universität wien

BACHELORARBEIT / BACHELOR'S THESIS

Titel der Bachelorarbeit / Title of the Bachelor's Thesis

„Smart Student Support Chatbot“

verfasst von / submitted by

Muza Sofiia Shutova

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Bachelor of Science (BSc)

Wien, 2025 / Vienna, 2025

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 033 521
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Bachelorstudium Informatik
Betreut von / Supervisor:	Dipl.-Ing. Dr. Kalchgruber
Mitbetreut von / Co-Supervisor:	>Akademische(r) Grad(e) Vorname Zuname< / >degree(s) first name family name<

Acknowledgements

Ich möchte mich herzlich bei allen bedanken, die mich während der Erstellung dieser Bachelorarbeit unterstützt haben.

Mein besonderer Dank gilt Herrn Dipl.-Ing. Dr. Kalchgruber für seine fachliche Unterstützung, seine schnelle Erreichbarkeit und die hilfreichen Rückmeldungen auf meine Herausforderungen. Seine Betreuung war für mich sehr wertvoll.

Ebenso danke ich der Universität Wien für die zur Verfügung gestellten Ressourcen und die umfassende Unterstützung während meines gesamten Studiums.

Mein tiefster Dank gilt meinem Ehemann, der mich in dieser Zeit stark unterstützt hat – sowohl fachlich als auch emotional. Er hat mir nicht nur viel geholfen, sondern auch den Mut und die Motivation gegeben, überhaupt ein Informatikstudium zu beginnen.

Nicht zuletzt danke ich meinen beiden Katern, die mir zu Hause mit ihrer ruhigen und liebevollen Art einen großen mentalen Rückhalt gegeben haben.

Kurzfassung

In dieser Bachelorarbeit wird die Entwicklung eines KI-basierten Chatbots für den universitären Einsatz untersucht. Ziel war es, einen prototypischen „Smart Student Support Chatbot“ zu erstellen, der häufige Fragen von Studierenden automatisiert und präzise beantworten kann. Dabei kamen moderne Open-Source-Technologien wie Ollama, Flask und Vue.js zum Einsatz. Der Chatbot basiert auf einer Retrieval-Augmented Generation (RAG)-Architektur, unterstützt mehrere Sprachen und wurde lokal auf Universitätsservern betrieben. Für die Wissensbasis wurden strukturierte Daten aus Curricula, Uni-Webseiten und XML-Dateien verwendet. Die Evaluation zeigte, dass das gewählte Sprachmodell (Gemma3:27b) in Kombination mit dem Embedding-Modell (bge-m3) qualitativ hochwertige, kontextbezogene Antworten liefert – auch ohne externes Feintuning. Die Arbeit zeigt das Potenzial datenschutzkonformer KI-Systeme für Bildungsinstitutionen und gibt Empfehlungen für zukünftige Weiterentwicklungen.

Hinweis zum Einsatz von KI-basierten Werkzeugen

Im Rahmen dieser Arbeit wurde ChatGPT von OpenAI (Version Stand Juni 2025) zur sprachlichen Unterstützung eingesetzt. Die KI wurde ausschließlich verwendet, um Formulierungen umzustellen, grammatikalische Korrekturen vorzunehmen oder Texte stilistisch zu verbessern. Die inhaltliche Ausarbeitung, Argumentation sowie die finale Textverantwortung liegen vollständig bei der Verfasserin.

Contents

List of Tables

List of Figures

1. Einleitung

1.1. Problemstellung: Warum brauchen Studierende einen Support-Chatbot?

Die Geschichte der Kommunikation zwischen Mensch und Maschine beginnt nicht erst mit modernen Chatbots. Bereits in den 1930er-Jahren legte Alan Turing mit dem Konzept der Turing-Maschine den Grundstein für das, was wir heute als künstliche Intelligenz kennen. Später formulierte er den berühmten Turing-Test – die Idee, dass eine Maschine dann intelligent genannt werden kann, wenn ein Mensch im Gespräch mit ihr nicht mehr unterscheiden kann, ob er mit einem Menschen oder einer Maschine kommuniziert.[Ins23]

Fast ein Jahrhundert später, im Jahr 2024/2025, sind wir dieser Vision näher als je zuvor. Viele Menschen erleben heute im Alltag, wie Maschinen scheinbar menschlich antworten, kontextbasiert reagieren und sogar einfühlsam wirken[SV24], [vGM25]. Sprachmodelle wie ChatGPT[Ope25] haben in kurzer Zeit Millionen von Nutzenden erreicht – nicht nur in der Technik-Community, sondern auch in Schule, Studium und Beruf. Einige nutzen KI-Chatbots täglich zum Lernen, zur Recherche oder zum Schreiben. Andere berichten sogar, dass der Chatbot ihr „bester Studienfreund“ geworden sei – oder „die interessanteste Person im Raum, mit der man sprechen kann“.

Diese Entwicklung zeigt: Die Grenzen zwischen Mensch-Maschine-Interaktion und echter Kommunikation sind für viele Nutzerinnen und Nutzer heute kaum noch spürbar. Künstliche Intelligenz ist nicht länger ein abstraktes Konzept, sondern ein praktisches Werkzeug, das unser Leben beeinflusst – und vielleicht sogar verändert.[CJC+24]

Vor diesem Hintergrund stellt sich die Frage: Wie nützlich wäre ein KI-basierter Support-Chatbot für eine große Universität wie die Universität Wien? Und: Welche Herausforderungen und Chancen ergeben sich, wenn ein solcher Prototyp in ein echtes Produkt überführt werden soll?[Uni25a]

Mit über 85.000 Studierenden und rund 10.000 Mitarbeitenden ist die Universität Wien eine der größten Bildungseinrichtungen im deutschsprachigen Raum [Uni25g], [Zen25a].Jedes Semester entstehen tausende Fragen rund um das Studium: von Anmeldemodalitäten über Prüfungsfristen bis hin zu spezifischen Infos zu Lehrveranstaltungen, Curricula oder Ansprechpartnern. Die Idee eines intelligenten, stets erreichbaren Chatbots, der auf diese Fragen rund um die Uhr präzise Antworten liefern kann, ist nicht nur technisch spannend, sondern auch gesellschaftlich relevant.

Ziel dieses Projekts war es, einen funktionsfähigen Prototyp eines solchen Chatbots zu entwickeln – mit Fokus auf realistischen Anforderungen, aktueller Open-Source-Technologie, und unter Berücksichtigung praktischer Einschränkungen wie Rechenleistung, Datenschutz und Mehrsprachigkeit.

Diese Arbeit dokumentiert die Umsetzung eines Prototyps, analysiert die damit verbundenen technischen und organisatorischen Herausforderungen und zeigt auf, wie moderne KI-Technologien potenziell sinnvoll in den universitären Alltag integriert werden können. Darüber hinaus werden aktuelle Entwicklungen, Markttrends sowie mögliche Weiterentwicklungen und Optimierungspotenziale für zukünftige Anwendungen betrachtet.

2. Hintergrund und verwandte Arbeiten

2.1. Überblick über bestehende Systeme (Uni-Websites, Foren, klassische FAQs)

Generell existieren sehr viele Quellen, in denen Studierende, Universitätsmitarbeitende oder potenziell Interessierte verschiedene relevante Informationen finden können. Einerseits profitieren Menschen immer davon, wenn Informationen im Internet verfügbar sind. Andererseits kann es jedoch auch verwirrend sein – bei der Vielzahl an Quellen ist es sehr wahrscheinlich, dass Informationen, obwohl vorhanden, leicht übersehen werden können. Insbesondere zu Beginn des Studiums besteht häufig Unsicherheit darüber, wo zentrale Informationen zu finden sind und welche davon aktuell und verlässlich sind. Es gibt aber Mentoring-Programme zur Unterstützung und mit der Zeit gewöhnt man sich auch daran, wo man gezielt nach Informationen suchen sollte.

Hier ist ein kurzer Überblick über Quellen, die häufig zur Informationssuche genutzt werden:

1. StudienServiceCenter (SSC) Informatik SSC der Fakultät für Informatik ist die zentrale Anlaufstelle für alle administrativen Belange rund um das Informatikstudium. Hier erhalten Studierende Unterstützung bei Fragen zur Lehrveranstaltungsanmeldung, Prüfungsorganisation, Studienabschluss und Anerkennungen. [Uni25d]

2. Studienprogrammleitung (SPL)

SPL ist für studienrechtliche Angelegenheiten zuständig. Sie berät bei Fragen zu Curricula, Studienplänen und prüfungsrechtlichen Belangen.[Uni25c]

3. Digitale Plattformen: u:space und u:find

Die Universität Wien stellt zwei zentrale digitale Plattformen zur Verfügung:

u:space: Hier können Studierende ihre Lehrveranstaltungen und Prüfungen verwalten, Noten einsehen, Studienbestätigungen herunterladen und den Semesterplaner nutzen. Besonders hilfreich ist der Semesterplaner für die Planung des Studienverlaufs. [Uni25f]

u:find: Das Vorlesungsverzeichnis bietet eine Übersicht über alle angebotenen Lehrveranstaltungen, inklusive Informationen zu Terminen, Lehrenden und Anmeldemodalitäten. [Uni25e]

4. Häufig gestellte Fragen (FAQs)

Die Studienvertretung Informatik (StV & FV) hat eine umfangreiche FAQ-Sektion erstellt, die Antworten auf häufige Fragen bietet, darunter: Studienplanung und Semesterplanung; Anmeldung zu Lehrveranstaltungen und Prüfungen; Unterschiede zwischen Lehrveranstaltungstypen (VO, UE, VU, etc.); Curriculumwechsel und Anerkennungen. Diese FAQs sind besonders hilfreich für Erstsemestrige und bieten praxisnahe Tipps für den Studienalltag. [Stu25a]

5. Studienvertretung Informatik (StV & FV)

Die Studienvertretung Informatik ist eine von Studierenden organisierte Initiative, die Beratung und Unterstützung bei studienbezogenen Fragen bietet. Sie organisiert zudem Veranstaltungen und fördert den Austausch unter Studierenden.[Stu25b]

6. IT-Support: ZID-Helpdesk

Der Zentrale Informatikdienst (ZID) bietet Unterstützung bei IT-bezogenen Fragen und Problemen.[Zen25b]

7. Weitere Unterstützungsangebote

Zusätzlich zu den oben genannten Ressourcen bietet die Universität Wien weitere Unterstützungsangebote für Studierende:

Mentoring-Programme: Erfahrene Studierende begleiten Studienanfänger*innen im ersten Semester [Uni25b]

Studienassistent*innen: Ansprechpersonen für spezifische Lehrveranstaltungen.

Basisgruppen: Studienrichtungsspezifische Gruppen zur Vernetzung und Unterstützung.

8. Studo [Stu25c]

Diese Angebote sollen den Einstieg ins Studium erleichtern und die Integration ins universitäre Leben fördern.

