BHT Berliner Hochschule für Technik

Semantische Kontextualisierung in RAG-Systemen

Ein Vorgehensmodell zur LLM-gestützten Vorverarbeitung strukturierter Wiki-Inhalte

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 26. Januar 2026

Studiere Zukunft

Version 1.0α
letzte Änderung: 26. Januar 2026

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

Inhaltsverzeichnis

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit mit dem Titel

Semantische Kontextualisierung für RAG-Systeme
Ein Vorgehensmodell zur LLM-gestützten Vorverarbeitung strukturierter Wiki-Inhalte

selbstständig und ohne unerlaubte Hilfe angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Sofern im Rahmen der Erstellung dieser Arbeit Softwarewerkzeuge (z. B. Textverarbeitungs-, Übersetzungs- oder KI-basierte Assistenzsysteme) eingesetzt wurden, erfolgte dies ausschließlich unterstützend; die inhaltliche Verantwortung für Auswahl, Bewertung und Einordnung der Ergebnisse sowie die finale Ausarbeitung liegt vollständig bei mir. Alle verwendeten Quellen und Hilfsmittel sind im Literatur- bzw. Hilfsmittelverzeichnis angegeben.

Ort, Datum: _____

Unterschrift: _____

Kapitel 1

Einleitung

In großen Organisationen stellt nicht nur der Wissenserwerb, sondern insbesondere der nachhaltige Wissenstransfer eine zentrale Herausforderung dar. Über Jahre gewachsene Strukturen, verteilte Zuständigkeiten und heterogene Dokumentationspraktiken führen dazu, dass relevantes Wissen über IT-Systeme auf verschiedene Stellen, Formate und Verantwortlichkeiten verteilt ist. Insbesondere im behördlichen IT-Umfeld wird dieses Wissen häufig in internen Wikis, Servicedokumentationen und herstellerspezifischen PDF-Handbüchern festgehalten, deren Qualität, Aktualität und Struktur stark variieren. Dadurch entsteht eine hohe Abhängigkeit von einzelnen Administrator*innen, während Vertretungs- oder Übergabesituationen nur eingeschränkt unterstützt werden. Obwohl diese Dokumentationen grundsätzlich ein hohes Potenzial für das Wissensmanagement besitzen, erschweren ihre Verstreutheit sowie eingeschränkte Such- und Zugriffsmöglichkeiten eine effektive Nutzung. Vor diesem Hintergrund liegt es nahe, bestehende Wissensquellen mithilfe von Large Language Models (LLMs) aufzubereiten und über einen Chatbot-gestützten Zugriff nutzbar zu machen.

Der RAG-Ansatz (*Retrieval Augmented Generation*, vgl. Abschnitt 3.2) ermöglicht es, externe Dokumente in die Antwortgenerierung vortrainierter Large Language Models (LLMs, vgl. Abschnitt 3.1) einzubeziehen. Die relevanten Dokumente werden anhand einer Anfrage abgerufen und dem LLM zusammen mit dem Prompt bereitgestellt, sodass quellenbasierte Antworten erzeugt werden können. Während dieser Ansatz für natürlichsprachlichen Fließtext gut funktioniert [Lewis u. a. 2021], stößt er bei der Verarbeitung strukturierter und semistrukturierter Inhalte an seine Grenzen. Insbesondere tabellarische Darstellungen und Übersichten liefern dem LLM häufig zu wenig expliziten Kontext, um die enthaltenen Informationen adäquat zu erfassen und in Beziehung zu setzen [Sui u. a. 2023]. Dieses Problem tritt im IT-Betrieb besonders deutlich auf, da technische Dokumentationen stark von strukturierten Darstellungen wie Tabellen zu IP-Adressräumen, Portfreigaben oder Systemumgebungen geprägt sind. Für domänenspezifische, semistrukturierte IT-Administrationsdaten, wie sie typischerweise in internen Wikis vorliegen, existiert bislang nur begrenzte Forschung im Kontext von RAG-Systemen.

Daraus ergibt sich die folgende Forschungsfrage: Inwiefern trägt eine LLM-gestützte semantische Kontextualisierung strukturierter Wiki-Inhalte zur verbesserten Nutzbarkeit dieser Inhalte in RAG-Systemen für das IT-administrative Wissensmanagement bei? Der Arbeit liegt die folgende Hypothese zugrunde: Die LLM-gestützte semantische Kontextualisierung strukturierter Wiki-Inhalte führt im Vergleich zu nicht kontextualisierten Tabellen zu einer verbesserten funktionalen Nutzbarkeit der resultierenden Wissensdokumente in RAG-Systemen.

Zur Beantwortung der Forschungsfrage und zur Überprüfung der zugrunde liegenden Hypothese wird in dieser Arbeit ein LLM-gestütztes Vorgehensmodell zur Vorverarbeitung strukturierter Wiki-Daten entwickelt. Die Vorverarbeitung erfolgt vor der Integration der Daten in eine RAG-

Wissensbasis und zielt darauf ab, strukturierte und semistrukturierte Inhalte für die nachgelagerte Nutzung durch ein RAG-System aufzubereiten.

Die Vorverarbeitung umfasst zunächst ein Parsing der extrahierten Wiki-Dokumente, da diese initial im HTML-Format vorliegen. Dieses Parsing umfasst zunächst eine Bereinigung der Daten sowie die Umwandlung in eine Objektstruktur. Im nächsten Schritt werden ausgewählte Inhalte, insbesondere Tabellen und Übersichten, mithilfe eines LLMs semantisch kontextualisiert und in erklärenden Fließtext überführt. Dadurch entstehen prosebasierte Wissenseinheiten, die von einem RAG-System ohne spezielle Behandlung strukturierter Daten verarbeitet werden können.

Zunächst werden in Kapitel 3 die Grundlagen zu Large Language Models und Retrieval Augmented Generation dargelegt. Danach wird in Kapitel 4 der Forschungsstand zu RAG im Zusammenhang mit strukturierten Daten beschrieben und in Kapitel 5 die hier vorgeschlagene Methode zur Überbrückung der Forschungslücke.

Die entwickelte Pipeline[1] wird in Kapitel 6 näher beschrieben. Die so aufbereiteten Dokumente bilden anschließend die Grundlage einer Wissensbasis, die für einen Chatbot im Rahmen eines RAG-Ansatzes genutzt wird. In Kapitel 7 erfolgt eine qualitative, funktionale Evaluation des resultierenden RAG-Systems, die zugleich Rückschlüsse auf die Qualität der Vorverarbeitung erlaubt. Die Ergebnisse werden in Kapitel 8 diskutiert und in Kapitel 9 zusammengefasst.

Die Beiträge dieser Arbeit lassen sich wie folgt zusammenfassen:

- Entwicklung eines LLM-gestützten Vorgehensmodells zur semantischen Kontextualisierung strukturierter Wiki-Inhalte als vorgelagerter Schritt in RAG-Systemen.
- Konzeptionelle Trennung zwischen deterministischer Vorverarbeitung (Parsing, Strukturierung) und probabilistischer, LLM-basierter Transformation strukturierter Inhalte in prosebasierte Wissenseinheiten.
- Qualitative, funktionale Evaluation der Auswirkungen der kontextualisierten Wissensbasis auf die Nutzbarkeit eines RAG-Systems im domänenspezifischen Kontext des IT-administrativen Wissensmanagements.

Diese Arbeit adressiert damit eine Forschungslücke im Spannungsfeld zwischen RAG-Systemen und domänenspezifischem, strukturiertem Wissen aus administrativen Wikis und zeigt anhand einer prototypischen Umsetzung, wie eine LLM-gestützte Vorverarbeitung zur verbesserten Nutzbarkeit solcher Wissensbestände beitragen kann.

[1] `https://github.com/free-da/struct2prose`

Kapitel 2

KI-Nutzung

Als Hilfsmittel für die Erstellung der Arbeit wurde KI benutzt, speziell in der Form von OpenAI ChatGPT 5.0, ChatGPT 5.1 sowie ChatGPT 5.2[1]. Zunächst wurde für die Themenfindung viel mit dem Chatbot gechattet, wobei eine grobe Idee in Richtung „RAG-LLM für Wissensmanagement in IT-Organisation“ von der Verfasserin dieser Arbeit vorgeschlagen wurde. Diese hat sich mit der Zeit immer weiter verfeinert, während von dem Chatbot wertvolle Hinweise gegeben wurden. Der Chatbot hat auch bei einem theoretischen Einstieg in das Thema geholfen, indem er Literaturvorschläge gemacht hat, und das Forschungsfeld erläuterte. Auch beim Lesen und Verstehen der Texte hat der Chatbot geholfen, Sachverhalte zu erläutern, sowie beim Übersetzen von englischen Fachbegriffen ins Deutsche. Später hat der Chatbot dazu beigetragen, die Arbeit zu strukturieren und zu gliedern und auch bei der Herangehensweise konnte er wertvolle Tipps geben. Der Chatbot hat die ganze Arbeit begleitet, indem er bei technischen und auch psychologischen Hürden half, einen neuen Ansatz zu finden. Er diente der Verfasserin sowohl als Coach als auch als „Sparring-Partner“ in der Evaluation von Ideen, Konzepten und Herangehensweisen.

Inhaltlich hat der Chatbot die ersten Dummy-Daten und auch erste Code-Snippets für die Python-Skripte erzeugt. Diese wurden von der Verfasserin der Arbeit kuratiert, bewertet, überarbeitet, verbessert und weiterentwickelt. Die Nutzung der KI diente immer dem Einstieg und dem Befähigen zur Weiterentwicklung. Die daraus entstandenen Gedanken, Texte und Skripte sind die der Verfasserin.

[1] `https://chatgpt.com/`

Kapitel 3

Theoretische Grundlagen

In dieser Arbeit werden Grundlagen zu Large Language Modellen unabdingbar sein. Zunächst soll der Begriff der Large Language Models allgemein und der *Foundation Models* im Speziellen geklärt werden. Danach wird der RAG-Ansatz stärker beleuchtet.

3.1 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 Models* 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.

3.2 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].

