**AI Agents - Como criar agentes de IA Generativa com ReACT usando OpenAI, Langchain e Astra DB**

**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.**

Se você está criando aplicações com linha generativa, você já deve ter ouvido falar dos agentes, né? A ideia dos agentes é responder perguntas mais complexas, que demandam um certo raciocínio e utilização, consulta a diferentes origens de dados para chegar a uma resposta melhor para o usuário final. Esse vídeo, então, é o primeiro de uma série que eu vou fazer aqui no YouTube, cobrindo agentes. É um assunto bem extenso, tem muitas coisas acontecendo, muitos conceitos surgindo frequentemente, diversos frameworks sendo criados, a Crew.AI, a LangFlow, o próprio LangChain sempre tendo algumas coisas novas, o LangGraph, por exemplo.

Então, eu vou fazer alguns vídeos cobrindo esses tópicos aqui. Deixa aqui no comentário que vídeo que você tem interesse em saber e se inscreve no canal para ficar sabendo quando eu lançar esses caras. É importante a gente entender os conceitos e, para isso, eu escolhi começar essa série aqui com os React Agents.

Talvez ele não seja o mais indicado para você colocar em produção, ele tem algumas questões que precisam ser consideradas, mas ele ajuda muito a gente a entender o conceito dos agentes, de como ele raciocina, como ele executa as ações, até chegar a uma resposta final. Então, a gente vai começar hoje tratando dos React Agents. Vamos lá.

Então, a ideia do React é que um agente consiga resolver problemas um pouco mais complexos que ele precisa raciocinar para, daí, tomar ações. Então, ele vai pensar no que ele precisa fazer para chegar a uma resposta e, daí, com as ferramentas que ele tem para tomar as ações, ele vai especificar as ações que ele quer que a gente execute para ele e o resultado dessas ações volta para ele, para, daí, ele pensar no próximo passo. Reacting também de reasoning, de raciocínio, e Acting, de acionamento ou de ação, de execução de ações, para chegar a uma determinada resposta.

Bom, o que esse agente aqui vai fazer, então? Ele vai responder perguntas sobre passagens aéreas que eu comprei. Então, eu posso perguntar qual é o número do meu voo, quanto eu paguei por voo, tudo que está relacionado à passagem. Por que eu usei esse exemplo? Porque, aqui, ele vai ter que, primeiro, encontrar qual é a passagem que eu estou me referindo, para, depois, poder encontrar detalhes dessa passagem.

Ele vai ter algumas tools, duas funções aqui para encontrar esses casos e, aí, a gente vai ver como ele raciocina para chegar a uma resposta. Então, nesse caso aqui, quanto eu paguei pela minha passagem para Nova York? Ele está dizendo que eu paguei 1.200 dólares pela passagem para Nova York. Vamos ver aqui como o React resolve essa pergunta.

Então, tem o paper que está publicado no arcado, eu vou deixar o link aqui embaixo. E, aqui, tem alguns exemplos de comparação do benchmark que foi usado para chegar nesse cara. E, aí, tem um prompt que é bem legal, que foi publicado no LangSmith, no site do LangChain, que tem um prompt que a gente vai usar aqui para implementar esse caso.

Esse prompt, então, ele vai ter, no começo, as tools. A gente vai falar um pouco sobre as tools. Que são as ações que o modelo tem para acionar, para resolver uma determinada... para chegar a uma determinada resposta.

E ele vai ter um padrão de pensamentos, ação e entrada da ação, e a observação, que é o resultado da ação. Ou seja, para esse caso da passagem, ele vai falar, bom, ele está pedindo o preço de uma passagem. Então, eu preciso, primeiro, encontrar qual é essa passagem.

Então, essa vai ser a ação. E, aí, vai ser a especificação, por exemplo, do aeroporto de destino. E a observação vai ser o resultado.

Então, todas as passagens que a função, que essa tool, que essa ferramenta vai encontrar. Para daí, ele fala, bom, encontrei todas essas passagens. A passagem que eu quero é essa.

E, aí, a aproximação, então, vai ter. Beleza. Então, me dá o detalhe dessa passagem.