2.2. Forschung zu AI-gestützten Chatbots für Bildungseinrichtungen

2.2.1. Wie Studierende heute lernen und die Rolle von Chatbots im Lernalltag:

In den letzten paar Jahren haben wir weltweit eine große Veränderung darin beobachtet, wie Studierende ihre Art zu lernen verändert haben. Die „klassische“ Ausbildung, die vor allem aus Vorlesungen, Seminaren und Büchern bestand, wird heute zunehmend durch digitale Hilfsmittel ergänzt. Besonders der Einsatz von KI in Form von Chatbots wie ChatGPT, Gemini oder Claude hat sich rasant verbreitet. Fast alle Studierenden nutzen diese Tools jeden Tag , um komplexe Themen zu verstehen, Zusammenfassungen zu schreiben, Aufgaben zu bearbeiten oder Feedback zu ihren Texten zu bekommen[HEH24].

Diese Entwicklung stellt eine zentrale und große Frage: Welche Rolle spielen KI-Chatbots im Lernprozess? Tragen sie zur Verbesserung der Lernleistungen bei – oder führen sie eher zu oberflächlichem Wissen?

Solche Chatbots haben das Potenzial, das Lernen sehr effizient zu unterstützen. Sie können personalisierte Antworten auf Fragen geben, interaktive Gespräche führen und – was generell sehr wichtig ist – auch außerhalb der regulären Lehrveranstaltungen jederzeit zur Verfügung stehen [DLG+23]. Studierende merken, dass die Motivation steigt, da man ein besseres Verständnis von komplexen Inhalten gewinnt [Bak24].

Gleichzeitig besteht jedoch die Gefahr, dass sich der Umgang mit Informationen verschlechtert und die Fähigkeit, unterschiedliche Informationen zu analysieren und dabei das Wesentliche zu erkennen, verloren geht. Hinzu kommt die Gefahr, dass man sich zu stark auf die Antworten verlässt. Eine übermäßige Nutzung könnte langfristig die

selbstregulierte Lernfähigkeit beeinträchtigen und zu einer passiven Wissensaufnahme führen [SML24].

Wie in der Studie [GRCG24] gezeigt wird, gibt es noch weitere Veränderungen im Bildungssystem – man hat die Möglichkeit, mit solchen Tools den Lernprozess selbstständiger und stärker personalisiert zu gestalten (zum Beispiel durch gezielte Tipps, Erinnerungen oder Denkanstöße). Damit dieser Prozess wirklich gut funktioniert, müssen die Lernenden jedoch über eine gewisse digitale und geistige Reife verfügen. Sonst besteht die Gefahr, dass Studierende nur Informationen aufnehmen, ohne sie wirklich zu verstehen oder einordnen zu können [LGM23].

2.2.2. Empirische Erkenntnisse aus der aktuellen Forschung

In den letzten Jahren hat sich die Forschung zur Nutzung von Chatbots im Bildungssystem deutlich verändert. Immer mehr Leute interessieren sich dafür. Dabei geht es um verschiedene Ziele: Zum Beispiel, wie man das Lernen besser an einzelne Personen anpassen kann, wie man emotionale Hilfe in den Lernprozess einbaut, oder welche Art, Informationen zu zeigen, am besten hilft.

Ein gutes Beispiel ist das Projekt EduChat. Es handelt sich um ein Chatbot-System, das auf großen Sprachmodellen basiert und für den Einsatz in der Bildung entwickelt wurde. EduChat kann verschiedene Dinge machen, zum Beispiel Texte bewerten, emotionale Unterstützung geben, mit der sokratischen Methode arbeiten oder offene Fragen beantworten [DLG+23]. Das System ist offen zugänglich und kann an verschiedene Unterrichtsarten angepasst werden. Erste Untersuchungen zeigen, dass es die Motivation und den Lernerfolg verbessern kann.

Ein anderes System heißt ChatEd. Es verbindet ChatGPT mit Methoden, die normalerweise beim Suchen von Informationen benutzt werden. Ziel ist es, das Lernen besser zu machen und gleichzeitig Probleme von großen Sprachmodellen – wie zum Beispiel falsche Informationen ("halluzinieren") – zu verringern [Bak24]. Besonders spannend ist, dass man mit dem System automatisch strukturierte Lerninhalte geben kann und dabei trotzdem auf einzelne Personen eingehen kann.

2.2.3. Personalisierung, Feedback und emotionale Intelligenz

In der Studie von [SSC+24] wird gezeigt, dass KI-Chatbots nicht nur Informationen geben, sondern auch wie persönliche Lernhelfer funktionieren können. Sie passen sich an den Lernstil und das Wissen von den Studierenden an. Dadurch wird das Lernen oft einfacher, und der ganze Lernprozess kann besser werden. Solche Systeme geben Feedback, das zum eigenen Lernstand passt, und helfen so, dass man die Inhalte besser und langfristiger versteht.

Ein besonders wichtiger Aspekt ist die emotionale Unterstützung. Wie in der Studie von [DLG+23] beschrieben wird, haben empathische Chatbots einen positiven Einfluss auf das Wohlbefinden von Lernenden. In stressigen Phasen im Studium oder bei Unsicherheiten können solche Systeme entlasten, weil sie menschliche Gespräche nachahmen

und verständnisvoll reagieren. Das kann auch helfen, Hürden abzubauen – sowohl auf psychologischer Ebene als auch bei sprachlichen Schwierigkeiten.

2.2.4. Internationalisierung und Diversität

Ein Bereich, der bisher oft wenig beachtet wurde, ist die Rolle von Chatbots bei der Internationalisierung an Hochschulen. Laut [HEH24] können KI-basierte Chatbots vor allem für internationale Studierende eine wichtige Hilfe sein. Sie unterstützen nicht nur beim Lernen der Sprache, sondern auch beim Verstehen von kulturellen Unterschieden und organisatorischen Abläufen. Systeme, die mehrere Sprachen unterstützen, können dadurch aktiv zur besseren Chancengleichheit beitragen.

Auch in Ländern des Globalen Südens sieht man das Potenzial von KI-Systemen. In der Studie von [BAEV+22] wird am Beispiel von Ghana gezeigt, dass Studierende mit Hilfe eines virtuellen Tutors deutlich bessere Leistungen erreichen konnten. Die Chatbots haben dabei wie Mentor:innen funktioniert – sie gaben Rückmeldung und halfen beim Planen der Lernzeit. Besonders wichtig ist hier, dass solche Systeme auch in Regionen mit wenigen Ressourcen den Zugang zu guter Bildung ermöglichen.

2.2.5. Systematische Übersichten und Anwendungsrahmen

Die Übersicht von [GRCG24] zeigt, dass Chatbots das selbstständige Lernen stark unterstützen können – vor allem durch gezielte Fragen und Anleitungen. Auch die Studien von [LGM23] und [SML24] betonen, wie wichtig dabei Dinge wie Nachdenken über das eigene Lernen, das Setzen von Zielen und regelmäßiges Feedback sind. Trotzdem zeigen beide Untersuchungen auch, dass es noch Unsicherheiten gibt: Viele Studierende vertrauen den Antworten der Chatbots nicht ganz, finden sie zu allgemein oder wünschen sich mehr Bezug zum echten Lernalltag.

Für die organisatorische Einbindung schlagen [MS24] ein Modell mit mehreren Stufen vor. Dabei sollen KI-Chatbots nicht nur für das Lernen, sondern auch für Verwaltungsaufgaben an Fernhochschulen eingesetzt werden. Wichtig ist dabei, dass nicht nur die Technik gut funktioniert, sondern auch ethische Fragen beachtet werden – zum Beispiel Datenschutz, Transparenz der Technik und dass Menschen weiterhin die Kontrolle behalten.

2.2.6. Technologische Entwicklungen: Lokale Systeme und Ollama

Ein interessanter Punkt in der aktuellen Diskussion ist die Möglichkeit, KI-Chatbots lokal zu betreiben – zum Beispiel mit Frameworks wie Ollama, das Open-Source-Modelle einfach auf eigenen Geräten ausführen lässt. Dadurch wird nicht nur der Datenschutz gewährleistet, sondern es ist auch möglich, die Systeme an die speziellen Bedürfnisse der Institutionen anzupassen. Erste Pilotprojekte zeigen, dass so maßgeschneiderte Lernsysteme entwickelt werden können, die ohne US-basierte Cloudlösungen auskommen ([SSC+24], [DLG+23]).

Diese Technologie eröffnet besonders im europäischen Hochschulbereich neue Möglichkeiten, da sie sowohl die Kontrolle über Bildungsdaten stärkt als auch die Nutzung regionaler Sprachmodelle ermöglicht.

2.2.7. Fazit

KI-gestützte Chatbots sind mittlerweile ein fester Bestandteil im Hochschulalltag. Sie bieten viele Vorteile, wie die Möglichkeit, das Lernen der Studierenden individuell anzupassen, ihre Motivation zu steigern und ihnen Unterstützung zu bieten. Doch es gibt auch Herausforderungen. Dazu gehören die Qualität der bereitgestellten Informationen, die Gefahr einer zu starken emotionalen Abhängigkeit und die Tendenz zu passivem Lernen. Die aktuellen Studien zeigen ein gemischtes Bild und verdeutlichen, wie wichtig es ist, Chatbots gezielt und überlegt einzusetzen. Es ist notwendig, technologische Innovationen mit pädagogischen Konzepten und ethischen Richtlinien in Einklang zu bringen. Eine mögliche Lösung könnte die Integration von lokal laufenden Systemen wie Ollama sein, um datenschutzkonforme, skalierbare und kulturell angepasste Bildungslösungen zu schaffen.

2.3. Technologien: NLP, Machine Learning, Chatbot-Frameworks

Damit ein KI-gestützter Chatbot wie ChatGPT oder EduChat überhaupt funktionieren kann, braucht es eine Kombination aus verschiedenen Technologien. Drei besonders wichtige Bereiche sind: die Verarbeitung natürlicher Sprache (NLP), maschinelles Lernen (ML) und spezielle Software-Frameworks zur Entwicklung von Chatbots. In diesem Abschnitt wird erklärt, was diese Begriffe bedeuten und wie sie im Bildungskontext zusammenwirken.

2.3.1. Natürliche Sprachverarbeitung (NLP)

NLP steht für „Natural Language Processing“. Dabei geht es darum, dass Computer menschliche Sprache verstehen, verarbeiten und erzeugen können. Chatbots müssen Texte nicht nur lesen, sondern auch richtig interpretieren. Zum Beispiel sollen sie erkennen, ob eine Frage gestellt wurde, welchen Inhalt sie hat und wie die Antwort formuliert sein soll. NLP besteht aus mehreren Aufgaben. Einige davon sind:

- Tokenisierung: Der Text wird in Wörter oder Sätze zerlegt.
- Spracherkennung: Die Bedeutung von Wörtern wird analysiert, z.B. mit Wortarten oder Satzstruktur.
- Entitäten-Erkennung: Wichtige Begriffe wie Personennamen oder Daten werden erkannt.
- Stimmungserkennung (Sentiment Analysis): Die Emotion in einem Text wird eingeschätzt – z.B. ob jemand verärgert oder dankbar ist.

Diese Prozesse sind sehr wichtig, damit Chatbots sinnvoll auf Anfragen reagieren können. Besonders in Bildungssystemen, wo es um genaue Antworten und eine klare Kommunikation geht, ist gute NLP-Leistung entscheidend[MJ09].