In Abbildung 3.1 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

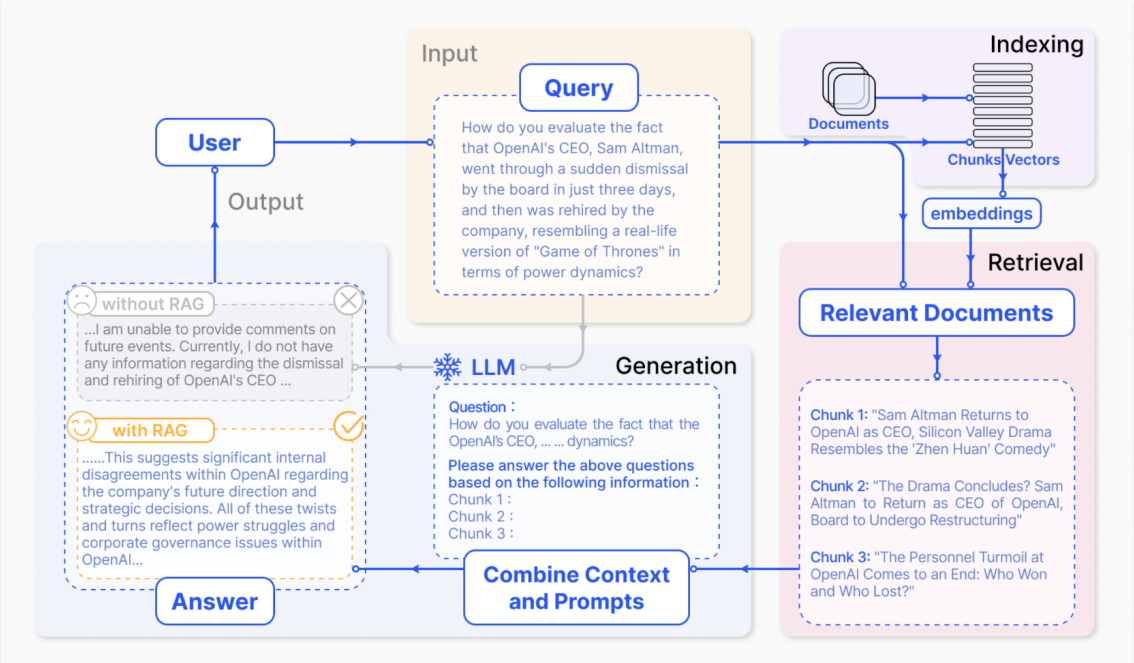

Abbildung 3.1: Überblick über die Funktionsweise von RAG nach [Gao u. a. 2024, S. 3]

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].

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

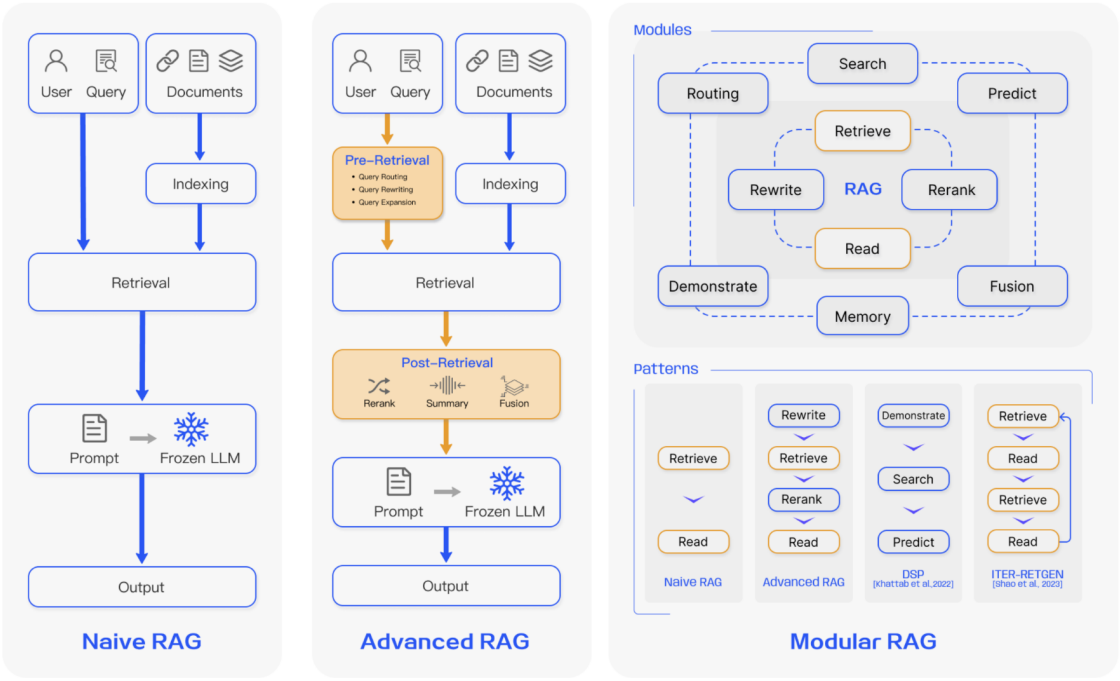

Abbildung 3.2: Überblick über *Naive RAG*, *Advanced RAG* und *Modular RAG* [Gao u. a. 2024, S. 3]

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].

Kapitel 4

Related Works

Verschiedene Arbeiten beschäftigen sich mit der Frage, wie LLM strukturierte und semi-strukturierte Daten erfassen und „verstehen“ können. In [Sui u. a. 2023] wird hierfür der Ansatz *self-augmented prompting* verwendet. In Abbildung 4.2 ist die Funktionsweise dieses Ansatzes dargestellt. Eine Anfrage, die auf strukturierten Daten, wie beispielsweise einer Tabelle, basiert, durchläuft das LLM in zwei Phasen. Bei der ersten werden wichtige Informationen und Wertebereiche der Tabelle durch das LLM identifiziert und beim zweiten Durchlauf werden natürlichsprachige Informationen für die relevanten Bereiche augmentiert.

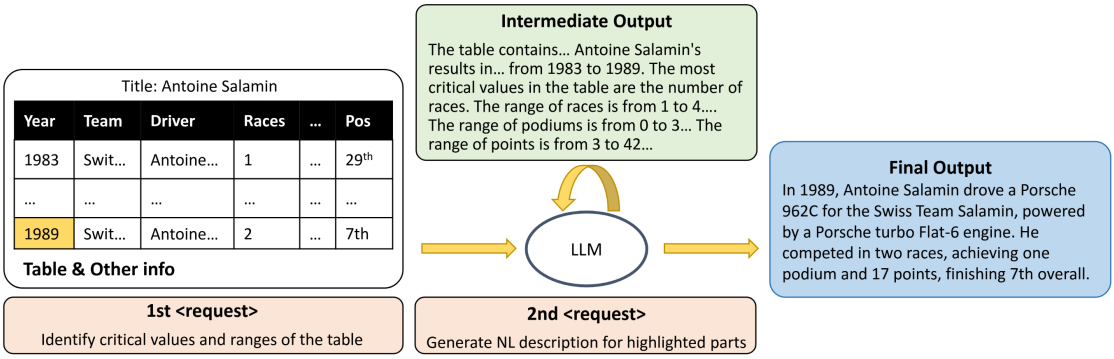

Abbildung 4.1: Illustration des *self-augmented promptong* nach [Sui u. a. 2023]

Im Unterschied zu der hier vorliegenden Arbeit ist in [Sui u. a. 2023] die Serialisierung in dem Abfrageschritt ans LLM enthalten. Hier soll die Datenvorverarbeitung als abgeschlossener Schritt passieren, bevor die Daten in die Wissensdatenbank des RAG-Modells einfließen. Der Vorteil liegt darin, dass die Datenvorverarbeitung so model-agnostisch funktioniert und auf beliebige Anwendungen anwendbar ist. Selbst wenn die Daten durch ein geschlossenes System weiterverarbeitet werden können, in dem kein Zugriff auf den LLM-Lifecycle besteht, können die Daten derart vorverarbeitet werden und die Anwendungen von der Pipeline profitieren.

In [Yu, Jian und Chen 2025] wird ein SQL-basiertes Framework namens *TableRAG* vorgestellt, was zum Ziel hat, die Grenzen des RAG-Ansatzes angewendet auf heterogene Dokumente, zu adressieren. Es besteht aus vier Schritten:

1. kontextsensitive Dekonstruktion der Eingabe-Querys
2. Text-Abruf
3. SQL-Programmierung

4. Ausführung

Zusätzlich gibt es ein Benchmark, *HeteQA*, was mehrschrittige heterogene *Reasoning*-Fähigkeiten evaluieren soll. Dieser Ansatz ist sehr fortgeschritten und befähigt sich durch die Nutzung von SQL und der Konstruktion einer Offline-Datenbank. Hier werden als zwei zentrale Defizite in der Verarbeitung von Tabellen durch RAG der Verlust struktureller Informationen und die fehlende globale Sicht auf die Daten genannt, denen entgegengewirkt werden will.

Dieser Ansatz strebt an, den *Retriever* im RAG-Ansatz zu verbessern, während in der vorliegenden Arbeit die Wissensbasis transformiert wird, sodass der RAG-Ansatz keine Besonderheit mehr zu bewältigen hat. Somit bleibt der hier vorliegende Versuch eine niedrigschwellig implementierbare Alternative zu aufwendigen RAG-Implementierungen.

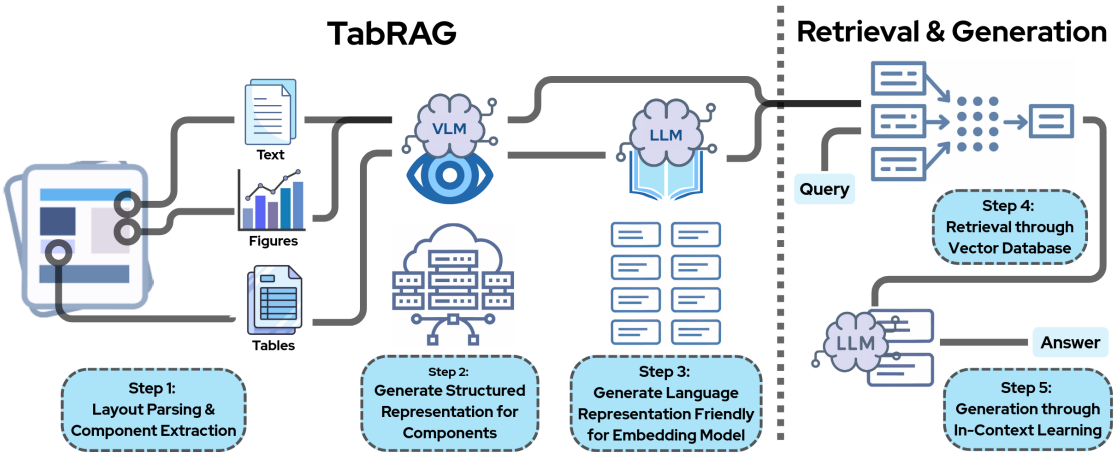

Abbildung 4.2: Illustration der TabRAG-Pipeline nach [Si u. a. 2025]

Ein für die vorliegende Arbeit sehr interessanter Ansatz ist der von TabRAG [Si u. a. 2025]. Auch hier wird eine Pipeline vorgeschlagen, die heterogene Daten vorverarbeitet, um die Extraktionsergebnisse von RAG zu verbessern.

TabRAG zeigt implizit, dass selbst hochspezialisierte, tabellenzentrierte RAG-Architekturen am Ende auf natürlichsprachliche Repräsentationen angewiesen sind. Im Gegensatz dazu setzt diese Arbeit bewusst früher an und transformiert strukturierte Wiki-Inhalte vollständig in prosebasierte Wissenseinheiten, sodass keine tabellenspezifischen Mechanismen im RAG-System erforderlich sind.

