

EDITH Publication

LARGE LANGUAGE MODELS LEITFADEN

**EINFÜHRUNG IN GROSSE
SPRACHMODELLE UND ERSTE
EINSATZMÖGLICHKEITEN
FÜR
PRODUZIERENDE
UNTERNEHMEN**



Impressum

Titel: Einführung in große Sprachmodelle und erste Einsatzmöglichkeiten für produzierende Unternehmen

Autoren:

Evrin Cicek, M.Sc.

Wissenschaftlicher Mitarbeiterin
Institut für Produktionsmanagement,
Technologie und Werkzeugmaschinen
Technische Universität Darmstadt
E.Cicek@PTW.TU-Darmstadt.de

Prof. Dr.-Ing. Joachim Metternich

Institut für Produktionsmanagement,
Technologie und Werkzeugmaschinen
Technische Universität Darmstadt
J.Metternich@PTW.TU-Darmstadt.de

Ansprechpartnerin:

Evrin Cicek

Wissenschaftliche Mitarbeiterin - Institut PTW, TU Darmstadt
T +49 6151 8229-612
E-Mail: E.Cicek@PTW.TU-Darmstadt.de

Vorwort

Dieses Dokument bietet eine fundierte Einführung in Sprachmodelle, international als Large Language Models (LLMs) bezeichnet, und erläutert methodische Ansätze zu ihrem Einsatz. Es vermittelt einen systematischen Überblick über die Methoden, die es ermöglichen, LLMs innerhalb von Unternehmen, insbesondere unter Berücksichtigung datenschutzrechtlicher Anforderungen, lokal zu nutzen und in potenziellen Anwendungsfällen einzusetzen.

Der Leitfaden versteht sich als orientierende Guidance für kleine und mittlere Unternehmen (KMU) und soll aufzeigen, welche Möglichkeiten LLMs für die Unternehmenspraxis in der Zukunft eröffnen können. Die vorgestellten Empfehlungen dienen als beispielhafte Optionen und erfordern eine individuelle Prüfung und Anpassung innerhalb des jeweiligen Unternehmens.

Die Darstellung erfolgt bewusst auf einer übersichtlichen, oberflächlichen Ebene, um die Komplexität zu reduzieren und den Einstieg in das Thema zu erleichtern. Sie ersetzt keine individuelle fachliche Beratung. Für die Nutzung der Inhalte wird keine Haftung übernommen

Inhaltsverzeichnis

Impressum	1
Vorwort	1
1. Einführung in die Generative KI	3
2. Was sind große Sprachmodelle?	4
2.1 Begriffseinführung und Pretraining	4
2.2 Transformerarchitektur und Entwicklung von Sprachmodellen	4
2.3 Leistungsfähigkeit, Limitationen und kritische Betrachtung	5
3. Erweiterte Methoden zur Nutzung von LLMs	5
3.1 Finetuning	6
3.2 Prompt Engineering	6
3.3 Retrieval Augmented Generation (RAG)	8
3.4 Agentische Systeme	9
3.4.1 Multiagenten und Zusammenarbeit	10
3.4.2 Herausforderungen und zukünftige Entwicklungen	11
4. Erste Herausforderung und Schritte zur Einführung von Sprachmodellen	11
4.1 Herausforderungen beim Einsatz von Sprachmodellen	11
4.2 Lokales Deployment von Sprachmodellen	13
5. Potenziale und Anwendungsfelder für die zukünftige Arbeitswelt	14
6. Fazit	16
Literaturverzeichnis	17

1. Einführung in die Generative KI

Menschen verfügen über die einzigartige Fähigkeit, sich durch Sprache auszudrücken und zu kommunizieren, eine zentrale Grundlage für zwischenmenschliche Interaktionen und den Wissensaustausch [1]. Die Nachbildung und Erweiterung solcher kognitiven Fähigkeiten in Maschinen stellt ein zentrales Ziel der Künstlichen Intelligenz (KI) dar, einem Teilgebiet der Informatik, das Systeme entwickelt, die Aufgaben übernehmen können, die üblicherweise menschliche Intelligenz erfordern, etwa Schlussfolgern, Lernen oder Problemlösen [2]. Intelligenz lässt sich dabei als die Fähigkeit definieren, aus Erfahrungen zu lernen, sich an neue Situationen anzupassen, abstrakte Konzepte zu verstehen und Wissen gezielt einzusetzen, um die Umwelt zu beeinflussen [3].

Ein besonders herausfordernder Bereich innerhalb der KI ist die Verarbeitung und Generierung menschlicher Sprache. Dieses Forschungsfeld wird durch das Natural Language Processing (NLP) abgedeckt, das darauf abzielt, Maschinen zu befähigen, Sprache zu verstehen, zu interpretieren und selbst zu erzeugen. Seit seinen Anfängen in den 1950er-Jahren hat NLP eine erhebliche Entwicklung durchlaufen: von regelbasierten Systemen der 1960er-Jahre, die auf manuell erstellten linguistischen Regeln beruhten, hin zu modernen, datengetriebenen Ansätzen. Heutige NLP-Systeme analysieren große Textmengen und unterscheiden dabei typischerweise zwischen Natural Language Understanding (NLU), der Extraktion semantischer Bedeutung, und Natural Language Generation (NLG), der kohärenten Sprachproduktion. [2]

Innerhalb dieses Kontextes haben sich Sprachmodelle als zentrale methodische Werkzeuge etabliert, um die Struktur und den Gebrauch von Sprache zu erfassen und zu generieren. Sie basieren auf der Modellierung von Wahrscheinlichkeiten über Wortsequenzen und bilden die Grundlage moderner NLP-Ansätze. Eine vertiefte Betrachtung dieser Modelle erfolgt in einem späteren Abschnitt. Die Leistungsfähigkeit solcher Ansätze bildet die Grundlage für einen weitergehenden Paradigmenwechsel hin zur Generativen Künstlichen Intelligenz (KI). Generative KI, ein Teilbereich des maschinellen Lernens, beschäftigt sich mit der Erstellung neuer Inhalte, darunter Text, Bilder, Audio, Video, Code und Simulationen [4]. Im Gegensatz zu diskriminativen Modellen, die auf Klassifikation und Vorhersage fokussiert sind, lernen generative Modelle die zugrundeliegende Wahrscheinlichkeitsverteilung von Daten, um neue, ähnliche Instanzen zu erzeugen [5]. Grundlage dieser Verfahren ist das Deep Learning, ein auf neuronalen Netzwerken basierender Teilbereich des maschinellen Lernens, der große Datensätze nutzt, um komplexe Muster über verschiedene Modalitäten hinweg zu erfassen [6, 7]. Ein bedeutender Fortschritt innerhalb der Generativen KI ist die Entwicklung multimodaler Modelle. Diese sind in der Lage, Informationen aus unterschiedlichen Datentypen gleichzeitig zu verarbeiten und zu erzeugen, wodurch der Anwendungsbereich von KI-Systemen über rein textbasierte Aufgaben hinaus erweitert wird [8]. Beispielsweise können sie visuelle und sprachliche Daten miteinander in Beziehung setzen und so eine gemeinsame Repräsentation ermöglichen [9–11]. Obwohl sie primär diskriminativ sind, bilden sie eine wichtige Grundlage für fortgeschrittene generative Systeme, da sie das Verständnis zwischen Modalitäten verbessern.

Aufbauend auf diesen Entwicklungen integrieren moderne Systeme wie GPT-4o sowohl generative als auch diskriminative Komponenten, um die Inhaltserstellung zu steuern und Informationen über mehrere Modalitäten hinweg zu verarbeiten [12, 13]. Obwohl sich aktuelle Anwendungen vor allem in der Informatik und Technologiebranche finden, reicht ihr Potenzial weit darüber hinaus. Jüngste Forschung zeigt vielversprechende Einsatzmöglichkeiten in Bereichen wie Gesundheit, Recht und Bildung, in denen die Fähigkeit zur Interpretation und Generierung multimodaler Daten zu transformativen Innovationen führen kann [9].

2. Was sind große Sprachmodelle?

2.1 Begriffseinführung und Pretraining

Große Sprachmodelle (engl. Large Language Models - LLMs) sind transformerbasierte neuronale Netzwerke, die menschliche Sprache verarbeiten, Texte generieren und auf unterschiedliche Aufgaben generalisieren können [14, 15]. Sie stellen die Weiterentwicklung von Pretrained Language Models (PLMs) dar [15].

Dabei ist vor allem das Pretraining (Vortraining) ein zentraler Schritt für die Leistungsfähigkeit von LLMs: Modelle werden auf umfangreichen Textkorpora trainiert, um statistische Zusammenhänge zwischen Tokens zu erfassen [8, 15]. Die Qualität und Vielfalt der Pretraining-Daten beeinflussen die spätere Modelleleistung erheblich, während die Kapazität des Modells maßgeblich von Umfang und Methoden des Trainings abhängt [8].

