FINE-TUNING

"IT´S FINE" FRAMEWORK

PARA LEIGOS

SANDECO MACEDO

Os meus alunos do Instituto Federal de Goiás, amo vocês!


Meus Livros sobre
Inteligência Artificial
crewai
Para iniciantes
Agentes
Inteligentes
vol. 1
Sandeco Macedo
Para iniciantes
com IA
Python para
Inteligência
Artificial
Sandeco Macedo
Tudo o que você precisa saber
do Python antes de criar suas IAs
Prompts em Ação
Engenharia de Prompts
para leigos
Sandeco Macedo
Resultado
das minhas
pesquisas em IA

Sumário

CAPÍTULO 1

O modelo que não falava a sua língua

Antes de qualquer ferramenta, de qualquer **dataset** e de qualquer **GPU** esquentando, este capítulo tem um único trabalho: fazer você entender, de verdade, o que é o **fine-tuning**, quando vale a pena usá-lo e o que ele é capaz de entregar. Você não vai ver uma linha de código aqui, nem um comando, e isso é de propósito. Comece por uma história. Ela parece simples, mas dentro dela mora quase tudo o que você vai aprender ao longo do livro.

1.1 UMA HISTÓRIA: O MODELO QUE NÃO FALAVA A SUA LÍNGUA

Conheça a Marina. Ela é veterinária e dona de uma clínica que atende grandes animais no interior de Goiás. Num fim de semana, animada com a febre da inteligência artificial, a Marina resolveu colocar um **LLM** para responder às dúvidas que chegavam pelo WhatsApp da clínica. Em dez minutos ela tinha um assistente de propósito geral conectado, daqueles que respondem qualquer coisa, de receita de bolo a teoria da relatividade. Funcionou de primeira, e ela foi dormir achando que tinha resolvido a vida.

Na segunda-feira chegou a primeira pergunta de verdade, escrita do jeito que o pessoal da fazenda escreve. Era mais ou menos assim.

> Doutora, minha vaca tá com mole-mole e o casco soltando. É tristeza?

O assistente respondeu na hora, com toda a confiança do mundo.

> Sinto muito que você esteja triste. A tristeza é uma emoção natural e passageira. Que tal respirar fundo, tomar um copo de água e conversar com alguém querido? Se o sentimento persistir, procure um profissional de saúde mental.

Repare no tamanho do mal-entendido. Quem é do meio sabe que 'tristeza' ali é o nome popular da babesiose, uma doença causada por um parasita no sangue do gado, e que 'mole-mole' com 'casco soltando' são sinais clínicos que a Marina reconhece de olhos fechados. O **modelo base**, porém, não fazia a menor ideia disso. Ele não estava errado por ser burro. Ele estava errado porque ninguém nunca o ensinou a falar a língua da Marina.

E aqui mora a sacada deste livro. Esse assistente genérico leu boa parte da internet e sabe um pouquinho de quase tudo, mas não foi treinado para ser bom no que a Marina precisa. Ele é como um recém-formado brilhante que chegou de outro país: domina a teoria, fala bonito, mas ainda tropeça nos termos da casa, nos costumes do cliente e no jeito certo de responder. Para esse profissional virar gente da equipe, falta uma coisa só, e essa coisa tem nome.

Essa coisa é o **fine-tuning**. Em vez de a Marina se conformar com um modelo que fala difícil sobre tudo e direito sobre nada do mundo dela, ela pode pegar esse mesmo modelo e continuar o treino dele com exemplos da própria clínica: perguntas reais de produtores, respostas no tom certo, o vocabulário da pecuária, os protocolos que ela confia. Depois desse ajuste, quando chegar de novo um 'minha vaca tá com tristeza', o modelo entende na hora que o assunto é babesiose e responde como a Marina responderia.

Perceba que ela não construiu um modelo do zero, e nem precisava. Treinar um **LLM** do nada custa milhões e exige uma montanha de dados que quase ninguém tem. O que a Marina fez foi aproveitar todo o conhecimento que já estava dentro do modelo e apenas especializá-lo, dando os últimos retoques que o transformam de um sabe-tudo genérico em um especialista que fala a língua certa. É a diferença entre formar um médico do zero e dar a um médico já formado uma residência na sua área.

