HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Einsatz eines Retrieval-Augmented Generation Systems für die Verarbeitung von Produktdaten aus relationalen Datenbanken im Kontext eines digitalen Einkaufsassistenten

Bachelorarbeit

Im Studiengang Wirtschaftsingenieurwesen Agrar/Lebensmittel

an der Fakultät Agrarwissenschaften und Landschaftsarchitektur

vorgelegt von:	Fabian Stöppel
Matrikel-Nr.:	951950
Ausgabedatum:	23.09.2024
Abgabedatum:	02.01.2025
Erstprüfer:	Prof. Dr. Nicolas Meseth
Zweitprüfer:	Philipp Zmijewski

Inhaltsverzeichnis

Abkürzungsverzeichnis

DBMS ..... Datenbankmanagementsysteme
LLM ..... Large Language Model
NLP ..... Natural Language Processing
RAG ..... Retrieval-Augmented Generation
RDB ..... Relationale Datenbank
SQL ..... Structured Query Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Entwicklung moderner Kommunikationssysteme wird durch eine fortschreitende Digitalisierung und insbesondere von dem Einsatz generativer Sprachmodelle geprägt (PERUCHINI und TEIXEIRA 2024, o. S.). Einer der Anwendungsbereiche dieser Sprachmodelle, allgemein auch bekannt als Large Language Model (LLM), ist das „Question Answering“ (QA), welches in den letzten Jahren in Form von „Chatbots“ an Bedeutung gewonnen hat (SHARMA et al. 2024, o. S.). Generative Sprachmodelle ermöglichen eine Verarbeitung von Benutzereingaben in natürlicher Sprache und tragen dadurch wesentlich dazu bei, dass verschiedene Aufgaben wie Inhalte zusammenzufassen, zu klassifizieren oder zu kategorisieren effektiver gelöst werden können (COLLOBERT et al. 2011; MINAEE et al. 2024).

Trotz des großen Potenzials sehen sich die LLMs mit verschiedenen Herausforderungen konfrontiert, beispielsweise das „Halluzinieren“ oder die Abhängigkeit von den Trainingsdaten (HANDSCHUH 2024, S. 24 ff.). Um eben genau diese Herausforderungen bewältigen zu können, wurden Retrieval-Augmented Generation (RAG) Systeme entwickelt. RAG-Systeme kombinieren die Funktionen von generativen Sprachmodellen mit der Einbindung von externem Wissen (GAO et al. 2024a, S. 1; LIU et al. 2023, o. S.; SHUSTER et al. 2021, o. S.; ZHAO et al. 2024, S. 1 f.).

Diese Systeme werden in der Praxis in den unterschiedlichsten Bereichen angewendet. Dazu zählen der Finanzbereich (SETTY et al. 2024), das Gesundheitswesen (AL GHADBAN et al. 2023) sowie der Bereich der Bildung (MODRAN et al. 2024). Ein weiteres Anwendungsgebiet ist der Kundenservice, welcher zur Beantwortung produktspezifischer Fragen genutzt wird (GUPTA et al. 2024, o. S.). Dieser Bereich betrifft auch das aktuelle Forschungsprojekt der Hochschule Osnabrück. Das Projekt „InVerBio - Innovative Verkaufsstrategien für regionale Bio-Lebensmittel. Potenziale automatisierter Einkaufsprozesse in städtischen und ländlichen Räumen.“, zielt darauf ab, einen autonomen Einkaufsprozess im Rahmen eines Fallbeispiels, dem „RegioStore“, zu realisieren. Ein Schwerpunkt dieses Projekts ist die Entwicklung eines digitalen Einkaufsassistenten, der ein vollständig personalfreies Einkaufserlebnis ermöglichen soll. Der Assistent wird verschiedene Aufgaben übernehmen, wobei die Beantwortung von

Kundenfragen zu Produkten eine davon sein wird. Die Nutzung eines RAG-Systems kann klare Vorteile bieten, da es auf spezifische Informationen zugreifen und präzise Antworten generieren kann (GAO et al. 2024a, S. 1).

Eine Herausforderung dieses Projekts liegt in der Implementierung eines RAG-Systems auf der Basis strukturierter Daten aus einer relationalen Datenbank. Dieser Ansatz unterscheidet sich von der bislang vorwiegenden Nutzung unstrukturierter Textdaten in den Systemen (YANG et al. 2024, S. 64). Daraus ergibt sich die zentrale Forschungsfrage: „Wie lässt sich ein Retrieval-Augmented Generation System auf strukturierte Daten aus einer relationalen Datenbank anwenden und wie beeinflussen unterschiedliche Kundenfragen die Qualität des Retrievals und der generierten Antworten?“. Diese Fragestellung wird im Verlauf der Arbeit untersucht, wobei insbesondere analysiert wird, inwieweit eine grundlegende Version eines RAG-Systems in der Lage ist, die relevanten Dokumente zu potenziellen Kundenfragen zu identifizieren und die Fragen inhaltlich korrekt zu beantworten.

Das Ziel dieser Arbeit ist die Entwicklung eines RAG-Systems, welches strukturierte Daten verarbeiten kann. Um das System evaluieren zu können, wird ein Fragenkatalog mit 45 unterschiedlichen Fragen erstellt, die verschiedene Szenarien und Intentionen wie beispielsweise Synonyme oder Rechtschreibfehler abdecken. Um das System umfassend beurteilen zu können, werden unterschiedlich komplexe Fragen berücksichtigt.

Der Aufbau der Arbeit gliedert sich wie folgt: Nach einem einleitenden Kapitel werden im nachfolgenden Kapitel die theoretischen Grundlagen zu relationalen Datenbanken dargelegt. Daran anknüpfend wird im dritten Kapitel der inhaltliche Theorierahmen zu den generativen Sprachmodellen erläutert. In Kapitel vier wird Retrieval-Augmented Generation und dessen Methoden und Kriterien zur Bewertung beleuchtet. In diesen drei Kapiteln wird das grundlegende Verständnis dieser Technologien vermittelt, um in Kapitel fünf den Versuchsaufbau verstehen zu können. Dieses Kapitel beschreibt die technische Umsetzung des Systems, die Erstellung des Fragenkatalogs und die

Bewertung. Anschließend werden die Ergebnisse der Evaluierung präsentiert und im Kontext der Forschungsfrage diskutiert. Abschließend wird ein Ausblick auf mögliche Weiterentwicklungen und Optimierungen gegeben, bevor am Ende eine Zusammenfassung der Arbeit folgt.

2 Relationale Datenbanken

In diesem Kapitel wird die grundlegende Struktur und Funktionsweise relationaler Datenbanken erläutert, um die Herausforderungen der Datenstruktur, welche als domänenspezifische Informationen für das RAG dienen, zu verdeutlichen.

Datenbanken sind essenzielle Komponenten für die Speicherung und Verwaltung von Informationen in zahlreichen modernen Anwendungen. PIEPMEYER (2011, S. 3) definiert den Begriff Datenbank (DB) wie folgt: „Eine Datenbank ist eine Sammlung von Daten, die von einem Datenbankmanagementsystem (DBMS) verwaltet wird.“ Die in einer DB gespeicherten Informationen können in drei zentrale Formen unterteilt werden: strukturierte, unstrukturierte und halbstrukturierte Daten (C. et al. 2015, S. 41). Strukturierte Daten, wie beispielsweise Transaktionsdaten, folgen einem vorgegebenen Schema und werden häufig in tabellarischer Form gespeichert. Unstrukturierte Daten hingegen können in beliebigen Formaten, wie Audiodateien, Videos oder Texte, auftreten. Halbstrukturierte Daten, wie Lebensläufe oder Produktrezensionen, weisen Merkmale beider Kategorien auf und können nicht eindeutig zugeordnet werden.

Die Verwaltung von Datenbanken erfolgt durch Datenbankmanagementsysteme (DBMS), die grundlegende Funktionen wie das Einfügen, Löschen, Ändern und Abrufen von Daten bereitstellen (PIEPMEYER 2011, S. 2). Die Datenorganisation hängt dabei von der gewählten Datenmodellierung ab, die verschiedene Ansätze wie hierarchische, objektorientierte oder relationale Modelle umfassen kann (UNTERSTEIN und MATTHIESSEN 2012, S. 9). Die in dieser Arbeit verwendeten Informationen werden in der relationalen Datenmodellierung vorliegen. CODD (1970) stellte das relationale Datenbankmodell erstmals vor. Die Untersuchungen die darin beschrieben worden sind,

führten zu mathematischen Theorien die Grundlage heutiger Datenbanksysteme sind (SUMATHI und ESAKKIRAJAN 2007, S. 5 f.).

Relationale Datenbanken (RDB) speichern Informationen in einer strukturierten, tabellarischen Form (SUMATHI und ESAKKIRAJAN 2007, S. 67 ff.). Dadurch wird die Grundlage für die Nutzung der standardisierten Sprache, Structured Query Language (SQL) gebildet, die eine effiziente Abfrage, Einfügung, Aktualisierung und Löschung von Daten ermöglicht (SUMATHI und ESAKKIRAJAN 2007, S. 111). SQL dient als Schnittstelle zwischen der RDB und dem Anwendungsprogramm (UNTERSTEIN und MATTHIESSEN 2012, S. 35).

SCHICKER (2017) beschreibt formale Begriffe im Kontext relationaler Datenbanken. Die Begriffe lauten u.a. Relation, Tupel, Attribut, Kardinalität, Grad und Primärschlüssel. Diese werden in Abbildung 1 an einem Beispiel veranschaulicht. Eine Relation wird informell als Tabelle bezeichnet, ein Tupel entspricht einer Zeile der Tabelle und ein Attribut ist eine Spalte. Die Kardinalität gibt die Anzahl der Zeilen einer Tabelle an, während der Grad die Anzahl der Spalten angibt. Ein Primärschlüssel ist ein eindeutiger Identifikator, dessen Funktionsweise im weiteren Verlauf der Arbeit detailliert erläutert wird (SCHICKER 2017, S. 26).

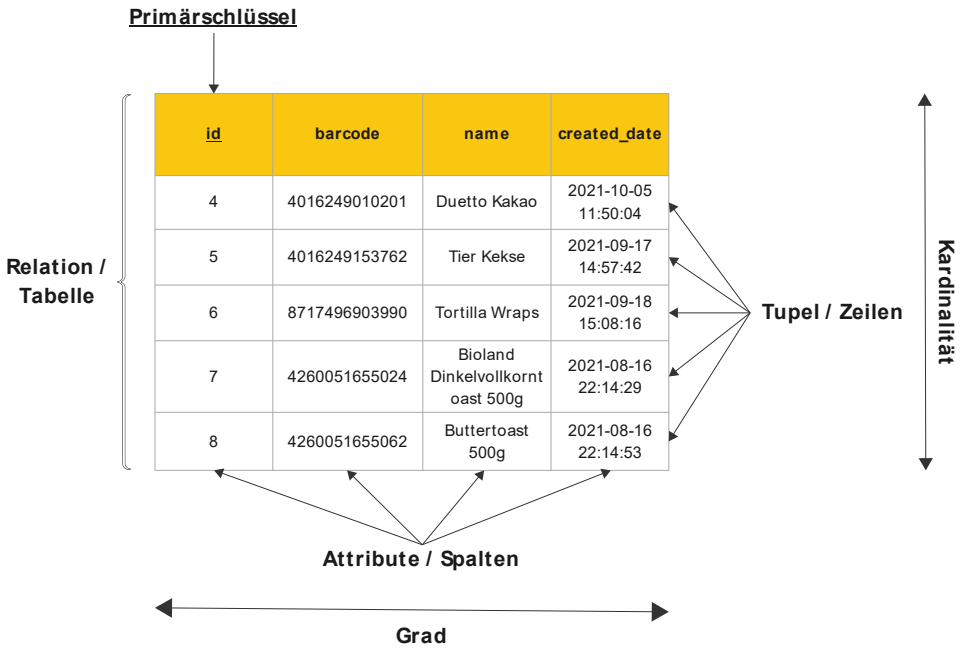


Abbildung 1: Begriffe einer RDB (Modifikation: Mit Werten aus der in der Arbeit verwendeten Tabelle aktualisiert), Quelle: eigene Darstellung nach (SCHICKER 2017, S. 26)

SCHICKER (2017, S. 29) definiert in seinem Werk „Datenbanken und SQL“ den Begriff relationale Datenbank wie folgt: „Eine relationale Datenbank ist demnach eine Ansammlung von Relationen, die zueinander in Beziehung stehen, und die von einem Verwaltungssystem verwaltet werden.“. UNTERSTEIN und MATTHIESSEN (2012) bestätigen diese Definition durch die Verwendung gleicher Begrifflichkeiten wie Relation, Tupel und Attribut sowie weitere Fachbegriffe.

Die genannten Begrifflichkeiten werden im Folgenden anhand der in dieser Arbeit verwendeten Relation „product“ exemplarisch erläutert. Die Tupel repräsentieren einzelne Einträge der Produkte, die spezifische Attribute wie „id“, „identifier“, „barcode“, „created_date“, „description“ und weitere enthalten (siehe Anhang 1c). Den Attributen werden bei der Erstellung der Relation bestimmte Datentypen, sogenannte Domänen zugeordnet (SCHICKER 2017, S. 145 f.; UNTERSTEIN und MATTHIESSEN 2012, S. 21). Beispielsweise erhält das Attribut „id“ den Datentyp „INT“ (Kurzform von „INTEGER“), der ganze Zahlen speichert. Das Attribut „identifier“ wird mit dem Datentyp „VARCHAR“ (Kurzform von „CHARACTER VARYING“) definiert, welcher

Zeichenketten variabler Länge bis zu einer festgelegten maximalen Anzahl speichern kann. Für das Attribut „description“ wird der Datentyp „TEXT“ festgelegt, der ähnlich wie „VARCHAR“ Zeichen nur mit deutlich größeren Längen und einer maximalen Kapazität von 65.535 Zeichen speichern kann (MYSQL 2024a). „FLOAT“ wird für die Speicherung von Gleitkommazahlen in Dezimaldarstellung und „DATETIME“ für die Speicherung von Datum- und Zeitinformationen im Format JJJJ-MM-TT HH:MM:SS verwendet. Zusätzlich können Datentypen implizite Standardwerte erhalten, wie „DEFAULT NULL“ oder „NOT NULL“, um sicherzustellen, dass entweder ein Wert verpflichtend ist oder alternativ ein Nullwert für fehlende Informationen verwendet wird (MYSQL 2024b). Der Nullwert dient dabei zur Darstellung eines unbekannten oder nicht vorhandenen Status, wodurch gewährleistet wird, dass Attribute die eindeutige Werte erfordern korrekt validiert werden (IBM 2021).

Die korrekte Speicherung der Daten ist für die Funktionsweise der relationalen Datenbanken von entscheidender Bedeutung (SCHICKER 2017, S. 34). Deshalb werden die Entitäts- und Referenz-Integritätsregel für die Sicherstellung der Datenkonsistenz eingesetzt, wie von UNTERSTEIN und MATTHIESSEN (2012, S. 30 ff.) und SCHICKER (2017, S. 36 ff.) beschrieben. Die Entitäts-Integritätsregel befasst sich mit dem Primärschlüssel, der jedes Tupel eindeutig identifizieren kann. Dementsprechend muss jedes Tupel über einen einzigartigen Primärschlüssel verfügen, der keine Nullwerte enthalten darf (SCHICKER 2017, S. 36). Somit ist diese Art von Schlüssel in jeder Relation ein eindeutiges Attribut oder eine eindeutige Kombination von Attributen und kann nicht dupliziert werden (UNTERSTEIN und MATTHIESSEN 2012, S. 30). Die Referenz-Integritätsregel bezieht sich auf den Fremdschlüssel, der entweder ein einzelnes Attribut oder eine Gruppe von Attributen ist, die auf den Primärschlüssel einer anderen Relation verweist. Dieser Fremdschlüssel muss mit dem entsprechenden Primärschlüssel übereinstimmen, wodurch eine Beziehung zwischen den Relationen hergestellt wird. Auf diese Weise wird sichergestellt, dass die Daten auch über mehrere Relationen hinweg konsistent bleiben (UNTERSTEIN und MATTHIESSEN 2012, S. 30 ff.).

Relationale Datenbanken bieten aufgrund ihrer strukturierten und skalierbaren Datenorganisation einen erheblichen Vorteil in der effizienten Verwaltung und Abfrage von

Daten. Aus diesem Grund werden diese Systeme von vielen Unternehmen genutzt, wie auch von dem RegioStore.

3 Generative Sprachmodelle

Large Language Models stellen einen zentralen Fortschritt in der Verarbeitung natürlicher Sprache dar und zeichnen sich durch ihre Fähigkeit aus, komplexe Sprachmuster zu erkennen und zu generieren (MINAEE et al. 2024, o. S.). Die Forschung von HUANG et al. (2024) verdeutlicht, wie LLMs durch die Nutzung von Transformer-Architekturen eine hohe Qualität in der Wissensverarbeitung im Bereich für maschinelles Lernen erreichen. Diese Modelle sind in der Lage Muster und Zusammenhänge aus großen Textmengen zu verarbeiten. Die Erkenntnisse werden für das Bewältigen von unterschiedlichen Aufgaben genutzt.