LLMs arbeiten auf der Ebene von Tokens, die Wörter, Zahlen oder Satzzeichen darstellen. Häufige Wörter bilden meist einzelne Tokens, während seltene oder längere Wörter in mehrere Tokens zerlegt werden (siehe Abb. 1) [16, 17]. Eingaben in Form von Prompts werden tokenisiert, und das Modell sagt iterativ die nächsten Tokens auf Grundlage des bisherigen Kontexts und der gelernten Wahrscheinlichkeitsverteilungen voraus. Dieses Verfahren wird als autoregressives Decoding bezeichnet, wobei die Wahrscheinlichkeit einer gesamten Token-Sequenz als Produkt der bedingten Wahrscheinlichkeiten jedes Tokens gegeben dem bisherigen Kontext dargestellt wird [18].

Beispielsatz

Welche fundamentale Rolle spielen Tokens, und welche komplexe Sprachmodell-Datenverarbeitung wird damit überhaupt unterstützt?

Mögliche tokenisierte Form

[Welche] [fundamentale] [Rolle] [spielen] [Tokens] [,] [und] [welche] [komplexe]
[Sprach] [modell] [-] [Daten] [verarbeitung] [wird] [damit] [überhaupt] [unterstützt] [?]

Abb. 1 Mögliche Tokenisierung anhand eines Beispielsatzes

Durch Pretraining auf großen Datensätzen werden PLMs zu LLMs, die viele Aufgaben direkt über Prompting lösen können [9, 19]. Modelle wie GPT-3 und GPT-4 zeigen diese Fähigkeit, und erreichen hervorragende Leistungen in Aufgaben wie maschineller Übersetzung, Fragebeantwortung oder Code-Generierung [9, 14]. Zudem treten mit zunehmender Modellgröße emergente Fähigkeiten auf, die kleinere Modelle nicht besitzen [9, 14].

2.2 Transformerarchitektur und Entwicklung von Sprachmodellen

LLMs basieren auf der Transformer-Architektur, einem 2017 eingeführten neuronalen Netzwerkframework, das eine zentrale Rolle in moderner NLP einnimmt [20, 21]. Ein zentrales Merkmal von Transformern ist der Self-Attention-Mechanismus, der es dem Modell ermöglicht, die Relevanz jedes Tokens in Bezug auf alle anderen Tokens in einer Sequenz zu gewichten, unabhängig von deren Position. Dadurch kann das Modell komplexe Zusammenhänge zwischen Wörtern erfassen und Sequenzen parallel verarbeiten, was die Effizienz erhöht und

die Modellierung langfristiger Abhängigkeiten gegenüber rekurrenten Ansätzen verbessert [20, 21].

Die Multi-Head-Attention erweitert dieses Prinzip, indem mehrere Self-Attention-Operationen parallel ausgeführt werden. Jeder „Head“ projiziert Queries, Keys und Values mittels erlernter linearer Transformationen in unterschiedliche Unterräume, sodass das Modell gleichzeitig verschiedene Arten von Wortbeziehungen berücksichtigen kann. Die Ergebnisse aller Attention-Heads werden zusammengeführt und erneut projiziert, um die finale Repräsentation zu bilden [20]. Das Stapeln mehrerer Self-Attention- und Feed-Forward-Schichten verstärkt zusätzlich die Fähigkeit des Modells, reichhaltige, kontextabhängige Merkmalsdarstellungen zu extrahieren [17].

Transformers folgen einer Encoder-Decoder-Struktur. Der Encoder wandelt Eingabesequenzen in kontinuierliche Vektor-Repräsentationen um, die den Kontext jedes Tokens abbilden. Der Decoder generiert die Ausgabesequenz, wobei er jedes nächste Token auf Grundlage des bisherigen Kontexts und der kodierten Repräsentation vorhersagt [17, 21]. Diese Architektur ermöglicht die Verarbeitung großer Textmengen und multimodaler Daten wie Bilder oder Audio [17]. Sie bildete die Grundlage für PLMs: Encoder-basierte Modelle wie BERT nutzen das Paradigma „Pre-Training gefolgt von Fine-Tuning“, während decoder-basierte Modelle wie GPT einen generativen Ansatz verfolgen [7, 22].

Aufbauend auf diesen Entwicklungen erweitern LLMs sowohl ihre Skalierung als auch ihre Leistungsfähigkeit, indem sie riesige Datensätze nutzen, um flexible und generalisierbare Sprachverständnis- und Sprachgenerierungsfähigkeiten zu erreichen. Durch die Kombination von Self-Attention, Multi-Head-Attention, gestapelten Schichten, der Encoder-Decoder-Struktur und selbstüberwachtem Lernen können LLMs komplexe Muster in menschlicher Sprache sowie multimodalen Daten modellieren [17, 23].

2.3 Leistungsfähigkeit, Limitationen und kritische Betrachtung

Die Leistung von LLMs hängt entscheidend von der Qualität des Pretrainings, der Größe der Trainingsdaten und der Anzahl der Modellparameter ab [8]. Mit zunehmender Modellgröße entstehen emergente Fähigkeiten, die kleinere Modelle nicht besitzen [9]. Gleichzeitig steigen Modellkomplexität und Hardwarebedarf [18].

Trotz dieser Fortschritte bleiben LLMs statistische Modelle. Sie erzeugen Texte auf Basis gelernter Wahrscheinlichkeiten, zeigen oberflächliches Verständnis und können abhängig vom Trainingsdatensatz inkonsistente oder fehlerhafte Ergebnisse liefern. Dazu gehören Halluzinationen, bei denen plausible Inhalte faktisch inkorrekt sind [24–26]. Daher ist es notwendig, die Ausgaben kritisch zu prüfen, insbesondere in spezialisierten oder dynamischen Anwendungsfeldern [17, 27, 28].

Zusammenfassend kombinieren LLMs transformerbasierte Architekturen, groß angelegtes Pretraining, statistische Token-Vorhersage und autoregressives Decoding. Dadurch ermöglichen sie leistungsstarke Textverarbeitung und -generierung, während ihre Limitationen sorgfältig berücksichtigt werden müssen.

3. Erweiterte Methoden zur Nutzung von LLMs

Dieses Kapitel erläutert, wie LLMs effizient eingesetzt werden können, um zielgerichtete Antworten und Outputs zu generieren. Die vorgestellten Ansätze reichen von ressourcenintensiven Methoden, wie dem Fine-Tuning, das die Modellparameter direkt beeinflusst, bis hin zu weniger aufwändigen Optimierungen, die dennoch signifikante Verbesserungen ermöglichen. Hierzu zählen insbesondere Prompt Engineering sowie die

Nutzung eigener Dokumente, auch in großen Mengen, um den Kontext der Modelle zu erweitern und die Relevanz der generierten Antworten zu erhöhen. Darüber hinaus wird gezeigt, wie LLMs über klassische Chatfunktionen hinaus komplexe Aufgaben im Unternehmen lösen können, indem sie in Agentensysteme integriert werden. Ziel des Kapitels ist es, die methodische Bandbreite der Nutzung aufzuzeigen, von einfachen Optimierungen bis hin zu fortgeschrittenen Integrationen, und damit Unternehmen konkrete Wege für einen Einsatz aufzuzeigen.

3.1 Finetuning

Wie zuvor erwähnt, entwickeln LLMs während der Vortrainingsphase (Pretraining) eine umfassende sprachliche Kompetenz, indem sie große und heterogene Textkorpora verarbeiten. Dabei erlernen sie grammatische Strukturen, semantische Muster, kontextabhängiges Verständnis sowie grundlegende Schlussfolgerungen und Commonsense-Reasoning [29, 30]. Dieses Vorwissen bildet die Basis für viele Anwendungen, stößt jedoch an seine Grenzen, wenn Fachwissen oder domänenspezifische Terminologie benötigt wird, wie sie etwa in Medizin, Recht, Technik oder Finanzwesen relevant ist [30].

Um diese Lücke zu schließen, werden LLMs aufgabenspezifisch angepasst, ein Prozess, der traditionell als Fine-Tuning bezeichnet wird. Dabei werden die Modellparameter mit kleineren, gezielt ausgewählten Datensätzen modifiziert, sodass die allgemeine Sprachkompetenz erhalten bleibt, während die Leistung in spezialisierten Aufgaben steigt [29, 30]. Fine-Tuning ist zentral für Anwendungen wie Textklassifikation, Zusammenfassungen, Frage-Antwort-Systeme oder Sentiment-Analyse [29, 31].