Guarde essa imagem da Marina, porque ela vai voltar em todos os capítulos. O livro inteiro é, no fundo, a história de como pegar um modelo que não fala a sua língua e ensiná-lo a falar, sem que você precise virar programador para isso. A Figura 1.1 resume o drama da Marina: a mesma pergunta, dois modelos, dois mundos de diferença.

1.2 O QUE É FINE-TUNING, AFINAL

Agora que você sentiu o problema na pele, vamos dar nome aos bois. Fazer **fine-tuning** é continuar o treino de um modelo que já foi treinado antes, alimentando-o com exemplos do

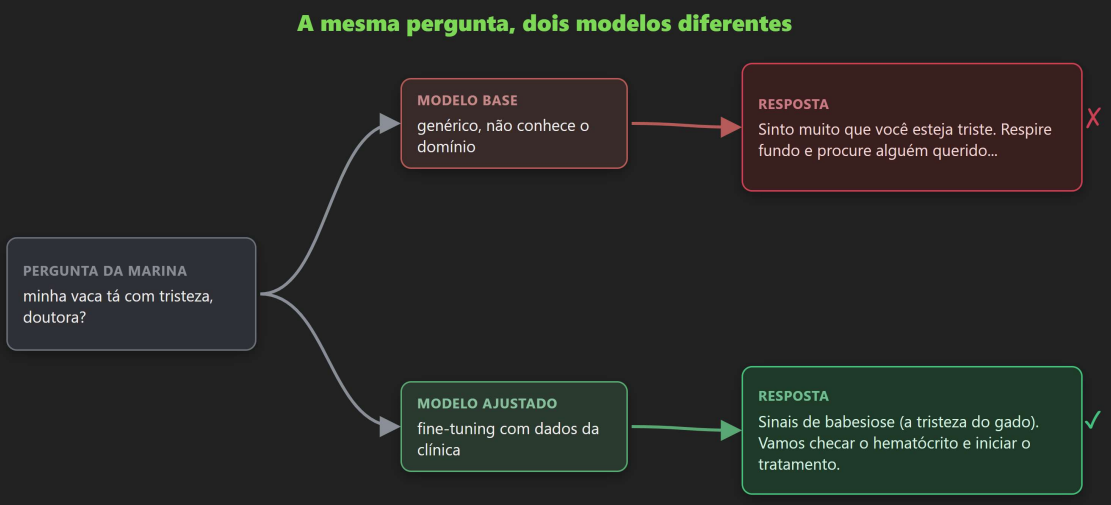

Figura 1.1: A mesma pergunta passando por um modelo base genérico e por um modelo ajustado com dados da clínica. O primeiro entende 'tristeza' como emoção; o segundo entende como babesiose.

que você quer que ele faça, para que ele ajuste o próprio comportamento. Repare na palavra 'continuar'. Você não começa do zero, você parte de um **modelo base** que já sabe português, já sabe raciocinar e já viu meio mundo, e apenas dá os retoques finais que faltavam.

Pense em como você contrataria a Marina de novo, agora do outro lado. Imagine que chega na clínica um veterinário recém-formado, cheio de teoria na cabeça. Ele sabe fisiologia, sabe farmacologia, sabe ler um exame. O que falta não é inteligência, é vivência: os termos que o produtor da região usa, o jeito de explicar sem assustar, os protocolos que aquela clínica adota. Ninguém manda esse profissional de volta para a faculdade por causa disso. A gente faz uma coisa muito mais barata, que é acompanhá-lo por algumas semanas mostrando casos reais até ele pegar o jeito da casa. O **fine-tuning** é exatamente essa residência, só que para o modelo. A Figura 1.2 coloca as duas jornadas lado a lado.

Esse detalhe de partir de um modelo pronto é o que torna a brincadeira viável. Treinar um **LLM** do zero, o tal do pré-treino, consome milhões de reais em computação e exige uma quantidade absurda de texto. Quase ninguém faz isso, e você quase certamente não vai precisar fazer. O que você vai fazer é pegar um modelo aberto, desses que qualquer um baixa, e especializá-lo com um punhado de exemplos bem escolhidos. É a diferença de ordem de grandeza entre construir uma universidade e pagar uma especialização.