Um ein Grundverständnis von den Aufgaben und der Entwicklung der Sprachmodelle zu erlangen, folgt eine Visualisierung und Erläuterung dieser Modelle (siehe Abbildung 2). Die ersten Sprachmodelle, sogenannte Statistical Language Models (SLM) basieren auf statistischen Methoden, die Wahrscheinlichkeitsberechnungen zur Erkennung und Vorhersage von Sprachmustern nutzen (MINAEE et al. 2024, o. S.; ZHAO et al. 2023, S. 1 ff.). Mit diesen Modellen können grundlegende sprachliche Aufgaben bewältigt werden. Neural Language Models (NLM) basieren auf neuronalen Netzwerken und verbessern die Leistung bei NLP-Aufgaben, wie beispielsweise der Textzusammenfassung, dem Chunking oder der Named Entity Recognition (COLLOBERT et al. 2011, S. 2). Ein weiterer Fortschritt wurde mit der Entwicklung des Pre-trained Language Models (PLM) gemacht. Diese Modelle, wie BERT und die ersten Versionen von GPT (GPT-1 und GPT-2), sind mit großen Textmengen trainiert worden und können somit vielfältige NLP-Aufgaben effektiver lösen. Der derzeitige Stand der Technik wird von den Large Language Models repräsentiert, die mit noch größeren Datensätzen trainiert wurden. Zu diesen Modellen zählen u. a. aktuelle Versionen von GPT, LLaMA oder PaLM. Wie bereits erwähnt, können diese Modelle Texte in menschenähnlicher Weise verstehen und generieren, wodurch komplexe Probleme und anspruchsvolle Aufgaben bewältigen werden können (MINAEE et al. 2024, o. S.; ZHAO et al. 2023,

S. 1 ff.). In Zukunft wird die Weiterentwicklung dieser Sprachmodelle zu immer komplexeren Modellen führen (HANDSCHUH 2024, S. 22).

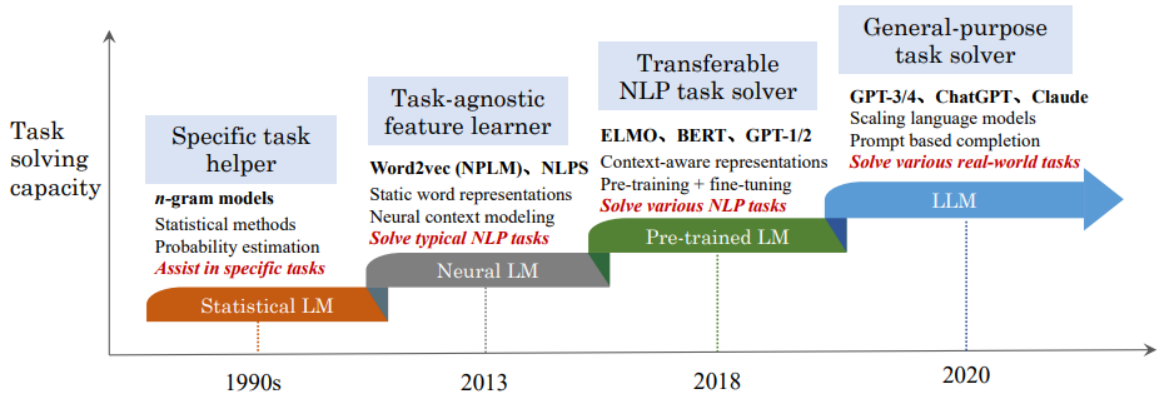


Abbildung 2: Entwicklung der Sprachmodelle, Quelle: (ZHAO et al. 2023, S. 2)

ALBRECHT (2023) beschreibt die Entwicklung der heutigen Sprachmodelle als technologischen Durchbruch. Die Weiterentwicklung der künstlichen Neuronalen Netze durch Transformer-Architekturen ermöglichen die effiziente Umwandlung von Sprache in mathematische Parameter. Diese Transformer nutzen sogenannte Self-Attention-Mechanismen, die es den Modellen erlauben lange Textabschnitte gleichzeitig zu verarbeiten und die Zusammenhänge zwischen Wörtern unabhängig von deren Position zu erfassen (ALBRECHT 2023, S. 20 f.; VASWANI et al. 2023). Dadurch wird die Effizienz und Genauigkeit der Sprachverarbeitung verbessert, was einen Fortschritt im Vergleich zu vorherigen, rein sequenziellen Ansätzen darstellt (MINAEE et al. 2024, o. S.). Ein Beispiel für die Anwendung dieser Architektur ist das Sprachmodell GPT-3, welches auf 175 Milliarden Parametern basiert und mit 300 Milliarden Token trainiert wurde (ALBRECHT 2023, S. 9). Als Token werden textliche Bausteine wie Wörter, Wortbestandteile oder Wortkombinationen verstanden, die es dem Modell ermöglichen menschenähnliche Texte zu generieren (BROWN et al. 2020, S. 5). Die Leistung der LLMs wird nicht ausschließlich von der Modellgröße oder der zugrunde liegenden Architektur bestimmt, sondern auch von der Qualität und Vielfalt der Trainingsdaten (KAPLAN et al. 2020, S. 1). Diese Modelle werden mit umfangreichen Datensätzen trainiert, wodurch sie ein tiefergehendes Verständnis linguistischer Strukturen entwickeln und kontextabhängige Antworten generieren können.

Das Training der GPT-Modelle erfolgt in zwei zentralen Schritten (ALBRECHT 2023, S. 25 f.). Der erste Schritt, das selbstüberwachte Lernen, kann von dem Modell eigenständig absolviert werden, indem es mit großen, nicht kategorisierten Datenmengen vortrainiert wird. ZHAO et al. (2023, S. 13 f.) kategorisieren diese Daten in fünf Bereiche: Webseiten, Bücher, Wikipedia, Code und „Weitere“. Im zweiten Schritt, dem überwachten Lernen, auch als Fine-Tuning bezeichnet, wird das Modell weiter angepasst. Dieser Prozess basiert auf einem Ansatz des Verstärkungslernens, dem sogenannten „Reinforcement Learning from Human Feedback“ (RLHF), bei dem menschliches Feedback genutzt wird, um die Qualität und Relevanz der generierten Inhalte zu optimieren (STRICKER 2024, S. 14).

Die für das Training verwendeten Internetquellen bringen spezifische Herausforderungen mit sich, insbesondere im Hinblick auf die Aktualität und domänenspezifische Relevanz der Informationen. Da die Modelle stark von dem Inhalt und der Qualität der Trainingsdaten abhängig sind, ergeben sich Einschränkungen bei der Fähigkeit, auf aktuelle Ereignisse oder spezialisiertes Wissen einzugehen (BARNETT et al. 2024, S. 1). Ein wesentliches Problem bei der Nutzung von LLMs ist das Phänomen der „Halluzinationen“. Dabei handelt es sich um die Generierung von Informationen, die irreführend oder nicht existent sind (JI et al. 2023, S. 3). Forschungsergebnisse von LIU et al. (2023, o. S.) zeigen, dass die Fähigkeit von LLMs, große Kontexte zu verarbeiten, nicht zwangsläufig zu einer Verbesserung der Leistung führt. Insbesondere bei langen Texteingaben in denen relevante Informationen mittig positioniert sind. Das Phänomen ist auch als „Lost in the Middle“ bekannt (LIU et al. 2023, o. S.). Darüber hinaus können implizite Gewichtungen in den Trainingsdaten zu weiteren Einschränkungen führen. Diese resultieren aus der Über- oder Unterrepräsentation bestimmter Inhalte und sind oft schwer zu identifizieren (ALBRECHT 2023, S. 42).

Zur Bewältigung der beschriebenen Herausforderungen werden in der Literatur zwei Lösungsansätze dargelegt. Der erste Ansatz umfasst das Fine-Tuning, dabei werden die Sprachmodelle mit den domänenspezifischen Daten trainiert. Der zweite Ansatz basiert auf der Integration von RAG-Systemen, die externe Informationsquellen

einbinden, um spezifisches Wissen gezielt abzurufen (BARNETT et al. 2024, S. 1). Im Rahmen dieser Arbeit wird der Ansatz des RAG-Systems gewählt. Dieser Ansatz ist kostengünstiger (RADEVA et al. 2024, S. 2) und erfüllt die Anforderungen des digitalen Einkaufsassistenten, indem das RAG-System ständig auf aktualisierte Daten zugreifen kann. Dies ist ein entscheidender Faktor für die Einbindung von Produktdaten.

4 Retrieval-Augmented Generation

Die folgenden Abschnitte werden die grundlegenden Funktionen von Retrieval-Augmented Generation erklären und die besonderen Merkmale von RAG in Verbindung mit strukturierten Daten aus relationalen Datenbanken erläutern. Außerdem werden die verschiedenen Bewertungsmethoden zur Evaluierung solcher Systeme untersucht.

4.1 Grundlagen

GAO et al. (2024a, S. 1) beschreibt Retrieval-Augmented Generation als Erweiterung von generativen Sprachmodellen durch Einbindung von Informationen aus externen Wissensquellen. Ziel dieser Systeme ist es, bekannte Herausforderungen der Large Language Models zu überwinden, indem externe Informationsquellen integriert werden. Dadurch können in einem QA RAG-System inhaltlich korrekte und auf den Kontext bezogene Antworten generiert werden (ZHAO et al. 2024).

Die Funktionsweise von RAG-Systemen folgt einem zweistufigen Prozess, bestehend aus dem Abruf von Informationen (Retrieval) und der anschließenden Generierung von Ausgaben (Generation). Zunächst werden die für die Benutzereingabe (Query) relevanten Daten aus einer umfangreichen Datenbank abgerufen. Anschließend nutzen die Sprachmodelle die abgerufenen Informationen, um eine finale Antwort auf die zuvor gestellte Query zu erstellen, die den spezifischen Kontext berücksichtigt (ZHAO et al. 2024, S. 1). Durch diesen Ansatz wird das Wissen der LLMs erweitert.

GAO et al. (2024a, S. 3 ff.) bezeichnen diesen grundlegenden Prozess als Naive RAG, der im Folgenden beschrieben wird. Der Ablauf eines solchen RAG-Systems wird zusätzlich in Abbildung 3 beispielhaft dargestellt.

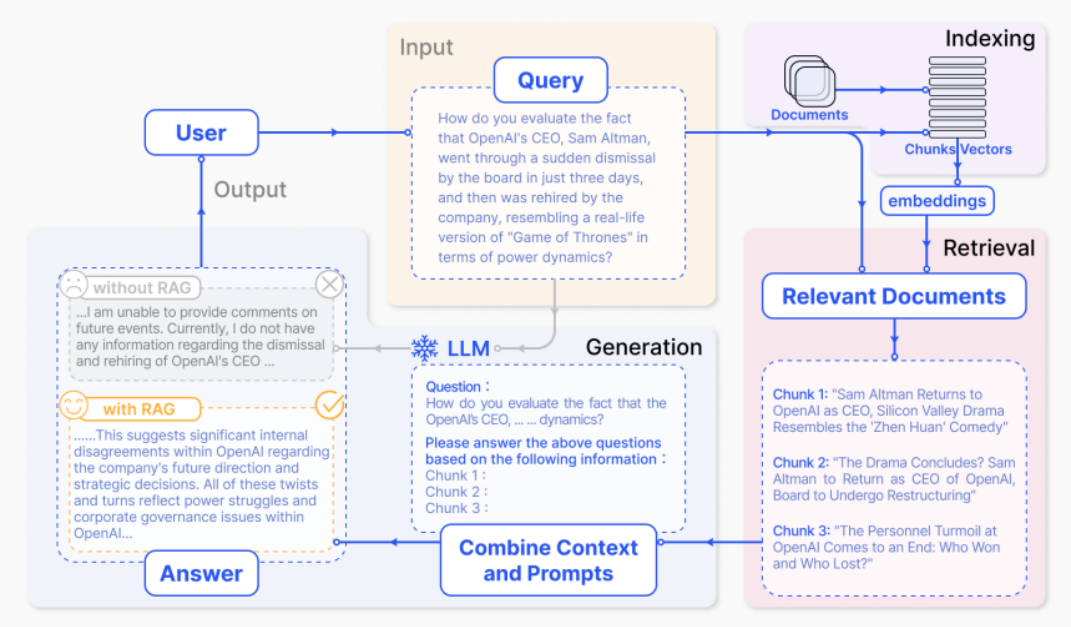


Abbildung 3: Ein repräsentativer Ablauf eines Frage-Antwort RAG-Systems, Quelle: (GAO et al. 2024a, S. 3)

In der Indexierung erfolgt die Vorverarbeitung der Daten. Wie in Kapitel 2 bereits erwähnt, können die Daten in strukturierter, unstrukturierter oder halbstrukturierter Form vorliegen. Während strukturierte Daten meist ohne Datenvorbereitung verwendet werden können, müssen unstrukturierte und halbstrukturierte Daten zunächst in lesbare Textformate wie PDF, Word oder Markdown konvertiert werden. Im nächsten Schritt werden die Dokumente in kleinere Textsegmente, sogenannte Chunks, unterteilt. Diese Textsegmente stellen den Kontext dar, auf dessen Grundlage die Benutzereingabe beantwortet wird. Die Größe dieser Chunks ist von zentraler Bedeutung, da sie sowohl den Informationsabruf als auch die Generierung der finalen Antwort beeinflusst. Sind die Chunks zu groß, kann dies dazu führen, dass sie nicht in das Kontextfenster des Large Language Models passen oder das Sprachmodell kann Schwierigkeiten bezüglich des Phänomens „Lost in the Middle“ bekommen (siehe Kapitel 3). Sind die Chunks hingegen zu klein, können potenziell wichtige Zusammenhänge zwischen den Textsegmenten verloren gehen. WANG et al. (2024, S. 5 f.) zeigen, dass die Methode

zur Einteilung der Chunks ebenfalls Einfluss auf das Retrieval haben kann. Anschließend werden die Chunks mithilfe eines Einbettungsmodells (Embedding-Modell) in Embedding-Vektoren umgewandelt. Als Vektoren werden numerische Darstellungen von Textdaten bezeichnet, die in einer hochdimensionalen Vektordatenbank gespeichert werden. Vektordatenbanken nutzen dabei Methoden wie Approximate Nearest Neighbor (ANN) oder k-Nearest-Neighbor (kNN), um ähnliche Vektoren identifizieren zu können, was für den folgenden Retrieval-Prozess entscheidend ist (FRIEDLAND 2024, S. 161 f.; WANG et al. 2024, S. 6).

Nach dem Eingang der Query im RAG-System, wird diese mithilfe desselben Einbettungsmodells wie bei der Indexierung in einen Vektor umgewandelt. Während des Retrievals wird dieser Vektor mit den in der Datenbank gespeicherten Vektoren der Chunks verglichen. Diese Vektoren, sowohl von der Query als auch von den Chunks, repräsentieren die semantische Bedeutung des Inhalts und ermöglichen einen Vergleich zwischen ihnen. Dieser Vergleich kann mithilfe verschiedenen metrischen Verfahren durchgeführt werden, wie der Kosinus-Ähnlichkeit, dem euklidischen Abstand (L2-Distanz) oder dem inneren Produkt, auch Skalarprodukt genannt (ZHAO et al. 2024, S. 4). Diese Vektorabstände dienen als Maß für deren Ähnlichkeit, kleine Abstände deuten auf eine hohe Verwandtschaft hin, große Abstände auf eine geringe (OPENAI o. J.).

Während der Generierungsphase werden die Chunks mit der größten Übereinstimmung gemeinsam mit der Benutzereingabe in einem Prompt an das LLM übergeben. Das Sprachmodell generiert basierend auf diesen Informationen eine kohärente und inhaltlich korrekte Antwort auf die gestellte Frage (GAO et al. 2024a, S. 3). Um die Antworten von dem LLM korrekt generieren lassen zu können, ist das Retrieval und somit die Auswahl der relevanten Chunks entscheidend. Eine falsche Auswahl der Chunks kann zu fehlerhaften oder inkorrekten Antworten führen (ES et al. 2023, o. S.).

Trotz der vielfältigen Einsatzmöglichkeiten können RAG-Systeme Einschränkungen haben. ZHAO et al. (2024, S. 16) identifizieren fünf zentrale Herausforderungen:

„Noises in Retrieval“, „extra Overhead“, „the Gap between Retrievers and Generators“, „increased System complexity“ und „lengthy context“. Zusätzlich können speziell die LLMs Schwächen in RAG-Systemen aufweisen. Dazu zählen “Noise Robustness”, “Negative Rejection”, “Information Integration” und “Counterfactual Robustness” (CHEN et al. 2023, o. S.)

4.2 RAG mit strukturierten Daten

Die überwiegende Anwendung von RAG-Systemen liegt in der Einbindung von unstrukturierten Daten (YANG et al. 2024, S. 64). Im Folgenden Abschnitt wird die Architektur des RAG-Systems beschrieben, welches strukturierte Daten in den Prozess einbinden kann.

