

# Konzeption und Implementierung eines Wissensmanagementsystems basierend auf Retrieval- Augmented Generation

BACHELORTHESIS ZUR ERLANGUNG DES AKADEMISCHEN GRADES „BACHELOR  
OF SCIENCE – IT- UND SOFTWARESYSTEME“

VON HARDICK, TIM GEBOREN AM 17.07.2001

Bearbeitungszeitraum: 24.05.2024 – 22.07.2024

Prüfer: Prof. Dr. Sebastian Bab

Zweitprüfer: Prof. Dr. Thomas Königsmann

Inhalt

Abstract .....	3
1 Einleitung.....	4
2 Grundlagen.....	5
2.1 Künstliche Intelligenz .....	5
2.2 Neuronale Netze .....	6
2.3 Generative Künstliche Intelligenz .....	6
2.4 Retrieval-Augmented-Generation .....	7
2.5 Pipeline-Architektur.....	9
2.6 Hürden von Sprachmodellen (als Beispiel ChatGPT) .....	9
2.7 Prompt Engineering .....	9
2.8 Daten.....	10
2.9 k-nearest-neighbor .....	10
2.10 Principal Component Analysis .....	11
2.11 Uniform Manifold Approximation and Projection .....	12
2.12 Vektordatenbank .....	13
2.13 Wissensmanagement.....	14
2.14 RAG im Vergleich zu Fine Tuning .....	15
2.15 Geschichte .....	17
2.16 Geschichte von RAG.....	17
3 Analyse und Technologieauswahl .....	18
3.1 LLM-App-Framework .....	18
3.2 Vektordatenbanksystem.....	19
3.3 Embedding-Modelle .....	19
3.4 LLM.....	20
3.5 Infrastruktur.....	21
3.6 User-Interface .....	21
3.7 RAG-Modellierung .....	22
3.8 Datenvorbereitung .....	24
3.9 Umsetzung und Anwendungsfälle .....	24
3.10 Übersicht.....	25
4 Erste Schritte .....	26
4.1.1 Erste Schritte mit LangChain .....	26
4.1.2 Erste Schritte mit Chunking .....	27
4.1.3 Erste Schritte mit Embedding.....	27
4.1.4 Erste Schritte mit Vektordatenbanken.....	28

4.1.5	Erster MVP.....	29
5	Umsetzung.....	30
5.1	Backend.....	30
5.1.1	Relationales Datenbanksystem .....	30
5.1.2	Vektordatenbanksystem .....	31
5.1.3	Config.....	31
5.1.4	Endpunkte .....	32
5.1.5	Core .....	34
5.2	Frontend .....	37
5.2.1	Grundlegender Aufbau.....	37
5.2.2	Styling .....	38
5.2.3	User .....	39
5.2.4	Config.....	39
5.2.5	Chat .....	40
5.2.6	Data .....	41
5.2.7	Info.....	43
5.3	Virtualisierung.....	43
5.3.1	Datenbanken .....	44
5.3.2	Anwendungen .....	44
5.3.3	Modelle.....	45
5.4	Tests.....	45
5.5	Anwendungsfall Versicherungsassistent .....	47
5.5.1	Fall 1 .....	48
5.5.2	Fall 2 .....	49
5.5.3	Fall 3 .....	51
6	Ausblick .....	53
6.1	Offene Punkte .....	53
6.1.1	Anwendungsfall Confluence-Suche.....	53
6.1.2	Änderungen am Quellcode.....	53
6.1.3	Modularität.....	54
6.2	Erkenntnisse.....	55
7	Fazit .....	56
8	Literaturverzeichnis.....	57

## Abkürzungsverzeichnis

KI (eng. AI)	Künstliche Intelligenz (eng. Artificial Intelligence)
GenAI	Generative Künstliche Intelligenz
RAG	Retrieval-Augmented Generation
LLM	Large language model (deut. großes Sprachmodell)
NLP	Natural language processing (deut. Natural language processing)
DB	Datenbank
DBMS	Datenbank-Managementsystem
MVP	Minimum viable product (deut. Minimales lebensfähiges Produkt)
ML	maschinelles lernen
JWT	JSON Web Token
VN	Versicherungsnehmer

## Abstract

Das Ziel dieser Arbeit ist die Modellierung und Implementierung eines KI basierendem Assistent-Systems für die Handhabung von internem Wissen. Konkrete Methodik, welche dabei genutzt werden soll, ist Retrieval-Augmented Generation. Im theoretischen Teil werden die grundlegenden damit verbundene Themen erklärt und eingeordnet. Der Modellierungsteil der Arbeit beschäftigt sich mit der Evaluation und Auswahl von notwendigen Technologien, die bei der Entwicklung verwendet werden sollen. Anschließend wird das zu implementierende Wissensmanagementsystem geplant und modelliert. Im Umsetzungsteil wird das vorangegangene geplante System umgesetzt und dabei anfallende Abweichungen beschrieben und begründet. Abschließend wird ein Fazit aus den gewonnenen Erkenntnissen, die während der Entwicklung angefallen sind, gezogen. Somit zeigt dieses Projekt die Machbarkeit und den damit einhergehenden Mehrwert für KI basierte Systeme.

The aim of this thesis is the modelling and implementation of an AI-based assistant system for the handling of internal knowledge. The concrete methodology to be used is Retrieval-Augmented Generation. In the theoretical part, the basic related topics are explained and categorised. The modelling part of the thesis deals with the evaluation and selection of necessary technologies to be used in the development. The knowledge management system to be implemented is then planned and modelled. In the implementation section, the previously planned system is realised and any deviations are described and justified. Finally, a conclusion is drawn from the knowledge gained during development. This project thus demonstrates the feasibility and the associated added value for AI-based systems. Translated with DeepL.com (free version)

## 1 Einleitung

Aktuell schreitet die Entwicklung im Gebiet der Künstlichen Intelligenz rasant voran. Da diese Technologie vor allem Vorteile im Bereich der Datenverwaltung und der Automatisierung mit sich bringt, wächst auch das Interesse von Unternehmen, diese Technologie zu implementieren. Die vorangegangene Projektarbeit zu dem Thema "Künstliche Intelligenz und der Einsatz in der Versicherungsbranche" hat sich zum einen mit den Grundlagen von Künstlicher Intelligenz beschäftigt und zum anderen die Potenziale und Risiken aufgezeigt, welche diese Technologie beim Einsatz in der Versicherungsbranche mit sich bringt.

Es hat sich gezeigt, dass es unterschiedliche Möglichkeiten gibt, KI in ein Unternehmen einzuführen und dabei einhergehende Risiken beachtet werden müssen. Der Einsatz und die Entwicklung von KI und damit verbundenen Systemen birgt vor allem Hürden im Bereich des Datenschutzes. Außerdem ist die Thematik noch jung und daher existiert wenig Erfahrung im Bezug auf Entwicklung und Wartung solcher Systeme. Heutzutage besitzen LLM eine Vielzahl an Einsatzmöglichkeiten, welche sich rund um das Thema Wissensmanagement einordnen lassen. Viele dieser Einsatzmöglichkeiten arbeiten mit unternehmensbezogenen und nicht personenbezogenen Daten, wodurch eine Umsetzung nicht allzu viele Datenschutzprobleme mit sich bringt. Aus dieser aktuellen Gegebenheit haben sich die VOLKSWOHL BUND Versicherungen dazu entschieden RAG (Retrieval-Augmented Generation) in Form eines Prototyps umzusetzen. Es soll ein Assistent-System innerhalb des Unternehmens aufgebaut werden, welches die aktuellen Möglichkeiten von KI anhand einer praktischen Implementierung aufzeigt. Der Aufbau von Erfahrung und Wissen in Verbindung mit Künstlicher Intelligenz spielen eine zentrale Rolle in dieser Arbeit. Dieses Projekt zeigt die Machbarkeit für KI-basierte Systeme auf und soll zudem die bei der Umsetzung anfallenden Erkenntnisse dokumentieren. Außerdem wird durch das Präsentieren eines solchen Systems das Thema Künstliche Intelligenz entmystifiziert und die Benutzer können Erfahrung mit der Bedienung von KI-Systemen im Arbeitsleben sammeln. Zudem ist es leichter abschätzbar, ob ein System bzw. eine Technologie Mehrwert bietet, wenn diese in einem Prototyp implementiert ist.

Vor der eigentlichen Planung und Implementierung erläutert diese Arbeit die Grundlagen von RAG. Hier findet sich zum einen ein Teil über die Historie der Thematik und zum anderen werden alle Begriffe, Konzepte und Zusammenhänge erklärt. Generell soll dieser Teil als Grundlage für den Hauptteil der Arbeit dienen und dazu beitragen, ein umfassenderes Verständnis des Themas zu erhalten.

Der Hauptteil befasst sich mit der Entwicklung eines Prototypen-Systems, konkret wird sich mit der Planung, der Umsetzung und den dabei auftretenden Erkenntnisgewinnen auseinandergesetzt. Um ein besseres Verständnis über die mögliche Umsetzung des Projektes zu bekommen, wird sich in der Planungsphase mit der notwendigen Infrastruktur und den Technologien beschäftigt. So wird für alle notwendigen Bestandteile ein kleinerer Vergleich durchgeführt, um die Auswahl von Datenbanksystem, Sprachmodell und der generellen Systemumgebung begründen zu können. Die Umsetzung behandelt alle während der Entwicklung auftretenden Unregelmäßigkeiten sowie auch mögliche Abweichungen zu Entscheidungen, die in der Planungsphase getroffen worden sind. Außerdem werden Erkenntnisgewinne, die bei der Entwicklung und Implementierung anfallen werden, aufgeführt.

Beendet wird diese Arbeit mit dem Durchlauf des implementierten Systems für vorher definierte Anwendungsfälle. Der Fokus liegt hierbei darauf, ob der Prototyp bereits für bestimmte Anwendungsfällen nutzbar ist oder ob dieser noch weit von einer theoretischen Inbetriebnahme entfernt ist. Abschließend wird ein Ausblick zu noch offenen Punkten und weiteren möglichen Verbesserungen gegeben.

## 2 Grundlagen

Um das Verständnis für die Grundthematik und die Zugänglichkeit dieser Arbeit zu steigern, wurde in den nachfolgenden Teilkapiteln aus der Projektarbeit „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche“ (Hardick, 2024) zitiert. Hierbei handelt es sich teilweise um gesamte Teilkapitel, jedoch auch um gekürzte Abschnitte, um sich auf die wesentlichen Grundlagen für das Thema „Konzeption und Implementierung eines Wissensmanagementsystems basierend auf Retrieval-Augmented Generation“ zu konzentrieren.

### 2.1 Künstliche Intelligenz

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche - 2.4 Künstliche Intelligenz“:

Mit Künstlicher Intelligenz wird ein sehr breites Forschungsfeld der Informatik bezeichnet. Dabei fällt auf, dass heutzutage selbst in unserem Alltag viele Technologien im Einsatz sind, welche zu der Künstlichen Intelligenz zählen. Zu nennen wären da persönliche Assistenten wie Alexa, Siri oder Google Home, welche per Spracheingabe bedient werden können, um Termine in den Kalender einzutragen, Einkauflisten zu erstellen oder Smart Home Systeme zu bedienen. Übersetzungssoftware wie Google Translate oder DeepL werden mit Künstlicher Intelligenz betrieben, um die Sprachverarbeitung zu ermöglichen. Auch im Bereich der Bilderkennung wird Künstliche Intelligenz eingesetzt, so ist eine im Alltag anzutreffende Einsatzmöglichkeit die Gesichtserkennung, um Smartphones zu entsperren. Zudem werden Empfehlungsalgorithmen auch mit Künstlicher Intelligenz betrieben, diese sind heutzutage auf Netflix, Spotify und Amazon anzufinden. Des Weiteren nimmt die Bedeutung von Robotern zu, sodass der Einsatz sich nicht mehr nur auf die Manufaktur und Logistik beschränkt. So wird der Einsatz von Service-Robotern, welche auf Künstlicher Intelligenz zurückgreifen, vor allem in der Gastronomie und Pflege ausgebaut. Auch autonomes Fahren wird durch Künstliche Intelligenz betrieben und hat in den letzten Jahren rasante Fortschritte gemacht.

Eine allgemeingültige Definition vom Begriff „Künstlicher Intelligenz“ ist jedoch nicht vorhanden. Vor allem durch die breite Anzahl an unterschiedlichen Methoden, um Künstliche Intelligenz zu realisieren, wird klar, dass es nicht gerade leicht ist eine klare Beschreibung zu finden. Um ein besseres Gefühl dafür zu bekommen, schauen wir uns einige Definitionen des Themas an.

Künstliche Intelligenz ist ein Teilgebiet der Informatik. Sie imitiert menschliche kognitive Fähigkeiten, indem sie Informationen aus Eingabedaten erkennt und sortiert. Diese Intelligenz kann auf programmierten Abläufen basieren oder durch maschinelles Lernen erzeugt werden.“ (Def. KI - Fraunhofer, 2023)

Fraunhofer beschreibt Künstlicher Intelligenz als Forschungsgebiet, welches versucht, menschliche Fähigkeiten zu imitieren. KI beschäftigt sich mit der Frage, wie man mit intelligentem Verhalten Probleme lösen kann. Das dabei gewonnene Wissen wird dann dazu genutzt Systeme zu entwickeln, welche automatisiert Lösungen für Probleme finden sollen. Damit bezeichnet Künstliche Intelligenz die Kapazität einer Maschine, um Aufgaben auszuführen, für die kognitive Fähigkeiten notwendig sind. Diese Aufgaben umfassen unter anderem das Lernen, das Schlussfolgern, das Lösen von Problemen sowie die Entscheidungsfindung.

Wenn man in diesem Zusammenhang von der Entwicklung von Künstlicher Intelligenz spricht, meint man die Entwicklung von Software. Dabei werden Suchalgorithmen, Lernalgorithmen und stochastische Verfahren, häufig auch kombiniert, verwendet. Die grundlegenden Methoden, auf die bei der Entwicklung von Künstlicher Intelligenz zurückgegriffen wird, lässt sich den Bereichen der Logik, der linearen Algebra, der Graphentheorie und der Stochastik zuordnen. Für die Umsetzung bedeutet das, dass diese simulierte Intelligenz auf regelbasierten Abläufen basieren kann oder durch Verfahren erzeugt wird, bei denen Algorithmen nicht explizit programmiert werden. Ein zentrales Ziel der KI-Forschung ist es, Algorithmen und Modelle zu entwickeln, die es Computern ermöglichen, Wissen zu extrahieren, Muster zu erkennen und Schlussfolgerungen zu ziehen.

## 2.2 Neuronale Netze

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche - 2.8 Neuronale Netze“:

Bei Künstlichen Neuronalen Netzen handelt es sich um eine Methodik, um Künstliche Intelligenz zu realisieren, dabei ist die Methode von der Struktur des menschlichen Gehirns inspiriert. Sie stellen ein grundlegendes Modell im Gebiet von modernen KI-Systemen dar. Ein solches Künstliches Neuronales Netz besteht dabei aus vielen, in Software umgesetzten Knoten, welche von biologischen Neuronen inspiriert sind. Die künstlichen Neuronen sind mit Neuronen aus der nächsten Schicht verbunden, diese Verbindungen besitzen dabei Gewichtungen. Im Laufe des Trainingsprozesses werden die Gewichtungen angepasst, um die Genauigkeit des Neuronalen Netzes, ein bestimmtes Problem zu lösen, zu erhöhen. Jedes künstliche Neuron verarbeitet die eingehende Werte, in dem es die eingehenden Werte mit den Gewichtungen verrechnet, zusätzlich wird noch ein Wert, der sogenannte Bias, mit eingerechnet. Der daraus resultierende Wert wird anschließend von einer Aktivierungsfunktion verarbeitet und als Ausgabe des künstlichen Neurons verwendet.

## 2.3 Generative Künstliche Intelligenz

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 3 Generative Künstliche Intelligenz“:

Generative Künstliche Intelligenz ist ein Sammelbegriff für KI-Systeme, die darauf ausgerichtet sind, neue, einzigartige Inhalte, auf Basis der in der Trainingsphase genutzten Daten, zu generieren. Sie stellt einen Gegensatz zu diskriminativen KI dar, welche bei der Mustererkennung und Datenanalyse zum Einsatz kommt. Dadurch, dass beim Lernprozess die Struktur des Inhalts verstanden werden soll, ist das damit verbundene Ziel von generativer KI neue kreative und realistische Inhalte generieren zu können, welche sich an die Struktur des gelernten halten. Technisch gesehen ist ein generatives Neuronales Netz ein statistisches Modell, um die wahrscheinlichste Ausgabe auf Basis einer Eingabe zu generieren. Die Interaktion mit einer generativen KI wird dabei mittels einer Eingabe vorgenommen, dem sogenannten Prompt.

Der Prompt soll dabei das zu generierende Ergebnis beschreiben und wird vom statistischen Modell verwendet, um das wahrscheinlichste Ergebnis zu produzieren.

Generative KI besitzt das Potenzial, verschiedene Branchen zu transformieren, indem sie innovative Wege für die Content-Erstellung aufzeigt. Trotz dieser vielversprechenden Entwicklungen gibt es gleichzeitig berechtigte Bedenken bezüglich möglicher Missbrauchsszenarien von generativer KI, speziell im Kontext der Erzeugung gefälschter Inhalte oder der Produktion von Deepfake-Videos. Daher ist es von großer Bedeutung, die Risiken und Herausforderungen, die mit dieser Technologie verbunden sind, sorgfältig zu analysieren und anzugehen, um eine sichere und verantwortungsbewusste Nutzung zu gewährleisten.

### 2.4 Retrieval-Augmented-Generation

Retrieval-Augmented Generation ist ein Ansatz, welcher Aspekte der Informationssuche mit der KI basierten Generierung kombiniert, mit dem Ziel, hochwertige Inhalte zu erzeugen. Beweggründe für das Aufkommen von RAG ist das Anreichern des allgemeinen Wissensgrundsatzes eines LLMs mit zusätzlichen Informationen. Vor allem sollen dadurch genauere Antworten mit einem höherem Kontextbezug generiert werden, welche dabei eine geringere Rate an Halluzinationen aufweisen.

Notwendig ist eine Datenbank, welche als grundlegender Wissensspeicher des Systems fungiert. Die Inhalte dieser Datenbank unterscheiden sich dabei je nach Anwendungsfall, das könnte unternehmensinternes Wissen, Fachliteratur oder E-Mail-Logs sein. Bei einer Anfrage an das System wird in einem ersten Schritt in der Datenbank nach relevanten Daten gesucht und die Anfrage damit angereichert. Hierbei ist das Ziel die Anfrage möglichst kontextbezogen mit weiteren Daten anzureichern. Anschließend wird eine generative-KI eine Antwort auf die Anfrage erstellen. Durch die Datenanreicherung kann den Anforderungen der eigentlichen Anfrage besser gerecht werden, da ein RAG die Vorteile von vorhandenem Wissen und kreativer Sprachgenerierung kombiniert. Es werden relevante Inhalte generiert, welche auf fundiertem Wissen und auf der kontextuellen Anpassung basiert.

Aktuell werden moderne Sprachmodelle mit riesigen Datenmengen trainiert. So ist es möglich ein breites Spektrum an allgemeinem Wissen in den Gewichtungungen des neuronalen Netzes zu speichern. Falls das Sprachmodell jedoch angewiesen wird etwas zu generieren, wofür Wissen notwendig ist, welches nicht in den Trainingsdaten enthalten war, wird die Ausgabe Ungenauigkeiten und Halluzinationen enthalten oder kann schlichtweg nicht beantwortet werden. Daher versucht RAG das Allgemeine Wissen eines LLM mit zusätzlichem Kontext anzureichern. So zeigt die folgende Abbildung eine Anfrage, welche das LLM, in diesem Fall GPT-4, nicht korrekt beantwortet.



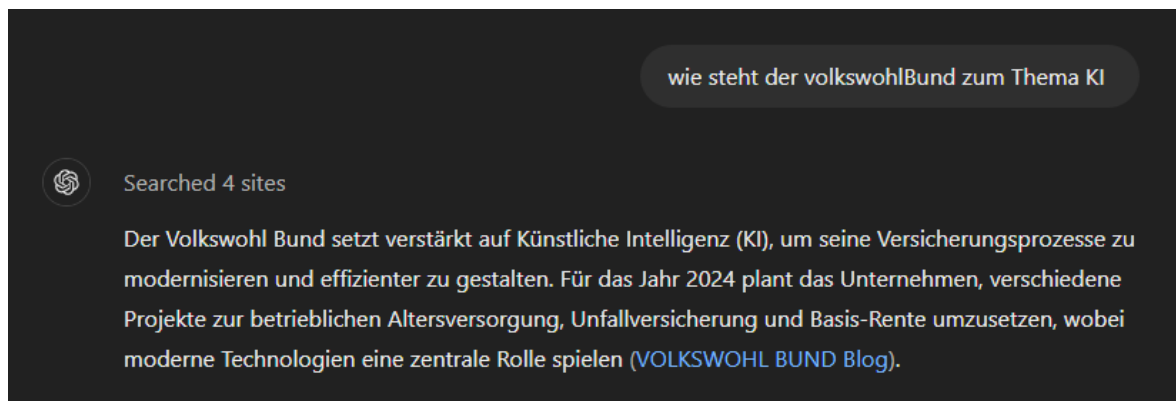


Abbildung 1 Beispiel von Halluzinationen

Seither werden Neuronale Netze durch Feinabstimmung des Modells an domänenspezifische oder proprietäre Informationen angepasst. Diese Technik ist zwar effektiv, aber auch aufwändig rechenintensiv und erfordert technisches Fachwissen, so dass das Neuronale Netz sich weniger flexibel an sich verändernde Informationen anpassen lässt. Bei Retrieval-Augmented Generation arbeitet ein Retriever-Teil und ein Generativer-Teil so zusammen, dass aus externen Wissensquellen, dem LLM, Informationen bereitgestellt werden können, welche zudem leichter aktualisierbar sind.

Ein Vergleich des Kaggle-Nutzers jjinho, welcher RAG bildhaft erklärt, lautet: RAG ist für LLMs das, was eine Open-Book-Klausur für einen Studenten ist. Bei einer Open-Book-Klausur haben Studenten Zugang zu Notizen oder Lehrbücher, mit relevanten Informationen für die Beantwortung der Fragen.<sup>1</sup>

Ein RAG-System besteht somit aus zwei Wissensquellen. Zum einen das parametrische Wissen, welches beim Trainingsprozess erlernt wurde und implizit in den Gewichtungen des NN wiederzufinden ist. Und zum anderen das externe Wissen, welches in den meisten Implementierungen in einer Vektordatenbank persistiert wird.

Der Name Retrieval-Augmented Generation beschreibt zudem grob den Ablauf von sich selbst.<sup>2</sup>

**Retrieval (deut. Abruf):** Die Benutzeranfrage wird verwendet, um relevanten Kontext aus einer externen Wissensquelle abzurufen. Dazu wird die Benutzeranfrage mit einem Embedding-Model (deut. Einbettungsmodell) in denselben Vektorraum eingebettet wie das zusätzliche Wissen in der Vektordatenbank. Dies ermöglicht eine Ähnlichkeitssuche, bei der die am nächsten liegenden Datenobjekte aus der Vektordatenbank zurückgegeben werden.

**Augmented (deut. Erweitern):** Die Benutzeranfrage und der ermittelte, zusätzliche Kontext werden in ein Prompt-Template eingefügt.

**Generation (deut. Erzeugen):** Schließlich wird der angereicherte Prompt an das LLM weitergeleitet, welches eine Antwort generiert.

<sup>1</sup> (jjinho, 2023)

<sup>2</sup> (Monigatti, Retrieval-Augmented Generation (RAG): From Theory to LangChain Implementation, 2023)

## 2.5 Pipeline-Architektur

Die Pipeline-Architektur, auch als Pipes-and-Filter-Architektur bekannt, ist ein Software-Designmuster, welches aus Filtern, den separaten Verarbeitungsschritten, und Pipes, den Verbindungen zwischen den Verarbeitungsschritten, besteht. Jeder Verarbeitungsschritt nimmt Eingabedaten entgegen, verarbeitet diese und gibt die Ausgabe weiter. Die Daten fließen dabei linear durch die Pipeline, wodurch eine strukturierte Verarbeitung von Datenströmen ermöglicht wird. Daher ist es nicht ungewöhnlich, dass diese Architektur bei Daten- und Signalverarbeitungen eingesetzt wird und daher auch im KI-Kontext anzutreffen ist. Da jeder Filter eine spezifische Aufgabe erfüllt, können einzelne Filter leicht ausgetauscht werden, daraus folgt ein hoher Grad an Modularität. Zudem ist die Wiederverwendbarkeit sowie Erweiterbarkeit von Filtern gegeben, da durch den simplen Aufbau ein hohes Maß an Flexibilität erreicht wird.<sup>3</sup>

## 2.6 Hürden von Sprachmodellen (als Beispiel ChatGPT)

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 3.4 ChatGPT“:

Bei der Verwendung von ChatGPT können unvorhersehbare und oft realitätsferne Ergebnisse auftreten. Insbesondere, wenn es an ausreichenden Informationen im Trainingsmaterial für bestimmte Fragen, vor allem zu Nischenthemen, mangelt. In solchen Fällen, auch als Closed Domains bezeichnet, neigt GPT dazu, Fakten zu erfinden, was als Halluzination bezeichnet wird. Daher ist es entscheidend, die von GPT generierten Texte auf Wahrheitsgehalt und Richtigkeit zu überprüfen, bevor sie verwendet werden. GPT sollte vorrangig für Brainstorming und als Inspirationsquelle genutzt werden, wobei die finale Verantwortung und Intelligenz nicht auf GPT delegiert werden sollten.