Der Unterschied zwischen Vortraining und Fine-Tuning liegt primär in Reichweite und Spezialisierung. Während das Vortraining auf umfangreichen Daten universelle Sprachmuster und semantische Strukturen internalisiert und das gesamte Netzwerk trainiert, fokussiert sich die aufgabenspezifische Anpassung auf Parameterbereiche, die für die jeweilige Aufgabe besonders relevant sind [32]. Vollständiges Fine-Tuning aktualisiert alle Modellparameter, was hohe Genauigkeit ermöglicht, jedoch sehr rechen- und speicherintensiv ist [32, 33]. Parameter-effiziente Ansätze, wie Adapter-basierte Methoden oder Low-Rank Adaptation (LoRA), verändern nur Teilbereiche oder niedrigdimensionale Subräume des Netzwerks [34]. Dies reduziert Rechen- und Speicheraufwand, ermöglicht dennoch eine effektive Spezialisierung und unterstützt die Anpassung auch bei kleineren Datenmengen [34]. Solche Techniken sind besonders relevant, wenn mehrere Domänen oder Aufgaben gleichzeitig adressiert werden sollen, da sie das Modell modular und speicherökonomisch erweitern.

Ergänzende Fine-Tuning-Strategien können die Leistung weiter steigern. Instruction-based Training nutzt gezielte Anweisungen, um das Modell während des Trainings auf bestimmte Aufgaben vorzubereiten, während Knowledge Distillation das Wissen großer Modelle auf kleinere, effizientere Architekturen überträgt und deren ressourcenschonenden Einsatz ermöglicht [29–31]. Beide Ansätze verändern aktiv die Modellparameter und erweitern so die Fähigkeiten der LLMs für spezialisierte Aufgaben, ohne dass das gesamte Modell von Grund auf neu trainiert werden muss.

3.2 Prompt Engineering

Prompt Engineering stellt ein eigenständiges Paradigma dar, das sich von klassischen Fine-Tuning-Ansätzen unterscheidet: Statt die Modellparameter zu verändern, liegt der Fokus auf

der gezielten Gestaltung der Eingaben [30, 34, 35]. Durch strategisches Formulieren von Prompts lassen sich vortrainierte Modelle effektiv dazu bringen, neue Aufgaben zu lösen, ohne dass eine zusätzliche Modellanpassung notwendig ist. Dies macht sie besonders anpassungsfähig für eine Vielzahl von Anwendungen [30, 34].

Dieser inputzentrierte Ansatz ermöglicht es Large Language Models, Zero-Shot- und Few-Shot-Lernen durchzuführen, bei dem das Modell aus minimalen oder sogar keinen aufgabenspezifischen Beispielen generalisiert [30, 34, 36]. Insbesondere Few-Shot-Prompting liefert anschauliche Beispiele (siehe Abb. 2 links), die das gewünschte Eingabe-Ausgabe-Verhalten demonstrieren, wodurch das Modell sein bereits vorhandenes Wissen nutzen kann, während es flexibel auf unterschiedliche Aufgabenvariationen reagiert [36].

Neben der Effizienz reduziert Prompt Engineering den Rechenaufwand im Vergleich zu vollständigem Fine-Tuning, da keine Modellparameter angepasst werden. Dies ermöglicht eine schnelle Aufgabenanpassung selbst in ressourcenbeschränkten Umgebungen [30, 34]. Diese Agilität ist entscheidend in dynamischen Szenarien, in denen sich Anforderungen schnell ändern und iterative Experimente zur Optimierung der Ergebnisse erforderlich sind [35].

Fortgeschrittene Prompting-Techniken wie Chain-of-Thought (CoT)-Reasoning erweitern die Fähigkeit von Modellen, komplexe Probleme zu lösen, indem sie durch explizite Zwischenschritte strukturiert werden (siehe Abb. 2 rechts) [37]. In Kombination mit iterativer Verfeinerung sowie Prompt-Chaining-Strategien erlauben diese Methoden, allgemeines Sprachverständnis mit aufgabenspezifischem Problemlösen zu verbinden. Dadurch wird die Lücke zwischen generischer Modellkompetenz und domänenspezifischen Anforderungen weiter reduziert [35–37].

Few-Shot Prompting

Deine Aufgabe ist es, Inhalte für unseren Partner {partner_name} zu erstellen. Hier ist etwas Information über den Partner: {partner_beschreibung}
Unten siehst du einige Beispiele von Inhalten, die wir in der Vergangenheit aus Meetings erstellt haben:

Beispiel 1:

Meeting: {{meeting_1}}
Inhalt: {{content_1}}

Beispiel 2:

Meeting: {{meeting_2}}
Inhalt: {{content_2}}

Erstelle nun Inhalte basierend auf folgendem Meeting:

Meeting: {{meeting_description}}
Inhalt:

Chain-of-Thought Prompting

Deine Aufgabe ist es zu analysieren, wie sich die Stromkosten im letzten Monat zusammengesetzt haben, basierend auf den Informationen im Kontext: {Kontext}

Vorgehensweise:

Lass uns Schritt für Schritt vorgehen:

1. Zielwert bestimmen: Prüfe den Gesamtwert der Stromkosten im Kontext.
2. Relevante Faktoren identifizieren: Suche nach Bereichen oder Geräten, die Energie verbrauchen.
3. Zusammenhänge herstellen: Überlege, wie jeder Faktor die Gesamtkosten beeinflusst.
4. Kostenanteile schätzen: Ordne jedem Faktor einen geschätzten Anteil zu und notiere Annahmen.
5. Analyse zusammenfassen: Fasse die größten Kostentreiber zusammen und notiere mögliche Optimierungen.

Abb. 2 Beispiel für Few-Shot [38] und Chain-of-Thought Prompting

Insgesamt ermöglicht Prompt Engineering die Entwicklung spezialisierter und zugleich flexibler KI-Systeme, ohne dass aufwendige Trainingsprozesse erforderlich sind. Dies ist insbesondere relevant für Anwendungen wie Konversationsagenten, oder explorative und

analytische Einsatzszenarien, in denen schnelle Anpassbarkeit und Effizienz entscheidend sind.

3.3 Retrieval Augmented Generation (RAG)

Fine-Tuning und Prompt Engineering helfen dabei, Sprachmodelle gezielt für bestimmte Aufgaben oder Fachgebiete zu optimieren. Beide Methoden nutzen jedoch nur das bereits im Modell vorhandene Wissen. Wenn es aber um aktuelle, umfangreiche oder überprüfbare Informationen geht, reichen diese Ansätze oft nicht aus. Um Sprachmodelle auch in solchen Fällen leistungsfähig zu machen, können externe Wissensquellen systematisch eingebunden werden. Genau hier kommt das Konzept der Retrieval-Augmented Generation (RAG) ins Spiel.

Retrieval-Augmented Generation (RAG) ist ein Architekturansatz, der große Sprachmodelle (LLMs) erweitert, indem externe Informationssuche mit generativer Textproduktion kombiniert wird, wodurch kontextbezogene und wissensbasierte Antworten erzeugt werden können [39–42]. Im Gegensatz zu klassischen Generationsmodellen, die nur auf intern gespeichertes Wissen zurückgreifen, folgt RAG einem Retrieve-then-Read-Prinzip: Zunächst wird eine Anfrage formuliert, die den Informationsbedarf einer Aufgabe, zum Beispiel einer Nutzerfrage, abbildet; anschließend werden relevante Inhalte aus einem externen Wissenskorpus abgerufen, bevor die Antwort generiert wird [39–41]. Diese Einbindung externer, überprüfbarer Informationen verbessert die Faktengenauigkeit, Aktualität und Relevanz der Antworten, während die generative Flexibilität erhalten bleibt [41, 42].

Ein RAG-System besteht typischerweise aus mehreren Komponenten (siehe Abb.3): einer Such- oder Retriever-Ebene, die relevante Informationen aus Datenbanken oder sonstigen Datenquellen abrufen, einer Integrationsschicht, die die abgerufenen Informationen konsolidiert, und einem Generator, meist ein LLM, das die finale Antwort erstellt [40]. Der Retriever wandelt sowohl die Nutzeranfrage als auch gespeicherte Wissens Elemente in Text-Embeddings um, hochdimensionale Vektorrepräsentationen, die die semantische Bedeutung der Texte erfassen [40, 42]. Auf dieser Grundlage erfolgt die semantische Suche, bei der Textsegmente auf Basis ihrer Ähnlichkeit zur Anfrage bewertet werden [42]. Die Ähnlichkeit wird meist mit Kosinus-Ähnlichkeit oder ähnlichen Metriken berechnet, wobei kleinere Winkel eine stärkere semantische Nähe anzeigen [42].

Um effizient zu arbeiten, werden große Dokumente in kleinere Einheiten unterteilt, das sogenannte Chunking [40–42]. Hierbei gibt es unterschiedliche Strategien:

- Naives Chunking: Dokumente werden in gleich große Segmente mit definierten Überlappungen geteilt, unabhängig von inhaltlichen Grenzen. Einfach umzusetzen, gut kontrollierbar für Speicher- und Rechenaufwand [41].
- Semantisches Chunking: Dokumente werden an inhaltlichen Übergängen segmentiert, sodass jedes Segment eine zusammenhängende Bedeutungseinheit enthält. Dies verbessert die Qualität des Retrievals [41].
- Rekursives oder regelbasiertes Chunking: Dokumente werden schrittweise in handhabbare Segmente zerlegt, um sowohl semantische Kohärenz als auch Verarbeitungseffizienz zu gewährleisten [42].