2.3.2. Maschinelles Lernen (ML)

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz, bei dem ein System aus Daten „lernt". Das heißt: Der Chatbot wird mit sehr vielen Textbeispielen trainiert, sodass er selbst Regeln und Muster erkennen kann. Dadurch kann er später auch auf neue, unbekannte Anfragen reagieren.

Ein besonders bekanntes Verfahren ist das sogenannte Deep Learning, bei dem künstliche neuronale Netzwerke verwendet werden. Moderne Sprachmodelle wie GPT-3 oder GPT-4 (von OpenAI) basieren auf sogenannten Transformer-Architekturen. Diese Modelle haben Milliarden von Parametern und können sehr komplexe Sprachstrukturen verarbeiten.

In der Bildung bedeutet das: Ein Chatbot kann mit der Zeit besser verstehen, wie Studierende fragen, wo ihre Probleme liegen, und welche Form der Antwort am besten passt[VSP+17].

2.3.3. Chatbot-Frameworks

Damit man nicht bei null anfangen muss, gibt es sogenannte Chatbot-Frameworks. Diese Software-Tools helfen Entwickler:innen dabei, Chatbots schneller und strukturierter zu erstellen. Sie bieten fertige Bausteine, z.B. für das Verarbeiten von Eingaben, das Steuern von Dialogen oder die Anbindung an Datenbanken.

Einige bekannte Frameworks sind:

- Rasa: Ein Open-Source-Framework, das sich gut für datenschutzfreundliche, lokale Chatbots eignet. Es wird oft in der Forschung und in Unternehmen verwendet.[Ras25]
- Microsoft Bot Framework: Bietet viele Funktionen, besonders für Integration in Microsoft-Dienste.[Mic25]
- Google Dialogflow: Wird häufig im Kundenservice verwendet und ist stark auf NLP optimiert.[Clo25]
- Ollama: Ein relativ neues Framework, das den lokalen Betrieb von Sprachmodellen wie LLaMA auf eigenen Geräten erlaubt – besonders interessant im Bildungskontext wegen Datenschutz.[Oll25b]

Diese Frameworks erlauben es, die eingesetzten Technologien flexibel an die Bedürfnisse von Hochschulen anzupassen. Besonders bei Systemen, die datenschutzkonform und unabhängig von US-amerikanischen Cloud-Diensten funktionieren sollen, spielen Open-Source-Lösungen wie Rasa oder Ollama eine wichtige Rolle[BFPN17], [FA24].

3. Methodik

3.1. Systemarchitektur des Chatbots

Der Chatbot ist in einer Microservices-Architektur implementiert und wird mit Hilfe von Docker bereitgestellt.

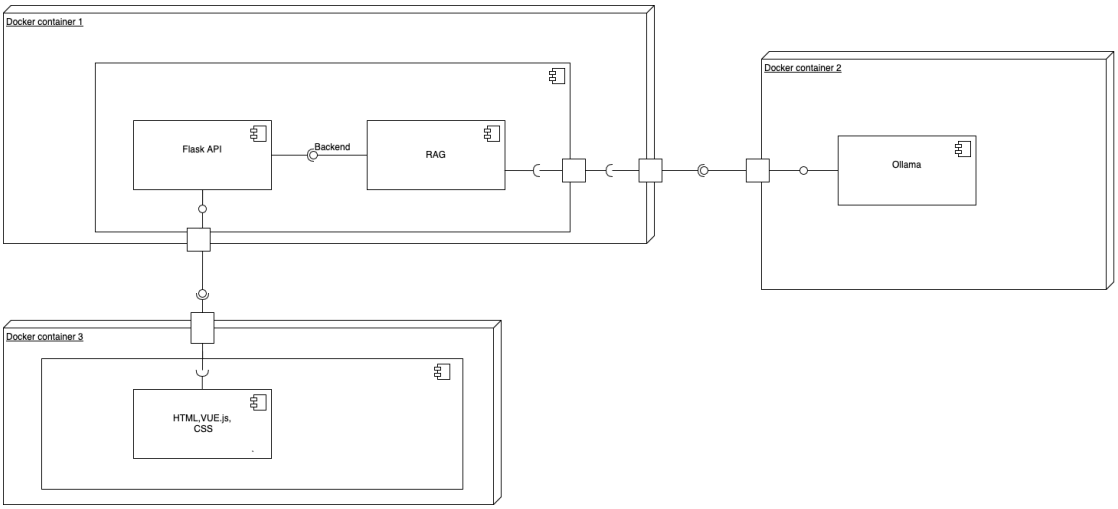

Figure 3.1.: Komponentendiagramm

Die gesamte Architektur meines Projekts basiert auf der Containerisierung mittels Docker. Dabei wurde die Anwendung in drei separate Docker-Container aufgeteilt: einer für das Frontend, einer für das Backend sowie ein dedizierter Container für die LLM-Komponente (Ollama). Die Nutzung von Docker spielt eine zentrale Rolle, da sie zahlreiche Vorteile für Entwicklung, Deployment und Wartung bietet.[Doc25]

Vorteile durch Docker: Docker ermöglicht es, Software in sogenannten Containern zu isolieren. Jeder Container enthält alle notwendigen Komponenten – wie Code, Laufzeitumgebungen, Systembibliotheken und Abhängigkeiten. Dadurch ist die Anwendung plattformunabhängig, was die Portabilität deutlich erhöht. Ein Container, der lokal auf einem Entwicklungssystem funktioniert, kann ohne Änderungen auf einem Server oder in der Cloud betrieben werden.

Ein weiterer zentraler Vorteil ist die Modularität. Durch die Aufteilung in getrennte Container können einzelne Komponenten unabhängig voneinander entwickelt, aktualisiert oder ausgetauscht werden. Dies vereinfacht nicht nur die Wartung, sondern erhöht

auch die Skalierbarkeit. Bei Bedarf kann zum Beispiel der LLM-Container auf einer leistungsfähigeren Maschine betrieben werden, ohne dass das Frontend oder Backend angepasst werden muss. Aufbau der Architektur

Die Architektur besteht aus folgenden drei Hauptkomponenten, die jeweils in eigenen Docker-Containern laufen:

1.Frontend (Vue.js, HTML, CSS, JavaScript) Das Frontend stellt die Benutzeroberfläche bereit und ermöglicht die Eingabe von Nutzeranfragen in einem chatähnlichen Interface. Die Kommunikation mit dem Backend erfolgt über REST-APIs.

2.Backend (Python Flask) Das Backend empfängt Anfragen vom Frontend und koordiniert die Interaktion mit der LLM-Komponente. Es führt außerdem eine Vorverarbeitung der Daten durch und kümmert sich um den Zugriff auf die vektorbasierte Datenbank (Chroma), um kontextrelevante Informationen bereitzustellen.

3. LLM – Ollama Die Komponente für das Sprachmodell (LLM) – hier realisiert mit Ollama sowie den Modellen Gemma 3:27b und BGE-M3 (Embedding-Modell) – ist in einem eigenen Docker-Container untergebracht. Diese Trennung bietet mehrere Vorteile: Das Sprachmodell kann unabhängig vom restlichen System aktualisiert oder ausgetauscht werden, ohne Auswirkungen auf Backend oder Frontend. Dadurch bleibt die Gesamtarchitektur stabil, modular und leichter wartbar.

Was ich besonders praktisch finde: Ollama unterstützt mehrere kostenlose, vortrainierte Sprachmodelle, wie z.B. LLaMA oder Mistral. Man kann sehr einfach zwischen diesen Modellen wechseln – oft reicht es, nur den Modellnamen in der Konfiguration zu ändern. Weil das alles in einem eigenen Container läuft, muss ich nichts neu installieren oder am restlichen System herumbasteln. Das spart Zeit und Nerven, vor allem wenn man verschiedene Modelle ausprobieren will. Die genauere Beschreibung von möglichen Modelen und Auswertung mache ich später in Implementierung Section. Gerade beim Testen und Experimentieren, wie man es in Projekten oder bei der Forschung oft macht, ist diese Flexibilität wirklich hilfreich. Ich konnte verschiedene Modelle schnell vergleichen und schauen, welches am besten zu meiner Anwendung passt – ohne das System jedes Mal neu aufzubauen. Das zeigt, wie nützlich und komfortabel Docker in Kombination mit Ollama sein kann.

Vergleich mit bestehender Forschung: Ein ganz ähnlicher Ansatz wurde auch in der Arbeit von Stadelmann et al. (2023) verfolgt [RT23]. Dort haben sie ihre Architektur ebenfalls mit Docker umgesetzt und zwei Container genutzt – einen für das Backend und einen für das LLM-Modell. Die Idee dahinter war, die Anwendung modular aufzubauen, also die verschiedenen Komponenten voneinander zu trennen, damit sie unabhängig voneinander laufen und leichter zu verwalten sind.

3.1.1. Visuelle Darstellung der Systemarchitektur

Zur besseren Veranschaulichung der Architektur und der Abläufe im System wurden zwei Diagramme erstellt, die den technischen Datenfluss zwischen den einzelnen Komponenten darstellen. Ablaufdiagramm: Anfrageverarbeitung im Chatbot-System

Diese Diagramme (siehe Abbildung Datenflussdiagramm 1 und Datenflussdiagramm 2) zeigt den typischen Ablauf, nachdem ein Benutzer eine Frage im Chatinterface eingibt.

Der Prozess beginnt im Frontend (Vue.js), das die Nutzereingabe entgegennimmt und diese als HTTP-Anfrage an das Backend (Flask API) weiterleitet. Dort wird geprüft, ob die Anfrage gültig ist. Falls nicht, wird direkt eine Fehlermeldung zurückgegeben. Ist die Anfrage jedoch korrekt, wird sie an den LLM-Container (Ollama) weitergeleitet. Ollama verarbeitet die Anfrage mithilfe des Sprachmodells und sendet die generierte Antwort zurück an das Backend, das diese wiederum an das Frontend überträgt. Schließlich wird die Antwort dem Benutzer angezeigt. Dieses Ablaufdiagramm hilft, den linearen und logischen Fluss der Daten zu verstehen und zeigt, wie sauber die drei Container – Frontend, Backend und LLM – miteinander kommunizieren. Durch diese klare Struktur ist das System leicht nachvollziehbar und gut wartbar.

