BHT Berliner Hochschule für Technik

Wissensmanagement in der IT-Administration durch wissensbasierte Large Language Models

vorgelegt von

Friederike Buchner

EDV.Nr.:xxxxxx

dem Fachbereich VI – Informatik und Medien
der Berliner Hochschule für Technik Berlin
vorgelegte Arbeit
zur Absolvierung des Moduls

Wissenschaftliches Seminar

im Studiengang

Medieninformatik (Online)

Tag der Abgabe 17. November 2025

Studiere Zukunft

Version 0.2α
letzte Änderung: 17. November 2025

Gutachter
Prof. Dr. S. Edlich Berliner Hochschule für Technik

Kurzfassung

Die Kurzfassung gibt ein kurzes und prägnantes Bild der gesamten Arbeit. Sie soll den Leser neugierig machen und klarmachen, was zu erwarten ist. Erreichte Ergebnisse werden kurz umrissen.

Abstract

Bachelor and Master-Thesises usually are often written in german. Nevertheless, their content may be intersting for people being not able to read german. In order to awaken their interest in this topic, an abstract in english is given. The experimental results and analysis are shown in short

Inhaltsverzeichnis

Kapitel 1

Einleitung

Nutzung von KI (GPT-5):

- Übersetzung (Englisch-Deutsch):
  - self-supervision
  - downstream tasks
  - homogenization
  - emergence
  - transfer learning
  - sentiment analysis

Kapitel 2

Theoretische und technische Grundlagen

2.1 Large Language Models (LLMs) und Foundation Models

Foundation Models nach (Bommasani u. a. 2022) sind Modelle, die üblicherweise selbstüberwacht (*self-supervised*) auf umfassenden Daten trainiert sind und auf eine große Anzahl an nachgelagerten Aufgaben (*downstream tasks*) angepasst werden können. Aktuelle Beispiele beinhalten BERT, GPT-3 und CLIP. Von einem technologischen Standpunkt her sind *Foundation Models* nicht neu, da sie auf tiefen neuronalen Netzwerken und selbstüberwachtem Lernen basieren, was beides bereits seit Jahrzehnten existiert. Beachtenswert sind *Foundation Models* heutzutage deshalb, weil sich ihre schiere Größe in den letzten Jahren vervielfacht hat und sie somit alle Vorstellungen dessen, was man vor wenigen Jahren für möglich hielt, sprengen. GPT-3 beispielsweise hat 175 Milliarden Parameter und kann durch natürlichsprachige Prompts so angepasst werden, dass es eine passable Leistung in vielfältigen Aufgaben zeigt, obwohl es nicht explizit für diese Aufgaben trainiert wurde (Bommasani u. a. 2022, S. 3).
Nach technischen Gesichtspunkten funktionieren *Foundation Models* durch Transferlernen (*transfer learning*) und Skalierung (*scale*). Transferlernen bedeutet, das „Wissen", was in einer Anwendung (bspw. Bilderkennung) erlernt wurde, auf eine andere Aufgabe (bspw. das Erkennen von Aktivitäten in Videos) zu übertragen. Innerhalb des *Deep Learning* ist das Vortraining (*pretraining*) der vorherrschende Ansatz für Transferlernen: ein Model trainiert eine Ersatzaufgabe und wird dann via *fine-tuning* für die eigentlich relevante nachgelagerte Aufgabe angepasst. Zusammen mit der Skalierung von *Foundation Models* entsteht nun eine sehr mächtige Komination. Hierfür werden drei entscheidende Punkte wichtig: die Verbesserungen in Computer Hardware, die Entwicklung der Transformer Architektur und die Verfügbarkeit von viel mehr Trainingsdaten (Bommasani u. a. 2022, S. 4).
Letzteres kann nicht deutlich genug hervorgehoben werden: die Wichtigkeit des Vorhandenseins von Daten und die Fähigkeit, sich diese zunutze zu machen. Transferlernen durch annotierte Datensätze war jahrelang die gängige Praxis. Die hohen Kosten von Annotationen, insbesondere von hochqualitativen, händisch erzeugten Annotationen, haben jedoch eine natürliche Grenze in der Skalierung von Trainingsdaten dargestellt. Im selbstüberwachten Lernen ergibt sich die Ersatzaufgabe für das Vortraining automatisch aus unannotierten Daten. Selbstüberwachte Aufgaben sind nicht nur besser skalierbar und ausschließlich abhängig von unannotierten Daten, sondern sie zwingen das Modell dazu, Teile der Eingabe vorherzusehen, was sie reichhaltiger und potentiell nützlicher machen als Modelle, die in einem begrenzteren Sprachaum trainiert sind (Bommasani u. a. 2022, S. 4).
Selbstüberwachtes Lernen war zunächst eine Unterdisziplin von NLP, die sich parallel zu anderen Entwicklungen ergab. Ab der Einführung des BERT-Modells (Devlin u. a. o. D.) 2019 wurde