Para deixar concreto desde já, observe o tipo de resposta que um **modelo base** aberto devolve quando recebe a pergunta da Marina, antes de qualquer ajuste. A Marina fez esse teste com um modelo pequeno, desses que rodam num notebook comum, e a conversa foi assim.

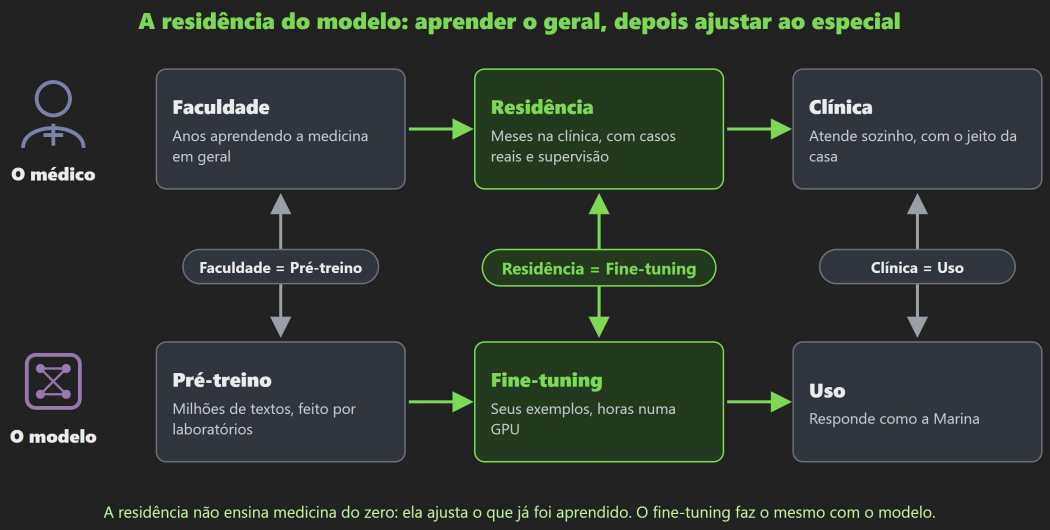

Figura 1.2: A vida de um médico e a vida de um modelo, lado a lado. A faculdade é o pré-treino, a residência é o fine-tuning, a clínica é o uso. Ninguém refaz a faculdade para ensinar o jeito da casa.

Doutora, minha vaca tá com tristeza. O que faço?

Lamento que sua vaca pareça abatida. Animais também podem passar por momentos de desânimo. Ofereça um ambiente tranquilo, água fresca e observe se ela volta a se alimentar normalmente nos próximos dias.

Repare que a resposta é educada, bem escrita e completamente inútil. O modelo trata 'tristeza' como desânimo, porque é só isso que ele conhece. Note que não há nada de errado com o modelo. O que falta é ensino, e ensino, no nosso mundo, tem nome de **fine-tuning**. Guarde esse exemplo, porque no fim do livro você vai ver este mesmo modelo, depois de ajustado, responder como a Marina responderia.

1.3 ONDE O FINE-TUNING MORA NA VIDA DE UM LLM

Para o **fine-tuning** fazer sentido na sua cabeça, você precisa enxergar onde ele mora na linha do tempo de um modelo. Um **LLM** moderno não nasce pronto de uma vez, ele passa por etapas, e cada etapa empilha uma camada de competência sobre a anterior. Acompanhe a Figura 1.3 enquanto a gente percorre essa jornada.

Tudo começa no pré-treino. Nessa fase o modelo lê uma montanha gigantesca de texto

Figura 1.3: As etapas da vida de um LLM. O fine-tuning acontece depois do pré-treino, quando o modelo já sabe o básico e só precisa ser especializado.

e aprende a prever a próxima palavra. Parece pouco, mas é desse joguinho que emerge o conhecimento de mundo, a gramática e a capacidade de raciocinar. Essa etapa é a cara, a demorada e a que você não vai fazer. Ela entrega um modelo que sabe muito, mas ainda não sabe conversar nem seguir ordens direito.