## 2.7 Prompt Engineering

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 3.5 Prompt Engineering“:

Ein Prompt ist eine Eingabe an ein generatives KI-System und hat den Zweck, dem System zu beschreiben, was es Generieren soll. Man kann es als die Schnittstelle ansehen, um mit der KI-Anwendung zu kommunizieren. Dazugehörend beschäftigt sich das Prompt Engineering mit dem Verfassen von Prompts, um möglichst qualitativ hochwertige und auf den Anwendungsfall passende Ausgaben zu erzeugen. Ziel dieser Disziplin ist es also, die Anweisungen an ein System so zu gestalten, dass eine nützliche Antwort generiert wird, um eine effektive Interaktion zu ermöglichen.

Allgemein kann man durch das reine Vergleichen von Prompts keine Aussage darüber treffen, welcher besser oder schlechter ist. Da immer eine Ausgabe generiert werden soll, sollte man die Qualität der Ausgabe berücksichtigen, falls man die Qualität von Prompts vergleichen will. Generell sollte man bestimmte Grundprinzipien bei der Erstellung von Prompts beachten. So sollte eine möglichst klare und präzise Anweisung gemacht werden. Bei einem guten Prompt werden viele Informationen angegeben, dazu gehören unter anderem folgende Informationen: Anweisung, Kontext, Stil, Format, Rolle, Adressat, Struktur und Beispiele.

---

<sup>3</sup> (Vollmer, 2022)

## 2.8 Daten

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 5.6 Daten“:

Die Thematik Künstliche Intelligenz ist vor allem in den letzten Jahren sehr stark gewachsen. Dies ist zum einen durch die leistungsstarke Hardware und Software zu erklären, aber vor allem spielt der Zugang zu riesigen Datenmengen eine entscheidende Rolle dabei. So benötigen vor allem Neuronale Netze bei den Deep Learning Verfahren eine große Datenmenge, um zufriedenstellende Ergebnisse zu liefern. In den letzten Jahren konnte ein enormes Wachstum des aufkommenden Datenvolumens beobachtet werden, diese Daten werden als Big Data bezeichnet. Bei der genaueren Betrachtung der Thematik stellen die fünf „V“s die Merkmale von Big Data dar.

1. "Volume" (deut. Menge) bezieht sich auf die Menge der verfügbaren Daten, welche durch eine zunehmend vernetzte Welt zu umfangreicheren Datenaufkommen führt.
2. "Velocity" (deut. Geschwindigkeit) beschreibt die Geschwindigkeit der Datengenerierung oder Aktualisierung, ermöglicht durch Echtzeiterfassung und automatisierte Prozesse.
3. "Variety" (deut. Vielfältigkeit) bezieht sich auf die Vielzahl interner und externer Datenquellen sowie unterschiedlicher Datenformate, wie strukturierte, semi- und unstrukturierte Daten.
4. "Veracity" (deut. Qualität) befasst sich mit der Qualität der Daten und Datenquellen, einschließlich Korrektheit, Vollständigkeit, Konsistenz, Aktualität und Vertrauenswürdigkeit, um den formalen Informationsgehalt sicherzustellen.
5. "Value" (deut. Wert) der Daten bezieht sich auf ihren eigentlichen Wert für die konkrete Nutzung.

Bei dem Arbeiten mit Künstlicher Intelligenz hat sich herausgestellt, dass die Umsetzung von KI-Projekten mit der Verfügbarkeit und damit verbundenen Handhabung der notwendigen Daten steht und fällt. Viele Unternehmen besitzen umfangreiche Datenbestände, die im Rahmen ihres Geschäftsbetriebs generiert wurden. Leider bleibt dieser als Small Data bezeichnete Datenschatz oft unbeachtet und ungenutzt. Daher liegt eine wichtige Aufgabe im Datenmanagement darin, die unternehmenseigenen Daten und Datenquellen systematisch zu erfassen. Eine weitere Herausforderung im Datenmanagement besteht darin zu überprüfen, für welche Prozesse und Einsatzfelder diese Daten am besten genutzt werden können.

## 2.9 k-nearest-neighbor

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche - 2.7.2.2 k-nearest-neighbor“:

Der k-nearest-neighbor-algorithm (deut. k-Nächster-Nachbar-Algorithmus) ist ein Verfahren, welches eine Gruppierung auf Grundlage von der Entfernung eines Datenpunktes zu anderen, bereits klassifizierten, Datenpunkten durchführt. So wird angenommen, dass sich ähnliche Datenpunkte nah beieinander befinden.

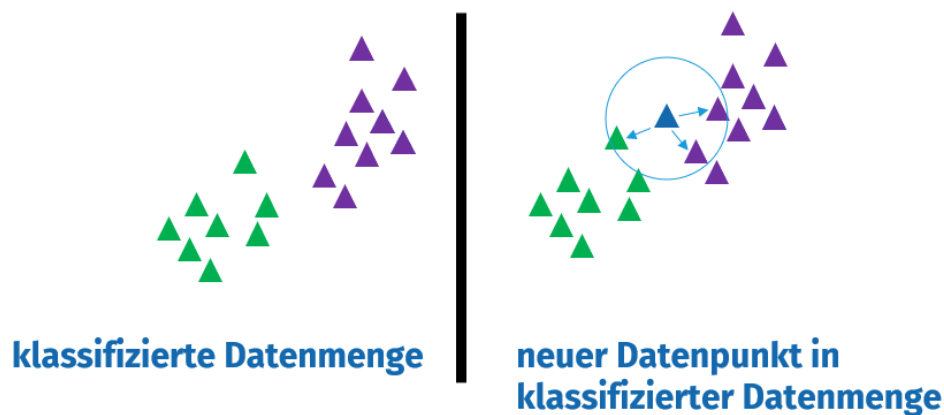


Abbildung 2 Klassifizierung eines Datenpunktes anhand der Entfernung zu klassifizierten Daten, eigene Abbildung

Der k-Wert legt fest, wie viele Entfernungen zu Nachbarn betrachtet werden, um die Gruppe des neuen Datenpunktes zu bestimmen. Ein zu kleiner k-Wert kann dazu führen, dass die Varianz der Ergebnisse erhöht wird und eine zu großer k-Wert kann zu einer geringen Varianz der Ergebnisse führen. Zudem ist es empfehlenswert eine ungerade Zahl für k zu wählen, damit ein „Unentschieden“ zu vermeiden. Zudem wird häufig ein Bereich von dem neuen Datenpunkt ausgehend definiert, um das Messen der Entfernung zu anderen Datenpunkten effizienter zu gestalten.

## 2.10 Principal Component Analysis

Principal Component Analysis (kurz PCA) ist eine Methode des maschinellen Lernens, welche angewendet werden kann, um die Dimension von hochdimensionalen Vektoren zu reduzieren. Diese Technik transformiert die ursprünglichen Daten, dabei wird eine Reduktion der Daten vorgenommen. Ziel ist dabei einen möglichst geringen Informationsverlust zu erzeugen. Der Kerngedanke von PCA ist, dass mehrere Daten in einem Datensatz dasselbe aussagen können, also eine Korrelation besteht.

Ein Beispiel für eine hohe Korrelation stellt eine Menge an Daten dar, welche die Schuhgröße und die Körpergröße von Menschen enthält. Da große Menschen häufig auch größere Schuhe tragen, würde durch das Entfernen der Schuhgröße aus der Datenmenge der Informationsgehalt nur minimal verringert werden.

Ziel ist es, die Dimensionen zu weniger sogenannten Hauptkomponenten zusammenfassen, ohne dass der Informationsgehalt des Datensatzes signifikant darunter leidet. Hauptkomponenten sind dabei Richtungen im Datenraum, die die maximale Varianz der Daten erklären. Die Schritte zur Durchführung von PCA sind wie folgt:

1. Zunächst werden die Daten zentriert, indem der Mittelwert jeder Variablen von den Datenpunkten subtrahiert wird. Dies stellt sicher, dass der neue Ursprung des Datensatzes der Mittelwert der ursprünglichen Daten ist. Anschließend wird die Kovarianzmatrix der zentrierten Daten berechnet, diese beschreibt, wie die Variablen zusammen variieren.

2. Der nächste Schritt beinhaltet die Eigenwertzerlegung der Kovarianzmatrix. Hierbei werden die Eigenwerte und Eigenvektoren der Matrix berechnet. Die Eigenvektoren geben die Richtungen der Hauptkomponenten an, während die Eigenwerte die Varianz entlang dieser Richtungen darstellen. Die Hauptkomponenten werden dann in absteigender Reihenfolge ihrer Eigenwerte sortiert, wobei die ersten Hauptkomponenten, die größte Varianz der Daten enthalten.
3. Für die Reduktion der Dimensionen werden die ersten Hauptkomponenten ausgewählt, die einen Großteil der Varianz beibehalten. Schließlich werden die ursprünglichen Daten auf diese ausgewählten Hauptkomponenten projiziert, was zu einem neuen Datensatz mit reduzierter Dimension führt.

Der Hauptvorteil von PCA liegt in der Fähigkeit, die Dimension der Daten erheblich zu reduzieren, während die wichtigsten Informationen erhalten bleiben. Dies erleichtert die Analyse und Visualisierung von Daten, die in ihrem ursprünglichen hochdimensionalen Zustand schwer zu handhaben wären. Für die Datenvisualisierung ermöglicht PCA die Darstellung komplexer Datensätze in zwei oder drei Dimensionen, wodurch Muster und Zusammengehörigkeit von Daten leichter erkennbar werden.<sup>4</sup>

### 2.11 Uniform Manifold Approximation and Projection

Uniform Manifold Approximation and Projection (kurz UMAP) ist eine nicht-lineare Dimensionsreduktionsmethode, die besonders effektiv ist, um hochdimensionale Vektoren auf eine niedrigere Dimension zu projizieren, während die Struktur und die Abstände in den Daten erhalten bleiben. UMAP basiert auf topologischen und geometrischen Prinzipien und bietet eine leistungsfähige Alternative zu anderen Dimensionsreduktionsmethoden wie PCA. Das grobe Vorgehen von UMAP lässt sich in den folgenden Schritten zusammenfassen:

1. Zunächst wird versucht die Mannigfaltigkeit der Datenmenge zu identifizieren
2. Modellierung der Datenstruktur mittels Punkte, Linien und Dreiecke zur Bildung einer unscharfen topologischen Struktur
3. K-NN-Algorithmus wird angewandt, um für jeden Punkt den nächsten Nachbarn zu finden und durch gewichtete Kanten zu verbinden
4. Distanzfunktion wird angewandt, um Ähnlichkeiten zwischen hoch- und niedrigdimensionalen Punkten zu messen
5. Optimierung der Punktpositionen im niedrigdimensionalen Raum durch Minimierung der Distanzdivergenzen

Zusammenfassend ist UMAP eine leistungsstarke Methode zur Dimensionsreduktion, die es ermöglicht, die komplexe Struktur hochdimensionaler Daten auf eine niedrigere Dimension zu projizieren, während die wesentlichen Muster und Beziehungen erhalten bleiben. Die Dimensionsreduktion erleichtert die Interpretation und Visualisierung der Daten erheblich.<sup>5</sup>

---

<sup>4</sup> (Jaadi, 2024)

<sup>5</sup> (J.D., 2023)

## 2.12 Vektordatenbank

Vektordatenbanken gab es schon, bevor LLM die heutige Relevanz erlangt haben und wurden ursprünglich in Empfehlungssystemen eingesetzt, da schnell ähnliche Objekte für eine bestimmte Anfrage bezogen werden können. Im Kontext von KI und LLMs fungieren Vektordatenbanken als Langzeitgedächtnis für LLMs und werden zunehmend innerhalb von LLM-Anwendungen, wie z.B. RAG-Implementierungen, eingesetzt.

Eine Vektordatenbank ist ein Datenbanktyp, welcher unstrukturierte Datenobjekte in Vektoreinbettungen, mehrdimensionale Vektoren, speichert und verwaltet. Durch die Vektorrepräsentation sollen das Auffinden und das Abrufen von ähnlichen Datenobjekten leichter und performanter gestaltet werden als bei herkömmlichen Datenbanktypen. Diese Datenobjekte können von unterschiedlicher Natur sein, hinreichend von Text- über Bild- bis hin zu Audio-Daten.

Die Art der gespeicherten Daten beeinflusst auch, wie die Daten abgerufen werden. In relationalen Datenbanken basieren die Abfrageergebnisse auf Übereinstimmungen mit bestimmten Schlüsselwörtern, da die Daten dort in strukturierter Form gespeichert werden. Relationale Datenbanken sind daher auch in Tabellenform darstellbar. Hier zeigt sich jedoch auch die Einschränkung von relationalen Datenbanken, da bereits bei der Modellierung der Struktur alle für Abfragen relevanten Informationen berücksichtigt werden müssen.

Als Beispiel dient eine Bücherverwaltung, welche Bücher in einer relationalen Datenbank persistiert. Dazu werden Attribute wie Titel, Autor und das Genre gespeichert. Dadurch kann im System nach allen Büchern eines bestimmten Genres gesucht werden oder die Bücher können, nach Titel, Alphabetisch sortiert ausgegeben werden. Falls jedoch alle Bücher aufgezeigt werden sollen, in denen die Handlung in Amerika spielt, gibt es dafür keine direkte Möglichkeit. Hierfür müssten die Datensätze mit einem Attribut z.B. Handlungsort erweitert werden, jedes Buch müsste ausgewertet werden, um das neue Attribut entsprechend setzen zu können. Dies ist je nach Komplexität und Menge der Daten zeit- und ressourcenaufwändig.

In Vektor-Datenbanken basieren die Abfrageergebnisse auf Ähnlichkeit. Vektor-Datenbanken speichern ihre Daten in einer unstrukturierten Form. Konkret werden die Datenobjekte durch ML in eine numerische Darstellung umgewandelt, wobei die Informationen des Datenobjektes erhalten bleiben. Diese numerische Darstellung wird als vector-embedding (deut. Vektor-Einbettung) bezeichnet und besitzt eine Vielzahl an Dimensionen.

Durch die Vektor-Einbettung ist es nun möglich, mathematische Berechnungen auf Datenobjekten durchzuführen, die sich normalerweise nicht für Berechnungen eignen würden. So können die Abstände zwischen den Einbettungen berechnet werden, um eine Art Ähnlichkeitswert zu erhalten. Je näher zwei eingebettete Objekte beieinander liegen, desto ähnlicher sind sie sich.

Eine Vektor-Datenbanken ist in der Lage, ähnliche Objekte einer Abfrage schnell aufzufinden, da sie diese durch die Einbettungen bereits vorberechnet hat. Die Ähnlichkeitssuche zwischen einer Abfrage und den eingebetteten Datenobjekten kann mithilfe eines k-nearest neighbors-Algorithmus durchgeführt werden. Nachteil hier ist jedoch, dass die Suche deutlich zeitaufwändiger wird, je größer die Anzahl an Datenobjekten ist.

Als Beispiel könnte folgende Operation mit den Vektoren durchgeführt werden:

$\text{Vektor}(\text{Berlin}) - \text{Vektor}(\text{Deutschland}) + \text{Vektor}(\text{Frankreich}) = \text{Vektor}(\text{Paris})$

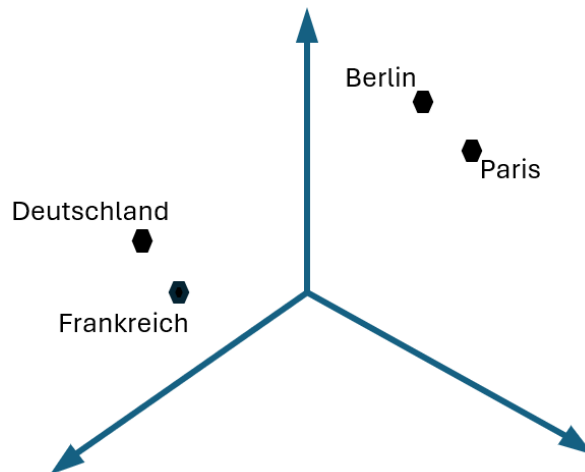


Abbildung 3 Beispielhafte Abbildung eines Vektorraumes, eigene Abbildung

Dadurch kann im Vektorraum der Unterscheid zwischen einer Hauptstadt und seinem zugehörigen Land wiederum durch einen Vektor dargestellt werden. So ist es möglich, durch diesen Vektor die Hauptstadt eines anderen Landes herauszufinden, wenn entsprechend die Vektordarstellung des Landes vorhanden ist.

Das genutzte, zugrundeliegende Konzept der Ähnlichkeitssuche in einer modernen Vektordatenbank baut auf den kNN-Algorithmus auf und nennt sich ANN-Suche (Approximate Nearest Neighbor: deut. Ungefährer nächster Nachbar), bei den verschiedenen Algorithmen zur Indizierung und Berechnung von Ähnlichkeiten verwendet werden.

Genauer gesagt wird auf eine Indizierung der Vektor-Einbettungen zurückgegriffen, um die Vorteile von schnelleren Suchen nutzen zu können. Dafür werden die Vektoren auf eine Datenstruktur abgebildet. Um aus den indizierten Vektoren die nächstgelegenen Nachbarn der Abfrage zu finden, wendet eine Vektordatenbank eine Berechnung zur Ähnlichkeitsermittlung an. Zu den gängigen Methoden gehören Kosinus-Ähnlichkeit, Skalarprodukt, Euklidischer Abstand, Manhattan-Metrik und Hamming-Abstand. Auf die genauere Funktionsweise der mathematischen Methoden in Bezug auf die ANN-Suche wird hier aus Komplexitätsgründen verzichtet.<sup>6</sup>

### 2.13 Wissensmanagement

Prof. Dr. Michael Müller, welcher sich mit dem Lehrgebiet „Organisation und Management“, insbesondere „Wissensmanagement“ beschäftigt, definiert den Begriff „Wissensmanagement“ folgendermaßen:

„Planung, Durchführung, Controlling und Weiterentwicklung aller auf die Ressource Wissen bezogenen Prozesse in Organisationen und Gesellschaften (aus der Sicht von Technik, Organisation und Mensch).“ (Müller, 2022, S. 25)<sup>7</sup>

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 4.5.2 Wissensmanagement“:

---

<sup>6</sup> (Monigatti, 2023)

<sup>7</sup> (Müller, 2022)

Eine mögliche Implementierung von KI in ein Versicherungsunternehmen, stellt das Wissensmanagement dar. Hierbei handelt es sich um ein System, welches sich mit der Verwaltung von geschäftsinernen Wissen befasst.

Dabei handelt es sich generell um Informationen über das Unternehmen und deren Organisation, wie auch um Prozessbeschreibungen und fachlichen Beschreibungen, wie Bedingungen für eine Versicherung. Dieses System hat als Ziel, den Mitarbeitern eine einfache und schnelle Möglichkeit zu bieten, sich über unternehmensbezogene Themen zu informieren.

Für diesen Einsatzzweck eignet sich der Einsatz von Large-Language-Modellen, so wie ein Aufbau nach dem Vorbild von ChatGPT. Konkret bedeutet das, ein System zu schaffen, welches dem Nutzer eine freie Gestaltung von Eingaben ermöglicht, mit dem Ziel in einer konversationsartigen Interaktion die Fragen des Nutzers zufriedenstellend zu beantworten. Bei einem mehr oder weniger direkten Einsatz von Large-Language-Modells, für diesen Anwendungsfall, kommt es zu einigen Problemen. So ist ein LLM nach dem Trainingsprozess statisch, das heißt, dass die Informationen nicht aktuell sein müssen. Des Weiteren werden die größten Modelle auf allgemeines Wissen trainiert, was bei der Abfrage nach spezifischen Wissen zu Problemen führen kann. Allgemein ist die Nachvollziehbarkeit, wie ein Large-Language-Model zu einem Ergebnis kommt, nicht transparent. Zum Beispiel ist nicht direkt erkennbar, aus welchen Quellen die Informationen bezogen werden. Weitere Herausforderungen, die mit dem Einsatz von Generativer-AI verbunden sind, haben mit den finanziellen und personellen Ressourcen zu tun.

Die W&W-Gruppe hat solch ein Wissensmanagement-System im Rahmen einer Sandbox-Umgebung entwickelt. Dabei wird auf GPT-3,5 zugegriffen, welches auf einer europäischen Instanz der Azure Cloud betrieben wird. Der eigenentwickelte Teil, sprich der Retriever, wie auch die dafür benötigte Datenhaltung, wurden ebenfalls in einer europäischen Instanz von Microsofts Cloud Dienst betrieben. Alternativ zu GPT können auch andere LLM angebunden werden, hierbei sind vor allem Open Source Modelle, wie Llama 2 oder Falcon, interessant, um den Betrieb auch theoretisch auf eigener Infrastruktur betreiben zu können. Des Weiteren ist für den erfolgreichen Betrieb eine aufwendige Datenaufbereitung notwendig. Die W&W-Gruppe betont die schnelle Umsetzung von KI-Prototypen, weist aber darauf hin, dass die Produktivsetzung vorallem duch Datenschutzthemen deutlich länger dauert.<sup>8</sup>

## 2.14 RAG im Vergleich zu Fine Tuning

Finetuning (deut. Feinabstimmung) bezeichnet einen Prozess, bei dem ein bereits vortrainiertes LLM an eine spezifische Domäne oder einen bestimmten Anwendungsfall angepasst wird. Es werden die Gewichte des grundlegenden NN auf Basis der zusätzlichen Trainingsdaten zugeschnitten, um so die Leistung des LLM zu verbessern.

Sowohl RAG als auch Finetuning können eingesetzt werden, um die Performance einer LLM-basierten Anwendung zu steigern, jedoch setzen die Methoden an unterschiedlichen Punkten dafür an.

---

<sup>8</sup> (GenAI Workshop, 2023)



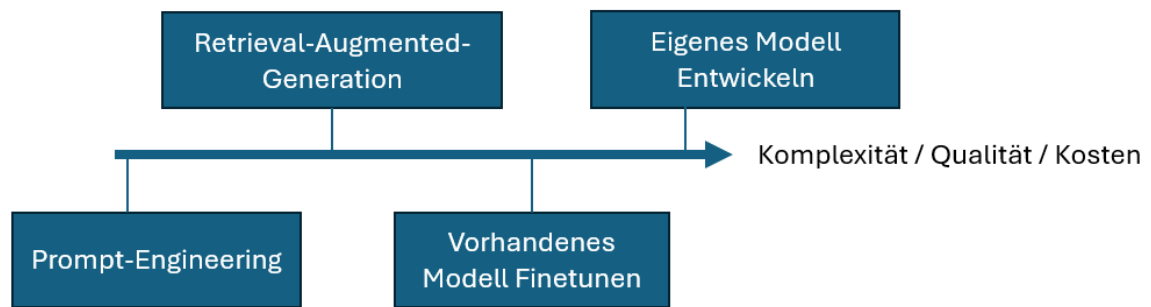


Abbildung 4 mögliche LLM-Optimierungen, eigene Abbildung

Als grobe Orientierung bietet RAG die geringere Komplexität bei niedrigeren Kosten. Finetuning ist hier komplexer und aufwändiger, kann dabei aber eine gute Qualitätsverbesserung mit sich bringen. Die höchste Qualität kann mit einem von Grund auf neu trainiertem System erreicht werden, jedoch ist dieser Vorgang auch sehr komplex und kostenintensiv. Es ist aber sehr wichtig zu beachten, dass RAG und Feinabstimmung nicht als Methoden betrachtet werden sollten, welche dasselbe Problem mit unterschiedlichem Aufwand lösen wollen. Es sind verschiedene Ansätze, um die Qualität einer LLM-Anwendung zu verbessern und haben je nach Anforderungen Vor- und Nachteile. Die Frage, ob RAG oder Finetuning genutzt werden sollte, ist ähnlich zu der Frage, ob ein Messer oder ein Löffel für das Essen genommen werden soll. Es kommt darauf an, was gegessen wird bzw. welche Anforderung erfüllt werden soll. Bei der Entscheidung zwischen Finetuning eines LLM oder der Verwendung von RAG ist eine der ersten Überlegungen, ob die Anwendung Zugang zu Datenquellen benötigt. Falls dies der Fall ist, wird sich RAG als bessere Alternative darstellen. Im Gegensatz dazu ist es zwar möglich, ein LLM so einzustellen, dass es ein gewisses externes Wissen erlernen wird, doch erfordert dies einen großen Datensatz aus dem Zielbereich, welcher sich für ein Trainingsprozess eignet. Wenn das System einen speziellen Schreibstil oder eine tiefgreifende Anpassung an domänenspezifische Fachsprache und Konventionen erfordert, stellt die Feinabstimmung einen direkteren Weg dar, um diese Anpassung zu erreichen.

Die Feinabstimmung kann dazu beitragen, Halluzinationen bis zu einem gewissen Grad zu reduzieren, indem das Modell auf die Trainingsdaten eines bestimmten Bereichs ausgerichtet wird. Allerdings kann das Modell immer noch Antworten fälschen, wenn es mit unbekannten Eingaben konfrontiert wird. Im Gegensatz dazu ist RAG von Natur aus weniger anfällig für Halluzinationen, da jede Antwort auf angereicherten Daten beruhen. Das Abrufen von unterstützenden Informationen, vor der Antwortgenerierung, verschafft RAG einen Vorteil bei der Sicherstellung fachlich korrekter Ergebnisse. Bei RAG basierten Anwendungen ist vor allem die Genauigkeit und die Unterdrückung von Halluzinationen von entscheidender Bedeutung. Bei der Entscheidung zwischen RAG und Finetuning ist ein entscheidender Faktor die Menge an domänenspezifischen, gelabelten Trainingsdaten, die zur Verfügung steht. Wenn eine Fülle von gelabelten Daten vorhanden ist, welche die Feinheiten des Fachgebiets erfassen, kann die Feinabstimmung ein maßgeschneidertes und verfeinertes Verhalten des Modells bieten. Aber in Szenarien, in denen solche Daten begrenzt sind, bietet ein RAG-System eine robuste Alternative.