E, aí, tendo o detalhamento de toda aquela passagem, ele vai poder pegar a informação que eu estou pedindo. Nesse caso, o preço. Mas poderia ser qualquer uma dessas outras.

Então, e, aí, todos esses resultados, ele vai colocando nesse agentSquaredPad, que é onde ele vai acumulando o retorno de todas as ações. Então, ele é um padrão muito baseado em prompt. E a gente pode editar esse prompt.

A gente vai até ver aqui no código. O que eu precisei, inclusive, editar e colocar aquele return as to JSON sem comentários. E isso é uma coisa boa desse modelo, que a gente consegue especificar melhor como a gente quer que ele trabalhe as ações.

Porém, como ele é muito dependente do prompt, ele, eventualmente, vai gerar respostas diferentes, que não são necessariamente sempre do mesmo jeito. E elas variam bastante de modelo para modelo. Então, dependendo da versão de modelo que você estiver utilizando, ele vai gerar essas ações de maneira diferente.

Então, é um ponto de atenção para quando você estiver implementando esse caso. Vamos dar uma olhada rápida aqui no código. O código não tem nada muito complexo aqui.

Basicamente, é a API, expondo a API e invocando o agente. Então, eu criei esse FlightAgent, que está todo definido nesse outro código aqui. O que esse cara tem? Tem o prompt, tem basicamente aquele mesmo prompt.

Eu posso acrescentar outras coisas, que nem nesse caso aqui. Eu acrescentei o ID do usuário para que o modelo saiba qual é o ID na hora de especificar uma ação, por exemplo. Eu posso fazer isso.

Poderia fazer de outras maneiras, mas aqui a ideia é mostrar que você pode colocar outras informações dentro do prompt para que o modelo considere elas na hora de gerar as ações. Então, eu vou ter as tools, vou ter os nomes das tools e aí a gente vai especificar aqui. Na hora de tomar uma ação, você precisa escolher entre uma das tools que você tem.

As tools, nesse caso, serão GetSchedulerDeFlights e GetFlightDT, ou seja, o primeiro retorna uma lista, o outro retorna o detalhe de uma passagem específica. Poderia ter mais aqui, poderia ser diferente. O input da ação, então, aqui seria o que vai ser enviado para essa ação.

Então, quais são os parâmetros que a gente vai executar dentro dessa ação? E aqui, eu forcei para ele retornar em JSON sem comentários. E aí, a observação, o resultado da ação. No final, o scratchpad.

Então, a gente vai criar esse prompt substituindo o que a gente já tem, então as tools, os toolnames e o CustomerID, que são variáveis que a gente colocou aqui dentro do prompt. O LLM que a gente está usando aqui vai ser o da OpenAI e aí a gente vai criar o agent aqui encadeando aqui os passos que a gente vai ter. Então, o agent vai ter o input, que é a pergunta do usuário, o scratchpad, que vai toda hora receber o resultado e colocar ali dentro, acrescentar nessa variável o resultado da ação.

O prompt, que a gente definiu ali em cima, o modelo e o ReactSingleInputOutputParser. O ReactSingleInputOutputParser é um objeto do LangChain, um método do LangChain que permite a gente interagir, colocar as variáveis dentro do prompt e poder extrair o resultado para a gente poder usar dentro do código, tá? Então, beleza, quando eu inicializar isso aqui, eu vou fazer um log e aí tem a execução de fato. Então, primeiro eu inicializo o agent e depois eu faço a invocação, que é o que está sendo feito ali na API.

Então, eu estou invocando esse agent. Um ponto importante aqui são as tools. Então, a parte importante aqui do React é que a gente está falando quais as ferramentas que o modelo tem para resolver e eu estou especificando essas duas ferramentas aqui.

Como que a gente cria elas? Nesse caso aqui, eu estou usando o Astra como para armazenar os dados das minhas passagens. Então, vamos dar uma olhada aqui no Astra. Eu tenho dentro do meu banco de dados, esse, apesar do Astra ser um vectorDB, ele não é só um vectorDB, ele é um no-cycle database.

