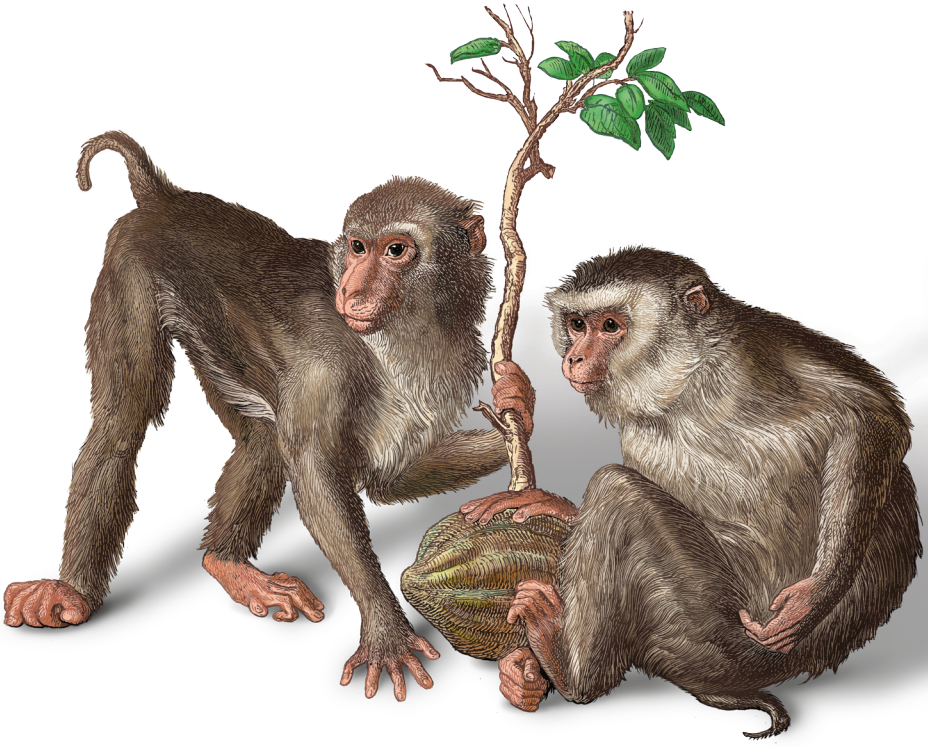
O'REILLY®

Deutsche Ausgabe

KI-Agenten entwickeln

Entwurf, Implementierung, Monitoring



Michael Albada

Übersetzung von Frank Langenau

Hinweise zur Benutzung

Dieses E-Book ist **urheberrechtlich geschützt**. Mit dem Erwerb des E-Books haben Sie sich verpflichtet, die Urheberrechte anzuerkennen und einzuhalten. Sie sind berechtigt, dieses E-Book für persönliche Zwecke zu nutzen. Sie dürfen es auch ausdrucken und kopieren, aber auch dies nur für den persönlichen Gebrauch. Die Weitergabe einer elektronischen oder gedruckten Kopie an Dritte ist dagegen nicht erlaubt, weder ganz noch in Teilen. Und auch nicht eine Veröffentlichung im Internet oder in einem Firmennetzwerk.

Copyright-Vermerk

Nutzungs- und Verwertungsrechte

Sie sind berechtigt, dieses E-Book ausschließlich für persönliche Zwecke zu nutzen. Insbesondere sind Sie berechtigt, das E-Book für Ihren eigenen Gebrauch auszudrucken oder eine Kopie herzustellen, sofern Sie diese Kopie auf einem von Ihnen alleine und persönlich genutzten Endgerät speichern. Zu anderen oder weitergehenden Nutzungen und Verwertungen sind Sie nicht berechtigt.

So ist es insbesondere unzulässig, eine elektronische oder gedruckte Kopie an Dritte weiterzugeben. Unzulässig und nicht erlaubt ist des Weiteren, das E-Book im Internet, in Intranets oder auf andere Weise zu verbreiten oder Dritten zur Verfügung zu stellen. Eine öffentliche Wiedergabe oder sonstige Weiterveröffentlichung und jegliche den persönlichen Gebrauch übersteigende Vervielfältigung des E-Books ist ausdrücklich untersagt. Das vorstehend Gesagte gilt nicht nur für das E-Book insgesamt, sondern auch für seine Teile (z. B. Grafiken, Fotos, Tabellen, Textabschnitte).

Urheberrechtsvermerke, Markenzeichen und andere Rechtsvorbehalte dürfen aus dem E-Book nicht entfernt werden.

Die automatisierte Analyse des Werkes, um daraus Informationen insbesondere über Muster, Trends und Korrelationen gemäß § 44b UrhG (»Text und Data Mining«) zu gewinnen, ist untersagt.

Markenschutz

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.

Haftungsausschluss

Ungeachtet der Sorgfalt, die auf die Erstellung von Text, Abbildungen und Programmen verwendet wurde, können weder Verlag noch Autor*innen, Herausgeber*innen oder Übersetzer*innen für mögliche Fehler und deren Folgen eine juristische Verantwortung oder irgendeine Haftung übernehmen.

Stimmen zum Buch *KI-Agenten entwickeln*

Endlich ein Buch darüber, wie KI in die menschliche Arbeitswelt integriert werden kann. Michael nutzt sein Fachwissen aus skalierbaren Unternehmen wie Uber und Microsoft, um technischen Führungskräften in kleinen und mittleren Unternehmen zu vermitteln, wie sie wirklich skalierbare agentenbasierte Lösungen für ihre Transformation entwickeln können.

– Birju Shah, Professor für Produktmanagement und KI an der Kellogg School of Management der Northwestern University, ehemaliger Leiter des Uber-KI-Produktteams

KI-Agenten entwickeln ist ein prägnanter, praxisorientierter Leitfaden, der Führungskräften den Übergang vom Hype um generative KI zu realen Systemen erleichtert. Er fasst komplexe Konzepte zu umsetzbaren Strategien zusammen und schlägt eine Brücke zwischen Vision und Umsetzung für Unternehmen, die messbare Effizienz und Wettbewerbsfähigkeit anstreben.

– Amanda Cheng, Partner von Founders Bay

Als Klinikarzt, der an der Schnittstelle zwischen Medizin und Technologie arbeitet, halte ich dieses Buch für eine unverzichtbare Lektüre für alle, die KI-Agenten entwickeln – klar, praktisch und reich an Einblicken in Tools, Orchestrierung und Designmuster, die für Anwendungsfälle im Gesundheitswesen wie Aufnahme, Triage und Workflow-Integration relevant sind.

– Carrie Ho, MD, Assistenzprofessorin, Hämatologin/Onkologin, UCSF

Dies ist das Buch, das ich jedem Team vor dem Einsatz von Agenten empfehlen würde: ein klarer, rigoroser Ansatz für Architektur, Sicherheit und Messung, der die Bereitstellung beschleunigt und Risiken reduziert.

– Brad Sarsfield, Senior Director, Microsoft Security, KI-Forschung und -Entwicklung

Die beste einbändige Einführung in den Aufbau von KI-Agentensystemen – Sie können Hunderte von Artikeln lesen, oder eben dieses eine Buch.

– Arun Rao, ehemalige Meta GenAI-Gruppe, außerordentlicher Professor an der UCLA

KI-Agenten entwickeln

Entwurf, Implementierung, Monitoring

Michael Albada

Übersetzung von Frank Langenau

O'REILLY®

Wir hoffen, dass Sie Freude an diesem Buch haben und sich Ihre Erwartungen erfüllen. Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: **service@rheinwerk-verlag.de**.

Informationen zu unserem Verlag und Kontaktmöglichkeiten finden Sie auf unserer Verlagswebsite **www.dpunkt.de**. Dort können Sie sich auch umfassend über unser aktuelles Programm informieren und unsere Bücher und E-Books bestellen.

Autor: Michael Albada
Übersetzung: Frank Langenau
Lektorat: Alexandra Follenius
Buchmanagement: Julia Griebel
Copy-Editing: Claudia Lötschert, *www.richtiger-text.de*
Satz: III-satz, *www.drei-satz.de*
Herstellung: Stefanie Weidner
Covergestaltung: Karen Montgomery, Michael Oréal, *www.oreal.de*

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.dnb.de* abrufbar.

ISBN Print: 978-3-96009-286-5
ISBN PDF: 978-3-96010-976-1
ISBN ePub: 978-3-96010-977-8

1. Auflage 2026

Authorized German translation of the English edition of *Building Applications with AI Agents: Designing and Implementing Multiagent Systems*, ISBN 978-1098176501. Copyright © 2025 Advance AI LLC. This translation is published and sold by permission of O'Reilly Media Inc., the owner of all rights to publish and sell the same.