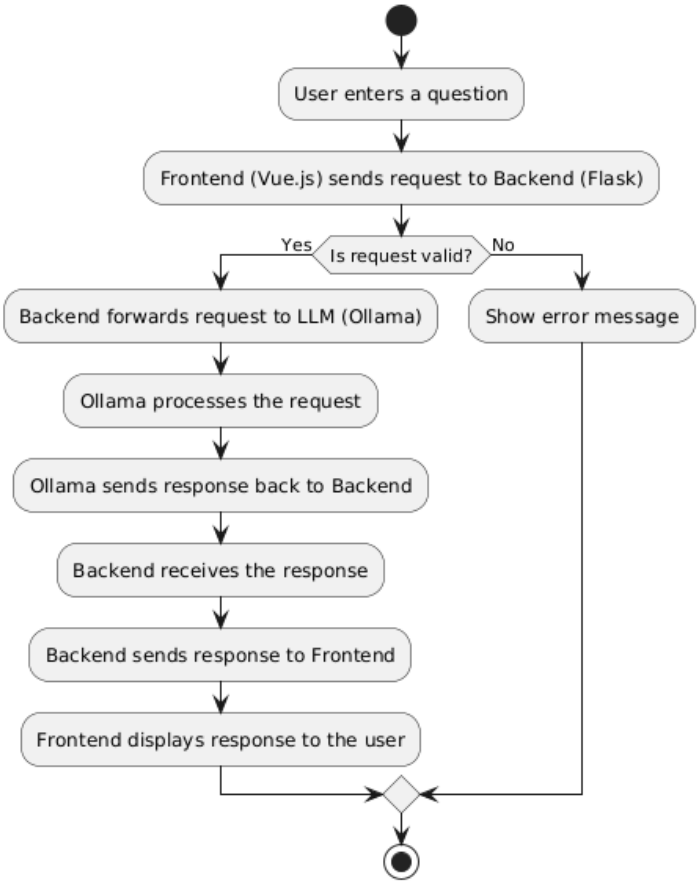

Figure 3.2.: Datenflussdiagramm 1

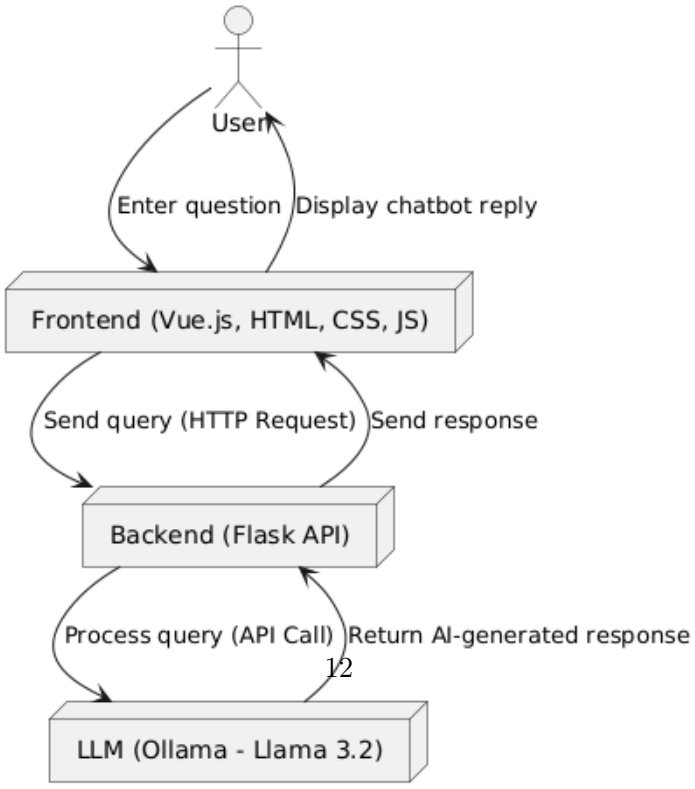

Figure 3.3.: Datenflussdiagramm 2

3.2. Wahl der Technologien (Python, Flask, NLP, ML, Chatbot Framework)

Für die Entwicklung des Chatbots wurden bewusst moderne und leichtgewichtige Technologien ausgewählt, die gut dokumentiert sind. Ziel war es, ein System zu schaffen, das sowohl lokal als auch in containerbasierten Umgebungen stabil läuft. Zudem sollte es gut wartbar und bei Bedarf einfach erweiterbar sein.

Die eingesetzten Technologien stellen eine ausgewogene Kombination aus Benutzerfreundlichkeit, Leistungsfähigkeit und starker Unterstützung durch die Entwickler-Community dar. Im Folgenden werden die zentralen Technologien vorgestellt und die Gründe für ihre Auswahl erläutert.

3.2.1. Python als Hauptprogrammiersprache

Für dieses Projekt wurde Python als Hauptprogrammiersprache gewählt, weil es sehr gut geeignet ist für Aufgaben im Bereich Künstliche Intelligenz, Datenverarbeitung und Webentwicklung. Die Sprache hat viele nützliche Bibliotheken, die besonders für Natural Language Processing (NLP), Machine Learning (ML) und die Entwicklung von APIs gemacht sind [Gom22]. Dadurch kann man Chatbots mit Python sehr funktional machen, weil die Bibliotheken viel Arbeit erleichtern. Besonders hilfreich in diesem Projekt waren:

- Flask für die Erstellung der REST-API,
- LangChain für die Arbeit mit Sprachmodellen,
- PyMuPDF, pandas, unstructured und andere für die Vorbereitung von Daten,
- Chroma als vektorbasierte Datenbank, um passende Informationen im Kontext abzurufen.

Ein weiterer Vorteil ist die einfache Syntax von Python. Sie hilft besonders bei schnellem Prototyping, Debugging und bei der iterativen Entwicklung. Auch die große Community bietet viele Ressourcen und Hilfe im Internet.

In mehreren wissenschaftlichen Studien wird Python als eine der besten Programmiersprachen für Chatbots mit KI im Hochschulbereich genannt. Diese Entscheidung ist nicht zufällig. Python hat bestimmte Vorteile, die dort auch genauer erklärt werden. In der Studie [K20] wird zum Beispiel gesagt, dass Python sehr benutzerfreundlich ist. Das zeigt, dass Python mit seiner einfachen Syntax und guten Unterstützung von NLP-Technologien besonders geeignet ist für solche Projekte.

Außerdem kann Python gut mit anderen Technologien und Plattformen kombiniert werden. Das ist sehr wichtig für Universitäten, weil Chatbots oft in bestehende Systeme integriert werden müssen.

3.2.2. Flask als Backend-Framework

[Pal24] Für mein Projekt habe ich das Webframework Flask verwendet. Es ist ein kleines und leichtes Framework in Python, das sehr gut für Microservices geeignet ist. Man kann es schnell einrichten, und es gibt viele Erweiterungen, die man bei Bedarf nutzen kann.

In meinem Fall hat das Backend mit Flask diese Aufgaben übernommen:

- Anfragen von Benutzer empfangen und prüfen,
- Texteingaben vorbereiten (z.B. bereinigen oder analysieren),
- Verbindung mit dem LLM-Modell über HTTP herstellen,
- Antwort zurück an das Frontend schicken, in strukturierter Form.

Flask eignet sich besonders gut für REST-Architekturen und lässt sich problemlos in eine Docker-Umgebung integrieren. Das Framework zeichnet sich durch einfache Verständlichkeit und hohe Flexibilität aus – Eigenschaften, die insbesondere bei Projekten mit begrenzten Ressourcen oder Einzelentwicklung von Vorteil sind.

3.2.3. Vue.js für das Frontend

Für die Gestaltung der Benutzeroberfläche des Chatbots wurde Vue.js als JavaScript-Framework gewählt. Vue.js ist modern, leicht zu erlernen und bietet gleichzeitig eine Vielzahl an Möglichkeiten für die Entwicklung interaktiver Webanwendungen. Die klare und gut strukturierte Dokumentation erleichtert den Einstieg und unterstützt eine effiziente Umsetzung [Vue24], [Li24].

3.2.4. LLM mit Ollama

[Oll25b]

Ollama bietet eine praktische und lokal ausführbare Lösung zur Nutzung großer Sprachmodelle – ein entscheidender Vorteil in Projekten, bei denen Datenschutz, Reproduzierbarkeit und volle Kontrolle über das Modellverhalten wichtig sind. Für mein Projekt war besonders relevant, dass Ollama mit verschiedenen LLMs funktioniert und sich direkt in bestehende Entwicklungsumgebungen integrieren lässt – ohne komplizierte Cloud-Infrastruktur oder externe API-Abhängigkeiten.

Die Studie von [LKH24] zeigt, wie Ollama in der Industrie eingesetzt werden kann, um PDF-basierte Informationen effizient zu extrahieren und in einem lokalen Chatbot-System zugänglich zu machen. Interessant war dabei, dass das System nicht nur Antworten aus dem Modell generiert, sondern durch sogenannte RAG-Techniken (Retrieval-Augmented Generation) Inhalte aus technischen Dokumenten gezielt heranzieht. Gerade bei mehrspaltigen Layouts und komplex strukturierten Informationen zeigte sich Ollama stabiler als viele Alternativen. Für mein Hochschulumfeld bedeutet das: Auch große Mengen an Modulhandbüchern oder Prüfungsordnungen könnten damit automatisiert verarbeitet und zugänglich gemacht werden.

Die Arbeit von [YWP24] geht noch weiter und demonstriert, wie domänenspezifische Sprachmodelle, die lokal über Ollama laufen, besonders gut auf spezifische akademische Fachbereiche angepasst werden können. Sie trainierten das System mit über einer Million fachspezifischer Artikel aus der Materialwissenschaft und zeigten, dass lokal betriebene Chatbots qualitativ hochwertige, kontextnahe Antworten liefern – vergleichbar mit kommerziellen Cloud-Diensten. Besonders hervorzuheben ist: Die Autor:innen betonen, wie wichtig lokale Kontrolle über das Modell für Forschung und Wissenschaft ist, vor allem wenn Datensicherheit eine Rolle spielt.

Ollama stellt nicht nur eine technische Lösung dar, sondern bietet auch einen wichtigen Schritt in Richtung Unabhängigkeit. Es ermöglicht die Entwicklung intelligenter Chatbots, die sowohl hohe technische Leistungsfähigkeit als auch Datenschutzkonformität vereinen – Anforderungen, die insbesondere im Hochschulkontext eine zentrale Rolle spielen.

3.2.5. Modellauswahl in Ollama: Embedding und Antwortgenerierung

Für die Funktionalität des Chatbots benötigt man zwei verschiedene Modelle, die jeweils eine zentrale Rolle im Informationsabruf und der Antwortgenerierung spielen. Erstens wird ein sogenanntes **Embedding-Modell** verwendet, das die Anfragen der Nutzer semantisch analysiert und passende Textabschnitte aus einer Vektordatenbank identifiziert. Diese Vektoren wurden zuvor aus den Inhalten meiner Wissensbasis generiert. Zweitens kommt ein **großes Sprachmodell (LLM)** zum Einsatz, das auf Grundlage dieser extrahierten Inhalte eine kohärente und möglichst kontextnahe Antwort formuliert.

Die Entscheidung, welche konkreten Modelle für diese beiden Aufgaben verwendet werden, ist entscheidend für die Gesamtqualität des Chatbots. Sowohl Genauigkeit als auch Antwortzeit und thematische Relevanz hängen stark davon ab, wie gut die gewählten Modelle zu meiner Anwendung passen. Die konkrete Auswahl und Evaluierung der Modelle werden detailliert in Abschnitt Implementierung beschrieben. An dieser Stelle gibt es jedoch einen Überblick über gängige Modelle geben, die mit Ollama kompatibel sind, und deren typisches Einsatzspektrum erläutern.

