Assignment
–
Retrieval-Augmented Generation (RAG) und Agentensysteme

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WI2022B

1 Theoretische Recherche und Einordnung

1.1 RAG-Systeme

Retrieval-Augmented Generation (RAG) stellt einen bedeutenden Fortschritt in der Verarbeitung natürlicher Sprache dar, insbesondere in ihrer Fähigkeit, die Leistung großer Sprachmodelle (LLMs) in einer Vielzahl von Anwendungen zu verbessern. Durch die Nutzung von Retrieval-Funktionen ermöglicht RAG es den Modellen, während der Textgenerierung auf externe Datenquellen zuzugreifen und so die in traditionellen LLMs inhärenten Einschränkungen wie faktische Ungenauigkeit, Inhaltsrelevanz und Aktualität zu adressieren. Diese Integration von Retrieval-Mechanismen befähigt Modelle dazu, präzisere, kontextuell relevantere und vertrauenswürdigere Ergebnisse zu erzeugen, indem sie dynamisch auf aktuelle und domänenspezifische Informationen verweisen.[1]

1.1.1 Architektur eines typischen RAG-Systems

Die Architektur eines typischen RAG-Systems besteht aus zwei Hauptkomponenten: einem Retrieval-Mechanismus und der Generierung. Die Voraussetzung dafür ist eine vorbereitete Wissensbasis, wie z.B. PDFs, Webseiten, Datenbanken oder Transkripte. Diese werden eingelesen und in kleinere Textsegmente zerlegt und anschließend mithilfe von Encoder-Modellen in eine Vektorrepräsentation überführt.[2] Bei einer Anfrage wird das vorliegende Dokument bzw. die Datenbank anhand des Retrieval-Moduls nach relevanten Textsegmenten durchsucht und diese zusammen mit der ursprünglichen Frage an das Sprachmodell (LLM) weitergegeben. Dies bildet den Teil der Generierung, bei dem durch die Verarbeitung des Prompts und der relevanten Textsegmente eine Antwort erstellt wird. Anschließend durchläuft das Ergebnis noch einen Post-Processing-Schritt, der sicherstellt, dass die Antwort z. B. hinsichtlich Konsistenz, Formatierung oder faktischer Genauigkeit überprüft und gegebenenfalls angepasst wird. In Abbildung 1 ist die Architektur nochmals vereinfacht dargestellt.[3]

[1] Unlu et al. 2024; Kelbert 2024
[2] Lee, Kim 2024
[3] Kelbert 2024; Dünschede o. J.

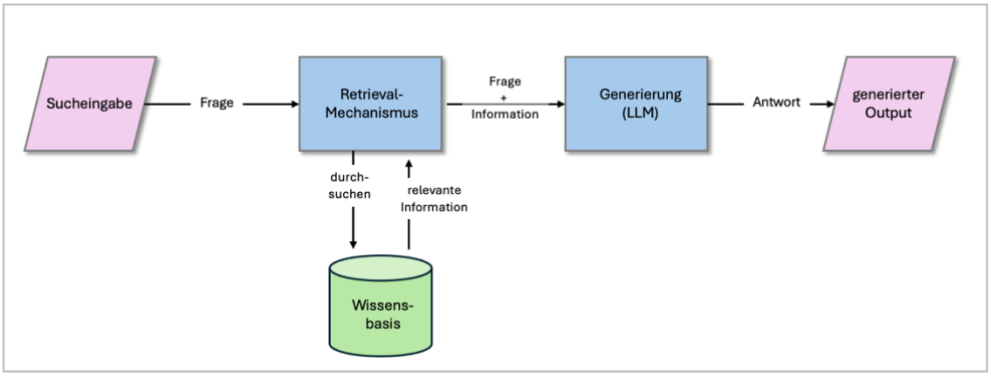

Abbildung 1: Achitektur eines RAG-Systems (vereinfacht)[4]