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

Rheinwerk Verlag GmbH • Rheinwerkallee 4 • 53227 Bonn
service@rheinwerk-verlag.de

Inhalt

Vorwort

Als ich anfing, Sprachmodelle, Tools, Orchestrierung und Speicher zu dem zu verbinden, was wir heute als Agenten bezeichnen, war ich überrascht, wie leistungsfähig dieses Entwurfsmuster war und wie viel Verwirrung es zu diesem Thema gab. Während ich Agenten entwickelte und meine Erkenntnisse zu Vorfalluntersuchungen, der Suche nach Sicherheitsbedrohungen, der Erkennung von Schwachstellen und mehr teilte, stellte ich fest, dass dieses neueste Entwurfsmuster es uns ermöglichte, ganz neue Arten von Problemen zu lösen, aber auch viele Hürden mit sich brachte, die es zu überwinden galt, um die Zuverlässigkeit in Praxisanwendungen zu gewährleisten. Ingenieure, Wissenschaftlerinnen, Produktmanager und Führungskräfte wollten alle mehr wissen:

- »Wie bringe ich meinen Agenten zum Laufen?«
- »Meinen Agenten kann ich zwar manchmal zum Laufen bringen, aber wie schaffe ich es, dass er meistens oder immer funktioniert?«
- »Wie wähle ich ein Modell für meinen Anwendungsfall aus?«
- »Wie entwerfe ich gute Tools für meinen Agenten?«
- »Welche Art von Speicher benötige ich?«
- »Sollte ich RAG (Retrieval-Augmented Generation) verwenden?«
- »Sollte ich ein Single-Agent- oder ein Multi-Agent-System erstellen?«
- »Welche Architektur sollte ich verwenden?«
- »Muss ich Feinabstimmungen vornehmen?«
- »Wie kann ich Agenten in die Lage versetzen, aus Erfahrungen zu lernen und sich im Laufe der Zeit zu verbessern?«

Es gibt zwar viele Blogbeiträge und Forschungsarbeiten, die sich mit bestimmten Aspekten des Themas »Entwerfen von Agentensystemen« befassen, doch mir wurde klar, dass es an zugänglichen, ganzheitlichen und vertrauenswürdigen Leitfäden dazu mangelt. Ich konnte kein Buch finden, das ich meinen Kolleginnen und Kollegen empfehlen wollte, also habe ich mich daran gemacht, selbst eines zu schreiben.

Über intensive Diskussionen habe ich Teams dabei geholfen, sich in der Komplexität von KI-Agenten zurechtzufinden, wobei ich ihre individuellen Ziele, Einschränkungen und Umgebungen berücksichtigt habe. KI-Agentensysteme sind komplex und verbinden Auto-

nomie, Entscheidungsfindung und Interaktion auf eine Weise, wie es herkömmliche Software nicht tut. Sie sind datengesteuert, anpassungsfähig und umfassen mehrere Komponenten wie Wahrnehmung, Schlussfolgern, Handeln und Lernen, während sie gleichzeitig mit Benutzern, Tools und anderen Agenten interagieren. Erschwerend kommt hinzu, dass die Grundmodelle, auf denen diese Agenten basieren, von Natur aus probabilistisch und stochastisch sind, was das Bewerten und Testen erschwert.

Dieses Buch zeigt, wie sich Anwendungen mit KI-Agenten erstellen lassen. Es deckt den gesamten Lebenszyklus ab, von der konzeptionellen Entwicklung bis zur Bereitstellung und Wartung. Veranschaulicht wird dies an praktischen Fallstudien, unterstützt durch Referenzen und überprüft von Fachleuten aus der Branche. Abschnitte zu fortgeschrittenen Themen – wie Agentenarchitekturen, Tool-Integration, Speichersysteme, Orchestrierung, Multiagent-Koordinierung, Messung, Überwachung, Sicherheit und ethische Überlegungen – werden durch Expertenbeiträge weiter verfeinert.

Dieses Buch zu schreiben, war auch für mich eine Entdeckungsreise. Die ersten Entwürfe lösten Diskussionen aus, die meine Ansichten infrage stellten und neue Ideen hervorbrachten. Ich hoffe, dass sich dieser Prozess fortsetzt, wenn Sie das Buch lesen und eigene Erkenntnisse beisteuern. Teilen Sie mir gern Ihr Feedback zu diesem Buch über X (Twitter), LinkedIn, meine persönliche Website oder andere Kanäle mit.

Worum es in diesem Buch geht

Dieses Buch bietet einen praktischen Rahmen, um robuste Anwendungen mithilfe von KI-Agenten erstellen zu können. Es behandelt wichtige Herausforderungen und bietet Lösungen an für Fragen wie:

- Was macht einen KI-Agenten aus und wann sollte ich einen einsetzen? Wie unterscheiden sich Agenten von herkömmlichen ML-Systemen (Machine Learning)?
- Wie entwerfe ich Agentenarchitekturen für spezifische Anwendungsfälle, einschließlich Szenarioauswahl und Kernkomponenten wie Tools, Speicher, Planung und Orchestrierung?
- Was sind effektive Strategien für die Planung, Argumentation, Ausführung, Tool-Auswahl und Topologien wie Ketten, Bäume und Graphen von Agenten?
- Wie kann ich Agenten dazu bringen, über nichtparametrische Methoden, Feinabstimmung und Transferlernen aus Erfahrungen zu lernen?
- Wie skaliert man von Einzelagenten- zu Multiagentensystemen, einschließlich der Koordinationsmuster wie demokratisch, hierarchisch oder Actor-Critic-Ansätze?
- Wie kann ich die Performance von Agenten mit Metriken, Tests und Produktionsüberwachung bewerten und verbessern?
- Welche Tools und Frameworks eignen sich am besten für die Entwicklung, Bereitstellung und Absicherung von Agenten gegen Risiken?
- Wie gewährleiste ich, dass Agenten sicher, ethisch und skalierbar sind, und zwar unter Berücksichtigung von Benutzererlebnis (User Experience, UX), Vertrauen, Voreingenommenheit und Einhaltung gesetzlicher Vorschriften?

Inhaltlich basiert das Buch auf etablierten technischen Prinzipien und sich entwickelnden Praktiken bei KI-Agenten. Es liefert Fallstudien (wie zum Beispiel Kundensupport, persönliche Assistenz, Rechts-, Werbe- und Codeüberprüfungsagenten) und Diskussionen über Kompromisse, damit Sie Lösungen auf Ihre Bedürfnisse zuschneiden können.

Was dieses Buch nicht ist

Dieses Buch ist keine Einführung in die Grundlagen von KI oder ML. Es setzt Kenntnisse mit Konzepten wie neuronale Netze, Verarbeitung natürlicher Sprache und grundlegender Programmierung in Sprachen wie Python voraus. Sollten Sie mit diesen Themen noch nicht vertraut sein, finden Sie Hinweise auf entsprechende Quellen, doch liegt der Schwerpunkt auf der angewandten Agentenentwicklung.

Es handelt sich auch nicht um Schritt-für-Schritt-Anleitungen für bestimmte Tools, da sich Technologien schnell weiterentwickeln. Stattdessen unterstützt es Sie bei der Bewertung und der Auswahl von Tools, wobei Pseudocode und Beispiele die Konzepte veranschaulichen. Für die praktische Umsetzung werden Online-Tutorials und Dokumentationen empfohlen, darunter Frameworks wie LangChain und AutoGen.

Für wen dieses Buch gedacht ist

Dieses Buch richtet sich an Ingenieure, Entwicklerinnen und technische Führungskräfte, die agentenbasierte KI-Anwendungen entwickeln möchten. Es ist auf Rollen wie KI-Ingenieure, Softwareentwicklerinnen, ML-Ingenieure. Data Scientists und Produktmanagerinnen mit technischem Hintergrund zugeschnitten. Vielleicht können Sie sich mit folgenden Szenarios identifizieren:

- Sie haben die Aufgabe, ein autonomes System für die Entscheidungsunterstützung oder interaktive Dienste zu entwickeln.
- Sie verfügen über einen funktionierenden Agentenprototyp, den Sie stabilisieren und für den Einsatz in der Produktion vorbereiten möchten.
- Ihr Team hat Probleme mit der Zuverlässigkeit von Agenten – Umgang mit Ausfällen, Anpassen an dynamische Umgebungen oder Orchestrieren komplexer Aufgaben –, und Sie suchen nach systematischen Ansätzen, einschließlich Orchestrierung, Speicher und Lernen aus Erfahrungen.
- Sie integrieren Agenten in bestehende Workflows und suchen nach Best Practices für Skalierbarkeit, Multiagentenkoordinierung, UX-Design, Messung, Validierung, Überwachung und Sicherheit.

Auch als Tool-Entwickler, der Lücken im Agentenökosystem identifiziert, als Forscherin, die Anwendungen untersucht, oder als Arbeitssuchender, der sich auf Aufgaben im Bereich KI-Agenten vorbereitet, können Sie von diesem Buch profitieren.

Wegweiser zu den Themen des Buchs

Die Kapitel folgen dem Lebenszyklus beim Erstellen einer KI-Agentenanwendung und sind in drei Hauptabschnitte gegliedert.

Die ersten drei Kapitel behandeln die Kernkonzepte, Designprinzipien und wesentliche Komponenten.

- Kapitel 1 führt Agenten ein, beschreibt ihre Vorteile, nennt Anwendungsfälle und vergleicht sie mit herkömmlichem ML und neuesten Entwicklungen.
- Kapitel 2 bietet einen Überblick über die Gestaltung von Agentensystemen, unter anderem über Szenarioauswahl, Kernkomponenten (Modellauswahl, Tools, Speicher, Planung), Design-Kompromisse, Architekturmuster (Einzelagent, Multiagent, modular) und Best Practices.
- Kapitel 3 konzentriert sich auf das UX-Design und behandelt Interaktionsmodalitäten (Text, Grafik, Sprache, Video), vergleicht synchrone mit asynchronen Erlebnissen und geht auf Kontextbeibehaltung, Kommunikationsfähigkeiten, Vertrauen und wichtige UX-Prinzipien ein.

