deste database enablando o vector, e então, assim que nós perguntarmos uma questão, nós vamos olhar os documentos relevantes nesse database, nós vamos passar isso para o LLM como uma lista de revistas, e então ele vai ser capaz de procurar essas revistas e responder a nossa pergunta.

Ok? Então isso é a base básica do VectorSearch. Deixe-me mostrar como fazemos isso. Então nós vamos dizer de LengChain underscore Ollama nós vamos importar os embedidinhos Ollama.

Ok? Agora, uma coisa que nós precisamos quando fazemos este processo de vectorização é um modelo de embedidinhos. Este modelo vai ser capaz de pegar texto e converti-lo em um vector. Isto é, essencialmente, números que nós podemos então usar para procurar dados realmente eficazmente.

Próximo, nós vamos dizer de LengChain e então underscore Chroma e nós vamos importar Chroma como isto, que vai ser o nosso VectorStore. Nós vamos então dizer de LengChain underscore Core .Document Importar um documento. Nós vamos criar documentos e depois passar isto para o nosso o que você chama de database Chroma.

Nós vamos então importar os e nós vamos importar algo que eu esqueci de instalar antes, que é Pandas como pd. Ok? Agora, Pandas é uma biblioteca que podemos usar para ler realmente facilmente no nosso arquivo CSV. Então, rapidamente, antes de eu esquecer, nós precisamos instalar isto.

Então, o mesmo que antes, nós vamos escrever PIP, instalar Pandas em nosso ambiente virtual e então nós devemos instalar essa dependência e ser capaz de usá-la. Eu também vou adicionar isto para o arquivo requirements.txt. Então, se vocês tivessem que ter downloadado isto antes, vocês já o teriam. Ok.

Então, Pandas está se instalando. Nós podemos esperar para isso funcionar e começar a escrever mais código. Então, primeiro de tudo, nós vamos carregar o nosso arquivo CSV.

Nós vamos usar o dados do arquivo CSV para o nosso VectorStore, então, é claro, nós vamos precisar dos dados. Então, nós vamos dizer df, que significa DataFrame, e isto vai ser pd.read underscore CSV e nós vamos ler o realistic underscore restaurant underscore reviews dot csv e, obviamente, lê o nome do seu arquivo CSV. Eu acho que eu escrevi isso corretamente, talvez não, restaurant, vamos ver, você sabe o que, nós podemos fazer isto, renomear, copiar, e então pastar aqui para evitar qualquer espalhamento.

Ok. De qualquer forma, então nós temos o nosso DataFrame aqui. Próximo, nós vamos trazer o modelo de embedição.

Então, nós vamos dizer embedings é igual ao olama embedings e então nós vamos dizer model é igual a e então o nome do modelo que nós instalamos, que é MXBAI dash embed dash large. Ok? Agora, depois disso, nós vamos especificar a localização onde nós queremos armazenar o nosso database de vectores. Então, eu vou dizer .slash e então chroma underscore lang chain e então isto vai ser underscore db.

Você pode chamar isto de qualquer coisa que você quiser, mas isto vai ser apenas um folder onde nós armazenamos o nosso database. Ok? Próximo, depois disso, nós vamos dizer add underscore documents é igual a e então nós vamos dizer .os .path .exist e então a localização do database. Agora, o que eu quero fazer é ver se este database já existe.

Se ele existe, isso significa que eu já performei o processo de converter o arquivo CSV para vectores e adicioná-lo ao database. Se ele não existe, então significa que eu preciso fazer isso. Ok? Então, nós não precisamos continuar fazendo isto a cada vez.

Nós podemos, apenas uma vez, vectorizar o nosso data, e então, uma vez que ele está vectorizado e está no database, nós não precisamos fazer isso de novo. Nós podemos apenas começar a usá-lo. Então, abaixo aqui, eu vou dizer if add documents então, se nós realmente precisarmos adicioná-los, então nós vamos dizer o seguinte, nós vamos dizer documents é igual a uma lista vazia e nós vamos dizer ids é igual a uma lista vazia também.