Der hier vorgeschlagene Ansatz soll implementierungsarm, leichtgewichtig und technologieoffen sein, indem er die Daten, die der RAG-Wissensbasis zugrundeliegen in Prosa transformiert.

Kapitel 5

Methode

Das wissenschaftliche Vorgehensmodell soll in diesem folgenden Kapitel erläutert werden. Zunächst wird im Unterkapitel 5.1 die Zielsetzung und Forschungslogik dargelegt. Hier wird die Forschungsfrage konkretisiert und das Forschungsdesign erläutert, außerdem kurz umrissen, was diese Arbeit nicht enthalten wird. Im Unterkapitel 5.2 wird das Vorgehen zur Beantwortung der Forschungsfrage erläutert. Im Unterkapitel 5.3 werden Entscheidungen zur Auswahl der Versuchsdaten dargelegt und im Unterkapitel 5.4 die angewandten Transformationsschritte erläutert.

5.1 Zielsetzung und Forschungslogik

Die zentrale Forschungsfrage dieser Arbeit lautet: „Inwiefern trägt eine LLM-gestützte semantische Kontextualisierung strukturierter Wiki-Inhalte zur verbesserten Nutzbarkeit dieser Inhalte in RAG-Systemen bei?“. Sie zielt darauf ab, die bekannte Schwäche von RAG-Modellen in der Verarbeitung strukturierter und semistrukturierter Daten in der Wissensbasis auszugleichen und somit Informationen, die in vielfältigen Formaten vorliegen, für ein RAG-System nutzbar zu machen.

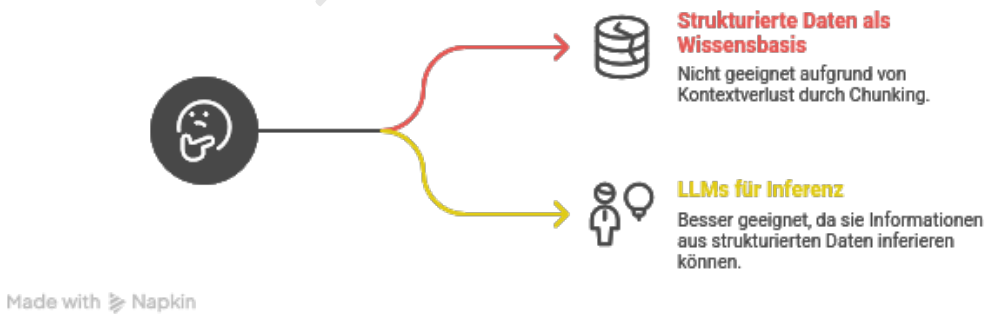

Abbildung 5.1: Unterschied in der Fähigkeit zur Verarbeitung strukturierter Daten

Strukturierte Daten finden wir in der Nutzung interner IT-Dokumentationen zuhauf: Dokumentationen über Benutzerpflege, Inventarisierungen und Konfigurationen von IT-Systemen im Allgemeinen. Da keine formelle Redaktion vorhanden ist, schreiben alle Administrator*innen ih-

re Dokumentationen selbst. Dies führt zu einer strukturellen Heterogenität der Daten in der Wissensrepräsentation. Zusammen mit der Schwäche von RAG-Systemen bei der Verarbeitung strukturierter Daten ergibt sich ein Problem, das nicht trivial zu lösen ist.

An dieser Stelle ist eine Klärung wichtig: obwohl RAG-Systeme wie dargestellt nicht ausreichend in der Lage sind, mit strukturierten Daten als Wissenbasis umzugehen, können LLMs an sich sehrwohl strukturierte Daten verarbeiten und daraus Informationen inferieren. Dieser Unterschied wird in Abbildung 5.1 dargestellt und soll in der folgenden Arbeit zunutze gemacht werden.

Das Ziel dieser Arbeit ist, Erkenntnis darüber zu schaffen, inwiefern die Vorverarbeitung der Daten eine erhebliche Verbesserung in ihrer Nutzbarkeit durch ein RAG-Modell erreichen kann. Der Weg zu diesem Ziel ist eine design-orientierte Untersuchung, in der explorativ und artefaktbasiert eine solche Vorverarbeitung durchgeführt, und das Ergebnis mit einem vergleichbaren System ohne diese Vorverarbeitung unter Beibehaltung identischer Parameter verglichen wird.

Das zentrale Artefakt ist eine mehrstufige LLM-gestützte Vorverarbeitungs- und Transformationspipeline, die durch mehrere Transformationsschritte reproduzierbar und überprüfbar das Format der zugrundeliegenden Daten an die Verarbeitungsweise eines RAG-Systems anpasst. Die Überbrückung der Lücke zwischen heterogenen strukturierten Rohdaten und der begrenzten Fähigkeit von RAG-Systemen, strukturierte Daten ohne vorgelagerte Aufbereitung sinnvoll zu nutzen durch die semantische Kontextualisierung bietet die Grundlage der Nutzbarmachung von Wiki-Daten für ein RAG-LLM mit domänenspezifischen, internen IT-Daten.

Hier werden gewisse Vereinfachungen bewusst gemacht. Für eine komplette Pipeline wären Schnittstellendefinitionen für den Ein- und Ausgang der Daten nötig, die hier händisch gemacht wurden. Diese Entscheidung liegt darin begründet, dass diese Arbeit eine Prüfung der Transformation sein soll. Erst im Falle einer positiven Bewertung dieses Ansatzes ist es sinnvoll, sich auf konkrete APIs für Ein- und Ausgang der Pipeline oder andere technische Werkzeuge festzulegen.

Desweiteren finden kein Benchmarking in der Evaluation und auch keine Nutzerstudien statt. Es werden auch keine LLMs oder RAG-Systeme miteinander verglichen, da der Fokus auf der Vorverarbeitung der Daten liegt und nicht auf der Bewertung eines LLMs. Das fertige RAG-LLM oder gar ein Chatbot als Anwendungsbereich für das Wissensmanagement könnten das Ergebnis weiterer Arbeiten sein, die auf dieser Arbeit aufbauen. Diese Arbeit ist aber explizit eine Untersuchung über die Tauglichkeit dieses konkreten Vorverarbeitungs-Ansatzes.

5.2 Überblick über das Vorgehensmodell

Die vorgeschlagene Lösung für das Problem der Verarbeitung von strukturierten Daten bei RAG-Systemen ist ein Vorgehensmodell, was strukturierte Wiki-Inhalte in eine RAG-geeignete Wissensrepräsentation überführt. Operationalisiert wird dieses Vorgehensmodell mit einer Datenvorverarbeitungs- und Transformationspipeline, die in drei Schritten die Wiki-Inhalte transformiert. Der dritte Schritt ist das Kernstück dieser Pipeline: die LLM-gestützte semantische Kontextualisierung strukturierter Inhalte in Fließtext.

Die drei Transformationsschritte sind die folgenden:

1. **DOM-Vorverarbeitung** - Hier findet die strukturelle Bereinigung des Dokuments statt, in der überflüssige Informationen über Präsentation und Navigation verworfen werden.
2. **Strukturelles Parsing** - Dies ist der Schritt, in dem die bereinigten Informationen in eine interne Objektstruktur überführt werden.

3. **Semantische Kontextualisierung** - An dieser Stelle wird die bereinigte Objektrepräsentation an ein LLM übergeben mit dem Ziel, diese in einen kontextreichen Fließtext umzuformulieren.

Die kontextualisierten Daten dienen als Grundlage zur Evaluation.

5.3 Datenselektion und Datenextraktion

Die Rohdaten, die transformiert werden sollen, entstammen einem Wiki zur internen Dokumentation einer größeren IT-Organisation. Sie sind textlastig, aber durch den vermehrten Einsatz von Tabellen, Listen und Übersichten sehr heterogen, mit semi-strukturierten bis strukturierten Fragmenten.

Es wurde sich auf die Auswahl von fünf geeigneten Wiki-Seiten beschränkt. Diese wurden ausgewählt, da sie von der Autorin dieser Arbeit selbst verfasst wurden, und somit Fragen der Urheberschaft zu eindeutig geklärt sind. Außerdem sind sie dadurch ähnlicher in der inhaltlichen Tiefe, als wenn sie von mehreren Autor*innen verfasst worden wären. Darüber hinaus finden sich in den Seiten verschiedene semistrukturierte Repräsentationen von Informationen, wie beispielsweise Konfigurationstabellen oder Datenbankübersichten. Dies ist interessant, da hier leicht zu beobachten ist, wie die geplante Transformation die Tauglichkeit der Daten für die Verwendung im RAG-LLM verändert. Die Auswahl der Eingangsdaten wurde bewusst auf diese fünf Seiten eingegrenzt, da der Fokus dieser Arbeit auf der grundsätzlichen Erprobung des Vorgehensmodells liegt, und nicht auf der Anwendbarkeit oder Skalierung.

Die Extraktion der Seiten manuell durchgeführt, sodass reine Textdateien lokal gespeichert wurden. Es wurde auf die Sonderbehandlung von Bildern und Anhängen verzichtet, beziehungsweise wurde bei der Auswahl der Beispielseiten gezielt darauf geachtet, dass keine wichtigen Bilder enthalten sind. Die bewusste Entscheidung der manuellen Extraktion liegt darin begründet, dass die Datentransformation im Fokus dieser Arbeit liegt. Die manuelle Extraktion ist einfach und zweckdienlich und bietet eine Entkopplung dieses Schritts von der Transformation.

Nach der Extraktion fand eine Anonymisierung statt, bei der Zeichenketten, die auf den Unternehmenskontext hinweisen könnten, durch arbiträre Zeichenketten ersetzt wurden. Dazu gehörten die Web-Domäne, der Unternehmensname, der Nutzername der Bearbeiterin und weitere organisationsinterne Namen und Bezeichnungen. Dieser Schritt war nötig, um keine Bedenken bezüglich der Verarbeitung personenbezogener oder anderer sensibler Daten auf privater Hardware oder im Internet haben zu müssen.

Die Auswahl der Daten bedeutet natürlich auch eine Einschränkung für die Ergebnisse dieser Arbeit. Die hier erarbeitete Pipeline dient zur generellen Erprobung des vorgeschlagenen Vorgehensmodells. Im Falle einer positiven Evaluation ist weitere Arbeit zur Anwendbarkeit auf weitere Domänen, Portierung verschiedener Datenquellen, Automatisierung des Datenein- und Ausgangs und insbesondere zur Skalierung notwendig.

5.4 Regelbasierte vs. LLM-basierte Transformation

An dieser Stelle soll einmal gezielt darauf eingegangen werden, warum in zwei der Transformationsschritte ein regelbasierter Ansatz und nur beim dritten Transformationsschritt ein probabilistischer Ansatz gewählt wurde. Zunächst werden die beiden Ansätze konzeptionell eingeordnet.