Die nächsten fünf Kapitel befassen sich vor allem mit dem Erstellen, der Orchestrierung und dem Skalieren von Agenten:

- Kapitel 4 befasst sich ausführlich mit Tools, unter anderem mit dem Design (lokal, API-basiert, Plug-in, Hierarchien) und der automatisierten Tool-Entwicklung (Codegenerierung, Imitationslernen, Tool-Lernen aus Belohnungen).
- Kapitel 5 behandelt die Orchestrierung mit Grundlagen (Parametrisierung, Tool-Auswahl, Ausführung), Methoden zur Tool-Auswahl (generativ, semantisch, hierarchisch, maschinell gelernt), Tool-Topologien (Zerlegung, einzelne/parallele/sequenzielle Ausführung, Ketten, Bäume, Graphen) und Planungsstrategien (inkrementelle Ausführung, Zero-Shot, Few-Shot, ReAct).
- Kapitel 6 untersucht die Speichertechnik, einschließlich grundlegender Ansätze (Kontextfenster, Schlüsselwort-basiert), semantischer Speicher und Vektorspeicher (semantische Suche, RAG, Erfahrungsspeicher), GraphRAG (Wissensgraphen) und Arbeitsspeicher (Whiteboards, Notizen).
- Kapitel 7 befasst sich mit Lernen aus Erfahrungen, mit nichtparametrischem Lernen (Erfahrungen als Beispiele, Erforschung/Auswertung, Reflexion), parametrisches Lernen (Feinabstimmung großer/kleiner Modelle) und Transferlernen.
- In Kapitel 8 geht es um Skalierung – von einem Agenten auf viele Agenten –, einschließlich der Frage, wann Mulitagenten eingesetzt werden sollten, wie sie sich koordinieren lassen (demokratisch, verwaltungsmäßig, hierarchisch, Actor-Critic, automatisiertes Design) und welche Frameworks (wie LangChain) infrage kommen.

Die letzten fünf Kapitel befassen sich mit Validierung, Überwachung, Sicherheit, Verbesserung und dem Zusammenwirken von Mensch und Agent:

- Kapitel 9 behandelt Messung und Validierung hinsichtlich der Schlüsselziele (Genauigkeit, Robustheit, Effizienz usw.), Bewertungssätzen, Komponententests (Tools, Pla-

nung, Speicher, Lernen), Integrationstests (Ende-zu-Ende, Konsistenz, Halluzinationen), Einschränkungen und Vorbereitungen für die Bereitstellung.

- Kapitel 10 konzentriert sich auf die Produktionsüberwachung, wobei es um Fehlerursachen, Agentenmetriken (Systemzustand, automatisierte/menschliche Bewertung, Feedback), Verteilungsverschiebungen und Überwachung in großem Maßstab (Analysen, Warnmeldungen, Protokollierung) geht.
- Kapitel 11 erläutert Verbesserungsschleifen mit Feedback-Pipelines (Problemerkennung, menschliche Überprüfung, Verfeinerung, Priorisierung), Experimenten (Schattenbereitstellungen, A/B-Tests, adaptives Gating) und kontinuierlichem Lernen (im Kontext, Offline-Retraining, Online-Reinforcement).
- Kapitel 12 befasst sich mit dem Schutz von Agentensystemen und behandelt dabei spezifische Risiken, die Sicherung von LLMs (Modellauswahl, Abwehrmaßnahmen, Red Teaming, Feinabstimmung), Datenschutz (Privatsphäre, Herkunft), die Sicherung von Agenten (Sicherheitsvorkehrungen, externe/interne Schutzmaßnahmen) und Governance/Compliance.
- Kapitel 13 diskutiert das Verhältnis von Menschen und Agenten, wobei ethische Prinzipien (Aufsicht, Transparenz, Fairness, Erklärbarkeit, Datenschutz), der Aufbau von Vertrauen/Aufsicht, der Umgang mit Voreingenommenheit und Überlegungen zur Rechenschaftspflicht/Regulierung berücksichtigt werden.

Die Abschnitte zu Themen, mit denen Sie bereits vertraut sind, können Sie gern überspringen – das Buch ist modular strukturiert.

Hinweis: Ich verwende häufig »wir«, um mich auf Sie (die Leserin oder den Leser) und mich zu beziehen und so eine Atmosphäre des gemeinsamen Lernens zu vermitteln.

Konventionen in diesem Buch

In diesem Buch werden folgende typografische Konventionen verwendet:

Kursiv
: Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Schreibmaschinenschrift`
: Wird in Programmlistings verwendet und im Fließtext für Programmelemente wie zum Beispiel Variablen- oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter.

`Schreibmaschinenschrift fett`
: Kennzeichnet Befehle oder andere Texte, die vom Benutzer buchstäblich eingegeben werden sollen.

`Schreibmaschinenschrift kursiv`
: Zeigt Text, der ersetzt werden soll durch Werte, die die Benutzerin oder der Benutzer bereitstellt, oder Werte, die sich aus dem Kontext ergeben.

Codebeispiele

Zusätzliches Material (Codebeispiele, Übungen usw.) finden Sie zum Herunterladen unter *https://oreil.ly/building-applications-with-ai-agents-supp*.

Dieses Buch soll Ihnen bei Ihrer Arbeit helfen. Ganz allgemein gilt: Wenn in diesem Buch Beispielcode angeboten wird, können Sie ihn in Ihren Programmen und Dokumentationen verwenden. Sie müssen sich dafür nicht unsere Erlaubnis einholen, es sei denn, Sie reproduzieren einen großen Teil des Codes. Schreiben Sie zum Beispiel ein Programm, das mehrere Teile des Codes aus diesem Buch benutzt, brauchen Sie keine Erlaubnis. Verkaufen oder vertreiben Sie Beispiele aus O'Reilly-Büchern, brauchen Sie allerdings eine Erlaubnis. Beantworten Sie eine Frage, indem Sie dieses Buch und Beispielcode daraus zitieren, brauchen Sie keine Erlaubnis. Binden Sie einen großen Anteil des Beispielcodes aus diesem Buch in die Dokumentation Ihres Produkts ein, brauchen Sie wiederum eine Erlaubnis.

Wir freuen uns über eine Quellenangabe, verlangen diese jedoch in der Regel nicht. Eine Quellenangabe enthält üblicherweise Titel, Autor, Verlag und ISBN, zum Beispiel: »*KI-Agenten entwickeln: Entwurf, Implementierung, Monitoring* von Michael Albada (O'Reilly). Copyright 2026 Rheinwerk Verlag GmbH, ISBN 978-3-96009-286-5.«

Falls Sie befürchten, zu viele Codebeispiele zu verwenden oder die oben genannten Befugnisse zu überschreiten, kontaktieren Sie uns unter *kommentar@oreilly.de*.

Danksagungen

Als Erstautor staunt man schon, wie viele Menschen an der Entstehung eines Buchs beteiligt sind, und dank der Beiträge vieler wunderbarer Menschen ist dieses Buch nun fertiggestellt worden.

Es hat über ein Jahr gedauert, dieses Buch zu schreiben, und ich bin besonders den technischen Gutachtern dankbar, die sich die wertvolle Zeit genommen haben, mir ihr ausführliches Feedback, ihre Perspektiven und ihre Erkenntnisse mitzuteilen. Nuno Campos hat mir mit seinem unschätzbaren Fachwissen über Agenten und LangChain geholfen und mich auf Konzepte hingewiesen, die ich übersehen hatte. Prashanth Josyula hat die Texte und Codebeispiele auf einem hohen technischen Niveau gehalten und sein fundiertes Fachwissen eingebracht. Megan MacLennan war meine Expertin für technisches Schreiben und hat dazu beigetragen, die Zugänglichkeit und Relevanz für ein breites Publikum sicherzustellen. Frühe Entwürfe sind immer unvollkommen, und ich bin meinen technischen Gutachtern zutiefst dankbar, dass sie meine Fehler und Versäumnisse mit Nachsicht aufgenommen haben. Vielen Dank für ihre Geduld und ihre wertvollen Vorschläge.

Ein besonderer Dank gilt auch Anthony Wainman, der mich von Anfang an als Ideengeber begleitet und mir wertvolle Hinweise zur Struktur, zum Inhalt, zu den Beispielen und vielem mehr gegeben hat.

Ohne das fantastische Team von O'Reilly, insbesondere meine Entwicklungsredakteurin Shira Evans, die das Projekt begleitet hat, wäre dieses Buch nicht möglich gewesen. Vielen Dank an Melissa Potter für ihr frühzeitiges Feedback und ihre Rezensionen sowie an meine

Produktionsredakteure Ashley Stussy und Gregory Hyman. Nicole Butterfield hat einen unschätzbaren Beitrag dazu geleistet, Konzepte in die Realität umzusetzen.

Ich möchte mich auch bei allen bedanken, die die Vorabversion des Buchs gelesen und Vorschläge und Ermutigungen gegeben haben, darunter Tiago Dufau de Vargas, Jenny Song, Leonidas Askianakis, Karthik Rao und Drew Hoskins.

Sehr viel verdanke ich auch meinen brillanten aktuellen und ehemaligen Kollegen bei Microsoft, ServiceNow und Uber, insbesondere Olcay Cirit, Dawn Woodard, Sameera Poduri, Zoubin Ghahramani, Piero Molino, Pablo Bellver, Jaikumar Ganesh, Jay Stokes, Marc-Alexandre Cote, Chi Wang, Anush Sankaran, Amir Abdi, Tong Wang, Antonios Matakos, Max Golovanov, Abe Starosta, Francis Beckert, Malachi Jones, Taylor Black, Ryan Sweet, Lital Badash, Amir Pirogovsky, Alexander Stojanovic, Brad Sarsfield, Chang Kawaguchi, Jure Leskovic, Chiyu Zhang, Andrew Zhao, Matthieu Lin und vielen, vielen anderen. Vielen Dank für ihre Weisheit, ihre Einsichten, ihre Geduld, ihre Mentorenschaft und ihre zahlreichen Vorschläge.

Ich möchte Luke Miratrix danken, der mich in die Statistik eingeführt und mir das Programmieren beigebracht hat. Ich möchte auch meinen wichtigsten akademischen Mentoren Lisa Schmitt, Lise Shelton, James Sheehan, Finbarr Livesey, Matthew Sommer, James Ward, Charles Isbell, Michael Littman, Zsolt Kira und Constantine Dovrolis dafür danken, dass sie mein Denken sowohl im Großen als auch im Kleinen geprägt haben.

Dieses Buch ist in vielerlei Hinsicht eine Zusammenfassung der Lektionen, die ich im Laufe meines Lebens und meiner Karriere gelernt habe, und ich bin vielen weiteren Menschen dankbar, die ich hier nicht alle nennen kann. Ich bin zutiefst dankbar für die Möglichkeit, dieses Buch zu veröffentlichen, und hoffe sehr, dass es Ihnen von Nutzen sein wird.