Em seguida vem o **SFT**, o supervised fine-tuning, que é a primeira forma de ajuste fino e a mais importante deste livro. Aqui o modelo vê milhares de exemplos no formato 'recebi tal pedido, respondi assim' e aprende a se comportar como um assistente que obedece instruções. Foi o **SFT** que transformou aqueles modelos crus em algo com que dá para conversar. Depois, opcionalmente, vem o alinhamento por preferência, com técnicas como **RLHF**, **DPO** e **ORPO**, que refinam o gosto do modelo mostrando a ele qual resposta as pessoas preferem. Por fim, o modelo vai para o uso de verdade, que é colocá-lo de pé respondendo a gente real.

Perceba uma coisa importante: quando a gente fala 'fine-tuning' neste livro, está falando principalmente do **SFT**, que é onde você entra com os seus dados e a sua intenção. O pré-treino fica para os grandes laboratórios; o alinhamento por preferência você verá como o próximo degrau, no último capítulo. O miolo, que é onde o modelo deixa de ser genérico e vira o seu, é todo seu, e é sobre ele que os próximos capítulos vão se debruçar.

1.4 PROMPT, RAG OU FINE-TUNING: O MAPA DE DECISÃO

Antes de você sair treinando modelo por aí, precisa ouvir uma verdade que economiza tempo e dinheiro: nem todo problema pede **fine-tuning**. Muita gente se anima, parte direto para o treino e descobre tarde demais que um caminho mais simples resolveria. Existem três ferramentas na mesa, e a sabedoria está em escolher a certa. Use a Figura 1.4 como

bússola enquanto a gente discute cada uma.

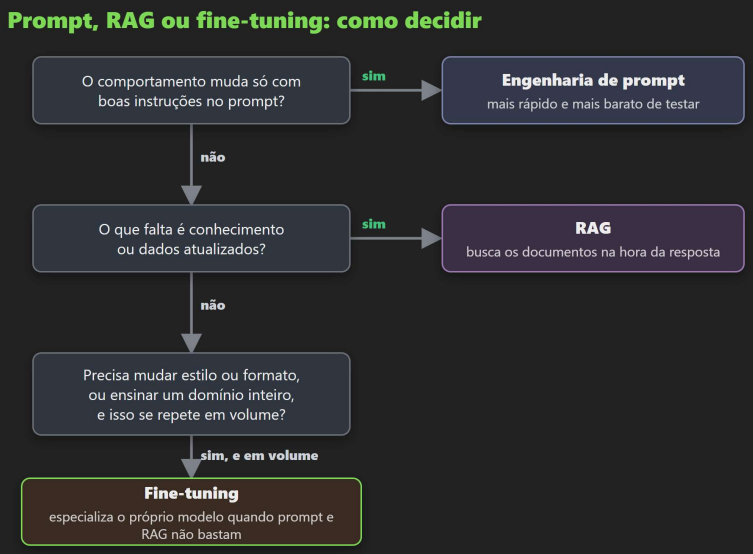

Figura 1.4: Um mapa de decisão simples para escolher entre engenharia de prompt, RAG e fine-tuning conforme a natureza do seu problema.

A primeira ferramenta é a engenharia de **prompt**, que é simplesmente escrever instruções melhores. Se você consegue fazer o modelo se comportar do jeito certo só explicando bem o que quer, dando exemplos no próprio pedido, pare por aí. É a opção mais rápida, mais barata e que não exige treino nenhum. Trocar o comportamento é tão simples quanto reescrever um parágrafo. Boa parte do que as pessoas acham que precisa de treino, na verdade, precisa só de um **prompt** mais caprichado.

A segunda é o **RAG**, que você talvez já conheça. Quando o problema não é de comportamento e sim de conhecimento, quando o modelo precisa saber fatos que não estavam nos dados dele, ou informações que mudam toda hora, como o estoque da sua loja ou a bula mais recente de um remédio, a resposta é buscar esses documentos na hora e entregá-los ao modelo junto com a pergunta. O **RAG** dá ao modelo uma consulta à biblioteca, sem mexer no que ele é.