Ein weiterer grundlegender Aspekt, der bei der Entscheidung zwischen RAG und Finetuning zu berücksichtigen ist, ist die Dynamik der Daten. Die Feinabstimmung eines LLM auf einen bestimmten Datensatz bedeutet, dass das Wissen des Modells eine statische Momentaufnahme dieser Daten zum Zeitpunkt des Trainings ist. Bei Änderungen der Daten muss entsprechend ein neuer Trainingsprozess gestartet werden. Wenn der Inhalt der angebundenen Datenbank aktualisiert wird, integriert das RAG-System diese Änderungen nahtlos und behält so seine Relevanz bei, ohne dass das Modell häufig neu trainiert werden muss.

Eine weitere Frage stellt sich im Bezug auf die Transparenz des Systems. Die Feinabstimmung eines LLM ist zwar unglaublich leistungsfähig, funktioniert aber wie eine Blackbox. Es ist dadurch schwierig, die Quelle oder die Begründung hinter einer Ausgabe zu erkennen. RAG-Systeme hingegen bieten ein Maß an Transparenz. Die Abfragekomponente ermöglicht die Überprüfung, welche externen Dokumente oder Datenpunkte als relevant ausgewählt wurden.<sup>9 10</sup>

## 2.15 Geschichte

Aus „Künstliche Intelligenz und der Einsatz in der Versicherungsbranche – 2.1 Geschichte“:

Ab 2004 wurden neue Lernverfahren für Neuronale Netze entwickelt, sogenannte Deep Learning Verfahren, wodurch ab 2010 der Einsatz von KI-Anwendungen deutlich gestiegen ist. Zudem ist heutzutage die Menge an nutzbaren Daten, wie auch die verfügbare Rechenleistung, dermaßen angestiegen, dass die Entwicklung von Künstlicher Intelligenz von Jahr zu Jahr rasante Fortschritte erzielen konnte. So hat der Bereich der KI einen bemerkenswerten Aufschwung erlebt, auch zu nennen sind Fortschritte im Deep Learning, einer Unterdisziplin des maschinellen Lernens. Deep Learning-Modelle haben herausragende Leistungen in Bereichen wie Bilderkennung, Sprachverarbeitung und automatischem Übersetzen erzielt. Dies hat die Anwendungsmöglichkeiten von KI erheblich erweitert und die Grundlage für viele moderne KI-Anwendungen gelegt. Bemerkbar wurde das vor allem durch die Veröffentlichung von ChatGPT im November 2022, wodurch die Thematik in der breiten Öffentlichkeit und in vielen Unternehmen präsent wurde.

## 2.16 Geschichte von RAG

Als ersten Beitrag zum Thema Retrieval-Augmented Generation kann folgende Ausarbeitung genannt werden. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" von Patrick Lewis, Ethan Perez, Aleksandara Piktus et al. Diese Arbeit wurde auf der Konferenz NeurIPS 2020 vorgestellt. In diesem Beitrag wurde zum ersten Mal Retrieval-Augmented Generation in der Form als flexiblere Technik für die Generierung wissensintensiven Ausgaben aufgezeigt. In dieser Ausarbeitung kombinierten die Forscher eine Generative-KI mit einem Retriever-Modul, um zusätzliche Informationen aus einer externen Wissensquelle bereitzustellen, die leichter aktuell gehalten werden können. Darauffolgend wurden im Laufe der Jahre weitere Forschungspapiere und wissenschaftliche Arbeiten veröffentlicht, welche verschiedene Aspekte von RAG untersuchen, erweitern und diskutieren. Außerdem hat sich bis heute eine Vielzahl an Blog-Beiträgen angehäuft, welche RAG mit unterschiedlichen Technologien und für verschiedene Anwendungsfälle implementieren.

---

<sup>9</sup> (Hotz, 2023)

<sup>10</sup> (OpenAI, 2024)

### 3 Analyse und Technologieauswahl

Um eine anschauliche und realitätsnahe Implementierung von RAG präsentieren zu können, muss sich mit den notwendigen Komponenten des Systems beschäftigt werden. Grundlegend für die Entwicklung ist ein LLM-APP-Framework, welches es ermöglicht KI-Technologien einzubauen. Dazu gehört die Handhabung einer Vektordatenbank und das verwendete Embedding-Modell. Außerdem wird die Verwendung eines Sprachmodells durch das Framework unkompliziert ermöglicht. Darüber hinaus muss sich mit der für den produktiven Betrieb notwendigen Infrastruktur auseinandergesetzt werden.

#### 3.1 LLM-App-Framework

Für die Entwicklung von Anwendungen, welche LLMs nutzen, bietet sich der Einsatz von LLM-App-Frameworks an. Dadurch wird dem Entwickler eine modulare und flexible Struktur geboten. Es werden standardisierte Schnittstellen vorgegeben, unterschiedliche Indexierungstechniken bereitgestellt und der Entwickler kann flexible Pipelines zur Orchestrierung von LLM-bezogenen Operationen aufbauen. Dadurch können verschiedene Datenobjekte und -quellen, sowie Sprachmodelle in der eigenen Anwendung integriert werden. Des Weiteren können komplexe und in Eigenentwicklung aufwendig umzusetzende Abläufe unkompliziert und wartbar erstellt werden. Im Bereich der LLM-App-Frameworks haben sich LangChain, LlamaIndex und Haystack als die größten Vertreter herausgestellt.<sup>11 12 13</sup>

LangChain bietet Flexibilität und Vielseitigkeit, hier ist vor allem die Erweiterbarkeit herauszustellen. Durch die umfangreiche Dokumentation und durch die einfache Bedienung genießt LangChain große Beliebtheit. Zudem besitzt LangChain eine Vielzahl an integrierbaren LLMs und Indizierungstools. Für die Erstellung von robusten und zugleich anpassbaren Anwendungen ist LangChain eine vielversprechende Wahl und hat dadurch die höchste Verbreitung innerhalb von Unternehmen erlangt.

LlamaIndex zeichnete sich durch seine Performance in Bezug auf das Datenhandling aus. Der Fokus von LlamaIndex liegt hauptsächlich auf dieser, weshalb eine effiziente Indizierung und schnelles Abrufen von Daten ermöglicht werden. Entsprechend bietet sich der Einsatz vor allem bei Anwendungen an, bei denen Daten im Mittelpunkt stehen. Im Vergleich zu LangChain ist es jedoch nicht so umfassend und bietet auch keine so umfangreiche Palette an LLM-Anbietern wie LangChain.

Haystack ist das unpopulärste Framework der drei aufgeführten, besitzt jedoch eine sehr gute Dokumentationsqualität. Es wird als das unkomplizierteste dargestellt und zeichnet sich für die Entwicklung von kleineren Anwendungen und Prototypen aus.

LangChain besitzt eine steile Lernkurve und ist das umfangreichste und flexibelste Tool unter den Frameworks ist. Außerdem bietet es für den Unternehmenskontext die vielversprechendsten Einsatzmöglichkeiten, weshalb dieses Framework verwendet wird.<sup>14</sup>

---

<sup>11</sup> (llamaindex, 2024)

<sup>12</sup> (langchain, langchain, 2024)

<sup>13</sup> (deepset, 2024)

<sup>14</sup> (Ali, 2023)

3.2 Vektordatenbanksystem

Im Kontext von RAG wird in den meisten Implementierungen ein vektorenbasiertes Datenbankmanagementsystem verwendet. Da die Qualität an semantischen Suchen in einem solchen System am höchsten ist und momentan die meisten RAG-Systeme auf diese zurückgreifen, wird sich dazu entschieden, in dieser Arbeit ebenfalls eine Vektordatenbank zu nutzen. Durch die sehr modulare Struktur eines RAG lässt sich das zugrundeliegende Datenbanksystem leicht austauschen.

Daher wird in der ersten Entwicklungsphase der Fokus auf das Umsetzen eines Grundsystems gelegt. Die Seite db-engines.com stellt monatlich Vergleiche zu DBMS-Systemen an. LangChain bietet unkomplizierte Anbindungen der Systeme Chroma und Pinecone an, weshalb das Nutzen dieser Vektordatenbanken fokussiert wird. Durch die gute Dokumentation von Chroma im Kontext von LangChain, wird sich dazu entschieden, Chroma während der Entwicklungsphase zu implementieren. Im späteren Verlauf der Systementwicklung sollen unterschiedliche vektorbasierte Datenbanksysteme in der RAG-Anwendung ausgetauscht und verglichen werden, um die bestmögliche Wahl für das System zu gewährleisten.<sup>15</sup>























Rang			DBMS	Datenbankmodell	Punkte		
Jun 2024	Mai 2024	Jun 2023			Jun 2024	Mai 2024	Jun 2023
1.	1.	1.	Kdb 	Multi-Model 	7,71	+0,16	-0,29
2.	2.	 3.	Pinecone	Vektor	3,23	+0,07	+1,11
3.	3.	 5.	Milvus 	Vektor	2,78	+0,48	+1,48
4.	4.	 2.	Chroma	Vektor	2,00	-0,07	-0,39
5.	5.	 6.	Weaviate 	Vektor	1,52	-0,20	+0,44
6.	6.	 10.	Qdrant	Vektor	1,28	+0,12	+0,70
7.	7.	 4.	AllegroGraph 	Multi-Model 	1,13	+0,07	-0,23
8.	8.	 7.	CrateDB 	Multi-Model 	0,71	-0,02	-0,26
9.	 10.	9.	Vespa	Multi-Model 	0,62	+0,08	+0,03
10.	 9.	 8.	Vald	Vektor	0,60	-0,03	-0,31
11.	11.	11.	Deep Lake	Vektor	0,31	+0,04	-0,20
12.	12.	12.	MyScale	Multi-Model 	0,20	0,00	-0,06
13.	13.	13.	JaguarDB	Multi-Model 	0,06	+0,06	+0,06
14.	 13.		Transwarp Hippo	Vektor	0,05	+0,05	

Abbildung 5 Vergleich von Vektordatenbanken, db-engines.com<sup>16</sup>

3.3 Embedding-Modelle

Embedding-Modelle erzeugen eine Vekordarstellung von Daten, dadurch kann die Bedeutung und die semantische Beziehung von z.B. Textstücken erfasst werden. Es wird uns ermöglicht, in einer größeren Menge an Daten, ähnliche Daten ausfindig zu machen. Diese numerische Darstellung der eigentlichen Daten wird als Embedding bezeichnet.

<sup>15</sup> (langchain, Vector stores, 2024)

<sup>16</sup> (solid-it, 2024)

Ähnlich wie bei den Vektordatenbanken gibt es eine Vielzahl an unterschiedlichen Implementierungen, welche in einem RAG-System genutzt werden können. Da auch hier das Austauschen von konkreten Modellen problemlos möglich ist, werden vorerst die Modelle von OpenAI und Cohere genutzt. Für diese Modelle existierten umfangreiche Dokumentationen für die Integration in einer LangChain-Anwendung. Im späteren Verlauf der Entwicklung soll auch in diesem Bereich ein Vergleich durchgeführt werden, um das bestmögliche Modell zu nutzen. Zudem ist das Betreiben eines lokalen Embedding-Modells möglich, jedoch bei durchschnittlicher Hardware nicht zielführend. Grund ist, dass die Einbettungen keine hohe Qualität aufweisen und dazu viel Rechenzeit beanspruchen, weshalb die Möglichkeit des lokalen Betriebes nicht weiter behandelt wird.<sup>17 18 19</sup>

### 3.4 LLM

Das Herzstück einer LLM-basierten Anwendung stellt naheliegenderweise das Sprachmodell selbst dar. Dieses Modell ist für den generativen Teil von Retrieval-Augmented-Generation verantwortlich und sollte daher ein hohes Maß an Qualität besitzen. Ein weiterer wesentlicher Aspekt ist die Austauschbarkeit der LLMs innerhalb eines RAG-Systems. Dank dieser Austauschbarkeit ist es möglich, nach der grundlegenden Entwicklung des Systems unterschiedliche LLMs zu vergleichen und zu evaluieren, ohne großen Aufwand betreiben zu müssen. Dies eröffnet die Möglichkeit, stets das beste verfügbare Modell für die spezifischen Anforderungen der Anwendung zu nutzen und flexibel auf neue Entwicklungen im Bereich der Sprachmodelle zu reagieren. Für das zu entwickelnde RAG-System sollen hauptsächlich zwei Modelle verwendet werden. Aus Qualitätsgründen wird auf ChatGPT zurückgegriffen, sowie das quelloffene und lokal betreibbare Llama3 soll verwendet werden. Während ChatGPT durch seine exzellente Performance überzeugt, bietet Llama3 durch seine Flexibilität und nicht vorhandenen Kosten eine wertvolle Ergänzung.<sup>20 21</sup>

Der Einsatz von ChatGPT hat den Vorteil, dass es sehr gute Ergebnisse liefert, die besonders bei der Entwicklung eines Prototyps von hohem Wert sind. So können die bestmöglichen Ergebnisse geliefert werden und dadurch werden die Potenziale, welche der Prototyp aufzeigt, gut skizziert. Allerdings sind damit Kosten verbunden, die je nach Quantität der Nutzung variieren können.

Auf der anderen Seite ermöglicht der Einsatz eines lokal betriebenen Llama3, eine Minimierung der Entwicklungskosten. Dies ist besonders vorteilhaft in Phasen, in denen der generative Teil der Anwendung nicht im Fokus steht. Ein weiterer Vorteil der Nutzung von Llama3 ist die Flexibilität, die es bietet. Da Llama3 quelloffen ist, kann es an spezifische Bedürfnisse und Anforderungen angepasst werden, was die Entwicklung und Optimierung des RAG-Systems erleichtert. Zudem ermöglicht der lokale Betrieb von Llama3 eine bessere Kontrolle über die Daten und die Einhaltung von Datenschutzbestimmungen.

Insgesamt bietet die Kombination von ChatGPT und Llama3 eine ausgewogene Herangehensweise, die sowohl qualitativ hochwertige Ergebnisse als auch kosteneffiziente Entwicklung ermöglicht. Diese Strategie erlaubt es, die Stärken beider Modelle optimal zu nutzen und ein leistungsfähiges, effizientes RAG-System zu entwickeln.

---

<sup>17</sup> (Chohere, 2024)

<sup>18</sup> (langchain, text embedding langchain, 2024)

<sup>19</sup> (ollama, local embedding , 2024)

<sup>20</sup> (langchain, LLMs with langchain, 2024)

<sup>21</sup> (ollama, llama3 with ollama, 2024)

### 3.5 Infrastruktur

Die Wahl der geeigneten Infrastruktur für den Betrieb von RAG hängt stark von den Anforderungen und Prioritäten des Projektes ab. Eine lokale Implementierung stellt die kostenfreundlichste und datenschutzverträglichste Möglichkeit dar. Große Performanceabstriche und nur begrenzte Skalierungsmöglichkeiten werden hierbei in Kauf genommen. Diese Möglichkeit kann mit dem Tool Ollama umgesetzt werden, welches es ermöglicht, lokale Instanzen von LLMs und Embedding-Modellen zu betreiben. Vor Allem bei der Entwicklung und Präsentation von Prototypen ist eine lokale Implementierung die flexibelste Wahl, mit einer ausgezeichneten Handhabung. Neben einer lokalen Umgebung bietet sich auch die Möglichkeit auf einen Cloud-Anbieter zurückzugreifen. Dadurch kann die Verfügbarkeit der Anwendung auf globales Niveau gehoben werden bei einer umfangreichen und problemlosen Skalierung. Des Weiteren besitzen die meisten Cloud-Anbieter KI-Dienste, wodurch die Entwicklung von LLM-Anwendungen erleichtert wird. Als Nachteile müssen hier Datenschutzbedenken aufgeführt werden, wie auch eine komplexe Einrichtung und Verwaltung der Cloud-Ressourcen.

Selbstverständlich ist das Verwenden von Cloud-Diensten auch mit Kosten verbunden. Ein Großer Cloud-Betreiber, der zu nennen wäre, ist der Marktführer AWS (Amazon Web Services), welcher eine breite Palette von unterstützenden Diensten anbietet und ein hohes Maß an Flexibilität ermöglicht. Google Cloud Platform und Microsoft Azure wären weitere Cloud-Lösungen, welche ähnlich wie AWS eine breite Palette an Diensten bereitstellen, die bei der Entwicklung von LLM-Anwendungen die Arbeit erleichtern können. Generell lässt sich sagen, dass für den produktiven Einsatz ein Cloud-Anbieter eine gute Wahl ist. Da durch den Fokus auf die Entwicklung eines Prototyps die Entwicklung selbst im Vordergrund steht, wird eine lokale Infrastruktur, konkret der Entwickler-PC, genutzt. Falls während der Bearbeitung auffällt, dass der Zeitliche Rahmen eingehalten werden kann, wird versucht den Prototypen auch auf einer Cloud-Infrastruktur in Betrieb zu nehmen.

### 3.6 User-Interface

Damit eine nutzerfreundliche Interaktion mit dem System ermöglicht wird, soll ein kleines User-Interface in Form einer Web-Anwendung, welche auf das Kernsystem über eine REST-Schnittstelle zugreift, gebaut werden. Für die Umsetzung bietet sich eines der drei großen JavaScript-Web-Frameworks, Angular, React oder Vue an.

Angular ist ein umfassendes von Google entwickeltes Framework, das eine Vielzahl von eingebauten Funktionen und eine starke Struktur bietet. Es ist geeignet für große und komplexe Anwendungen, da es mit einer Menge an Vorgaben und integrierten Tools daherkommt. Es besitzt eine umfangreiche Dokumentation und vor allem Unternehmen greifen auf Angular zurück.

Ein weiterer von einem Tech-Unternehmen ins Leben gerufene Framework ist React. Die Weiterentwicklung und Wartung von React werden vom Meta-Konzern gesichert. Zudem besitzt React eine große Community und stellt das Populärste der drei Frameworks dar. React besitzt eine gute Balance zwischen Flexibilität, Komplexität und Robustheit.

Vue ist ein Framework, welches von einem ehemaligen Google-Entwickler ins Leben gerufen wurde und durch ein internationales Entwicklerteam und einer großen Community weiterentwickelt wird. Durch den Fokus auf einen simplen Aufbau können kleinere performante Anwendung schnell und mit geringem Aufwand entwickelt werden.

Alle drei Frontend-Frameworks stehen als Open-Source-Software auf dem freien Markt zur Verfügung und bieten ausreichende Funktionalität und Features an, um das User-Interface für RAG zu realisieren. Da hierbei jedoch nur ein minimalistisches Frontend umgesetzt werden soll wird Vue für die Umsetzung verwendet. Grund ist die schnelle und kompakte Entwicklung mit diesem Framework.

Die Vorteile eines Frontends sind sowohl für die Entwicklung als auch für die potenziellen Interessenten offensichtlich. Angesichts dieser Vorteile wurde beschlossen, ein Frontend mit dem Framework Vue zu realisieren. Vue bietet eine flexible und performante Grundlage für die Entwicklung von Webanwendungen, was es zur idealen Wahl für dieses Projekt macht.<sup>22</sup>

### 3.7 RAG-Modellierung

Eine typische RAG-Anwendung besteht aus zwei Komponenten. Zum einen die Datenaufbereitung, welche vor dem Verwenden der eigentlichen Anwendung passiert, hier werden Daten aufgenommen und in eine für das Hauptsystem notwendigen Form umgewandelt und persistiert. Die andere Komponente nimmt eine Nutzerfrage entgegen, ruft zur eingegebenen Frage passende persistierte Daten ab und gibt diese weiter an ein Sprachmodell.

In einem ersten Schritt werden die Daten, welche indiziert werden sollen, geladen. Dafür wird je nach Art der Daten und Dokumente ein entsprechendes Modul verwendet, um das Laden zu ermöglichen. Anschließend werden die Daten aufgeteilt, die Dokumententeile werden auch als Chunks bezeichnet. Diese Chunks haben den Vorteil, dass sie einfacher zu durchsuchen sind und die Kontextgröße eines Sprachmodelles einhalten. Um gute Ergebnisse bei Ähnlichkeitssuchen zu erzielen, bietet sich eine Vektordatenbank an. Damit die Chunks in der Datenbank abgelegt werden können, müssen diese in eine Vektorform umgewandelt werden, dafür wird ein Embedding-Model genutzt.



Abbildung 6 Darstellung der Daten-Indizierung, eigene Abbildung

Die Hauptkomponente von RAG nimmt eine Nutzerfrage entgegen und bezieht für die Frage relevante Chunks aus dem Datenspeicher. Dieser Schritt ist das Retrieval von RAG. In diesem Schritt sollte das gleiche Embedding-Model verwendet werden, welches bei der Indizierung verwendet wurde, um relevante Ergebnisse geliefert zu bekommen. Anschließend wird die eigentliche Anfrage mit den gelieferten Chunks angereichert und an ein LLM gegeben, um eine Antwort zu generieren.

---

<sup>22</sup> (Noack, 2024)



## Konzeption und Implementierung eines Wissensmanagementsystems basierend auf Retrieval-Augmented Generation

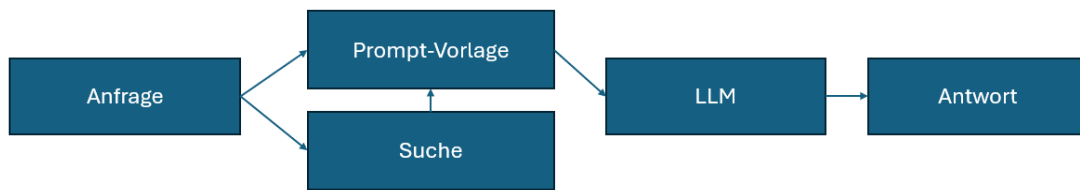


Abbildung 7 Ablauf der Datenanreicherung und Generierung, eigene Abbildung

Beide skizzierten Abläufe, die Datenaufbereitung und die Hauptkomponente, können praktisch durch eine Daten-Pipeline-Architektur realisiert werden. Das Datenbanksystem, das Embedding-Modell und das Sprachmodell sind austauschbar, weshalb bei der Entwicklung auf Modularität hohen Wert gelegt. Während der Entwicklung soll zwar der Betrieb der entwickelten Systeme auf einem lokalen Computer erfolgen, nichtsdestotrotz soll die RAG-Anwendung Systemunabhängig lauffähig sein, wodurch das Thema Virtualisierung eine Rolle spielen wird.

Der Genaue Ablauf und das Zusammenspiel aller Komponenten vom RAG -Hauptsystem zeigt folgendes Diagramm auf.<sup>23</sup>

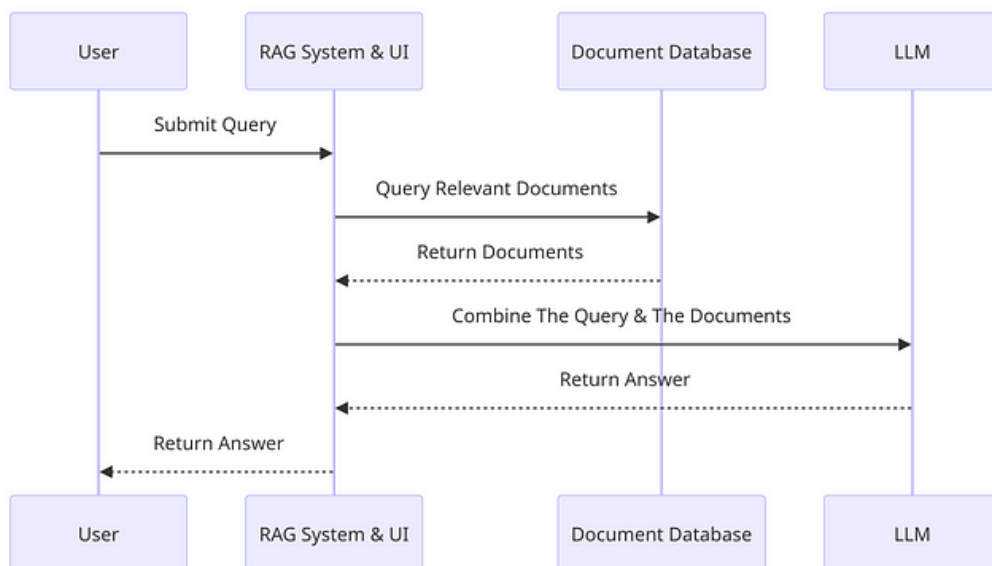


Abbildung 8 Ablaufdiagramm von RAG

1. Der Benutzer gibt eine Anfrage in das System ein. Dies ist der erste Schritt und die eigentliche Anfrage des Benutzers wird erfasst.
2. Das RAG-System verarbeitet die Anfrage des Benutzers und sucht nach relevanten Datenobjekten in der Vektor-Datenbank.
3. Die Vektor-Datenbank erhält die Anfrage nach relevanten Datenobjekten, sucht durch eine semantische Suche passende Datenobjekte und gibt die gefundenen Dokumente an das RAG-System zurück.
4. Das RAG-System nimmt die von der Vektor-Datenbank gelieferten Datenobjekte und kombiniert diese mit der ursprünglichen Anfrage.
5. Die kombinierte Anfrage und die Datenobjekte werden an ein LLM gesendet, welches eine Antwort auf der Grundlage der bereitgestellten Informationen generiert.
6. Die vom LLM generierte Antwort wird über das RAG-System an den Nutzer ausgegeben.