selbstüberwachtes Lernen zur Norm in NLP. Die Akzeptanz, dass ein einzelnes Modell derart nützlich über eine weite Bandbreite von Aufgaben sein könnte, markiert den Beginn der Ära von *Foundation Models* (Bommasani u. a. 2022, S. 5).
Homogenisierung ist ein Ergebnis der Konsolidierung von Systemen für Maschinelles Lernen über eine weite Palette an Anwendungen. Es ermöglicht das Erledigen vieler Aufgaben aber bildet auch *single points of failure* (Bommasani u. a. 2022, S. 3). *Foundation Models* haben ein nie zuvor gesehenes Maß an Homogenisierung herbeigeführt, da fast alle *state-of-the-art* NLP-Modelle aus einem von wenigen Modellen wie BERT, GPT o.ä. hervorgehen. Dadurch können alle NLP-Anwendungen direkt von Verbesserungen in *Foundation Models* profitieren. Es birgt aber auch die Gefahr, dass alle KI-Systeme dieselben problematischen Verzerrungen (*biases*) einiger weniger *Foundation Modeld* erben.
Ein zweites Charakteristikum von *Foundation Models* ist die Emergenz. Das bedeutet, dass das Verhalten eines Systems implizit induziert ist, anstatt explizit konstruiert. Das zeigt sich, indem ein System (zur Überraschung seiner Schaffer*innen) Verhaltensweisen oder Fähigkeiten aufweist, die nicht von außen definiert wurden, sondern die sich eher als Nebenprodukt zum hauptsächlichen Einsatzzweck ergeben. Es ist gleichzeitig die Quelle wissenschaftlicher Erregung sowie Besorgnis über unerwartete Konsequenzen (Bommasani u. a. 2022, S. 3). Emergenz wird umso bedeutender, je größer das Modell skaliert ist. Während GPT-2 mit 1,5 Milliarden Parametern trainiert wurde, wurde GPT-3 mit 175 Milliarden Parametern trainiert, was kontextsensitives Lernen ermöglichte, in welchem das Sprachmodell einfach auf eine nachgelagerte Aufgabe angepasst wird, indem es mit einer natürlichsprachlichen Beschreibung einer Aufgabe (*prompt*) versorgt wird. Dies war eine emergente Fähigkeit, die weder speziell trainiert noch überhaupt antizipiert wurde (Bommasani u. a. 2022, S. 5).
Homogenisierung und Emergenz interagieren miteinander auf potenziell beunruhigende Art und Weise. Homogenisierung kann potenziell enorme Vortaile für viele Domänen bringen, in denen aufgabenspezifische Daten knapp sind. Auf der anderen Seite kann jeder Fehler im Modell blind von allen angepassten Modellen geerbt werden. Da die Macht von *Foundation Models* viel mehr in ihren emergenten Qualitäten als in ihrer expliziten Konstruktion steckt, sind die existierenden Modelle schwer zu verstehen und haben unvorhergesehene Fehlverhalten. Da Emergenz substanzielle Unsicherheiten über die Fähigkeiten und Fehler von *Foundation Models* erzeugt, ist mit der umfassenden Homogenisierung über diese Modelle hinweg ein erhebliches Risiko verbunden. Dieses Risiko zu mitigieren ist eine der zentralen Herausforderungen in der weiteren Entwicklung von *Foundation Models* aus einer ethischen sowie aus einer KI-Sicherheitsperspektive (Bommasani u. a. 2022, S. 6).
Zur Bezeichnung von *Foundation Models* und zur Abgrenzung von Sprachmodellen allgemein kann man sagen, dass der Geltungsbereich von *Foundation Models* schlichtweg weit über die Grenzen von Sprache hinaus reicht. Es wurden auch andere Bezeichnungen wie beispielsweise *General-Purpose Model* oder *Multi-Purpose Model* in Betracht gezogen, da sie den Aspekt, dass diese Modelle vielfältige nachgelagerte Aufgaben bewältigen können, besser einfängen. Sie täuschen aber darüber hinweg, dass *Foundation Models* unfertig sind und weiter angepasst werden müssen. Weitere Namensvorschläge wie *Task-agnostic Model* würden zwar die Art des Trainings wiederspiegeln, aber nicht die Relevanz für weitere nachgelagerte Aufgaben. Es wurde sich also für den Begriff *Foundation* (engl. für Basis, Grundlage) entschieden, da ein *Foundation Model* an sich unfertig ist, aber als allgemeine Grundlage dient, aus der vielfältige aufgabenspezifische Modelle durch Anpassung entstehen können. Gleichzeitig weist der Begriff *Foundation* auch auf die Wichtigkeit von architektonischer Stabilität, funktionaler Sicherheit (engl. *safety*) sowie dem Schutz vor Angriffen (engl. *security*) hin. So wie schlecht konstruierte Fundamente fast schon eine Garantie für strukturelles Versagen sind, sind gut ausgeführte Fundamente verlässliche Grundlagen für zukünftige Anwendungen. Zu betonen ist weiterhin, dass momentan die Natur oder Qualität dieser Art von Fundamenten, die *Foundation Models* bieten, nicht in Gänze verstanden wird und dass nicht einwandfrei beurteilt werden kann, ob diese Fundamente verlässlich sind,