Eine regelbasierte Transformation bezeichnet die Veränderung von Daten aufgrund von festgelegten Regeln. Dieser Ansatz ist deterministisch, und damit reproduzierbar und transparent. Wenn dieselbe Transformation vielfach ausgeführt wird, soll stets dasselbe Ergebnis erzeugt wer-

den. Die Ergebnisse lassen sich stets anhand der Regeln ableiten und theoretisch sollten sogar die Ursprungsdaten aus den transformierten Daten wiederhergestellt werden.

Eine LLM-basierte Transformation unterscheidet sich von der regelbasierten Transformation, indem sie probabilistisch ist, also auf Wahrscheinlichkeiten basierend. Die Ergebnisse einer solchen Transformation können bei wiederholten Durchläufen leicht variieren, insofern als dass sie unterschiedlich formuliert sein können oder Informationen unterschiedlich gewichtet werden. Somit ist dieser Ansatz nicht vollständig deterministisch und man kann vom Ergebnis auch nicht eindeutig auf die Eingabe rückschließen.

Für die DOM-Verarbeitung und das strukturelle Parsing, also Schritt 1 und 2 in der Vorverarbeitungs-Pipeline, wurde ein regelbasierter Ansatz gewählt, da es sich hier um klare Strukturen und formale Kriterien handelt, anhand derer das gewünschte Ergebnis erreicht werden kann. Für diese Vorverarbeitungsschritte ist kein semantisches Verständnis der Informationen nötig, somit wäre ein LLM-basierter Ansatz nicht sinnvoll. Ein LLM sollte aus Gründen der Datensparsamkeit, Ressourcenschonung und Nachvollziehbarkeit nur eingesetzt werden, wenn eine regelbasierte nicht ausreicht.

Beim dritten Verarbeitungsschritt, der semantischen Kontextualisierung, ist jedoch genau dies der Fall. Hier reichen Regeln nicht aus, da aufgrund der Heterogenität der Datenlage ein semantisches Verständnis unerlässlich ist. Strukturierte und semistrukturierte Daten müssen aus dem Kontext heraus erfasst und verstanden werden. Hier ist der Einsatz eines LLM angemessen, da es in der Lage ist, die Bedeutung der Daten zu erschließen, sie in einen Kontext einzubetten und entsprechend der Anweisung umzuvormulieren.

Eine Trennung dieser beiden Ansätze in Form von drei Verarbeitungsschritten, ist sinnvoll, da die regelbasierte Vorverarbeitung von Schritt 1 und 2 die Grundlage für die Verwendung des LLM in Schritt 3 bieten. Durch den probabilistischen Charakter der semantischen Kontextualisierung ist die Nachvollziehbarkeit nicht in demselben Maße gegeben, wie bei der regelbasierten Transformation. Durch die Kontrolle über den Eingang der Daten in den probabilistischen Transformationsschritt wird Fehlern vorgebeugt und die Qualität der Ergebnisse erhöht. Dies ermöglicht einen sparsamen Einsatz eines LLM zugunsten besserer Kontrollierbarkeit, einer klareren Evaluation und einer besseren Reproduzierbarkeit.

Die Forschungsfrage zielt auf die Erprobung der semantischen Kontextualisierung zur Nutzbarmachung strukturierter Daten in einem RAG-System ab, aber für eine gezielte und reproduzierbare Untersuchung des Ansatzes ist der Einsatz von sowohl regelbasierter als auch LLM-gestützter Vorverarbeitungsschritte notwendig. Die Trennung zwischen den drei Vorverarbeitungsschritten ist sinnvoll, da dadurch die Möglichkeit besteht, den Transformationsprozess transparent zu überwachen und bei Fehlern gezielt nachzusteuern.

Kapitel 6

LLM-basierte Preprocessing-Pipeline

Dieses Kapitel beschreibt die konkrete Ausgestaltung der im vorherigen Kapitel eingeführten Vorverarbeitungspipeline zur Transformation strukturierter und semistrukturierter Wiki-Inhalte in eine für RAG-Systeme geeignete Wissensbasis. Die Pipeline ist dabei als vorgelagertes Preprocessing-System konzipiert, das zwischen dem Quellsystem (XWiki) und dem eigentlichen RAG-Modell angesiedelt ist. Sie ist nicht Bestandteil des RAG-Systems selbst, sondern stellt eine vorbereitende Infrastruktur dar, deren Aufgabe in der strukturierten Aufbereitung und semantischen Kontextualisierung der zugrundeliegenden Inhalte liegt.

Aus systemischer Sicht übernimmt die Pipeline die Verantwortung für die Transformation der Darstellungs- und Organisationsform von Wissen, nicht jedoch für dessen inhaltliche Erweiterung oder Bewertung. Ausgangspunkt sind HTML-basierte Wiki-Seiten, die neben Fließtext in hohem Maße strukturierte Elemente wie Tabellen, Listen und Konfigurationsfragmente enthalten. Diese Strukturen sind für menschliche Leser gut erfassbar, erweisen sich jedoch als problematisch für die direkte Nutzung in retrievalbasierten Sprachmodellen. Die Pipeline adressiert diese Diskrepanz, indem sie solche Inhalte in prosebasierte, kontextualisierte Wissenseinheiten überführt, ohne dabei externes Wissen einzubringen oder bestehende Aussagen semantisch zu verändern.

Die Pipeline operiert dabei offline und batch-orientiert und ist zeitlich vor der Indexierung der Inhalte in einem Vektorspeicher angesiedelt. Sie greift weder in den Retrieval- noch in den Generationsprozess des RAG-Systems ein und trifft keine Entscheidungen zur Laufzeit von Nutzeranfragen. Ihr Ergebnis ist ein persistentes Korpus kontextualisierter Texte, das als Eingabe für nachgelagerte RAG-Komponenten dient und deren Nutzbarkeit maßgeblich beeinflusst.

Kern des Vorgehens ist eine LLM-gestützte semantische Kontextualisierung, bei der strukturierte oder semistrukturierte Inhalte gezielt in Fließtext überführt werden. Diese Kontextualisierung stellt den einzigen probabilistischen Verarbeitungsschritt innerhalb der Pipeline dar und ist bewusst von deterministischen Vorverarbeitungsschritten wie Dokumentbereinigung, strukturellem Parsing und Chunking getrennt. Auf diese Weise wird eine klare Trennung zwischen regelbasierter Strukturverarbeitung und probabilistischer Sprachmodellnutzung realisiert.

Im Folgenden wird zunächst die konkrete Ausgestaltung dieser Pipeline beschrieben. Dazu werden das eingesetzte Prompt-Design für die semantische Kontextualisierung strukturierter Inhalte, die zugrunde liegende Architektur sowie die technische Implementierung der einzelnen Verarbeitungsschritte dargestellt.

6.1 Architektur

Das Vorgehen folgt dem Prinzip einer mehrstufigen ETL-artigen Datenpipeline, in der Rohdaten aus einem Wiki extrahiert, schrittweise transformiert und schließlich in eine für RAG-Systeme geeignete Wissensbasis überführt werden. Ziel der Architektur ist eine modulare, reproduzierbare Datenpipeline, in der regelbasierte und LLM-basierte Transformationsschritte klar voneinander getrennt sind. Die dafür nötigen Schritte sind in Abbildung 6.1 zu sehen und werden im Folgenden näher beschrieben.

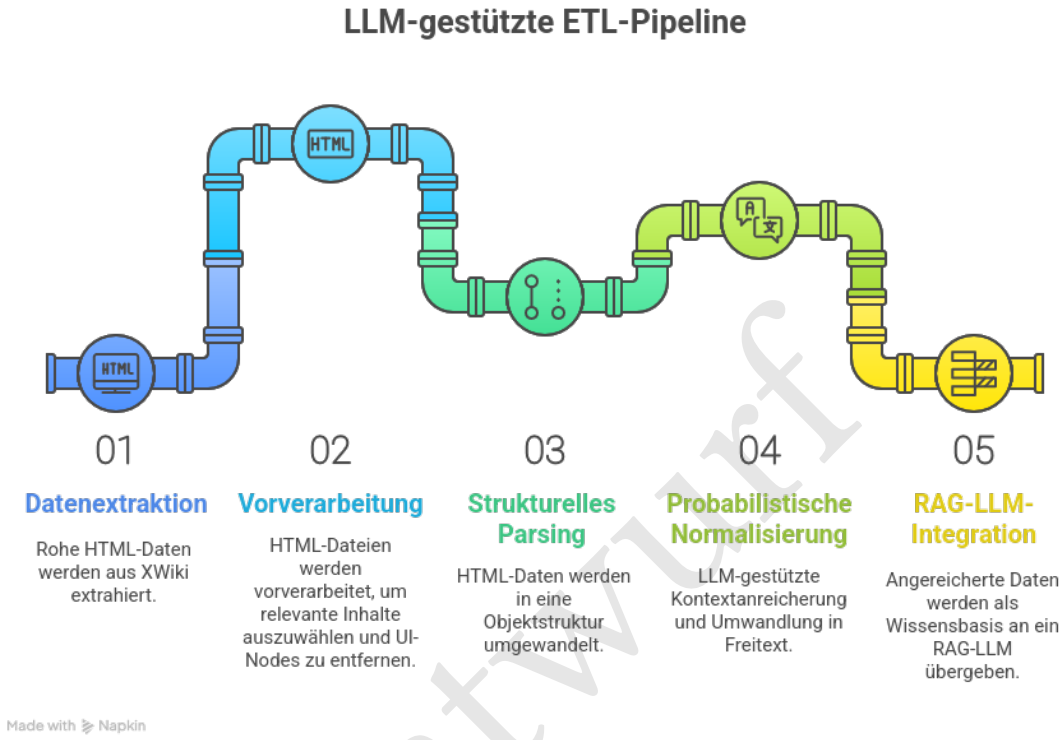

Abbildung 6.1: Illustration der Datenvorverarbeitung

6.1.1 Datenextraktion

Der Extraktionsschritt stellt den Einstiegspunkt der Pipeline dar und umfasst die Bereitstellung der Rohdokumente in einem einheitlichen Eingangsformat. Da wie in 5.3 beschrieben, eine händische Extraktion stattfand, wird dieser Punkt in dieser Arbeit nicht weiter beachtet.