A terceira é o **fine-tuning**, e ele brilha quando as duas anteriores não bastam. Use-o quando precisar mudar o jeito do modelo de forma profunda e consistente: adotar um estilo ou formato muito específico, dominar o vocabulário e os padrões de um domínio inteiro, ou quando você tem volume suficiente de exemplos do comportamento desejado. A Marina é o caso clássico: ela não queria que o modelo consultasse um documento, ela queria que ele pensasse como veterinária. Isso se ensina, e ensinar é **fine-tuning**. Vale dizer que essas ferramentas não brigam entre si; é comum um sistema sério usar as três juntas, um modelo ajustado que ainda consulta documentos por **RAG** e recebe instruções claras no **prompt**.

1.5 QUANDO NÃO FAZER FINE-TUNING

Já que estamos sendo honestos, vale uma seção inteira para os sinais de que você deveria fechar o notebook e não treinar nada. Saber quando não fazer é tão valioso quanto saber fazer, e vai te poupar de muita frustração. A Figura 1.5 resume os três sinais de alerta.

Figura 1.5: Três placas de alerta antes de treinar: sem dados de qualidade, conhecimento que muda todo dia, e o prompt já resolve. Qualquer uma delas pede outro caminho.

O primeiro sinal é não ter dados. **Fine-tuning** se faz com exemplos, e exemplos de qualidade, em quantidade razoável. Se você tem dez frases jogadas e a esperança de que o modelo 'aprenda o padrão', sinto muito, mas isso não vai acontecer como você imagina. Sem um **dataset** decente, treinar só vai ensinar o modelo a errar com mais confiança. O capítulo de dados existe justamente porque essa é a parte que mais separa quem entrega um bom modelo de quem perde tempo.

O segundo sinal é o problema ser de conhecimento que muda. Se a informação que você precisa injetar vive mudando, como preços, notícias ou registros novos todo dia, não adianta treinar, porque no dia seguinte o modelo já estará desatualizado e você teria que treinar de novo. Esse é o território do **RAG**, como vimos na seção anterior. Tentar resolver com **fine-tuning** um problema que é de **RAG** é como decorar o jornal de hoje achando que vai saber a notícia de amanhã.

O terceiro sinal é mais sutil: às vezes o **prompt** já resolve e você não percebeu. Antes de montar um projeto de treino, gaste algumas horas tentando fazer o modelo acertar só com instruções e exemplos no próprio pedido. É surpreendente quantas vezes isso basta. Treinar

dá trabalho, custa **GPU**, exige avaliação cuidadosa, então deixe essa artilharia pesada para quando os tiros mais leves já falharam. Se depois de tudo isso você ainda precisa mudar o comportamento do modelo de forma profunda e tem dados para isso, ótimo, então é hora de **fine-tuning**, e o resto do livro é para você.

1.6 O QUE O FINE-TUNING APRENDE DE VERDADE

Existe uma expectativa errada que derruba mais projetos do que qualquer erro técnico, e é melhor você ouvir dela agora, no capítulo gratuito, do que descobrir depois de gastar horas de **GPU**. Muita gente imagina que o **fine-tuning** é uma forma de 'colocar os documentos dentro do modelo', como se o modelo fosse uma gaveta e o treino fosse guardar os papéis lá. Não é. O modelo não é uma gaveta. Ele é mais parecido com um estagiário que leu os seus documentos algumas vezes e pegou o jeito.

Pense de novo na residência do veterinário recém-formado. Depois de algumas semanas na clínica, ele aprende o tom com que a Marina fala com o produtor, o formato dos laudos, o vocabulário da região, a ordem em que ela investiga um caso. Isso ele absorve rápido, porque é padrão, e padrão se aprende por repetição. O que ele não faz é decorar de cor a dosagem exata de cada remédio de cada protocolo da clínica. Se você perguntar um número específico, ele pode chutar com confiança, e é justamente aí que mora o perigo. A Figura 1.6 separa os dois lados.