<sup>23</sup> (Demir, 2023)



### 3.8 Datenvorbereitung

Das Thema Daten spielt für den in dieser Arbeit umzusetzenden Prototypen eine zentrale Rolle. Damit in den ersten Gehversuchen mit RAG das Thema Datenschutz keine Relevanz besitzt, werden öffentlich zugängliche Daten genutzt. Hierzu gehören öffentlich abrufbare Versicherungsbedingungen, Brettspielanleitungen und der der EU-AI-Act. Der EU AI Act ist ein Gesetzesvorschlag der Europäischen Union, der darauf abzielt, die Nutzung und Entwicklung von Künstlicher Intelligenz zu regulieren. Der EU-AI-Act stellt beispielhaft eine große Menge an zusammengehörenden Kontext dar, wodurch mit den Themen Chunking, Embedding und Speicherung in einer Vektordatenbank Erfahrung gesammelt werden kann. Des Weiteren wird zu Testzwecken der EU-AI-Act in .txt-Format und in .pdf-Format verwendet, damit auch beim Thema Handhabung unterschiedlicher Dateiformate Erfahrung gesammelt wird, da der Informationsgehalt in beiden Formattypen identisch ist. Grund für die Nutzung Brettspielanleitungen ist, dass mit diesen die Qualität der Suchergebnisse, der Ausgaben sowie der Verarbeitungen gut verglichen werden können und keine besonderen fachlichen Kenntnisse notwendig sind. Konkret werden die Anleitungen von den Gesellschaftsspielen „Catan“, „Monopoly“, „Risiko“ und „Vier gewinnt“ genutzt. Damit ein unternehmensbezogener Anwendungsfall skizziert werden kann, wurde sich dazu entschieden Vertragsbedingungen von Versicherungsprodukten zu nutzen.

### 3.9 Umsetzung und Anwendungsfälle

Das Umsetzen des RAG-Systems erfolgt schrittweise, um sich mit den einzelnen Komponenten ausreichend beschäftigen zu können. Die ersten Schritte umfassen die Einrichtung von LangChain, die Integration der Vektordatenbank und die Verwendung des Embedding-Modells. In der initialen Phase soll die Handhabung von Chunking, Embedding und Speicherung in einer Vektordatenbank von unterschiedlichen Dateiformaten getestet werden. Darauf folgend wird die Anbindung eines Sprachmodells getestet. Nachdem ein grundlegender MVP erstellt wurde, wird sich um die Entwicklung eines soliden Gesamtsystems gekümmert. Dazu gehört die Aufteilung in die eigentliche RAG-Anwendung, sowie einer kleineren Frontend-Anwendung, um das System bedienen zu können. Dabei sollen Nutzer die Möglichkeit haben mit dem System in Chat-Form interagieren zu können. Die Datengrundlage soll durch das Hochladen von neuen Dokumenten, welche vom System durch die Datenaufbereitung verarbeitet wird, erweitert werden. Da das System austauschbare Komponenten besitzt soll eine Konfiguriermöglichkeit existieren. Damit die Leistungsfähigkeit des Systems aufgezeigt werden kann, wird sich mit möglichen Anwendungsfällen auseinandergesetzt. Es sollen bei den unterschiedlichen Anwendungsfällen nur minimale Änderungen am entwickelten Grundsystem vorgenommen werden müssen.

Ein simpler Anwendungsfall ist ein Assistent, welcher Gesellschaftsspiele erklären kann. Die Interaktion soll in Chat-Form erfolgen und die Datengrundlage soll gängige Gesellschaftsspiele beinhalten. Diese Einsatzmöglichkeit erfordert wenig Fachwissen und ist für jedermann zugänglich. Es ist eine gute Grundlage, um die RAG-Funktionalität zu testen und zu optimieren. Die Nutzer sollen einfache Fragen zu Spielregeln stellen können und können hilfreiche sowie informative Antworten erhalten.

Ein unternehmensspezifischer Anwendungsfall stellt das Abfragen und Vergleichen von Versicherungsbedingungen dar. Die Interaktion mit dem System erfolgt wieder in Chat-Form, dabei können Informationen zu verschiedenen Versicherungstarifen abgefragt werden. Das System soll einen Vergleich der Bedingungen zwischen unterschiedlichen Tarifgenerationen ermöglichen und dabei die Quelldokumente dem Nutzer zusätzlich zur Verfügung stellen.

### 3.10 Übersicht

In diesem Kapitel wurde eine fundierte Auswahl der verschiedenen Technologien für die Entwicklung einer RAG-Anwendung zusammengestellt. Dabei wurden die notwendigen Komponenten und deren Integration in das Gesamtsystem detailliert betrachtet. Die folgende Auflistung bietet eine Übersicht der Auswahl:

- **LLM-App-Framework:** LangChain wird als Framework genutzt, da es Flexibilität, Erweiterbarkeit und zahlreiche integrierbare Modelle und Indizierungstools bietet.
- **Vektor-Datenbanksystem:** Als grundlegende Vektor-DB wird Chroma genutzt, welche eine unkomplizierte Integration mit LangChain besitzt.
- **Embedding-Modell:** Das Embedding-Modell „Text-embedding-ada-002-v2“ von OpenAI wird aus Qualitätsgründen verwendet. Lokale Embedding-Modelle werden aufgrund von Leistungs- und Qualitätsaspekten vorerst nicht verfolgt.
- **Sprachmodell:** OpenAI und Llama3 werden als LLMs verwendet. OpenAI überzeugt durch exzellente Qualität, während Llama3 Kosteneffizienz und lokalen Betrieb bietet. Konkret wird das GPT-Modell mit der Bezeichnung „gpt-3.5-turbo-16k“ verwendet.
- **Infrastruktur:** Um Kosten zu minimieren und die Handhabung zu erleichtern, wird eine lokale Infrastruktur während der Entwicklung genutzt.
- **User-Interface:** Ein minimalistisches User-Interface wird mit dem Frontend-Framework Vue entwickelt.

Das nachfolgende Diagramm zeigt die unterschiedlichen Komponenten des zu entwickelnden Systems auf:

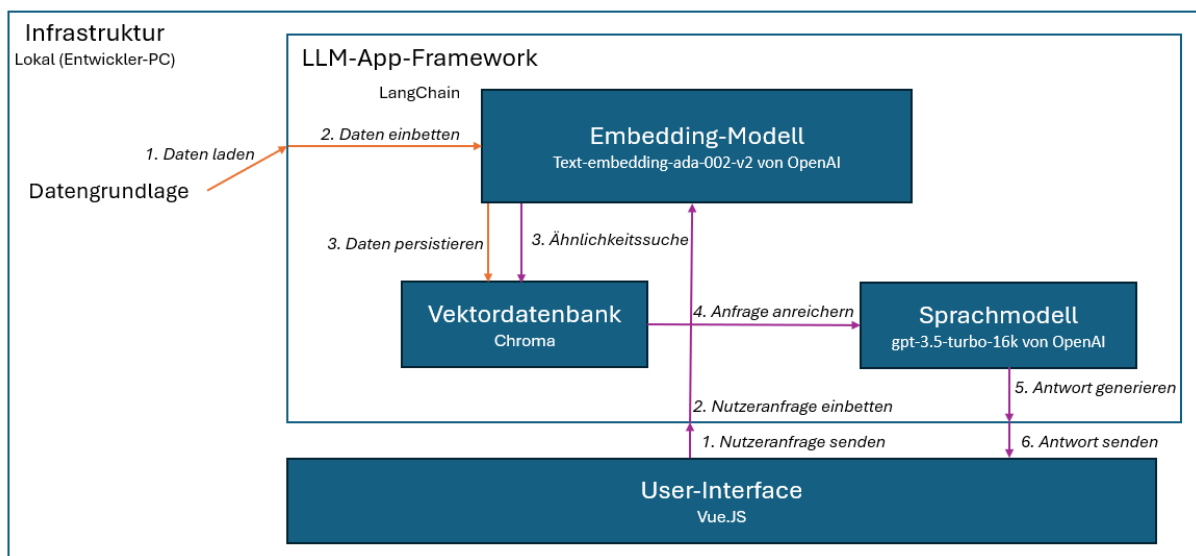


Abbildung 9 Komponenten des RAG-Systems, eigene Abbildung

## 4 Erste Schritte

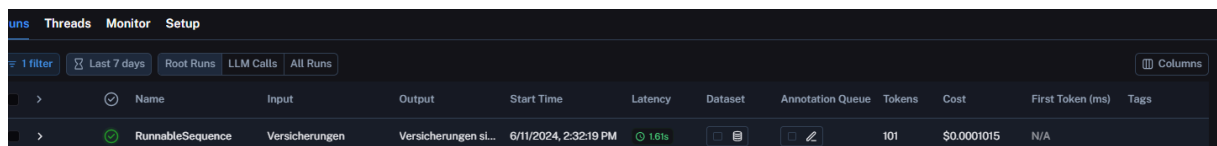
### 4.1.1 Erste Schritte mit LangChain

Um ein umfassendes Verständnis über LangChain als LLM-Framework zu entwickeln, wird im ersten Schritt intensiv mit den von LangChain selbst bereitgestellten Tutorials gearbeitet. Das Ziel dieses Schrittes ist es, eine solide Wissensbasis für die Arbeit mit dieser Technologie zu schaffen. Dabei werden die grundlegenden Komponenten von LangChain getestet, zu denen Prompt-Vorlagen, die Verbindung zu LLMs und das Einrichten von Ausgabeparsern gehören.

Ein konkretes Beispiel zur Veranschaulichung des Gelernten ist der Aufbau einer kleinen Pipeline aus Komponenten. Eine Pipeline, in dem Kontext von LangChain auch Kette genannt, ist dabei eine Abfolge von Arbeitsschritten. Diese Pipeline wird so konzipiert, dass sie mit LangSmith überwacht werden kann und mit LangServe als REST-API nutzbar ist. In LangChain-Anwendungen, die aus mehreren Schritten bestehen und teilweise komplexe Verarbeitungsschritte umfassen, ist es besonders vorteilhaft, genau zu überwachen, wie die Daten in den einzelnen Schritten aussehen. LangSmith ist ein speziell dafür entwickeltes Tool, das optional in einer LangChain-Anwendung genutzt werden kann, um diese Überwachung zu gewährleisten.

Die erste programmatische Umsetzung besteht darin, einen Prompt zusammenzubauen, der das LLM anweist, wie ein Assistent zu agieren, welcher komplexe Sachverhalte auf einfache Weise erklärt. Das Thema, das erläutert werden soll, wird als Eingabe an den Prompt übergeben. Der zusammengebaute Prompt wird dann an das angebundene LLM, in diesem Fall GPT, übergeben, um die gewünschte Erklärung zu generieren. Diese Vorgehensweise bietet eine praxisnahe Einführung in die Arbeit mit LangChain und zeigt, wie verschiedene Komponenten und Tools integriert werden können.

Beim Ausführen des Skripts wird die Ausgabe nicht ausgegeben, damit mit dem Tool LangSmith experimentiert werden kann.



	Name	Input	Output	Start Time	Latency	Dataset	Annotation Queue	Tokens	Cost	First Token (ms)	Tags
>	RunnableSequence	Versicherungen	Versicherungen sL...	6/11/2024, 2:32:19 PM	1.6s			101	\$0.0001015	N/A	

Abbildung 10 LangSmith Ansicht

LangSmith bietet eine umfassende Übersicht aller aufgerufenen Ketten, begleitet von entsprechenden Meta-Informationen. Zu diesen Informationen gehören unter anderem die Anzahl der genutzten Tokens, die dabei entstandenen Kosten und deren Erfolgsrate. Wenn eine bestimmte Pipeline ausgewählt wird, wie in unserem Beispiel die in diesem Kontext ausgeführte, können alle Unterteile der definierten Kette eingesehen werden. Dazu gehört beispielsweise das erstellte Pipeline-Objekt, dem das JSON-Objekt `{"input": "Versicherungen"}` übergeben wird, sowie der Aufruf von GPT.

Für die Systemanalyse ist besonders das schrittweise Nachverfolgen der Daten in den einzelnen Teilschritten von Vorteil. Diese detaillierte Nachverfolgung ermöglicht es uns, die Ausgabe der angestoßenen Kette zu analysieren und zu überprüfen. LangSmith erleichtert somit die genaue Untersuchung und Optimierung der verschiedenen Schritte innerhalb der Kette und stellt sicher, dass alle Prozesse wie erwartet funktionieren.

#### 4.1.2 Erste Schritte mit Chunking

Das Zuführen und semantische Suchen auf gesamten Dokumenten ist aus Kosten- und Qualitätsgründen nicht zielführend. Eine Möglichkeit hier Abhilfe zu schaffen ist, die Dokumente in kleinere Teile aufzuteilen, sogenannte Chunks. LangChain bietet eine Reihe an Möglichkeiten, Dokumente aufzuteilen und zu manipulieren. Um zwischen den einzelnen Teilen einen Kontext aufrechtzuhalten, können die Chunks Überschneidungen besitzen.

LangChain implementiert eine Vielzahl an unterschiedlichen Splittern, welche Dokumente aufteilen und somit das Chunking realisieren. Von speziellen Anwendungsfällen wie Quellcode- und HTML-Splittern, über Text-Splitter, bis hin zu Rekursiven-Splittern, welche die semantische Zusammengehörigkeit in den Fokus stellen.

Character-Chunking ist eine Möglichkeit, Dokumente und Texte anhand der Zeichenanzahl zu teilen. Ein Beispiel hierfür ist die Aufteilung des EU-AI-Acts, hier stellen die „- - -“ das Ende eines Chunks dar.

```
---
{SEC(2021) 167 final} - {SMD(2021) 84 final} - {SMD(2021) 85 final}

BEGRIFFLICHUNG

1.KONTEXT DES VORSCHLAGS

1.1.Gründe und Ziele des Vorschlags

---
Diese Begründung ist dem Vorschlag für eine Verordnung beigefügt, mit der harmonisierte Vorschriften für künstliche Intelligenz festgelegt werden (Gesetz über künstliche Intelligenz). Künstliche Intelligenz (KI) bezeichnet eine Reihe von Technologien, die sich rasant entwickeln und einen vielfältigen Nutzen für Wirtschaft und Gesellschaft über das gesamte Spektrum industrieller und gesellschaftlicher Aktivitäten hinweg hervorbringen können. Der Einsatz künstlicher Intelligenz zur Verbesserung von Prognosen, zur Optimierung von Abläufen und der Ressourcenzuweisung sowie zur Personalisierung der Dienstleistung kann für die Gesellschaft und die Umwelt von Nutzen sein und Unternehmen sowie der europäischen Wirtschaft Wettbewerbsvorteile verschaffen. Bedarf besteht insbesondere in Sektoren, von denen eine große Wirkung ausgeht, wie Klimaschutz, Umwelt und Gesundheit, öffentlicher Sektor, Finanzen, Mobilität, Inneres und Landwirtschaft. Dieselben Faktoren und Techniken, die für den sozioökonomischen Nutzen der KI sorgen, können aber auch neue Risiken oder Nachteile für den Einzelnen oder die Gesellschaft hervorbringen. Vor dem Hintergrund des rasanten technologischen Wandels und möglicher Herausforderungen ist die EU entschlossen, einen ausgewogenen Ansatz zu erreichen. Es liegt im Interesse der Union, die technische Führungsrolle der EU auszubauen und dafür zu sorgen, dass die Europäerinnen und Europäer von den im Einklang mit den Werten, Grundrechten und Prinzipien der Union entwickelten und funktionierenden neuen Technologien profitieren können.

---
Dieser Vorschlag geht auf das politische Engagement von Präsidentin von der Leyen zurück, die in ihren politischen Leitlinien für die Kommission (2019-2024) eine Union, die mehr erreichen will, angekündigt hat. Im Nachgang zu dieser Ankündigung veröffentlichte die Kommission am 19. Februar 2020 ihr Weißbuch zur KI. Ein europäisches Konzept für Exzellenz und Vertrauen 2. In dem Weißbuch legt sie die politischen Optionen dar, wie die Nutzung von KI gefördert und gleichzeitig die mit bestimmten Anwendungen dieser Technologie verbundenen Risiken eingedämmt werden können. Dieser Vorschlag zielt darauf ab, einen Rechtsrahmen für eine vertrauenswürdige KI zu schaffen, damit das zweite Ziel für den Aufbau eines KI-Ökosystems für Vertrauen umgesetzt werden kann. Der Vorschlag beruht auf den Werten und Grundrechten der EU und will erreichen, dass Privatpersonen und andere Nutzer KI-gestützten Lösungen vertrauen und gleichzeitig Unternehmen Anreize erhalten, diese zu entwickeln. KI sollte ein Instrument sein, das als positive Kraft für die Gesellschaft im Dienst der Menschen steht und das letztlich zu einem größeren Wohlbefinden der Menschen beiträgt. Vorschriften für KI, die auf dem Unionsmarkt verfügbar ist oder anderweitig Menschen in der Union beeinflusst, sollten daher auf den Menschen ausgerichtet sein, damit Menschen darauf vertrauen können, dass die Technik sicher angewandt wird und den Gesetzen, auch den Grundrechten, genügt. Nach Veröffentlichung des Weißbuchs leitete die Kommission eine breit angelegte Konsultation der Interessenträger ein, die reges Interesse zeigten und sich in großer Zahl beteiligten und die weitestgehend regulatorische Maßnahmen zur Bewältigung der Herausforderungen und Bedenken, die der zunehmende Einsatz von KI mit sich bringt, beantworteten.

---
```

Abbildung 11 Erstes Chunking

#### 4.1.3 Erste Schritte mit Embedding

Es gibt viele Modelle mit denen Embedding (deut. Einbetten) von Daten durchgeführt werden kann. Die Embedding-Modelle erstellen eine Vektorendarstellung von Text, dadurch wird es ermöglicht, semantische Suchen im Vektorenraum durchzuführen und den nächstähnlichen Vektor zu finden. Dieser Vektor kann anschließend in Text umgewandelt werden, um die Textdarstellung weiter zu nutzen.

In diesem Beispiel werden Zeichenketten in Vektoren-Form überführt, die dabei erzeugten Vektoren haben eine Dimension von 1536. Das Folgende Bild zeigt einen Ausschnitt eines Vektors an:

```
PS D:\Dev\Proj\rag\firstSteps> python .\embeddingsIntro.py
4 1536
[0.004034862853586674, 0.014992037788033485, -0.0028921770863234997, -0.022519579157233238, -0.024499183520674706, 0.017349909
991025925, -0.013252003118395805, 0.002033192664384842, 0.011423706077039242, -0.009689975529909134, 0.03149715065956116, 0.01
084369421005249, 0.008176902309060097, -0.005828485824167728, -0.00773558858782053, -0.012123501859605312, 0.04279476404190063
5, -0.010868911631405354, -0.002575377468019724, -0.0047598774544894695, -0.015925100073218346, 0.009078441187739372, 0.003075
0068835914135, 0.006518825422972441, -0.011921758763492107, -0.021725215017795563, 0.0038930124137550592, -0.02370481938123703
, 0.010219551622867584, 0.006225667428225279, 0.024146132171154022, -0.015899881720542908, -0.016429457813501358, -0.030488433
```

Abbildung 12 Vektor-Embedding Ausschnitt

Da die Vektorendarstellungen genutzt werden können, um mittels mathematischer Operationen Ähnlichkeitssuchen durchzuführen, wird dies beispielhaft mit den Vektoren der Texte „car“ und „truck“ sowie „car“ und „human“ durchgeführt. Dabei stellt ein kleinerer Wert eine höhere Ähnlichkeit dar. Das Folgende Bild zeigt die Ähnlichkeitswerte auf:

```
car and truck: {'score': 0.12827686108379288}
car and human: {'score': 0.18288189366165575}
```

Abbildung 13 Beispiel Ähnlichkeitswert

Laut der Ausgabe sind „car“ und „truck“ sich ähnlicher als „car“ und „human“.

#### 4.1.4 Erste Schritte mit Vektordatenbanken

Mit dem Aufkommen von Vektorendarstellungen ist die Notwendigkeit für eine effiziente Speicherung und Handhabung von Vektordarstellungen entstanden. In den letzten Jahren sind daher vermehrt spezialisierte Vektordatenbanken entwickelt worden. LangChain nutzt eine standardisierte Schnittstelle zur Kommunikation mit diesen Datenbanken, was die Interaktion sowohl mit lokalen als auch Cloud-basierten Datenbanken vereinfacht. Ein großer Vorteil, der sich dadurch ergibt, ist die hohe Modularität, da man problemlos die grundlegende Datenbank austauschen kann, ohne den Quellcode anpassen zu müssen. Dieses ermöglicht eine flexible Anpassung an unterschiedliche Anforderungen und verbessert die Wartbarkeit und Erweiterbarkeit des Systems erheblich.

Um die Implementierung und die Kommunikation mit Vektordatenbanken über LangChain zu testen, wurde eine Vektordatenbank mit den Chunks des EU-AI-Acts befüllt. Anschließend wurde eine semantische Suche auf dieser Datenbank durchgeführt. Diese Vorgehensweise hat es ermöglicht einen Retriever zu bauen. In produktiven RAG-Systemen besitzen Retriever meist komplexere Funktionalitäten als nur den semantisch ähnlichsten Chunk zurückzugeben. Sie können beispielsweise mehrere relevante Chunks kombinieren, Dokumentenverknüpfungen herstellen oder Kontextinformationen nutzen, um die Genauigkeit und Relevanz der Ergebnisse zu verbessern. Durch den Einsatz von LangChain in Kombination mit Vektordatenbanken kann die Effizienz und die Qualität unserer semantischen Suchprozesse erheblich gesteigert werden. Dies stellt sicher, dass das System nicht nur aktuelle Anforderungen erfüllt, sondern auch flexibel genug ist, um sich an zukünftige Entwicklungen und Erweiterungen anzupassen. Der modulare Aufbau und die Möglichkeit, verschiedene Datenbanken einfach zu integrieren, machen LangChain zu einem äußerst leistungsfähigen Werkzeug für moderne KI-Anwendungen.

#### 4.1.5 Erster MVP

Da eine ausreichende Wissensbasis aufgebaut wurde, um die notwendigen Ketten, die Datenaufbereitung und das RAG-Hauptsystem zu realisieren, wurde mit der Umsetzung dieser Komponenten begonnen.

Damit die Datenaufbereitung funktioniert, besteht die Kette aus folgenden Schritten:

1. Dokumente beziehen: In diesem Fall werden lokal gespeicherte Dateien verwendet.
2. Dokumente zerteilen: Hierfür wurde ein rekursiver Ansatz gewählt, um die Dokumente in kleinere Teile zu zerlegen.
3. Dokumente in Vektorform darstellen und in der Datenbank persistieren: Hierbei wird Chroma als Vektor-Datenbank verwendet und das OpenAI Embedding-Modell zur Vektorisierung der Dokumente genutzt.

Die Hauptkette des RAG-Systems hat den folgenden Aufbau:

1. Argumente entgegennehmen: Das System nimmt eine Nutzerfrage entgegen.
2. Semantische Suche: Die Nutzerfrage wird in der Vektordatenbank durchsucht, um relevante Dokumententeile zu finden.
3. Prompt zusammenbauen: Ein Prompt wird erstellt, der die gefundenen Dokumententeile als Kontext enthält.
4. Prompt an ein LLM übergeben: Der zusammengesetzte Prompt wird an ein Large Language Model übergeben, um eine kontextbezogene Antwort zu erhalten.
5. Antwort ausgeben: Die generierte Antwort wird ausgegeben.

Nach Ausführung der Datenaufbereitungskette befinden sich die Daten aufgeteilt und in numerischer Form in der Datenbank. Dies ermöglicht es, über die Hauptkette Anfragen zu stellen, die durch die gespeicherten Daten kontextbezogen angereichert werden.

## 5 Umsetzung

### 5.1 Backend

Nach den ersten Gehversuchen mit den neuen Technologien wurde eine Backend-Anwendung entworfen, um Retrieval-Augmented-Generation komfortabel nutzen zu können. Um einen realitätsnahen Prototypen vorstellen zu können, wurde sich dazu entschieden ein nutzerbasiertes System zu bauen. Ein Nutzer soll die Möglichkeit haben neue Chats zu erstellen und kann dadurch separate Anfragen an das System stellen. Da die grundsätzliche Struktur von RAG aus unterschiedlichen Komponenten besteht, welche modular ausgetauscht werden können, soll eine Konfigurierungsmöglichkeit dieser eingebaut werden.

#### 5.1.1 Relationales Datenbanksystem

Um eine relationale Datenbank anzusprechen, wird SQLAlchemy verwendet. SQLAlchemy ist eine Python-Bibliothek für Object Relational Mapping, diese Technik ermöglicht es Objekte in einer Programmiersprache mit den Daten in einer relationalen Datenbank zu verknüpfen, dadurch wird der Datenzugriff und die Speicherung vereinfacht. Die Anwendung nutzt MySQL für die Verwaltung von Nutzerdaten, den entstandenen Chats und dem Dokumentenhandling. Hauptgründe für die Nutzung von MySQL ist das schnelle und unkomplizierte Einrichten, sowie zusätzliche Administrationstools, wie PhpMyAdmin. Dies ist eine browserbasierte Schnittstelle zum DBMS, um die dort gespeicherten Elemente zu observieren und zu manipulieren.<sup>24</sup>

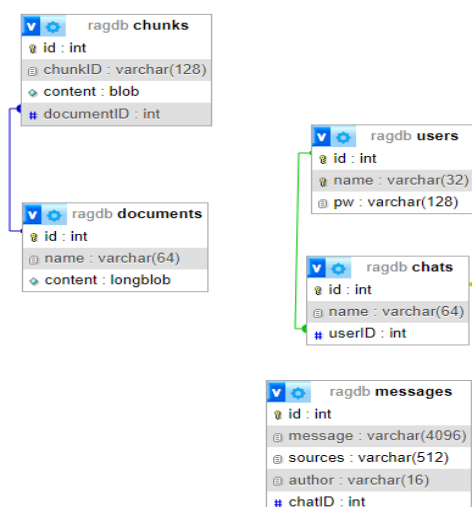


Abbildung 14 Abbildung des Datenbankmodells

Konkret gibt es Folgende Datenbanktabellen:

- **User:** Zuständig für die Speicherung von Benutzerdaten, wie der einzigartige Nutzernamen und ein gehashtes Passwort.
- **Chat:** Besitzt Chatnamen und zugehörigen Nutzer.