oder nicht.

2.2 Retrieval-Augmented Generation (RAG)

LLMs haben neben ihrem bemerkenswerten Erfolg auch signifikante Grenzen, speziell in domänenspezifischen oder wissensintensiven Aufgaben. Eins der größten Probleme ist das „Halluzinieren“ (Zhang u. a. 2025) beim Verarbeiten von Anfragen, die Informationen betreffen, die nicht in den Trainingsdaten enthalten waren. Um diese Herausforderungen zu bewältigen, werden LLMs per *Retrieval-Augmented-Generation* (RAG) erweitert, indem relevante Inhalte mithilfe semantischer Ähnlichkeitsberechnungen aus externen Wissensbasen abgerufen werden. Indem externes Wissen referenziert wird, reduziert RAG effektiv das Problem, faktisch inkorrekte inhalte zu generieren. Die Integration in LLMs ist mittlerweile weit verbreitet, was RAG zu einer Schlüsseltechnologie im Voranbringen von Chatbots und der Eignung von LLMs für Anwendungen in der realen Welt gemacht hat (Gao u. a. 2024).
Die Erforschung von RAG traf mit der Entwicklung der Transformer Architektur zeitlich aufeinander. Zu Beginn lag der Fokus darauf, Sprachmodelle durch zusätzliche Wissensquellen zu verbessern, insbesondere durch die Integration externer Informationen in vortrainierte Modelle (*Pretrained Models*, PTMs). Mit dem Aufkommen von ChatGPT gab es einen Wendepunkt: Große Sprachmodelle (LLMs) zeigten nun ihre Fähigkeit zum *In-Context Learning* (ICL), also dazu, zur Laufzeit neues Wissen aus Eingabekontexten aufzunehmen und zu verwenden. Das führte die RAG-Forschung dahin, bessere Informationen für LLMs bereitzustellen, um komplexere und wissensintensive Aufgaben in der Inferenz-Phase (also während der Antwortgenerierung) beantworten zu können. Mit voranschreitender Forschung war RAG dann nicht mehr auf die Inferenz-Phase beschränkt, sondern fügte sich immer mehr in LLM *fine-tuning*-Techniken, also das gezielte Nachtrainieren der Modelle mit domänenspezifischen oder aufgabenspezifischen Daten, ein (Gao u. a. 2024).