1.1.2 Unterschiede zu reinen generativen Modellen und klassischen QA-Systemen

RAG-Systeme unterscheiden sich sowohl von reinen generativen Modellen als auch von klassischen QA-Systemen, da es Elemente beider beinhaltet und auch deren Schwächen aufzeigt. Generative Modelle arbeiten ausschließlich mit dem internen, während des Trainings festgelegten Wissen. Sie reagieren daher nicht auf neue oder externe Informationen und neigen deshalb auch öfter zu Halluzinationen.[5] Klassische Frage-Antwort-Systeme hingegen basieren überwiegend auf „Information Retrieval“, also auf reiner Suchlogik, ohne dass neue Texte oder Zusammenfassungen generiert werden. Sie geben meist lediglich Textfragmente aus Dokumenten zurück.[6] RAG kombiniert diese Ansätze, indem es generative Elemente mit einem Retrieval aus Vektordatenbanken verbindet, statt sich allein auf das gespeicherte Wissen aus den Trainingsdaten eines LLMs zu stützen. Dadurch lassen sich kontextualisierte und aktuelle Antworten erzeugen.[7]

1.1.3 Beschreibung zentraler Komponenten

Embeddings und Vektordatenbanken: Embeddings übersetzen Wörter, Sätze oder ganze Dokumente in Vektoren, die semantische Zusammenhänge und Bedeutungen abbilden. Dadurch können RAG-Systeme nicht nur nach Schlagworten, sondern nach inhaltlicher Nähe suchen. Nutzeranfragen und Dokumente werden in diesem Vektorraum verglichen, sodass

[4] Mit Änderungen entnommen aus: Kelly 2025
[5] ambersearch.de 2023; Bröker 2025
[6] Selvaraj 2024
[7] Bröker 2025

relevante Textsegmente identifiziert und dem Sprachmodell als Kontext zur Verfügung gestellt werden.[8]

Vektordatenbanken bilden die Infrastruktur, um diese Embeddings effizient zu speichern und zu durchsuchen. Sie ermöglichen es, auch in großen Datenbeständen passende Informationen schnell zu finden und für die Generierung nutzbar zu machen.[9] Damit sind Embeddings und Vektordatenbanken zentrale Bausteine, die RAG-Systeme leistungsfähig und skalierbar machen.

Prompting und Kontextkonstruktion: Prompting spielt eine zentrale Rolle, da es die Anfrage so strukturiert, dass relevante externe Informationen in den Generierungsprozess einbezogen werden können. Durch gezieltes Prompt-Engineering lässt sich die Genauigkeit erhöhen, Halluzinationen werden reduziert und Antworten können besser auf die gewünschte Aufgabenstellung oder Rolle zugeschnitten werden. Die Kontextkonstruktion ergänzt diesen Prozess, indem abgerufene Dokumentfragmente in den Prompt integriert und dem Modell als zusätzliche Evidenz bereitgestellt werden. Damit wird sichergestellt, dass die Antworten nicht allein auf Trainingswissen beruhen, sondern durch aktuelle und nachvollziehbare Informationen gestützt werden.[10]

Modellwahl und Kostenaspekte: Die Architektur von RAG erlaubt den flexiblen Einsatz unterschiedlicher Sprachmodelle. Große Modelle (z. B. GPT-4) eignen sich für komplexe generative Aufgaben, während kleinere spezialisierte Modelle kosteneffizient für Zusatzfunktionen wie Klassifikation oder Relevanzbewertung eingesetzt werden können. Ein Vorteil von RAG ist, dass aktuelle Informationen dynamisch eingebunden werden, ohne das Hauptmodell ständig nachtrainieren zu müssen.[11] Kosten entstehen vor allem durch Modellgröße, Anzahl der Retrieval-Schritte, Kontextlängen sowie die Infrastruktur für Vektordatenbanken und Embeddings. Im Vergleich zum klassischen Finetuning großer Modelle ist RAG jedoch ressourcenschonender, da nur relevante Informationen abgerufen und verarbeitet werden.[12]

1.1.4 Herausforderungen bei der Umsetzung

RAG-Systeme können dennoch Herausforderungen mit sich bringen. Eine davon ist die Skalierbarkeit, denn diese wird eingeschränkt bei großen Datenmengen sowie deren Erfassung, Aktualisierung und Verwaltung. Dies führt zu einer schlechteren Datenqualität und damit zu längeren Antwortzeiten. Auch Latenzen können ein Problem darstellen, insbesondere