<sup>24</sup> (SQLAlchemy)



- **Message:** Speichert den Inhalt, sowie den Verfasser einer Nachricht. Falls Quellen bei der Inhaltserstellung genutzt wurden, werden die ChunkIDs in Textform in einer JSON-Liste gespeichert. Außerdem existiert eine Referenz zu einem Chat.
- **Document:** Speichert ein Dokument in Binärform mit dem zugehörigen Dokumentennamen.
- **Chunk:** Teilstück eines Dokuments, gespeichert in Binärform. Besitzt entsprechend eine Referenz zu einem Dokument und speichert die ChunkID, welche ein aus dokumentspezifischen Parametern, wie der Seitenzahl bei einer .pdf-Datei, zusammengesetzter Name ist.

Die Tabellen werden in der Datenbank erstellt, sofern sie noch nicht existieren, wodurch eine komfortable Nutzung mit lokalen aber auch separat betriebenen Datenbanken ermöglicht wird. Dieses Setup ermöglicht eine robuste und skalierbare Datenverwaltung für die RAG-Anwendung. So können Benutzerinteraktionen und Datenbereitstellung flexibel und unkompliziert genutzt werden.

#### 5.1.2 Vektordatenbanksystem

Chroma ist ein Tool, das für den Umgang mit Vektordatenbanken verwendet wird und wegen einer hohen Beliebtheit und umfangreicher Dokumentation eine angenehme Lernkurve bietet. Es ermöglicht die effiziente Speicherung, Suche und Verwaltung von Vektordaten durch das Einbinden von maschinellem Lernen und anderen fortgeschrittenen Analysemethoden. Durch die Vektordatenbank können hochdimensionale Daten in einem eingebetteten Raum gespeichert werden, was in dieser Anwendung für die Ähnlichkeitssuche von Anfragen zu Dokumenten genutzt wird. Für die Einbettung von Daten zu Vektoren wird ein Embedding-Modell genutzt, welches durch das modulare Design flexibel ausgetauscht werden kann. Diese Struktur bietet eine robuste Basis für die RAG-Anwendung, die eine schnelle und genaue Verarbeitung und Wiederauffindung von spezifischen Datensätzen erfordert. Neben den reinen Vektoren wird als Metadatum zu jedem Vektor eine ID gespeichert, diese ID entspricht der ChunkID eines Chunks in der relationalen Datenbank. Es wird also der Name des ursprünglichen Dokuments mit weiteren spezifischen Parametern wie der Seitenzahl gespeichert.

#### 5.1.3 Config

Da ein RAG-System eine hohe Modularisierbarkeit besitzt, wird eine zentrale Konfiguration erstellt, in welcher alle notwendigen Parameter einfach angepasst werden können. Dies soll eine einheitliche und effiziente Verwaltung der Einstellungen ermöglichen und gleichzeitig die Wartung und Erweiterung des Systems erleichtern. Um dieses Ziel zu erreichen, wird eine Python-Datei erstellt, in der alle notwendigen Attribute definiert werden. Diese Datei kann dann von verschiedenen Komponenten des Systems wie der Datenaufbereitung, und dem Hauptsystem importiert werden.

Die zentrale Konfigurationsdatei bietet mehrere Vorteile. Sie reduziert die Redundanz, indem sie sicherstellt, dass alle Komponenten auf die gleichen Parameter zugreifen, und erleichtert Änderungen, da Anpassungen an einer zentralen Stelle vorgenommen werden können. Dies fördert eine konsistente und fehlerfreie Konfiguration des gesamten Systems.



In Der Konfigurationsdatei werden folgende Dinge eingestellt:

- Pfad zur relationalen Datenbank
- Parameter für die Ähnlichkeitssuche (Anzahl der Elemente)
- Hashing-Algorithmus
- Gültigkeitsdauer des JSON-Web-Token
- LangChain API-Key
- Konfigurationsparameter für Datei-Splitter
- Embedding-Modelle (Cohere und lokales-Modell nicht funktionsfähig implementiert)
- Sprachmodelle
- Sprachmodell-Temperatur
- Vektordatenbanken (Pinecone nicht funktionsfähig implementiert)
- Auswahl einer RAG-Optimierung
- Prompt-Vorlagen

Diese zentrale Konfigurationsdatei stellt sicher, dass alle Komponenten des RAG-Systems nahtlos und effizient zusammenarbeiten können, indem sie eine einheitliche Quelle für alle notwendigen Einstellungen bereitstellt. Dies trägt zur Stabilität und Erweiterbarkeit des Systems bei, da Änderungen und Erweiterungen an einer einzigen Stelle vorgenommen werden können, ohne dass alle beteiligten Komponenten einzeln angepasst werden müssen.

#### 5.1.4 Endpunkte

Die Hauptanwendung befindet sich in der **serve.py**-Datei, dort wird eine FastAPI-Anwendung erstellt mit dem Titel "Wissensmanagementsystem". Diese Anwendung ermöglicht die Verwaltung und den Abruf von Endpunkten durch verschiedene API-Routen, die in den Modulen **user**, **chat**, **config** und **document** definiert sind. Um Cross-Origin Resource Sharing (CORS) zu unterstützen, wird eine Middleware hinzugefügt, die Anfragen von beliebigen Ursprüngen, Methoden und Headern zulässt. CORS ist eine Sicherheitsfunktion, die es Webanwendungen ermöglicht, Ressourcen von einer anderen Domain als der eigenen sicher anzufordern und zu verwenden. Die Backendanwendung läuft auf localhost unter Port 9000.<sup>25</sup>

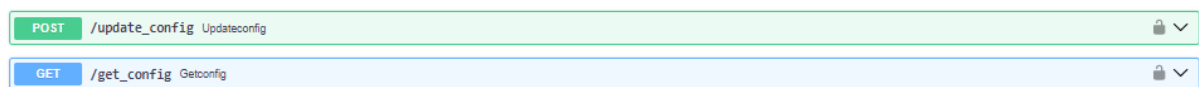
Das Modul **chat** definiert Routen zur Verwaltung von Konversationen mit dem RAG-System. Es ermöglicht das Hinzufügen, das Umbenennen und das Löschen von Chats sowie das Abrufen von benutzereigenen Chats. Die Hauptfunktion ermöglicht es, eine Nachricht an das RAG-System zu senden, welche mit der Antwort zu einem bestehenden Chat hinzugefügt wird. Wenn kein Chat vorhanden ist, wird ein neuer Chat erstellt, welcher die Nachrichten übergeben bekommt.

GET	/chat	Getmsgbychat	🔒
POST	/chat	Chat	🔒
POST	/rename_chat	Renamechat	🔒
GET	/user_chats	Getchatsbyuser	🔒
POST	/delete_chat	Deletechat	🔒
POST	/query	Queryendpoint	🔒

Abbildung 15 Chat-Endpunkte

<sup>25</sup> (FastAPI)

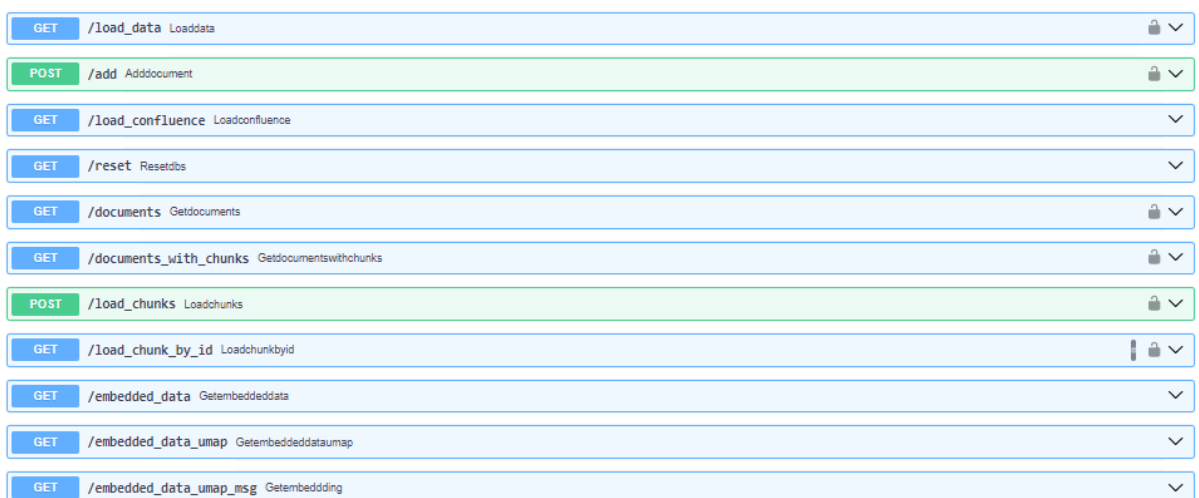
Im **config** Modul werden Endpunkte zur Verwaltung der Konfiguration definiert. Teile der aktuellen Konfigurationsdaten können hierüber abgerufen und aktualisiert werden. Konkret kann ausgewählt werden, ob und welche RAG-Optimierungs-Methode angewandt wird. Dazugehörend können die Prompt-Vorlagen der Methoden angepasst werden. Das Sprachmodell und die Temperatur, welche eine Art Kreativitätswert des LLMs darstellt, können eingestellt werden. Damit das Persistieren der Dokumente optimiert werden kann, ist die Größe und die mögliche Überlappung der Chunks auch anpassbar. Zudem kann die Anzahl der zu suchenden Chunks für die Anreicherung der Nutzeranfrage verändert werden. Darüber hinaus ist bereits eine Auswahl für unterschiedliche Vektordatenbanken und Embedding-Modelle vorhanden, jedoch noch nicht funktionsfähig, da sich während der Entwicklung auf die Fertigstellung eines funktionierenden Prototyps fokussiert wurde.



POST	/update_config	Updateconfig	🔒	▼
GET	/get_config	Getconfig	🔒	▼

Abbildung 16 Config-Endpunkte

In dem Modul **documents** werden Routen zur Verwaltung von Dokumenten und deren Chunks definiert. Es ermöglicht das Hochladen von Dokumenten sowie das Laden von Daten aus verschiedenen Quellen, wie lokal vorhandenen Dateien. Das Modul bietet umfassende Funktionen zur Verwaltung und Abfrage von Dokumenten, sowie deren in Chunks verwalteten Teilabschnitten. Darüber hinaus stellt **documents** Endpunkte bereit, die es ermöglichen, mit Vektordaten zu arbeiten und diese mittels Methoden wie PCA und UMAP in niedrigere Dimensionen zu überführen. Dadurch können die Vektordaten in eine visualisierbare Form gebracht werden. Dies unterstützt die Analyse und Interpretation der Dokumentenstruktur und der Inhalte, die in der Vektordatenbank persistiert sind.



GET	/load_data	Loaddata	🔒	▼
POST	/add	Adddocument	🔒	▼
GET	/load_confluence	Loadconfluence		▼
GET	/reset	Resetdbs		▼
GET	/documents	Getdocuments	🔒	▼
GET	/documents_with_chunks	Getdocumentswithchunks	🔒	▼
POST	/load_chunks	Loadchunks	🔒	▼
GET	/load_chunk_by_id	Loadchunkbyid	🔒	▼
GET	/embedded_data	Getembeddeddata		▼
GET	/embedded_data_umap	Getembeddeddataumap		▼
GET	/embedded_data_umap_msg	Getembedding		▼

Abbildung 17 Document-Endpunkte

Das Modul **user** behandelt die Benutzerverwaltung und Authentifizierung. Es verwendet JSON Web Token für die Authentifizierung und bietet Endpunkte zur Erstellung, Löschung und Abfrage von Benutzern. JWT ermöglicht eine sichere und effiziente Methode zur Authentifizierung und Autorisierung von Benutzern, indem es bei der Anmeldung ein Token generiert und zurückgibt, das für die Authentifizierung bei weiteren Anfragen verwendet wird.

POST	/token	Loginforaccesstoken	▼
POST	/user	Createuser	▼
POST	/user/delete	Deleteuser	🔒 ▼
GET	/users	Getusers	🔒 ▼
GET	/users/me	Readusersme	🔒 ▼

Abbildung 18 User-Endpunkte

Insgesamt bietet dieser Aufbau eine separierte und strukturierte Struktur zur Implementierung von RAG. So können die modularen Komponenten eine flexible und zudem robuste Nutzung ermöglichen. Alle Endpunkte können unter dem Pfad `/docs` gesichtet werden.

### 5.1.5 Core

Die wesentlichen Bestandteile der Anwendung, welche sich mit den Komponenten des Retrieval-Augmented Generation beschäftigen, sind innerhalb vom Ordner **core** anzufinden. Hier sind die Dateien **indexing.py** und **rag.py** angesiedelt. Die Datei **indexing.py** beinhaltet Funktionen zur Umwandlung und Speicherung von Dokumenten und Daten, damit diese in der Vektordatenbank gespeichert werden können. Durch **rag.py** wird die grundlegende Retrieval-Augmented-Generation Chain implementiert, um kontextangereicherte Anfragen an ein LLM zu senden und dessen Antwort verarbeiten zu können.

#### 5.1.5.1 Indexing

Die Datei **indexing.py** umfasst verschiedene Funktionen, die mit dem Laden, Aufteilen und Speichern von Dokumenten zu tun haben. Eine zentrale Funktion ist **loadData**, welche alle vorhandenen Dokumente aus der relationalen Datenbank lädt, sie verarbeitet und sie zur Speicherung in die Vektordatenbank weitergibt.

Eine weitere wichtige Funktion ist **persist**. Sie nimmt ein Dokument entgegen, teilt es in kleinere Chunks auf, erzeugt davon Vektordarstellungen und speichert diese in der Vektordatenbank. Hierbei kommt ein Text-Splitter zum Einsatz, der das Dokument in handhabbare Chunks unterteilt. Diese Chunks werden dann durch ein Embedding-Modell in eine Vektorform umgewandelt und der Datenbank gespeichert, wobei nur neue Chunks hinzugefügt werden, die noch nicht vorhanden sind.

Die Hilfsfunktion **getChunkIDs** sorgt dafür, dass jeder Chunk eine eindeutige ID erhält, die auf dem Namen und der Seite des grundlegenden Dokuments basiert. Diese IDs sind entscheidend für die spätere Identifikation der Chunks in der Vektordatenbank.

Zudem kann mit der Funktion **loadDocument** ein durch den Nutzer hochgeladenes Dokument in den Datenbanken gespeichert werden. Dabei können die Dateiformate PDF, Text und HTML genutzt werden, wobei je nach Dateityp eine entsprechende Loder-Komponente verwendet wird. Zuerst wird das neue Dokument in der relationalen Datenbank hinzugefügt, anschließend wird es in Chunks aufgeteilt und in der Vektordatenbank persistiert.

#### 5.1.5.2 RAG

Die Datei **rag.py** befasst sich mit der Anreicherung von Abfragen durch Chunks von Dokumenten für Retrieval-Augmented Generation.

Die Funktion **querySimple** implementiert klassisches RAG. Es wird auf Basis der Nutzeranfrage eine semantische Suche in der Vektordatenbank durchgeführt. Die dabei erhaltenen Chunks, welche eine hohe Ähnlichkeit mit der Anfrage besitzen, werden mit der eigentlichen Nutzeranfrage in eine Prompt-Vorlage gegeben. Der nun gefüllte Prompt wird an ein Sprachmodell gegeben, damit dieses eine möglichst qualitative Antwort für die Anfrage generieren kann. Da das Sprachmodell durch die Anreicherung der Chunks Zugriff auf einen größeren Kontext hat, kann sich die Antwort auf Quellen beziehen und nützlichere Antworten geben.

#### 5.1.5.3 Chat-Kontext

Ein gutes Beispiel für eine simple und doch merkbare Verbesserung für RAG ist, dem LLM den bereits angefallenen Chatverlauf mitzugeben. Mit dem vollständigen Chatverlauf kann das LLM präzisere und relevantere Antworten generieren, was die Benutzerzufriedenheit erhöht und die Anwendung effektiver macht. Da die Chats so konzipiert sind, dass Nachrichten persistiert werden, können diese problemlos bezogen und für das LLM passend angereichert werden. Die aufbereiteten Nachrichten werden in eine passende Prompt-Vorlage mit der neuen Nutzeranfrage gegeben. Dies bedeutet, dass die Generierung von Antworten nun mit dem angefallenem Chatverlauf erfolgt. Das LLM ist somit in der Lage, den gesamten Verlauf des Gesprächs berücksichtigen. Die Implementierung des Chat-Kontexts in die LLM-basierte Chat-Anwendung gestaltet sich daher überraschend einfach. Diese Vorgehensweise verbessert die Benutzererfahrung, da natürlichere und flüssigere Interaktion mit dem Sprachmodell ermöglicht wird.

#### 5.1.5.4 Query Translation

Query Translation im Kontext von Retrieval-Augmented Generation bezieht sich auf Techniken, die verwendet werden, um die ursprüngliche Anfrage in eine Form zu übersetzen, die für die Informationsanreicherung optimiert ist. Diese Übersetzung kann verschiedene Formen annehmen, und etwa durch Umschreiben oder Zerlegen der Nutzeranfrage erfolgen. Ziel dabei ist, die relevantesten und genauesten Informationen zur Anfrage abzurufen, damit die qualitativste Antwort generiert werden kann. In der Praxis bedeutet dies, dass das RAG-System die ursprüngliche Nutzeranfrage nimmt und sie so transformiert, dass sie besser mit den verfügbaren Datenquellen übereinstimmt. Mögliche Techniken, welche Query Translation umsetzen:

- Umschreiben der Anfrage: Die ursprüngliche Anfrage wird umformuliert, um Mehrdeutigkeiten zu beseitigen und den Fokus auf relevante Schlüsselbegriffe zu lenken.
- Zerlegen der Anfrage: Komplexe Anfragen werden in einfachere, spezifischere Teile aufgespalten, die jeweils leichter beantwortet werden können.
- Klärung der Anfrage: Unklare Anfragen werden präzisiert, indem zusätzliche Kontextinformationen hinzugefügt werden, um die Intention des Nutzers und das Problem besser zu erfassen.

Diese Techniken tragen dazu bei, dass die abgerufenen Informationen und die generierten Antworten sowohl genauer als auch kontextuell relevanter sind.<sup>26 27</sup>

Multi Query ist eine Methode, welche der Query Translation zugeordnet ist, die verschiedene Perspektiven oder Aspekte einer Anfrage nutzt, um umfassendere und vielfältigere Ergebnisse zu erzielen. Im Gegensatz zu einer einzigen Abfrage, die nur einen Blickwinkel abdeckt, ermöglicht Multi Query eine tiefere und breitere Durchsuchung der verfügbaren Dokumententeile. Dies erhöht die Wahrscheinlichkeit, relevante Teile zu finden, die dann zur Generierung einer präzisen und umfassenden Antwort durch ein Sprachmodell genutzt werden können.

Für die Implementierung in ein RAG-System müssen folgende Punkte durchgeführt werden:

1. Abfragen multiplizieren: Aus der Nutzeranfrage müssen weitere oder ähnliche Anfragen generiert werden.
2. Chunk-Retrieval auf allen Anfragen anwenden.
3. Einzigartige Chunks identifizieren.
4. Antwortgenerierung mit anreichern aller gegebenen Chunks.

In dem entwickelten System übernimmt die Funktion **multiQuery** das Vervielfachen der Nutzeranfrage. Mithilfe von **retrievalChain** werden die ähnlichsten Chunks der generierten Anfragen bezogen. Die Funktion **getUniqueDocs** sorgt dafür, dass aus der Liste der gegebenen Chunks die Duplikate entfernt werden, so dass nur einzigartige Chunks für die Weiterverarbeitung verwendet werden. Abschließend wird mit der Funktion **queryMulti** die eigentliche Antwort auf die Nutzeranfrage mit der angereicherten Menge an Chunks generiert.<sup>28</sup>

### 5.1.5.5 HyDE

Eine mögliche Methode den Kontext für Retrieval-Augmented Generation zu optimieren, nennt sich Hypothetical Document Embeddings (deut. Hypothetische Dokumenteinbettungen). Anstatt vorhandene Stichworte aus der Anfrage bzw. die Anfrage selbst zu nutzen, um eine semantische Suche durchzuführen, erstellt HyDE künstliche Chunks basierend auf der Nutzeranfrage und führt damit eine Suche durch. Diese Methode verbessert die Genauigkeit beim Abrufen von ähnlichen Chunks aus der Vektordatenbank, da nicht nur eine Frage mit den Chunks verglichen wird, sondern ein künstlich erzeugter Chunk mit der tatsächlichen Datenmenge. Im Kontext von Retrieval-Augmented Generation (RAG) bedeutet dies, dass HyDE genutzt werden kann, um präziser relevante Dokumententeile zu identifizieren, die dann für die Generierung von Antworten verwendet werden. Dadurch wird die Qualität der abgerufenen Informationen und die Relevanz der generierten Inhalte gesteigert.<sup>29</sup>

---

<sup>26</sup> (Parti, 2024)

<sup>27</sup> (Technology, 2024)

<sup>28</sup> (langchain, rag-from-scratch, 2024)

<sup>29</sup> (Gao, 2022)

## 5.2 Frontend

Nachdem die ersten Erfahrungen mit den noch unbekannten Techniken und Frameworks gesammelt wurden, wurde sich mit der Realisierung der Nutzeroberfläche beschäftigt. Beim Frontend wurde der Fokus auf die Performance und die Benutzerfreundlichkeit gelegt. Das Interface wurde so gestaltet, dass es auch bei umfangreicheren Datenmengen reibungslos funktioniert. Durch den Einsatz von Vue.js kann sichergestellt werden, dass die Anwendung auf Desktop- als auch auf mobilen Geräten optimal läuft. Insgesamt bietet das Frontend eine robuste und flexible Oberfläche für die Nutzung von Retrieval-Augmented Generation.

### 5.2.1 Grundlegender Aufbau

Die Frontend-Anwendung des Wissensmanagementsystems ist in einzelnen Komponenten unterteilt, um eine modulare Architektur zu ermöglichen. Dadurch kann die Steuerung des RAG-Systems in Chat Form, die Nutzerverwaltung und die Veranschaulichung der Datengrundlage, optimal separiert werden. Damit die diversen Parameter und Konfigurationseinstellungen des RAG-Systems bei der Bedienung vorgenommen werden können, wurde eine Komponente entworfen, in der diese Einstellungen in Form eines Formulars verändert werden können. Zudem existiert eine Infoseite, dort sind Log-Einträge zu finden, welche bei der Kommunikation mit dem Backend-System auftreten.

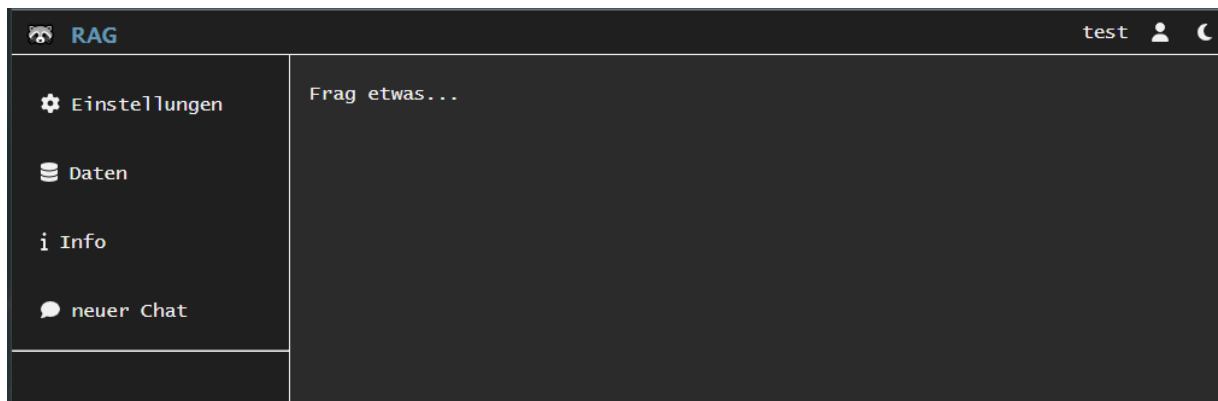


Abbildung 19 Layout der Frontend-Anwendung

Die Anwendung besitzt ein Header-Element am oberen Ende des Sichtfeldes. In diesem Header befindet sich an der linken Seite ein Logo und der Schriftzug „RAG“. Auf der rechten Seite des Headers ist die User-Komponente und ein Knopf zum Umschalten zwischen heller und dunkler Darstellung. Damit eine problemlose Navigation zwischen den einzelnen Seiten ermöglicht wird, wurde ein Menu-Element gebaut, welches sich an der linken Seite befindet. Dieses Navigationselement ermöglicht es den Benutzern, nahtlos zwischen den verschiedenen Seiten der Anwendung zu wechseln. Der übrige Bereich ist dafür da, die einzelnen Komponenten der Anwendungen zu bedienen. Dies erleichtert den Zugriff auf Navigationselemente und erhöht somit die Bedienfreundlichkeit und Effizienz der Systeminteraktion.