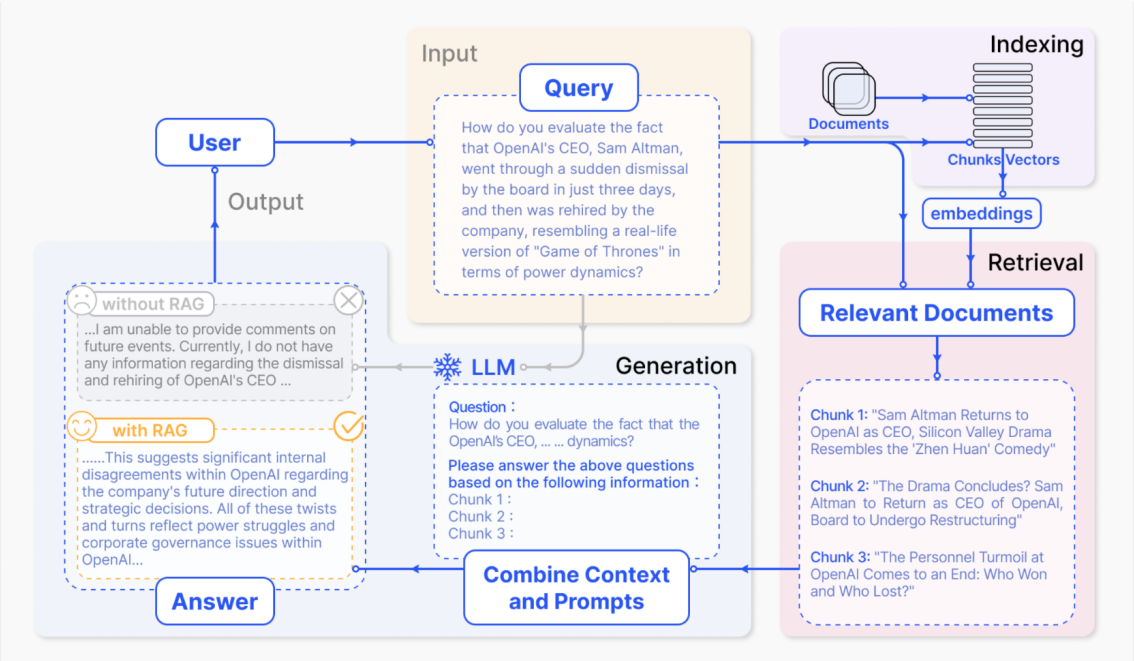

Abbildung 2.1: Überblick über die Funktionsweise von RAG nach (Gao u. a. 2024, S. 3)

In Abbildung 2.1 ist die grundsätzliche Funktionsweise von *Naive RAG*, angewendet auf die Aufgabe der Fragenbeantwortung, dargestellt. Die in der Abbildung dargestellte Frage bezieht sich

auf aktuelle Ereignisse, das heißt, dass die zur Beantwortung benötigten Informationen nicht in den Trainingsdaten enthalten gewesen sein können. RAG überbrückt diese Lücke, indem das benötigte Wissen aus externen Datenbanken abgerufen wird und zusammen mit der initialen Frage einen umfassenden Prompt ergibt, der das LLM befähigt, eine wohlinformierte Antwort zu generieren. Im Wesentlichen besteht *Naive RAG* aus drei Schritten:

1. *Indexing*: Dokumente werden in Abschnitte (engl. *chunks*) unterteilt, in Vektoren kodiert und in einer Vektordatenbank gespeichert
2. *Retrieval*: die relevantesten Top k Abschnitte werden abgerufen, basierend auf semantischer Ähnlichkeit
3. *Generation*: die ursprüngliche Frage wird gemeinsam mit den abgerufenen Abschnitten an ein LLM übergeben, um eine Antwort zu generieren

Obwohl dieses sogenannte *Retrieve-Read*-Framework des *Naive RAG* kosteneffektiv ist und die Performanz des nativen LLM weit übertrifft, hat es dennoch mehrere Schwächen die durch die Entwicklung von *Advanced RAG* sowie *Modular RAG* adressiert wurden (Gao u. a. 2024).
Beim *Advanced RAG* wird ein Fokus auf die Steigerung der Qualität der Abrufe (engl. *retrievals*) gesetzt. Dies passiert, indem Strategien für die Vor- und Nachbearbeitung von Abrufen angewendet werden. Um die Indizierung zu verbessern, werden feinere Segmentierungsgrade sowie Metadaten zusätzlich zu weiteren Optimierungsmethoden angewendet, um die originale Frage klarer zu machen und für die Abfrage aufzubereiten. In der Nachbearbeitung der Abfrage werden die abgefragten Informationen aufbereitet, um die relevantesten Informationsblöcke hervorzuheben. Die direkte Übergabe aller relevanten Dokumente an das LLM könnte zu einer Informationsüberlastung führen. Um dies zu vermeiden, konzentriert sich die Nachbereitung der Abfrage auf die Auswahl essentieller Informationen und die Kürzung des zu verarbeitenden Kontextes. Der Gesamtprozess ähnelt jedoch weiterhin dem des *Naive RAG* und folgt ebenso einer linearen Struktur (Gao u. a. 2024).