Então, o que nós vamos fazer é nós vamos iterar através das nossas linhas. Então, nós vamos dizer for i ,row em df .iter rows Isto simplesmente vai ir linha por linha através do nosso arquivo CSV e então nos permite acessar os vários entros. Agora, o que nós vamos fazer é nós vamos criar documentos indivíduos, nós vamos adicioná-los para a lista dos documentos e então nós vamos adicioná-los para a nossa loja de vectores.

Ok? Então, nós vamos dizer document é igual a document e dentro deste documento, nós precisamos passar três coisas. Nós precisamos passar um conteúdo de página e este conteúdo de página vai ser o que nós realmente vamos vectorizar e o que nós vamos olhar. Então, se você quiser ajustar isto para o seu próprio exemplo, qualquer conteúdo que você queira usar para realmente olhar a informação no database, isso precisa ir no conteúdo de página.

Então, o que nós vamos fazer é que nós vamos combinar o título da revista com a revista em si para que nós tenhamos um monte de informação para poder realmente queryar nossos dados. Ok? Existem todos os tipos de coisas diferentes que você pode fazer aqui, mas você quer incluir a informação importante que você vai querer baseado no conteúdo da página. Então, nós vamos pegar row at title e então nós vamos dizer plus um espaço e então row at review.

Ok? Então nós vamos especificar algumas metadatas, o Copilot já está fazendo isso para mim. Então nós temos metadata e então rating e isso é row rating. E então nós vamos ter date e então isso vai ser row date.

Então a metadata é apenas informação adicional que nós vamos pegar junto com o documento, mas nós não vamos querer baseado na metadata. Ok? Então, espero que isso tenha sentido. De novo, apenas dados adicionais que serão incluídos no documento, mas não serão necessariamente usados para escrever.

E então, finalmente, nós podemos especificar um id. Então nós vamos dizer que o id é a linha de i, que é apenas o índice deste valor no row ou no... Como ele chama? O arquivo CSV. E apenas faça com que você converte isso para uma linha.

Ok, então eu acho que isso deve ser bom por agora. Depois disso, o que nós vamos fazer é dizer ids.append e nós vamos apender a linha i. E então nós vamos dizer documents.append e nós vamos apender nosso documento. E a razão pela qual nós precisamos armazenar os ids é porque quando nós realmente criamos esse dado no VectorStore, por algum motivo, nós precisamos de duas listas separadas.

Nós precisamos de uma lista de documentos e então nós precisamos de uma lista de seus ids associados, caso por algum motivo eles sejam diferentes. Então eu sei que parece um pouco estranho que nós tenhamos o id duas vezes, mas apenas siga-nos porque nós precisamos disso para esse processo. Ok, então agora nós já preparamos o dado em documentos e a próxima coisa que nós precisamos fazer é adicioná-lo ao VectorStore.

Então nós precisamos criar o VectorStore. Então nós vamos dizer vector underscore store é igual a chroma e então dentro do chroma nós vamos especificar a localização e o nome da coleção. Então nós vamos dizer que o nome da coleção é igual a restaurant reviews.

Nós vamos dizer que o diretório persistente Copilot está me enganando aqui, vai ser igual a localização db. Agora isso só significa que nós vamos guardá-lo persistentemente em vez de só guardá-lo na memória. Você não precisa fazer isso, mas eu recomendo que você guarde isso permanente para que você não precise continuar regenerando esse databases de chroma.

E então, finalmente, nós precisamos passar a função de embedição, que será igual a nossos embedidos do Olava. Ok? Então nós estamos usando tudo isso localmente, nós temos o chroma db localmente, nós temos o modelo de embedidos locais e agora nós temos o VectorStore. A próxima coisa que eu vou fazer é um if statement e eu vou dizer if add documents então nós vamos dizer VectorStore.add documents e então isso vai ser documents é igual a documentos e ids é igual a ids.