1. Modelle für Embedding-Aufgaben Da für den Chatbot ein leistungsfähiges Embedding-Modell erforderlich ist, wurden verschiedene in Ollama verfügbare Modelle betrachtet. Je nach Projektanforderung weisen diese Modelle unterschiedliche Stärken auf. Im Folgenden findet sich ein Überblick über die untersuchten Modelle:

mxbai-embed-large: Das Modell verfügt über etwa 334 Millionen Parameter und erzielt in verschiedenen Anwendungsbereichen sehr gute Ergebnisse. In Benchmarks wie MTEB übertrifft es sogar andere Modelle vergleichbarer Größe. Es eignet sich besonders für präzise semantische Suchanfragen, beispielsweise bei der Verarbeitung langer oder komplexer Fragestellungen – ein typisches Szenario im Hochschulumfeld[Oll24c].
nomic-embed-text: Mit rund 137 Millionen Parametern hat dieses Modell einen großen Vorteil: Es kann sehr viel Text auf einmal verarbeiten – bis zu 8192 Tokens. In Tests ist es sogar besser als bekannte Modelle wie OpenAI's text-embedding-ada-002. Das kann

praktisch sein, wenn man mit langen Texten arbeitet, z.B. wenn ein Bot Kursinformationen oder längere Richtlinien durchsuchen soll[Oll24d].
all-minilm: Dieses Modell ist viel kleiner (zwischen 22 und 33 Millionen Parameter), aber trotzdem leistungsstark. Es eignet sich gut für einfache Suchen oder wenn man auf einem kleinen Server arbeitet. Wenn man zum Beispiel ein Projekt auf einem Laptop testen will oder wenig Speicher hat, ist das Modell gut geeignet[Oll24a].
snowflake-arctic-embed: Das Modell ist in verschiedenen Größen erhältlich, die von etwa 22 Millionen bis 335 Millionen Parametern reichen, was eine flexible Anpassung an unterschiedliche Leistungsanforderungen ermöglicht. Diese Skalierbarkeit erlaubt die Auswahl eines Modells, das optimal auf verfügbare Hardware-Ressourcen und spezifische Anwendungsfälle abgestimmt ist. Dadurch eignet sich das Modell besonders gut für Szenarien, in denen ein ausgewogenes Verhältnis zwischen Effizienz und Genauigkeit erforderlich ist – beispielsweise bei mobilen Chatbots mit begrenzter Rechenkapazität oder bei Systemen, die eine hohe Anzahl gleichzeitiger Anfragen verarbeiten müssen. Zudem unterstützt diese Flexibilität den Einsatz in unterschiedlichsten Umgebungen, von ressourcenarmen Geräten bis hin zu leistungsstarken Serverinfrastrukturen.[Sno24]
bge-m3: Dieses Modell unterstützt mehrere Vektorarten (z.B. dichte und spärliche Vektoren) und kann auch in mehreren Sprachen arbeiten. Es kann gut angewendet werden, wenn man ein Projekt macht, das für verschiedene Sprachen gedacht ist, wie zum Beispiel für eine internationale Universität oder mehrsprachige Webseiten[Oll24b].
shaw/dmeta-embedding-zh: Dieses Modell hat 400 Millionen Parameter und ist speziell für chinesische Sprache gemacht. Es wurde extra für chinesische Aufgaben trainiert und liefert dort sehr gute Ergebnisse. Wenn man also z.B. einen Chatbot für chinesischsprachige Studierende bauen möchte, ist dieses Modell die richtige Wahl[Oll25a].

2. Hauptmodelle für Antwortgenerierung Für den Chatbot braucht man ein Hauptmodell, das die Antworten generiert. Dieses Modell muss die von der Embedding-Funktion gefundenen relevanten Textabschnitte verstehen und daraus eine passende Antwort für den Benutzer formulieren. Die Auswahl des richtigen Hauptmodells ist entscheidend für die Qualität und Relevanz der Antworten. Hier sind einige der Hauptmodelle, die Ollama unterstützt: LLaMA (von Meta),Mistral, Mixtral, DeepSeek, Phi (von Microsoft), Claude (von Anthropic), Qwen (von Alibaba), LLaVA, Dolphin, OLMo (von AI2), OpenThinker, WizardLM, Hermes, Yi, Orca (von Microsoft), SmolLM, Moondream. Eine vollständige Liste unterstützter Modelle und deren Eigenschaften ist über die Ollama-Plattform abrufbar: https://ollama.com/search.

3.2.6. NLP und ML-Komponenten

Es wurden verschiedene NLP-Techniken benutzt. Sie halfen dabei, Benutzeranfragen zu verstehen, passende Informationen herauszufiltern und die Antworten des LLM übersichtlich darzustellen. Dabei wurden unter anderem folgende Tools verwendet:

3.2.7. Tokenisierung und Vektorisierung für semantische Suche:

Um Texte mit ähnlicher Bedeutung finden zu können – selbst wenn sie unterschiedliche Wörter verwenden – müssen Texte zunächst in eine Form gebracht werden, die Maschinen verstehen: sogenannte Vektoren.[Ful23]. Dafür wird der Text zunächst tokenisiert, also in kleinere Einheiten wie Wörter oder Wortteile zerlegt. Anschließend wird jeder dieser Texte mit Hilfe eines Sprachmodells vektorisiert – das bedeutet, er wird in einen Zahlenvektor umgewandelt, der die Bedeutung des Textes im „semantischen Raum“ repräsentiert.

Diese Vektoren lassen sich dann vergleichen: Je ähnlicher zwei Vektoren sind, desto ähnlicher ist auch die Bedeutung der zugrunde liegenden Texte. Auf diese Weise können bei Suchanfragen Inhalte gefunden werden, die nicht exakt dieselben Wörter enthalten, aber denselben Sinn.[Sch11]

3.2.8. Retrieval-Augmented Generation (RAG) mit Chroma-DB:

Bei Retrieval-Augmented Generation (RAG) werden aktuelle oder externe Informationen aus einer Datenbank herangezogen, um die Antworten eines Sprachmodells zu verbessern. Anstatt sich nur auf das interne Wissen des Modells zu verlassen, sucht das System zuerst nach passenden Textstellen (z.B. in einer eigenen Wissensdatenbank) und nutzt diese als zusätzliche Grundlage für die Antwort.
Chroma-DB [Chr23]ist dabei eine sogenannte Vektordatenbank, in der Texte als Zahlenvektoren gespeichert sind. So können ähnliche Inhalte schnell gefunden werden. Im Zusammenspiel mit RAG ermöglicht Chroma-DB, dass das Modell gezielt auf relevante Informationen zugreift und damit präzisere und nachvollziehbare Antworten gibt.

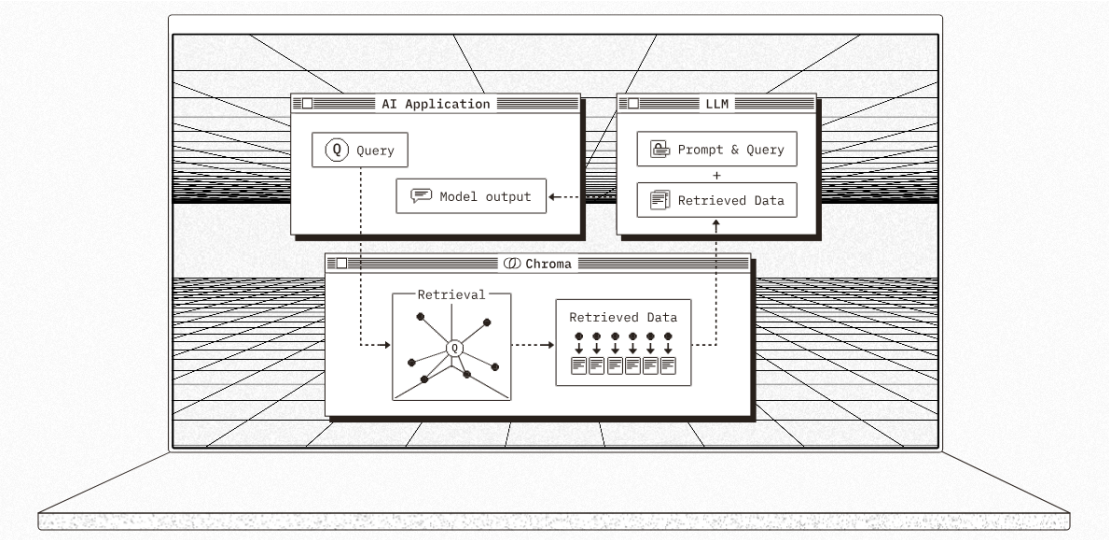

Figure 3.4.: ChromaDB[Chr23]

3.2.9. Prompt Engineering zur Optimierung von Model-Antworten:

Um die Antworten des Sprachmodells zu verbessern, wurde gezielt mit sogenannten Prompts gearbeitet – also mit der Art und Weise, wie Fragen oder Aufgaben an das Modell gestellt wurden. Durch das sogenannte Prompt Engineering wurden verschiedene Varianten getestet, z.B. durch das Hinzufügen von Kontextinformationen, das Formulieren klarer Anweisungen oder das Einbetten von Beispielen[Kag25]. Ziel war es, die Reaktionen des Modells besser steuerbar und nachvollziehbarer zu machen. Im Rahmen des Projekts wurden verschiedene Varianten von Prompts ausprobiert – von sehr ausführlichen, detaillierten Anweisungen mit präzisen Informationen bis hin zu kurzen, minimalistischen Eingaben, die nur aus einem Satz bestanden.

Die Beobachtung ist: Lange und gut strukturierte Prompts können die Qualität der Antworten deutlich verbessern. Allerdings bringt das auch Herausforderungen mit sich – besonders dann, wenn man mit einem kleineren lokalen Sprachmodell arbeitet, das nur über eine begrenzte Kontextlänge verfügt. In solchen Fällen zählt jeder Buchstabe eines Prompts zum Gesamtkontext, den das Modell verarbeiten muss.

Wenn der kombinierte Text aus Prompt, Nutzeranfrage und zusätzlichem Kontext (z.B. aus Dokumenten) die maximale Eingabelänge des Modells überschreitet, kann ein Teil des Kontexts abgeschnitten werden – was sich negativ auf die Antwortqualität auswirkt.

Das bedeutet: Effiziente Prompts sind vor allem bei kleineren Modellen entscheidend, um innerhalb der technischen Grenzen möglichst viel relevante Information zu übermitteln.

Zusätzlich wurde die Dokumentation von LangChain durchgesehen und dort empfohlene Prompt-Vorlagen getestet. Besonders hervorzuheben ist ein sehr minimalistischer Prompt, der trotz seiner Kürze erstaunlich gute Ergebnisse geliefert hat.[Lan24]

3.3. Datensammlung und -verarbeitung für Antworten

Die Knowledge Base für den Uni-Chatbot besteht aus verschiedenen Quellen. Die Informationen liegen sowohl auf Englisch als auch auf Deutsch vor, was zusätzliche Komplexität mit sich bringt. Außerdem sind die Daten in unterschiedlichen Formaten gespeichert, was bei der Verarbeitung gewisse Schwierigkeiten verursacht. Auf diese Herausforderungen und die Lösungsansätze wird später ausführlicher in der Sektion Implementierung eingegangen.