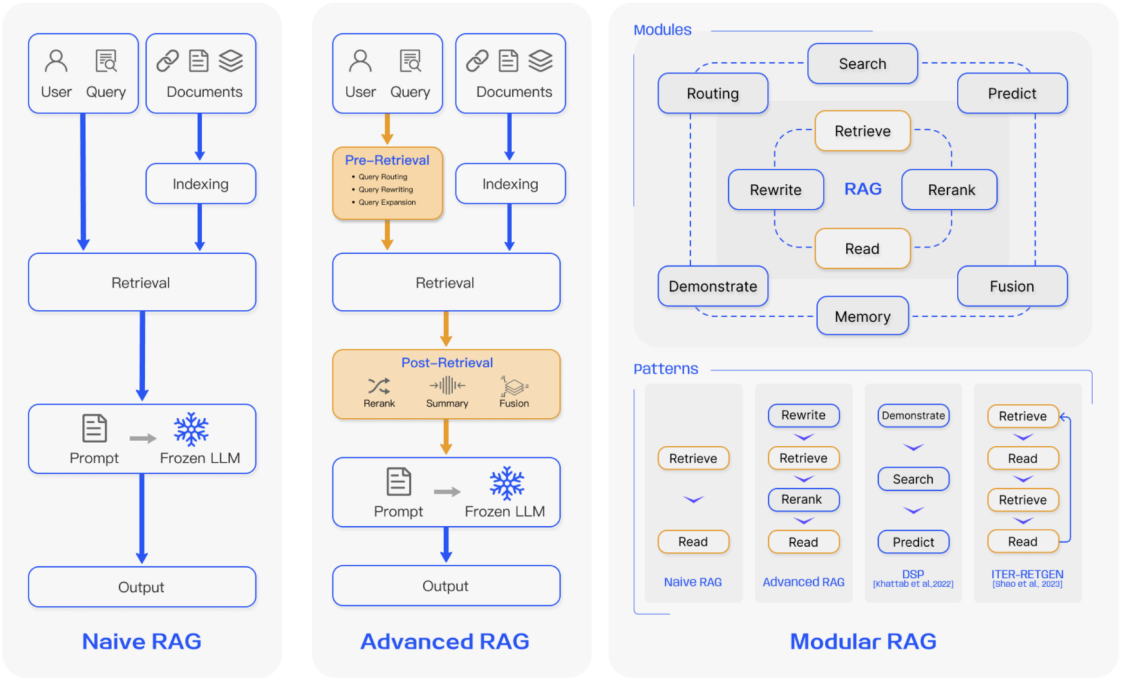

Abbildung 2.2: Überblick über *Naive RAG*, *Advanced RAG* und *Modular RAG* (Gao u. a. 2024, S. 3)

Bei der modularen RAG-Architektur wird diese lineare Struktur hinter sich gelassen und eine größere Anpassbarkeit und Vielseitigkeit geboten. Das modulare RAG-Framework führt spezialisierte Komponenten ein, um die Abfrage- und Verarbeitungskapazitäten zu erhöhen. Ein Such-Modul ermöglicht direkte Suchen über diverse Datenquellen hinweg. Ein Speicher-Modul nutzt den Speicher des LLM, um Abfragen zu steuern. Ein Vorhersage-Modul zielt darauf ab, Redundanz und Rauschen zu verringern, indem es Kontext direkt durch das LLM generiert, und damit Relevanz und Akkuratheit sicherstellt. Diese und weitere Module steigern die Qualität und Relevanz der abgefragten Informationen und ermöglichen so das Ausführen einer Vielfalt von Aufgaben mit erhöhter Präzision und Flexibilität (Gao u. a. 2024).
Modulares RAG geht über die bisherige lineare Struktur von *Naive* sowie *Advanced* RAG hinaus und erlaubt durch seinen modularen Charakter eine bemerkenswerte Anpassbarkeit, indem Module ausgetauscht und rekonfiguriert werden können. Der herkömmliche *Read-Retrieve*-Ansatz wird durch Erfindungen wie *Rewrite-Retrieve-Read*, *Generate-Read*, *Recite-Read* oder anderen erweitert und bietet viele Möglichkeiten, die Fähigkeit des jeweiligen Models, spezifische Aufgaben zu behandeln, zu verbessern. Die flexible Orchestrierung von modularem RAG zeigt mit weiteren Abfragetechniken wie „FLARE“ (Jiang u. a. 2023) und „Self-RAG“ (Asai u. a. 2023), dass dieser Ansatz den starren RAG-Abfrage-Prozess übertrifft, indem die Notwendigkeit einer Abfrage je nach verschiedenem Scenario bewertet wird. Ein weiterer Vorteil der flexiblen Architektur ist, dass das RAG-System leichter mit anderen Technologien (wie etwa *fine-tuning* oder *reinforcement learning*) kombiniert werden kann (Gao u. a. 2024).