Das grundlegende Konzept eines RAG-Systems wird unverändert bleiben, insbesondere im Hinblick auf die Retrieval- und Generierungsprozesse, wie sie in dem vorherigen Abschnitt 4.1 erläutert wurden (BYUN et al. 2024, S. 5). Die entscheidende Anpassung erfolgt in der Indexierungsphase, in der die Daten vorverarbeitet werden. Die strukturierten Daten aus der relationalen Datenbank werden in ein Textformat konvertiert, sodass anschließend die Informationen mit einem Embedding-Modell verarbeitet und in einer Vektordatenbank gespeichert werden können (KOHLLEFFEL 2024; YANG et al. 2024, S. 6). Dieses Format trägt die Bezeichnung JavaScript Object Notation (JSON) und speichert Informationen in Form von Schlüssel-Werte-Paaren (Key-Value-Pairs). Der Schlüssel ist eine Zeichenfolge, während der Wert entweder eine Zeichenfolge, eine Zahl, ein boolescher Ausdruck, ein Array oder ein Objekt sein kann (FROZZA et al. 2018, S. 357). Eine schematische Darstellung dieses Formats ist in Abbildung 4 zu sehen. In diesem Fall ist die ID der Schlüssel und die Zeichenfolge der Wert (siehe Abbildung 4). Die Informationen, die später mit dem Embedding-Modell verarbeitet und in der Vektordatenbank gespeichert werden, können ebenfalls in Chunks aufgeteilt (BYUN et al. 2024, S. 5).

```
{
    "id": "563bf305-2edd-4c4d-ae4e-8491098d28b1",
    "text": {
      "identifier": "Spareribs Schwein (Dicke Eiche)",
      "price_per_unit": 7.9,
      "name": "Spareribs Schwein",
      "description": null,
      "producer": "Dicke Eiche",
      "brand": null,
      "origin": "Bissendorf-Astrup",
      "supplier": "Dicke Eiche",
      "ingredients": null,
      "categories": "MEAT_FISH_FROZEN",
      "category_groups": "FROZEN"
    },
    "metadata": {
      "id": 2619
    },
    "embedding": [
      [
        -0.006565576884895563,
        0.04223889485001564,
        -0.014177992939949036,
        0.03667449206113815,
        0.02600938081741333,
        -0.025348957628011703,
        ...
      ]
    ]
}
```


Abbildung 4: Beispiel für ein JSON-Format (Modifikation: Mit Werten aus der in der Arbeit verwendeten Tabelle), Quelle: eigene Darstellung nach (BYUN et al. 2024, S. 6)

Die Entscheidung für die Nutzung eines RAG-Systems auf Basis strukturierter Daten, anstelle eines traditionellen Abfragesystems, wird unteranderem durch die Fähigkeit des Retrievals begründet. Vektordatenbanken sind in der Lage, sowohl Daten in strukturierter als auch unstrukturierter Form zu verwalten. Durch die Einbettung und die damit verbundene Umwandlung in Vektoren wird es möglich, unstrukturierte Benutzereingaben in natürlicher Sprache mit den strukturierten Daten der relationalen Datenbank zu vergleichen (BYUN et al. 2024, S. 5; YANG et al. 2024, S. 65).

Im Vergleich zu anderen Abfragesystemen, die auf exakten Suchmethoden basieren, ermöglicht die Vektortransformation eine Ähnlichkeitssuche. Dabei wird die Benutzereingabe, die vor der Verarbeitung in natürlicher Sprache vorliegt, mit den Vektoren der strukturierten Daten aus der Vektordatenbank verglichen. Dies kann zu besseren Ergebnissen führen (BYUN et al. 2024, S. 5). Ob ein solches RAG-System auch in der Praxis gute Ergebnisse vorweisen kann, wird in Kapitel 7 diskutiert.

4.3 Methoden und Kriterien zur Bewertung

In diesem Abschnitt werden mögliche Bewertungsmethoden und Bewertungskriterien aus der Literatur besprochen.

Bei der Evaluierung wird die Funktion der beiden zentralen Aspekte des RAG-Systems bewertet. Im Fokus steht, ob die gefundenen Chunks relevant sind und ob die Antwort korrekt formuliert ist (GAO et al. 2024a, S. 12). Kriterien dafür sind „Context Relevance“, „Answer Faithfulness“ und „Answer Relevance“ bewertet (ES et al. 2023, S. 3; SAAD-FALCON et al. 2023, o. S.). Darüber hinaus kann spezifisch die Leistung des LLMs bewertet werden, anhand von „Noise Robustness“, „Negative Rejection“, „Information Integration“ und „Counterfactual Robustness“ (CHEN et al. 2023, o. S.; GAO et al. 2024a, S. 12).

Bei dem Kriterium „Context Relevance“ wird bewertet, wie relevant die gefundenen Dokumente im Verhältnis zur Benutzereingabe sind. Der bereitgestellte Kontext soll dabei möglichst wenig irrelevante Informationen enthalten. Bei der „Answer Relevance“ wird evaluiert, ob die generierte Antwort in direktem Bezug zur Anfrage steht und die Kernfrage vollständig adressiert wird. „Answer Faithfulness“ prüft, ob die generierte Antwort ausschließlich auf dem bereitgestellten Kontext basiert, ohne zusätzliche, nicht verifizierbare Informationen eingefügt zu haben (ES et al. 2023, o. S.; GAO et al. 2024a, S. 12).

Die spezifischen Fähigkeiten eines LLMs sind entscheidender für dessen Leistung und verdeutlichen den Einfluss und die Anpassungsfähigkeit von RAG-Systemen auf die Sprachmodelle (CHEN et al. 2023, o. S.). „Noise Robustness“ beschreibt, inwieweit das LLM aus Dokumenten mit Rauschen relevante Informationen extrahieren kann. „Negative Rejection“ bedeutet, dass keine Antwort generiert werden soll, wenn die benötigten Informationen in den abgerufenen Dokumenten nicht enthalten sind. In diesen Fällen wird jedoch oft darauf hingewiesen, dass die gestellte Benutzereingabe mit den vorliegenden Informationen nicht beantwortet werden kann. Bei der „Information Integration“ wird bewertet, inwieweit das Modell Antworten auf komplexe Anfragen generieren kann, bei denen relevante Informationen aus mehreren Dokumenten kombiniert werden müssen. „Counterfactual Robustness“ bezeichnet die Fähigkeit des Sprachmodells, Ungenauigkeiten oder widersprüchliche Informationen zu erkennen, selbst wenn diese explizit in den abgerufenen Dokumenten enthalten sind (CHEN et al. 2023, o. S.). Mit den Aspekten „Context Relevance“ und „Noise Robustness“ wird die Qualität des Retrievals bewertet, während „Answer Faithfulness“, „Answer Relevance“, „Negative Rejection“, „Information Integration“ und „Counterfactual Robustness“ die Qualität der Generierung bewerten (GAO et al. 2024a, S. 12).

Die grundlegende Bewertung eines RAG-Systems kann entweder manuell durch Menschen oder automatisiert mithilfe von Frameworks erfolgen. Bei der manuellen Bewertung werden die vom RAG-System abgerufenen Dokumente und die generierten Antworten anhand eines sogenannten Evaluationsdatensatzes, auch als Goldstandard oder Benchmark bezeichnet, überprüft. Dazu werden im Vorfeld die für jede Frage erwarteten relevanten Dokumente sowie die korrekten Antworten definiert (KLEEBAUM 2024, S. 19 ff.). Die vom System generierten Antworten können anschließend einzeln anhand von zuvor beschriebenen Kriterien evaluiert werden.

Die manuelle Bewertung von RAG-Systemen ist durch ihre Abhängigkeit von subjektiven Einschätzungen eingeschränkt (CHIANG und LEE 2023). Automatisierte Evaluierungsmethoden bieten hier eine objektivere und ressourceneffizientere Alternative (SAAD-FALCON et al. 2023, o. S.). Die Bewertung großer Datenquellen werden durch Frameworks wie RAGAs, ARES und weiteren ermöglicht (ES et al. 2023, o. S.; SAAD-

FALCON et al. 2023, o. S.). Dadurch können RAG-Systeme in einem breiteren Kontext evaluiert werden. Die automatische Bewertung basiert auf Metriken. Für den Retrieval-Prozess werden häufig Metriken wie Accuracy, Precision und Recall verwendet, während für die Generierung Metriken wie BLEU, ROUGE oder BERTScore eingesetzt werden (YU et al. 2024, S. 9 ff.). Aufgrund von inhaltlichen Kapazitätsgrenzen werden die einzelnen Metriken in dieser Arbeit nicht behandelt.

5 Versuchsaufbau

Eine zentrale Aufgabe des digitalen Einkaufsassistenten wird die Beantwortung von Fragen zu spezifischen Produkten sein. Um diese produktbezogenen Fragen beantworten zu können, muss der Assistent über ein umfassendes Wissen der Produkte verfügen. In diesem Kapitel wird der Aufbau und die Umsetzung des Experiments detailliert erläutert. Dabei wird beschrieben, wie das RAG-System entwickelt wurde, welche Hyperparameter dabei eine Rolle spielen und nach welchen Kriterien der Fragenkatalog erstellt wurde. Abschließend wird das Verfahren zur Evaluation des Systems vorgestellt.

5.1 Architektur

Die Entwicklung des RAG-Systems verfolgt das Ziel eine einfache Weiterentwicklung ermöglichen zu können, um im weiteren Verlauf des Projekts Anpassungen und Optimierungen vornehmen zu können. Das Konzept der Entwicklung wurde so gestaltet, dass die zentralen Schritte des Prozesses steuerbar sind. Bei diesem RAG-System erfolgt die Benutzereingabe durch einen Fragenkatalog. Bei der Erstellung der Fragen werden auf einige Faktoren und Intentionen wie die Einbindung von Synonymen und Rechtschreibfehlern geachtet, um am Ende ein aussagekräftiges Ergebnis zu haben. Dies wird im nächsten Unterkapitel detailliert erläutert. Das RAG-System verarbeitet diesen Fragenkatalog und liefert am Ende eine Antwort auf die gestellten Fragen.

Das im weiteren Verlauf beschriebene RAG-System ist anhand von Daten aus einer einzigen Relation entwickelt und getestet worden, und nicht für Daten aus mehreren miteinander verknüpften Relationen einer RDB. Wie in Kapitel 4.1 beschrieben,

besteht ein RAG-System aus drei Phasen: Indexierung, Retrieval und Generierung. Entsprechend dieser Struktur werden im Folgenden die einzelnen Prozessphasen und ihre Hyperparameter erläutert.

Indexierung

Der für das RAG-System verwendete Datensatz besteht aus den Produktdaten des RegioStores. Diese Daten werden aus der relationalen Datenbank „MySQL“ mithilfe einer Python-Bibliothek abgerufen. Der Datensatz umfasst 3001 Einträge mit jeweils 30 Attributen, welcher im Rahmen der Indexierung gefiltert wird (siehe Anhang 1c). So wird sichergestellt, dass nur die relevanten Informationen in dem weiteren Prozess verarbeitet werden. Es werden dabei ausschließlich Attribute ausgewählt, die für potenzielle Produktfragen von Bedeutung sein können und ausreichende Werte enthalten. Zu diesen Attributen gehören beispielsweise „name“, „description“ und „identifier“ (siehe Anhang 1a). Attribute wie „next_delivery“ werden nicht berücksichtigt, da nur Werte wie „null“ enthalten sind oder die Attribute ausschließlich für administrative Zwecke genutzt werden, als Beispiel „created_by“ oder „last_modified_by“.

Das Speichern der Daten in kleine Chunks wird in diesem RAG-System nicht umgesetzt. In diesem Fall entspricht ein Chunk einem Tupel, das die relevanten Informationen eines Produkts aus den gefilterten Attributen enthält. Diese Chunks werden anschließend mit einem Embedding-Modell in Vektoren umgewandelt.

Das Embedding-Modell stellt eine der drei zentralen Hauptkomponenten eines RAG-Systems dar, die in der Abbildung 5 farblich dargestellt sind. Gleichzeitig ist der grundlegende Ablauf eines RAG-Systems zu erkennen. Es wird das Modell „text-embedding-3-large“ von OpenAI verwendet, welches Embedding-Vektoren in 3072 Dimensionen generieren kann und Benchmark-Werte von 54,9 % beim MIRACL-Benchmark und 64,6 % beim MTEB-Benchmark erzielt. Diese Benchmarks werden in der Literatur für die Bewertung der Leistung von Embedding-Modellen genutzt (VENKATESH und RAMAN 2024, S. 866; (SHARABIANI et al. 2024, S. 17 f.).

Bevor die generierten Embeddings in der Vektordatenbank gespeichert werden, werden sie zunächst zusammen mit den Textinhalten der Chunks, den zugehörigen Metadaten und einer eindeutigen Kennung (unique identifier) in einer JSON-Datei abgelegt. In den Metadaten sind in der aktuellen Entwicklung nur die IDs der Produkte enthalten. Die Daten werden in der JSON-Datei aus zwei Gründen gespeichert. Zum einen besteht die Möglichkeit, die Daten überprüfen zu können und zum anderen kann die Datei als zentraler Speicherort genutzt werden, von dem aus alle Daten in die Vektordatenbank übertragen werden. Um eine bessere Kontrolle über diesen Teil des Prozesses gewährleisten zu können, erfolgt die Einbettung der Chunks als separater Schritt und wird nicht von der Vektordatenbank durchgeführt.

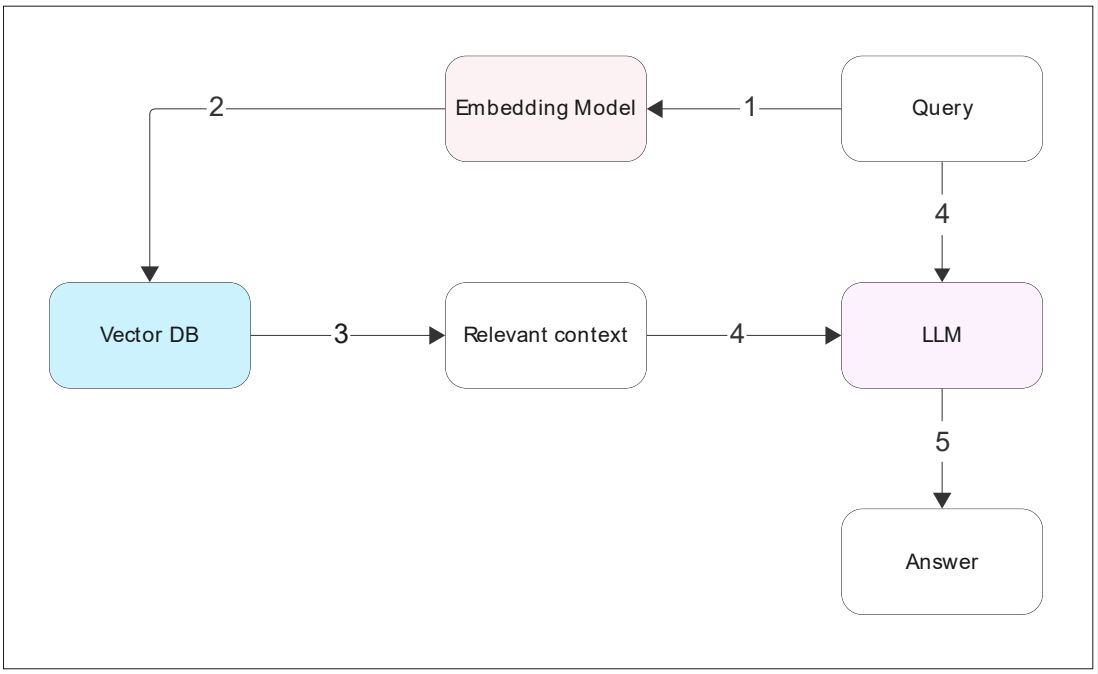


Abbildung 5: Drei Hauptkomponenten des RAG-Systems, Quelle: eigene Darstellung

Retrieval

Bei dem Retrieval wird die Benutzereingabe mithilfe desselben Embedding-Modells wie in der Indexierungsphase in Vektoren umgewandelt. In diesem System dient die Excel-Datei mit den Fragen als Input für das System. Jede Frage wird dabei eingebettet und mit den Vektoren aus der Vektordatenbank verglichen, um sicherzustellen,

dass die relevanten Chunks für jede Frage identifiziert werden. Das Retrieval umfasst die Identifizierung der relevanten Chunks. Dieser Prozess erfolgt in der Vektordatenbank, welche die zweite zentrale Komponente eines RAG-Systems ist (siehe Abbildung 5).

Für dieses System werden verschiedene Anforderungen an die Vektordatenbanken gestellt. Aufgrund verschiedener Vorteile wurde ChromaDB als Datenbank ausgewählt. Sie unterstützt eine große Anzahl an Dimensionen, ist mit dem verwendeten Embedding-Modell kompatibel, kostenfrei verfügbar, cloudbasiert, einfach in Python integrierbar und bietet eine gute Grundlage für die Implementierung in Chatbots (CHROMA o. J.; TAIPALUS 2024, S. 6 ff.). Die Metriken und Algorithmen zur Ähnlichkeitssuche wurden bereits in Kapitel 4.1 genannt. ChromaDB verwendet standardmäßig die Methode des euklidischen Abstands (L2) zur Berechnung der Distanzen zwischen den Vektoren, welche in diesem System unverändert übernommen wurde (CHROMA o. J.).