Die Auswahl der Vector Store-Implementierung ist für RAG-Systeme entscheidend. Beliebte Lösungen sind Faiss, bekannt für effiziente Ähnlichkeitssuche und mehrere Indexierungsstrategien für große Datenmengen, sowie Chroma, das zusätzliche Funktionen wie Metadatenfilterung und hybride Suche bietet und besonders für mittelgroße Datensätze nutzerfreundlich ist [41]. Moderne Vektordatenbanken verwenden Approximate-Nearest-

Neighbor-Suche, Indexierungstechniken wie HNSW oder IVF und ermöglichen eine schnelle semantische Suche in hochdimensionalen Embedding-Räumen [41]. Die Konfiguration von Chunking-Strategie, Embedding-Modell und Vector Store bestimmt die Balance zwischen Retrieval-Qualität (Relevanz der abgerufenen Informationen) und Retrieval-Effizienz (Geschwindigkeit und Ressourcenverbrauch) und ist damit ein entscheidender Faktor für die Praxis [40, 41].

Nach der Auswahl der Top-k relevantesten Segmente werden diese über verschiedene Fusionsstrategien in den Generationsprozess integriert, entweder direkt auf Eingabeebene, innerhalb latenter Repräsentationen oder auf Ausgabebene, um die Antwort robust und inhaltlich kohärent zu gestalten [40, 42].

Insgesamt ermöglicht RAG damit LLMs, komplexe und wissensintensive Anfragen präzise zu beantworten und aktuelle Unternehmensinformationen gezielt einzubeziehen. Daher wird dieser Ansatz zunehmend in Conversational AI, Wissensmanagement in Unternehmen und fachlichen Entscheidungsunterstützungssystemen eingesetzt [41, 42].

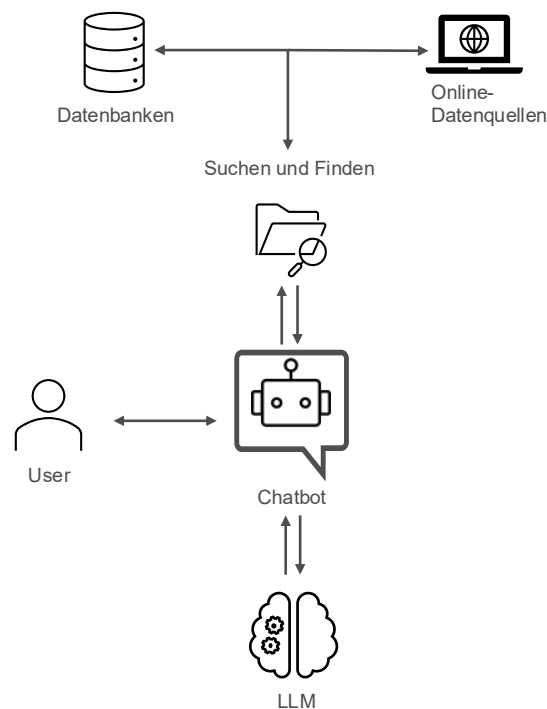


Abb. 3 Vereinfachter Wissensabruf- und Integrationsworkflow mithilfe von RAG

3.4 Agentische Systeme

RAG ermöglicht LLMs, nicht nur auf internes Wissen zurückzugreifen, sondern auch aktuelle und kontextspezifische Informationen zu integrieren [39–42]. Dadurch wird zwar die Anwendungsbreite der Modelle erweitert, doch bleibt ihre Funktionsweise meist reaktiv. Sie verarbeiten Eingaben, führen jedoch weder langfristige Strategien aus noch handeln sie eigenständig. Um LLMs für anspruchsvolle, dynamische Aufgaben weiterzuentwickeln, gewinnt die Erforschung agentenbasierter Systeme an Bedeutung. In solchen Architekturen übernehmen LLMs die Rolle zentraler kognitiver Instanzen [43].

Der rasche Fortschritt bei großen Sprachmodellen (LLMs) hat das Anwendungsspektrum der künstlichen Intelligenz erheblich erweitert und das Design sowie den Einsatz intelligenter Systeme neu gestaltet [43]. Frühe Implementierungen von LLMs nutzten überwiegend statische Interaktionsparadigmen, bei denen die Modelle Antworten auf isolierte Eingaben generierten, ohne kontinuierliches Schlussfolgern oder Interaktion mit dynamischen Umgebungen [44]. Obwohl solche Ansätze in der Sprachverständnis- und -generierung effektiv waren, reichten sie für komplexe, reale Aufgaben, die iteratives Denken, kontextuelle Anpassung und strukturierte Entscheidungsfindung erfordern, nicht aus [44].

Autonome Agenten werden definiert als Entitäten, die Umgebungszustände wahrnehmen, relevante kontextuelle Informationen interpretieren und Handlungen ausführen, um festgelegte Ziele zu erreichen [43]. Eine Kernfähigkeit ist die Planung, die das Sequenzieren von Aktionen über die Zeit sowie die Anwendung höherstufiger Schlussfolgerungen und adaptiver Entscheidungsprozesse umfasst [43].

Innerhalb solcher Rahmenwerke entwickeln sich LLMs über einfache Textgenerierungswerkzeuge hinaus zu Schlussfolgerungsmaschinen. Sie unterstützen strategisches Urteilsvermögen, Aufgabenkoordination und dynamisches Problemlösen [45]. Strukturierte Planungsmechanismen zerlegen komplexe Ziele in handhabbare Teilaufgaben und transformieren die direkte Eingabe-Ausgabe-Verarbeitung in durchdachte Workflows mit iterativer Verfeinerung und Zwischenschritten [44]. Persistente Gedächtnismodule ermöglichen es Agenten, kontextuelle Informationen über Interaktionen hinweg zu speichern, wodurch Kontinuität, kumulatives Lernen und individualisierte Reaktionen unterstützt werden [45].

Darüber hinaus können Agenten auf externe Werkzeuge, APIs und Datenquellen zugreifen, um Echtzeitinformationen abzurufen, operative Prozesse auszuführen und ihr Verhalten an sich verändernde Umgebungen anzupassen. Autonome Entscheidungsschleifen erlauben es Agenten, Zwischenergebnisse zu überwachen, Strategien zu verfeinern und Aufgaben fortzuführen, bis die Ziele erreicht sind. So wird mehrstufiges Problemlösen mit kontinuierlichem Schlussfolgern ermöglicht [43, 45].

3.4.1 Multiagenten und Zusammenarbeit

Während ein einzelner Agent bereits komplexe Aufgaben bearbeiten kann, stoßen einzelne Systeme bei heterogenen Problemfeldern an ihre Grenzen [46]. Einschränkungen in Skalierbarkeit, Effizienz und Robustheit haben die Einführung kollaborativer Multi-Agenten-Systeme vorangetrieben [44, 46]. In solchen Architekturen kommunizieren und koordinieren mehrere spezialisierte Agenten über strukturierte Protokolle und verteilen kognitive sowie operative Aufgaben auf modulare Komponenten [44, 46].

Die Zusammenarbeit mehrerer Agenten verbessert die Qualität des Schlussfolgerns, die faktische Konsistenz und die Gesamtleistung bei Aufgaben. Interaktionsmuster wie strukturierte Debatten oder die Aggregation unabhängig generierter Ergebnisse können die Entscheidungsfindung verbessern und Fehler reduzieren [46]. Durch die Nutzung komplementärer Expertise und Rollendifferenzierung erreichen Multi-Agenten-Systeme Aufgaben, die einzelnen Agenten verwehrt bleiben, insbesondere in Bereichen, die komplexe kognitive Aufgaben, Kreativität oder adaptives Problemlösen erfordern [44, 46].

Solche Architekturen werden zunehmend auf kontinuierliche Informationsgewinnung und kontextsensitive Anpassung angewendet, etwa in Empfehlungssystemen und Analyse-

Pipelines. Hierbei verarbeiten Agenten historische Daten iterativ, um maßgeschneiderte Ergebnisse zu erzeugen [45].

3.4.2 Herausforderungen und zukünftige Entwicklungen

Multi-Agenten-Architekturen bringen zusätzliche Komplexitäten mit sich, die über die Einschränkungen einzelner Agenten hinausgehen. Einige Herausforderungen resultieren aus intrinsischen Schwächen der Modelle, wie Halluzinationen oder unvollständiger Befolgung von Anweisungen [47], andere aus systembedingten Designfaktoren, etwa der Aufteilung von Workflows, unklaren Aufgabenbeschreibungen oder der Koordination mehrerer Agenten [44, 47].