KAPITEL 1

Einführung in Agenten

Wir sind Zeitzeugen eines tiefgreifenden technologischen Wandels, der von autonomen Agenten vorangetrieben wird – d.h. von intelligenten Softwaresystemen, die in der Lage sind, selbstständig zu schlussfolgern, Entscheidungen zu treffen und innerhalb dynamischer Umgebungen effektiv zu interagieren. Im Gegensatz zu herkömmlicher Software interpretieren autonome Agenten Kontexte, passen sich an sich ändernde Szenarios an und führen anspruchsvolle Aktionen mit minimaler menschlicher Aufsicht aus.

KI-Agenten – eine Definition

Autonome Agenten sind intelligente Systeme, die darauf ausgelegt sind, Daten unabhängig zu analysieren, ihre Umgebung zu interpretieren und kontextbezogene Entscheidungen zu treffen. Mit zunehmender Popularität des Begriffs »Agent« hat sich seine Bedeutung verwässert und er wird oft auf Systeme angewendet, denen echte Autonomie fehlt. In der Praxis gibt es ein ganzes Spektrum von Agenten. Wirklich autonome Agenten zeichnen sich durch sinnvolle Entscheidungsfindung, kontextbezogene Schlussfolgerungen und adaptives Verhalten aus. Umgekehrt führen viele Systeme, die als »Agenten« betitelt sind, lediglich deterministische Skripte oder streng gesteuerte Workflows aus. Es ist eine große Herausforderung, wirklich autonome und adaptive Agenten zu entwickeln, weshalb viele Teams einfachere Ansätze wählen, um schnellere Ergebnisse zu erreichen. Für einen echten Agenten ist es daher entscheidend, ob er echte Entscheidungen trifft, anstatt statischen Skripten zu folgen.

Die rasante Entwicklung autonomer Agenten ist hauptsächlich auf Durchbrüche in Basismodellen und Reinforcement Learning zurückzuführen. Während sich herkömmliche Anwendungsfälle mit Basismodellen darauf konzentriert haben, für den Menschen verständliche Klartextausgaben zu generieren, ermöglichen es die neuesten Fortschritte diesen Modellen, strukturierte Funktionssignaturen und Parameterauswahlen zu generieren. Dann können Orchestrierungs-Frameworks diese Funktionen ausführen, sodass Agenten Daten nachschlagen, externe Systeme manipulieren und konkrete Aktionen ausführen können. In diesem Buch beschreiben wir mit dem Begriff »agentenorientiertes System« die gesamte unterstützende Funktionalität, die es einem Assistenten ermöglicht, effektiv zu arbeiten. Dazu gehören Tools, Speicher, Basismodell, Orchestrierung und unterstützende Infrastruktur.

Mit einer wachsenden Auswahl an Protokollen wie dem Model Context Protocol (mit dem sich Kapitel 4 beschäftigt) und dem Agent-to-Agent Protocol (in Kapitel 8 erläutert) werden diese Agenten in der Lage sein, Remote-Tools zu verwenden und mit anderen Agenten zusammenzuarbeiten, um Probleme zu lösen. Dies eröffnet enorme Möglichkeiten für eine hochentwickelte Automatisierung – bringt aber auch eine große Verantwortung mit sich, diese Systeme sorgfältig zu entwerfen, zu messen und zu verwalten, um sicherzustellen, dass ihre Handlungen im Einklang mit menschlichen Werten stehen und sie in komplexen, dynamischen Umgebungen sicher funktionieren.

Die Revolution beim Vortraining

Herkömmliches ML[1] stellt zwar eine unglaublich leistungsfähige Technik dar, wird aber normalerweise durch die Quantität und Qualität des Datensatzes eingeschränkt. ML-Praktiker werden Ihnen in der Regel sagen, dass sie den Großteil ihrer Zeit nicht mit dem Trainieren von Modellen verbringen, sondern mit dem Sammeln und Bereinigen von Datensätzen, die sie für das Trainieren heranziehen können. Der unglaubliche Erfolg generativer Modelle, die mit großen Datenmengen trainiert wurden, hat gezeigt, dass einzelne Modelle sich nun ohne zusätzliches Training an einen weiten Bereich von Aufgaben anpassen können. Dies stellt jahrelange Praxis auf den Kopf. Um eine Anwendung zu entwickeln, die ML nutzt, musste man früher einen ML-Ingenieur oder Data Scientist einstellen, ihn die Daten sammeln lassen, und dann dieses Modell bereitstellen. Dank der neuesten Entwicklungen bei großen, vortrainierten generativen Modellen sind nun hochwertige Modelle verfügbar, die für viele Anwendungsfälle recht gut funktionieren. Dazu genügt ein einziger Aufruf eines gehosteten Modells, ohne dass irgendein Training oder Hosting erforderlich ist. Dies senkt die Kosten und die Komplexität der Entwicklung von Anwendungen, die ML und KI nutzen, erheblich.

Jüngste Fortschritte bei Large Language Models (LLMs[2]) wie GPT-5, Claude von Anthropic, Llama von Meta, Gemini Ultra von Google und V3 von DeepSeek haben die Performance für einen ganzen Bereich schwieriger Aufgaben noch weiter gesteigert und damit den Umfang der Probleme, die sich mit vortrainierten Modellen lösen lassen, erweitert. Diese Basismodelle erlauben es durch folgende Komponenten, die Funktionalität von Agenten zu verbessern:

Verständnis natürlicher Sprache
: Benutzereingaben intuitiv interpretieren und darauf reagieren

Kontextbezogene Interaktion
: Kontext für relevante und genaue Antworten bei längeren Interaktionen beibehalten

Strukturierte Inhaltsgenerierung
: Text, Code und strukturierte Ausgaben erzeugen, die für analytische und kreative Aufgaben unerlässlich sind

1 ML – Machine Learning = maschinelles Lernen

2 LLMs – Large Language Models = große Sprachmodelle

Auch wenn diese Modelle für sich genommen sehr leistungsfähig sind, lassen sie sich auch verwenden, um Entscheidungen in klar abgegrenzten Bereichen zu treffen, sich an neue Informationen anzupassen und Tools aufzurufen, um echte Arbeiten zu erledigen. Die Integration in ausgefeilte Orchestrierungs-Frameworks ermöglicht es diesen Modellen, direkt mit externen Systemen zu interagieren und praktische Aufgaben wie die folgenden auszuführen:

Kontextbezogene Interpretation und Entscheidungsfindung
: Navigieren in mehrdeutigen Situationen ohne gründliche Vorprogrammierung

Verwendung von Tools
: Aufrufen anderer Software, um Informationen abzurufen oder Aktionen auszuführen

Adaptive Planung
: Autonomes Planen und Ausführen komplexer, mehrstufiger Aktionen

Zusammenfassung von Informationen
: Schnelles Verarbeiten umfangreicher Dokumente, Extrahieren wichtiger Erkenntnisse und damit Unterstützung bei rechtlichen Analysen, der Zusammenfassung von Forschungsergebnissen und der Pflege von Inhalten

Verwaltung unstrukturierter Daten
: Intelligentes Interpretieren und Reagieren auf unstrukturierte Texte wie zum Beispiel E-Mails, Dokumente, Protokolle und Berichte

Codegenerierung
: Schreiben und Ausführen von Code und Schreiben von Komponententests

Automatisierung von Routineaufgaben
: Effiziente Behandlung wiederholter Aktivitäten im Kundenservice und in Verwaltungsabläufen, sodass sich die Mitarbeitenden auf komplexere Aufgaben konzentrieren können

Multimodale Informationssynthese
: Durchführen komplexer Analysen von Bild-, Audio- oder Videodaten in großem Umfang

Diese verbesserte Flexibilität ermöglicht es autonomen Agenten, mit komplexen und dynamischen Szenarios effektiv umzugehen, die statische ML-Modelle in der Regel nicht bewältigen können.

Arten von Agenten

Aufgrund seiner zunehmenden Popularität hat man die Bedeutung des Begriffs »Agent« auf eine Vielzahl von KI-fähigen Systemen ausgeweitet, was oftmals zu Verwirrung darüber führt, was einen KI-Agenten wirklich ausmacht. Die Webseite *The Information* ordnet die verschiedenen Agenten sieben praktischen Kategorien (*https://oreil.ly/99_ZV*) zu, die widerspiegeln, wie diese Technologien heute angewendet werden:

Agenten für Geschäftsaufgaben
: Diese Agenten automatisieren vordefinierte Geschäftsabläufe, wie beispielsweise die robotergestützte Prozessautomatisierung von UIPath, die Low-Code-Abläufe von Micro-

soft Power Automate oder die App-Integrationen von Zapier. Sie führen Sequenzen deterministischer Aktionen aus, die in der Regel durch Ereignisse ausgelöst werden, wobei sie nur minimal kontextbezogene Schlussfolgerungen anwenden.

Konversationsagenten
: In diese Kategorie fallen Chatbots und Agenten für Kundendienste, die über Oberflächen für natürliche Sprache mit Benutzerinnen und Benutzern interagieren. Optimiert sind sie für die Verwaltung von Dialogen, das Erkennen von Absichten und die Gesprächsführung. Beispiele hierfür sind virtuelle Assistenten, die in Plattformen für Kundendienste eingebettet sind.

Rechercheagenten
: Rechercheagenten leiten Aufgaben, die Informationen beschaffen, synthetisieren und zusammenfassen. Sie scannen Dokumente, Wissensdatenbanken oder das Web, um strukturierte Ausgaben zu liefern, die menschliche Analysten unterstützen. Als Beispiele seinen Perplexity AI und Elicit genannt.