[8] data2type.de O.J.; Kelbert 2024
[9] Jonas 2024
[10] sap.com O. J.; Diehl 2025
[11] Sändig O. J.; Fachhochschule Graubünde 2024; Bröker 2025
[12] data2type.de O.J.

bei der Durchsuchung großer Datenmengen und der Berechnung von Embeddings. Verfügen Systeme nicht über die erforderliche Leistungsfähigkeit, kann dies zu weiteren Problemen führen. Eine weitere Herausforderung ist die begrenzte Kontrolle über Retrieval und Generierung, was es schwierig macht, Genauigkeit und Konsistenz zu gewährleisten.[13]

1.1.5 Bewertung und Vergleich zweier Open-Source-RAG-Frameworks

LlangChain vs. LLamaIndex

Diese beiden RAG-Frameworks unterscheiden sich in mehreren Kategorien. LangChain ist ein Framework, das den Aufbau verschiedenster KI-Anwendungen mit großen Sprachmodellen unterstützt, wie zum Beispiel Chatbots oder komplexe Verarbeitungsketten. LlamaIndex dagegen zeichnet sich dadurch aus, dass es große Datenmengen so aufbereiten und durchsuchen kann, dass relevante Informationen schnell gefunden und für RAG-Anwendungen genutzt werden können. Beim Abruf von Informationen kombiniert LangChain Such- und Generierungsfunktionen, um kontextbewusste Antworten zu erzeugen. LlamaIndex setzt vor allem auf semantische Suche, um besonders schnell passende Dokumente zu liefern. Auch in der Handhabung unterscheiden sich beide: LangChain ist meist einfacher einzurichten und bietet breite Unterstützung durch APIs, während LlamaIndex stärker auf Leistung bei großen Datenbeständen ausgelegt ist, aber etwas mehr Konfiguration erfordern kann.[14]

LangChain eignet sich vor allem für Entwickler, die flexible und vielseitige Anwendungen mit großen Sprachmodellen umsetzen möchten. LlamaIndex überzeugt vor allem durch seine Spezialisierung auf das schnelle und effiziente Verarbeiten sowie Abrufen großer Datenmengen und bietet dadurch klare Vorteile in datenintensiven RAG-Szenarien, ist jedoch weniger breit einsetzbar.

[13] Eruysaliz 2024b; Bruckhaus 2024
[14] Mavroudis 2024; Baghel 2025

1.2 Agentensysteme

1.2.1 Was sind Agenten?

Im Kontext der Large-Language-Modelle (LLM) wird ein Agent als „denkfähige“, autonome Einheit verstanden, die in der Lage ist, Eingaben zu verarbeiten, mehrere Schritte zu planen und zu koordinieren, Entscheidungen zu treffen und komplexe Aufgaben auszuführen. Er besitzt zudem die Fähigkeit, seine Handlungen an die Ergebnisse vorheriger Schritte anzupassen. Im Gegensatz zu einfachen Prompt-Ketten, die aus sequenziellen Aufrufen von LLMs in fester Reihenfolge bestehen, verfügen Agenten über Entscheidungslogik. Dadurch können Agentensysteme eine Vielzahl von Tools oder APIs nutzen, um externe Daten abzurufen, Berechnungen durchzuführen oder eigene Aktionen einzuleiten.

Darüber hinaus nutzen Agenten Speichermechanismen, um Kontextinformationen abzulegen und in späteren Interaktionen wiederzuverwenden. Dies erleichtert die Schaffung lernender und personalisierter Benutzererfahrungen.[15]

1.2.2 Zentrale Fähigkeiten von Agentensystemen

Tool-Nutzung:

Eine der Fähigkeiten, die Agentensysteme von anderen Ansätzen abhebt, ist die Integration externer Tools und APIs. Dadurch können sie über die reine Textgenerierung hinaus auch Aufgaben wie Berechnungen ausführen. Sie greifen auf unterschiedliche Ressourcen zu, um komplexe Workflows zu bewältigen. Die Tool-Nutzung ist somit eine Kernfunktion von Agenten und erweitert ihre Handlungsmöglichkeiten.[16]

Orchestrierung:

Ein weiteres wichtiges Merkmal ist die Orchestrierung, also die Planung und Koordination mehrerer Schritte. Fortschrittliche Agentensysteme nutzen dazu explizite Planungsmechanismen nach dem Prinzip „Plan, Reason, Act“. Auf diese Weise können sie komplexe Aufgaben strukturierter bearbeiten, was zu verbesserter Zuverlässigkeit und Effizienz führt. Zudem ermöglicht die Orchestrierung das dynamische Anpassen von Handlungen auf Grundlage von Zwischenergebnissen oder Eingaben aus der Umgebung.[17]

Routing:

Darüber hinaus spielt Routing, also die Aufgabenverteilung an spezialisierte Subagenten, eine zentrale Rolle in Multi-Agenten-Systemen. Einzelne Subagenten übernehmen dabei Teilaufgaben, wodurch Synergien entstehen und komplexe Probleme modular gelöst werden

[15] Siebert 2025; Belcic, Stryker 2025; Li et al. 2024
[16] Mackay et al. 2025; Welling 2025
[17] Siebert 2025; Welling 2025

können. Routing kann somit die Automatisierung komplexer Geschäftsprozesse unterstützen.[18]

Memory, Planning, Reaktion auf Nutzerkontext:

Schließlich ist auch die Kombination von Memory, Planning und Kontextreaktion entscheidend. Memory-Systeme speichern relevante Informationen und frühere Interaktionen, sodass personalisierte und kontextbewusste Antworten über längere Zeiträume möglich sind. Neuere Ansätze modellieren Memory als dynamisches Gedächtnis mit selektiver Aktualisierung, das eng mit Planungsmechanismen gekoppelt ist. Dadurch können Agentensysteme auf Nutzerbedürfnisse reagieren und langfristig konsistente, individualisierte Unterstützung leisten.[19]

1.2.3 Aktuelle Ansätze zur Entwicklung von Agentensystemen mit LLMs

LLMs werden zunehmend in Agentenarchitekturen eingebunden. Parallel dazu gewinnen Multi-Agenten-Systeme (MAS) und Ansätze der sogenannten Agentic-AI an Bedeutung. In diesen Systemen arbeiten mehrere spezialisierte Agenten zusammen, teilen Aufgaben auf und koordinieren ihre Ergebnisse. Dadurch lassen sich Prozesse strukturierter gestalten und die Skalierbarkeit sowie die Fehlertoleranz verbessern.[20]

Neben der Verteilung von Aufgaben rücken auch neuro-symbolische Konzepte in den Vordergrund. Mit Large Action Models (LAM) werden symbolische Methoden mit neuronalen Modellen kombiniert. Dies erlaubt es, Handlungsschritte nicht nur sprachlich zu beschreiben, sondern auch in digitalen Umgebungen auszuführen und zu kontrollieren. Auf diese Weise können Agentensysteme eigenständig Abläufe realisieren.[21]

Darüber hinaus weisen Forschende auf das Potenzial kleinerer Sprachmodelle (SLMs) hin. Diese Modelle können für klar abgegrenzte Aufgaben eingesetzt werden, bei denen große Modelle nicht erforderlich sind. Der gezielte Einsatz von SLMs erhöht die Effizienz und senkt Kosten, während leistungsstärkere LLMs für anspruchsvolle Aufgaben bleiben. Agentensysteme entstehen damit zunehmend als hybride Architekturen, die große und kleine Modelle kombinieren und je nach Anforderung flexibel einsetzen.[22]

1.2.4 Typische Frameworks

Typische Frameworks für Agentensysteme bilden die Grundlage, um Software-Agenten zu entwickeln, indem sie zentrale Funktionen wie Gedächtnis, Planung, Entscheidungsfindung

[18] Siebert 2025
[19] Belcic, Stryker 2025; Welling 2025, S. 5
[20] Siebert 2025; Welling 2025
[21] Hille 2025
[22] Kemper 2025

und Lernfähigkeit bereitstellen. Moderne KI-Agenten-Frameworks wie LangChain, AutoGen und CrewAI erweitern dieses Konzept durch die Integration großer Sprachmodelle (LLMs), die Autonomie, Aufgabenkoordination und flexible Interaktionen ermöglichen.[23] In Tabelle 1 werden AutoGen und LangChain anhand verschiedener Kriterien miteinander verglichen.