6.1.2 DOM-Vorverarbeitung

An dieser Stelle beginnt in der praktischen Umsetzung der erste regelbasierte Transformationsschritt. Hier findet eine DOM-Vorverarbeit statt, in der das inhaltsstärkste DOM-Element basierend auf bekannter Wiki-Syntax gefunden und der Rest verworfen wird. Aus diesem Block können dann noch UI-Nodes wie Navigations- und Bearbeitungslinks entfernt werden. Dieser Schritt dient dazu, das Rauschen in den Daten zu reduzieren, und damit die Ergebnisqualität zu erhöhen. Durch die Reduktion von unnötigen Datenflüssen in der Pipeline werden außerdem Speicherressourcen gespart. Das Eingangsformat ist aus anwendungsspezifischen Gründen `.htm`, und das Ausgangsformat aus diesem Verarbeitungsschritt wird `.html` sein, da es einfach bekannter und intuitiver ist. In Abgrenzung zu den anderen Transformationsschritten ist dieser Schritt regelbasiert und deterministisch und sollte deshalb klar vor dem semantischen Parsing liegen.

6.1.3 Strukturelles Parsing

Im zweiten Transformationsschritt findet ein strukturelles Parsing statt. Das Ziel dieses Schritts ist, aus dem bereinigten HTML eine Objektstruktur zu machen. Überschriften werden in Sections umgewandelt, Absätze in Paragraph-Blöcke, Tabellen in Tabellen-Blöcke und so weiter. Das Eingabeformat ist das Ergebnis der DOM-Vorverarbeitung, also `.html`-Blöcke. Das Ausgangsformat sind `.json`-Dateien, die eine Objektrepräsentation mit hoher Informationsdichte sein sollen, die von Steuer- und Anzeigeinformationen bereinigt sind und somit als Eingangsformat in den nächsten Transformationsschritt dienen. Dieser Schritt ist ein struktureller Adapter zwischen bereinigten `.html`-Dateien und interner Objektrepresentation und begründet somit seine Kapselung.

6.1.4 Probabilistische Kontextualisierung

Die probabilistische Kontextualisierung ist der dritte Transformationsschritt, in dem das erste Mal ein LLM in die Pipeline integriert wird. Dieses bekommt die geparsten Objekte aus dem Semantischen Parsing zugeliefert und übergibt sie an ein LLM mit der Anweisung, diese Informationen in einen Fließtext umzuwandeln. Das LLM soll die Datei erfassen, einen Kontext schaffen, indem es die Datei zusammenfasst, selbst Strukturen wie Parameter-/Wert-Paare erkennen, und die in den Kontext einbetten. Der wichtigste Teil dieses Schrittes liegt darin, dass semi- und strukturierte Daten durch das LLM erläutert werden, selbst wenn das bedeutet, dass auf repetetive Weise derselbe Satz immer wieder wiederholt wird. Das Ziel dieses Schritts ist, die Fähigkeit eines LLMs zur Inferenz zu nutzen, um strukturierte Daten zu verstehen und umzuformen und damit bekannte Schwächen von RAG-Systemen im Umgang mit strukturierten Wissensrepräsentationen zu adressieren. Das Schwierige an diesem Schritt ist, den Systemprompt so generisch wie möglich zu halten, damit verschiedenste Eingangsdaten erfasst und weiterverarbeitet werden, aber trotzdem präzise genug, um brauchbare Ergebnisse zu erhalten. Das Eingangsformat ist das Ergebnis des letzten Verarbeitungsschritts und das Ausgangsformat ist `.md`.

6.1.5 RAG-LLM-Integration

Hier ist der *Load*-Schritt der klassischen ETL-Pipeline, nur dass der in diesem Projekt die Bereitstellung der transformierten Daten an die Wissensbasis des RAG-Modells umfasst. Das Ergebnis dieses Schritts ist die Integration in die RAG-Wissensbasis. Hier wird dies, genauso wie der Extraktionsschritt händisch durchgeführt, indem die Ergebnisdateien der probabilistischen Kontextualisierung in eine Wissensbasis in der Open WebUI-Anwendung überführt werden. Zukünftig sollte auch dieser Schritt automatisiert stattfinden, indem die Übergabe an die RAG-Wissensbasis ebenfalls per API-Schnittstelle stattfindet, um Fehlern vorzubeugen.

NEU

6.2 Implementierung

Die Implementierung der LLM-gestützten Vorverarbeitungspipeline wurde in Python realisiert und als eigenständiges Paket mit der Bezeichnung *struct2prose* umgesetzt. Ziel der Implementierung ist es, die in Kapitel 5 beschriebene Vorgehensweise in eine reproduzierbare, modular ausführbare Pipeline zu überführen, ohne dabei unnötige technische Komplexität einzuführen. Der Fokus liegt auf der Nachvollziehbarkeit der einzelnen Verarbeitungsschritte sowie auf der klaren Trennung deterministischer und LLM-basierter Transformationslogik.

6.2.1 Technischer Rahmen und Projektstruktur

Die Pipeline wird über eine Kommandozeilenschnittstelle gesteuert, die den Einstiegspunkt des Systems bildet. Die Ausführung erfolgt über den Befehl `python -m struct2prose`, wobei sowohl einzelne Verarbeitungsschritte als auch ein vollständiger End-to-End-Durchlauf angestoßen werden können. Die einzelnen Schritte sind als Subcommands implementiert und folgen einer linearen Abfolge von der Rohdatenverarbeitung bis zur semantisch kontextualisierten Ausgabe. Die zugehörige Verzeichnisstruktur orientiert sich an den Pipeline-Stufen und umfasst getrennte Ordner für Rohdaten (`raw_data`), bereinigte HTML-Dokumente (`clean_data`), strukturierte Zwischenrepräsentationen (`processed_data`) sowie kontextualisierte Markdown-Dokumente (`normalized_data`).

6.2.2 Interne Datenrepräsentation

Als zentrales internes Datenformat dient eine explizite Objektstruktur, bestehend aus den Klassen `WikiDocument`, `Section` und `ContentBlock`. Diese Repräsentation abstrahiert von der ursprünglichen HTML-Struktur und bildet stattdessen die logische Gliederung der Wiki-Seiten ab. Ein Dokument besteht aus einem Titel, einer geordneten Liste von Abschnitten sowie Metadaten zur Herkunftsdatei. Jeder Abschnitt enthält wiederum eine Folge typisierter Inhaltsblöcke, etwa Absätze, Tabellen, Listen oder Codeblöcke. Diese Struktur stellt sicher, dass strukturierte und unstrukturierte Inhalte bereits vor der LLM-gestützten Verarbeitung explizit unterschieden vorliegen.

6.2.3 Implementierung der Vorverarbeitungsschritte (Step 1–3)

Die Vorverarbeitungsschritte eins bis drei sind vollständig deterministisch implementiert. Im ersten Schritt wird aus den HTML-Exporten der zentrale Inhaltsbereich der Wiki-Seite extrahiert. Hierzu wird heuristisch ein definierter DOM-Knoten als Content-Root ausgewählt, um Navigations- und Metadatenbereiche frühzeitig auszuschließen. Der zweite Schritt entfernt verbliebene UI-Elemente wie Inhaltsverzeichnisse oder Editier-Controls anhand einfacher, regelbasierter Filter. Dieser Schritt ist bewusst minimal gehalten und verfolgt nicht das Ziel eines vollständigen DOM-Cleanings. Im dritten Schritt erfolgt das eigentliche Parsing des bereinigten HTML in ein strukturiertes JSON-Zwischenformat. Überschriften werden dabei als Abschnittsgrenzen interpretiert, während Absätze, Tabellen, Listen und Codeblöcke jeweils als eigenständige Inhaltsblöcke modelliert werden.

6.2.4 LLM-gestützte semantische Kontextualisierung (Step 4)

Die semantische Kontextualisierung strukturierter Inhalte erfolgt im vierten Schritt der Pipeline und stellt den zentralen Einsatzpunkt eines Large Language Models dar. Hierzu wird das Groq-Framework verwendet, wobei der Zugriff über einen API-Key erfolgt, der als Environment-Variable konfiguriert ist. Als Modell kommt `llama-3.3-70b-versatile` mit einer niedrigen Temperatur zum Einsatz, um möglichst konsistente und reproduzierbare Ausgaben zu erzielen. Die Kontextualisierung wird selektiv angewendet: Tabellen und optional Listen werden mittels promptbasierter LLM-Anfragen in beschreibenden Fließtext überführt, während bereits gut lesbare Absätze unverändert übernommen werden. Codeblöcke werden ebenfalls nicht inhaltlich verändert, sondern lediglich als zusammenhängende Codeabschnitte in die Ausgabe integriert.

6.2.5 Generierung der Markdown-Dokumente

Als Ergebnis der Kontextualisierung werden die Dokumente im Markdown-Format ausgegeben. Die Struktur der Ausgabedateien folgt dabei der ursprünglichen Abschnittsgliederung der Wiki-Seiten, ersetzt jedoch tabellarische Darstellungen durch prosebasierte Beschreibungen. Dieses

Format eignet sich besonders für die spätere Integration in Retrieval-Augmented-Generation-Systeme, da es semantisch kohärente Textsegmente erzeugt und gängige Chunking-Strategien unterstützt.

6.2.6 Limitationen der Implementierung

Die Implementierung ist bewusst als Minimalprototyp ausgelegt. Sie verzichtet auf komplexe Heuristiken, tiefgreifende HTML-Kontextualisierung oder umfangreiche Nachbearbeitung der LLM-Ausgaben. Diese Einschränkungen sind bewusst gewählt, um den methodischen Kern der Arbeit, die LLM-gestützte semantische Kontextualisierung strukturierter Inhalte, klar sichtbar zu machen und eine saubere Grundlage für die nachfolgende Evaluation zu schaffen.

Kapitel 7

Evaluation

NEU Um das hier vorgestellte System auf Tauglichkeit zu prüfen, wird im Folgenden zunächst die Zielsetzung der Evaluation definiert, dann die Methoden erläutert und zum Schluss die Ergebnisse vorgestellt.

7.1 Zielsetzung der Evaluation

Zur Evaluation der Pipeline und des daraus resultierenden LLM werden keine Benchmarks verwendet, sondern ein Test auf Funktion durchgeführt. Das Ziel der Evaluation liegt darin, zu zeigen, *dass* die Transformation der Daten die Antwortqualität von Fragen zu dem Inhalt der Wissensbasis verbessern.

7.2 Methode

Für einen Funktionstest wurden zwei Wissensbasen erstellt, eine mit den rohen, unkontextualisierten Daten, und eine mit den Dateien, die den Transformationsprozess mit der LLM-basierten ETL-Pipeline durchlaufen haben. Diese werden in einer selbstgehosteten OpenWeb UI-Anwendung[1] mit einem vortrainierten Large Language Modell verbunden. Dies muss aufgrund der Vergleichbarkeit dasselbe Modell sein und aus Gründen der Verfügbarkeit wurde hier das Modell `gptoss120b` verwendet. Hier soll bewusst kein Modellvergleich stattfinden und auch keine Optimierung.

Es entstehen zwei RAG-Modelle. Das RAG-Modell mit der rohen Wissensbasis trägt hier die Bezeichnung `GPT-Wiki-Raw` und das mit der kontextualisierten Wissensbasis heißt `GPT-Wiki-Normalized`. Beiden Modellen wurden stets dieselben Fragen gestellt. Es wurden bewusst Fragen gestellt, deren Antwort in Tabellen zu finden ist. Da der Fokus dieser Arbeit in der Transformation strukturierter Daten liegt, wurden auch diese getestet.