2.3 Evaluation von RAG-Modellen

Eine gelungene, durchdachte und gewissenhafte Evaluation ist von großer Bedeutung für den Erfolg von LLMs. Sie kann ein gutes Leitbild für die Verbesserung von Mensch-LLM-Interaktion bieten, was sich positiv auf zukünftige Entwicklungen von Interaktionsdesign und -implementierung auswirkt. Außerdem unterstreicht die weitreichende Anwendbarkeit von LLMs die Wichtigkeit von Sicherheit und Zuverlässigkeit, vor allem in sicherheitssensiblen Sektoren wie Finanzinstitutionen oder staatlichen Behörden. Evaluationsmethoden müssen konstant überprüft und angepasst werden, da LLMs mit steigender Größe und mehr emergenten Fähigkeiten potenzielle neue Risiken entwickeln, die existierende Evaluationsprotokolle nicht abdecken (Chang u. a. 2023).
Im Folgenden soll auf die Evaluation von RAG-Modellen eingegangen werden. Hierfür liefern (Knollmeyer u. a. 2024) einen guten Überblick über die Möglichkeiten vom Benchmarking. Die Resultate ihrer Recherche sollen hier kurz zusammengefasst werden. Sie teilen sich auf in folgende Bereiche: Die Vorhersagen-Evaluation (engl. *predictive evaluation*) und die Datensatz-Evaluation (engl. *dataset evaluation*). Wir wollen uns hier zunächst auf die Evaluation der Vorhersagen konzentrieren.
In der vorgeschlagenen Evaluations-Pipeline werden verschiedene Dimensionen evaluiert und dabei ein Fokus auf die *Retrieval*- und Generierungs-Stufen eines typischen RAG-Systems gelegt. Wie in der Abbildung 2.3 zu sehen, startet der Evaluationsprozess mit dem *Retrieval*-Schritt, in dem die Kontextrelevanz als kritische Evaluationsdimension zu betrachten ist. Danach beginnt die Generierungs-Evaluation, in der die Evaluationsdimensionen Antwortrelevanz, Korrektheit, Faktentreue (engl. *faithfulness*) und Qualität der Quellenangaben (engl. *citation quality*) überprüft werden (Knollmeyer u. a. 2024, S. 142).
Für die Auswahl von Metriken und die Kalkulation der jeweiligen Evaluationsdimension ist der ausgewählte Evaluator zuständig. Hier werden durch (Knollmeyer u. a. 2024, s.142) drei verschiedene aufgeführt: Lexikalische Übereinstimmung (engl. *lexical matching*), Semantische Ähnlichkeit (engl. *semantic similarity*), und LLM als Richter (engl. *LLM as a judge*) (Knollmeyer u. a. 2024, S. 142). Der Unterschied liegt darin, ob der Evaluationsfokus auf exakter wörtlicher Übereinstimmung, konzeptueller Ähnlichkeit oder einer nuancierten Abwägung des Kontexts durch ein LLM liegt.

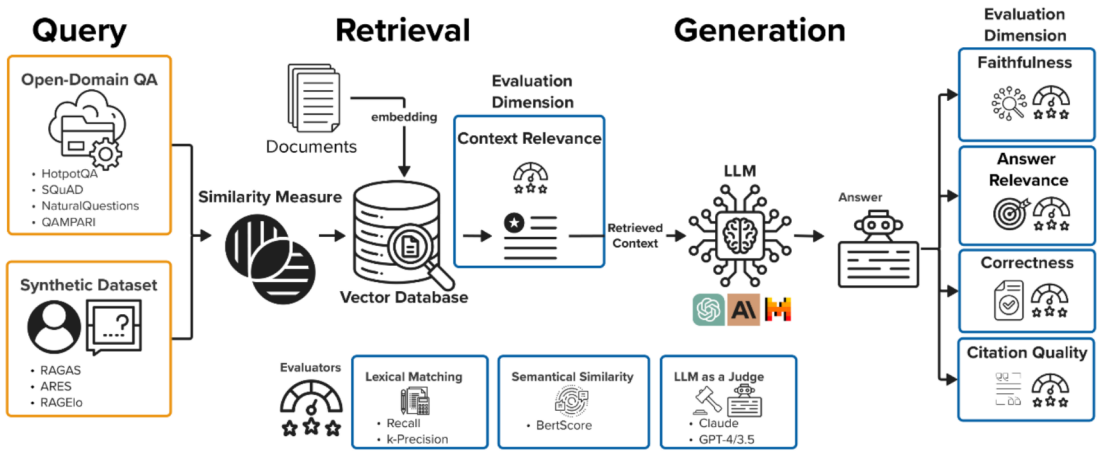

Abbildung 2.3: Evaluationsprozess mit Evaluationsdimensionen nach (Knollmeyer u. a. 2024, S. 142).

Durch (Knollmeyer u. a. 2024, S. 142ff.) werden die Evaluationsdimensionen genauer dargelegt . Hier soll ein kurzer Überblick erfolgen, um eine Grundlage für das spätere Evaluationsdesign zu schaffen. Details zur Implementierung werden dann in Kapitel xxx dargelegt.