Die Kommunikation mit dem Backend erfolgt mit Axios. Diese JavaScript-Bibliothek ermöglicht ein komfortables Senden von HTTP-Anfragen. Zudem wird die Authentifizierung der Anfragen durch JSON-Web-Token sichergestellt, dieser wird im LocalStorage des Browsers gespeichert und bei jeder Anfrage in den Anfragen-Header geschrieben.

Dadurch kann der Server die Anfrage bei gültigem Token verarbeiten und bei ungültigem ignorieren. Diese Methode garantiert eine sichere und effiziente Datenübertragung und gewährleistet die Integrität der Benutzer.<sup>30</sup>

Um innerhalb der Frontwend-Anwendung einen komponentenübergreifenden Datenpool nutzen zu können, wurde ein Vuex Store verwendet. Dieser Verwaltet die Daten zentral, so dass von überall in der Anwendung auf die dort hinterlegten Chats, den angemeldeten Nutzer und die entstandenen Logs zugegriffen werden kann.<sup>31</sup>

### 5.2.2 Styling

Um ein anschaulichen Prototypen vorstellen zu können, wurde sich für ein minimalistisches und intuitives Design entschieden. Anwendungsübergreifendes Styling ist in der **main.css**-Datei beschrieben. Das grundlegende Design wurde von den einzelnen Komponenten je nach Bedarf durch ein lokales Styling erweitert und teilweise überschrieben. Damit die Benutzerfreundlichkeit erhöht wird, werden Icons genutzt, welche zum Beispiel im Menü und in der User-Komponente anzufinden sind. Die Icons stammen vom Anbieter FontAwesome, welcher eine umfangreiche und einfach zu integrierende Sammlung an Icons zur Verfügung stellt.<sup>32</sup>



Abbildung 20 Theme-wechsel-Icons

Das Design des Frontends ermöglicht eine Auswahl zwischen einem hellen und einem dunklen Thema, die über ein intuitives Steuerelement oben rechts einfach zu wechseln sind.

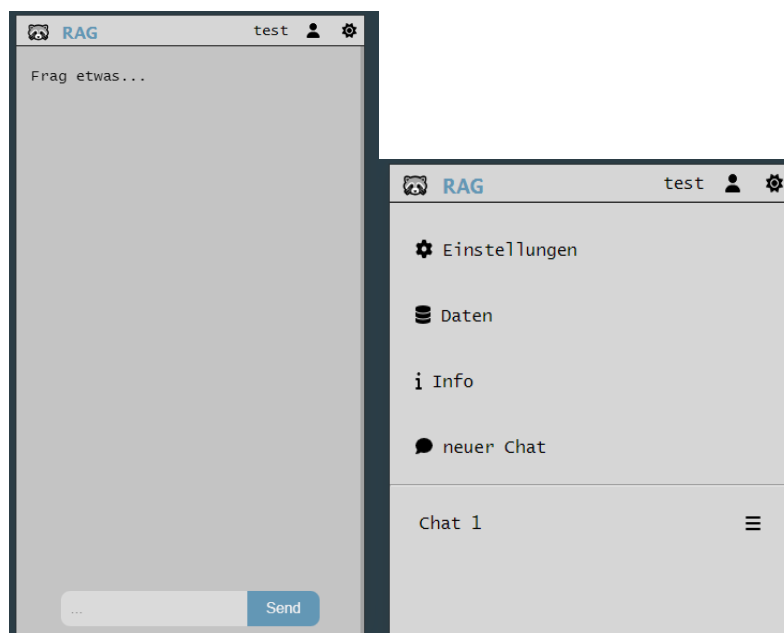


Abbildung 21 Mobile Darstellung der Anwendung

<sup>30</sup> (axios)

<sup>31</sup> (vuex)

<sup>32</sup> (fontawesome)

Das responsive Design verwendet ein flex-box-Layout, das durch minimale Verwendung von Media-Queries sowohl für Desktop- als auch für Mobilgeräte optimiert ist. Die Möglichkeit, das Menü durch einfaches Klicken auf das Logo oder den Schriftzug ein und auszublenden, verbessert die Benutzerfreundlichkeit und unterstützt die nahtlose Navigation.

### 5.2.3 User

Die User-Komponente ist hauptsächlich dafür verantwortlich, eine Authentifizierung am Backendsystem vorzunehmen. Dafür wird vor der eigentlichen Nutzung der Anwendung der Nutzernamen und das zugehörige Passwort abgefragt. Falls noch kein Nutzer vorhanden ist, kann ein neuer Nutzer angelegt werden.

Nachdem eine Anfrage an das Backendsystem verschickt wurde und anschließend geprüft worden ist, erhält der Nutzer einen zeitlich begrenzten Authentifizierungstoken, welche im LocalStorage des Browsers gespeichert wird. Nach einer Anmeldung wird der Name des Nutzers neben dem User-Icon angezeigt.

Diese Funktionalität der Anwendung ist im Kontext des Prototyps nur von demonstrativer Natur und soll die Integrierbarkeit vom Nutzermanagement aufzeigen. Da eine umfangreichere Nutzer Funktionalität mit einem zugehörigen Rechte System und einem robusteren Authentifizierungsmechanismus keine Relevanz für den RAG-Part der Anwendung darstellt, wird sich in dieser Arbeit nicht tiefer mit dem Nutzermanagement auseinandergesetzt.

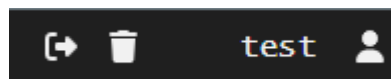


Abbildung 22 User-Komponente

Im Prototypen ist noch die Funktion einen eingeloggten Nutzer, auszuloggen und löschen zu können, vorhanden. Diese Funktionen können über die entsprechenden Knöpfe ausgeführt werden, welche durch Drücken des Nutzer-Icons sichtbar werden.

### 5.2.4 Config

Zusätzlich gibt es eine Konfigurationsseite, auf dem der Nutzer verschiedene Einstellungen zu den eingesetzten Komponenten des zugrundeliegendes RAG-Systems vornehmen kann. Die Konfigurationsseite ermöglicht den Nutzern, spezifische Einstellungen vorzunehmen. Dies umfasst die Selektion und Konfiguration der Modelle, konkret dem genutzten Embedding-Modell und dem Sprachmodell. Des Weiteren kann ein Vektordatenbanksystem ausgewählt werden, in dem die zugrundeliegende Datenmenge in Form von Vektoren persistiert wird. Da für die Umwandlung von Dokumenten in eine Vektorform Chunking notwendig ist, können die dafür notwendigen Parameter wie die Größe der Chunks oder die Größe der vorhandenen Überlappung, konfiguriert werden. Zudem werden bei einer semantischen Suche in der Vektordatenbank die ähnlichsten Ergebnisse zurückgegeben, die Anzahl wird über den k-Wert angegeben. Auch der Kreativitätswert des Sprachmodells, auch bekannt als Temperatur, kann verändert werden.

Da beim Arbeiten mit Sprachmodellen Prompts und entsprechende Prompt-Vorlagen verwendet werden, ist eine Anpassung dieser ziemlich nützlich. Diese können auch in der Konfigurationsseite angepasst werden, um so ein schnelles und möglichst komfortables Testen und Probieren von unterschiedlichen Prompts zu ermöglichen.



**Sprachmodell:** openai

**Embedding-Model:** openai

**Vektor-DB:** chroma

**rag-Methode:** simple

**Prompt:**  
Du bist ein Assistent zur Beantwortung von Fragen im Bezug auf Versicherungsbedingungen. Damit du genaue Informationen geben kannst, werden dir Teile aus Dokumenten zur Verfügung gestellt. Dafür wird die Nutzeranfrage verarbeitet und eine Ähnlichkeitssuche in den Dokumenten gemacht. Diese Dokumente beinhalten die Bedingungen unterschiedlicher Produkte und unterschiedlicher Generationen.

**MultiQuery-Prompt:**  
Du bist ein KI-Sprachmodell-Assistent. Deine Aufgabe ist es, fünf verschiedene Versionen der gegebenen Benutzerfrage zu generieren, um relevante Dokumente aus einer Vektordatenbank zu finden. Ziel ist es einige der Einschränkungen der entfernungsbasierten Ähnlichkeitssuche zu überwinden, indem du mehrere Perspektiven auf die Frage des Benutzers erzeugst. Gib diese alternativen Fragen durch Zeilenumbrüche getrennt aus.

**HyDE-Prompt:**  
Schreibe einen Abschnitt wie er in Versicherungsbedingungen stehen könnte, um die folgende Frage zu beantworten  
Frage: {question}  
Abschnitt:

**LLM-Kreativität**  
0

**Chunk-Anzahl bei Suche**  
3

**Chunk-Größe:**  
800

**Chunk-Overlap:**  
80

Abbildung 23 Config-Komponente

### 5.2.5 Chat

Die Hauptkomponente der Anwendung ist die Chat-Seite, welche es dem Nutzer ermöglicht, Anfragen an das RAG-System zu stellen. Diese Seite bietet den Nutzern die Möglichkeit einen Dialog zu führen, um Informationen abzurufen und kontextspezifische Fragen zu stellen. Das Chat-Interface wurde intuitiv gestaltet, um eine einfache und effektive Interaktion zu gewährleisten.

Das Styling des Chat-Interfaces wurde wie eine klassische Messenger-Anwendung aufgebaut. Das Eingabefeld für Anfragen befindet sich zentral unten mit einem Knopf, um die Anfrage zu versenden. Eigene Nachrichten befinden sich rechts orientiert und haben einen dunklen Hintergrund, die Antworten sind links orientiert mit hellerem Hintergrund. Die Antworten des Systems werden zusammen mit den entsprechenden Quellenangaben im relationalen DBMS gespeichert. Durch Drücken auf eine bestimmte Quellenangabe, wird diese Unter der Antwort des Systems angezeigt. Dadurch kann der Nutzer direkt nachvollziehen, wie die Antwort des Systems zustande gekommen ist.

Einer der skizzierten Anwendungsfälle des Systems ist der Einsatz als Assistent für Gesellschaftsspiele. Dem System sollen Anleitungen zugeführt werden, um auf dessen Basis Anfragen beantworten zu können. Eine mögliche Anfrage eines Nutzers könnte lauten „Wie komme ich bei monopoly aus dem gefängnis“:

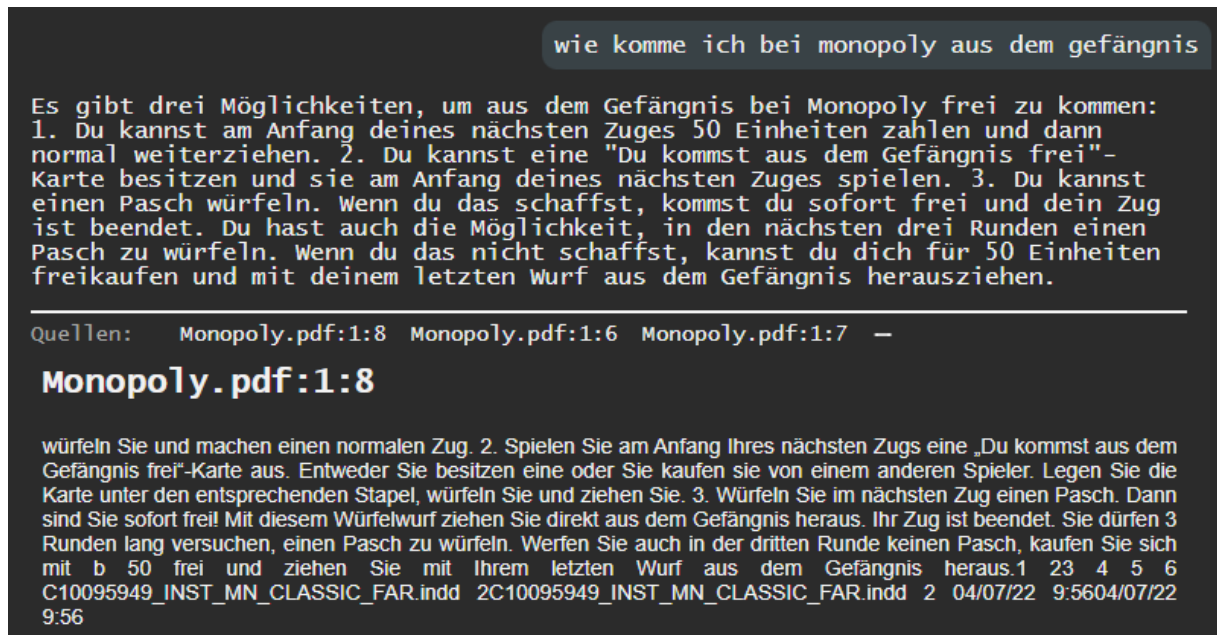


Abbildung 24 Chat-Komponente

Das System gibt die korrekte Antwort und nutzt für die Beantwortung die dafür notwendigen Informationen aus den Chunks der Dokumente. Durch Senden einer Nachricht an das System wird ein neuer Chat erstellt und unter der Menüleiste hinzugefügt. Dort kann der Name des Chats durch einen Klick auf die drei Striche neben dem Chat-Namen geändert werden. Außerdem kann hier der Chat wieder gelöscht werden.

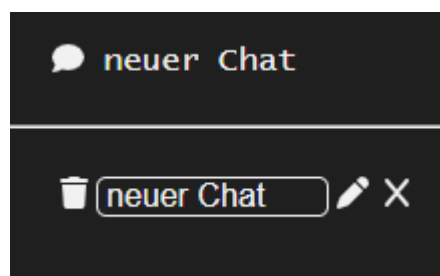


Abbildung 25 Interaktionsmöglichkeiten mit einem Chat-Element

#### 5.2.6 Data

Ein weiteres wichtiges Feature der Anwendung ist die Ansicht aller dem System zur Verfügung stehenden Dokumente. Diese Seite ermöglicht es den Nutzern, einen Überblick über alle vorhandenen Dokumente zu erhalten. Hierbei können auch die einzelnen Chunks in denen das Dokument aufgeteilt wurde, eingesehen werden. Bei Bedarf können über den Knopf „Datei auswählen“ über den Dateixplorer neue Dokumente ausgewählt werden, welche durch den Knopf „Upload“ in das System hinzugefügt werden. Vorteilhaft ist dieses Feature für die kontinuierliche Erweiterung und Aktualisierung der Wissensbasis des Systems.

Datenansicht wechseln

ID	Name	
15	BED_IR_01_2022.pdf	Show Chunks
16	BED_IR_01_2023.pdf	Show Chunks
17	BED_RL_01_2022.pdf	Show Chunks

Datei auswählen Upload

Abbildung 26 Ansicht der Daten in der SQL-DB

Eine andere Sicht auf die Daten kann durch den Knopf „Datenansicht wechseln“ aufgerufen werden. Diese Ansicht zeigt eine Vektordarstellung der Chunks in einem dreidimensionalen Raum. Für die Darstellung wurde die JavaScript-Bibliothek Plotly verwendet. Da die Chunks in der Vektordatenbank als hochdimensionale Vektoren gespeichert sind, ist eine Dimensionsreduktion notwendig. Mit dem Knopf „Dimensionsreduzierung wechseln“ kann zwischen den Methoden PCA und UMAP für die Reduktion gewechselt werden.<sup>33</sup>

Falls vor dem Aufruf der Daten-Komponente eine Chat-Anfrage durchgeführt wurde, werden die angegebenen Quellen der Antwort automatisch durch größere und in Rot dargestellten Punkten markiert. Zusätzlich werden die Quellen über der Darstellung aufgeführt. Mit einem Mausklick auf einen Datenpunkt kann der entsprechende Textinhalt des Chunks angezeigt werden. Alle aus einem Dokument erzeugten Chunks besitzen in der Ansicht dieselbe Farbe, um die Zugehörigkeit der Daten darzustellen.

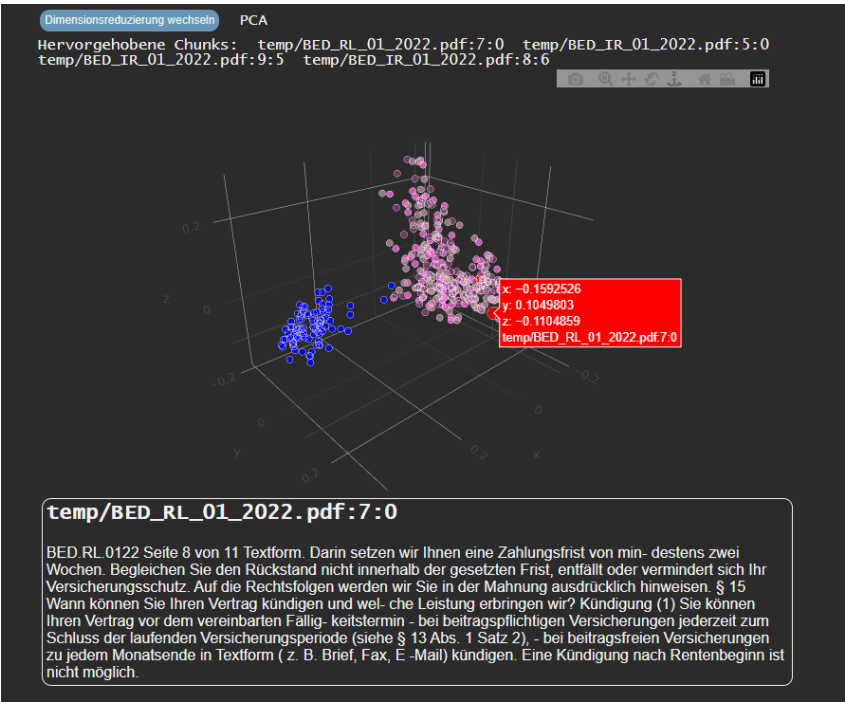


Abbildung 27 Ansicht der Vektordaten (PCA)

<sup>33</sup> (Plotly )

Diese Datenansicht ist eine gute Möglichkeit, um ein Gefühl für die in der Vektordatenbank gespeicherten Daten zu bekommen. So kann sich der Nutzer durch die visuelle Darstellung die semantische Nähe zwischen den einzelnen Chunks aufzeigen lassen.

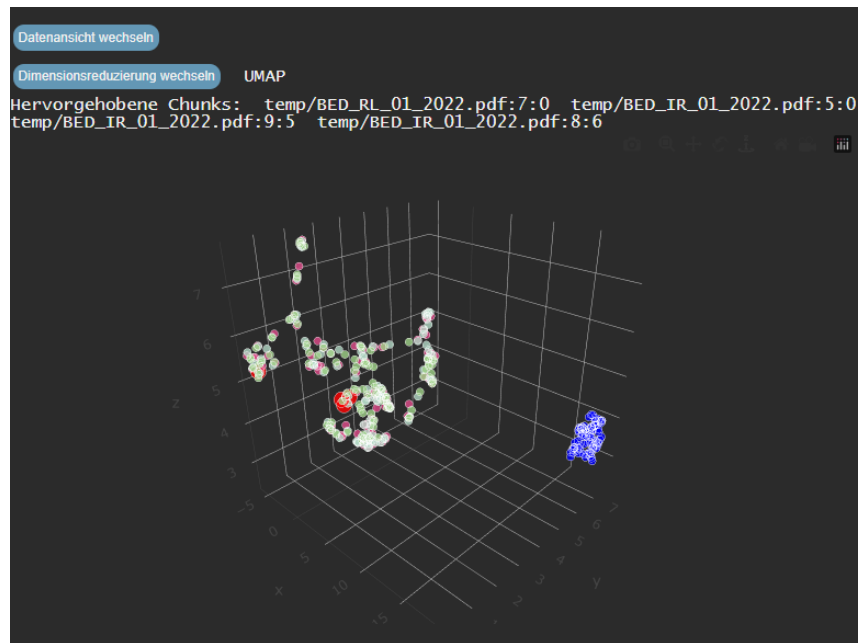


Abbildung 28 Ansicht der Vektordaten (UMAP)

#### 5.2.7 Info

Auf der Info-Seite befindet sich eine Art Logging-Mechanismus benutzt, um Informationen zu den Anfragen an das Backendsystem aufzuführen. Hintergrund ist hauptsächlich ein besseres Verständnis über die Systeminteraktion zu erhalten. Zusätzlich kann dieses Logging während der Entwicklung und Erweiterung als ein komfortables Debugging-Werkzeug verwendet werden.

### 5.3 Virtualisierung

Um das Frontend, das Backend und die Datenbanken zu betreiben, wurde sich dazu entschieden, Virtualisierung zu nutzen. Die Virtualisierung mittels Docker bietet eine Vielzahl an Vorteilen, die moderne Softwareentwicklung und -bereitstellung erheblich erleichtern. Ein großer Vorteil ist die Portabilität der Docker-Container. Die Docker-Container laufen unabhängig vom zugrundeliegenden Betriebssystem und der Infrastruktur, was die Migration und Skalierung der Anwendung über verschiedene Umgebungen hinweg erleichtert. Das bedeutet, dass eine Anwendung, die auf einem lokalen Entwicklungssystem funktioniert, auch in der Produktionsumgebung reibungslos läuft, was die Entwicklungs- und Betriebsprozesse vereinheitlicht und vereinfacht. Zusammengefasst fördert Docker die Effizienz, Flexibilität und Sicherheit in der Softwareentwicklung und -bereitstellung und bietet somit eine moderne und leistungsfähige Lösung für die Herausforderungen beim Entwickeln und Betreiben.<sup>34</sup>

---

<sup>34</sup> (Docker)

### 5.3.1 Datenbanken

Die Konfiguration erfolgt über eine Docker-Compose-Datei, in dieser werden drei Dienste definiert: ein MySQL-Datenbankserver, phpMyAdmin als Verwaltungsoberfläche für den MySQL-Server und eine Chroma-Vektordatenbank.

Der MySQL-Dienst ist als db benannt und verwendet das neueste MySQL-Image. Dieser Dienst wird so konfiguriert, dass er im Fehlerfall automatisch neu startet. Es wird eine Datenbank ragdb erstellt, und ein Benutzer mit den Anmeldeinformationen angelegt. Der Dienst ist auf dem Port 3306 verfügbar. Zusätzlich wird ein phpMyAdmin-Dienst bereitgestellt, um die Verwaltung der MySQL-Datenbank zu erleichtern. Dieser Dienst nutzt das Image phpmyadmin/phpmyadmin und wird ebenfalls so konfiguriert, dass er bei Bedarf automatisch neu startet. phpMyAdmin ist über den Port 8080 erreichbar. Der dritte Dienst in diesem Setup ist chromadb, welcher die Vektordatenbank Chroma in einem Container erzeugt. Dieser Dienst stellt eine weitere Datenbank bereit und ist auf Port 8000 verfügbar.

Um die Daten persistent zu speichern, wird ein Docker-Volumen verwendet. Dies stellt sicher, dass die Daten auch dann erhalten bleiben, wenn der Container neu gestartet oder entfernt wird. Dieses Docker-Setup ermöglicht somit die Bereitstellung einer MySQL-Datenbank, die durch phpMyAdmin verwaltet werden kann, sowie einer zusätzlichen Chroma-Datenbank. Alle Dienste laufen in isolierten Docker-Containern, was eine reproduzierbare, leicht verwaltbare und skalierbare Umgebung für die Anwendung schafft.

### 5.3.2 Anwendungen

Damit die Frontend- und Backend-Anwendungen auch über die Docker-Compose-Datei genutzt werden können, benötigen diese jeweils eigene Dockerfiles, um als virtualisierter Service genutzt werden zu können.

Die Backend-Anwendung ist verantwortlich für die Verarbeitung und Speicherung von Daten in der MySQL-Datenbank und der Chroma-Datenbank. Sie stellt API-Endpunkte für die Frontend-Anwendung sowie andere mögliche Clients bereit und implementiert die Kernlogik des RAG-Systems. Das Backend ist als backend-Dienst benannt und wartet die Erreichbarkeit der Datenbanken ab, bevor es hochfährt und unter Port 9000 erreichbar ist.

Die Frontend-Anwendung wird als frontend-Dienst bereitgestellt und wartet die Erreichbarkeit des Backend-Dienstes ab, bevor es zum Starten kommt. Verantwortlich ist dieser Dienst für die Bereitstellung der clientseitige Benutzeroberfläche, die mit der Backend-Anwendung und den Datenbanken interagiert. Innerhalb des Docker-Container wird ein Nginx-Webserver verwendet, um die Vue.js App zu betreiben. Der Dienst ist nach dem erfolgreichen Starten über den Port 80, welcher für Webseiten genutzt wird, aufrufbar. Dieses Setup für einzelne Bestandteile dieser Anwendung sorgt für eine isolierte, reproduzierbare und skalierbare Umgebung, die leicht verwaltet werden kann.<sup>35</sup>

---

<sup>35</sup> (Nginx)

### 5.3.3 Modelle

Die Implementierung von Sprachmodellen und Embedding-Modellen in einer RAG-Anwendung kann auf verschiedene Arten erfolgen. Zu den gängigsten Möglichkeiten gehört das Betreiben der Modelle lokal oder in einer Cloud. Außerdem gibt es die Möglichkeiten über API-Aufrufe auf bereitgestellte Modelle von Dritten zuzugreifen.