Kriterium	**LangChain Agents**	**AutoGen**
Architektur	Modular, graphbasiert mit LangGraph für komplexe Multi-Agenten-Workflows, eigene Steuerung von Ressourcen und Ketten (Chains). Betonung auf Kontrolle und Überwachung von Zuständen.	Dialogorientierte Multi-Agenten-Architektur mit Fokus auf Konversation und Zusammenarbeit. Bietet Low-Code-Schnittstelle zur schnellen Agent-Entwicklung.
Modularität	Sehr hoch, mit über 600 Integrationen zu LLMs, Datenbanken, APIs und Tools. Flexibel, aber komplex und kann bei vielen Komponenten unübersichtlich werden.	Flexibel bezüglich Kombination verschiedener LLMs und Tools, aber Bibliothek klein und noch im Aufbau.
Typische Use-Cases	API-getriebene Assistenten, komplexe Datenabrufe, Retrieval-augmented Generation (RAG), umfassende Workflows, Chatbots mit verschiedenen Tools.	Multi-Agenten-Systeme für Code-Assistenz, Softwareentwicklung und kollaborative Aufgaben, exploratives Prototyping, Forschung.

[23] Sprick 2025

Kriterium	LangChain Agents	AutoGen
Technische Einstiegshürden	Steile Lernkurve durch komplexe Architektur und umfangreiche APIs. Dokumentation gut, aber Überangebot kann überfordern. Erfordert Erfahrung mit Graph- und Zustandsverwaltung.	Intuitiver Einstieg dank Low-Code-Interface, jedoch manuell zu designende Agenteninteraktionen und Orchestrierung erfordern Verständnis. Dokumentation noch in Wachstumsphase.

Tabelle 1: Vergleich LangChain vs. AutoGen[24]

[24] Lizaveta 2025

Literaturverzeichnis

Baghel, Vaishnavi 2025. *LlamaIndex vs LangChain: Key Differences, Features & Use Cases*, https://www.openxcell.com/blog/llamaindex-vs-langchain/ (Zugriff vom 26.8.2025).

Belcic, Ivan; Stryker, Cole 2025. *KI-Agenten im Jahr 2025: Erwartungen vs. Realität | IBM*, https://www.ibm.com/de-de/think/insights/ai-agents-2025-expectations-vs-reality (Zugriff vom 27.8.2025).

Bröker, Mena Marie 2025. *Retrieval Augmented Generation (RAG): Eine Brücke zwischen KI und internem Wissen*, https://blog.flavia-it.de/retrieval-augmented-generation-rag-eine-bruecke-zwischen-ki-und-internem-wissen/ (Zugriff vom 26.8.2025).

Bruckhaus, Tilmann 2024. *RAG Does Not Work for Enterprises*. arXiv.

Diehl, Andreas 2025. *12 Prompting Techniken für bessere Ergebnisse in der Arbeit mit LLMs*, https://digitaleneuordnung.de/blog/prompting-techniken (Zugriff vom 26.8.2025).

Dünschede, Hanna o. J. . *Wenn KI den Spickzettel zückt (1/2): Wie funktioniert Retrieval Augmented Generation (RAG)?*, https://blog.ordix.de/wenn-ki-den-spickzettel-zueckt-1 (Zugriff vom 26.8.2025).

Eruysaliz, Ismail 2024b. *Top 7 Herausforderungen bei der Retrieval-Augmented Generation*, https://www.valprovia.com/de/blog/top-7-herausforderungen-bei-der-retrieval-augmented-generation (Zugriff vom 26.8.2025).

Hille, Mona 2025. *Large Action Models – neue KI-basierte Interaktionsformen mit digitalen Technologien*, https://publikationen.bibliothek.kit.edu/1000179508 (Zugriff vom 27.8.2025).

Jonas, Tobias 2024. *Vektordatenbanken: Effiziente Datenverwaltung für KI*, https://innfactory.ai/ai/tech/vektordatenbanken-effiziente-datenverwaltung-fuer-moderne-ki-anwendungen/ (Zugriff vom 26.8.2025).