Analytikagenten
: Agenten für Analytiker wie Power BI Copilot oder Glean sind vor allem darauf ausgelegt, strukturierte Datensätze zu interpretieren und Erkenntnisse, Dashboards und Berichte zu generieren. Oftmals sind sie nahtlos in Warehouses für Unternehmensdaten integriert, sodass Benutzerinnen und Benutzer komplexe Daten in natürlicher Sprache abfragen können.

Entwickleragenten
: Tools wie Cursor, Windsurf und GitHub Copilot verkörpern Codierungsassistenten, die Entwicklerinnen und Entwickler dabei unterstützen, Code zu erzeugen, zu refaktorisieren und zu erklären. Sie sind tief in die Workflows der IDEs integriert, um die Produktivität der Softwareentwicklung zu steigern.

Domänenspezifische Agenten
: Diese Agenten sind für spezialisierte Branchen optimiert, wie zum Beispiel Rechtswesen (Harvey), Medizin (Hippocratic AI) oder Finanzwesen. Sie kombinieren domänenspezifisches Wissen mit strukturierten Workflows, um gezielte Unterstützung auf Expertenniveau zu bieten.

Browserbasierte Agenten
: Diese Agenten navigieren auf Websites, interagieren mit ihnen, extrahieren Informationen und führen Aktionen aus, ohne dass eine menschliche Interaktion erforderlich ist. Im Gegensatz zur herkömmlichen robotergestützten Prozessautomatisierung, die vorgegebenen Schritten folgt, kombinieren moderne browserbasierte Agenten das Sprachverständnis, die optische Wahrnehmung und die dynamische Planung, um sich spontan anzupassen.

Neben diesen sieben Arten von Agenten sind Sprach- und Videoagenten wichtig und werden voraussichtlich in den kommenden Jahren zunehmend eingesetzt werden:

Sprachagenten
: Derartige Agenten beruhen auf Sprachverständnis- und -generierung von Anfang bis Ende. Sie ermöglichen die Automatisierung von Gesprächen in Bereichen wie Kundenservice, Terminplanung und sogar der Abwicklung von Bestellungen in Echtzeit.

Videoagenten

Diese Agenten präsentieren den Benutzerinnen und Benutzern avatarbasierte Videoantworten, die lippensynchrone Sprache, Mimik und Gestik kombinieren. Sie sind auf dem Vormarsch in den Bereichen Vertrieb, Kunden-Onboarding, Marketing und Tools der virtuellen Präsenz – sodass sie skalierbare, personalisierte Videointeraktionen ohne manuelle Produktion erlauben.

Wichtig ist, dass die Anzahl und Vielfalt der Agententypen rapide zunimmt und wir wahrscheinlich neue Arten von Agenten auf vielen Gebieten sehen werden, wenn sich das Feld und die zugrunde liegenden Technologien weiterentwickeln. In diesem Buch legen wir den Schwerpunkt auf die Kernkategorie der Agenten, die auf Sprachmodellen basieren, insbesondere solche, die Text und Code verwenden. Wir gehen zwar auch auf die Automatisierung von Geschäftsaufgaben, Sprache und Video ein, aber in den folgenden Kapiteln werden wir uns hauptsächlich mit Agenten befassen, die auf Sprachmodellen basieren – mit ihren Architekturen, ihrem Denken und ihrer Benutzererfahrung.

Nachdem Sie nun im Überblick die sich entwickelnden Arten von Agenten kennen, lautet die nächste entscheidende Frage: Welches Modell sollten Sie für Ihren Agenten wählen? Die Modellauswahl ist ein komplexer und sich schnell ändernder Bereich. Wie der nächste Abschnitt erläutert, müssen Sie Faktoren wie Aufgabenkomplexität, Modalitätsunterstützung, Latenz- und Kostenbeschränkungen sowie Integrationsanforderungen abwägen, um die richtige Wahl für Ihren Agenten zu treffen.

Modellauswahl

Heute sind wir in der glücklichen Lage, dass sowohl kommerzielle Anbieter als auch die Open-Source-Community eine Vielzahl leistungsstarker Modelle zur Verfügung stellen. OpenAI, Anthropic, Google, Meta und DeepSeek bieten jeweils hochmoderne Basismodelle mit beeindruckenden Universalfähigkeiten an. Gleichzeitig erweitern Open-Weight-Modelle wie Llama, Mistral und Gemma die Grenzen dessen, was sich mit lokalen oder fein abgestimmten Bereitstellungen erreichen lässt. Noch auffälliger ist der rasante Fortschritt bei kleinen und mittelgroßen Modellen. Neue Techniken für Destillation, Quantisierung und synthetische Datengenerierung ermöglichen es kompakten Modellen, überraschend hohe Fähigkeitsgrade von ihren größeren Pendants zu erben.

Diese explosionsartige Zunahme der Auswahlmöglichkeiten ist eine gute Nachricht: Der Wettbewerb treibt schnellere Innovationen, bessere Leistung und niedrigere Kosten voran. Aber er schafft auch ein Dilemma: Wie wählt man das richtige Modell für das eigene agentenbasierte System aus? Die Wahrheit ist, dass es keine allgemeingültige Antwort gibt. Tatsächlich beginnt man am besten damit, das neueste Allzweckmodell von einem führenden Anbieter wie OpenAI oder Anthropic zu verwenden. Wie Sie Tabelle 1.1 entnehmen können, bieten diese Modelle von Haus aus eine starke Performance, die nur wenige Anpassungen erfordern und Sie bei vielen Anwendungen überraschend weit bringen. GPT-5 mini (Aug 2025) führt insgesamt mit dem höchsten mittleren Score (0,819), dicht gefolgt von o4-mini (0,812) und o3 (0,811). Proprietäre und frei zugängliche Modelle wie Qwen3, Grok 4, Claude 4 und Kimi K2 zeigen ebenfalls konkurrenzfähige Ergebnisse.

Tabelle 1.1: HELM-Core-Scenario-Rangliste (August 2025). Vergleichende Benchmark-Performance der 10 besten Modelle bei Schlussfolgerungs- und Bewertungsaufgaben: MMLU-Pro, GPQA, IFEval, WildBench und Omni-MATH

Modell	Mittlerer Score	MMLU-Pro – COT korrekt	GPQA – COT korrekt	IFEval – IFEval Strict Acc	WildBench – WB Score	Omni-MATH – Acc
GPT-5 mini (2025-08-07)	0,819	0,835	0,756	0,927	0,855	0,722
o4-mini (2025-04-16)	0,812	0,82	0,735	0,929	0,854	0,72
o3 (2025-04-16)	0,811	0,859	0,753	0,869	0,861	0,714
GPT-5 (2025-08-07)	0,807	0,863	0,791	0,875	0,857	0,647
Qwen3 235B A22B Instruct 2507 FP8	0,798	0,844	0,726	0,835	0,866	0,718
Grok 4 (0709)	0,785	0,851	0,726	0,949	0,797	0,603
Claude 4 Opus (20250514, erweitertes Denken)	0,78	0,875	0,709	0,849	0,852	0,616
gpt-oss-120b	0,77	0,795	0,684	0,836	0,845	0,688
Kimi K2 Instruct	0,768	0,819	0,652	0,85	0,862	0,654
Claude 4 Sonnet (20250514, erweitertes Denken)	0,766	0,843	0,706	0,84	0,838	0,602

Allerdings sind sie nicht immer die effizienteste Wahl. Für viele Aufgaben – insbesondere diejenigen, die klar umrissen sind, eine geringere Latenz bieten oder einen engen Kostenrahmen haben – können wesentlich kleinere Modelle eine nahezu gleichwertige Performance zu einem Bruchteil der Kosten bieten. Dies hat zu einem wachsenden Trend geführt: der automatisierten Modellauswahl. Einige Plattformen leiten einfachere Abfragen an schnelle, kostengünstige kleine Modelle weiter und reservieren die großen, teuren Modelle für komplexe Schlussfolgerungen. Diese dynamische Optimierung während der Testphase erweist sich als effektiv und deutet auf eine Zukunft hin, in der Multimodellsysteme zur Norm werden.

Letztlich optimieren Sie die Modellauswahl mit enormen Anstrengungen, erzielen aber nur minimale Gewinne – aber wenn Ihre Größenordnung oder Ihre Einschränkungen dies nicht erfordern, ist ein einfacher Anfang sinnvoll. Mit der Zeit lohnt es sich oft, mit kleineren Modellen zu experimentieren, Feinabstimmungen vorzunehmen oder Abrufe hinzuzufügen, um die Performance zu verbessern und die Kosten zu senken. Denken Sie einfach daran: Die Zukunft ist mit ziemlicher Sicherheit Multimodell-orientiert, und wenn Sie jetzt auf Flexibilität setzen, wird sich das später auszahlen.

Von synchronen zu asynchronen Abläufen

Herkömmliche Softwaresysteme führen Aufgaben in der Regel synchron aus, indem sie Schritt für Schritt vorgehen und warten, bis jede Aktion abgeschlossen ist, bevor sie mit der nächsten beginnen. Dieser Ansatz ist zwar unkompliziert, kann aber recht ineffizient sein – insbesondere beim Warten auf externe Eingaben oder bei der Verarbeitung großer Datenmengen.

Im Gegensatz dazu sind autonome Agenten für eine asynchrone Arbeitsweise konzipiert. Sie können mehrere Aufgaben parallel verwalten, sich schnell an neue Informationen anpassen und Prioritäten dynamisch setzen. Diese asynchrone Verarbeitung steigert die Effizienz erheblich, reduziert Leerlaufzeiten und optimiert die Nutzung von Rechenressourcen.