Ein wichtiger Teil der Daten stammt aus dem Portal u:find. Dort wurde Informationen im XML-Format extrahiert. Diese Dateien enthalten umfangreiche Details über die Universitätsmitarbeiter:innen, darunter Kontaktdaten, mögliche Sprechzeiten, Arbeitsstunden sowie Informationen zu den Lehrveranstaltungen, in denen sie unterrichten. Es ist auch ersichtlich, welchen Kursen die jeweiligen Personen zugewiesen sind und wer welche Inhalte vermittelt.

Der Chatbot wird auch dabei helfen, detaillierte Informationen zu den Curricula verschiedener Studienprogramme bereitzustellen. Das ist besonders hilfreich für Studierende, die sich über den Aufbau und die Anforderungen ihres Studiengangs informieren möchten.

Dafür habe ich folgende PDF-files verwendet:
Bachelorstudien:

- Informatik (521 [5] – Version 2022)
- Wirtschaftsinformatik (526 [3] – Version 2016)
- Lehramt UF Digitale Grundbildung und Informatik (193 053 [2], 198 414 [2])

Masterstudien:

- Bioinformatik (875 [1])
- Data Science (645 [1])
- Informatik (921 [2] – Version 2022)
- Medieninformatik (935 [3] – Version 2022)
- Wirtschaftsinformatik (926 [2] – Version 2016)
- Lehramt UF Digitale Grundbildung und Informatik (196 053 [2], 199 514 [2] – Version 2024)

Auslaufende Studienprogramme:

- Bachelor Informatik (521 [4] – Version 2016)
- Bachelor Informatik – Data Science (521 [4] – Version 2016)
- Bachelor Informatik – Medieninformatik (521 [4] – Version 2016)
- Bachelor Informatik – Medizininformatik (521 [4] – Version 2016)
- Bachelor Informatik – Scientific Computing (521 [4] – Version 2016)
- Bachelor Lehramt UF Informatik (193 053 [1], 198 414 [1])
- Master Lehramt UF Informatik (196 053 [1], 199 514 [1])

Diese Informationen werden automatisch aus den aktuellen Curricula übernommen, sodass der Chatbot immer auf dem neuesten Stand bleibt. Für Studierende, die sich nicht gut mit der Struktur der Studienpläne auskennen, kann das eine große Hilfe sein.

Weitere wichtige Quellen für die Knowledge Base sind die Website des StudienService-Centers (SSC)[Uni25d], die Website der Fakultät[Fak25] sowie damit verknüpfte Unterseiten. Diese Seiten enthalten eine Vielzahl an organisatorisch relevanten Informationen, die für Studierende besonders nützlich sind. Dazu gehören unter anderem die Struktur der Universität, die vorhandenen Fakultäten und ihre jeweiligen Spezialisierungen.

Außerdem finden sich dort Informationen zum Ablauf von Prüfungsanmeldungen, welche Unterlagen dafür erforderlich sind und wie diese korrekt vorzubereiten sind. Für

internationale Studierende ist besonders hilfreich, dass viele Informationen speziell für Nicht-österreichische Studierende bereitgestellt werden – etwa zu administrativen Schritten, Fristen oder Unterstützungsangeboten.

Ein weiterer wichtiger Aspekt dieser Webressourcen sind Hinweise auf aktuelle Veranstaltungen und Events, die für das Studium oder die Vernetzung innerhalb der Fakultät relevant sein können.

Table 3.1.: Überblick über genutzte Datenquellen für die Knowledge Base

Quelle	**Format(e)**	**Sprache(n)**	**Inhalte**
u:find Portal	XML	Deutsch, Englisch	Universitätsmitarbeiter:innen: Kontaktdaten, Sprechzeiten, Arbeitsstunden, Kurszuordnung, unterrichtete Fächer, LV.
Curriculum-PDFs/MDs	PDF	Deutsch	Studienpläne für Bachelor-, Master- und auslaufende Programme
SSC-Webseite	HTML	Deutsch, Englisch	Prüfungsablauf, notwendige Unterlagen, Hinweise für internationale Studierende, administrative Infos
Fakultätswebseite	HTML	Deutsch, Englisch	Fakultätsstruktur, Spezialisierungen, Veranstaltungen, aktuelle Events

4. Implementierung

4.1. Überblick über die Software-Entwicklung

Die Implementierung des Projekts begann mit der Einrichtung einer Docker-Umgebung. Rückblickend erwies sich dieser Schritt als sehr sinnvoll, da das Projekt im Laufe des Semesters auf drei unterschiedlichen Servern betrieben wurden, um die Systemleistung zu verbessern.

Ein zentrales Problem bestand darin, dass ein lokaler Laptop über eine begrenzte Rechenleistung (CPU/GPU) verfügte. Dies führte zu langen Antwortzeiten, insbesondere bei größeren Modellen. Durch die Nutzung von Docker konnte man jedoch die gesamte Umgebung flexibel und unkompliziert auf andere Server übertragen. Der Umstieg auf einen neuen Server war in wenigen Minuten möglich und erforderte lediglich das Ausführen einiger weniger Befehle.

Um verschiedene Hardwarekonfigurationen gezielt zu testen, wurden zwei separate Docker-Compose-Dateien erstellt – eine für den Betrieb mit CPU und eine für den Betrieb mit GPU. Dadurch kann man effizient vergleichen, wie sich unterschiedliche Rechenressourcen auf die Leistung und Antwortgeschwindigkeit des Systems auswirken.

4.1.1. Datenaufbereitung für das Training

Ein wesentlicher Schritt in der Umsetzung des Projekts war die Aufbereitung der Trainingsdaten. Da die Daten aus verschiedenen Quellen stammten, war dieser Schritt deutlich aufwendiger als ursprünglich angenommen.

Die Informationen von Webseiten konnten relativ einfach verarbeitet werden. Das Modell war in der Lage, die Inhalte ohne zusätzliche Anpassungen zu extrahieren und zu verstehen. Komplexer war hingegen die Verarbeitung von XML-Dateien, in denen relevante Informationen zu Lehrveranstaltungen und Dozierenden enthalten waren. Hierzu wurde ein Python-Skript entwickelt, das gezielt die benötigten Daten aus dem u:find Portal filterte. In diesem Zusammenhang wurden auch erste Bereinigungen vorgenommen – insbesondere die Entfernung von Bilddaten, da in der geplanten Chatbot-Anwendung keine Bildverarbeitung vorgesehen war. Durch das Entfernen dieser Bilder wurden die Dateien übersichtlicher und deutlich kleiner.

Am aufwendigsten gestaltete sich jedoch die Verarbeitung von PDF-Dokumenten, insbesondere der Studienverlaufspläne und Modulhandbücher. Diese Dokumente enthielten zahlreiche Tabellen, in denen häufig die wichtigsten Informationen strukturiert dargestellt waren. Lokale Sprachmodelle hatten jedoch große Schwierigkeiten, Inhalte aus Tabellen zuverlässig zu erkennen und den Kontext korrekt zu erfassen.

Daher war es notwendig, sämtliche PDFs manuell in das Markdown-Format zu überführen und die Tabellen entsprechend umzustrukturieren, um eine sinnvolle Tokenisierung und Kontextverarbeitung zu ermöglichen. Es wurden verschiedene Online-Tools getestet, um diesen Schritt zu automatisieren, jedoch stellten sich diese schnell als ungeeignet heraus: Die meisten kostenlosen Tools erlaubten nur die Konvertierung von 0,5 bis 1 Seite pro Dokument – für die Bedürfnisse mit teils über 20 Seiten pro Datei war das unpraktikabel[Notnd].

Auch der Versuch, moderne KI-basierte Tools zur automatisierten Umwandlung einzusetzen, war wenig erfolgreich. Diese lieferten häufig unvollständige oder fehlerhafte Ergebnisse und konnten den Inhalt nicht präzise genug strukturieren.

Zusammenfassend lässt sich sagen: Wenn hohe Genauigkeit und Datenqualität erforderlich sind, kann man sich aktuell nicht vollständig auf automatische Konvertierungstools verlassen. Aus diesem Grund wurde die Entscheidung getroffen, alle relevanten PDFs manuell in strukturierte Markdown-Dateien zu überführen.

4.1.2. Dynamische Datenverwaltung und Erweiterbarkeit

Eine zentrale Anforderung des Projekts bestand darin, die Datenbasis möglichst flexibel zu gestalten. Es sollte einfach möglich sein, neue Datenquellen hinzuzufügen oder veraltete Informationen zu entfernen – und das ohne aufwendige Änderungen im Quellcode vornehmen zu müssen.

Zur Umsetzung dieser Anforderung wurde eine Konfigurationsdatei namens sources.json erstellt. In dieser Datei definiert man strukturiert, aus welchen Verzeichnissen bzw. Quellen die Daten geladen werden sollen. Dazu gehören:

- der Ordner mds/, in dem sich alle Markdown-Dateien befinden, die zuvor aus PDF-Dokumenten umgewandelt wurden
- der Ordner xmls/, in dem alle relevanten XML-Dateien abgelegt sind
- sowie eine Liste mit URLs von Webseiten, die als zusätzliche Wissensquelle dienen.

Mit diesem Ansatz wird die Wartung und Erweiterung der Datenbasis stark vereinfacht: Möchte man beispielsweise ein neues Markdown-Dokument hinzufügen, genügt es, dieses in den mds/-Ordner zu legen – es wird beim nächsten Durchlauf automatisch berücksichtigt. Gleiches gilt für XML-Dateien. Möchte man eine neue Webseite als Quelle integrieren, reicht es aus, den entsprechenden Link in der sources.json-Datei zu ergänzen.

Auch das Entfernen von veralteten Informationen funktioniert nach demselben Prinzip: Dateien oder URLs müssen lediglich aus den jeweiligen Verzeichnissen oder der Konfigurationsdatei gelöscht werden, und sie werden beim nächsten Datenimport nicht mehr berücksichtigt.

Dieser modulare Aufbau spart nicht nur Zeit, sondern reduziert auch die Fehleranfälligkeit bei der Pflege der Wissensbasis – insbesondere in Szenarien, in denen regelmäßig mit aktualisierten Inhalten gearbeitet werden muss.

Figure 4.1.: Aufbau vom file "sources.json"

4.1.3. Auswahl und Konfiguration der Modelle

Ein besonders aufwendiger Teil des Projekts war die Auswahl geeigneter Modelle sowie deren Konfiguration und Anpassung, um sowohl die funktionalen als auch die nicht-funktionalen Anforderungen aus der Projektdefinition erfüllen zu können.

Bereits zu Beginn wurde eine zentrale Datei namens config.py erstellt, in der alle wichtigen Parameter für das Backend gesammelt sind. Diese Entscheidung hat sich im Laufe der Entwicklung als sehr hilfreich erwiesen, da ich zahlreiche Einstellungen mehrfach anpassen musste – etwa beim Wechsel zwischen verschiedenen Modellen oder beim Testen unterschiedlicher Konfigurationen. Durch die zentrale Verwaltung war es möglich, Änderungen schnell und unkompliziert vorzunehmen, ohne im gesamten Code nach einzelnen Parametern suchen zu müssen.