OpenAI bietet die Möglichkeit auf Sprachmodellen wie GPT-4 oder auf Embedding-Modelle, über eine API zuzugreifen. Diese Modelle werden in OpenAI-Rechenzentren betrieben, welche sich hauptsächlich in Amerika befinden. Dadurch existiert das Problem, dass diese durch rechtliche Hürden nicht in produktiven Anwendungen von europäischen Betreibern genutzt werden können. Grund ist das Thema Datenschutz, welches in europäischen Ländern strikter ist als im internationalen Vergleich. Einer der Vorteile auf diese Modelle zurückzugreifen ist die Qualität, welche einem geboten wird. Um diese Modelle zu nutzen, muss die OpenAI-API in der Anwendung integriert werden. Dafür muss ein API-Schlüssel bei OpenAI erstellt werden. Die Nutzung der Modelle ist hierbei auch kostenpflichtig, so dass auf dem OpenAI-Konto ein Guthaben hinterlegt sein muss. Für produktive Szenarien in Europa, in denen Datenschutz und geringe Latenzzeiten im Fokus stehen, bietet sich der Betrieb von lokalen Modellen an. Ollama ist eine Plattform, die den Betrieb von solchen Modellen auf eigener Hardware ermöglicht. So können lokale Modelle einfach installiert und betrieben werden. Diese Modelle können über Schnittstellen, welche von LLM-App-Frameworks bereitgestellt werden, in eigene Anwendungen integriert werden. Durch die Verwendung der lokalen Modelle werden die Daten nicht an externe geschickt und sind daher nicht abgreifbar. Das Thema Datenschutz kann so besser umgesetzt werden.

Eine weitere Möglichkeit zur Integration von Sprach- und Embedding-Modellen in einer Anwendung ist die Nutzung von Cloud-Diensten. Anbieter wie AWS, Google Cloud und Microsoft Azure ermöglichen das Nutzen von speziellen Ressourcen, welche für das Betreiben von leistungsstarken KI-Modelle genutzt werden können. Diese Cloud-Variante bietet eine flexible und skalierbare Lösung für den Einsatz von Modellen. Die in der Cloud betriebenen Modelle können dann über API-Anfragen angesprochen werden, um so in der Anwendung integriert werden zu können.

Das Nutzen einer Cloud-Lösung kann vor allem dann nützlich sein, wenn das Thema Skalierung im Fokus steht oder die lokale Infrastruktur nicht ausreicht, um die benötigte Rechenleistung bereitzustellen. Die Wahl der Integration hängt von den individuellen Anforderungen der Anwendung ab. Themen wie Datenschutz, Latenz, Skalierbarkeit und Kosten sollten bei der Auswahl berücksichtigt werden. Für diesen Prototypen wurde auf eine Integration von OpenAI gesetzt, um die qualitativsten Ergebnisse bei der Generierung zu erzeugen. Zudem ist das Betreiben von Embedding-Modellen ziemlich ressourcenintensiv, weshalb hier auch auf OpenAI gesetzt wurde. Um die Implementierung von lokalbetrieblenen Modellen zu testen, wurde sich auch mit der Nutzung von lokalen LLMs mit Ollama beschäftigt.

## 5.4 Tests

Pytest ist ein häufig verwendetes Test-Framework für Python, welches es ermöglicht, automatisierte Tests zu schreiben und auszuführen. Damit kann sichergestellt werden, dass Funktionen eines Softwaresystems nach einer Anpassung des Quellcodes wie erwartet

funktionieren. Dadurch wird die Gesamtqualität des Projekts verbessert und es kann frühzeitig im Entwicklungsprozess erkannt werden, ob sich Fehler eingeschlichen haben.<sup>36</sup>

Damit die Qualität der erzeugten Ausgaben des Backendsystems getestet werden können, werden Tests erstellt, welche basierend auf einer Anfrage eine erwartete Ausgabe überprüfen. Da es sehr unwahrscheinlich ist, dass die tatsächliche Ausgabe genau mit der erwarteten Ausgabe übereinstimmt, wird ein Sprachmodell zur Unterstützung genutzt. In den Tests werden sowohl die erwartete Antwort als auch die tatsächliche Antwort an das LLM weitergegeben. Das LLM soll dann beurteilen, ob die beiden Antworten semantisch dasselbe Ausdrücken. Dieses Vorgehen ermöglicht es, die Ausgaben des Systems nicht nur auf einfache Textgleichheit zu überprüfen, sondern auch auf inhaltliche Übereinstimmung. Dadurch kann sichergestellt werden, dass das System die gewünschten Ergebnisse liefert, selbst wenn die Formulierungen variieren.

Basierend auf der Bewertung des LLM wird eine Rückmeldung darüber gegeben, ob die Ausgabe des Systems den Erwartungen entspricht. Durch diese Rückmeldung kann festgestellt werden, ob das System einen gewissen qualitativen Standard erfüllt. Sollte die Bewertung des LLM anzeigen, dass die Ausgaben nicht semantisch gleichwertig sind, können gezielt Anpassungen und Verbesserungen am System vorgenommen werden. Aktuell werden die Tests nach Anpassung des Quellcodes händisch über die Commandozeile ausgeführt, bei einer produktiven Anwendung würde das Ausführen in den Deployment-Prozess aufgenommen werden.

Dieses Verfahren stellt sicher, dass die Qualität der erzeugten Ausgaben kontinuierlich überprüft werden. Die Verwendung eines LLM zur semantischen Bewertung bietet eine ungewohnte, aber hilfreiche Methode, um die Genauigkeit und Zuverlässigkeit des Systems zu gewährleisten. Da ein LLM über einfache Textvergleiche hinausgeht und inhaltliche Übereinstimmungen berücksichtigt, ist dies ein vielversprechender Einsatzfall. Auf diese Weise kann die Leistungsfähigkeit des Systems umfassend sichergestellt werden.

Mit den Folgenden Kombinationen aus Frage und erwarteter Antwort wurde die Anwendung während der Entwicklung getestet:

- **Vier-Gewinnt-Test**

**Frage:** Wie gewinne ich vier gewinnt?

**erwartete Antwort:** Um bei Vier Gewinnt zu gewinnen, musst du als erster Spieler vier Spielchips in einer Reihe platzieren.

- **Catan-Test**

**Frage:** Wie gewinne ich Catan?

**erwartete Antwort:** Um Catan zu gewinnen, musst du als erster Spieler 10 Siegpunkte erreichen. Du kannst Siegpunkte erhalten, indem du neue Straßen und Siedlungen baust und Siedlungen zu Städten erweiterst.

---

<sup>36</sup> (pytest)

- **Monopoly-Gefängnis-Test**

**Frage:** Wie komme ich bei Monopoly aus dem Gefängnis?

**erwartete Antwort:**

- 1. Du kannst am Anfang deines nächsten Zuges eine 'Du kommst aus dem Gefängnis frei'-Karte spielen, entweder indem du sie bereits besitzt oder indem du sie von einem anderen Spieler kaufst. Dann würfelst du und ziehst normal weiter.
- 2. Wenn du im nächsten Zug einen Pasch würfelst, kommst du sofort frei und ziehst aus dem Gefängnis heraus. Dein Zug ist dann beendet.
- 3. Du hast insgesamt drei Runden lang die Möglichkeit, einen Pasch zu würfeln und dich so freizukaufen. Wenn du nach drei Runden keinen Pasch würfelst, kannst du dich für 50 Einheiten freikaufen und mit deinem letzten Wurf aus dem Gefängnis herausziehen.

- **Negativ-Test** (Test soll absichtlich einen Fehler erzeugen)

**Frage:** Wie gewinne ich in Risiko

**erwartete Antwort:** Wenn du vier Spielchips in einer Reihe platzierst hast du gewonnen

## 5.5 Anwendungsfall Versicherungsassistent

Ein weiterer möglicher Anwendungsfall für ein RAG-System ist der Einsatz als Assistent für Vertragsbedingungen von Versicherungsprodukten. Damit realitätsnahe Anfragen genutzt werden können, wurde sich mit Kollegen aus der Abteilung Service-Center der VOLKSWOHL BUND VERSICHERUNGEN zusammengesetzt, um tatsächlich aufgetretene Anfragen von Nutzern zu sammeln. Darüber hinaus wurden die richtigen Antworten und der Abschnitt in der entsprechenden Bedingung, in dem die Antwort zu finden ist vermerkt. Damit kann zum einen die Qualität der System-Antworten sichergestellt werden und zum anderen kann damit fehlendes Fachwissen des Entwicklers ausgeglichen werden.

Bei der Interaktion mit dem System wurde die simple Implementierung von RAG verwendet, welche direkt auf Grundlage der Nutzeranfrage eine Ähnlichkeitssuche durchführt. Zum Zeitpunkt der Ausführung befanden sich folgende Dokumente in der Vektordatenbank:

ID	Name	
1	vier-gewinnt.pdf	Show Chunks
2	Risiko.pdf	Show Chunks
3	Monopoly.pdf	Show Chunks
4	catan.pdf	Show Chunks
5	BED_RL_01_2022.pdf	Show Chunks
6	EU_AI_ACT.txt	Show Chunks
7	BED_FR_01_2017.pdf	Show Chunks
8	BED_SBU_06_2022.pdf	Show Chunks
9	BED_FR_06_2022.pdf	Show Chunks
10	BED_SVL_01_2022.pdf	Show Chunks
11	BED_SBU_05_2010.pdf	Show Chunks

Abbildung 29 Datengrundlage



### 5.5.1 Fall 1

Im ersten Fall hat der Versicherungsnehmer einen seit 12 Monate beitragsfreien Vertrag einer fondsgebundenen Rentenversicherung aus dem Jahr 2017. Der VN fragt sich, ob er diesen Vertrag wieder aktivieren kann und ob dies mit Veränderungen verbunden ist.

Die korrekte Antwort ist, dass eine Wiederinkraftsetzung bis 3 Jahre nach Beitragsfreistellung möglich ist und sich zudem die Rechnungsgrundlage verändert, wenn die Beitragsfreistellung länger als 6 Monate her ist.

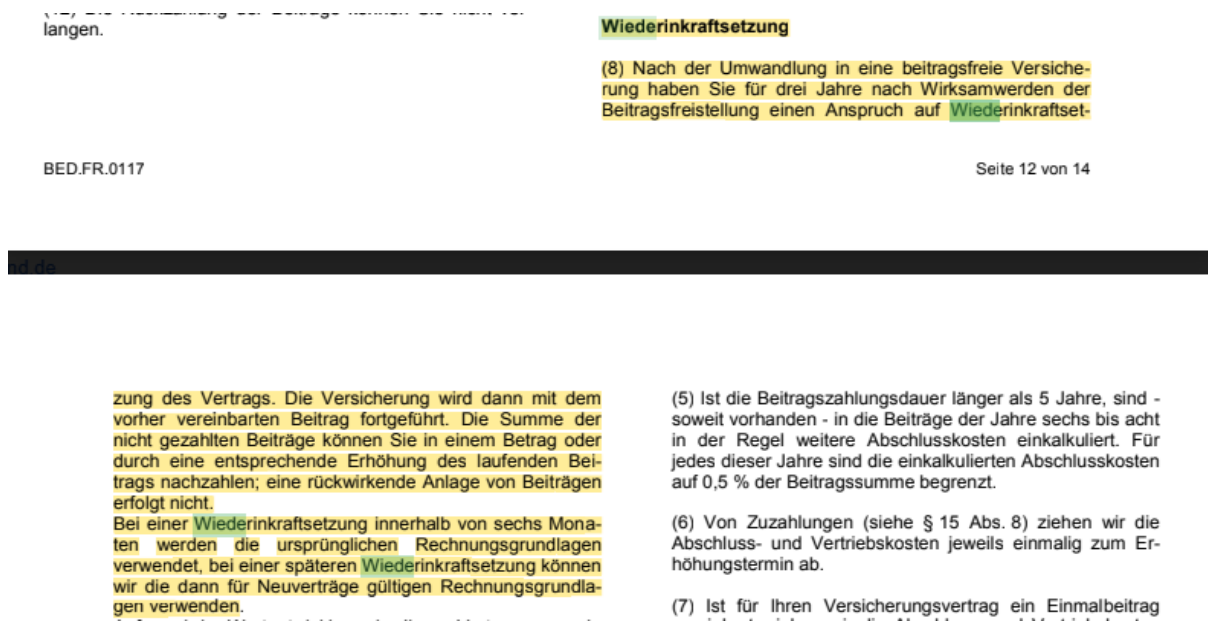


Abbildung 30 Ausschnitt Bedingung FR-2017

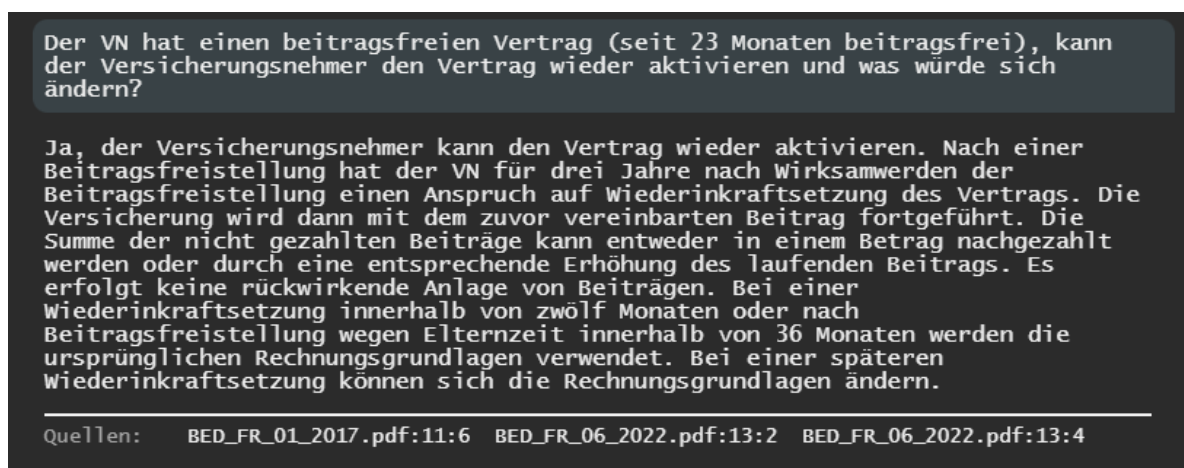


Abbildung 31 RAG-Antwort zum Fall 1

Die Antwort ist teilweise korrekt, beinhaltet jedoch Inhalt aus den Vertragsbedingungen der 2022er Generation, weshalb Informationen aufgeführt werden, die für den Versicherungsnehmer nicht zutreffend sind. Ein Grund dafür könnte sein, dass die Bedingungen aus den unterschiedlichen Generationen zu ähnlich sind und daher bei der Ähnlichkeitssuche eigentlich irrelevante Dokumententeile bezogen wurden.

Die Antwort auf die Frage ist durch den aktuell genutzten Indizierungsmechanismus nicht korrekt formatiert und in zwei Dokumententeilen zu finden, wodurch vermutet wird, dass die Ähnlichkeitssuche in diesem Fall nicht wie gewünscht funktioniert.

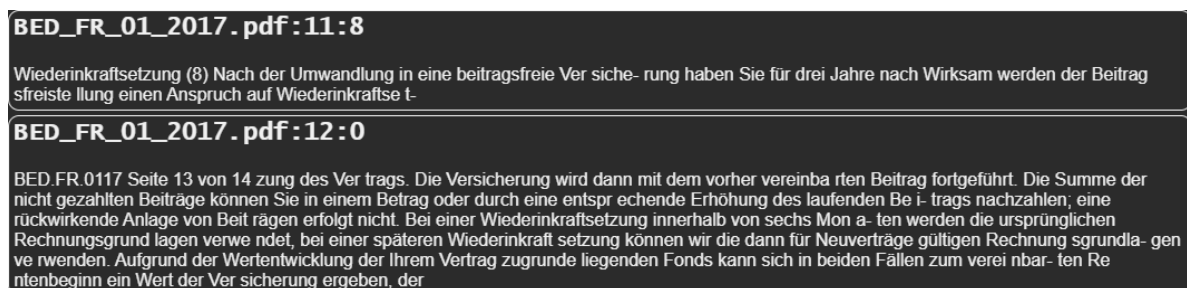


Abbildung 32 Chunks mit tatsächlichem Antwortinhalt

### 5.5.2 Fall 2

Im zweiten Fall besitzt der VN eine Berufsunfähigkeitsversicherung mit einer BU-Rente aus 2022 von 1.000€. Da der Versicherungsnehmer ein Kind bekommen hat, will er die BU-Rente gerne erhöhen und möchte wissen, ob dies möglich ist.

Da eine Nachversicherungsmöglichkeit gegeben ist, ist eine Erhöhung von maximal 500€ ohne erneute Gesundheitsprüfung möglich. Die Erhöhung muss innerhalb von 12 Monaten nach dem Ereignis erfolgen. Ansonsten ist einer Erhöhung mit erneuter Gesundheitsprüfung immer möglich.

#### **Nachversicherungsmöglichkeit bei Erhöhung des Einkommens**

(3) Sie haben das Recht, Ihre versicherte Berufsunfähigkeitsrente ohne erneute Gesundheitsprüfung zu erhöhen, wenn bei der versicherten Person ein Einkommenssprung stattgefunden hat. Unter einem Einkommenssprung verstehen wir, dass

- sich im Falle einer nichtselbstständigen Tätigkeit das regelmäßige jährliche Bruttoeinkommen um mindestens 10 % erhöht (z. B. durch Gehaltserhöhung, Beförderung, Wechsel des Arbeitgebers, Erhalt von Prokura) oder
- bei Selbstständigen der durchschnittliche Gewinn vor Steuern der letzten drei Jahre um mindestens 30 % höher ist als der durchschnittliche Gewinn des davorliegenden Dreijahreszeitraums.

Für die Erhöhungen der Berufsunfähigkeitsrente aufgrund von Einkommenssprüngen gelten folgende Grenzen:

- Durch einen Einkommenssprung darf die vereinbarte Gesamtmonatsrente aus allen bei uns bestehenden Berufsunfähigkeits-, Erwerbsunfähigkeits- und Grundfähigkeitsversicherungen maximal um 500 Euro erhöht werden.
- Bei mehreren Einkommenssprüngen sind auch mehrere Erhöhungen möglich, die Summe aller Erhöhungsbeträge darf jedoch bezogen auf die Monatsrente 1.500 Euro nicht überschreiten.
- Die Gesamtmonatsrente aus allen bei uns bestehenden Berufsunfähigkeits-, Erwerbsunfähigkeits- und Grundfähigkeitsversicherungen darf sich maximal auf 4.000 Euro erhöhen. Lag die Rente bereits zu Beginn Ihrer Versicherung über 2.500 Euro, ist eine Erhöhung auf maximal 6.000 Euro möglich.

#### **Nachversicherungsmöglichkeit bei besonderen Ereignissen**

(4) Sie haben das Recht, Ihre versicherte Berufsunfähigkeitsrente bei nachstehenden Ereignissen ohne erneute Gesundheitsprüfung zu erhöhen.

Die persönliche Situation der versicherten Person verändert sich durch:

- erstmalige Gründung eines eigenen Hausstandes,
- Erwerb und Finanzierung einer Immobilie mit einem Finanzierungsbetrag von mindestens 50.000 Euro,
- Heirat,
- **Geburt eines Kindes,**
- Adoption eines Kindes,
- Eintritt der Volljährigkeit,
- Scheidung bzw. Aufhebung einer eingetragenen Lebenspartnerschaft oder
- Tod des Ehepartners bzw. des eingetragenen Lebenspartners.

Die berufliche Situation der versicherten Person verändert sich durch:

- Wechsel aus einer mindestens ein Jahr laufenden sozialversicherungspflichtigen Teilzeittätigkeit in eine unbefristete Vollzeitstelle,
- Abschluss einer beruflichen Qualifikation, wie zum Beispiel Meisterbrief,
- Abschluss einer akademischen Weiterqualifizierung, wie zum Beispiel Facharzt Ausbildung, Promotion, Master, sofern die versicherte Person eine der Weiterqualifizierung entsprechende berufliche Tätigkeit ausübt,
- Abschluss einer beruflichen Qualifikation für in einem Kammerberuf selbstständig Tätige, wie zum Beispiel Fachanwalt, Wirtschaftsprüfer, oder
- Wechsel in die volle berufliche Selbstständigkeit.

Die Versorgungssituation der versicherten Person verändert sich durch:

- Ausscheiden aus der gesetzlichen Rentenversicherung, wenn die versicherte Person zum Beispiel als Handwerker die Mindestpflichtversicherungszeit erfüllt,
- erstmalige Überschreitung der Beitragsbemessungsgrenze in der gesetzlichen Rentenversicherung, sofern die versicherte Person in der gesetzlichen Rentenversicherung oder in einem berufsständischen Versorgungswerk versichert ist; oder
- Reduzierung oder Wegfall der Absicherung gegen Berufsunfähigkeit aus einer arbeitgeberfinanzierten betrieblichen Altersversorgung oder einem berufsständischen Versorgungswerk, in dem die versicherte Person aufgrund einer Kammerzugehörigkeit pflichtversichert ist.

Bei jedem der zuvor genannten Ereignisse darf die vereinbarte Gesamtmonatsrente aus allen bei uns bestehenden Berufsunfähigkeits-, Erwerbsunfähigkeits- und Grundfähigkeitsversicherungen maximal um 500 Euro auf maximal 2.500 Euro erhöht werden.

#### **Nachversicherungsmöglichkeit bei einer Änderung der Überschussbeteiligung**

(5) Sofern die Überschüsse für eine Bonusrente verwendet werden und diese Bonusrente durch eine Verminderung der Überschussanteile sinkt, haben Sie innerhalb von sechs Monaten das Recht, im bestehenden Vertrag die Berufsunfähigkeitsrente ohne erneute Gesundheitsprüfung bis zu der vor der Überschussenkung gültigen Höhe aufzustocken.

#### **Nachversicherungsmöglichkeit ohne besonderen Anlass**

(6) Haben Sie mit uns die Individuelle Nachversicherungsgarantie vereinbart, können Sie ohne besonderen Anlass innerhalb der ersten fünf Versicherungsjahre die vereinbarte Berufsunfähigkeitsrente ohne erneute Gesundheitsprüfung bis auf den vereinbarten Betrag anheben.

#### **Weitere Regelungen**

(7) Bei allen Erhöhungen mit Ausnahme der Erhöhungen im Rahmen von Absatz 5 muss die neue Berufsunfähigkeitsrente mindestens um 50 Euro monatlich höher als die vor der Erhöhung vereinbarte Berufsunfähigkeitsrente sein.

(8) Bei Erhöhungen im Rahmen der Absätze 2, 3, 4 und 6 darf die gesamte Anwartschaft auf Berufsunfähigkeits-, Erwerbsunfähigkeits- und Grundfähigkeitsrente einschließlich anderweitig bestehender privater und betrieblicher Anwartschaften nach der Erhöhung nicht mehr als 60 % des regelmäßigen jährlichen Bruttoeinkommens betragen. Hierzu können wir Einkommensnachweise verlangen. Bei Beamten darf ein Anteil von 25 % nicht überschritten werden.

(9) Eine Erhöhung ohne erneute Gesundheitsprüfung ist nur möglich, wenn bisher weder ein Versicherungsfall im Rahmen einer Berufsunfähigkeits-, Erwerbsunfähigkeits- oder Grundfähigkeitsversicherung eingetreten ist noch Leistungen aus einer dieser Versicherungen beantragt wurden. Hierzu gehören auch Leistungen wegen Arbeitsunfähigkeit.

(10) Die Nachversicherungsmöglichkeiten gemäß den Absätzen 1 bis 4 erlöschen, wenn die versicherte Person das 50. Lebensjahr vollendet hat, jedoch nicht innerhalb von fünf Jahren nach Versicherungsbeginn.

Abbildung 33 Ausschnitt Bedingung SBU-2022

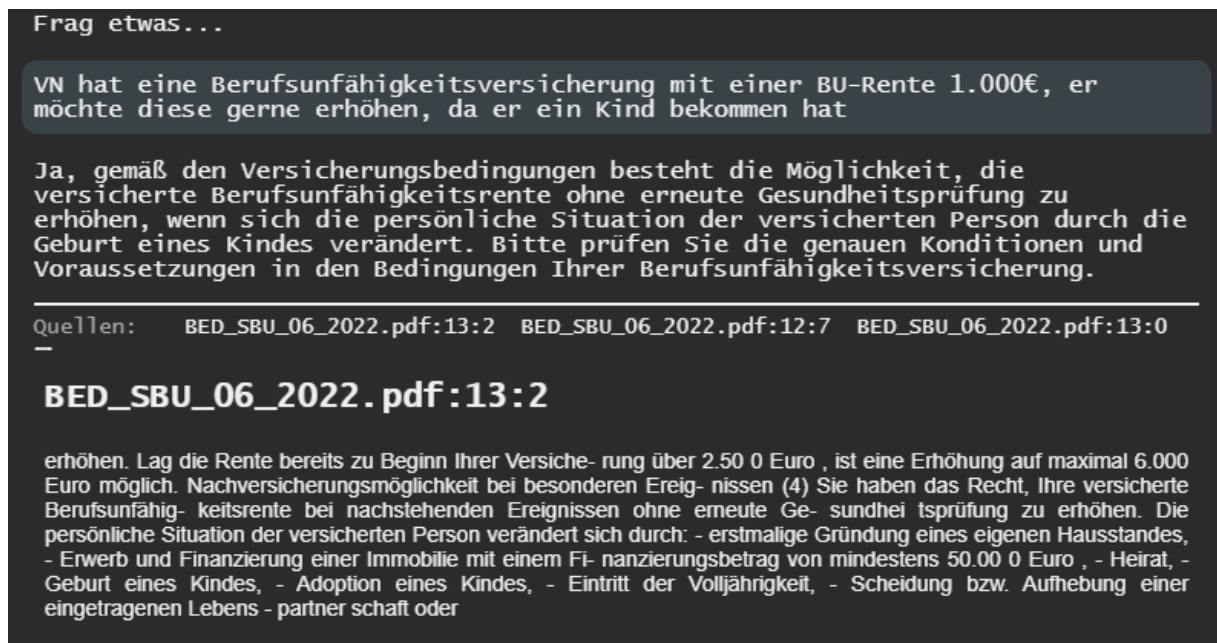


Abbildung 34 RAG-Antwort zum Fall 2

Bei dieser Antwort hat das System sich auf die dafür notwendigen Dokumententeile bezogen und konnte daraus eine korrekte Antwort generieren. Jedoch hat das System keine Angabe zum Betrag der möglichen Erhöhung gemacht.