Die praktischen Auswirkungen dieser Veränderung sind erheblich. Zum Beispiel:

- E-Mails können mit bereits vorbereiteten Antwortentwürfen eintreffen.
- Rechnungen können mit vorausgefüllten Zahlungsdetails versandt werden.
- Softwareentwicklerinnen und -entwickler können Tickets mit dem entsprechenden Code erhalten, um das Problem zu lösen, und Komponententests, um sie zu bewerten.
- Kundendienstagenten können mit vorgeschlagenen Antworten und empfohlenen Handlungsanweisungen versorgt werden.
- Sicherheitsanalystinnen können Warnmeldungen erhalten, die bereits automatisch untersucht und mit relevanten Bedrohungsinformationen angereichert wurden.

In jedem Fall beschleunigen Agenten nicht nur routinemäßige Workflows, sondern verändern auch die Art der Arbeit selbst. Diese Entwicklung wandelt die Rollen des Menschen vom Ausführenden zum Manager von Aufgaben. Anstatt Zeit mit sich wiederholenden oder mechanischen Schritten zu verbringen, können sich die Mitarbeitenden ganz auf strategische Kontrolle, Überprüfung und hochwertige Entscheidungsfindung konzentrieren – was die menschliche Kreativität und Urteilsfähigkeit stärkt, während die Mitarbeitenden sich um die operativen Details kümmern. Diese Agenten machen es viel einfacher, proaktiv statt reaktiv zu handeln.

Praktische Anwendungen und Use Cases

Die Vielseitigkeit autonomer Agenten eröffnet eine Vielzahl von Anwendungsmöglichkeiten in verschiedenen Branchen. Um dieses Buch auf klare, konkrete Anwendungsfälle zu beschränken, habe ich sieben Beispiele aus der Praxis mit Bewertungssystemen im öffentlichen GitHub-Repository zu diesem Buch zusammengestellt (*https://oreil.ly/GitHub-scenarios*). Wir werden häufig auf diese Beispiele zurückkommen, wenn wir die wichtigsten Aspekte von Agentensystemen untersuchen:

Kundendienstagenten
: Der Kundensupport ist eine der vorherrschenden Anwendungen für autonome Agenten. Diese Agenten bearbeiten allgemeine Anfragen, wickeln Rückerstattungen ab, aktualisieren Bestellungen und leiten komplexe Probleme an menschliche Mitarbeitende weiter. So bieten sie einen 24/7-Support, verbessern die Kundenzufriedenheit und senken die Betriebskosten.

Finanzdienstleistungsagenten
: Im Banken- und Finanzwesen unterstützen Agenten die Kontenverwaltung, Kreditbearbeitung, Betrugsaufklärung und Neugewichtung von Anlageportfolios. Sie optimieren den Kundenservice, beschleunigen die Transaktionsabwicklung und verbessern die Sicherheit, indem sie verdächtige Aktivitäten in Echtzeit erkennen.

Agenten für die Aufnahme und Triage von Patienten im Gesundheitswesen
: Diese Agenten unterstützen den Frontline-Betrieb im Gesundheitswesen, indem sie neue Patientinnen und Patienten registrieren, Versicherungen überprüfen, Symptome bewerten, um die Versorgung zu priorisieren, Termine planen, Krankengeschichten verwalten und Überweisungen koordinieren, wodurch die Workflows effizienter gestaltet und die Behandlungsergebnisse verbessert werden.

IT-Helpdesk-Agenten
: Agenten für IT-Helpdesks verwalten den Benutzerzugriff, beheben Netzwerk- und Systemprobleme, stellen Softwareupdates bereit und leiten ungelöste Probleme an Spezialisten weiter. Sie steigern die Produktivität, indem sie häufig auftretende technische Probleme schnell lösen.

Agenten für die Prüfung von Rechtsdokumenten
: Agenten im Rechtswesen unterstützen Anwältinnen und Anwaltsgehilfen, indem sie Verträge überprüfen, Rechtsrecherchen durchführen, Klientinnen und Klienten aufnehmen und Konflikte prüfen, die Beweisaufnahme verwalten, die Einhaltung von Vorschriften bewerten, Schadensberechnungen durchführen und Fristen verfolgen. Dies trägt zur Rationalisierung der Arbeitsabläufe bei und verbessert die Genauigkeit bei rechtlichen Angelegenheiten.

Analystenagenten im Security Operations Center (SOC)
: Agenten als SOC-Analysten untersuchen Sicherheitswarnungen, sammeln Informationen zu Bedrohungen, fragen Protokolle ab, klassifizieren Vorfälle, isolieren kompromittierte Hosts und bringen Sicherheitsteams auf den neuesten Stand. Sie beschleunigen die Reaktion auf Vorfälle und stärken die Sicherheitslage des Unternehmens.

Agenten für Lieferketten und Logistik
: Agenten in der Lieferkettenverwaltung optimieren den Lagerbestand, verfolgen Lieferungen, bewerten Lieferanten, koordinieren Lagerabläufe, prognostizieren den Bedarf, bewältigen Störungen und kümmern sich um Compliance-Anforderungen. Diese Fähigkeiten tragen dazu bei, die Resilienz und Effizienz in globalen Netzwerken aufrechtzuerhalten.

Autonome Agenten bieten ein erhebliches Potenzial für verschiedene Anwendungsfälle, vom Kundensupport und der persönlichen Assistenz bis hin zu Rechtsdienstleistungen und Werbung. Indem sie diese Agenten in ihre Abläufe integrieren, können Unternehmen effizienter arbeiten, die Servicequalität verbessern und neue Möglichkeiten für Innovation und Wachstum erschließen. Während wir in diesem Buch die Fähigkeiten und Anwendungsmöglichkeiten autonomer Agenten weiter untersuchen, wird deutlich, dass ihre Auswirkungen in vielen Branchen tiefgreifend und weitreichend sein werden.

Nachdem wir uns nun einige Beispielagenten angesehen haben, diskutieren wir im nächsten Abschnitt einige der wichtigsten Überlegungen bei der Entwicklung unserer Agentensysteme.

Workflows und Agenten

In vielen realen Projekten kann die Wahl zwischen einem einfachen Skript, einem deterministischen Workflow, einem herkömmlichen Chatbot, einem RAG[3]-System (Retrieval-Augmented Generation) oder einem vollwertigen autonomen Agenten den Unterschied zwischen einer eleganten Lösung und einem überzüchteten, schwer wartbarem Durcheinander ausmachen. Um diese Entscheidung zu erleichtern, sollten Sie vier wichtige Faktoren berücksichtigen: die Variabilität Ihrer Eingaben, die Komplexität der erforderlichen Argumentation, etwaige Performance- oder Compliance-Beschränkungen und den laufenden Wartungsaufwand.

Erstens: Wann würden Sie sich dafür entscheiden, kein Basismodell – oder überhaupt keine ML-Komponente – zu verwenden? Wenn Ihre Eingaben vollständig vorhersehbar sind und jede mögliche Ausgabe im Voraus beschrieben werden kann, sind einige Zeilen prozeduraler Code oft schneller, kostengünstiger und viel einfacher zu testen als eine ML-basierte Pipeline. Beispielsweise lässt sich eine Logdatei, die immer dem Format »YYYY-MM-DD HH:MM:SS – Nachricht« folgt, zuverlässig mit einem kleinen, auf regulären Ausdrücken basierenden Parser in Python oder Go verarbeiten. Wenn Ihre Anwendung eine Latenz im Millisekundenbereich verlangt – wie zum Beispiel ein eingebettetes System, das in Echtzeit auf Sensordaten reagieren muss –, bleibt einfach keine Zeit für einen API-Aufruf eines Sprachmodells. In derartigen Fällen ist herkömmlicher Code die richtige Wahl. Schließlich erfordern regulierte Bereiche (medizinische Geräte, Luftfahrt, bestimmte Finanzsysteme) oft eine vollständig deterministische, überprüfbare Entscheidungslogik – Blackbox-Modelle, die mit neuronalen Netzen arbeiten, erfüllen die Zertifizierungsanforderungen nicht. Trifft eine dieser Bedingungen zu – deterministische Eingaben, strenge Performance- oder Erklärbarkeitsanforderungen oder ein statischer Problembereich –, ist einfacher Code fast immer einem Basismodell vorzuziehen.

Als Nächstes sollten Sie deterministische oder halbautomatisierte Workflows betrachten. Hier lässt sich die Logik als endliche Menge von Schritten oder Verzweigungen ausdrücken, und Sie wissen im Voraus, wo möglicherweise menschliches Eingreifen oder zusätzliche Fehlerbehandlung erforderlich ist. Angenommen, Sie erfassen Rechnungen von einer kleinen Gruppe von Lieferanten und die einzelnen Rechnungen erhalten Sie in einem von drei bekannten Formaten: CSV, JSON oder PDF. Nun können Sie einen Workflow erstellen, der jedes Format an den entsprechenden Parser weiterleitet, auf Unstimmigkeiten prüft und für eine manuelle Überprüfung anhält, wenn ein Feld einen einfachen Abgleich nicht besteht – hier ist kein tiefes semantisches Verständnis erforderlich. Wenn Ihr System fehlgeschlagene Schritte mit exponentiellem Backoff wiederholen oder für die Genehmigung durch eine Managerin oder einen Manager pausieren muss, bietet eine Workflow-Engine (wie Airflow, AWS Step Functions oder eine gut strukturierte Reihe von Skripts) eine klarere Kontrolle über Fehlerpfade als ein LLM. Deterministische Workflows sind immer dann sinnvoll, wenn Sie alle Entscheidungszweige im Voraus aufzählen können und eine strenge, überprüfbare Kontrolle über jeden Zweig benötigen. In derartigen Szenarios lassen

3 RAG – Retrieval-Augmented Generation: ein Softwaresystem, das das Abrufen von Informationen mit einem LLM kombiniert

sich Workflows natürlicher skalieren als große Ad-hoc-Skripte, vermeiden aber dennoch die Komplexität und die Kosten einer agentenbasierten Pipeline.