Die konfigurierbaren Einstellungen in dieser Datei umfassen u.a.:

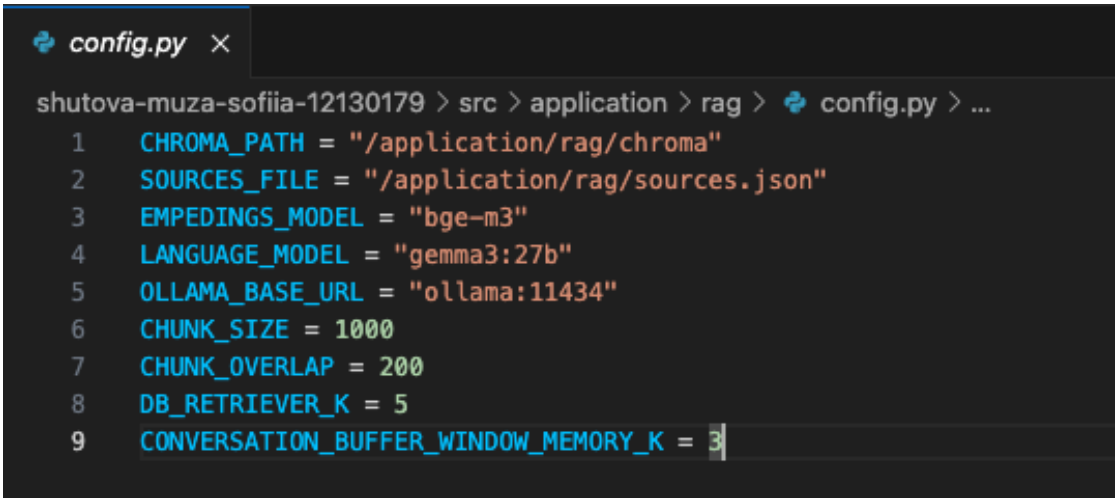

Figure 4.2.: Konfiguration file

Einige dieser Werte – wie zum Beispiel "CHUNK SIZE" und "CHUNK OVERLAP" – wurden auf Basis von Empfehlungen in der Dokumentation festgelegt. Diese Einstellungen gelten bei Projekten im Bereich RAG häufig als bewährte Standardwerte, insbesondere bei Chatbots[The23].

Der Parameter "DB RETRIEVER K" wurde auf 5 gesetzt. Das bedeutet, dass aus der Vektordatenbank jeweils die fünf relevantesten Chunks (Textabschnitte von je ca. 1.000 Zeichen) abgerufen werden, um sie dem Sprachmodell für die Antwortgenerierung bereitzustellen. Diese Anzahl wurde bewusst gewählt, da die verwendeten Sprachmodelle – vor allem in lokalen Umgebungen mit begrenzten Ressourcen – nur eine eingeschränkte Menge an Kontext verarbeiten können. Wären beispielsweise 10 oder 15 Chunks übergeben worden, könnten wichtige Informationen abgeschnitten oder ignoriert werden. Ein weiterer zentraler Parameter in der Konfiguration war "CONVERSATION BUFFER WINDOW MEMORY K". Dieser steht im Zusammenhang mit dem implementierten History-Feature, das dem System ermöglicht, den Verlauf eines Gesprächs zu berücksichtigen.

Konkret bedeutet dieser Parameter, dass das Modell die letzten drei Nutzeranfragen samt zugehöriger Antworten speichert und beim Verfassen der aktuellen Antwort einbezieht. Dadurch wird ein dialogorientiertes Verhalten gefördert, bei dem Nutzer nicht jedes Mal ihre Fragen vollständig neu formulieren müssen. Stattdessen können sie Anschlussfragen stellen, Rückfragen formulieren oder auf vorherige Aussagen Bezug nehmen – ähnlich wie in einem natürlichen Gespräch.

Dieses Feature trägt erheblich zur Benutzerfreundlichkeit bei, da die Interaktion dadurch weniger wie ein statischer Frage-Antwort-Ablauf wirkt und eher einem echten Dialog ähnelt.

Es wurde K = 3 gewählt, da dies aus der Sicht einen guten Kompromiss zwischen Funktionalität und Systemleistung darstellt. Je größer dieser Wert ist, desto mehr Kontext muss das Modell bei jeder Anfrage verarbeiten, was zu höherem Speicherbedarf

und längeren Antwortzeiten führen kann. Aus den Tests wurde sicher gestellt, dass K = 3 ein stabiles und performantes Verhalten ermöglichte, ohne spürbare Qualitätseinbußen im Gesprächsverlauf. Im ersten Schritt wurde das Modell gezielt nach den Kontaktdaten eines bestimmten Professors gefragt. Nachdem daraufhin eine passende Antwort erhalten wurde, wurde im nächsten Schritt lediglich der Name eines anderen Professors eingegeben – ohne dabei explizit zu erwähnen, dass erneut an den Kontaktdaten Interesse besteht.

Trotz der ungenauen Formulierung konnte das Modell durch den zuvor gespeicherten Gesprächsverlauf korrekt erkennen, dass sich die neue Anfrage inhaltlich auf denselben Kontext bezieht. Nutzerinnen und Nutzer müssen ihre Fragen nicht immer vollständig formulieren, sondern können wie in einem echten Gespräch auf vorherige Inhalte Bezug nehmen – was insbesondere im universitären Umfeld sehr hilfreich sein kann.

Figure 4.3.: Erste genaue Frage

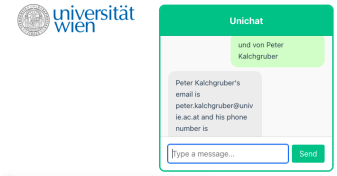

Figure 4.4.: Zweite unpräzise Frage

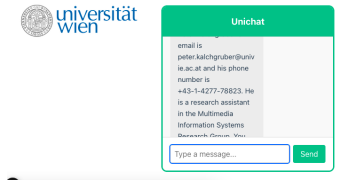

Figure 4.5.: Die Frage wurde korrekt beantwortet

Figure 4.6.: Verlauf einer Konversation mit aktivierter History-Funktion

Weitere wichtige Konfigurationen betreffen die Auswahl der Modelle selbst:

- Das Embedding-Modell: bge-m3, welches für die semantische Repräsentation der Texte verantwortlich ist.
- Das Sprachmodell: gemma3:27b, das auf Basis der gefundenen Chunks eine verständliche und thematisch passende Antwort generiert.

4.1.4. Embedding-Modell

Eine der wichtigsten Entscheidungen in dem Projekt war die Auswahl der passenden Embedding-Funktion. Da bei einer Retrieval-Augmented-Generation-Architektur die Embedding-Komponente darüber entscheidet, welche Textabschnitte aus der Datenbank für die Antwort verwendet werden, hat sie einen direkten Einfluss auf die Qualität der Ergebnisse. Das Sprachmodell selbst kann nur dann eine passende Antwort formulieren, wenn es vorher relevante Informationen geliefert bekommt – und genau dafür ist die Embedding-Funktion verantwortlich[Red23], [Kum24].

Es wurde deshalb intensiv recherchiert, welche der von Ollama unterstützten Embedding-Modelle sich besonders gut für Anwendungen wie Frage-Antwort-Systeme, Chatbots und Text Retrieval eignen. Eine zusätzliche Herausforderung war, dass die Datenbank Inhalte

in zwei Sprachen enthält – Deutsch und Englisch. Viele Modelle sind primär auf Englisch trainiert, was bei gemischten Inhalten schnell zu Qualitätseinbußen führen kann. Es war daher besonders wichtig, ein Modell zu finden, das auch mit deutschsprachigen Inhalten zuverlässig umgehen kann.

Anfangs wurden einige der vorgeschlagenen Standardmodelle getestet. Letztendlich wurde jedoch den praktischen Weg gegangen und wurde nach dem Prinzip „Trial and Error“ mehrere Embedding-Modelle aus der Ollama-Liste ausprobiert – mit dem Ziel, das Modell zu finden, das in dem konkreten Fall die besten Ergebnisse liefert.

Option 1: stanus74/e5-base-sts-en-de
[Hei23], [sta24], [Hei24]
Dieses Modell ist ein multilingualer Sentence Embedding Encoder, der auf dem E5-Ansatz basiert und speziell für Semantic Textual Similarity (STS) zwischen Englisch und Deutsch entwickelt wurde. Es wurde primär für folgende Aufgaben konzipiert:

- Vergleich von zwei Sätzen – auch in unterschiedlichen Sprachen – auf semantische Übereinstimmung
- Verbindung deutschsprachiger Fragen mit englischen Dokumenten (und umgekehrt)
- Erkennung ähnlicher oder doppelter Inhalte in mehrsprachigen Datensätzen
- Embedding-basierte Suche, z.B. in FAQ-Systemen

Überraschend war, dass dieses Modell – obwohl es nicht direkt für Chatbots oder klassische QA-Systeme optimiert ist – in dem Projekt sehr gute Ergebnisse geliefert hat. Besser sogar als einige Modelle, die speziell für den Chatbot-Einsatz entwickelt wurden.

Option 2: bge-m3
[CXZ+24a], [CXZ+24b]
Dieses Modell wurde vom Beijing Academy of Artificial Intelligence (BAAI) entwickelt und ist ein modernes, multilinguales Multitask-Modell, das für eine Vielzahl von Aufgaben eingesetzt werden kann – unter anderem:

- Dense Retrieval bei Fragen und Dokumenten
- Klassifikation, Ähnlichkeitsanalyse und Topic Modeling
- Frage-Antwort-Systeme als Teil einer RAG-Architektur

Besonders hervorzuheben ist die breite Sprachunterstützung: bge-m3 wurde auf mehr als 100 Sprachen trainiert, darunter auch Deutsch. Außerdem gehört es laut aktuellen Benchmarks (z.B. MTEB) zu den leistungsstärkeren Modellen für Retrieval-Aufgaben – selbst ohne Feintuning[CXZ+24a], [MTMR22]. Nach mehreren Tests und Auswertungen hat sich das Modell bge-m3 als die beste Lösung für das Projekt herausgestellt. Es erfüllte nicht nur die funktionalen Anforderungen, sondern konnte auch im mehrsprachigen Kontext überzeugen. Die detaillierten Evaluationsergebnisse und die Auswirkungen auf die Antwortqualität wird im folgenden Kapitel näher darstellen.

4.1.5. Haupt-Modell

[Tea24b], Für das Hauptmodell, also das Large Language Model (LLM), wurden im Verlauf des Projekts mehrere Varianten getestet. Am überzeugendsten waren dabei zwei Modelle: Gemma und LLaMA. Nach umfangreichen Vergleichen erwies sich Gemma als die geeignetste Wahl, da es im konkreten Anwendungsszenario eines Chatbots die besten Ergebnisse lieferte.

Gemma ist ein modernes Sprachmodell, das besonders gut für Aufgaben wie Frage-Antwort-Systeme, Informationssuche (Retrieval QA) und strukturierte Antworten geeignet ist. Es kann präzise und kontextbezogene Texte erzeugen. Gleichzeitig braucht es weniger Rechenleistung als viele größere Modelle.[TRP+24]. Für den Chatbot war dieses Modell sehr gut geeignet, weil es einerseits qualitativ gute und sprachlich passende Antworten gegeben hat. Andererseits konnte es auch auf der vorhandenen Infrastruktur, besonders dem Server von der Universität, noch schnell und ohne Probleme laufens.[Tea24a]

Die konkrete Evaluierung der getesteten Sprachmodelle und ein direkter Vergleich der Ergebnisse zwischen Gemma und LLaMA erfolgen im nächsten Kapitel.