### 5.5.3 Fall 3

Der dritte Fall bezieht sich auf eine fondsgebundene Rentenversicherung aus dem Jahr 2022. Der Versicherungsnehmer zahlt einen Betrag von 500€ jährlich und möchte diesen aufgrund von finanzieller Not herabsetzen.

Die Herabsetzung ist auf ein Minimum von 120€ jährlich möglich.

#### Herabsetzung des Beitrags

(5) Anstelle der Umwandlung in eine beitragsfreie Versiche- rung können Sie die Höhe der Beiträge reduzieren.

(6) Die Herabsetzung des Beitrags ist nur möglich, wenn der verbleibende Jahresbeitrag 120 Euro nicht unterschreitet.

(7) Bei Vereinbarung eines verminderten Anfangsbeitrags (§ 15 Abs. 3) entspricht eine Fortzahlung nur des verminderten Beitrags einer Beitragsherabsetzung. Alternativ kann der Zeitraum, für den der verminderte Anfangsbeitrag gilt, mit unserer Zustimmung auf maximal fünf Jahre verlängert werden. Umgekehrt können Sie diesen Zeitraum auch abkürzen.

Abbildung 35 Bedingung FR-2022

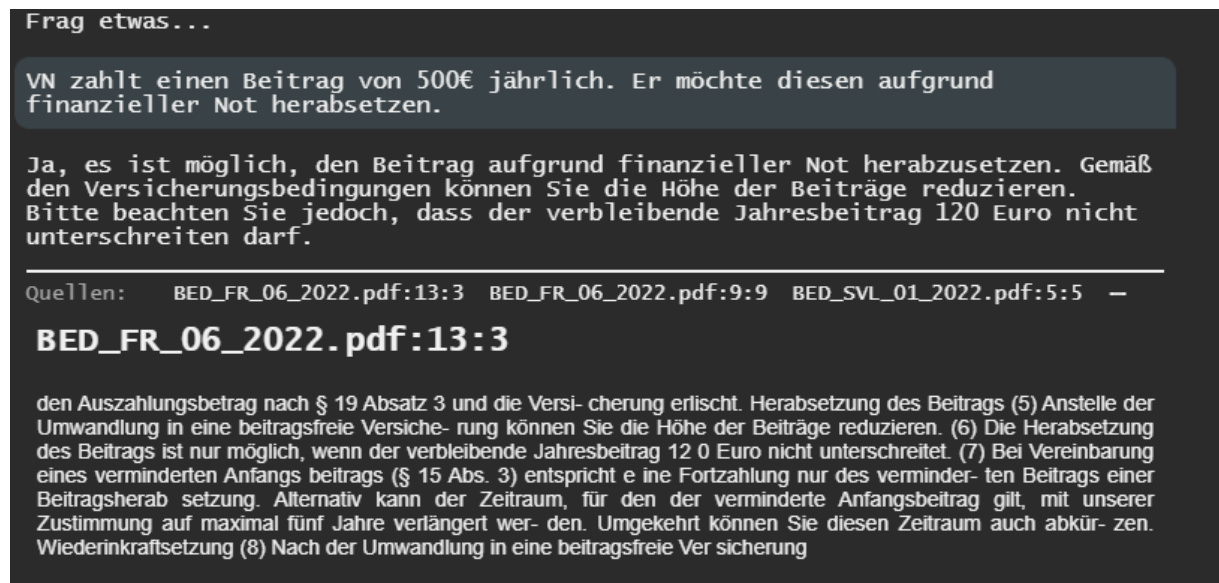


Abbildung 36 RAG-Antwort zum Fall 3

Das System generiert in diesem Fall die korrekte Antwort und nutzt dafür den richtigen Teil aus dem korrekten Dokument. Da die richtige Antwort in dem Dokument sehr kurz ist und das System die Anfrage mit drei Dokumententeile anreichern muss, wird die Anfrage mit Teilen angereichert, welche für die Beantwortung nicht relevant sind. Bei der Generierung der Antwort wurden diese Teile korrekterweise nicht berücksichtigt.



## 6 Ausblick

### 6.1 Offene Punkte

#### 6.1.1 Anwendungsfall Confluence-Suche

Ein vielversprechender Anwendungsfall, welcher mit diesem System angegangen werden könnte, ist die Verbesserung der Confluence-Suche. Confluence ist ein Tool zur Verwaltung und Organisation von Wissen. Der Einsatz ist häufig innerhalb von Unternehmen anzufinden. Da die Suche nach gezielten Informationen bei großen Confluence-Systemen erfahrungsgemäß schlecht funktioniert, bietet sich hier eine Möglichkeit, RAG zu nutzen. Dafür müsste das RAG-System Zugriff auf die Wissensbasis des Confluence-Systems haben, um diese in eine Vektordatenbank zu überführen. Im Rahmen dieser Arbeit wurde nicht weiter auf diesen Anwendungsfall eingegangen. Die zukünftige Weiterentwicklung des RAG-Prototyps wird sich weiterhin mit der Thematik Confluence-Suche beschäftigen.

#### 6.1.2 Änderungen am Quellcode

Das entwickelte Softwaresystem befindet sich derzeit im Prototypenstatus und benötigt noch verschiedene Verbesserungen und Ergänzungen, um in einen möglichen produktiven Einsatz zu kommen.

Dazu gehört die Implementierung einer Reset-Funktion, um die Inhalte der Vektor-Datenbank wie auch der SQL-Datenbank zurücksetzen. Momentan erfolgt das Bereinigen der Datengrundlage über manuelles entfernen des Datenbankspeichers. Dies würde die Wiederherstellung der Ausgangszustände erleichtern und unterstützt den Entwicklungs- und Anpassungsprozess von Features.

Die Architektur des Systems ist so gestaltet, dass weitere Embedding- und Sprach-Modelle sowie Vektordatenbanken einfach integriert und ausgetauscht werden können. Dadurch kann die bestmögliche Kombination aus Komponenten für den entsprechenden Anwendungsfall genutzt werden. Entsprechend müssen diese Systeme angebunden und verglichen werden, um das bestmögliche RAG-System zu verwirklichen.

Um die Zuverlässigkeit des Systems sicherzustellen, ist das Implementieren von Integrationstests aller API-Endpunkte ein weiterer offener Punkt. Diese Tests sollen überprüfen, ob die verschiedenen Komponenten des Systems nahtlos zusammenarbeiten und erwartungsgemäße Ergebnisse liefern. Dies ist vor allem bei der Implementierung neuer Features sehr vorteilhaft.

Das System soll benutzerfreundlicher gestaltet werden. Eine Möglichkeit dafür sind Feedback-Elemente, die den Nutzern bei erfolgreichem Hinzufügen von Dokumenten dies deutlich anzeigen. Dazu gehören auch Erklärungen zur Systembedienung und eine allgemeine Verbesserung der Benutzeroberfläche.

Derzeit werden Chat-Antworten als einfacher Text ausgegeben. Das Design soll so angepasst werden, dass vom Sprachmodell formatierte Antworten z.B. Aufzählungen auch in der Frontendanwendung dargestellt werden können. Dadurch wird die Lesbarkeit und Verständlichkeit der Antworten erheblich verbessert.

Die aktuelle Benutzerverwaltung ist minimal und müsste für einen produktiven Betrieb erweitert werden. Es sollte eine robuste Benutzerverwaltung implementiert oder angebunden werden, die Funktionen wie Rollen- und Rechteverwaltung und Benutzerprofile bietet.

Diese Verbesserungen und Erweiterungen sind entscheidend, um das System von einem Prototyp zu einer stabilen und produktiveinsetzbaren Softwarelösung weiterzuentwickeln. Darauf folgend müsste sich mit dem Thema Inbetriebnahme auf einem Server auseinandergesetzt werden.

Für den weiteren Einsatz im Unternehmenskontext müssten Anpassungen vorgenommen werden, um die Anwendung innerhalb eines Unternehmens nutzen zu können. Dies umfasst die Integration in die bestehende IT-Landschaft, die Berücksichtigung von Sicherheitsanforderungen und die Ausarbeitung von Anwendungsfällen.

### 6.1.3 Modularität

Die Effizienz und Qualität des RAG-Systems hängen direkt von den Arbeitsschritten, den verwendeten Komponenten und deren Konfiguration ab. Daher ist es wichtig, diese systematisch zu vergleichen und entsprechend zu evaluieren, um die bestmögliche Systemkonfiguration zu erhalten. Da die Anzahl an Komponenten und deren Konfigurationsmöglichkeiten sehr hoch ist und das Herausstellen der notwendigen Qualitätsmerkmale und Gewichtungen für einen Vergleich sich als sehr komplex ergeben hat, wurde in dieser Arbeit kein Vergleich durchgeführt.

Ein Arbeitsschritt ist das Chunking, dieser Prozess ist zuständig für die Dokumententeilung. Verschiedene Methoden des Chunkings müssen verglichen werden. Ein Vergleich jeder Methode, insbesondere hinsichtlich der Genauigkeit für die semantischen Suche und der Effizienz der Datenverarbeitung, ist notwendig. Ziel ist es, die geeignetste Methode für verschiedene Dokumenttypen und Anwendungsfälle zu identifizieren.

Die Embedding-Modelle sind notwendig für die Erstellung von Vektordarstellungen und damit grundlegend für RAG. Bei einem Vergleich würden Faktoren wie die benötigte Rechenleistung, die Qualität und die Verfügbarkeit eine Rolle spielen.

Die Wahl der Vektordatenbank beeinflusst die Effizienz der semantischen Suche maßgeblich. Ziel ist es, die Vektordatenbank zu identifizieren, die die besten Ergebnisse in Bezug auf Performance, Skalierbarkeit und Genauigkeit liefert.

Da die Qualität der generierten Antworten direkt von den verwendeten Prompts abhängt, ist die Optimierung der Prompts ein Ansatz, um das bestmögliche System zu erhalten. Eine systematische Evaluation verschiedener Prompts mit zugehörigen Antworten könnte Aufschluss über die optimalen Formulierungen geben.

Das Herzstück des RAG-Systems ist das Sprachmodell, welches für die Antwortgenerierung genutzt wird. Wichtige Kriterien bei einem Vergleich wären die Anpassungsfähigkeit der Modelle an spezifische Anforderungen, die anfallenden Betriebskosten sowie Performance und Integrationsfähigkeit.

## 6.2 Erkenntnisse

Bei der Entwicklung des Systems sind mehrere wichtige Erkenntnisse gewonnen worden, die sowohl mit der Vielseitigkeit als auch mit der Komplexität des Systems zu tun haben. Ein faszinierendes Merkmal von RAG ist seine vielfältig einsetzbare Grundstruktur. Diese ermöglicht es, das System für unterschiedliche Anwendungsfälle zu nutzen, von Suchen in Datenmengen über spezifische Assistent-Systeme bis hin zur einfachen Verbesserung von generativer KI. Durch die modulare Architektur können spezifische Anpassungen schnell und gezielt vorgenommen werden, um den jeweiligen Anforderungen gerecht zu werden. Ein RAG-System sollte daher nicht als starres Konstrukt betrachtet werden, sondern vielmehr als dynamische Plattform, die je nach Bedarf optimiert und erweitert werden kann. Diese Anpassungsfähigkeit ist ein Vorteil, um die Leistungsfähigkeit und Effizienz des Systems kontinuierlich zu verbessern. Die Vielseitigkeit des Systems bringt jedoch auch eine erhöhte Komplexität mit sich. Viele Komponenten müssen nahtlos zusammenarbeiten, um die gewünschten Ergebnisse zu erzielen. Die Vielzahl an Komponenten kann daher bei fehlerhafter Konfiguration zu unerwünschtem Verhalten führen, weshalb diese sorgfältig getestet werden muss.

Ein zentraler Leitsatz bei der Entwicklung des RAG-Systems war „Probieren geht über Studieren“. Im Vergleich zu starren algorithmischen Abläufen, in denen die Ausgabe vor Durchführung bereits bekannt oder zumindest herleitbar ist, ist dies bei LLM-basierten Anwendungen nicht der Fall. Da die Ausgabe vor Durchlauf des Systems nicht genau bekannt war, hatte sich der Entwicklungsprozess spannender gestaltet als bei einem klassischen Softwaresystem. Oft konnte erst nach dem Testen von neuen Features oder Abläufen abgeschätzt werden, ob die Implementierung den gewünschten Effekt hatte. Entsprechend wurde während der Entwicklung ein iterativer Ansatz verfolgt, bei dem durch kontinuierliches Testen nach Quellcodeanpassungen das gewünschte Verhalten geprüft wurde.

RAG besitzt eine Vielzahl an Optimierungsmöglichkeiten, davon wurden in dieser Arbeit Query Translation und HyDE implementiert. Dabei hängt die Effektivität der Optimierung mit dem konkreten Anwendungsfall zusammen, für welcher RAG verwendet werden soll. Es hat sich ergeben, dass bestimmte Optimierungen für bestimmte Anwendungsfälle sehr gute Ergebnisse liefern und dieselbe Optimierung für einen anderen Anwendungsfall die Ergebnisse teilweise sogar verschlechtern kann. Durch Ausprobieren und Vergleichen der Ergebnisse mit und ohne implementierter RAG-Optimierung kann abgeschätzt werden, ob Verbesserungen für einen bestimmten Anwendungsfall erzielt werden.

Eine umfangreiche und qualitative Datengrundlage ist essenziell für die Qualität des RAG-Systems. Je nach Anwendungsfall hat sich gezeigt, dass die wichtigsten Faktoren für gute Ergebnisse die Relevanz der Daten, ihre Struktur und der vorhandene Informationsgehalt sind. Entsprechend muss je nach Anwendungsfall die Datengrundlage gezielt ausgewählt und für die genaue Systemkonfiguration aufbereitet werden, um die qualitativsten Ergebnisse zu erzielen.

Die Nutzung von OpenAI-Modellen ist mit Kosten verbunden, die je nach Umfang der Nutzung variieren können. Im Rahmen der Entwicklung des Prototyps sind Kosten von 2,16 US-Dollar für die Verwendung des Embedding- und Sprach-Modells aufgekommen. Insgesamt wurden dem Sprachmodell ca. 800 Tausend Tokens zugeführt. Dem Embedding-Modell wurden ca. 8,8 Millionen Tokens zugeführt.



## 7 Fazit

Die vorliegende Arbeit hat sich intensiv mit der Entwicklung und Implementierung von Retrieval-Augmented Generation beschäftigt, welche verschiedene fortschrittliche Technologien integriert. Ziel war es, ein System zu schaffen, das durch die Kombination von semantischer Suche und generativer KI qualitativ hochwertige und kontextspezifische Antworten liefern kann. Das RAG-System nutzt dazu eine modulare Architektur, die flexible Anpassungen und Erweiterungen ermöglicht.

Zu Beginn wurde eine Analyse und Auswahl der geeigneten Technologien durchgeführt. Es wurde sich mit dem LLM-App-Framework LangChain auseinandergesetzt und deren Verwendung erlernt, um eine solide Basis für die Implementierung von RAG zu haben. Darauf aufbauend wurden die Grundkonzepte von RAG implementiert. Nach der Planung des Gesamtsystems wurde dieses umgesetzt und anhand von beispielhaften Anwendungsfällen getestet.

Ein geplantes Element war die Implementierung eines User-Interfaces, das eine intuitive und benutzerfreundliche Interaktion mit dem Backend-System ermöglicht. Das Backend-System wurde auf Grundlage der neu erlernten Fähigkeiten umgesetzt. Damit ein Gesamtpaket entsteht, wurde die Anwendung für den Betrieb virtualisiert.

Die Entwicklung eines leistungsfähigen RAG-Systems ist durch den Einsatz moderner Technologien und durch die gezielte Anpassung auf einen Anwendungsfall bezogen möglich. Die erzielten Ergebnisse sind vielversprechend und bieten eine solide Grundlage für zukünftige Weiterentwicklungen. Durch die Kombination von semantischer Suche und generativer KI können qualitativ hochwertige und kontextspezifische Antworten geliefert werden, die in verschiedenen Anwendungsszenarien einen hohen Mehrwert bieten. Die flexible und modulare Architektur des Systems ermöglicht eine kontinuierliche Anpassung und Erweiterung, um unterschiedlichen Anforderungen gerecht zu werden. Insgesamt stellt das entwickelte RAG-System einen robusten und funktionsfähigen Prototyp dar. Die gewonnenen Erkenntnisse und die entwickelte Technologie bieten eine vielversprechende Grundlage für zukünftige Forschungs- und Entwicklungsprojekte im Bereich der KI basierten Wissensmanagementsysteme.

Der Quellcode des Prototyps ist unter folgenden GitHub-Repository zu finden:

<https://github.com/PatrickPerezPiktus/RAG>

## 8 Literaturverzeichnis

1. Ali, Z. (2023). *Comparison of LlamaIndex and LangChain*. <https://medium.com/@zahidali133/comparison-of-llamaindex-and-langchain-4900989752ac>. abgerufen am 21.07.2024
2. axios. (kein Datum). <https://axios-http.com/docs/intro>. abgerufen am 21.07.2024
3. Cohere. (2024). *Cohere-Embedding*. <https://dashboard.cohere.com/>. abgerufen am 21.07.2024
4. deepset. (2024). *haystack*. <https://haystack.deepset.ai/overview/quick-start>. abgerufen am 21.07.2024
5. Def. KI - Fraunhofer. (2023). Von Fraunhofer: <https://www.iks.fraunhofer.de/de/themen/kuenstliche-intelligenz.html> abgerufen am 21.07.2024
6. Demir, N. (2023). *Hands-On with RAG: Step-by-Step Guide to Integrating Retrieval Augmented Generation in LLMs*. <https://blog.demir.io/hands-on-with-rag-step-by-step-guide-to-integrating-retrieval-augmented-generation-in-llms-ac3cb075ab6f>. abgerufen am 21.07.2024
7. Docker. (kein Datum). <https://www.docker.com/>. abgerufen am 21.07.2024
8. Ertel, W. (2021). *Grundkurs Künstliche Intelligenz*.
9. FastAPI. (kein Datum). <https://fastapi.tiangolo.com/>. abgerufen am 21.07.2024
10. fontawesome. (kein Datum). <https://fontawesome.com/>. abgerufen am 21.07.2024
11. Gao, L. (2022). *Precise Zero-Shot Dense Retrieval without Relevance Labels*. <https://arxiv.org/abs/2212.10496>. abgerufen am 21.07.2024
12. GenAI Workshop. (2023). *Versicherungsforen Leipzig - GenAI Workshop*, (S. <https://www.versicherungsforen.net/strategie-innovation/inhouse-seminar-chatgpt>). Leipzig. abgerufen am 21.07.2024
13. Hotz, H. (2023). *RAG vs Finetuning — Which Is the Best Tool to Boost Your LLM Application?* <https://towardsdatascience.com/rag-vs-finetuning-which-is-the-best-tool-to-boost-your-llm-application-94654b1eaba7>. abgerufen am 21.07.2024
14. J.D. (2023). *UMAP – “Uniform Manifold Approximation and Projection” und die Riemannschen Mannigfaltigkeiten*. <https://atlane.de/umap-uniform-manifold-approximation-and-projectionumap/>. abgerufen am 21.07.2024
15. Jaadi, Z. (2024). *Principal Component Analysis (PCA): A Step-by-Step Explanation*. <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>. abgerufen am 21.07.2024
16. jjinho. (2023). *Open Book LLM Science Exam*. <https://www.kaggle.com/code/jjinho/open-book-llm-science-exam>. abgerufen am 21.07.2024
17. Kansal, A. (2024). *Building Generative AI-Powered Apps*.
18. langchain. (2024). *langchain*. <https://python.langchain.com/v0.2/docs/introduction/>. abgerufen am 21.07.2024

19. langchain. (2024). *LLMs with langchain*. [https://python.langchain.com/v0.1/docs/modules/model\\_io/llms/](https://python.langchain.com/v0.1/docs/modules/model_io/llms/). abgerufen am 21.07.2024
20. langchain. (2024). *rag-from-scratch*. [https://github.com/langchain-ai/rag-from-scratch/blob/main/rag\\_from\\_scratch\\_1\\_to\\_4.ipynb](https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb). abgerufen am 21.07.2024
21. langchain. (2024). *text embedding langchain*. [https://python.langchain.com/v0.1/docs/modules/data\\_connection/text\\_embedding/](https://python.langchain.com/v0.1/docs/modules/data_connection/text_embedding/). abgerufen am 21.07.2024
22. langchain. (2024). *Vector stores*. [https://python.langchain.com/v0.1/docs/modules/data\\_connection/vectorstores/](https://python.langchain.com/v0.1/docs/modules/data_connection/vectorstores/). abgerufen am 21.07.2024
23. Liang Xu, L. L. (2024). *Nanjing Yunjin intelligent question-answering system based on knowledge graphs and retrieval augmented generation technology*.
24. llamaindex. (2024). *llamaindex*. <https://docs.llamaindex.ai/en/stable/>. abgerufen am 21.07.2024
25. Monigatti, L. (2023). *Explaining Vector Databases in 3 Levels of Difficulty*. <https://towardsdatascience.com/explaining-vector-databases-in-3-levels-of-difficulty-fc392e48ab78>. abgerufen am 21.07.2024
26. Monigatti, L. (2023). *Retrieval-Augmented Generation (RAG): From Theory to LangChain Implementation*. <https://towardsdatascience.com/retrieval-augmented-generation-rag-from-theory-to-langchain-implementation-4e9bd5f6a4f2>. abgerufen am 21.07.2024
27. Müller, M. (2022). *Wissensmanagement klipp & klar*.
28. Nginx. (kein Datum). <https://nginx.org/en/>. abgerufen am 21.07.2024
29. Noack, N. (2024). *React, Angular und Vue.js: Eine Gegenüberstellung von Frontend-Frameworks*. <https://www.dotnetpro.de/frontend/user-interface/react-angular-vuejs-gegenueberstellung-frontend-frameworks-2910346.html>. abgerufen am 21.07.2024
30. novita.ai. (kein Datum). *What is RAG*. <https://blogs.novita.ai/what-is-rag-a-comprehensive-introduction-to-retrieval-augmented-generation/>. abgerufen am 21.07.2024
31. ollama. (2024). *llama3 with ollama*. <https://www.ollama.com/library/llama3>. abgerufen am 21.07.2024
32. ollama. (2024). *local embedding*. <https://www.ollama.com/library/nomic-embed-text>. abgerufen am 21.07.2024
33. OpenAI. (2024). *Optimizing LLMs for accuracy*. <https://platform.openai.com/docs/guides/optimizing-llm-accuracy>. abgerufen am 21.07.2024
34. Parti, A. (2024). *The Ultimate Guide to Retrieval-Augmented Generation (RAG)*. <https://pareto.ai/blog/retrieval-augmented-generation>. abgerufen am 21.07.2024
35. Plotly. (kein Datum). <https://plotly.com/javascript/>. abgerufen am 21.07.2024
36. pytest. (kein Datum). <https://docs.pytest.org/en/stable/contents.html>. abgerufen am 21.07.2024
37. solid-it. (2024). *DB-Engines Ranking von Vektor DBMS*. <https://db-engines.com/de/ranking/vektor+dbms>. abgerufen am 21.07.2024

38. *SQLAlchemy*. (kein Datum). <https://www.sqlalchemy.org/>. abgerufen am 21.07.2024
39. Technology, H. K. (2024). *RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation*. <https://arxiv.labs.arxiv.org/html/2404.00610>. abgerufen am 21.07.2024
40. Vollmer, P. D.-I. (2022). *Software-Architektur - Architekturstile*. [https://www.itc-dortmund.de/nextcloud/apps/files/?dir=/Sem4\\_Inhalte/MP-05-3\\_Softwarearchitektur\\_und\\_Design/Vorlesung&openfile=206690](https://www.itc-dortmund.de/nextcloud/apps/files/?dir=/Sem4_Inhalte/MP-05-3_Softwarearchitektur_und_Design/Vorlesung&openfile=206690) Folie 92-101. abgerufen am 21.07.2024
41. *vuex*. (kein Datum). <https://vuex.vuejs.org/guide/>. abgerufen am 21.07.2024
42. Yunfan Gao, Y. X. (2024). *Retrieval-Augmented Generation for Large Language Models: A Survey*.

### Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig angefertigt und mich keiner fremden Hilfe bedient sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

### Erklärung zu eingesetzten Hilfsmitteln

1) Korrekturservice der Fachhochschule bzw. des Fachbereichs genutzt: Ja ☐ Nein ☒

2) Einsatz eines externen (kommerziellen) Korrekturservice: Ja ☐ Nein ☒  
wenn ja, welcher

3) Folgende Personen haben die Arbeit zusätzlich Korrektur gelesen:

Bernd Hardick

4) Nutzung von Sprachmodellen für die Texterstellung (z.B. ChatGPT),  
wenn ja, welche und in welchen Abschnitten:

Ja ☐ Nein ☒

5) Sprachübersetzungstools (z.B. Google Übersetzer, DeepL),  
wenn ja, welche und in welchen Abschnitten:

Ja ☐ Nein ☐

DeepL - Abstract

6) Einsatz von Software zur Sprachkorrektur (z.B. Grammarly),  
wenn ja, welche und in welchen Abschnitten:

Ja ☐ Nein ☒

7) Einsatz anderer Hilfsmittel:

8) Ich stimme dem möglichen Einsatz von Software zur Plagiatserkennung zu:

Ja ☒ Nein ☐

Ich bestätige, dass obige Aussagen vollständig und nach bestem Wissen ausgefüllt wurden.

22.07.2024 Tim Beld  
Datum, Unterschrift des Verfassers