Bei der Planung zerlegen einzelne Agenten Aufgaben in handhabbare Schritte und nutzen Werkzeuge oder externe Daten, um Wissenslücken zu schließen (Paper 8). Multi-Agenten-Systeme erweitern dies zu globaler Planung, bei der Teilaufgaben über Agenten verteilt werden, während Ausrichtung, Konsistenz und Effizienz gegenüber den übergeordneten Zielen gewahrt bleiben [44]. Auch das Gedächtnis-Management wird komplexer: Agenten müssen sowohl den internen Kontext als auch die Interaktionshistorien untereinander verfolgen, um Kontinuität, Situationsbewusstsein und iteratives Denken zu gewährleisten [44].

Koordinationsfehler bleiben eine zentrale Fehlerquelle. Fehlende Zielübereinstimmung, ineffektive Kommunikation oder inkonsistentes Schlussfolgern können die kollektive Aufgabenerfüllung beeinträchtigen [47]. Die Bewältigung dieser Herausforderungen erfordert integrierte Strategien, die Workflow-Design, Zwischenschritte beim Schlussfolgern, Konfliktlösung und mehrstufige Überprüfungen kombinieren, um Robustheit und Zuverlässigkeit sicherzustellen [44, 47].

Diese Fähigkeiten ermöglichen praktische Anwendungen in verteilten Systemen, etwa in Blockchain, wo Agenten komplexe Operationen oder Verhandlungen über mehrere Knoten ausführen [44]. Insgesamt zeigt die Entwicklung hin zu Multi-Agenten-Systemen, dass die Leistungsfähigkeit von KI zunehmend auf sorgfältig gestalteten Workflows, Gedächtnisstrukturen und Koordinationsmechanismen beruht und nicht allein auf den Modellfähigkeiten. Dies unterstreicht die Bedeutung modularer, kollaborativer und iterativer Architekturen für reale, mehrstufige Aufgaben [44, 47].

4. Erste Herausforderung und Schritte zur Einführung von Sprachmodellen

4.1 Herausforderungen beim Einsatz von Sprachmodellen

Die Nutzung von großen Sprachmodellen wie GPT bietet beeindruckende Möglichkeiten, birgt jedoch auch Risiken. Besonders kritisch ist die Offenlegung sensibler Daten: Eingaben können vertrauliche Informationen preisgeben, und unklare Datenverarbeitungsprozesse können unter Umständen gegen Datenschutzgesetze verstoßen [48]. Eine praktikable Alternative ist der Einsatz von lokalen LLMs. Diese Lösung bietet mehrere Vorteile, da alle Daten intern bleiben, wodurch Datenschutz und Sicherheit deutlich erhöht werden [49].

Außerdem lassen sich lokale Modelle an firmenspezifische Inhalte anpassen, was die Flexibilität und Relevanz der Ergebnisse steigert.

Gleichzeitig gibt es Herausforderungen bei der Implementierung. Die Anfangsinvestitionen sind hoch, da spezialisierte Hardware und Infrastruktur erforderlich sind [50]. Zudem erfordert der Betrieb kontinuierliche Wartung und regelmäßige Updates, um sowohl Leistung als auch Sicherheit sicherzustellen [50][50]. Das Erstellen eines LLMs komplett von Grund auf ist in der Regel nicht sinnvoll, da der Trainingsaufwand extrem hoch ist und oft Kosten in Millionenhöhe für GPU-Ressourcen verursacht. Effizienter ist es, leistungsfähige Open-Source-Modelle zu nutzen und diese durch Methoden wie Prompt Engineering, Retrieval-Augmented Generation (RAG) oder Fine-Tuning an die eigenen Anforderungen anzupassen.

Die Auswahl eines geeigneten LLM hängt maßgeblich vom konkreten Einsatzszenario ab. Zentrale Kriterien bei den Einsätzen von Sprachmodellen, die bisher in ersten Umsetzungen aufgefallen sind, umfassen unter anderem den Verwendungszweck, die unterstützten Sprachen, etwa Deutsch oder Englisch, die verarbeiteten Datentypen, also Text, Code oder multimodale Inhalte wie Text und Bild, die maximale Kontextgröße, also die Menge an Text, die das Modell simultan verarbeiten kann, sowie wirtschaftliche und regulatorische Aspekte, wie Kosten, Lizenzbedingungen und die Verfügbarkeit als kommerzielle oder Open-Source-Lösung.

Die Ausführung großer Sprachmodelle auf lokalen Systemen ist stark abhängig vom verfügbaren Grafikspeicher (VRAM) ab. Kleine Modelle mit etwa 1,5 Mrd. Parametern benötigen in der Regel rund 4-6 GB VRAM und können auf herkömmlichen Desktop-GPUs betrieben werden [51]. Mittlere Modelle mit 7-14 Mrd. Parametern beanspruchen ca. 8-16 GB VRAM. GPUs wie die RTX 3080/3090 oder 4080/4090 erweisen sich hierbei als besonders geeignet. Eine Quantisierung auf 8 oder 4 Bit reduziert den Speicherbedarf zusätzlich, ohne die Leistungsfähigkeit wesentlich zu beeinträchtigen [51]. Mit steigender Modellgröße wächst nicht nur der Speicherbedarf für die Modellgewichte selbst, sondern auch für KV Caches, temporäre Aktivierungen und Framework-Overhead. Der KV Cache wächst linear mit der Länge des Textkontexts und kann bei sehr langen Eingaben sogar die Größe der Modellgewichte übersteigen [52]. Große Modelle mit rund 32 Mrd. Parametern benötigen daher 16-24 GB VRAM und laufen meist auf professionellen GPUs, wobei Model-Sharding oder Tensor-Parallelismus eine effiziente Verteilung der Rechenlast auf mehrere GPUs ermöglichen [51]

Sehr große Modelle mit etwa 70 Mrd. Parametern erfordern 32 GB VRAM oder mehr und laufen fast nahezu auf Multi-GPU-Setups oder spezialisierten Server-Clustern. Frameworks wie Megatron DeepSpeed sind entscheidend, um die Berechnungen effizient zu koordinieren [51]. Wird der Speicher überschritten und Teile des Modells auf den Hauptspeicher ausgelagert, sinkt die Inferenzgeschwindigkeit erheblich: Bei einem 70B-Modell fällt die Rate von über 25 Tokens pro Sekunde auf nur 3–5 Tokens pro Sekunde. CPU-only-Ausführung ist noch langsamer; ein 7B-Modell erreicht nur 2–5 Tokens pro Sekunde, während eine mittlere GPU über 40 Tokens pro Sekunde liefert [52].

Für die Praxis zeigt sich, dass 12 GB VRAM das Minimum für eine sinnvolle lokale Nutzung sind, während 24 GB VRAM optimale Leistung ermöglichen. Dank neuerer Modelle wie Qwen 3.5 9B, die lediglich 6,6 GB VRAM benötigen, sind auch kleinere Karten praktisch nutzbar. Damit lassen sich lokale Sprachmodelle sowohl auf High-End-GPUs als auch auf leistungsfähigen Consumer-Systemen betreiben, wobei die Hardware entscheidend für Geschwindigkeit, Kontextlänge und Einsatzmöglichkeiten bleibt. [52]

Für die Implementierung hat sich Python als bevorzugte Programmiersprache etabliert, da es eine breite Bibliotheksunterstützung für KI- und LLM-Frameworks bietet [53]. Lokale Modelle

werden typischerweise aus Repositories „gepullt“, zum Beispiel von Hugging Face oder Ollama, wodurch Unternehmen auf fertige, geprüfte Modelle zugreifen können, ohne sie selbst von Grund auf trainieren zu müssen [54, 55].

4.2 Lokales Deployment von Sprachmodellen

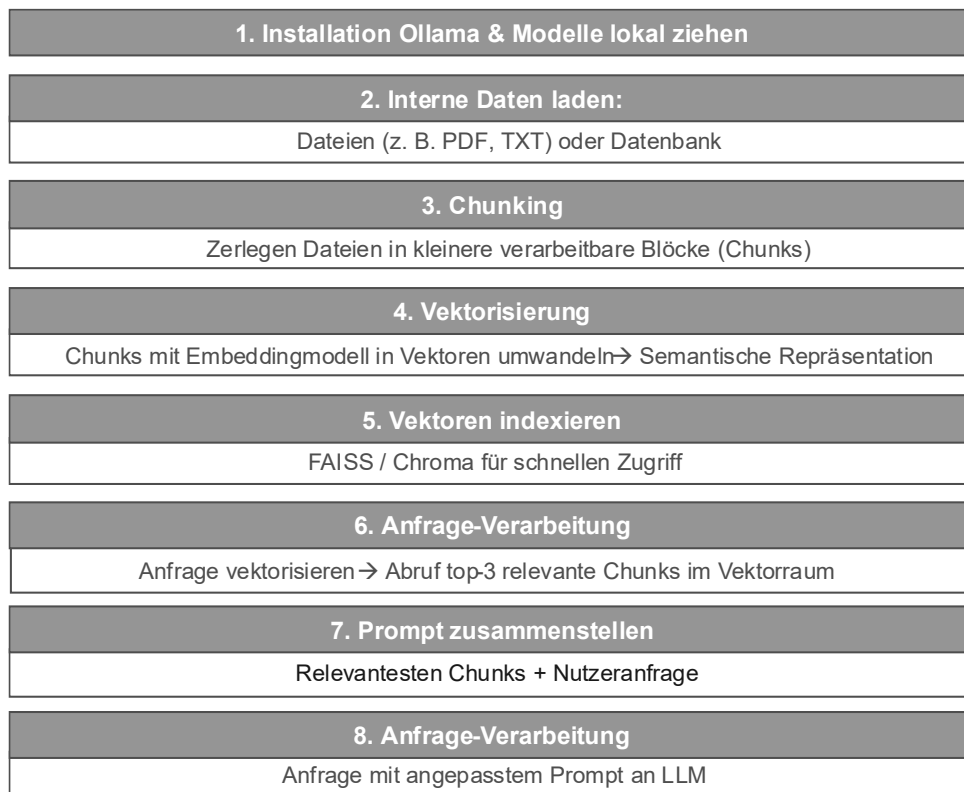


Abb. 4 Übersicht einer beispielhaften Vorgehensweise zur Implementierung eines lokalen großen Sprachmodells (LLM)