Durch diese Methode werden in der Vektordatenbank die Chunks identifiziert, die die größte Ähnlichkeit zur jeweiligen Frage aufweisen. Die Anzahl der abgerufenen Chunks ist dabei ein Hyperparameter im System, für den in dieser Arbeit eine Anzahl von acht Chunks pro Frage festgelegt wird. Die Anzahl der Chunks konnte erst nach Vollendung des Fragenkatalogs festgelegt werden, in der Entwicklungsphase wurde zuvor ein Richtwert angenommen. Die Anzahl an Chunks erfolgt durch die Berechnungen des Medians der erwarteten Chunks von jeder Kundenfrage. Dabei werden die Fragen, die keine relevanten Informationen in dem Datensatz enthalten können, nicht berücksichtigt. Auf diese Weise kann ein erweiterter Kontext für die Generierung bereitgestellt werden und es wird ermöglicht, dass gleichzeitig das LLM geprüft werden kann wie es mit Informationen umgeht, die nicht für die Beantwortung von Fragen relevant sind. In der abschließenden kritischen Auseinandersetzung in Kapitel 7 wird auf alternative Ansätze zur Chunk-Auswahl eingegangen, um mögliche Vor- und Nachteile dieses Vorgehens zu analysieren.

Generierung

Nachdem die relevanten Chunks für eine Frage identifiziert wurden, werden diese zusammen mit der Frage in einem Prompt an das LLM übergeben. Das Sprachmodell stellt die dritte Hauptkomponente eines RAG-Systems dar (siehe Abbildung 5). In diesem System wird das Modell „gpt-4o“ von OpenAI verwendet, das optimal mit dem ebenfalls von OpenAI entwickelten Embedding-Modell kompatibel ist. GPT-4 übertrifft bestehende Sprachmodelle in zahlreichen NLP Aufgaben (OPENAI et al. 2024, S. 14) und kann über ein Application Programming Interface (API) integriert werden. Eine API ermöglicht die Interaktion zwischen verschiedenen Programmen und erleichtert den Zugriff auf Daten und Funktionen (BAGGA 2023, S. 2).

Während der Generierung präziser Antworten spielt die Gestaltung des Prompts eine entscheidende Rolle (BYUN et al. 2024, S. 7). Ein Prompt besteht zum einen aus dem Systemprompt und zum anderen aus dem Nutzerprompt. Beide haben unterschiedliche Aufgaben und beeinflussen, wie das Modell auf Anfragen reagiert (ZHENG et al. 2024, o. S.). Der Systemprompt wird dem Modell vor dem Nutzerprompt übergeben und definiert den Grundrahmen sowie das Verhalten und die Rolle des Modells (ZHENG et al. 2024, o. S.). Der Nutzerprompt enthält die Benutzereingabe und den ermittelten Kontext aus dem Retrieval-Prozess. Auf Grundlage dieser Informationen erfolgt durch das LLM die Generierung der Antwort. An dieser Stelle wird angemerkt, dass aufgrund von Zeitbeschränkungen der Bachelorarbeit die Optimierung des Systemprompts nicht Gegenstand dieser Arbeit ist. Stattdessen ist der Prompt auf Basis existierender Beispiele aus der Literatur erstellt worden (BYUN et al. 2024; TOUVRON et al. 2023). In dem Nutzerprompt werden die Fragen und die abgerufenen Chunks sowie deren jeweilige IDs dargestellt, um den Vergleich mit den erwarteten Chunks automatisieren zu können.

Zusätzlich werden dem LLM zwei Parameter mitgegeben, die „temperature“ und „top_p“. Der Parameter „temperature“, der zwischen 0 und 1 liegen kann, beeinflusst die Kreativität und Vorhersehbarkeit der Antwort. In diesem RAG-System wird der Wert auf 0 gesetzt, sodass möglichst deterministische und reproduzierbare Antworten generiert werden (RADEVA et al. 2024, S. 11). Der Parameter „top_p“, der auch zwischen

0 und 1 liegen kann, bestimmt wie viele mögliche Wörter bei der Antwort berücksichtigt werden. Ein hoher Wert bedeutet, dass das Modell auch weniger wahrscheinliche Wörter einbeziehen kann. In diesem System wurde ein Wert von 0,5 gewählt, wodurch nur die wahrscheinlicheren Wörter berücksichtigt werden (VEGA 2023).

Der gesamte Prozess des entwickelten RAG-Systems, der in diesem Abschnitt beschrieben wurde, ist in Abbildung 6 schematisch veranschaulicht.

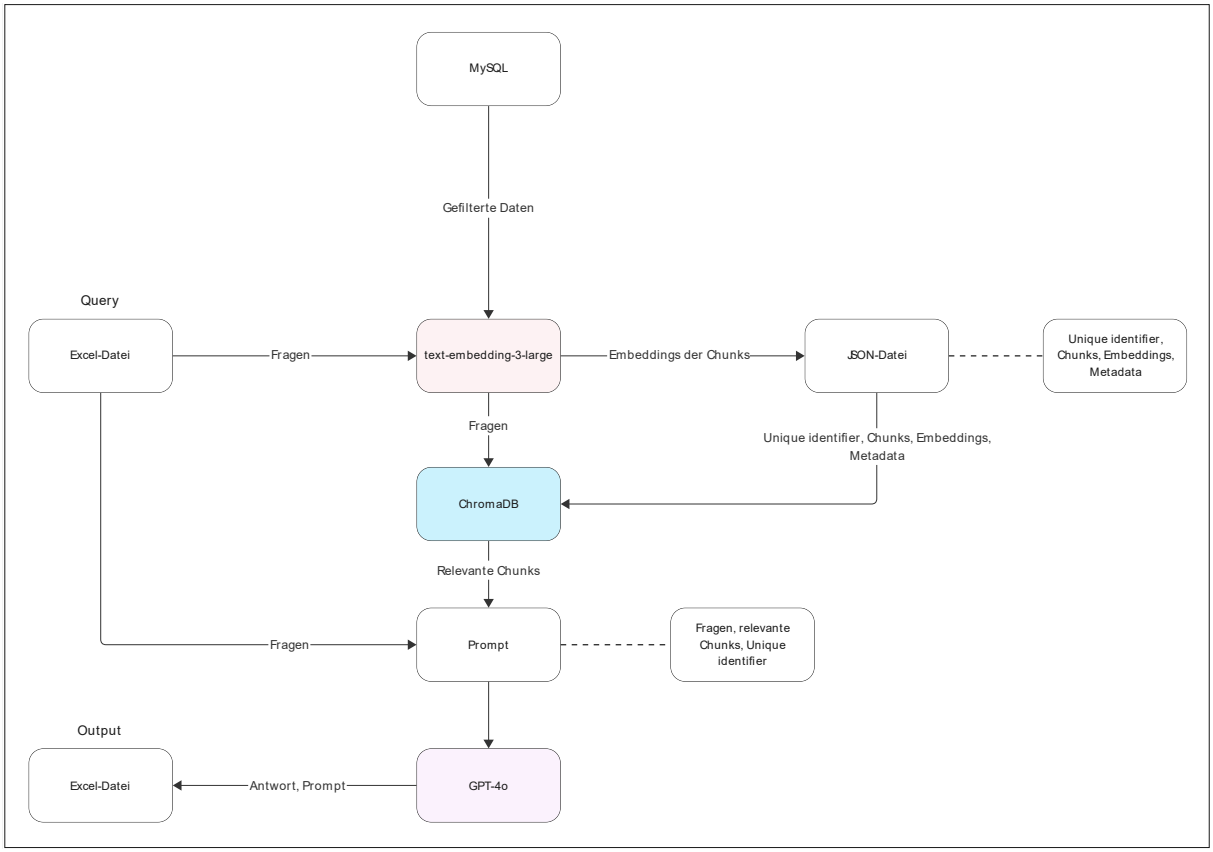


Abbildung 6: Ablauf des entwickelten RAG-Systems, Quelle: eigene Darstellung

5.2 Erstellung des Fragenkatalogs

Die Benutzereingabe erfolgt durch einen Fragenkatalog in Form einer Excel-Datei. Um bei der späteren Evaluierung aussagekräftige Ergebnisse erzielen zu können, die möglichst viele potenzielle Kundenfragen berücksichtigen, wurden insgesamt 45 Fragen formuliert. Die Fragen werden auf Grundlage der Produktdaten des RegioStores erstellt und beziehen sich auf verschiedene Anforderungen.

Die Fragen sind in verschiedene Kategorien und Typen unterteilt. Um bei der Generierung der Antworten das LLM auf unterschiedliche Antwortformate testen zu können, erfolgt die grundlegende Einteilung der Fragetypen in offene und geschlossene Fragen. Bei den geschlossenen Fragen können präzise und eindeutige Antworten generiert werden, die im Wesentlichen mit Ja oder Nein beantwortet werden können. Die offenen Fragen sollen dem LLM mehr Raum in der Beantwortung der Fragen bieten, wodurch längere und vielfältigere Antworten mit unterschiedlichen Aspekten berücksichtigt werden. Die Antworten auf offene Fragen sind oft nicht eindeutig und können mehrere mögliche Lösungsansätze umfassen (BOSEWITZ und BOSEWITZ 2021, S. 23).

Zusätzlich werden die offenen und geschlossenen Fragen anhand der jeweiligen Ziele und Funktionen der Frage in weitere Kategorien wie Existenzfrage, Überblicksfrage, Detailfrage, Interpretationsfrage, Toleranzfrage und Synonymfrage unterteilt. Abbildung 7 veranschaulicht diese Unterteilung mit Beispielen aus dem Fragenkatalog. Überblicksfragen sollen es ermöglichen, dass das Sortiment im Ganzen betrachtet werden kann. Detailfragen können gegenüber den Überblicksfragen spezifische Eigenschaften der Produkte oder Unterschiede innerhalb einer Produktgruppe erfragen. Existenzfragen, die ebenfalls nur in Form von geschlossenen Fragen formuliert werden, prüfen, ob ein bestimmtes Produkt im Sortiment vorhanden oder erhältlich ist. Zusätzlich werden die Fragen in Toleranz- und Synonymfragen eingeteilt. Die Toleranzfragen enthalten bewusst Rechtschreib- oder Tippfehler und testen, ob das Systems diese Fragen interpretieren und erkennen kann. Ziel ist es, eine fehlertolerante Antwort generieren zu können, auch wenn die Eingabe des Nutzers nicht präzise formuliert ist. Bei den Synonymfragen werden unterschiedliche Begriffe für das gleiche Produkt verwendet. Diese Fragen sollen prüfen, ob das RAG-System in der Lage ist, verschiedene Ausdrücke die dieselbe Bedeutung haben, identifizieren zu können. Die Interpretationsfragen stellen eine besondere Herausforderung dar, da die Bedeutung oder Absicht der Fragen nicht eindeutig formuliert werden. Sie erfordern vom RAG-System eine inhaltliche Deutung, um ermitteln zu können, welche Intention der Nutzer hat. Solche Art von Fragen kann verschiedene Antworten hervorrufen, da sie oft auf

subjektiven Einschätzungen basieren. Dies erschwert sowohl die Beantwortung als auch die Bewertung solcher Fragen.

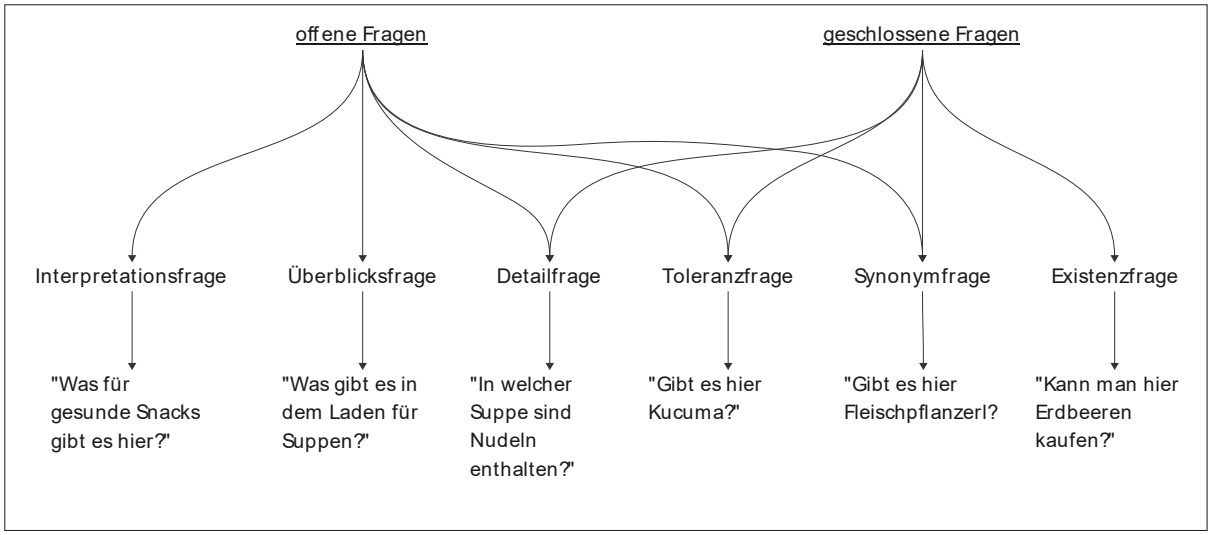


Abbildung 7: Übersicht Fragetypen und -kategorien mit Beispielen, Quelle: eigene Darstellung

Bei der Einteilung der Fragen wird darauf geachtet, ob eine Frage mehreren Kategorien zugeordnet werden kann. Im Fragenkatalog werden die Fragen so eingeteilt, dass jede Frage der Kategorie zugeordnet wird, die die größte Anforderung im jeweiligen Kontext darstellt. Eine Frage wie „Welche Saucen kann ich für mein Grillfleisch kaufen?“ könnte sowohl als Überblicksfrage eingeordnet werden, da sie einen Überblick über das Sortiment bietet, als auch als Interpretationsfrage, da die Antwort subjektive Präferenzen enthalten kann. Diese Frage wird dementsprechend in der Kategorie Interpretationsfrage eingeordnet.

Außerdem werden Fragen formuliert, die bestimmte Anforderungen an das LLM beinhalten können, unabhängig von den Kategorien und Typen. Sie werden so gestaltet, dass es keine relevanten Chunks zu den Fragen geben kann und somit eine Verneinung als Antwort gefordert wird. Dadurch wird überprüft, wie das RAG-System mit fehlenden Informationen umgeht und ob es trotzdem eine korrekte Antwort generieren kann. Andere Fragen enthalten mehrere Produkte in einer einzigen Anfrage, um zu testen, ob die Intention der Anfrage korrekt erfasst wird. Ein Beispiel dafür ist die Frage

„In welcher Suppe sind Nudeln enthalten?“ in der die Produkte Suppe und Nudeln angesprochen werden. Darüber hinaus werden Fragen erstellt, die sich auf spezifische Zutaten in Lebensmitteln beziehen wie laktosefreie Produkte oder Lebensmittel mit „Mais und Hackfleisch“, um die Fähigkeit des Systems zur präzisen Beantwortung solcher Anfragen bewerten zu können.

Ziel ist es, durch diese Fragen die Funktionalität stichprobenartig überprüfen zu können und sicherzustellen, dass das RAG-System in der praktischen Anwendung (dem digitalen Einkaufsassistenten) zuverlässig auf alle Nutzeranfragen reagieren kann. Die verschiedenen Fragestellungen dienen dazu, die Stärken und Schwächen des Systems aufzudecken.

5.3 Bewertung

Die Bewertung des RAG-Systems erfolgt manuell durch qualitative und quantitative Bewertungsaspekte, die in diesem Abschnitt erläutert werden. Die manuelle Bewertung wird auf der Grundlage von zwei Aspekten ausgewählt. Einerseits lässt der Umfang dieser Arbeit die Einführung automatisierter Verfahren nicht zu und andererseits kann dadurch sichergestellt werden, dass die Intention der speziell formulierten Fragen richtig interpretiert wird.

Die berechneten Distanzen in der Vektordatenbank, insbesondere bei hohen Dimensionen können schwer interpretiert und evaluiert werden (PENG et al. 2024, o. S.). Um die gefundenen Chunks und die Antworten dennoch bewerten zu können, muss ein Benchmark erstellt werden. Dieser wird in der Excel-Datei neben den generierten Antworten des LLMs gespeichert, in der auch die gesamte Bewertung stattfinden wird. Dabei werden die erwarteten Chunks und Antworten für die jeweiligen Fragen festgelegt. Die Chunks werden nicht in Textform, sondern in Form der Produkt-ID angegeben, um sie einfacher mit der ID der Chunks aus dem Prompt abgleichen zu können. Bei den Fragen, die keine erwarteten Chunks enthalten können, werden auch keine Chunks eingetragen.