Kontextrelevanz

Kontextrelevanz ist die essentielle Evaluationsdimension aus der *Retrieval*-Phase im RAG-Modell. Es bewertet, inwiefern der abgerufene Kontext nur wichtigen Informationen enthält, die nötig sind, um die Abfrage zu beantworten. Durch die Minimierung irrelevanten Kontexts werden Rechnerressourcen gespart und die Effizienz gesteigert. Zudem ist die Konzentration auf das Wesentliche auch deshalb wichtig, weil LLMs derzeit noch viel schlechter dazu in der Lage sind, aus größeren Sinnzusammenhängen relevante Informationen abzurufen als aus kleineren (Knollmeyer u. a. 2024, S.142).

Faktentreue

Die Faktentreue evaluiert (wie die folgenden Evaluationsdimensionen) die Generierungsphase eines LLM. Hier ist die Frage, wie sehr die Antwort des LLMs auf eine Eingabe in dem abgerufenen Kontext eingebettet ist. Das Ziel wäre, dass in einem RAG-Modell alle generierten Fakten direkt aus dem abgerufenen Kontext ableitbar ist. Dies ist ein wichtiger Evaluationsschritt, um Halluzinationen zu identifizieren, indem eine Diskrepanz aus generierter Antwort und zugrundeliegenden Fakten festgestellt wird (Knollmeyer u. a. 2024, S. 143).

Antwortrelevanz

Ebenfalls in der Generierungsphase verortet ist die Antwortrelevanz, die bewertet, ob das LLM die Eingabe direkt beantwortet. Hier werden unvollständige oder redundante Antworten bestraft, auch wenn sie inhaltlich korrekt wären.

Korrektheit

Bei der Korrektheit wird innerhalb der Generierungsphase evaluiert, wie

Kapitel 3

Konzeption des RAG-Prototyps

3.1 Zieldefinition und Anforderungen

3.2 Architekturentwurf

Kapitel 4

Implementierung eines Minimalbeispiels

4.1 Technologiewahl und Setup

Open WebUI

Für die Umsetzung der lokalen Modellbereitstellung wurde Open WebUI als technische Grundlage gewählt. Die Plattform ist als vollständig selbst gehostetes System konzipiert und ermöglicht einen Betrieb ohne externe Cloud-Dienste. Dies entspricht den Anforderungen an Datenschutz, Kontrolle über die Modellumgebung und Reproduzierbarkeit, wie sie beispielsweise im Behördenumfeld anzutreffen sind. In der offiziellen Dokumentation wird Open WebUI als „extensible, feature-rich, and user-friendly“ beschrieben und unterstützt eine Vielzahl von LLM-Laufzeitumgebungen sowie OpenAPI-basierten Werkzeugen (*Open WebUI* 2025).
Für das Setup wurde bewusst die manuelle Bereitstellung über eine Python-basierte Installation gewählt. Dieser Ansatz bietet eine direkte Kontrolle über die Abhängigkeiten, die lokale Umgebung und die verwendete Python-Laufzeit. Open WebUI stellt hierzu Installationspfade über die Werkzeuge `uv` oder `pip` bereit. Diese ermöglichen es, die Anwendung ohne Containerisierung aufzusetzen und bei Bedarf gezielt anzupassen (*Open WebUI* 2025). Gleichzeitig erfordert dieser Ansatz eine sorgfältige Konfiguration der Systemumgebung, einschließlich der Installation der benötigten Pakete sowie der Sicherstellung, dass WebSocket-Verbindungen unterstützt werden, da diese für den Betrieb der Benutzeroberfläche notwendig sind.
Die Dokumentation beschreibt zusätzlich eine Docker-basierte Bereitstellung als alternative Methode. Diese Variante bündelt sämtliche Abhängigkeiten in einem Container und erleichtert die Ausführung auf Systemen mit GPU-Unterstützung. Ein entsprechender Startbefehl wird in Form eines standardisierten Docker-Kommandos bereitgestellt. Die containerisierte Bereitstellung wurde in diesem Projekt jedoch nicht eingesetzt, da der Schwerpunkt auf einer fein steuerbaren lokalen Installation lag.
Zusammengefasst ermöglicht Open WebUI eine flexible Umgebung für die Ausführung großer Sprachmodelle, wobei die manuelle Installation eine hohe Transparenz und Anpassbarkeit bietet. Gleichzeitig erleichtert die alternative Docker-Variante den Einsatz in standardisierten Systemumgebungen.