Kelbert, Thorsten Honroth, Dr Julien Siebert, Patricia 2024. *Retrieval Augmented Generation (RAG): Chat mit eigenen Daten*, https://www.iese.fraunhofer.de/blog/retrieval-augmented-generation-rag/ (Zugriff vom 26.8.2025).

Kelly, Conor 2025. *8 Retrieval Augmented Generation (RAG) Architectures You Should Know in 2025*, https://humanloop.com/blog/rag-architectures (Zugriff vom 26.8.2025).

Kemper, Jonathan 2025. *Nvidia-Forschende plädieren für mehr kleine Modelle in KI-Agenten*, https://the-decoder.de/nvidia-forschende-plaedieren-fuer-mehr-kleine-modelle-in-ki-agenten/ (Zugriff vom 27.8.2025).

Lee, Ha-rin; Kim, Seo-hyun 2024. *Bring Retrieval Augmented Generation to Google Gemini via External API: An Evaluation with BIG-Bench Dataset*. Research Square.

Li, Xinyi et al. 2024. »A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges«, in *Vicinagearth* 1, 1, S. 9.

Lizaveta, Artsyman 2025. *Autogen vs LangChain vs CrewAI: Our AI Engineers' Ultimate Comparison Guide*, https://www.instinctools.com/blog/autogen-vs-langchain-vs-crewai/ (Zugriff vom 27.8.2025).

Mackay, Sina et al. 2025. »Die nächste Generation der medizinischen KI-Assistenz. ›Healthcare Agents‹ im Einsatz«.

Mavroudis, Vasilios 2024. *LangChain v0.3*.

Selvaraj, Timo 2024. *RAG vs Traditional Question-Answering Systems: A Paradigm Shift in AI*, https://medium.com/@tselvaraj/rag-vs-traditional-question-answering-systems-a-paradigm-shift-in-ai-cc771af17fe4 (Zugriff vom 26.8.2025).

Siebert, Dr Julien 2025. *Agentic AI – Multi-Agenten-Systeme im Zeitalter generativer KI - Blog des Fraunhofer IESE*, https://www.iese.fraunhofer.de/blog/agentic-ai-multi-agenten-systeme/ (Zugriff vom 27.8.2025).

Sprick, Florian 2025. *KI-Agenten im Fokus: LangGraph, CrewAI und AutoGen – Die Unterschiede*, https://www.eoda.de/wissen/blog/ki-agenten-entwicklung-frameworks/ (Zugriff vom 27.8.2025).

Unlu, Ozan et al. 2024. *Retrieval Augmented Generation Enabled Generative Pre-Trained Transformer 4 (GPT-4) Performance for Clinical Trial Screening*. medRxiv.

Welling, Oliver 2025. *KI-Agenten & Multi-Agenten: Top Papers Mai 2025*, https://kinews24.de/ki-agenten-multi-agenten-top-arxiv-papers-mai-2025/ (Zugriff vom 27.8.2025).

2023. *Was ist Retrieval Augmented Generation (RAG)? Nutzen & Vorteile*, https://ambersearch.de/retrieval-augmented-generation/ (Zugriff vom 26.8.2025).

2024a. *Ein Blick hinter die Kulissen: Das neue Retrieval Augmented Generation (RAG) Framework von Gion Sialm*, https://blog.fhgr.ch/ai/ein-blick-hinter-die-kulissen-das-neue-retrieval-augmented-generation-rag-framework-von-gion-sialm/ (Zugriff vom 26.8.2025).

O. J. a. *Publikationen/RAG-Studienreihe/Teil2*, https://www.data2type.de/publikationen/rag-studienreihe/teil2, /publikationen/rag-studienreihe/teil2 (Zugriff vom 26.8.2025).

O. J. b. *Was ist Prompt Engineering? | SAP*, https://www.sap.com/germany/resources/what-is-prompt-engineering (Zugriff vom 26.8.2025).

O. J. c. *Optimierung von RAG-Systemen durch Selbstreflexion | KI Journal Club 06/24*, https://www.ontolux.de/blog/optimierung-von-rag-systemen-durch-selbstreflexion/ (Zugriff vom 26.8.2025).

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *[Thema einfügen]* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

_____ _____

(Ort, Datum) (Unterschrift)