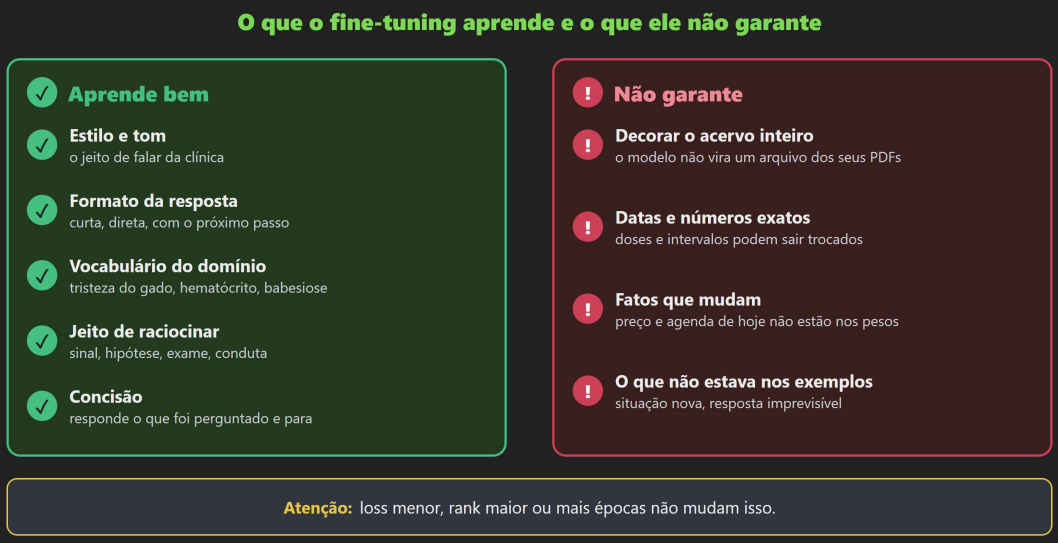

Figura 1.6: O que o fine-tuning aprende bem e o que ele não garante. Estilo, formato, vocabulário e jeito de raciocinar entram com facilidade. Decorar o acervo inteiro, com datas e números exatos, não está garantido.

Com o modelo é a mesma coisa. O **fine-tuning** ensina muito bem estilo, tom, formato da resposta, concisão, vocabulário do domínio e o jeito de raciocinar sobre um problema. Ele não garante que cada fato do seu acervo ficou gravado nos pesos, e nenhum truque de treino muda isso: treinar por mais tempo, com mais força ou com um adaptador maior não transforma o estagiário numa enciclopédia. Se o que você precisa é que o modelo cite o número certo de um documento específico, esse é o território do **RAG**, e os dois podem trabalhar juntos: o modelo ajustado responde no tom da Marina, com o trecho certo do protocolo buscado na hora.

Esse limite parece uma má notícia, mas é uma libertação. Ele diz exatamente o que medir. Se o **fine-tuning** ensina comportamento, então a pergunta certa no fim do projeto não é 'o modelo decorou tudo?', e sim 'o modelo responde como a Marina responderia, e melhor do que respondia antes?'. Você vai ver ao longo do livro que boa parte do trabalho sério de **fine-tuning** consiste em fazer essa pergunta de um jeito que possa ser respondido com números honestos, medidos antes e depois, no mesmo teste, e não com a sensação de que 'ficou bom'.

1.7 O QUE O ITS-FINE FAZ POR VOCÊ

Até alguns anos atrás, tudo que você leu até aqui terminava numa parede. Para fazer **fine-tuning** de verdade era preciso programar: preparar os dados em Python, configurar bibliotecas de treino, entender dezenas de parâmetros, vigiar a memória da **GPU** e ainda escrever a avaliação na mão. A Marina é veterinária, não programadora. E é aqui que este livro se afasta de todos os outros sobre o assunto.

A Marina fez o projeto inteiro conversando. Ela abriu o assistente de código no computador da clínica, descreveu o que queria em português, apontou a pasta onde estavam os protocolos em PDF, disse que preferia treinar na própria máquina, e a partir daí um framework chamado ITS-FINE conduziu o resto, com um agente fazendo perguntas quando precisava de uma decisão dela e executando o trabalho técnico sozinho. O nome é um trocadilho com 'está tudo bem', e a promessa é justamente essa: você descreve a intenção, e o resto fica bem. Acompanhe na Figura 1.7 as onze paradas dessa viagem, contadas do jeito que a Marina viveu.