Folgende Bewertungskriterien werden dabei angewendet:

1. Jeder faktuelle Aspekt der Antwort des LLM ist korrekt und beantwortet die Frage.
2. Es gibt keine halluzinierten Aspekte.
3. Es werden die richtigen Dateien als Quellen angegeben.

Bei der Beschreibung der Ergebnisse im folgenden Unterkapitel werden folgende Punkte dargestellt:

[1] https://github.com/open-webui/open-webui

1. Fragestellung
2. relevante Daten in Rohform (Auszug): hier werden aus Gründen der Lesbarkeit die Daten ohne ihre HTML-Tags und sonstige Metainformationen dargestellt. Die vollständigen Daten sind mit allen Zwischenschritten unter `https://github.com/free-da/struct2prose` zu finden.
3. Antwort des Roh-LLM
4. relevante Daten in transformierter Form (Auszug)
5. Antwort des kontextualisierten LLM
6. Bewertung der Antworten als „korrekt“ oder „inkorrekt“. Eine Antwort wird als „inkorrekt“ bewertet, wenn sie entweder faktisch falsche Aussagen enthält, wesentliche Aspekte der Fragestellung nicht beantwortet oder den Kontext der Daten fehlerhaft interpretiert.

Eine ausführlichere Diskussion der Ergebnisse folgt im folgenden Kapitel 8: Diskussion.

7.3 Ergebnisse

Hier werden drei beispielhafte Ergebnisse von den Test-Abfragen gezeigt und die Bewertung erläutert. Die kompletten Beispiele sind als Screenshots im Anhang zu finden.

7.3.1 Beispiel 1: Datenbankuser

Im ersten Beispiel wurde die Frage gestellt: „Was ist EISTHBStore_dev?“. Die Rohdaten liegen in folgender Form in der Datei `Doc-Gen.htm` vor:

```
Datenbank
  Server      188.455.6.52 (s-0941.db.domain.de)
  Port        1521
  SID         tbv19
  User        EISKARMA_dev
              EISJPAStore_dev
              EISLockRegistry_dev
              EISTBHStore_dev
              EISUserService_dev
```

Die Antwort von `GPT-Wiki-RAW` sagte, dass der nachgefragte Begriff ein „Entwicklungs-Component“ des Enterprise Integration Service (EIS) sei und zusammen mit weiteren EIS-Modulen eingesetzt werde. Als Quelle nannte er die Datei `Doc-Gen.htm`.

Die kontextualisierten Daten lagen in folgender Form vor in der Datei `Doc-Gen.md`:

```
Tabelle: EIS-Server-Konfiguration
Die Tabelle "EIS-Server-Konfiguration" stellt die Konfigurationsdetails für den Doc-Gen
 Integration Server (EIS) 1.8 dar. Sie enthält Informationen über die Datenbankverbindung,
 die für den Betrieb des Servers erforderlich sind.

Die Bedeutung jeder Spalte kann wie folgt beschrieben werden:

Die erste Spalte enthält die Kategorie oder den Typ der Konfigurationseinstellung (z.B.
 Datenbank, Server, Port, SID, User).
Die zweite Spalte enthält den entsprechenden Wert oder die Details zu jeder Kategorie.
Die Tabellenzeilen können wie folgt beschrieben werden:

Der Server für die Datenbank ist 188.455.6.52 (s-0941.db.domain.de).
```

```
Der Port für die Datenbankverbindung ist 1521.
Der SID (System Identifier) für die Datenbank ist tbv19.
Die Benutzer für die Datenbank sind EISKARMA_dev, EISJPAStore_dev, EISLockRegistry_dev,
 EISTBHStore_dev und EISUserService_dev.
Es ist zu beachten, dass die Tabelle keine explizite Überschrift enthält, daher wurde der
 Titel "EIS-Server-Konfiguration" basierend auf dem Inhalt gewählt.
```

Auf dieselbe Frage antwortete `GPT-Wiki-Normalized`, dass der nachgefragte Begriff einer der autorisierten Datenbank-Benutzer im Doc-Gen-Test-/Entwicklungs-Umfeld sei. Er werde verwendet, um sich an der Oracle-Datenbank [...] zu authentifizieren [...]. Damit diene er als spezieller *Dev-User* für Tests und Entwicklungsarbeiten an der Datenbank-Konfiguration. Als Quelle nannte er die Datei `Doc-Gen.md`.

Dieses Beispiel zeigt, dass zwar beide LLMs die richtigen Stellen mit den gesuchten Informationen gefunden haben, aber dem LLM mit den Rohdaten der Kontext gefehlt hat, um die Daten korrekt einzuordnen. Nur das LLM mit den kontextualisierten Daten hat die richtige Antwort genannt. Die kompletten Antworten sind als Screenshots im Anhang unter A.1 zu finden.

7.3.2 Beispiel 2: Modellnummer und Standort

Im zweiten Beispiel wurde die Frage gestellt: „An welchen Standorten ist das Modell „G7500, TC8“ im Einsatz?“. Die benötigten Informationen sind in der Rohform in der Datei `Polycom--Videokonferenzsysteme.htm` zu finden und liegen folgendermaßen vor:

```
Standort/Raum Name  VKA-IP  VKA-MAC SIP Modell  TC-IP TC-MAC  TC-SN VMR-ID
[...]
Standort2 A.1.1 Standort2-A.1.1 188.340.8.92  188.34.:74:b7:b4  10  G7500, TC8
 188.340.8.93  188.34.:80:ec:17  8L230980EC17FD  44410
Standort2 B.0.2.2 Standort2-B.0.2.2 188.340.8.83  188.34.:6a:b9:5e  11  StudioX50, TC8
 188.340.8.91  188.34.:80:e3:82  8L230880E382FD  44411
Standort2 B.0.2.3 Standort2-B.0.2.3 188.340.8.84  188.34.:6a:bf:c1  12  StudioX50, TC8
 188.340.8.85  188.34.:80:ea:07  8L230880EA07FD  44412
Standort2 B.1.1 Standort2-B.1.1 188.340.8.70  188.34.:74:b1:58  13  G7500, TC8
 188.340.8.65  188.34.:80:e3:76  8L230880E376FD  44413
Standort2 B.2.22  Standort2-B.2.22  188.340.8.94  188.34.:69:1d:52  14  StudioX50, TC8
 188.340.8.95  188.34.:80:f0:63  8L230980F063FD  44414
Standort2 B.4.10  Standort2-B.4.10  188.340.8.96  188.34.:53:c4:14  15  StudioX50, TC8
 188.340.8.97  188.34.:80:e2:05  8L230880E205FD  44415
Standort2 B.5.1 Standort2-B.5.1 188.340.8.79  188.34.:6a:b4:99  16  StudioX50, TC8
 188.340.8.60  188.34.:80:ef:4d  8L230980EF4DFD  44416
Standort2 B.5.20  Standort2-B.5.20  188.340.8.80  188.34.:6a:b6:7f  17  StudioX50, TC8
 188.340.8.59  188.34.:80:e2:bc  8L230880E2BCFD  44417
Standort2 B.6.17  Standort2-B.6.17  188.340.8.63  188.34.:74:b1:14  18  G7500, TC8
 188.340.8.58  188.34.:80:ef:e1  8L230980EFE1FD  44418
Standort2 D.1.2 Standort2-D.1.2 188.340.8.71  188.34.:6a:cf:0c  19  StudioX50, TC8
 188.340.8.54  188.34.:80:e6:b2  8L230880E6B2FD  44419
Standort2 D.1.10  Standort2-D.1.10  188.340.8.86  188.34.:6a:cd:f8  20  StudioX50, TC8
 188.340.8.87  188.34.:80:e6:6d  8L230880E66DFD  44420
Standort2 D.2.2 Standort2-D.2.2 188.340.8.57  188.34.:6a:ba:99  21  StudioX50, TC8
 188.340.8.53  188.34.:80:ef:1b  8L230980EF1BFD  44421
[...]
```

Auf die Frage antwortete `GPT-Wiki-RAW` mit der Angabe von zwei korrekten Standorten, es fehlte aber ein dritter.

In der transformierten Form lagen die Daten folgendermaßen vor:

```
Der Standort "Standort2 A.1.1" hat ein Videokonferenzsystem namens "Standort2-A.1.1" mit der IP-Adresse "188.340.8.92" und dem Modell "G7500, TC8".
Der Standort "Standort2 B.0.2.2" hat ein Videokonferenzsystem namens "Standort2-B.0.2.2" mit der IP-Adresse "188.340.8.83" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.0.2.3" hat ein Videokonferenzsystem namens "Standort2-B.0.2.3" mit der IP-Adresse "188.340.8.84" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.1.1" hat ein Videokonferenzsystem namens "Standort2-B.1.1" mit der IP-Adresse "188.340.8.70" und dem Modell "G7500, TC8".
Der Standort "Standort2 B.2.22" hat ein Videokonferenzsystem namens "Standort2-B.2.22" mit der IP-Adresse "188.340.8.94" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.4.10" hat ein Videokonferenzsystem namens "Standort2-B.4.10" mit der IP-Adresse "188.340.8.96" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.5.1" hat ein Videokonferenzsystem namens "Standort2-B.5.1" mit der IP-Adresse "188.340.8.79" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.5.20" hat ein Videokonferenzsystem namens "Standort2-B.5.20" mit der IP-Adresse "188.340.8.80" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 B.6.17" hat ein Videokonferenzsystem namens "Standort2-B.6.17" mit der IP-Adresse "188.340.8.63" und dem Modell "G7500, TC8".
Der Standort "Standort2 D.1.2" hat ein Videokonferenzsystem namens "Standort2-D.1.2" mit der IP-Adresse "188.340.8.71" und dem Modell "StudioX50, TC8".
Der Standort "Standort2 D.1.10" hat ein Videokonferenzsystem namens "Standort2-D.1.10" mit der IP-Adresse "188.340.8.86" und dem Modell "StudioX50, TC8".
```

Die Antwort von `GPT-Wiki-Normalized` identifizierte korrekt die drei Standorte A.1.1, B.1.1 sowie B.6.17 am Standort2. Damit erfüllt die Antwort des kontextualisierten LLM die definierten Bewertungskriterien, während die Antwort des Roh-LLM diese nicht vollständig erfüllt. Wieder haben beide LLMs die richtigen Dateien als Quellen genannt. Die kompletten Antworten sind den Screenshots im Anhang A.2 zu entnehmen.

7.3.3 Beispiel 3: Frage nach IP-Adresse

Im dritten Beispiel lautete die Fragestellung: „Was ist 188.340.8.90“[2]? Die benötigten Informationen lagen in derselben Tabelle vor wie im obigen Beispiel in 7.3.2 gezeigt und soll aus Gründen der Datensparsamkeit nicht wiederholt werden.

