M10 - Agenten

Stand: 03.2025

1 Was sind KI-Agenten?

KI-Agenten sind eine Weiterentwicklung traditioneller KI-Systeme, insbesondere von Large Language Models (LLMs). Im Gegensatz zu reaktiven Chatbots können KI-Agenten **proaktiv und autonom Aufgaben übernehmen und ausführen**.

Laut den Texten unterscheidet Anthropic zwischen zwei Haupttypen:

- **Workflows**: Systeme mit vordefinierten Codepfaden, bei denen LLMs und Tools orchestriert werden
- **Agenten**: Systeme, in denen LLMs ihre Prozesse und Tool-Nutzung dynamisch steuern und selbstständig Aufgaben erfüllen

2 Kernkomponenten eines KI-Agenten

1. **Wahrnehmung**: Fähigkeit, Informationen aus der Umgebung aufzunehmen
2. **Gehirn**: Das LLM für logisches Denken und Schlussfolgerungen
3. **Aktionen**: Fähigkeit, Aktionen auszuführen (z.B. durch API-Aufrufe)
4. **Planung**: Möglichkeit, komplexe Aufgaben zu analysieren und Lösungsstrategien zu entwickeln
5. **Anpassungsfähigkeit**: Fähigkeit zur Weiterentwicklung durch Lernen

3 Muster für agentische Systeme

Anthropic identifiziert folgende gängige Muster:

1. **Augmentiertes LLM**: Ein LLM mit Abruf, Tools und Speicher
2. **Prompt Chaining**: Zerlegung von Aufgaben in Sequenzen
3. **Routing**: Klassifizierung und Weiterleitung von Eingaben an spezialisierte Module
4. **Parallelisierung**: Mehrere LLMs arbeiten gleichzeitig an einer Aufgabe
5. **Orchestrator-Worker**: Zentrales LLM verteilt Aufgaben an Worker-LLMs
6. **Evaluator-Optimizer**: Ein LLM generiert, ein anderes bewertet
7. **Agenten**: Eigenständige Planung, Tool-Verwendung und Feedback-Integration

4 Verfügbare Frameworks

Es gibt verschiedene Frameworks zur Entwicklung von KI-Agenten:

- **LangChain**: Populäres Framework mit vielen Tools und Integrationen
- **Semantic Kernel (Microsoft)**: SDK für LLM-Integration und zielorientierte Agenten
- **AGI2**: Open-Source-Framework für Multi-Agenten-Systeme
- **Swarm (OpenAI)**: Experimentelles Framework zur Orchestrierung
- **CrewAI**: Framework für rollenbasierte Multi-Agenten-Zusammenarbeit
- **smolagents**: Leichtgewichtige Bibliothek mit Fokus auf Open-Source-Modellen
- **AutoGPT**: Open-Source-Projekt für autonome Aufgaben

5 Praktische Anwendungsbereiche

KI-Agenten werden bereits in verschiedenen Bereichen eingesetzt:

- Kundenservice
- Gesundheitswesen
- Finanzwesen
- E-Commerce
- Forschung
- Unternehmensautomatisierung (Buchhaltung, Vertragsprüfung, Personalwesen)

6 Implementierungsempfehlungen

Basierend auf den Texten sind folgende Prinzipien wichtig:

1. **Einfachheit**: Mit der einfachsten Lösung beginnen und Komplexität nur bei Bedarf erhöhen
2. **Transparenz**: Klare Planungsschritte und nachvollziehbare Entscheidungen
3. **Strukturierte Schnittstellen**: Saubere Dokumentation und Tests
4. **Kontinuierliche Bewertung**: Iterative Verbesserung durch Feedback und Tests

7 Herausforderungen

Die Texte nennen folgende Herausforderungen:

- Variierende Genauigkeit der einzelnen Schritte
- Ethische Überlegungen bezüglich Arbeitsmarktauswirkungen
- Standardisierungsbedarf bei Schnittstellen
- Datenschutz und Sicherheit
- Transparenz und Erklärbarkeit

8 Beispiel: Einfacher KI-Agent

Hier ein einfaches Python-Beispiel, wie du mit LangChain einen KI-Agenten erstellen könntest:

```python
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI
from langchain.agents import AgentExecutor, create_openai_tools_agent
from langchain_community.tools.tavily_search import TavilySearchResults

# LLM initialisieren
llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

# Tool definieren (hier: Suchfunktion)
search_tool = TavilySearchResults(max_results=3)
tools = [search_tool]

# Prompt-Template erstellen
prompt = ChatPromptTemplate.from_template("""
Du bist ein hilfreicher Assistent, der Recherche-Aufgaben durchführt.
Nutze die bereitgestellten Tools, um die Anfrage des Nutzers zu beantworten:
{input}
""")

# Agent erstellen
agent = create_openai_tools_agent(llm, tools, prompt)
agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)

# Agent ausführen
def run_agent(query):
    result = agent_executor.invoke({"input": query})
    return result["output"]

# Beispielausführung
if __name__ == "__main__":
    user_query = "Was sind die neuesten Entwicklungen bei KI-Agenten?"
    result = run_agent(user_query)
    print(result)
```

9 Fazit

Fazit

KI-Agenten bieten großes Potenzial für die Automatisierung komplexer Aufgaben und die intelligente Verarbeitung von Informationen. Sie können das volle Potenzial von LLMs ausschöpfen und ermöglichen personalisierte, adaptive Interaktionen.

Wie von Anthropic empfohlen, ist es sinnvoll, mit einfachen Lösungen zu beginnen und diese schrittweise zu erweitern. Der Schlüssel zum Erfolg liegt nicht in der Komplexität, sondern in der Wahl des richtigen Systems für die jeweilige Anwendung.

Referenz:

Building Effective AI Agents | Anthropic \ Anthropic