Primeiro, o agente perguntou a intenção: qual tarefa, como é uma resposta boa, onde estão os dados e onde treinar. Nenhuma pergunta sobre parâmetro técnico, porque isso é consequência, não decisão dela. Depois ele olhou o ambiente: reconheceu a placa de vídeo da máquina sem baixar nada, estimou se o treino caberia na memória e preparou um ambiente isolado para não bagunçar o computador da clínica. Em seguida cuidou dos dados: leu os PDFs página por página, gerou pares de pergunta e resposta usando um modelo

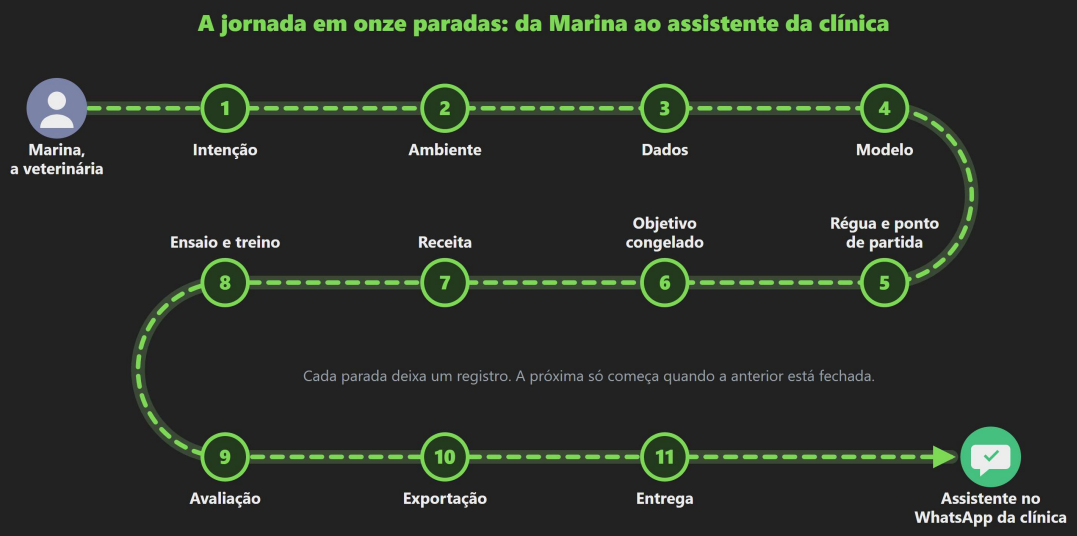

Figura 1.7: As onze paradas da jornada da Marina, da intenção ao assistente respondendo no WhatsApp da clínica. Cada capítulo deste livro explica algumas delas.

local, e só aceitou os pares cuja resposta estava literalmente escrita no documento, para não inventar nada. Separou os dados em treino, validação e teste garantindo que um mesmo protocolo nunca aparecesse em dois lados ao mesmo tempo.

Então o agente escolheu o modelo base, um modelo aberto que cabia na placa da Marina, e travou a versão exata dele, para o projeto ser reproduzível daqui a um ano. Antes de treinar, fez algo que a maioria das pessoas pula: montou uma régua de medição, testou essa régua com pegadinhas para garantir que ela não aprovaria uma resposta errada, e mediu quanto o modelo original já acertava. Só depois disso escreveu o objetivo, com um critério numérico de sucesso, e o congelou. A partir daí, nenhuma mudança de ideia poderia contaminar o resultado.

Com o objetivo travado, o agente propôs uma receita de treino, explicou cada escolha, e pediu autorização à Marina. Antes do treino de verdade, rodou um ensaio curto de poucos passos só para medir memória e tempo. Aí treinou, vigiando o processo e salvando pontos de retomada. No fim, avaliou o modelo novo contra o original no mesmo teste que estava trancado desde o início, e deu um veredito honesto: aprovado, reprovado ou inconclusivo. Aprovado, exportou o modelo para o formato que roda de graça num computador comum, mediu de novo nessa versão final, porque comprimir um modelo pode piorá-lo, e entregou o assistente com um recibo dizendo o que foi medido e quais são os limites.