Die Einführung von Sprachmodellen in einer Organisation kann zunächst komplex erscheinen, insbesondere wenn firmenspezifische Daten integriert werden sollen. Im Folgenden wird eine beispielhafte Vorgehensweise beschrieben, um ein lokales Sprachmodell erstmals für interne Zwecke und eigene Daten einzusetzen (siehe Abb. 4).

Zunächst, in **Schritt 1**, könnte eine Plattform wie Ollama oder eine vergleichbare Lösung installiert werden, und die gewünschten Modelle könnten lokal heruntergeladen werden. Dabei wäre es sinnvoll zu beachten, dass einige Modelle große Ressourcen benötigen, insbesondere GPUs, und dass die Hardwareanforderungen vorher geprüft werden sollten. In **Schritt 2** könnten die firmenspezifischen Daten vorbereitet werden, wie PDF- oder TXT-Dokumente sowie Inhalte aus internen Datenbanken, die für das Modell zugänglich gemacht werden sollen. Es wäre ratsam, sensible Informationen zu klassifizieren oder zu anonymisieren, bevor sie verarbeitet werden. In **Schritt 3** könnten die Dokumente in kleinere, verarbeitbare Blöcke, sogenannte Chunks, zerlegt werden. Die Größe und mögliche Überschneidungen der Chunks könnten die Genauigkeit späterer Suchanfragen beeinflussen. In **Schritt 4** könnten diese Chunks mithilfe eines Embedding-Modells in Vektoren umgewandelt werden, die die semantische Bedeutung der Inhalte repräsentieren. Dabei könnte ein auf deutsche Texte spezialisiertes Embedding-Modell bessere Ergebnisse

liefern, wenn die Inhalte überwiegend auf Deutsch vorliegen. In **Schritt 5** könnten die Embeddings in einem Index wie FAISS oder Chroma gespeichert werden, um schnellen Zugriff auf relevante Informationen zu ermöglichen. Dabei wäre es nützlich, Speicherbedarf, Persistenz und Update-Möglichkeiten zu berücksichtigen, insbesondere bei großen Datenmengen. Wenn eine Nutzeranfrage erfolgt, könnte in **Schritt 6** die Anfrage vektorisiert und die top-relevanten Chunks aus dem Index abgerufen werden. In **Schritt 7** könnten diese relevanten Chunks zusammen mit einem sorgfältig gestalteten Prompt an das lokale LLM übergeben werden. Schließlich könnte das Modell in **Schritt 8** die Anfrage verarbeiten und eine Antwort liefern, die auf den firmenspezifischen Daten basiert.

Für die Interaktion mit dem Modell könnten verschiedene Optionen für Benutzeroberflächen genutzt werden. Mit OpenWebUI könnte eine fertige GPT-ähnliche Oberfläche eingesetzt werden, die Modellwahl, Dokumenten-Upload und Chat-Funktionalität unterstützt [56]. Wer individuelle Web-Interfaces erstellen möchte, könnte auf Streamlit zurückgreifen, ein Python-Framework für schnelle Frontendentwicklung [57]. Für vollständig maßgeschneiderte Webanwendungen könnte Vue.js verwendet werden, das sich mit einem Python-Backend kombinieren lässt und vielfältige Anpassungsmöglichkeiten bietet [58]. Durch einen eigenen Workflow könnte die Datenverarbeitung innerhalb der eigenen Infrastruktur bleiben, wodurch Datenschutzrisiken minimiert würden. Idealerweise sollten Modelle optimal an die spezifischen Anforderungen und Bedürfnisse des Unternehmens angepasst werden, während die Benutzer eine gut strukturierte und benutzerfreundliche Schnittstelle zur Interaktion mit den LLM-Funktionen erhalten sollten.

5. Potenziale und Anwendungsfelder für die zukünftige Arbeitswelt

Nach der Darstellung von Generativer KI, LLMs, deren Pretraining und verschiedenen Methoden zur erweiterten Nutzung von LLMs stellt sich nun die Frage: Welche konkreten Potenziale eröffnen diese Technologien für die zukünftige Arbeitswelt? Die bisherigen Kapitel haben die technischen Grundlagen und Anpassungsmöglichkeiten von LLMs erläutert, während im Folgenden beispielhaft Optionen erläutert werden, wie Unternehmen und Mitarbeitende von diesen Entwicklungen profitieren können.

Generative KI, insbesondere multimodale LLMs, bietet dabei die Chance, Innovation, Effizienz und Kundennutzen entlang der gesamten Wertschöpfungskette erheblich zu steigern. Die Einsatzfelder entwickeln sich dynamisch und werden kontinuierlich durch Forschung sowie erste praktische Anwendungen erweitert [59, 60]. Dies betrifft sowohl direkte Wertschöpfungsbereiche, wie den Shopfloor und Produktionsbetrieb, als auch indirekte Funktionen sowie Entwicklungs- und Forschungseinheiten. Über die isolierte Betrachtung einzelner Bereiche hinaus deutet sich an, dass LLMs insbesondere an den Schnittstellen zwischen Idee, Design und Produktion eine integrative Rolle übernehmen könnten. Sie ermöglichen beispielsweise einen effizienteren Einsatz bestehender Kapazitäten [61]. Vor diesem Hintergrund ist es wesentlich, Generative KI nicht ausschließlich unter dem Gesichtspunkt kurzfristiger Effizienzgewinne oder direkter Kostenreduktionen zu betrachten. Ihr Einsatz erfordert eine umfassende Perspektive auf die zukünftige Arbeitswelt, einschließlich der Anpassung von Rollenprofilen über verschiedene Hierarchieebenen hinweg und der Entwicklung neuer, ethisch reflektierter Arbeitsstrukturen. Auf diese Weise könnte Generative KI langfristig sowohl die Leistungsfähigkeit von Unternehmen als auch das Wohl der Mitarbeitenden fördern und zur nachhaltigen Stärkung des Wirtschaftsstandorts Deutschland beitragen.

- **Konversationelle KI und Wissensmanagement**

Im Bereich der konversationellen KI zeigt sich bereits ein vergleichsweise weit entwickeltes Einsatzfeld. Chatbots und KI-Assistenzsysteme auf Basis von LLMs nutzen kontextsensitive Verarbeitung, Retrieval-Augmented Generation (RAG) und semantische Embeddings, um konsistente und kontextualisierte Antworten über mehrere Interaktionen hinweg bereitzustellen. [62]

Darauf aufbauend eröffnet die Kombination aus LLMs, Wissensdatenbanken und automatisierter Content-Generierung weitere Anwendungsmöglichkeiten. LLMs können große Mengen an Dokumenten analysieren, relevante Informationen extrahieren und adressatengerecht aufbereiten [62]. Ebenso lassen sich Inhalte für Marketing, Schulungen oder interne Kommunikation teilweise automatisiert generieren [62].

Insgesamt entsteht so das Potenzial für durchgängige Wissens- und Kommunikationsstrukturen, die interne Prozesse ebenso wie kundenorientierte Anwendungen unterstützen. Dies kann zu verbesserter Informationsverfügbarkeit, beschleunigten Entscheidungsprozessen und effizienterem Ressourceneinsatz führen. Anwendungen wie verständliche Aufbereitung komplexer Produktinformationen, personalisierte Kundeninteraktion sowie automatisierte Beratungs- und Upselling-Prozesse erscheinen grundsätzlich realisierbar.[62]

- **Produktentwicklung und Design**

Auch in der Produktentwicklung und im Design bestehen relevante Anwendungspotenziale. LLMs könnten die Generierung und Optimierung von Produkten unterstützen, indem sie Anforderungen hinsichtlich Material, Kosten und Leistung integrieren und mit simulationsgestützten Ansätzen verknüpfen [59, 61]. Dies könnte zu beschleunigter Prototypenentwicklung, reduziertem Materialeinsatz und effizienteren Produktdesigns führen [61, 62].