Ok? Então é assim que você adiciona isso, você só diz VectorStore.add documents você especifica os documentos que você quer adicionar que nós já preparamos aqui e então você especifica os ids correspondentes e nós estamos só fazendo isso se isso não já existiu porque se já existiu então nós não precisamos adicionar os documentos, certo? E nós já não teríamos preparado esse dados. Espero que isso tenha sentido, mas isso essencialmente vai criar o VectorStore para nós e automaticamente adicionar os dados a última coisa que nós vamos fazer é fazer esse VectorStore ser usável pelo nosso LLM então eu vou mostrar para você como fazer isso nós vamos dizer Retriever Ok? é igual ao VectorStore.as underscore Retriever Ok? Agora, dentro disso há alguns parâmetros que nós podemos passar por exemplo, nós podemos especificar o número de documentos que nós queremos olhar então eu vou dizer Search keyword arguments é igual a K e então 5 agora quando eu faço isso o que vai acontecer é que vai olhar 5 revistas relevantes e depois passar essas 5 revistas para o LLM agora se nós queríamos 10 revistas nós faríamos isso 10 se nós queríamos 1 revista nós faríamos isso 1 você pode especificar o quanto mais ou menos você quiser obviamente o mínimo de 1 mas eu vou ir com 5 Ok? Então agora nós temos o Retriever e o que esse Retriever vai nos permitir fazer é olhar documentos então nós podemos passar esses documentos para o prompto para o nosso LLM então rapidamente recapitulando nós importamos todos os dados relevantes nós trazemos o .csv nós definimos o modelo de embedding do Olama nós verificamos se esse local já existe se não existe então nós vamos preparar todos os nossos dados convertindo-os em documentos nós vamos inicializar o VectorStore se por algum motivo esse diretório já existe então não há necessidade de adicionar os dados mas se não existe então nós vamos adicionar esses dados para o VectorStore adicionando todos os nossos documentos isso vai automaticamente embedir todos os nossos documentos e adicioná-los para o VectorStore e então nós podemos criar esse Retriever do VectorStore que vai nos permitir pegar documentos relevantes então o último passo é simplesmente usar esse Retriever do nosso .py então nós vamos ir para o .py nós vamos dizer do Vector Import Retriever porque nós vamos importá-lo do outro arquivo e agora antes de realmente invocar essa linha nós podemos usar o Retriever para pegar os dados relevantes e então nós podemos passar os dados como um parâmetro para o nosso prompt ok? então para fazer isso nós vamos dizer os dados são igual ao Retriever e então isso vai ser .invoke nós vamos invocar isso com a nossa pergunta e então nós podemos simplesmente passar os dados que foram retornados aqui para a nossa linha então tudo o que nós fazemos aqui é dizer Retriever .invoke nós passamos a pergunta ou tipo a linha de pesquisa que nós queremos usar para analisar os dados relevantes o que vai acontecer é que o Retriever automaticamente vai embedir essa pergunta vai entrar no Vector Store vai analisar todos os dados relevantes usando um algoritmo vai pegar os 5 dados e vai passar isso para a nossa linha e então nós podemos imprimir o resultado e esperamos que nós tenhamos algo significativo baseado nesses dados então vamos fazer isso agora e esperemos que funcione com Python e então vamos em main.py dê um segundo para funcionar você pode ver que ele cria esse diretório chroma langchain.db vai levar um segundo porque ele precisa embedir todos os nossos documentos e agora nós podemos fazer uma pergunta então eu vou dizer como são as opções veganas se eu posso espelhar algo corretamente que aparentemente eu não posso ok então vamos ver o que temos aqui e você pode ver que ele pula algumas revisões diferentes aqui e diz baseado nas revisões aparentemente as opções do restaurante pizza são uma bagunça misturada na parte positiva alguns revisores falaram sobre as pizzas veganas dizendo que são gemas escondidos ok e até nos diz que documento ele pegou porém não todas as revisões estão brilhando um revisor teve uma experiência vastamente diferente com a opção cheese vegana chamando-o tasteless e diz em geral parece que as opções são hit ou miss mas tem potencial e nos deu um rating 3 de 5 baseado nas 2 revisões positivas dos 4 ok legal também podemos perguntar algo como como é o ambiente ou algo não sei se espelhei isso corretamente mas vamos ver o que diz diz em geral eu diria o ambiente o ambiente e o ambiente bom e o ambiente é muito bom e o ambiente é muito bom e o ambiente é muito bom e

**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.**