Herkömmliche Chatbots oder RAG-Systeme liegen auf der nächsten Komplexitätsschicht: Sie bieten das Verständnis natürlicher Sprache und das Abrufen von Dokumenten, reichen jedoch nicht bis zur autonomen, mehrstufigen Planung. Sollten Sie vor allem daran interessiert sein, Benutzerinnen und Benutzern die Möglichkeit zu geben, Fragen zu einer Wissensdatenbank zu stellen – beispielsweise als Suche in einem Produkthandbuch, einem juristischen Archiv oder in Unternehmens-Wikis –, kann ein RAG-System Dokumente in einen Vektorspeicher einbetten, relevante Passagen als Antwort auf eine Abfrage abrufen und kohärente, kontextbezogene Antworten generieren. So könnte ein interner IT-Helpdesk mithilfe von RAG die Frage: »Wie setze ich meine VPN-Anmeldedaten zurück?« beantworten, indem er den aktuellen Leitfaden zur Fehlerbehebung abruft und die relevanten Schritte zusammenfasst. Im Gegensatz zu autonomen Agenten treffen RAG-Systeme keine eigenständigen Entscheidungen über Folgeaktionen (wie zum Beispiel ein Ticket erstellen oder einen Rückruf zu planen) – sie stellen lediglich Informationen bereit. Ein herkömmlicher Ansatz mit Chatbot oder RAG ist sinnvoll, wenn in erster Linie Fragen zu strukturierten oder unstrukturierten Inhalten zu beantworten sind, wobei man im Wesentlichen ohne externe API-Aufrufe oder Entscheidungsorchestrierung auskommt. Die Wartungskosten sind geringer als bei Agenten – Ihr Mehraufwand ist vor allem darauf beschränkt, Dokumenteinbettungen auf dem neuesten Stand zu halten und Prompts zu optimieren –, doch Sie opfern die Fähigkeit des Agenten, mehrstufige Workflows zu planen oder aus Feedback-Schleifen zu lernen.

Schließlich kommen wir zu autonomen Agenten – Situationen, in denen weder einfacher Code noch starre Workflows oder RAG genügen, weil die Eingaben unstrukturiert, neuartig oder sehr variabel sind und weil Sie eine dynamische, mehrstufige Planung oder kontinuierliches Lernen aus Feedback benötigen. Betrachten wir ein Kundensupport-Center, das formlose E-Mails mit Problemen erhält, die von »Meine Laptop-Batterie bläht sich auf und könnte explodieren.« bis zu »Mir werden ständig Dienstleistungen in Rechnung gestellt, die ich gar nicht in Anspruch genommen habe.« reichen. Ein regelbasierter Workflow oder eine RAG-gestützte FAQ-Suche würde unter solcher unbegrenzten Vielfalt zusammenbrechen, aber ein Agent, der auf einem Basismodell beruht, kann die Absicht analysieren, relevante Elemente extrahieren, eine Wissensbasis konsultieren, eine angemessene Antwort entwerfen und bei Bedarf sogar an einen Menschen weiterleiten – und das alles, ohne dass ihm jede mögliche Verzweigung im Voraus mitgeteilt wird. Ähnlich verhält es sich im Lieferkettenmanagement: Ein Agent, der Lagerbestandsdaten, Lieferzeiten der Anbieter und Umsatzprognosen in Echtzeit erfasst, kann Versandpläne dynamisch neu planen, während ein deterministischer Workflow ständig manuelle Aktualisierungen erfordern würde, um Abweichungen zu berücksichtigen.

Agenten sind auch dann im Vorteil, wenn viele Teilaufgaben parallel laufen müssen – beispielsweise ein Agent für Sicherheitsoperationen, der gleichzeitig APIs für Bedrohungsinformationen abfragt, die Telemetrie des Netzwerks scannt und Sandbox-Analysen verdächtiger Binärdateien durchführt. Da Agenten asynchron arbeiten und anhand von Echtzeitdaten neu priorisiert werden, vermeiden sie die starre »ein Schritt nach dem anderen«-Natur von

Workflows oder RAG-Systemen. Um die höheren Rechen- und Wartungskosten für den Betrieb eines Basismodells zu rechtfertigen, benötigen Sie dieses Maß an kontextbezogenem Denken, paralleler Aufgabenorchestrierung oder kontinuierlicher Selbstverbesserung – Szenarios, in denen starre Codes, Workflows oder Chatbots zu unflexibel oder zu teuer in der Wartung wären.

Tabelle 1.2: Unterscheidung zwischen herkömmlichem Code, Workflows und Agenten

Eigenschaft	Herkömmlicher Code	Workflow	Autonomer Agent
Eingabestruktur	Vollständig vorhersagbare Schemas	Meistens vorhersehbar bei endlichen Verzweigungen	Hochgradig unstrukturierte oder neuartige Eingaben
Erklärbarkeit	Volle Transparenz; leicht überprüfbar	Expliziter Prüfpfad für jeden einzelnen Zweig	Blackbox-Komponenten, die zusätzliche Tools erfordern
Latenz	Extrem niedrige Latenz	Mittlere Latenz	Höhere Latenz
Anpassungs- und Lernfähigkeit	Keine	Begrenzt	Hoch (Lernen aus Feedback)

Jeder Weg hat Vor- und Nachteile. Reiner Code ist kostengünstig und schnell, aber unflexibel; Workflows bieten Kontrolle, versagen jedoch, wenn die Eingaben stark variieren; herkömmliche Chatbots oder RAG eignen sich hervorragend für die Beantwortung von Fragen zu Dokumenten, können jedoch keine mehrstufigen Aktionen koordinieren; und Agenten sind leistungsstark, aber anspruchsvoll – sowohl in Bezug auf Cloud Computing als auch auf den technischen Aufwand für Überwachung, Optimierung und Steuerung. Bevor Sie sich entscheiden, sollten Sie sich die folgenden Fragen stellen: Sind meine Eingaben unstrukturiert oder unvorhersehbar? Benötige ich eine mehrstufige Planung, die sich an Zwischenergebnisse anpasst? Genügt ein Dokumentenabrufsystem für die Informationsbedürfnisse meiner Benutzerinnen und Benutzer oder muss das System autonom entscheiden und handeln? Möchte ich, dass sich dieses System im Laufe der Zeit mit minimalem menschlichen Eingriff selbst verbessert? Und kann ich die Latenz und den Wartungsaufwand eines Basismodells tolerieren?

Kurz gesagt: Wenn Ihre Aufgabe eine feste, deterministische Transformation ist, schreiben Sie einfach normalen Code. Gibt es eine Handvoll bekannter Verzweigungen und Sie benötigen explizite Fehlerbehandlungs-Checkpoints, verwenden Sie einen deterministischen Workflow. Wenn Sie in erster Linie ein Frage-Antwort-System mit natürlicher Sprache für einen bestimmten Korpus benötigen, nehmen Sie eine herkömmliche Chatbot- oder RAG-Architektur. Haben Sie jedoch mit hoher Variabilität, unbeschränktem Schlussfolgern, Anforderungen an dynamische Planungen oder der Notwendigkeit für kontinuierliches Lernen zu tun, investieren Sie in einen autonomen Agenten. Treffen Sie diese Wahl wohlüberlegt, um sicherzustellen, dass Sie das richtige Gleichgewicht zwischen Einfachheit, Performance und Anpassungsfähigkeit finden – so bleibt Ihre Lösung sowohl effektiv als auch wartungsfreundlich, wenn sich die Anforderungen weiterentwickeln.

Grundsätze für den Aufbau effektiver agentenbasierter Systeme

Um erfolgreiche autonome Agenten zu erstellen, ist ein Ansatz erforderlich, der Skalierbarkeit, Modularität, kontinuierliches Lernen, Resilienz und Zukunftssicherheit priorisiert:

Skalierbarkeit
: Stellen Sie sicher, dass Agenten mit wachsenden Arbeitslasten und vielfältigen Aufgaben klarkommen, indem sie verteilte Architekturen, cloudbasierte Infrastrukturen und effiziente Algorithmen einsetzen, die parallele Verarbeitung und Ressourcenoptimierung unterstützen. Beispiel: Ein Kundensupport-Agent, der 10 Tickets pro Minute verarbeitet, kann abstürzen oder hängenbleiben, wenn der Datenverkehr auf 1.000 ansteigt, sofern er nicht durch eine automatisch skalierende Infrastruktur unterstützt wird.

Modularität
: Entwerfen Sie Agenten mit unabhängigen, austauschbaren Komponenten, die über klare Schnittstellen miteinander verbunden sind. Dieser modulare Ansatz vereinfacht die Wartung, fördert die Flexibilität und erleichtert eine schnelle Anpassung an neue Anforderungen oder Technologien. Beispiel: Ein schlecht modularisierter Agent, der alle seine Tools in seinem Agentendienst fest codiert, würde jedes Mal eine komplette Neubereitstellung erfordern, selbst wenn an einem Tool nur ganz wenig ergänzt oder modifiziert werden muss.

Kontinuierliches Lernen
: Statten Sie Agenten mit Mechanismen wie zum Beispiel kontextbezogenem Lernen aus, damit sie aus Erfahrungen lernen können. Beziehen Sie Benutzerfeedback ein, um das Verhalten der Agenten zu optimieren und den Performancebezug Relevanz aufrechtzuerhalten, wenn sich die Aufgaben weiterentwickeln. Beispiel: Agenten, die Feedback-Schleifen ignorieren, machen möglicherweise immer wieder die gleichen Fehler – wie die falsche Klassifizierung von Vertragsklauseln oder das Versäumnis, kritische Supportprobleme an eine höhere Stelle weiterzuleiten.