Erste Ansätze lassen sich bereits in der Luftfahrt- und Automobilindustrie beobachten, etwa bei der Optimierung einzelner Komponenten mit dem Ziel, leichtere und treibstoffeffizientere Bauteile zu entwickeln, ohne die strukturelle Integrität zu beeinträchtigen [59]. Die Verknüpfung von Design, Simulation und Fertigung kann perspektivisch den Übergang vom Konzept zur Produktion beschleunigen [61, 62].

- **Betriebsoptimierung, Qualitätssteigerung und Lieferkettenmanagement**

Multimodale LLMs verbessern zunehmend Betriebsabläufe, Produktivität und Qualität. Sie können heterogene Datenquellen – Sensordaten, Wartungsprotokolle, Bilder und technische Dokumentationen - zusammenführen, Muster erkennen, interpretieren und in verständliche Handlungsempfehlungen überführen [60, 61].

- **Vorausschauende Wartung:** Tieferes Verständnis von Maschinenausfällen durch Verknüpfung technischer Signale mit historischen Berichten und Expertenwissen [59, 61].
- **Qualitätskontrolle:** Erweiterte Fehlererkennung durch Kombination visueller Daten mit Texten und Produktionskontext, inklusive Generierung synthetischer Daten und interaktiver Analyse [60, 61].
- **Lieferkettenmanagement:** Integration strukturierter und unstrukturierter Daten ermöglicht kontextbasierte Simulationen, dynamische Anpassungen in Echtzeit und natürlichsprachliche Interaktion mit komplexen Planungssystemen [59, 60, 62]

- **Softwareentwicklung**

Im Bereich der Softwareentwicklung steigern LLMs die Produktivität, indem sie auf Basis ihres **vortrainierten Modells (Pretraining)** Muster in bestehenden Codebasen erkennen und den Übergang zwischen natürlicher Sprache und verschiedenen Programmiersprachen erleichtern. Dadurch wird Code für eine breitere Zielgruppe verständlicher, und LLMs können funktionalen Code direkt aus natürlichen Sprachbeschreibungen generieren. [62]

Ein bekanntes Beispiel ist **GitHub Copilot**, das von vielen Entwicklerinnen und Entwicklern zur Unterstützung der Softwareentwicklung eingesetzt wird. LLMs helfen hierbei bei Code-Autovervollständigung, Multi-Language-Support, Refactoring und Debugging. Unternehmen können diese Technologien nutzen, um Entwicklungszeiten zu verkürzen, die Codequalität zu verbessern und die Einführung neuer Technologien zu erleichtern. [62]

- **Nachhaltigkeit und Ressourceneffizienz**

Darüber hinaus fördern multimodale LLMs Nachhaltigkeit und Ressourceneffizienz, indem sie Zusammenhänge zwischen Produktionsprozessen, Energieverbrauch und Materialeinsatz ganzheitlich analysieren [59, 60]. Optimierungspotenziale lassen sich identifizieren und in konkrete Maßnahmen übersetzen, wodurch organisationale Entscheidungsprozesse unterstützt werden.

6. Fazit

Das vorliegende Dokument bietet eine fundierte Zusammenfassung der Funktionsweise und methodischen Potenziale von LLMs sowie ihrer Anwendbarkeit im unternehmerischen Kontext. Es wurde angedeutet, wie LLMs über die konventionellen Einsatzformen, wie klassische Chatbots, hinaus operationalisiert werden können, um Antworten präzise auf spezifische Informationsbedarfe auszurichten. Hierzu werden etablierte methodische Ansätze präsentiert, die die Verlässlichkeit und Aussagekraft der Modellgenerierungen signifikant erhöhen. Dazu zählen die Optimierung von Prompts, gezieltes Fine-Tuning der Modellparameter sowie die kontextuelle Anreicherung durch semantische Ähnlichkeitssuchen in unternehmensinternen Dokumenten.

Aufbauend auf diesen methodischen Grundlagen verdeutlicht das Dokument den strategischen Mehrwert multimodaler LLMs. Diese Systeme besitzen die Fähigkeit, heterogene Datentypen integrativ zu verarbeiten, kontextsensitiv zu interpretieren und in handlungsrelevantes Wissen zu überführen [60, 62].

Vor diesem Hintergrund eröffnen LLMs Unternehmen die Möglichkeit, Prozesse effizienter zu gestalten, fundierte Entscheidungen zu treffen und die Adaptionfähigkeit auf dynamische Veränderungen zu steigern [59, 61]. Sie fungieren somit als mögliches Bindeglied zwischen kreativer Ideenfindung und industrieller Umsetzung, wodurch Innovationszyklen verkürzt und die Wettbewerbsfähigkeit gesteigert werden können. Gleichzeitig ist zu betonen, dass das Ausmaß der erzielbaren Effizienzgewinne und der Innovationsvorteile wesentlich von der konkreten Implementierung, den organisatorischen Strukturen und den regulatorischen Rahmenbedingungen abhängt.

Literaturverzeichnis

- [1] M. U. Hadi *et al.*, "A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage," *TechRxiv*, pp. 1–29, 2023, doi: 10.36227/techrxiv.23589741.v1.
- [2] B. K. Mishra and R. Kumar, *Natural language processing in artificial intelligence*. Burlington, ON, Canada, Palm Bay, Florida, USA: Apple Academic Press, 2021.
- [3] R. Sternberg, *Human intelligence: Psychometric theories*. [Online]. Available: <https://www.britannica.com/science/human-intelligence-psychology/Psychometric-theories> (accessed: Sep. 3 2025).
- [4] A. Cheishvili, *The Future Of Artificial General Intelligence*. [Online]. Available: <https://www.forbes.com/councils/forbestechcouncil/2021/07/16/the-future-of-artificial-generalintelligence>
- [5] U. Mittal, S. Sai, V. Chamola, and D. Sangwan, "A Comprehensive Review on Generative AI for Education," *IEEE Access*, vol. 12, pp. 142733–142759, 2024, doi: 10.1109/ACCESS.2024.3468368.
- [6] R. Gozalo-Brizuela and E. E. G. Merchan, "A Survey of Generative AI Applications," *Journal of Computer Science*, vol. 20, no. 8, pp. 801–818, 2024, doi: 10.3844/jcssp.2024.801.818.
- [7] T. Schick *et al.*, "PEER: A Collaborative Language Model," 2022.
- [8] C. Jeong, "A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture," 04, 2023. [Online]. Available: <https://arxiv.org/pdf/2309.01105>
- [9] R. Bommasani *et al.*, "On the Opportunities and Risks of Foundation Models," 2021. [Online]. Available: <https://arxiv.org/pdf/2108.07258>
- [10] J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," 2019. [Online]. Available: <https://arxiv.org/pdf/1908.02265>
- [11] A. Radford *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," 2021. [Online]. Available: <https://arxiv.org/pdf/2103.00020>
- [12] J. Betker *et al.*, *Improving image generation with better captions*, 2023. [Online]. Available: <https://scholar.google.com/citations?user=2o3izlkaaaaj&hl=de&oi=sra>
- [13] OpenAI *et al.*, "GPT-4o System Card," 2024. [Online]. Available: <https://arxiv.org/pdf/2410.21276>
- [14] OpenAI *et al.*, "GPT-4 Technical Report," 2023. [Online]. Available: <https://arxiv.org/pdf/2303.08774>
- [15] H. Naveed *et al.*, "A Comprehensive Overview of Large Language Models," 2023. [Online]. Available: <https://arxiv.org/pdf/2307.06435>
- [16] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Prompting Is Programming: A Query Language for Large Language Models," PLDI, 2022. [Online]. Available: <https://arxiv.org/pdf/2212.06094>