Die Antwort von `GPT-Wiki-RAW` sagte aus, dass es keine Informationen über die IP-Adresse `188.340.8.90` finden könne und somit auch keine Angaben zu Geräten oder Servicen hinter dieser Adresse geben könne. Im Gegensatz dazu gab `GPT-Wiki-Normalized` an, dass die gesuchte IP-Adresse einem Videokonferenzsystem zugeordnet sei, was am Standort 2 in Raum D.2.11 installiert sei. Das Gerät trage den Namen Standort2-D.2.11 und verwende das Modell StudioX50, TC80. Die vom kontextualisierten LLM gelieferten Aussagen entsprechen alle dem, was die zugrunde liegenden Informationen hergaben und werden somit als korrekt bezeichnet. Dieses Beispiel verdeutlicht insbesondere die Fähigkeit des kontextualisierten Modells, implizite Zusammenhänge aus prosebasierten Repräsentationen korrekt zu rekonstruieren, selbst wenn die ursprünglichen Rohdaten stark verdichtet vorliegen. Auch hierfür sind die entsprechenden Screenshots im Anhang A.3 zu finden.

[2]Die IP-Adressen wurden wie andere sensible Daten anonymisiert, wodurch nicht-zulässige Zahlen zustande kamen. Dass dies Auswirkungen auf die Identifizierung einer IP-Adresse als solche haben könnte, wurde zu dem Zeitpunkt bedauerlicherweise nicht beachtet. In den späteren Tests schien es aber keine negativen Auswirkungen zu haben.

Kapitel 8

Diskussion

NEU Hier sollen die Ergebnis nun in das Gesamtprojekt eingebettet werden. Zunächst werden die Ergebnisse eingeordnet und die Beobachtungen erläutert. Dann wird die Forschungsfrage beantwortet. Danach werden die Grenzen dieses Versuchs erläutert und zum Schluss darauf eingegangen, was dieser Versuch für RAG-Systeme bedeutet.

8.1 Einordnung der Ergebnisse

Die im vorhergehenden Kapitel dargestellten Ergebnisse zeigen, dass eine vorgelagerte Kontextualisierung strukturierter Daten die Antwortqualität eines RAG-Systems unter bestimmten Bedingungen erhöhen kann. Insbesondere dann, wenn Informationen in stark strukturierter oder tabellarischer Form vorliegen und die RAG-Anwendung selbst keine gezielte Einflussnahme auf Chunking oder Indexierungsstrategien erlaubt, wirkt sich die semantische Kontextualisierung positiv auf die Nutzbarkeit der Inhalte aus. Dieser Effekt tritt vor allem bei umfangreicheren Tabellen auf, in denen relevante Informationen über mehrere Zeilen oder Spalten verteilt sind.

Gleichzeitig wurde beobachtet, dass die Antwortqualität des RAG-Systems auf Basis roher Daten in vielen Fällen mit der des Systems auf Basis kontextualisierter Daten vergleichbar ist. Dies verdeutlicht, dass der entscheidende Faktor nicht allein in der Unterscheidung zwischen strukturierter Darstellung und Fließtext liegt. Vielmehr ist ausschlaggebend, wie nah die zur Beantwortung einer Frage relevanten Kontextinformationen zueinander positioniert sind. Auch strukturierte Daten können somit für RAG-Systeme gut nutzbar sein, sofern zusammengehörige Parameter- und Wert-Informationen räumlich nahe beieinanderliegen und gemeinsam in einem Chunk repräsentiert werden.

Die Ergebnisse legen nahe, dass weniger die Datenstruktur an sich, sondern vielmehr das zugrunde liegende Chunking-Verhalten eine zentrale Rolle für die Antwortqualität spielt. In dem hier betrachteten Versuchsaufbau wurde das Chunking bewusst nicht verändert, sodass Unterschiede primär auf die veränderte Repräsentation der Inhalte zurückzuführen sind. Insbesondere bei Tabellen mit geringer Spaltenanzahl und einer zeilenweisen Anordnung von Parameter-Wert-Paaren zeigte sich, dass auch rohe Daten für RAG-Systeme ausreichend kontextuell erschlossen werden können.

Vor diesem Hintergrund erscheinen neben der hier eingesetzten LLM-basierten Kontextualisierung auch andere Formen der Vorverarbeitung denkbar, etwa durch die gezielte Wiederholung von Tabellenköpfen unter Beibehaltung der strukturierten Form. Die vorliegenden Ergebnisse unterstreichen somit, dass die semantische Kontextualisierung einen wirksamen Ansatz darstellt, insbesondere dann, wenn der Kontextverlust durch automatische Segmentierung nicht anderweitig kompensiert werden kann.

8.2 Beantwortung der Forschungsfrage

In dieser Arbeit wurde untersucht, inwiefern eine LLM-gestützte semantische Kontextualisierung strukturierter Wiki-Inhalte zur verbesserten Nutzbarkeit dieser Inhalte in RAG-Systemen beitragen kann. Zu diesem Zweck wurde eine ETL-ähnliche Vorverarbeitungspipeline entwickelt und exemplarisch auf ausgewählte Wiki-Dokumente angewendet, deren Inhalte anschließend im Rahmen einer qualitativen funktionalen Evaluation untersucht wurden.

Die in Kapitel 7 dargestellten Ergebnisse zeigen, dass durch die semantische Kontextualisierung eine verbesserte Nutzbarkeit der Wissensbasis erreicht werden kann. Insbesondere wurde deutlich, dass ein RAG-System auf Basis kontextualisierter Inhalte in der Lage ist, Fragen zu beantworten, deren relevante Informationen in umfangreichen tabellarischen Strukturen vorliegen. Durch die Transformation werden zusammengehörige Parameter- und Wert-Informationen explizit miteinander verknüpft und in prosebasierter Form dargestellt, wodurch sie für das RAG-System leichter auffindbar und interpretierbar werden.

Die erhöhte Antwortqualität lässt sich vor allem auf die Reduktion von Kontextverlusten zurückführen, die bei der Verarbeitung strukturierter Daten im Rahmen des Chunkings auftreten. Durch die Segmentierung der Rohdaten gehen Metainformationen wie Tabellenköpfe oder implizite Beziehungen zwischen Einträgen in einzelnen Chunks verloren. Das RAG-LLM kann in solchen Fällen zwar relevante Suchbegriffe identifizieren, diese jedoch nicht zuverlässig in ihren fachlichen Kontext einordnen. Die semantische Kontextualisierung adressiert dieses Problem, indem sie implizite Beziehungen explizit macht und dem Modell ermöglicht, gefundene Informationen korrekt zu interpretieren und in Beziehung zu setzen.

Die Aussagekraft dieser Ergebnisse ist jedoch kontextabhängig. Die Qualität der Kontextualisierung hängt maßgeblich von der Fähigkeit des eingesetzten LLM ab, strukturierte Inhalte korrekt zu erfassen und in eine geeignete textuelle Repräsentation zu überführen. Darüber hinaus lassen die vorliegenden Ergebnisse keine allgemeingültigen Aussagen über andere Domänen, Datenformate oder RAG-Konfigurationen zu, sondern beziehen sich auf den hier untersuchten Anwendungsfall.

8.3 Limitationen der Evaluation

Die in dieser Arbeit durchgeführte Evaluation ist in ihrer Aussagekraft bewusst begrenzt. Es handelt sich nicht um eine quantitative oder benchmarkbasierte Untersuchung, sondern um einen funktionalen Test, der zeigen soll, ob die vorgestellte Vorverarbeitung unter konkreten Bedingungen zu einer verbesserten Nutzbarkeit der Inhalte in einem RAG-System führen kann.

Die Evaluation basiert auf einer kleinen, manuell ausgewählten Menge von Wiki-Seiten sowie auf einer begrenzten Anzahl von Testfragen. Die Fragestellungen wurden gezielt so gewählt, dass sie sich auf tabellarische und semistrukturierte Inhalte beziehen. Eine systematische Abdeckung unterschiedlicher Fragetypen oder Datenformate fand nicht statt. Daraus ergibt sich, dass die Ergebnisse nicht ohne Weiteres auf andere Kontexte oder Anwendungsfälle übertragbar sind.

Weiterhin wurde in dem Versuchsaufbau nur ein einziges vortrainiertes Large Language Modell verwendet. Ein Vergleich unterschiedlicher Modelle oder RAG-Konfigurationen war nicht Teil dieser Arbeit. Auch Aspekte wie Chunking-Strategien, Indexierungsparameter oder Retrieval-Methoden wurden bewusst nicht variiert, um den Einfluss der semantischen Kontextualisierung isoliert betrachten zu können.

Die Qualität der Kontextualisierung hängt zudem direkt von der Leistungsfähigkeit des eingesetzten LLM ab. Fehler bei der Interpretation strukturierter Daten oder unvollständige Umformulierungen können sich unmittelbar auf die Qualität der Wissensbasis auswirken. Typische

Risiken beim Einsatz von LLMs, wie etwa Halluzinationen, können daher nicht vollständig ausgeschlossen werden.

Die vorliegenden Ergebnisse sind daher als indikativ zu verstehen. Sie zeigen, dass der gewählte Ansatz unter den gegebenen Rahmenbedingungen funktioniert, treffen jedoch keine allgemeingültigen Aussagen über die Leistungsfähigkeit von RAG-Systemen im Allgemeinen.

8.4 Implikationen für RAG-Systeme

Die Ergebnisse dieser Arbeit zeigen, dass die Nutzbarkeit einer Wissensbasis für RAG-Systeme nicht allein von der Leistungsfähigkeit des eingesetzten Large Language Modells abhängt, sondern in erheblichem Maße von der Aufbereitung der zugrunde liegenden Daten. Insbesondere strukturierte und tabellarische Inhalte stellen für RAG-Systeme eine Herausforderung dar, wenn implizite Beziehungen zwischen Einträgen nicht explizit repräsentiert sind.

Die vorgestellte LLM-gestützte semantische Kontextualisierung bietet einen Ansatz, um diese Problematik bereits vor der eigentlichen Nutzung im RAG-System zu adressieren. Durch die Umwandlung strukturierter Inhalte in prosebasierte Wissenseinheiten können Kontextverluste, die durch *Chunking* und *Retrieval* entstehen, reduziert werden. Dies ist insbesondere in Szenarien relevant, in denen die Konfiguration der RAG-Pipeline nur eingeschränkt angepasst werden kann oder heterogene Datenquellen integriert werden müssen.

Darüber hinaus verdeutlicht der Ansatz, dass Large Language Modelle nicht nur für die Beantwortung von Fragen, sondern auch für die vorgelagerte Wissensaufbereitung sinnvoll eingesetzt werden können. Die Trennung zwischen deterministischer Vorverarbeitung und probabilistischer, LLM-basierter Kontextualisierung erweist sich dabei als praktikabler Kompromiss zwischen Kontrollierbarkeit und semantischer Flexibilität.