Então, esse é um tipo de dado que não necessariamente eu vou precisar de um embedding. Eu vou fazer vector search. É um dado estruturado aqui que eu posso utilizar.

Então, eu gerei aqui uma massa de dados com as passagens, origem e destino. Então, é essa collection flight tickets que vai ser usada para gerar as respostas que o agente vai poder fazer. Então, quando eu fizer a pergunta, ele vai poder procurar todas as passagens disponíveis para ele utilizar e depois o detalhe de uma passagem quando ele tiver o ID dela.

Para eu criar uma tool, uma das coisas é eu ter que ter uma função e eu vou ter os parâmetros que eu quiser, que eu precisar e eu vou anotar ela com o tool e dizer qual é o esquema de entrada para esse cara. Nesse caso aqui, eu estou especificando quais são os parâmetros de entrada, então o ID do cliente e as condições que eu quero. Nesse caso aqui, eu quero só os códigos dos aeroportos e datas como parâmetros de entrada para essa função.

Então, a ideia é que com essas condições, eu encontre qual é a passagem e retorne uma lista de passagens que eu encontrar para o modelo para o modelo entender qual que é que faz sentido para daí sim ele ir para o próximo passo que seria pegar o detalhe de um ticket. Aí, a segunda função, ela vai ter só o ID do ticket como parâmetro e aí a mesma coisa. Então, eu tenho esse detalhe aqui, ele vai fazer um find one para encontrar a passagem específica que o modelo determinou.

Então, uma vez que eu tenho essas duas funções, eu posso especificar elas aqui dentro da minha lista de tools e elas vão ser usadas pelo agente. Ou seja, no final das contas, o que a gente está dizendo para a gente é você vai receber perguntas e você tem essas duas ferramentas para resolver. O modelo vai dizer para a gente agora eu preciso executar essa ferramenta com esses parâmetros e depois na próxima execução, eu preciso executar agora a função com esses parâmetros até ela chegar na resposta final.

Beleza, então vamos ver na prática como isso funciona. Eu vou rodar aqui a aplicação e vou mandar a mesma pergunta. Beleza, eu mandei a pergunta quanto eu paguei no meu voo para Nova York.

Vamos ver aqui o que está acontecendo. Bom, a primeira coisa que a gente vê é que ele entendeu a primeira coisa que ele tem que fazer é executar a função getScheduleAidFlights com o CustomerID esse aqui com a condição do aeroporto de destino, o JFK. E aí ele pensou, como é que ele chegou nisso? Para encontrar quanto o cliente pagou no seu voo para Nova York, eu preciso identificar qual voo especificamente ele está se referindo.

Isso pode ser feito buscando os voos agendados para esse cliente com o aeroporto de destino sendo um dos aeroportos de Nova York. Quando eu tiver a informação do voo, eu posso pegar os detalhes específicos do voo incluindo o preço pago. Então beleza, daqui ele entendeu que ele precisa executar o ScheduleAidFlights passando o aeroporto de chegada como o JFK.

Beleza, então essa é a entrada que ele passou para aquela função. A função vai ser executada. Esse aqui é o retorno dessa função.

Então aqui é o pensamento dele, agora que eu tenho o ID do voo para esse cliente eu posso buscar o detalhe desse voo para saber quanto ele pagou. Então dado um TicketID, ele vai executar a segunda função. Ele encontrou e aqui dentro foi a observação, o resultado dessa segunda execução.

E aí então ele pensa de novo, bom, encontrei os detalhes incluindo o valor. Como o ID do cliente bate com o que eu determinei, eu não preciso fazer nenhuma checagem e eu posso já retornar o resultado aqui e aí ele fala, você pagou 500 dólares no seu voo para Nova York. O que é legal então de a gente entender do React é que ele raciocina, então ele logar aqui qual é o pensamento dele é a parte do reasoning, é a parte do raciocínio dele de como ele chega na decisão do que ele tem que executar.

E aí ele passa para o código ali, para o Lanchain nesse caso, executar essa tool. O Lanchain executa através daquele ReactInputOutputParser. Ele executa e retorna esse resultado, que é o que a gente está vendo aqui em observation.