- [17] P. Peykani, F. Ramezanlou, C. Tanasescu, and S. Ghanidel, "Large Language Models: A Structured Taxonomy and Review of Challenges, Limitations, Solutions, and Future Directions," *Applied Sciences*, vol. 15, no. 14, 2025, doi: 10.3390/app15148103.
- [18] Y. Chang *et al.*, "A Survey on Evaluation of Large Language Models," 2023. [Online]. Available: <https://arxiv.org/pdf/2307.03109>
- [19] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," 2020. [Online]. Available: <https://arxiv.org/pdf/2005.14165>
- [20] A. Vaswani *et al.*, "Attention Is All You Need," 2017. [Online]. Available: <https://arxiv.org/pdf/1706.03762>
- [21] S. Höglund and J. Khedri, "Comparison Between RLHF and RLAIIF in Fine-Tuning a Large Language Model," 2023. [Online]. Available: <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1782683>
- [22] R. X. Gao, J. Krüger, M. Merklein, H.-C. Möhring, and J. Váncza, "Artificial Intelligence in manufacturing: State of the art, perspectives, and future directions," *CIRP Annals*, vol. 73, no. 2, pp. 723–749, 2024, doi: 10.1016/j.cirp.2024.04.101.
- [23] S. Höglund and J. Khedri, "Comparison Between RLHF and RLAIIF in Fine-Tuning a Large Language Model," 2023. [Online]. Available: <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1782683>
- [24] M. O'Neill and M. Connor, *Amplifying Limitations, Harms and Risks of Large Language Models*, 2023.
- [25] M. Turpin, J. Michael, E. Perez, and S. R. Bowman, "Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting," 2023. [Online]. Available: <https://arxiv.org/pdf/2305.04388>
- [26] C. Deng, Y. Zhao, X. Tang, M. Gerstein, and A. Cohan, "Investigating Data Contamination in Modern Benchmarks for Large Language Models," 2023. [Online]. Available: <https://arxiv.org/pdf/2311.09783>
- [27] H. He and W. J. Su, "A Law of Next-Token Prediction in Large Language Models," 2024. [Online]. Available: <https://arxiv.org/pdf/2408.13442>
- [28] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the Dangers of Stochastic Parrots," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, Virtual Event Canada, 2021, pp. 610–623.
- [29] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities," 2024. [Online]. Available: <https://arxiv.org/pdf/2408.13296>
- [30] X.-K. Wu *et al.*, "LLM Fine-Tuning: Concepts, Opportunities, and Challenges," *BDCC*, vol. 9, no. 4, p. 87, 2025, doi: 10.3390/bdcc9040087.
- [31] D. M. Anisuzzaman, J. G. Malins, P. A. Friedman, and Z. I. Attia, "Fine-Tuning Large Language Models for Specialized Use Cases," (in eng), *Mayo Clinic proceedings. Digital health*, vol. 3, no. 1, p. 100184, 2025, doi: 10.1016/j.mcpdig.2024.11.005.
- [32] C. Han *et al.*, "Facing the Elephant in the Room: Visual Prompt Tuning or Full Finetuning?," 2024. [Online]. Available: <https://arxiv.org/pdf/2401.12902>

- [33] C. Han *et al.*, "Facing the Elephant in the Room: Visual Prompt Tuning or Full Finetuning?," 2024. [Online]. Available: <https://arxiv.org/pdf/2401.12902>
- [34] Y. Wang, J. Chauhan, W. Wang, and C.-J. Hsieh, "Universality and Limitations of Prompt Tuning," 2023. [Online]. Available: <https://arxiv.org/pdf/2305.18787>
- [35] D. C. Schmidt, J. Spencer-Smith, Q. Fu, and J. White, "Towards a Catalog of Prompt Patterns to Enhance the Discipline of Prompt Engineering," *Ada Lett.*, vol. 43, no. 2, pp. 43–51, 2024, doi: 10.1145/3672359.3672364.
- [36] S. Schulhoff *et al.*, "The Prompt Report: A Systematic Survey of Prompt Engineering Techniques," 2024. [Online]. Available: <https://arxiv.org/pdf/2406.06608>
- [37] S. Jeon and H.-G. Kim, "A comparative evaluation of chain-of-thought-based prompt engineering techniques for medical question answering," (in eng), *Computers in biology and medicine*, vol. 196, Pt A, p. 110614, 2025, doi: 10.1016/j.combiomed.2025.110614.
- [38] Tethered Software Inc., *The Few Shot Prompting Guide*. [Online]. Available: <https://www.prompthub.us/blog/the-few-shot-prompting-guide> (accessed: Nov. 3 2025).
- [39] G. Dong, Y. Zhu, C. Zhang, Z. Wang, Z. Dou, and J.-R. Wen, "Understand What LLM Needs: Dual Preference Alignment for Retrieval-Augmented Generation," 2024. [Online]. Available: <https://arxiv.org/pdf/2406.18676>
- [40] S. Wu *et al.*, "Retrieval-Augmented Generation for Natural Language Processing: A Survey," 2024. [Online]. Available: <https://arxiv.org/pdf/2407.13193>
- [41] A. Ammar, A. Koubaa, O. Nacar, and W. Boulila, "Optimizing Retrieval-Augmented Generation: Analysis of Hyperparameter Impact on Performance and Efficiency," 2025. [Online]. Available: <https://arxiv.org/pdf/2505.08445>
- [42] S. Pokhrel, B. K. C., and P. B. Shah, "A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots: Combining Web Scraping, Vectorization, and Semantic Search," *TCSST*, vol. 6, no. 4, pp. 424–442, 2024, doi: 10.36548/jtcsst.2024.4.007.
- [43] X. Huang *et al.*, "Understanding the planning of LLM agents: A survey," 2024. [Online]. Available: <https://arxiv.org/pdf/2402.02716>
- [44] S. Han, Q. Zhang, W. Jin, and Z. Xu, "LLM Multi-Agent Systems: Challenges and Open Problems," 2024. [Online]. Available: <https://arxiv.org/pdf/2402.03578>
- [45] R. Y. Maragheh and Y. Deldjoo, "The Future is Agentic: Definitions, Perspectives, and Open Challenges of Multi-Agent Recommender Systems," 2025. [Online]. Available: <https://arxiv.org/pdf/2507.02097>
- [46] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang, "A Dynamic LLM-Powered Agent Network for Task-Oriented Agent Collaboration," 2023. [Online]. Available: <https://arxiv.org/pdf/2310.02170>
- [47] M. Cemri *et al.*, "Why Do Multi-Agent LLM Systems Fail?," 2025. [Online]. Available: <https://arxiv.org/pdf/2503.13657>
- [48] H. Nolte, M. Finck, and K. Meding, "Machine Learners Should Acknowledge the Legal Implications of Large Language Models as Personal Data," 2025. [Online]. Available: <https://arxiv.org/pdf/2503.01630>
- [49] X. Hou, J. Han, Y. Zhao, and H. Wang, "Unveiling the Landscape of LLM Deployment in the Wild: An Empirical Study," 2025. [Online]. Available: <https://arxiv.org/pdf/2505.02502>

- [50] G. Pan, V. Chodnekar, A. Roy, and H. Wang, "A Cost-Benefit Analysis of On-Premise Large Language Model Deployment: Breaking Even with Commercial LLM Services," 2025. [Online]. Available: <https://arxiv.org/pdf/2509.18101>
- [51] *General recommended VRAM Guidelines for LLMs*. [Online]. Available: https://dev.to/simplr_sh/general-recommended-vram-guidelines-for-llms-4ef3 (accessed: Sep. 3 2025).
- [52] Mark Bartlett, *Best VRAM Cheat Sheet for Local LLMs: Every Model, Every Quant*. [Online]. Available: <https://insiderllm.com/guides/vram-requirements-local-llms/> (accessed: Jan. 31 2026).
- [53] L. Twist *et al.*, "A Study of LLMs' Preferences for Libraries and Programming Languages," 2025. [Online]. Available: <https://arxiv.org/pdf/2503.17181>
- [54] Huggingface, *Introduction*. [Online]. Available: <https://huggingface.co/learn/llm-course/chapter1/1> (accessed: Oct. 1 2025).
- [55] Ollama, *Pricing*. [Online]. Available: <https://ollama.com/pricing> (accessed: Sep. 2 2025).
- [56] Open WebUI, *The self-hosted AI interface*. [Online]. Available: <https://openwebui.com/> (accessed: Sep. 1 2025).
- [57] Snowflake Inc., *Build powerful generative AI apps*. [Online]. Available: <https://streamlit.io/generative-ai> (accessed: Sep. 1 2025).
- [58] Vue.js, *Introduction*. [Online]. Available: <https://vuejs.org/guide/introduction> (accessed: Sep. 1 2025).
- [59] P. K. Haridasan and H. Jawale, "Generative AI in Manufacturing: A Review of Innovations, Challenges, and Future Prospects," *JAIMLD*, vol. 2, no. 2, pp. 1418–1424, 2024, doi: 10.51219/JAIMLD/praveen-haridasan/321.
- [60] S. Shafiee, "Generative AI in manufacturing: a literature review of recent applications and future prospects," *Procedia CIRP*, vol. 132, pp. 1–6, 2025, doi: 10.1016/j.procir.2025.01.001.
- [61] Y. Li *et al.*, "Large Language Models for Manufacturing," 2024. [Online]. Available: <https://arxiv.org/pdf/2410.21418>
- [62] U. Kamath, K. Keenan, G. Somers, and S. Sorenson, "LLMs in Production," in *Large language models: a deep dive: Bridging theory and practice*, U. Kamath, K. Keenan, G. Somers, and S. Sorenson, Eds., Cham: Imprint Springer, 2024, pp. 315–373.

EDITH C/O
HOUSE OF DIGITAL
TRANSFORMATION E.V.
MORNEWEGSTR. 30,
64293 DARMSTADT, GERMANY
INFO@EDITH-HESSEN.DE
WWW.EDITH-HESSEN.DE



LINKEDIN | WEBSITE

EDITH
GERMANY



Co-funded by
the European Union