Wie bereits erwähnt werden die Fragen und somit auch die Produkte für erwartete Antwort auf Basis des Produktdatensatz ausgewählt. Die Auswahl wird subjektiv vom Autor festgelegt. Dabei kann gleichzeitig die Kategorie der Frage, beispielsweise Interpretationsfrage die Auswahl intuitiv beeinflussen. Insbesondere bei dieser Art von Fragen können die erwarteten Antworten und Chunks aufgrund der möglichen unterschiedlichen Interpretation der Frage, nicht eindeutig festgelegt werden. Wie im vorherigen Abschnitt beschrieben, kann beispielsweise die Frage „Was gibt es für Brotsorten?“ unterschiedlich von den Bewertern interpretiert werden. Es bleibt unklar, ob Begriffe wie Toastbrot oder Knäckebrot unter den Begriff „Brot“ fallen. Für diesen Fall wird die Antwort auf „normales“ Brot eingeschränkt, um eine klare Bewertungsgrundlage schaffen zu können. Auch bei den anderen Interpretationsfragen werden die erwarteten Antworten und die zugehörigen Chunks intuitiv festgelegt. Diese Vorgehensweise der Auswahl der Chunks kann auch die spätere Bewertung der Ergebnisse beeinflussen, dazu in Kapitel 7 mehr.

Für die anderen Fragekategorien ergeben sich die erwarteten Antworten und Dokumente direkt aus der Fragestellung. Dabei werden in der Auswahl der Produkte alle möglichen Zustände oder Variationen eines Produkts berücksichtigt. Beispielsweise bei Erdbeeren die in gefrorener und frischer Form vorhanden sind oder bei verschiedenen Suppen. Es wird zudem unterschieden, ob die Frage auf das Vorhandensein eines bestimmten Produkts abzielt oder ob das gesuchte Produkt als Bestandteil anderer Produkte enthalten sein soll. Ein Beispiel hierfür ist die Frage 31: „Welche Produkte gibt es, in denen Rote Bete enthalten ist?“. Es wird nicht das eigentliche Produkt Rote Bete gesucht, sondern die Produkte in denen Rote Bete als einfache Zutat, jedoch nicht als Mikrozutat vorhanden sein kann.

Im Verlauf der Entwicklung des RAG-Systems wird festgestellt, dass die gefundenen Chunks des Retrievals auch bei mehreren Durchläufen identisch bleiben (siehe Anhang 1d-f). Um auch die generierte Antwort deterministischer machen zu können, werden die erwähnten Parameter „temperature“ und „top_p“ angepasst. So sind auch die Antworten meist gleichbleibend. Dadurch ist eine deterministische Bewertung des RAG-Systems möglich, bei der die Antworten nur einmalig generiert und dann bewertet

werden. Aufgrund der umfangreichen und komplizierten Aspekte bei der Bewertung der generierten Antwort, wird diese nur grundlegend evaluiert, der Fokus wird auf das Retrieval gelegt.

Bei der Bewertung des RAG-Systems wird zwischen den beiden Hauptprozessen unterschieden, der Antwortgenerierung und dem Dokumenten-Retrieval. Während das Retrieval quantitativ bewertet wird, erfolgt die Bewertung der Generierung qualitativ. Da die Generierung der Antworten wie beschrieben nicht im Fokus dieser Arbeit steht, erfolgt die Bewertung qualitativ. Dabei wird an den im Folgenden beschriebenen Kriterien evaluiert.

Die Bewertungskriterien orientieren sich an den in Kapitel 4.3 beschriebenen Aspekten. Das Retrieval wird nach dem Kriterium „Context Relevance“ bewertet und erfolgt durch die Berechnung der Trefferquote. Die Trefferquote wird definiert als das Verhältnis der Anzahl relevanter abgerufener Chunks zur Anzahl der erwarteten Chunks, wobei die maximale Anzahl durch den festgelegten Chunk-Input begrenzt wird. Sie lässt sich mathematische folgendermaßen formulieren:

$$\textbf{Trefferquote} = \frac{\text{Anzahl der relevanten abgerufen Chunks}}{\text{Anzahl der erwarteten Chunks } (\text{max. Chunk} - \text{Input})}$$

Folgendes Beispiel dient zur Erklärung. Dem LLM werden in diesem RAG-System acht Dokumente übergeben. Wenn vier davon relevant sind und insgesamt sechs relevante Dokumente erwartet werden, beträgt die Trefferquote etwa 0,67 oder 67%. In einem anderen Szenario sind alle der acht gefundenen Dokumente relevant und 20 verschiedene relevante Dokumente werden erwartet. In diesem Fall beträgt die Trefferquote dennoch 100 %, da maximal acht Chunks dem LLM als Input mitgegeben werden.

Bei den Fragen, zu denen keine relevanten Informationen vorliegen können, wird keine Trefferquote berechnet. Folglich werden diese Fragen auch nicht für die weiteren

Berechnungen bei der Bewertung des Retrievals einbezogen. Bei der Generierung werden diese Fragen hingegen trotzdem bewertet, um das Verhalten des LLMs evaluieren zu können.

Um die Auswertungen unter den einzelnen Fragekategorien und -Typen vergleichen zu können, werden die Mittelwerte, Standardabweichungen und Varianzen berechnet (siehe Anhang 1b). Die Standardabweichung gibt ein Verständnis über die Streuung der Mittelwerte (Wilke 2019). Sie hat die gleiche Maßeinheit wie der Mittelwert und sorgt so für eine bessere Interpretation. Eine höhere Standardabweichung zeigt große Streuung an und eine niedrige eine kleine Streuung, wodurch der Mittelwert eine stärkere Aussagekraft hat (CERESOLI et al. 2022, S. 59; WILKE 2019). Die mathematischen Formeln lauten:

$$\textbf{Mittelwert} = \frac{\text{Summe der Werte}}{\text{Anzahl der Werte}}$$

$$\textbf{Varianz} = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$n = \text{Gesamtzahl der Werte}$$
$$x_i = \text{Beobachtungswerte}$$
$$\bar{x} = \text{Mittelwert}$$

$$\textbf{Standardabweichung} = s = \sqrt[+]{s^2}$$

Die Bewertungsaspekte der Generierung orientieren sich ebenfalls an den in Kapitel 4.3 erläuterten Kriterien. Dazu werden die folgenden Bewertungskriterien für die spezifischen Anforderungen des Fragenkatalogs erstellt:

1. Wurde die Frage erkannt?
2. Bezieht sich die Antwort direkt auf die Frage?
3. Sind die Informationen in der Antwort faktisch korrekt und mit den zugrunde liegenden Daten konsistent?
4. Wurden falsche Informationen in der Antwort vermieden?
5. Wurde die Intention der Frage korrekt interpretiert?

Die Evaluierung erfolgt anhand eines binären Bewertungssystems, das auf einer Checkliste basiert. Die Kriterien werden entweder als „erfüllt“ oder „nicht erfüllt“ bewertet, wobei in der zugehörigen Excel-Datei eine „1“ für „erfüllt“ und eine „0“ für „nicht erfüllt“ eingetragen wird. Die Ergebnisse der Bewertung werden in der Excel-Datei systematisch dokumentiert (siehe Anhang 1g).

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse des Versuchs dargestellt. Die Bewertungen wurden vom Autor selbst durchgeführt. Die vollständigen Ergebnisse des Versuchs einschließlich der generierten Antworten des RAG-Systems, die Berechnung der Trefferquoten, die Bewertung der Antworten sowie der Code zum Erstellen der Visualisierungen und des RAG-Systems ist auf GitHub eingestellt (siehe Anhang 1). Im Folgenden werden die Ergebnisse zusammenfassend für die verschiedenen Fragekategorien und Fragetypen vorgestellt, wobei besondere Auffälligkeiten hervorgehoben werden, um einen umfassenden Überblick bieten zu können.

Abbildung 8 zeigt deutliche Unterschiede in der Funktionalität des Retrieval-Prozesses zwischen den einzelnen Fragekategorien. Diese sind an den verschiedenen Trefferquoten der gefundenen Chunks zu erkennen. Existenzfragen, Überblicksfragen und Detailfragen weisen Mittelwerte der Trefferquote von mindestens 90 % auf, wobei Überblicksfragen sogar 100 % und Existenzfragen 96 % erreichen. Im Gegensatz dazu liegen die Trefferquoten für Interpretationsfragen bei lediglich 60 % im Mittel, während Synonymfragen und Toleranzfragen mit 31 % bzw. 44 % deutlich unter der 50-Prozent-Marke bleiben. Wie in der Abbildung auch zu erkennen ist, ist die Standardabweichung

bei den Kategorien mit hohen Trefferquoten vergleichsweise geringer, mit 10 % bei den Existenz- und 23 % bei den Detailfragen. Dies bedeutet, dass die Trefferquoten im Schnitt um 10 % bzw. 23 % von den Mittelwerten abweichen. Besonders hervorzuheben ist der Wert von 0 % bei den Überblicksfragen. Bei den Detailfragen beträgt die Standardabweichung knapp 23 %, was auf Ausreißer innerhalb der Ergebnisse hindeutet. Diese können den Mittelwert negativ beeinflussen. Die drei Kategorien mit niedrigeren Trefferquoten weisen hingegen eine deutlich höhere Standardabweichung auf und weichen somit stärker von den Mittelwerten ab. Interpretationsfragen weisen eine Abweichung von 39 % auf, Synonymfragen 27 % und Toleranzfragen ebenfalls 39 %. Das zeigt eine deutliche Variabilität dieser Werte (siehe Anhang 1g).

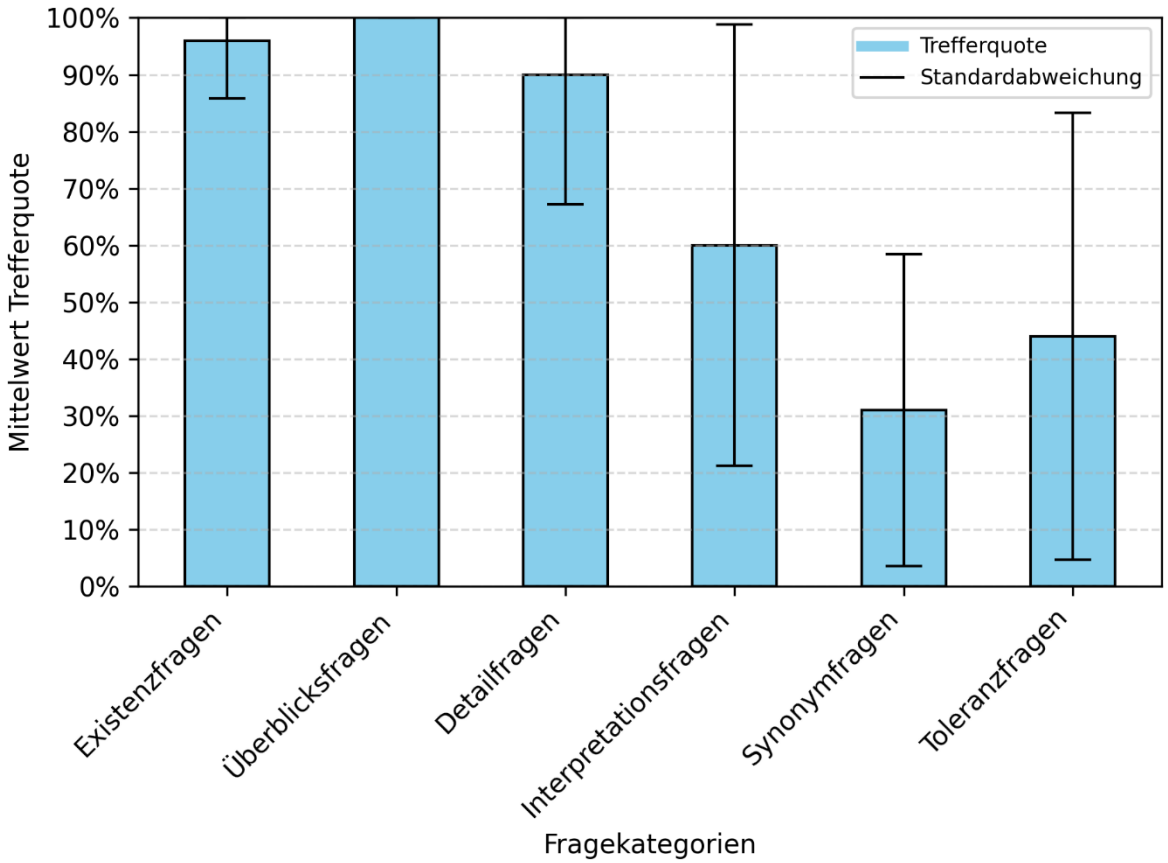


Abbildung 8: Vergleich der Trefferquoten und Standardabweichungen anhand der Fragekategorien, Quelle: eigene Darstellung

Die Trefferquoten lassen sich zudem zwischen den beiden Fragetypen geschlossene und offene Fragen vergleichen. Für die Berechnung der Mittelwerte wurden ebenfalls

die Fragen, bei denen vorsätzlich keine Chunks gefunden werden können wie Frage 2 oder Frage 7, ausgeschlossen (siehe Anhang 1g). Damit basieren die Berechnungen auf 17 geschlossenen und 23 offenen Fragen. In Abbildung 9 wird deutlich, dass sich die Trefferquoten der beiden Fragetypen mit 68 % bei den geschlossenen und 65 % bei den offenen Fragen nur geringfügig unterscheiden. Auch die Standardabweichung zeigt mit 40 % bei den geschlossenen und 37 % bei den offenen Fragen eine ähnliche Variabilität. Es kann festgehalten werden, dass eine insgesamte durchschnittliche Trefferquote von 66% erzielt wurde.

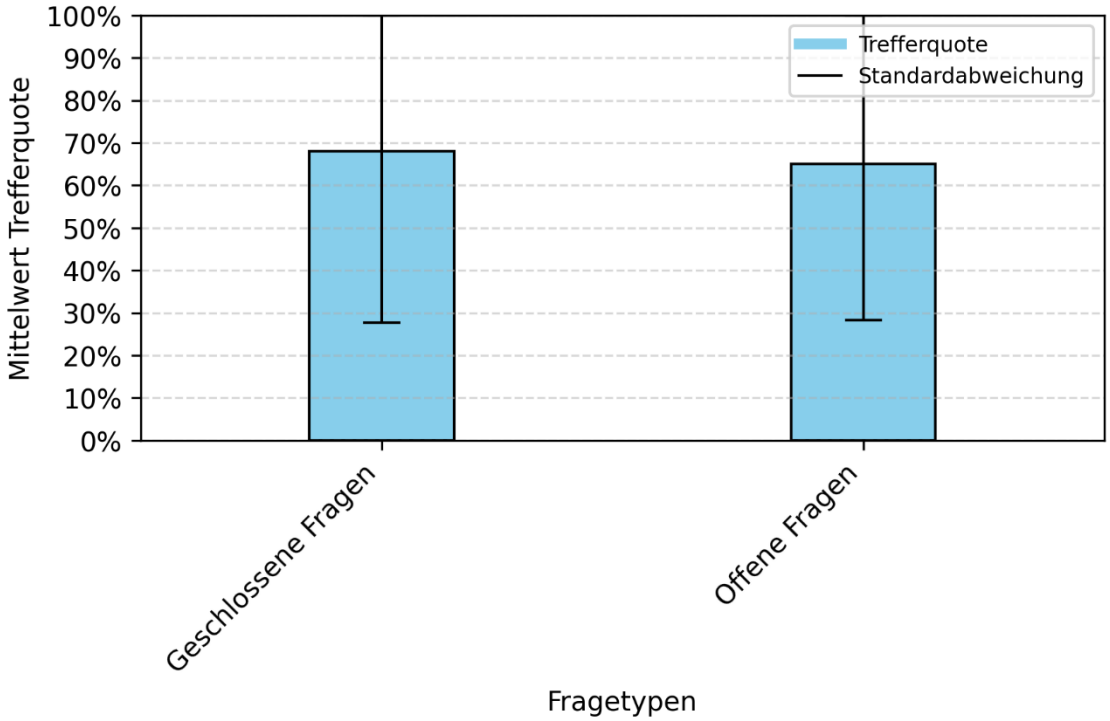


Abbildung 9: Vergleich der Trefferquoten und Standardabweichungen anhand der Fragetypen, Quelle: eigene Darstellung

Die Auswertung der generierten Antworten ist in Tabelle 1 zusammengefasst. Die Ergebnisse werden nach den Fragekategorien und Bewertungskriterien differenziert dargestellt. Diese Bewertungskriterien sind voneinander unabhängig. Deshalb kann eine Frage bei dem einen Kriterium als „nicht erfüllt“ gelten, während sie bei dem anderen als „erfüllt“ bewertet wird. Um eine vollständig richtige Antwort zu erlangen, müssen alle Kriterien als „erfüllt“ bewertet werden. Dieses Ergebnis wird in der Spalte „Richtige Antworten“ zusammengefasst. Wenn die Frage von dem Retrieval nicht erkannt wird,

werden auch die Ergebnisse der generierten Antwort beeinflusst. Dennoch wird in diesen Fällen die Antwort an den anderen Kriterien bewertet, ob die im Prompt vorhergesehene Antwort generiert wurde oder unzutreffende Informationen eingebunden wurden.