Resilienz
: Entwickeln Sie robuste resiliente Architekturen, die Fehler, Sicherheitsbedrohungen, Zeitüberschreitungen und unerwartete Bedingungen elegant bewältigen können. Schließen Sie eine umfassende Fehlerbehandlung, strenge Sicherheitsmaßnahmen und Redundanz ein, um einen zuverlässigen und kontinuierlichen Agentenbetrieb zu gewährleisten. Beispiel: Agenten ohne Wiederholungs- oder Fallback-Logik können schon komplett abstürzen, wenn ein einziger API-Aufruf scheitert, sodass die Benutzerin oder der Benutzer warten muss und verwirrt ist.

Zukunftssicherheit
: Bauen Sie Agentensysteme auf offenen Standards und einer skalierbaren Infrastruktur auf, was eine Innovationskultur fördert, um sich schnell an neue Technologien und sich ändernde Benutzererwartungen anzupassen. Beispiel: Eine enge Kopplung Ihres Agenten an das proprietäre Prompt-Format eines bestimmten Anbieters kann es erschweren, zwischen Modellen zu wechseln, und dadurch Experimente beschränken.

Wenn sich Unternehmen an diese Prinzipien halten, können sie autonome Agenten entwickeln, die effektiv und relevant bleiben und sich nahtlos an technologische Fortschritte und sich ändernde Betriebsumgebungen anpassen.

Den Aufbau von Agentensystemen erfolgreich organisieren

Die weit verbreitete Verfügbarkeit von Basismodellen über einfache API-Aufrufe hat in vielen Unternehmen zu umfangreichen Experimenten mit Agentensystemen geführt. Teams führen häufig unabhängige Proofs of Concept durch, die wertvolle Erkenntnisse und innovative Ideen hervorbringen. Diese einfache Experimentierbarkeit resultiert jedoch oftmals in einer Fragmentierung – sich überschneidende Projekte, doppelte Anstrengungen und unvollendete Experimente sind dann überall im Unternehmen verstreut. Umgekehrt könnte eine verfrühte Standardisierung die Kreativität ersticken und Unternehmen in starre Rahmenbedingungen oder herstellerspezifische Lösungen zwängen. Um Erfolg zu haben, müssen Sie ein Gleichgewicht zwischen Flexibilität für Experimente und ausreichender Abstimmung für Skalierbarkeit und Kohärenz finden.

In den frühen Phasen der Agentenentwicklung sollten Unternehmen die explorativen Bemühungen aktiv fördern und Teams die Möglichkeit geben, verschiedene Architekturen, Workflows und Modelle frei zu testen. Wenn sich im Laufe der Zeit erfolgreiche Muster und Best Practices herauskristallisieren, wird die strategische Abstimmung entscheidend. Die Implementierung einer Strategie »ein Standard pro große Gruppe« kann dieses Erfordernis effektiv ausgleichen. Innerhalb bestimmter Abteilungen oder Funktionsbereiche können Teams gemeinsame Tools und Methodologien standardisieren und so die Zusammenarbeit optimieren, ohne die Innovation im gesamten Unternehmen einzuschränken.

Ein weiterer wesentlicher Aspekt für den Erfolg ist es, Abhängigkeiten von bestimmten Anbietern zu vermeiden, was sich mit der Annahme offener Standards wie zum Beispiel OpenAPI und durch einen modularen Systementwurf erreichen lässt. Diese Praktiken begünstigen flexible Konzepte und verringern die Abhängigkeit von einer einzelnen Technologie oder einem einzelnen Anbieter, was die zukünftige Anpassungsfähigkeit erleichtert.

Ebenso entscheidend ist ein effektiver Wissensaustausch. Die sowohl aus erfolgreichen als auch aus erfolglosen Experimenten gewonnenen Erkenntnisse sollten über interne Foren, gemeinsame Repositorys und umfassende Dokumentationen breit kommuniziert werden. Dieser kollaborative Ansatz beschleunigt das Lernen in der Organisation, minimiert redundante Aufwendungen und fördert die kollektive Verbesserung.

Schließlich sollten Governance-Frameworks leichtgewichtig und flexibel bleiben, wobei sie die Leitprinzipien gegenüber starren Vorgaben bevorzugen. Eine rationelle Governance-Struktur ermöglicht es Teams, selbstbewusst innovativ zu sein und gleichzeitig auf die übergeordneten Unternehmensziele ausgerichtet zu bleiben.

Eine erfolgreiche Organisation rund um agentenbasierte Systeme läuft prinzipiell iterativ ab. Organisationen müssen ihre Strategien ständig neu bewerten, um ein dynamisches Gleichgewicht zwischen Exploration und Standardisierung aufrechtzuerhalten. Indem sie

eine Umgebung kultivieren, die Experimente, kollaboratives Lernen und offene Standards wertschätzt, können Organisationen agentenbasierte Systeme effektiv von isolierten Experimenten in skalierbare, transformative Lösungen überführen, die tief in ihre Betriebsprozesse integriert sind.

Agentenorientierte Frameworks

Derzeit gibt es zahlreiche Frameworks für die Entwicklung autonomer Agenten, die sich jeweils mit speziellen Funktionen wie der Integration von Fähigkeiten, Speicherverwaltung, Planung, Orchestrierung, Lernen aus Erfahrungen und Multiagentenkoordinierung befassen. Diese Liste ist sicherlich nicht vollständig, führt aber die führenden Frameworks auf.

LangGraph

Stärken
: Modulares Orchestrierungs-Framework auf Basis gerichteter Graphen, deren Knoten diskrete Logikeinheiten (oftmals Aufrufe des Basismodells) enthalten und deren Kanten den Datenfluss durch komplexe, potenziell zyklische Workflows verwalten; strenge Entwicklerergonomie; native Unterstützung für asynchrone Workflows und Wiederholungsversuche.

Kompromisse
: Erfordert benutzerdefinierte Logik für erweiterte Planung und Speicherung; weniger integrierte Unterstützung für die Zusammenarbeit mehrerer Agenten.

Am besten geeignet für
: Teams, die robuste Singleagenten- oder einfache Multiagentensysteme mit expliziter, überprüfbarer Flusssteuerung entwickeln

AutoGen

Stärken
: Leistungsstarke Multiagentenorchestrierung, dynamische Rollenzuweisung, flexible, nachrichtenbasierte Interaktion zwischen Agenten

Kompromisse
: Kann für einfache Anwendungsfälle zu schwerfällig oder komplex sein; eher eigenwillig in Bezug auf Interaktionsmuster zwischen Agenten

Am besten geeignet für
: Forschungs- und Produktionssysteme, die den Dialog zwischen mehreren Agenten beinhalten (zum Beispiel Manager-Mitarbeitende, Selbstreflexionsschleifen)

CrewAI

Stärken
: Einfach zu lernen und anzuwenden; schnelle Einrichtung für Prototyping; nützliche Abstraktionen wie »crew« und »tasks«

Kompromisse
: Begrenzte Anpassung und Kontrolle über die Interna der Orchestrierung; weniger ausgereift als LangGraph oder AutoGen für komplexe Workflows

Am besten geeignet für
: Entwicklerinnen und Entwickler, die schnell mit praktischen, auf den Menschen bezogenen Agenten wie Assistenten oder Support-Agenten beginnen möchten

OpenAI Agents Software Development Kit (SDK)

Stärken
: Tiefe Integration in das Tool-Ökosystem von OpenAI; sichere und benutzerfreundliche Funktionsaufrufe, Speicherprimitive und Tool-Routing

Kompromisse
: Eng an die Infrastruktur von OpenAI gekoppelt; möglicherweise weniger flexibel oder portabel für Stacks von benutzerdefinierten Agenten oder Open-Source-Toolchains

Am besten geeignet für
: Teams, die bereits mit der OpenAI-API arbeiten und nach einer schnellen Möglichkeit suchen, sichere, Tool-basierte Agenten mit minimalem Aufwand zu erstellen

Zwar zeichnet sich jedes Framework durch einzigartige Vorteile und Beschränkungen aus, doch ist davon auszugehen, dass kontinuierliche Innovation und Konkurrenz in diesem Bereich die weitere Entwicklung vorantreiben. Für frühe Prototypen können Sie mit CrewAI oder dem OpenAI Agents SDK schnell loslegen. Für skalierbare und produktionsreife Systeme bieten LangGraph und AutoGen mehr Kontrolle und Raffinesse. Diese Frameworks sind allerdings nicht unbedingt erforderlich, da viele Teams von vornherein direkt die APIs der Modellanbieter aufrufen. Dieses Buch konzentriert sich vor allem auf LangGraph, das aufgrund seines unkomplizierten und dennoch leistungsfähigen Ansatzes für die Entwicklung von Agentensystemen ausgewählt wurde. Mit detaillierten Erläuterungen, praktischen Beispielen und realen Szenarios zeigen wir, wie LangGraph effektiv die von modernen intelligenten Agenten geforderte Komplexität und Dynamik bewältigt.

Fazit

Autonome Agenten stellen eine transformative Entwicklung in der KI dar. Sie sind in der Lage, komplexe, dynamische Aufgaben mit einem hohen Grad von Autonomie auszuführen. Dieses Kapitel hat die grundlegenden Konzepte von Agenten skizziert und ihre praktischen Anwendungen und Beschränkungen erörtert. Je tiefer wir in das Design und die Implementierung dieser Systeme eintauchen, desto deutlicher wird, dass die durchdachte Integration von Agenten in verschiedene Bereiche das Potenzial hat, signifikante Innovation und Effizienz voranzutreiben.

Die in diesem Kapitel erörterten Ansätze für den Entwurf von autonomen Agenten haben zwar bedeutende Fähigkeiten und Möglichkeiten aufgezeigt, aber auch die Komplexität und die Herausforderungen hervorgehoben, die mit dem Erstellen von effektiven und anpassungsfähigen Systemen verbunden sind. Dabei bietet jede Methode – angefangen bei regel-

basierten Systemen bis hin zu fortschrittlichen kognitiven Architekturen – einzigartige Stärken, bringt aber auch gewisse Einschränkungen mit sich. In diesem Buch möchte ich diese Lücken schließen.