Essa observação então é usada na próxima execução do prompt. Então é um prompt que está sempre crescendo e sempre sendo executado em loop até ele chegar em um resultado ou até ele decidir que ele não consegue resolver aquela pergunta do usuário. Bom, esse aqui então é um método que de fato executa o agente.

Então ele fica em loop enquanto o passo que o Lanchain detectar que não é o agent finish, ele continua executando em loop até ele chegar. Então enquanto ele tem um passo desse, ele vai enviar a questão e os passos que já foram executados. Aqui a gente tem um print e aí se o resultado da ação é uma ação, é onde de fato ele executa.

Então a primeira coisa que eu vou encontrar qual é essa tool, são algumas funções auxiliares aqui que a gente tem. O input da função aqui eu vou tirar todos aqueles JSONs ou sujeiras que possa ter nesse retorno. E aqui de fato é onde eu vou executar a função.

E o resultado dessa função vai vir para a observação, para esse observation. E aí eu vou acrescentar ali então que a ação foi executada e o resultado dela está aqui dentro da observation e esse intermediate steps é utilizado na próxima no próxima iteração desse processo. Isso continua até que ele chegue no agent finish.

Quando ele chegou no agent finish, ele vai retornar ali o conteúdo para a API que retorna então ali para o Postman. Esse é um ponto importante de deixar claro, o modelo em si ele não executa nenhuma ação para a gente, sempre a gente tem que fazer a execução da ação do nosso lado. Então se eu tenho que consultar um banco de dados, se eu tenho que interagir com uma API de terceiro, se eu tenho que executar algum código, tudo isso acontece do lado da aplicação.

O modelo não faz isso, ele delega para a gente fazer essa execução, executar essa ferramenta e retornar para ele o resultado para que ele siga ali no processo dele. Se você quiser executar esse código aí do seu lado, eu vou deixar o repositório aqui na descrição do vídeo. E para carregar os dados tem algumas outras coisas que eu deixei aqui prontas.

Aqui tem uma função para carregar esses dados dentro do Astra e tem um arquivo aqui, sampleData.json que tem algumas passagens. Se você quiser gerar alguns dados novos, você pode usar esse prompt aqui no chat IPT para gerar um JSON e aí você atualiza o conteúdo desse arquivo e tem uma API chamada loadSampleData que você pode chamar com post. Ele vai ler esse arquivo aqui e vai carregar os dados dentro da Collection.

Dentro aqui do envSample tem as variáveis que você vai precisar. Você vai precisar de um token no Astra que você pode pegar ali no dashboard do Astra, o API endpoint, o nome da Collection, nesse caso aqui pode usar essa que eu usei, FlightTickets, a chave da OpenAI, qual é o modelo que você quer usar. Isso é até uma das coisas interessantes de você treinar, testar com modelos diferentes e um ID fake de cliente só para você usar em todos os seus testes.

Nesse código Astracom é onde a conexão com o Astra é criada. Então ele vai usar a Astrapy que é uma library do Astra para fazer a conexão e fazer a criação da Collection se ela não existir. O AstraTool tem todas as funções aqui que a gente vai executar e o FlightAgent é onde de fato a gente é criado, definido e executado.

Bom pessoal, então é isso. Aqui é uma simplificação do que é o React. Dá para a gente criar diversas outras tools, criar outros casos mais avançados, mas a ideia aqui é mostrar como uma resposta que exige um certo pensamento para você encontrar a resposta pode ser atendida com esse padrão.

O React continua sendo muito bom, ele é usado para casos específicos, ele permite você customizar o comportamento do LLM através do prompt. Por outro lado, o prompt dele continua sempre crescendo, então ele pode consumir um pouco mais de tokens na hora da geração das respostas. Então isso é sempre um ponto a ser considerado na hora da utilização dessa técnica específica.

Eu vou deixar as referências todas aqui na descrição do vídeo e eu quero saber de vocês o que vocês querem ver, quais outros temas são interessantes para eu tentar trabalhar nos próximos vídeos. Então deixem suas ideias, seus comentários aqui embaixo e a gente se vê no próximo vídeo. Valeu, até mais.

**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.**