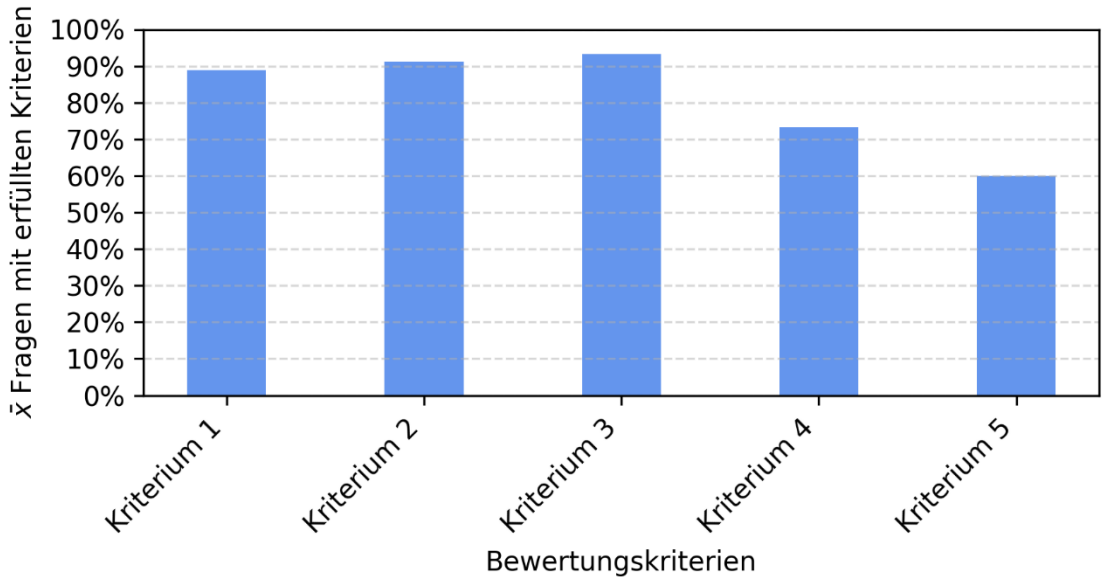
Tabelle 1: Bewertung der generierten Antworten nach den Kriterien und zusammengefasst für die Fragekategorien

Fragekategorien	Wurde die Frage erkannt?	Bezieht sich die Antwort direkt auf die Frage?	Sind die Informationen in der Antwort faktisch korrekt und mit den zugrunde liegenden Daten konsistent?	Wurden falsche Informationen in der Antwort vermieden?	Wurde die Intention der Frage korrekt interpretiert?	Richtige Antworten
Existenzfragen	10/10	10/10	10/10	8/10	8/10	8/10
Überblicksfragen	2/2	2/2	2/2	2/2	2/2	2/2
Detailfragen	10/10	10/10	10/10	10/10	9/10	9/10
Interpretationsfragen	9/9	8/9	7/9	2/9	2/9	2/9
Synonymfragen	4/6	5/6	6/6	4/6	2/6	2/6
Toleranzfragen	5/8	6/8	7/8	7/8	4/8	4/8
Summe	40/45	41/45	42/45	33/45	27/45	27/45

Anmerkung: Der Wert auf der linken Seite des Schrägstrichs gibt die Anzahl an Fragen an, die das jeweilige Kriterium in der Fragekategorie erfüllt haben und der Wert auf der rechten Seite gibt die Gesamtanzahl der Fragen in der jeweiligen Kategorie an. Zusätzlich werden die Ergebnisse aller Kriterien in der Zeile „Summe“ und die Anzahl an korrekten Antworten in den Kategorien in der Spalte „Richtige Antworten“ zusammengefasst.

Insgesamt wurden 40 von den 45 Fragen als „erkannt“ bewertet, dabei sind in den Kategorien Synonymfragen zwei und Toleranzfragen drei Fragen als „nicht erkannt“ eingestuft wurden. Wie die Bewertung dieser generierten Antworten aussehen kann, ist exemplarisch an den Fragen mit den IDs 34 oder 43 ersichtlich (siehe Anhang 1g). Dabei ist es wichtig zu prüfen wie das LLM in solchen Fällen reagiert, welche Antwort es folglich generiert. Das erklärt die Resultate, warum beispielsweise bei den Toleranzfragen nur fünf von acht Fragen erkannt wurden, aber bei den anderen Kriterien teilweise sechs oder sieben Fragen als „erfüllt“ bewertet wurden. Um die Gesamtanzahl der als „erfüllt“ bewerteten Fragen aller Kriterien zu verdeutlichen, wird in der Abbildung 10 ergänzend zu der Tabelle der prozentuale Mittelwert visualisiert. Dabei ist zu erkennen, dass 89 % der Fragen das erste Kriterium erfüllen, 91 % der Fragen das zweite Kriterium und 93% das dritte Kriterium. Diese Ergebnisse zeigen eine

insgesamt gute Leistung des Systems in diesen Aspekten. Die Vermeidung falscher Informationen wird jedoch nur bei 73 % der Fragen erreicht, während die korrekte Interpretation der Frage lediglich bei 60 % gelingt. Folglich werden die höchsten Ergebnisse bei dem dritten und die niedrigsten Ergebnisse bei dem fünften Kriterium erzielt. Insgesamt wurden 27 der 45 Fragen nach der Bewertungsmethodik als richtig beantwortet, was einer Erfolgsquote von 60 % entspricht.



Kriterium 1: Wurde die Frage erkannt?
Kriterium 2: Bezieht sich die Antwort direkt auf die Frage?
Kriterium 3: Sind die Informationen in der Antwort faktisch korrekt und mit den zugrunde liegenden Daten konsistent?
Kriterium 4: Wurden falsche Informationen in der Antwort vermieden?
Kriterium 5: Wurde die Intention der Frage korrekt interpretiert?

Abbildung 10: Mittelwert aller Fragen die das jeweilige Kriterium erfüllen, Quelle: eigene Darstellung

Insbesondere bei den Interpretationsfragen zeigt sich eine geringere Erfolgsquote, mit nur zwei von neun Fragen, welche die Kriterien 4 und 5 erfüllen (siehe Tabelle 1). Auch bei den Synonymfragen fällt die Erfassung der richtigen Intention mit zwei von sechs Fragen gering aus, ebenso wie bei den Toleranzfragen, bei denen vier von acht Fragen das Kriterium erfüllen. Auffällig sind die identischen Ergebnisse bei den Existenz-, Toleranz- und Überblicksfragen für die Kriterien 1 bis 3. Ebenfalls identisch sind die Ergebnisse für alle Kriterien bei den Überblicksfragen.

Die detaillierte Betrachtung der Ergebnisse zeigt weitere Auffälligkeiten bei den Antworten und Trefferquoten des RAG-Systems. Für genauere Informationen zu den nachfolgend beschriebenen Aspekten wird auf den Anhang 1 verwiesen. Teilweise wurden Antworten als richtig bewertet, sie erfüllen alle Bewertungskriterien, weisen jedoch trotzdem Unstimmigkeiten auf. Ein Beispiel dafür ist die Frage 29, bei der in der generierten Antwort ein relevantes Produkt fehlt, was von dem Retrieval korrekt gefunden wurde. Ein vergleichbares Problem zeigt sich bei der Frage 38, bei der zwei fettarme Joghurts von dem Retrieval an das LLM übergeben wurde, jedoch nur eines der beiden Produkte in der Antwort enthalten ist.

Besonders auffällig ist die Frage 34 „Welche Äfte gibt es?“. Die Trefferquote von 38 % weist auf korrekt identifizierte Chunks hin und auch die Informationen in der Antwort wurden faktisch korrekt wiedergegeben. Dennoch wurden die anderen Kriterien als „nicht erfüllt“ bewertet, aufgrund von falscher Interpretation und somit falschen Produkten in der Antwort. Sie enthält Apfelprodukte und nicht explizit Säfte, was die erwartete Antwort verfehlt. Bei den anderen Fragen, die das Kriterium „Wurde die Frage erkannt?“ nicht erfüllen, beträgt die Trefferquote 0 %.

Ähnlich ist es bei der Frage 33. Hier sind die Bewertung und das Ergebnis nicht eindeutig. Aufgrund der Trefferquote von 20 % wurde in diesem Fall angenommen, dass das Retrieval das falsch geschriebene Wort „Cilla“ teilweise erkennt, jedoch das LLM den Zusammenhang zwischen dem Wort und dem Produkt nicht. Somit wird es nicht in der Antwort verwendet. Diese Frage zeigt, dass die genutzte Bewertungsmethode nicht alle Aspekte berücksichtigen kann.

Bei vielen Ergebnissen wird die Abhängigkeit der generierten Antworten von dem Retrieval deutlich. Die Antworten bei Fragen mit niedrigen Trefferquoten wurden oft als nicht richtig bewertet. Dies kann hingegen in einigen Fällen nicht bestätigt werden. Fragen wie beispielsweise mit der ID 10 oder 24 haben eine Trefferquote von 100 %, erfüllen jedoch nicht alle Kriterien bei der Antwort.

Abschließend kann festgestellt werden, dass die Antworten und Trefferquoten in den Kategorien Existenz-, Überblicks- und Detailfragen deutlich bessere Ergebnisse erbringen als in den anderen drei Fragekategorien. Um die Ergebnisse in diesen weniger erfolgreichen Kategorien verbessern zu können, sind Optimierungen des RAG-Systems erforderlich, die in der anschließenden Auseinandersetzung und im Ausblick näher erläutert werden.

7 Kritische Auseinandersetzung

In dem folgenden Kapitel werden die Ergebnisse des Einsatzes eines RAG-Systems für die Verarbeitung von Produktdaten aus relationalen Datenbanken im Kontext eines digitalen Einkaufsassistenten analysiert. Dabei wird untersucht, ob die Forschungsfrage beantwortet werden kann. Neben den Optimierungsmöglichkeiten und technischen Erweiterungen des Systems, werden die Herausforderungen in der Evaluierung und den festgelegten Parametern des RAG-Systems beleuchtet.

In der vorliegenden Arbeit wurde das Ziel der Forschungsfrage umgesetzt, inwiefern sich ein RAG-System auf strukturierte Daten aus einer relationalen Datenbank anwenden lässt und wie potenzielle Kundenfragen die Qualität des Retrievals und der generierten Antworten beeinflusst. Dazu wurde ein grundlegendes RAG-System entwickelt, welches Produktdaten aus relationalen Datenbanken einbindet. Im Anschluss wurde es anhand von 45 potenziellen Kundenfragen getestet und bewertet. Die Ergebnisse dieser Bewertung zeigen, dass die Einbindung von strukturierten Daten generell möglich ist. Es wurden jedoch mehrere Herausforderungen im Zusammenhang mit den verschiedenen Fragen festgestellt.

Beginnend mit der unterschiedlichen Trefferquote, kann festgestellt werden, dass die niedrigeren Werte bei den Interpretations-, Synonym- und Toleranzfragen aus der falschen Interpretation oder fehlenden Erkennung der Frage resultieren. Bei den Interpretationsfragen liegt es größtenteils an den erwarteten Chunks von dem Benchmark. Diese wurden von dem Autor selbst festgelegt, weshalb auch die Interpretationen an die des Autors gebunden sind. Die Trefferquote kann in diesen Fällen besser ausfallen,

wenn andere Personen die Fragen unterschiedlich interpretieren. Beispielsweise könnten bei Frage 17 und 27, zusätzliche Grillfleischalternativen oder andere gesunde Produkte in dem Benchmark, einerseits eine höhere Trefferquote erzeugen und andererseits auch die Ergebnisse der Bewertung der generierten Antwort verbessern.

Bei den Synonym- und Toleranzfragen sind die geringeren Werte darauf zurückzuführen, dass nicht alle Fragen erkannt wurden. Um dieser Problematik entgegenzuwirken, könnten Verfahren wie „Query Rewriting“ eingeführt werden, sodass die Benutzereingaben angepasst werden (MA et al. 2023). Eine programmatische Lösung könnte außerdem die Einführung von Rückfragen sein. Dadurch kann das RAG-System bei missverständlichen Aussagen des Benutzers, eine Rückfrage stellen.

Während der Entwicklungsphase wurde das System an Fragen wie „Habt ihr Tee, der gegen Erkältung hilft?“ oder „Mein Hauptgang besteht aus Steak mit Bratkartoffeln, welchen Nachtisch kann ich machen?“ getestet. Dabei wurde festgestellt, dass das RAG-System auf diese Art von Fragen keine inhaltlich korrekte Antwort generieren kann. Solche Fragen setzen einen zweistufigen Prozess voraus, indem zunächst geklärt werden muss, welcher Tee gegen Erkältung hilft und dann abgefragt werden kann, ob die Teesorte in den Produkten vorhanden ist. Nur die Abfrage der vorhandenen Produkte kann durch das bestehende RAG-System durchgeführt werden. Aus diesem Grund sind diese Art von Fragen kein Bestandteil des Fragenkatalogs. Eine Erweiterung eines solchen Prozesses könnte die Anwendung des digitalen Einkaufsassistenten verbessern.

Selbst wenn die Herausforderungen des Retrievals durch die beschriebenen Ansätze bewältigt werden können, zeigen die Ergebnisse, dass die Antworten abhängig von der Deutung des LLMs sind. Trotz der korrekt gefundenen Chunks wurden nicht alle Informationen in der Antwort verwendet. Mögliche Ursachen dafür könnten die unzureichende Gewichtung oder Anordnung der Chunks sein. Zukünftige Arbeiten könnten Ansätze wie eine verbesserte Weitergabe der Chunks durch „Reranking“ oder Einbindung von Metadaten untersuchen (GAO et al. 2024b; WANG et al. 2024, S. 7 f.).

Ein weiterer Faktor ist der Prompt. Die Gestaltung des Prompt-Design kann die Leistung des LLM stark beeinflussen (BYUN et al. 2024, S. 7 f.). Neben den erwähnten Verbesserungen des Nutzerprompts durch Ansätze wie „Query Rewriting“, sollte der Systemprompt ebenfalls angepasst werden. Obwohl es keine Relevanz für die Bewertung hat, zeigen die generierten Antworten wie sie von Frage zu Frage variieren. Ein Beispiel dafür sind die Fragen 5, 8 und 20. Das Format wie die verschiedenen Produkte aufgelistet wurden, ist unterschiedlich. Außerdem beinhalten Antworten teilweise mehr Informationen als andere, wie Preis- und Herstellerangaben bei den Fragen 4 und 6. Eine mögliche Ursache dafür kann der Datensatz sein, der für den Einsatz im RAG-System nicht bereinigt und optimiert wurde (siehe Anhang 1c). Aufgrund der teilweise fehlenden Informationen bei den Attributen, kann die Qualität der generierten Antwort beeinträchtigt werden. Auch technische Parameter wie „temperatur“ oder „top_p“ können eine entscheidende Rolle bei der Beantwortung der Fragen spielen, wie in Kapitel 5.1 erläutert. Durch Analyse der Anforderungen für dieses RAG-System, wie und welche Informationen die Antwort enthalten soll, kann ein spezifisches Template für den Prompt erstellt werden, um die aufgezeigten Faktoren zu beheben (BYUN et al. 2024).

Die Verwendung von falschen Informationen in der Antwort kann nicht nur die Leistung des Sprachmodells, sondern auch das Retrieval betreffen. Die Anzahl der übergebenen Chunks an das LLM beeinflusst die Generierung der Antworten (GAO et al. 2024b, S. 5; SETTY et al. 2024, S. 5). Durch eine zu geringe Anzahl können nicht alle relevanten Informationen dargestellt werden, während bei einer zu hohen Anzahl an Chunks, die falschen Informationen in der Antwort enthalten sein können (WANG et al. 2024; YEPES et al. 2024). Dies zeigen die Fragen, bei denen die Antwort nur ein Produkt beinhalten soll, aber die generierte Antwort mehrere enthält. Da die Anzahl der Chunks in dem RAG-System aufgrund von Berechnungen des Medians der zu erwarteten Chunks, festgelegt wurde, sind die Ergebnisse stark davon abhängig. Dadurch kann die Anzahl der Chunks bei Veränderung des Benchmarks variieren, wodurch die Leistung des LLMs und die Trefferquote beeinflusst werden kann. An dieser Stelle könnte ein programmatischer Ansatz, bei dem die Anzahl der mitgegebenen Chunks für die Fragen variiert, Abhilfe schaffen.

Andere zentrale Einflussfaktoren können die speziell verwendeten Hauptkomponenten des RAG-Systems sein. Die Sprachmodelle werden mit verschiedenen Daten und teilweise für unterschiedliche Schwerpunkte trainiert, was Auswirkungen auf die Antwortgenerierung haben kann (HANDSCHUH 2024; KAPLAN et al. 2020). Auch das Embedding-Modell in Verbindung mit der Vektordatenbank kann insbesondere die Leistung des Retrievals beeinflussen (WANG et al. 2024, S. 6). Zu Beginn der Entwicklungsphase wurde noch das Embedding-Modell „text-embedding-3-small“ verwendet. Dabei konnten in keinem Testdurchlauf die erwarteten Chunks bei den beiden Fragen 19 und 20 gefunden werden. Ohne Veränderung anderer Parameter, nur bei dem Wechsel zu dem Embedding-Modell „text-embedding-3-large“, wurden die richtigen Chunks gefunden und die Fragen konnten beantwortet werden. Dies unterstützt die These, dass ein Embedding-Modell die Leistung des Retrievals beeinflussen kann. Gleiches gilt für die Vektordatenbank, die verschiedene Vor- und Nachteile mit sich bringen können (WANG et al. 2024, S. 6). Ob und welche Auswirkungen unterschiedliche Hauptkomponenten auf die Ergebnisse haben, könnte in anderen Arbeiten untersucht werden.