Repare no que a Marina não fez. Ela não escreveu código. Ela não escolheu parâmetro. Ela não copiou comando de tutorial. Ela tomou as decisões que só ela podia tomar: o que queria, quais dados usar, o que era uma resposta boa, se autorizava cada passo que custava tempo ou dinheiro, e se aceitava o critério de sucesso. O resto foi conduzido. E quando ela

quis saber onde o projeto tinha parado depois de uma semana longe, bastou perguntar, e o agente respondeu em que parada estava e qual era o próximo passo.

Também repare no que o framework não promete. Ele não promete que o modelo decorou os protocolos, porque você acabou de ler que isso não está garantido. Ele não promete que vai funcionar em qualquer computador, porque treinar exige uma placa de vídeo, ou um ambiente alugado na nuvem quando não há placa. E ele não aceita aprovar um resultado sem medir, nem repetir a mesma receita quando ela falhou. Essa honestidade é a parte mais valiosa do projeto, e é ela que separa um assistente que dá gosto de usar de um brinquedo que responde bonito e erra.

1.8 O QUE VOCÊ VAI CONSTRUIR AO LONGO DESTE LIVRO

Para fechar este capítulo de abertura, deixe-me mostrar o mapa da viagem, para você saber onde está pisando em cada parada. A Figura 1.8 mostra os nove capítulos, e a ordem deles não é acidental: é exatamente a ordem em que um projeto real de **fine-tuning** acontece, e é a ordem em que o ITS-FINE conduz a Marina.

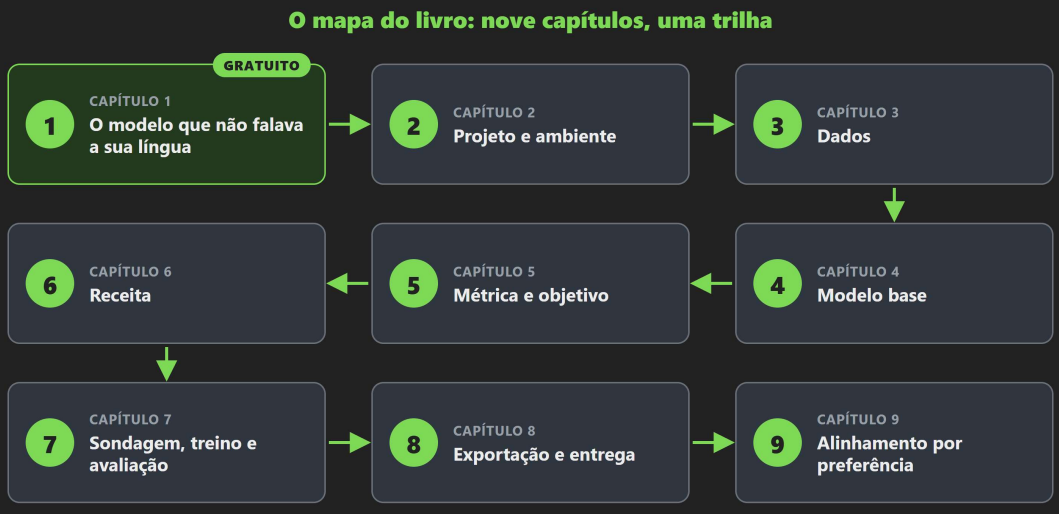

Figura 1.8: O mapa do livro. Cada capítulo é uma ou mais paradas da jornada, sempre com a teoria antes e a prática com o framework depois.

Todo capítulo daqui em diante tem o mesmo ritmo. Primeiro você entende o conceito e o porquê dele, com uma analogia do cotidiano, do jeito que expliquei a residência do veterinário. Só depois você vê como o framework aplica aquilo na prática, olhando a conversa entre a Marina e o agente e os comandos que o agente executou, sempre explicados. Você não precisa saber programar para acompanhar, mas vai terminar o livro entendendo o que