Für den praktischen Einsatz von RAG-Systemen legt diese Arbeit nahe, der Datenvorverarbeitung eine stärkere Bedeutung beizumessen. Insbesondere bei wissensintensiven Anwendungen mit historisch gewachsenen, strukturierten Dokumenten kann eine semantische Kontextualisierung einen wesentlichen Beitrag zur stabilen und nachvollziehbaren Nutzung der Inhalte leisten.

Kapitel 9

Fazit und Ausblick

NEU

Ziel dieser Arbeit war es, ein Vorgehensmodell zur LLM-gestützten Vorverarbeitung strukturierter Wiki-Inhalte zu entwickeln und zu untersuchen, inwiefern eine solche semantische Kontextualisierung zur verbesserten Nutzbarkeit dieser Inhalte in RAG-Systemen beitragen kann. Im Mittelpunkt stand dabei nicht der Vergleich unterschiedlicher Sprachmodelle, sondern die Frage, welchen Einfluss die Repräsentation der Wissensbasis auf die Antwortqualität eines RAG-Systems hat.

Die Ergebnisse der durchgeführten Evaluation zeigen, dass eine semantische Kontextualisierung strukturierter und semistrukturierter Inhalte die Nutzbarkeit der Wissensbasis insbesondere bei tabellarischen Daten erhöhen kann. Durch die Umwandlung impliziter Strukturen in explizite, prosebasierte Wissenseinheiten wird der Kontextverlust reduziert, der bei der Segmentierung und dem Retrieval von Rohdaten auftreten kann. Dadurch ist das RAG-System besser in der Lage, relevante Informationen korrekt zu identifizieren und einzuordnen.

Gleichzeitig wurde deutlich, dass strukturierte Daten nicht grundsätzlich ungeeignet für den Einsatz in RAG-Systemen sind. Vielmehr hängt ihre Nutzbarkeit stark davon ab, wie nah zusammengehörige Informationen in der Wissensbasis positioniert sind und wie das zugrunde liegende Chunking-Verfahren arbeitet. Die vorgestellte Kontextualisierung stellt daher keinen universellen Ersatz für andere Optimierungsmaßnahmen dar, sondern ergänzt bestehende Ansätze zur Aufbereitung von Wissensbasen.

Der Ausblick dieser Arbeit betrifft sowohl methodische als auch praktische Aspekte. Eine weiterführende Untersuchung könnte die vorgestellte Pipeline auf größere und heterogenere Datenbestände anwenden und unterschiedliche Chunking- und Retrieval-Strategien systematisch vergleichen. Auch ein Vergleich verschiedener Large Language Modelle im Kontextualisierungsschritt sowie eine quantitative Evaluation mit Nutzerstudien oder standardisierten Metriken wären denkbar.

Darüber hinaus eröffnet der Ansatz Perspektiven für den praktischen Einsatz in wissensintensiven Organisationen, in denen strukturierte Dokumente häufig historisch gewachsen sind und nicht ohne Weiteres angepasst werden können. In solchen Kontexten kann eine LLM-gestützte semantische Kontextualisierung einen pragmatischen Beitrag zur verbesserten Nutzung vorhandenen Wissens in RAG-Systemen leisten.

Anhang A

Anhang

A.1 Evaluation: Beispiel 1

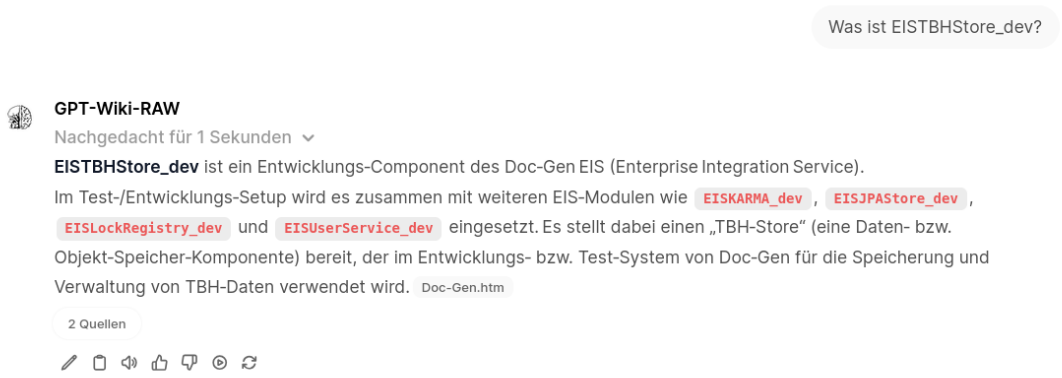

Abbildung A.1: Frage zu Datenbankusern – roh

Abbildung A.2: Frage zu Datenbankusern – normalisiert

A.2 Evaluation: Beispiel 2

Abbildung A.3: Frage zu Standort und Modell – roh

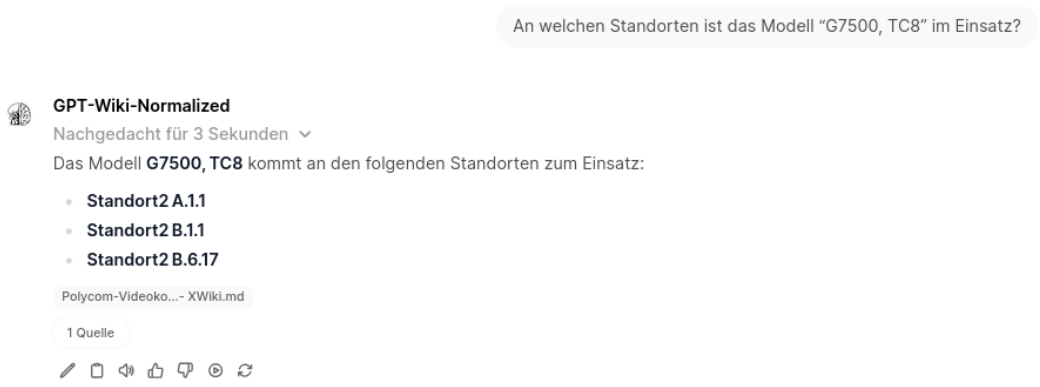

Abbildung A.4: Frage zu Standort und Modell – normalisiert

A.3 Evaluation: Beispiel 3

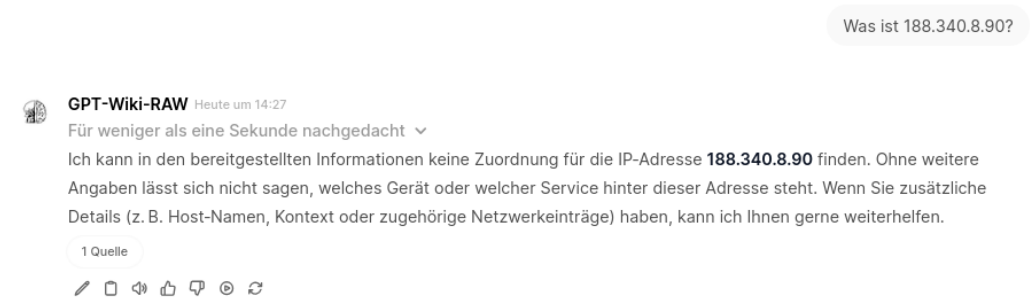

Abbildung A.5: Frage zu Standort und Modell – roh

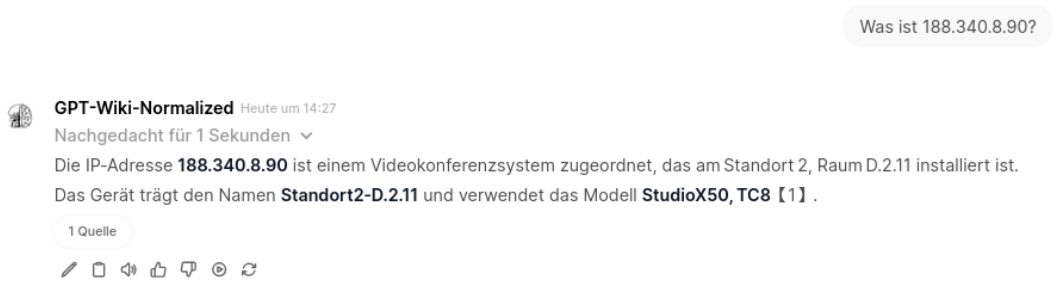

Abbildung A.6: Frage zu IP-Adresse – normalisiert

Literatur

Asai, Akari u. a. [17. Okt. 2023]. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. DOI: 10.48550/arXiv.2310.11511. arXiv: 2310.11511[cs]. URL: http://arxiv.org/abs/2310.11511 [besucht am 03. 11. 2025].

Bommasani, Rishi u. a. [12. Juli 2022]. *On the Opportunities and Risks of Foundation Models*. DOI: 10.48550/arXiv.2108.07258. arXiv: 2108.07258[cs]. URL: http://arxiv.org/abs/2108.07258 [besucht am 09. 10. 2025].

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*. DOI: 10.48550/arXiv.2312.10997. arXiv: 2312.10997[cs]. URL: http://arxiv.org/abs/2312.10997 [besucht am 03. 11. 2025].

Jiang, Zhengbao u. a. [22. Okt. 2023]. *Active Retrieval Augmented Generation*. DOI: 10.48550/arXiv.2305.06983. arXiv: 2305.06983[cs]. URL: http://arxiv.org/abs/2305.06983 [besucht am 03. 11. 2025].

Lewis, Patrick u. a. [12. Apr. 2021]. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. DOI: 10.48550/arXiv.2005.11401. arXiv: 2005.11401[cs]. URL: http://arxiv.org/abs/2005.11401 [besucht am 09. 10. 2025].

Si, Jacob u. a. [10. Nov. 2025]. *TabRAG: Tabular Document Retrieval via Structured Language Representations*. DOI: 10.48550/arXiv.2511.06582. arXiv: 2511.06582[cs]. URL: http://arxiv.org/abs/2511.06582 [besucht am 26. 01. 2026].

Sui, Yuan u. a. [22. Mai 2023]. *Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study*. arXiv.org. URL: https://arxiv.org/abs/2305.13062v5 [besucht am 25. 11. 2025].

Yu, Xiaohan, Pu Jian und Chong Chen [30. Sep. 2025]. *TableRAG: A Retrieval Augmented Generation Framework for Heterogeneous Document Reasoning*. DOI: 10.48550/arXiv.2506.10380. arXiv: 2506.10380[cs]. URL: http://arxiv.org/abs/2506.10380 [besucht am 16. 01. 2026].

Zhang, Yue u. a. [14. Sep. 2025]. *Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models*. DOI: 10.48550/arXiv.2309.01219. arXiv: 2309.01219[cs]. URL: http://arxiv.org/abs/2309.01219 [besucht am 03. 11. 2025].