Die Grundlage der Bewertung sind die vom Autor verfassten Fragen. Dieser Fragenkatalog kann nicht alle möglichen Szenarien oder Intentionen aus dem realen Alltag widerspiegeln. Außerdem ist zu beachten, dass die Fragen bestimmte Muster in der Formulierung aufweisen können, an welche sich das RAG-System anpassen kann. Dies kann zu verzerrten Ergebnissen führen. Auch die Gesamtanzahl der Fragen und die in den einzelnen Kategorien sind nur Stichproben, an denen das RAG-System getestet wird. Für aussagekräftigere Ergebnisse könnten noch weitere Fragen hinzugefügt werden. Ebenso können Fehler bei dem subjektiv erstellten Benchmark auftreten, was sich auf die Bewertung des Systems auswirkt.

Ein weiterer Aspekt betrifft die Bewertung selbst, sowohl die Berechnung der Trefferquote, als auch die Bewertung der Generierung mit der Checkliste. Obwohl die Trefferquote grundsätzlich ein geeignetes Maß für die Leistung des Retrievals darstellt, ist die Berechnung nicht ohne Schwächen zu betrachten. Wird die Trefferquote auf Basis des maximalen Chunk-Inputs berechnet, so haben Fragen mit vielen erwarteten

Chunks gegenüber solchen mit einer geringen Anzahl an erwarteten Chunks, unabhängig von der tatsächlichen Qualität des Retrievals eine höhere Wahrscheinlichkeit auf eine gute Trefferquote. Wird die Trefferquote nicht auf Basis des maximalen Chunks-Inputs, sondern auf Grundlage der insgesamt erwarteten Chunks berechnet, entstehen ebenfalls Verzerrungen des Ergebnisses. Im folgenden Beispiel wird ein maximal Chunk-Input von acht Chunks angenommen. Wenn von 60 erwarteten Dokumenten fünf korrekte und drei falsche dem LLM übergeben werden, ist das Retrieval bei dieser Frage tendenziell schlechter als wenn fünf richtige Chunks und drei falsche dem LLM übergeben werden bei einer Frage die nur acht Chunks insgesamt erwartet. Gleichzeitig könnte bei einer Frage, bei der alle acht Dokumente richtig übergeben werden, bei einer Berechnung basierend auf den insgesamt erwarteten 60 Dokumenten ein schlechtes Ergebnis aufweisen. Dies würde die tatsächliche Leistung des Retrievals nicht angemessen widerspiegeln, da alle möglichen Chunks korrekt identifiziert wurden.

Das Kernproblem der Bewertungsmethodik der generierten Antworten ist die Evaluierung anhand von qualitativen Kriterien, die nur mit „erfüllt“ oder „nicht erfüllt“ gekennzeichnet wurden. Die Ergebnisse sind abhängig von der subjektiven Meinung des Autors. Speziell wurden die Antworten nur inhaltlich bewertet, während die sprachlichen und stilistischen Unterschiede wie die Anordnung von Wörtern oder die Vollständigkeit der enthaltenen Daten (ob bei manchen Antworten mehr Informationen als bei anderen enthalten sind) nicht berücksichtigt wurden. Zusätzlich kann die Interpretation der als „richtig“ bewerteten Antworten kritisch gesehen werden. In der Bewertung müssen alle Kriterien erfüllt werden, um als „richtige Antwort“ gelten zu können. Die Fragen, die keine korrekten Chunks von dem Retrieval erhalten, da sie nicht erkannt wurde und dennoch die vorgesehene Standardantwort aus dem Prompt zurückgibt, kann diese Antwort auch als richtig bewertet werden.

Ein weiteres Hindernis bei der Interpretation der Ergebnisse kann sich aus den unterschiedlichen Stichprobengrößen der Fragekategorien und Fragetypen ergeben. Um eine einheitliche Vergleichbarkeit zwischen den Fragen gewährleisten zu können, sollten gleiche Stichprobengrößen verwendet werden.

Ausblick

Der zentrale Aspekt dieser Arbeit ist die Einbindung von den strukturierten Daten aus einer relationalen Datenbank. Das entwickelte RAG-System verarbeitet die Daten aus einer einzelnen Tabelle. In der Realität kann es jedoch vorkommen, dass die relevanten Daten von einem Unternehmen in mehreren Tabellen gespeichert sind, die miteinander verknüpft sind. Für diesen Anwendungsfall muss das RAG-System angepasst werden, sodass alle relevanten Informationen zusammenhängend in der Vektordatenbank vorhanden und zugreifbar sind.

Weitere Einflussfaktoren wurden bereits aufgeführt. Der primäre Kritikpunkt ist die Bewertungsmethode dieser Arbeit. Alle Ergebnisse basieren auf der Interpretation des Autors. Sowohl bei der Erstellung des Benchmarks, wodurch die Ergebnisse der Trefferquote beeinflusst werden, als auch bei der Bewertung der generierten Antworten, die anhand von qualitativen Kriterien durchgeführt wurde. Die generierten Antworten sollten in Zukunft von den in Kapitel 4.3 erwähnten quantitativen Bewertungsmethoden evaluiert werden. Benchmarks wie RAGAs, ARES oder CRUD sind nur Beispiele für die Auswahl an Möglichkeiten (ES et al. 2023; SAAD-FALCON et al. 2023; YU et al. 2024). Eine andere Methode könnte „LLM-as-a-judge“ sein. Das LLM bewertet den generierten Text an Kriterien wie Koheränz, Relevanz und Geläufigkeit (YU et al. 2024, S. 11). Die Skalierbarkeit und Erklärbarkeit werden als wesentliche Vorteile herausgestellt (ZHENG et al. 2023). Die LLMs können die Bewertung nicht nur automatisieren, sondern begründen auch deren Einschätzung, was zusätzliche Transparenz schafft.

Die bestehenden Herausforderungen für den Einsatz eines grundlegendes RAG-Systems wurden bereits in den Ergebnissen und zu Beginn des Kapitels dargestellt. Die Umsetzung dieser Ansätze in einem RAG-System werden auch Advanced RAG und Modular RAG genannt (GAO et al. 2024a, S. 3 f.). In dem Advanced RAG werden zwei Phasen hinzugefügt: Pre-Retrieval und Post-Retrieval. Bei dem Pre-Retrieval werden sowohl die Indexierung als auch die Benutzereingaben optimiert. In der Post-Retrieval-Phase wird die Einbindung der abgerufenen Chunks durch Techniken wie „Reranking“ oder Zusammenfassungen verbessert (GAO et al. 2024a, S. 4). Das Modular RAG unterscheidet sich grundlegend vom Naive RAG und Advanced RAG durch seine

modulare Architektur. Dabei werden spezifische Aufgaben von unterschiedlichen Modulen übernommen (GAO et al. 2024b, S. 1 ff.).

Für die Implementierung eines RAG-Systems in dem digitalen Einkaufsassistenten muss das bestehende System weiterentwickelt werden. Die beschriebenen Ansätze sollten in Verbindung mit den strukturierten Daten und den Kundenfragen in zukünftigen Forschungen getestet werden.

8 Zusammenfassung

RAG-Systeme werden immer häufiger in Bereichen wie dem Kundenservice zur Beantwortung von spezifischen Fragen eingesetzt. Dieser Anwendungsbereich zeigt sich auch in dem Projekt „InVerBio“ der Hochschule Osnabrück. Die Herausforderung ist die Einbindung von strukturierten Daten aus einer relationalen Datenbank.

Dieser Herausforderung widmet sich die vorliegende Arbeit mit der folgenden Forschungsfrage: „Wie lässt sich ein Retrieval-Augmented Generation System auf strukturierte Daten aus einer relationalen Datenbank anwenden und wie beeinflussen unterschiedliche Kundenfragen die Qualität des Retrievals und der generierten Antworten?“. Dabei wird das Ziel verfolgt, ein grundlegendes RAG-System zu entwickeln, welches anhand von potenziellen Kundenfragen getestet wird.

Ein solches RAG-System mit Einbindung von strukturierten Daten konnte in dieser Arbeit erfolgreich entwickelt werden. Dieses spezifische System verwendet sowohl das Embedding-Modell „text-embedding-3-large“, als auch das LLM „gpt-4o“ von OpenAI und als Vektordatenbank wird „ChromaDB“ verwendet. Bei der externen Wissensquelle wird auf eine Produkttabelle aus der Datenbank „MySQL“ zugegriffen. Jeder Eintrag der Tabelle ist ein Chunk, wobei Informationen aus spezifischen Feldern gefiltert werden. Für die Bewertung des Systems wird ein Fragenkatalog mit insgesamt 45 potenziellen Kundenfragen erstellt. Dabei werden verschiedene Intentionen und

Szenarien berücksichtigt, um das RAG-System mit möglichst realistischen Fragen zu testen.

Die Ergebnisse zeigen, dass ein RAG-System mit Einbindung von relationalen Daten gute Ergebnisse erzielen kann. Insgesamt wurden 27 der 45 Fragen, folglich 60 % richtig beantwortet und die Trefferquote des Retrievals liegt bei insgesamt 66 %. Insbesondere bei Fragen wie den die Existenz-, Überblicks- und Detailfragen konnte das System positive Ergebnisse erzielen. In diesen drei Kategorien wurden 19 von den 22 Fragen richtig beantwortet und die Trefferquote liegt bei allen über 90 %. Auffällig sind die Schwächen der Synonym- und Toleranzfragen, sowohl bei dem Retrieval, als auch bei der generierten Antwort. Die Trefferquote der beiden Kategorien liegt unter 50 % und insgesamt wurden nur sechs der 14 Fragen richtig beantwortet.

Die erzielten Ergebnisse werden im Kontext von technischen Erweiterungsmöglichkeiten des RAG-Systems und der Herausforderungen in der Evaluierung kritisch betrachtet. Es zeigt sich, dass die grundlegende Version dieses RAG-Systems erweitert werden sollte, um bessere Ergebnisse bei den speziellen Fragen zu erzielen. Als mögliche Optimierungen werden Systeme wie Advanced RAG oder Modular RAG genannt. Dazu zählen Verfahren wie „Query Rewriting“, „Reranking“ oder der Einsatz von Metadaten.

Abschließend kann die Forschungsfrage dieser Arbeit beantwortet werden: Das entwickelte RAG-System zeigt, dass die Einbindung von strukturierten Daten erfolgreich umgesetzt werden konnte. Dabei erzielt das System bei der Beantwortung von potenziellen Kundenfragen unterschiedliche Ergebnisse. Bei grundlegenden Fragen erweisen sich das Retrieval und die generierten Antworten als effektiv. Allerdings wurden komplexere Fragen teilweise nicht erkannt oder falsch beantwortet. Daraus ergibt sich, dass die Qualitäten des Retrievals und der generierten Antwort von den Anforderungen der Fragen abhängig sind. Bevor demnach geplant wird das System in den digitalen Einkaufsassistenten zu integrieren, sollten technische Erweiterungen untersucht werden.

9 Abstract

Retrieval-Augmented Generation (RAG) systems are mainly used for unstructured text data. This bachelor's thesis addresses the challenge of integrating structured data into a RAG system. The aim is to develop a basic RAG system that can integrate structured product data from a relational database. The embedding model ("text-embedding-3-large") and the LLM ("gpt-4o") are used from OpenAI. "ChromaDB" is chosen as the vector database. In order to be able to test and evaluate the system, a list of questions was created. These questions are designed to represent potential customer questions, while considering various intentions and possible scenarios. The results of the evaluation showed that a basic RAG system is insufficient to answer all complex customer questions. To overcome the identified challenges in future research, possible technical enhancements were presented.

Literaturverzeichnis

AL GHADBAN, Y., LU, H. (Yvonne), ADAVI, U., SHARMA, A., GARA, S., DAS, N., KUMAR, B., JOHN, R., DEVARSETTY, P., HIRST, J. E. (2023): Transforming Healthcare Education: Harnessing Large Language Models for Frontline Health Worker Capacity Building using Retrieval-Augmented Generation. DOI: https://doi.org/10.1101/2023.12.15.23300009.

ALBRECHT, S. (2023): ChatGPT und andere Computermodelle zur Sprachverarbeitung – Grundlagen, Anwendungs- potenziale und mögliche Auswirkungen.

BAGGA, J. (2023): Introduction to Integration Suite Capabilities: Learn SAP API Management, Open Connectors, Integration Advisor and Trading Partner Management. Berkeley, CA: Apress.

BARNETT, S., KURNIAWAN, S., THUDUMU, S., BRANNELLY, Z., ABDELRAZEK, M. (2024): Seven Failure Points When Engineering a Retrieval Augmented Generation System. http://arxiv.org/abs/2401.05856.

BOSEWITZ, A., BOSEWITZ, R. (2021): Erfolgreiche Vorstellungsgespräche auf Englisch: 101 Fragen und die besten Antworten. o. O.: Haufe Lexware.

BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D. M., WU, J., WINTER, C., HESSE, C., CHEN, M., SIGLER, E., LITWIN, M., GRAY, S., CHESS, B., CLARK, J., BERNER, C., MCCANDLISH, S., RADFORD, A., SUTSKEVER, I., AMODEI, D. (2020): Language Models are Few-Shot Learners. http://arxiv.org/abs/2005.14165.

BYUN, J., KIM, B., CHA, K.-A., LEE, E. (2024): Design and Implementation of an Interactive Question-Answering System with Retrieval-Augmented Generation for Personalized Databases. Applied Sciences 14 (17)7995. DOI: https://doi.org/10.3390/app14177995.

C., S., KASIVISWANATH, N., CHENNA, P. (2015): A Survey on Big Data Management and Job Scheduling. IJCA 130 (13)41–49. DOI: https://doi.org/10.5120/ijca2015907161.

CERESOLI, M., ABU-ZIDAN, F. M., STAUDENMAYER, K. L., CATENA, F., COCCOLINI, F. (Hrsg.) (2022): Statistics and Research Methods for Acute Care and General Surgeons. Cham: Springer International Publishing.

CHEN, J., LIN, H., HAN, X., SUN, L. (2023): Benchmarking Large Language Models in Retrieval-Augmented Generation. http://arxiv.org/abs/2309.01431.

CHIANG, C.-H., LEE, H. (2023): Can Large Language Models Be an Alternative to Human Evaluations? Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 15607–15631. DOI: https://doi.org/10.18653/v1/2023.acl-long.870.

CHROMA (o. J.): Chroma Docs - Distance-function. https://docs.trychroma.com/guides#changing-the-distance-function (Zugriff am 10.12.2024).

CODD, E. F. (1970): A relational model of data for large shared data banks. Commun. ACM 13 (6)377–387. DOI: https://doi.org/10.1145/362384.362685.

COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., KUKSA, P. (2011): Natural Language Processing (almost) from Scratch. http://arxiv.org/abs/1103.0398.

ES, S., JAMES, J., ESPINOSA-ANKE, L., SCHOCKAERT, S. (2023): RAGAS: Automated Evaluation of Retrieval Augmented Generation. http://arxiv.org/abs/2309.15217.

FRIEDLAND, G. (2024): Informationsgesteuertes maschinelles Lernen: Data Science als Ingenieurdisziplin. Cham: Springer International Publishing.

FROZZA, A. A., MELLO, R. D. S., COSTA, F. D. S. D. (2018): An Approach for Schema Extraction of JSON and Extended JSON Document Collections. 2018 IEEE International Conference on Information Reuse and Integration (IRI), 356–363. DOI: https://doi.org/10.1109/IRI.2018.00060.

GAO, Y., XIONG, Y., GAO, X., JIA, K., PAN, J., BI, Y., DAI, Y., SUN, J., GUO, Q., WANG, M., WANG, H. (2024a): Retrieval-Augmented Generation for Large Language Models: A Survey. http://arxiv.org/abs/2312.10997.

GAO, Y., XIONG, Y., WANG, M., WANG, H. (2024b): Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks. DOI: https://doi.org/10.48550/arXiv.2407.21059.

GUPTA, S., RANJAN, R., SINGH, S. N. (2024): A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. DOI: https://doi.org/10.48550/arXiv.2410.12837.

HANDSCHUH, S. (2024): Grosse Sprachmodelle. Informationswissenschaft: Theorie, Methode und Praxis 8 (1)11–29. DOI: https://doi.org/10.18755/iw.2024.3.

HUANG, Y., XU, J., LAI, J., JIANG, Z., CHEN, T., LI, Z., YAO, Y., MA, X., YANG, L., CHEN, H., LI, S., ZHAO, P. (2024): Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey. http://arxiv.org/abs/2311.12351.

IBM (2021): DB2 10.5.0: Not Null - Integritätsbedingung https://www.ibm.com/docs/de/db2/10.5?topic=constraints-not-null (Zugriff am 19.11.2024).