GPT-oss-120B

Die Entscheidung für den Einsatz von GPT-oss-120B wurde maßgeblich durch veröffentlichte Evaluationsergebnisse gestützt, die eine hohe Leistungsfähigkeit des Modells belegen. In einem umfassenden Benchmark, der mehrere programmierbezogene Aufgaben umfasste, erzielte das Modell durchgehend starke Resultate. So wurden unter anderem Bewertungen von bis zu 8,5

von 10 Punkten erreicht, wobei der Gesamtdurchschnitt mit rund 8,3 nur geringfügig hinter den führenden proprietären Modellen lag und deutlich über vergleichbaren Open-Source-Gewichten (*gpt-oss-120b Coding Evaluation* o. D.). Zusätzlich zeigte das Modell eine hohe Verarbeitungsgeschwindigkeit von über 400 Token pro Sekunde, was insbesondere für interaktive oder iterative Arbeitsprozesse von Vorteil ist (*gpt-oss-120b Coding Evaluation* o. D.). Aufgrund dieser Kombination aus inhaltlicher Leistungsstärke und technischer Effizienz stellte GPT -oss -120B im Rahmen der Modellabwägung die ausgewogenste Lösung dar und wurde daher als Grundlage der weiteren Arbeiten gewählt.

4.2 Umsetzung der Komponenten

Für den MACH-LLM-Prototypen müssen folgende Komponenten in der OpenWebUI-Oberfläche erstellt werden: eine Wissensbasis und ein vortrainiertes LLM. Für die Wissensbasis wird in drei Arten von Dokumenten unterschieden, die im IT-Betrieb der MACH-Software in der untersuchten Organisation eine Rolle spielen:

- offizielle Dokumentationen des Software-Herstellers (MACH AG), vorliegend im PDF-Format
- informelle technische Konfigurationsdokumentationen der Systemadmins, vorliegend hauptsächlich online (Wiki)
- fachliche Dokumentationen, wie das Rechte- und Rollenkonzept oder Leitfaden für Benutzerpflege, vorliegend im PDF-Format

Die Wissensbasis ist erstmal schnell erstellt. Das Open WebUI-Frontend bietet einen Dokumentenupload, in dem komfortabel per Drag'n'Drop Dokumente hochgeladen werden können. Nach dem Upload werden die Dokumente geparsed und in die Wissensdatenbank aufgenommen. Es werden für den MVP jeweils ein bis zwei Dokumente pro oben genannter Dokumentkategorie hochgeladen, um aussagekräftige Beispiele zu erhalten. Für die technischen Dokumentationen wurde zunächst ein Export der Wiki-Seiten über die integrierte Export-Funktion vorgenommen. Damit lagen die Daten im HTML-Format vor.
Nach der Erstellung der Wissensdatenbank wird das GPT-oss-120B-Modell mit der Wissensdatenbank verknüpft. Dem Modell muss ein Systemprompt verabreicht werden. Dieser steuert das Grundverhalten des Modells. Hier wurde ein Systemprompt mithilfe von OpenAI ChatGPT 5.1 verfasst.
TODO: Insert Systemprompt
Danach ist der RAG-Prototyp einsatzbereit für erste Chatanfragen. Es wurden

Kapitel 5

Evaluation

5.1 Evaluationsdesign

5.2 Ergebnisse (Dummy-Data)

Kapitel 6

Diskussion

Kapitel 7

Fazit und Ausblick

Literatur

Asai, Akari u. a. (17. Okt. 2023). *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. arXiv: 2310.11511[cs].

Bommasani, Rishi u. a. (12. Juli 2022). *On the Opportunities and Risks of Foundation Models*. arXiv: 2108.07258[cs].

Chang, Yupeng u. a. (29. Dez. 2023). *A Survey on Evaluation of Large Language Models*. arXiv: 2307.03109[cs].

Devlin, Jacob u. a. (o. D.). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: ().

Gao, Yunfan u. a. (27. März 2024). *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv: 2312.10997[cs].

gpt-oss-120b Coding Evaluation (o. D.). *gpt-oss-120b Coding Evaluation: New Top Open-Source Model*. 16x Eval. URL: https://eval.16x.engineer/blog/gpt-oss-120b-coding-evaluation-results (besucht am 13. 11. 2025).

Jiang, Zhengbao u. a. (22. Okt. 2023). *Active Retrieval Augmented Generation*. arXiv: 2305.06983[cs].

Knollmeyer, Simon u. a. (2024). "Benchmarking of Retrieval Augmented Generation: A Comprehensive Systematic Literature Review on Evaluation Dimensions, Evaluation Metrics and Datasets:" In: *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 16th International Conference on Knowledge Management and Information Systems. Porto, Portugal: SCITEPRESS - Science und Technology Publications, S. 137–148.

Open WebUI (13. Nov. 2025). OpenWebUI. URL: https://openwebui.com/ (besucht am 13. 11. 2025).

Zhang, Yue u. a. (14. Sep. 2025). *Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models*. arXiv: 2309.01219[cs].