JI, Z., LEE, N., FRIESKE, R., YU, T., SU, D., XU, Y., ISHII, E., BANG, Y. J., MADOTTO, A., FUNG, P. (2023): Survey of Hallucination in Natural Language Generation. ACM Comput. Surv. 55 (12)248:1-248:38. DOI: https://doi.org/10.1145/3571730.

KAPLAN, J., MCCANDLISH, S., HENIGHAN, T., BROWN, T. B., CHESS, B., CHILD, R., GRAY, S., RADFORD, A., WU, J., AMODEI, D. (2020): Scaling Laws for Neural Language Models. DOI: https://doi.org/10.48550/arXiv.2001.08361.

KLEEBAUM, A., HUSS, J. (2024): Der Weg zum autarken KI-Chatbot: Ein Praxisbericht. https://www.java-forum-stuttgart.de/vortraege/der-weg-zum-autarken-ai-chatbot-ein-praxisbericht/ (Zugriff am 26.12.2024).

KOHLLEFFEL, K. (2024): Rethinking the use of structured data in RAG: An example with Fivetran and Snowflake. https://medium.com/snowflake/rethinking-the-use-of-structured-data-in-rag-an-example-with-fivetran-and-snowflake-d524f939d6ba (Zugriff am 08.12.2024).

LIU, N. F., LIN, K., HEWITT, J., PARANJAPE, A., BEVILACQUA, M., PETRONI, F., LIANG, P. (2023): Lost in the Middle: How Language Models Use Long Contexts. http://arxiv.org/abs/2307.03172.

MA, X., GONG, Y., HE, P., ZHAO, H., DUAN, N. (2023): Query Rewriting for Retrieval-Augmented Large Language Models. DOI: https://doi.org/10.48550/arXiv.2305.14283.

MINAEE, S., MIKOLOV, T., NIKZAD, N., CHENAGHLU, M., SOCHER, R., AMATRIAIN, X., GAO, J. (2024): Large Language Models: A Survey. http://arxiv.org/abs/2402.06196.

MODRAN, H., BOGDAN, I. C., URSUȚIU, D., SAMOILA, C., MODRAN, P. L. (2024): LLM Intelligent Agent Tutoring in Higher Education Courses using a RAG Approach. DOI: https://doi.org/10.20944/preprints202407.0519.v1.

MYSQL (2024a): MySQL: MySQL 8.4 Reference Manual: 13.3.1 String Data Type Syntax. https://dev.mysql.com/doc/refman/8.4/en/string-type-syntax.html (Zugriff am 21.11.2024).

MYSQL (2024b): MySQL: MySQL 8.4 Reference Manual: 13.6 Data Type Default Values. https://dev.mysql.com/doc/refman/8.4/en/data-type-defaults.html (Zugriff am 21.11.2024).

OPENAI (o. J.): OpenAI Platform - Vector embeddings. https://platform.openai.com (Zugriff am 10.12.2024).

OPENAI, ACHIAM, J., ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L., ALMEIDA, D., ALTENSCHMIDT, J., ALTMAN, S., ANADKAT, S., AVILA, R., BABUSCHKIN, I., BALAJI, S., BALCOM, V., BALTESCU, P., BAO, H., BAVARIAN, M., BELGUM, J., BELLO, I., BERDINE, J., BERNADETT-SHAPIRO, G., BERNER, C., BOGDONOFF, L., BOIKO, O., BOYD, M., BRAKMAN, A.-L., BROCKMAN, G., BROOKS, T., BRUNDAGE, M., BUTTON, K., CAI, T., CAMPBELL, R., CANN, A., CAREY, B., CARLSON, C., CARMICHAEL, R., CHAN, B., CHANG, C., CHANTZIS, F., CHEN, D., CHEN, S., CHEN, R., CHEN, J., CHEN, M., CHESS, B., CHO, C., CHU, C., CHUNG, H. W., CUMMINGS, D., CURRIER, J., DAI, Y., DECAREAUX, C., DEGRY, T., DEUTSCH, N., DEVILLE, D., DHAR, A., DOHAN, D., DOWLING, S., DUNNING, S., ECOFFET, A., ELETI, A., ELOUNDOU, T., FARHI, D., FEDUS, L., FELIX, N., FISHMAN, S. P., FORTE, J., FULFORD, I., GAO, L., GEORGES, E., GIBSON, C., GOEL, V., GOGINENI, T., GOH, G., GONTIJO-LOPES, R., GORDON, J., GRAFSTEIN, M., GRAY, S., GREENE, R., GROSS, J., GU, S. S., GUO, Y., HALLACY, C., HAN, J., HARRIS, J., HE, Y., HEATON, M., HEIDECKE, J., HESSE, C., HICKEY, A., HICKEY, W., HOESCHELE, P., HOUGHTON, B., HSU, K., HU, S., HU, X., HUIZINGA,

J., JAIN, S., JAIN, S., JANG, J., JIANG, A., JIANG, R., JIN, H., JIN, D., JOMOTO, S., JONN, B., JUN, H., KAFTAN, T., KAISER, Ł., KAMALI, A., KANITSCHEIDER, I., KESKAR, N. S., KHAN, T., KILPATRICK, L., KIM, J. W., KIM, C., KIM, Y., KIRCHNER, J. H., KIROS, J., KNIGHT, M., KOKOTAJLO, D., KONDRACIUK, Ł., KONDRICH, A., KONSTANTINIDIS, A., KOSIC, K., KRUEGER, G., KUO, V., LAMPE, M., LAN, I., LEE, T., LEIKE, J., LEUNG, J., LEVY, D., LI, C. M., LIM, R., LIN, M., LIN, S., LITWIN, M., LOPEZ, T., LOWE, R., LUE, P., MAKANJU, A., MALFACINI, K., MANNING, S., MARKOV, T., MARKOVSKI, Y., MARTIN, B., MAYER, K., MAYNE, A., MCGREW, B., MCKINNEY, S. M., MCLEAVEY, C., MCMILLAN, P., MCNEIL, J., MEDINA, D., MEHTA, A., MENICK, J., METZ, L., MISHCHENKO, A., MISHKIN, P., MONACO, V., MORIKAWA, E., MOSSING, D., MU, T., MURATI, M., MURK, O., MÉLY, D., NAIR, A., NAKANO, R., NAYAK, R., NEELAKANTAN, A., NGO, R., NOH, H., OUYANG, L., O'KEEFE, C., PACHOCKI, J., PAINO, A., PALERMO, J., PANTULIANO, A., PARASCANDOLO, G., PARISH, J., PARPARITA, E., PASSOS, A., PAVLOV, M., PENG, A., PERELMAN, A., PERES, F. de A. B., PETROV, M., PINTO, H. P. de O., MICHAEL, POKORNY, POKRASS, M., PONG, V. H., POWELL, T., POWER, A., POWER, B., PROEHL, E., PURI, R., RADFORD, A., RAE, J., RAMESH, A., RAYMOND, C., REAL, F., RIMBACH, K., ROSS, C., ROTSTED, B., ROUSSEZ, H., RYDER, N., SALTARELLI, M., SANDERS, T., SANTURKAR, S., SASTRY, G., SCHMIDT, H., SCHNURR, D., SCHULMAN, J., SELSAM, D., SHEPPARD, K., SHERBAKOV, T., SHIEH, J., SHOKER, S., SHYAM, P., SIDOR, S., SIGLER, E., SIMENS, M., SITKIN, J., SLAMA, K., SOHL, I., SOKOLOWSKY, B., SONG, Y., STAUDACHER, N., SUCH, F. P., SUMMERS, N., SUTSKEVER, I., TANG, J., TEZAK, N., THOMPSON, M. B., TILLET, P., TOOTOONCHIAN, A., TSENG, E., TUGGLE, P., TURLEY, N., TWOREK, J., URIBE, J. F. C., VALLONE, A., VIJAYVERGIYA, A., VOSS, C., WAINWRIGHT, C., WANG, J. J., WANG, A., WANG, B., WARD, J., WEI, J., WEINMANN, C. J., WELIHINDA, A., WELINDER, P., WENG, J., WENG, L., WIETHOFF, M., WILLNER, D., WINTER, C., WOLRICH, S., WONG, H., WORKMAN, L., WU, S., WU, J., WU, M., XIAO, K., XU, T., YOO, S., YU, K., YUAN, Q., ZAREMBA, W., ZELLERS, R., ZHANG, C., ZHANG, M., ZHAO, S., ZHENG, T., ZHUANG, J., ZHUK, W., ZOPH, B. (2024): GPT-4 Technical Report. http://arxiv.org/abs/2303.08774.

PENG, D., GUI, Z., WU, H. (2024): Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect. DOI: https://doi.org/10.48550/arXiv.2401.00422.

PERUCHINI, M., TEIXEIRA, J. M. (2024): Analyzing Large language models chatbots: An experimental approach using a probability test. DOI: https://doi.org/10.48550/arXiv.2407.12862.

PIEPMEYER, L. (2011): Grundkurs Datenbanksysteme: von den Konzepten bis zur Anwendungsentwicklung. München Wien: Hanser.

RADEVA, I., POPCHEV, I., DOUKOVSKA, L., DIMITROVA, M. (2024): Web Application for Retrieval-Augmented Generation: Implementation and Testing. Electronics 13 (7)1361. DOI: https://doi.org/10.3390/electronics13071361.

SAAD-FALCON, J., KHATTAB, O., POTTS, C., ZAHARIA, M. (2023): ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. http://arxiv.org/abs/2311.09476.

SCHICKER, E. (2017): Datenbanken und SQL. Wiesbaden: Springer Fachmedien Wiesbaden.

SETTY, S., THAKKAR, H., LEE, A., CHUNG, E., VIDRA, N. (2024): Improving Retrieval for RAG based Question Answering Models on Financial Documents. DOI: https://doi.org/10.48550/arXiv.2404.07221.

SHARABIANI, M., MAHANI, A., BOTTLE, A., SRINIVASAN, Y., ISSITT, R., STOICA, S. (2024): GenAI Exceeds Clinical Experts in Predicting Acute Kidney Injury following Paediatric Cardiopulmonary Bypass^2. DOI: https://doi.org/10.1101/2024.05.14.24307372.

SHARMA, S., YOON, D. S., DERNONCOURT, F., SULTANIA, D., BAGGA, K., ZHANG, M., BUI, T., KOTTE, V. (2024): Retrieval Augmented Generation for Domain-specific Question Answering. DOI: https://doi.org/10.48550/arXiv.2404.14760.

SHUSTER, K., POFF, S., CHEN, M., KIELA, D., WESTON, J. (2021): Retrieval Augmentation Reduces Hallucination in Conversation. http://arxiv.org/abs/2104.07567.

STRICKER, H.-P. (2024): Sprachmodelle verstehen: Chatbots und generative künstliche Intelligenz im Zusammenhang. Berlin, Heidelberg: Springer Berlin Heidelberg.

SUMATHI, S., ESAKKIRAJAN, S. (2007): Fundamentals of Relational Database Management Systems. 47 Band Berlin, Heidelberg: Springer Berlin Heidelberg.

TAIPALUS, T. (2024): Vector database management systems: Fundamental concepts, use-cases, and current challenges. Cognitive Systems Research 85101216. DOI: https://doi.org/10.1016/j.cogsys.2024.101216.

TOUVRON, H., MARTIN, L., STONE, K., ALBERT, P., ALMAHAIRI, A., BABAEI, Y., BASHLYKOV, N., BATRA, S., BHARGAVA, P., BHOSALE, S., BIKEL, D., BLECHER, L., FERRER, C. C., CHEN, M., CUCURULL, G., ESIOBU, D., FERNANDES, J., FU, J., FU, W., FULLER, B., GAO, C., GOSWAMI, V., GOYAL, N., HARTSHORN, A., HOSSEINI, S., HOU, R., INAN, H., KARDAS, M., KERKEZ, V., KHABSA, M., KLOUMANN, I., KORENEV, A., KOURA, P. S., LACHAUX, M.-A., LAVRIL, T., LEE, J., LISKOVICH, D., LU, Y., MAO, Y., MARTINET, X., MIHAYLOV, T., MISHRA, P., MOLYBOG, I., NIE, Y., POULTON, A., REIZENSTEIN, J., RUNGTA, R., SALADI, K., SCHELTEN, A., SILVA, R., SMITH, E. M., SUBRAMANIAN, R., TAN, X. E., TANG, B., TAYLOR, R., WILLIAMS, A., KUAN, J. X., XU, P., YAN, Z., ZAROV, I., ZHANG, Y., FAN, A., KAMBADUR, M., NARANG, S., RODRIGUEZ, A., STOJNIC, R., EDUNOV, S., SCIALOM, T. (2023): Llama 2: Open Foundation and Fine-Tuned Chat Models. DOI: https://doi.org/10.48550/arXiv.2307.09288.

UNTERSTEIN, M., MATTHIESSEN, G. (2012): Relationale Datenbanken und SQL in Theorie und Praxis. Berlin, Heidelberg: Springer Berlin Heidelberg.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., POLOSUKHIN, I. (2023): Attention Is All You Need. DOI: https://doi.org/10.48550/arXiv.1706.03762.

VEGA, M. de la (2023): Understanding OpenAI's "Temperature" and "Top_p" Parameters in Language Models. https://medium.com/@1511425435311/understanding-openais-temperature-and-top-p-parameters-in-language-models-d2066504684f (Zugriff am 11.12.2024).

VENKATESH, D., RAMAN, S. (2024): BITS Pilani at SemEval-2024 Task 1: Using text-embedding-3-large and LaBSE embeddings for Semantic Textual Relatedness. Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), 865–868. DOI: https://doi.org/10.18653/v1/2024.semeval-1.124.

WANG, X., WANG, Z., GAO, X., ZHANG, F., WU, Y., XU, Z., SHI, T., WANG, Z., LI, S., QIAN, Q., YIN, R., LV, C., ZHENG, X., HUANG, X. (2024): Searching for Best Practices in Retrieval-Augmented Generation. http://arxiv.org/abs/2407.01219.

WILKE, C. O. (2019): Fundamentals of Data Visualization. California, CA: O'Reilly Media.

YANG, Y., LI, X., JIN, H., HUANG, K. (2024): Advancing Structured Query Processing in Retrieval-Augmented Generation with Generative Semantic Integration. FCIS 9 (3)64–71. DOI: https://doi.org/10.54097/z309gx59.

YEPES, A. J., YOU, Y., MILCZEK, J., LAVERDE, S., LI, R. (2024): Financial Report Chunking for Effective Retrieval Augmented Generation. http://arxiv.org/abs/2402.05131.

YU, H., GAN, A., ZHANG, K., TONG, S., LIU, Q., LIU, Z. (2024): Evaluation of Retrieval-Augmented Generation: A Survey. http://arxiv.org/abs/2405.07437.

ZHAO, P., ZHANG, H., YU, Q., WANG, Z., GENG, Y., FU, F., YANG, L., ZHANG, W., CUI, B. (2024): Retrieval-Augmented Generation for AI-Generated Content: A Survey. http://arxiv.org/abs/2402.19473.

ZHAO, W. X., ZHOU, K., LI, J., TANG, T., WANG, X., HOU, Y., MIN, Y., ZHANG, B., ZHANG, J., DONG, Z., DU, Y., YANG, C., CHEN, Y., CHEN, Z., JIANG, J., REN, R., LI, Y., TANG, X., LIU, Z., LIU, P., NIE, J.-Y., WEN, J.-R. (2023): A Survey of Large Language Models. http://arxiv.org/abs/2303.18223.

ZHENG, L., CHIANG, W.-L., SHENG, Y., ZHUANG, S., WU, Z., ZHUANG, Y., LIN, Z., LI, Z., LI, D., XING, E. P., ZHANG, H., GONZALEZ, J. E., STOICA, I. (2023): Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. DOI: https://doi.org/10.48550/arXiv.2306.05685.

ZHENG, M., PEI, J., LOGESWARAN, L., LEE, M., JURGENS, D. (2024): When „A Helpful Assistant" Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models. DOI: https://doi.org/10.48550/arXiv.2311.10054.

Anhang 1: GitHub-Repository

Anmerkung: Der Anhang 1 zeigt die Verzeichnisstruktur des GitHub-Repositorys. Es sind drei Ordner dargestellt, welche einzelne Dateien enthalten. Diese Dateien werden in der Arbeit referenziert und sind hinter ihrem Namen alphabetisch in Klammern gekennzeichnet. Die Reihenfolge dieser Kennzeichnung in den Klammern entspricht der Abfolge, in der sie in dem Repository gespeichert sind. Für eine bessere Übersicht sind ausschließlich die Dateien abgebildet, die in der Arbeit als Referenz verwendet werden.

/code

- connect.py (**a**)
- statistics.py (**b**)

/data

- product_tabelle.sql (**c**)

/excel

- Durchlauf1_2024_11_07__13_04.xlsx (**d**)
- Durchlauf2_2024_11_07__13_08.xlsx (**e**)
- Durchlauf3_2024_11_07__13_09.xlsx (**f**)
- Ergebnisse.xlxs (**g**)

Der Anhang ist digital in diesem GitHub-Repository zu finden:

https://github.com/FabianS123/bachelorthesis_rag_structured_data

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

_____ _____

Ort, Datum Unterschrift