Zur Steuerung des Modells wurde ein kompaktes, aber wirkungsvolles Prompt eingesetzt, das einerseits die Genauigkeit der Antwort erhöht und andererseits eine zu lange Ausgabe vermeidet. Dabei wurde der folgende Prompt verwendet:

```
QA_PROMPT = """
You are an assistant for question-answering tasks. Use the following retrieved context to answer the question. Be concise.
Answer the question in the same language in which the question was asked.
Context: {context}
Question: {question}
Answer:
"""
```

Figure 4.7.: Prompt

Dieser Prompt wurde so formuliert, dass:

- das Modell als spezialisierter Assistent für QA-Aufgaben agiert,
- ausschließlich der zuvor gefundene Kontext (aus der Chroma-Datenbank) berücksichtigt wird,
- die Antwort knapp, aber korrekt formuliert wird,
- und die Sprache der Antwort automatisch der Sprache der gestellten Frage angepasst wird – was besonders wichtig ist, da in meiner Anwendung sowohl deutsche als auch englische Eingaben vorkommen können.

Diese Kombination aus leistungsfähigem Modell und präzise definiertem Prompt war entscheidend für die Qualität und Konsistenz der Chatbot-Antworten im Projekt.

4.2. Schnittstellen

Während der Entwicklung des Chatbots wurde das System auf zwei unterschiedlichen Rechnern betrieben. Die ersten Schritte und Tests wurden auf dem privaten Gerät durchgeführt – einem Lenovo Thinkpad 15v Gen 2i, das über eine Standard-CPU(NVIDIA RTX A2000) verfügte.

Dabei zeigte sich schnell, dass auch kleinere Sprachmodelle auf dieser Hardware nur eingeschränkt nutzbar waren. Selbst bei einfachen Fragen lagen die Antwortzeiten regelmäßig zwischen 1,5 und 2 Minuten – was für eine interaktive Anwendung wie einen Chatbot inakzeptabel ist.

Um die Systemleistung und die Nutzerfreundlichkeit deutlich zu verbessern, wurde mir daraufhin Zugriff auf einen universitären Server gewährt. Dieser war mit einer leistungsstarken NVIDIA A100 GPU (80GB PCIe) ausgestattet und ermöglichte es, auch komplexere Modelle mit akzeptabler Antwortzeit auszuführen.

5. Evaluation & Ergebnisse

5.1. Testmethoden

Zur Bewertung des Chatbot-Prototyps wurde eine systematische Evaluation mit realistischen Nutzerfragen durchgeführt.
Dazu wurden zehn typische Fragen aus dem universitären Alltag ausgewählt. Das Model wird nach folgende Kriterien beurteilt:

Genauigkeit: Bewertet wird, ob die vom System generierte Antwort inhaltlich korrekt und relevant zur gestellten Frage ist. Die Bewertung erfolgt skaliert: in den Stufen *falsch, teilweise korrekt, vollständig korrekt.*

Verständlichkeit: Dieses Kriterium misst, wie klar, strukturiert und sprachlich verständlich die Antwort für typische Nutzer:innen formuliert ist. Dazu zählen auch Faktoren wie Grammatik, Lesbarkeit und sprachliche Kohärenz. Bewertet wird auf einer Skala von 1 (schlecht verständlich) bis 5 (sehr gut verständlich).

Antwortdauer: Die Antwortzeit wird in Sekunden gemessen und umfasst die Zeitspanne vom Absenden der Anfrage bis zur vollständigen Ausgabe der Antwort. Sie dient als Indikator für die technische Effizienz des Systems unter realistischen Bedingungen.

Gesamtbewertung: Diese subjektive Bewertung berücksichtigt den Gesamteindruck der Antwort aus Nutzersicht, d.h. eine Kombination aus Richtigkeit, Nützlichkeit, Stil und Vertrauen in die Antwort. Sie wird auf einer Skala von 1 (ungenügend) bis 5 (sehr gut) vergeben.

Table 5.1.: Evaluationsfragen für den Chatbot (Deutsch und Englisch)

Nr.	Frage	Sprache	Typ	Schwierigkeit
1	Wie lauten die Kontaktdaten von Professor Peter Reichl?	Deutsch	Personenbezogen	Einfach
2	Wie lauten die Öffnungszeiten und Kontaktdaten des SSC der Fakultät Informatik?	Deutsch	Administrativ	Mittel
3	Welche Teilnahmevoraussetzungen gibt es für das Softwarepraktikum im Rahmen der Bachelorarbeit?	Deutsch	Studienrechtlich	Hoch
4	Wie viele ECTS-Punkte hat das Fach PR2 (Programmieren 2)?	Deutsch	Curriculum	Einfach
5	Welche Aufnahmeverfahren gibt es generell an der Fakultät für Informatik?	Deutsch	Zulassung	Mittel
6	What courses does Professor Peter Kalchgruber teach?	Englisch	Personenbezogen	Einfach
7	How much is the tuition fee for non-EU students at the University of Vienna?	Englisch	Finanzen	Mittel
8	What courses are recommended in the third semester of the Bachelor in Business Informatics?	Englisch	Curriculum	Hoch
9	What is STEOP and why is it important?	Englisch	Studienbegriff	Mittel
10	Who is the dean of the Faculty of Computer Science and how can I contact them?	Englisch	Personenbezogen	Mittel

5.1.1. Vergleich von Embedding Modelle

Für die semantische Suche innerhalb der Wissensdatenbank wurden zwei verschiedene Embedding-Modelle getestet und miteinander verglichen: bge-m3 und e5-base-sts-en-de. Beide Modelle wurden unter identischen Bedingungen evaluiert. Als Hauptmodell zur Antwortgenerierung wurde in beiden Fällen dasselbe LLM Gemma3:27b verwendet, sodass ausschließlich die Leistung der Embedding-Komponente beurteilt werden konnte.

		bge-m3		
Frage	Genauigkeit	Verständlichkeit	Antwortdauer (s)	Gesamtbewertung
1	vollständig korrekt	5	4	5
2	vollständig korrekt	5	4	5
3	vollständig korrekt	5	3	5
4	vollständig korrekt	5	4	5
5	vollständig korrekt	5	5	5
6	vollständig korrekt	5	3	5
7	vollständig korrekt	5	4	5
8	FALSCH	3	6	1
9	vollständig korrekt	5	4	5
10	vollständig korrekt	5	5	5

Figure 5.1.: Modellvergleich: Embedding-Funktionen bge-m3

		e5-base-sts-en-de		
Frage	Genauigkeit	Verständlichkeit	Antwortdauer (s)	Gesamtbewertung
1	vollständig korrekt	5	5	5
2	vollständig korrekt	5	4	5
3	vollständig korrekt	5	6	5
4	vollständig korrekt	5	5	5
5	vollständig korrekt	5	5	5
6	vollständig korrekt	5	4	5
7	FALSCH	2	6	1
8	FALSCH	2	6	1
9	vollständig korrekt	5	3	5
10	vollständig korrekt	5	3	5

Figure 5.2.: Modellvergleich: Embedding-Funktionen e5-base-sts-en-de

Als Ergebnis der vergleichenden Evaluation lässt sich festhalten, dass das erste getestete Modell, bge-m3, 9 von 10 Nutzerfragen vollständig korrekt beantworten konnte. Dies entspricht einer Trefferquote von 90%. Das zweite Modell, e5-base-sts-en-de, erzielte mit 8 von 10 korrekt beantworteten Fragen ebenfalls ein solides Ergebnis von 80%. Beide Modelle zeigten insgesamt eine gute Performance, wobei bge-m3 hinsichtlich Genauigkeit und semantischer Treffergenauigkeit leicht überlegen war.

Auf Basis dieser Resultate fiel die Entscheidung, im finalen Prototyp des Chatbots das Modell bge-m3 als Standard-Embedding-Funktion zu integrieren.

Das vollständige Antwortprotokoll beider Modelle sowie die genaue Bewertung aller Antworten sind im Anhang dieser Arbeit dokumentiert (siehe Anhang).

5.1.2. Vergleich der Hauptmodelle

Neben dem Embedding-Modul wurde auch die Leistung verschiedener Large Language Models (LLMs) zur Antwortgenerierung evaluiert. Dabei kamen drei gängige Modelle zum Einsatz:

- Gemma3:27b (17 GB)
- LLaMA3.1:8b (4.9 GB)
- Gemma3:1b (815 MB)

Alle Tests wurden unter identischen Bedingungen durchgeführt, wobei das Embedding-Modell bge-m3 konstant blieb. Ziel war es, die inhaltliche Qualität, sprachliche Verständlichkeit, Antwortgeschwindigkeit sowie den Gesamteindruck der Antworten pro Modell zu vergleichen.

		Gemma3:27b		
Frage	Genauigkeit	Verständlichkeit	Antwortdauer	Gesamtbewertung
1	vollständig korrekt	5	4	5
2	vollständig korrekt	5	4	5
3	vollständig korrekt	5	3	5
4	vollständig korrekt	5	4	5
5	vollständig korrekt	5	5	5
6	vollständig korrekt	5	3	5
7	vollständig korrekt	5	4	5
8	FALSCH	3	6	1
9	vollständig korrekt	5	4	5
10	vollständig korrekt	5	5	5

Figure 5.3.: Modellvergleich: LLM gemma3:27b

		llama3.1:8b		
Frage	Genauigkeit	Verständlichkeit	Antwortdauer (s)	Gesamtbewertung
1	vollständig korrekt	5	4	5
2	vollständig korrekt	5	5	5
3	teilweise korrekt	3	4	3
4	vollständig korrekt	5	5	5
5	vollständig korrekt	5	3	5
6	vollständig korrekt	5	4	5
7	vollständig korrekt	5	5	5
8	FALSCH	2	4	1
9	vollständig korrekt	5	5	5
10	FALSCH	2	3	1

Figure 5.4.: Modellvergleich: LLM llama3.1:8b

		gemma3:1b		
Frage	Genauigkeit	Verständlichkeit	Antwortdauer (s)	Gesamtbewertung
1	vollständig korrekt	5	4	5
2	vollständig korrekt	5	2	5
3	FALSCH	1	1	1
4	vollständig korrekt	5	1	5
5	FALSCH	1	2	1
6	FALSCH	1	1	1
7	FALSCH	1	2	1
8	FALSCH	1	2	1
9	vollständig korrekt	3	2	4
10	FALSCH	3	2	1

Figure 5.5.: Modellvergleich: LLM gemma3:1b

Das stärkste Modell in dem Vergleich war Gemma 3:27b. Es konnte 9 von 10 Fragen richtig beantworten. Die Antworten waren nicht nur inhaltlich korrekt, sondern auch gut formuliert, verständlich und detailliert.

Das zweite getestete Modell war LLaMA 3.1:8b. Es hat insgesamt 7 von 10 Fragen richtig beantwortet. Zwei Antworten waren falsch, eine war nur teilweise korrekt. Insgesamt war die Qualität akzeptabel, aber im Vergleich zu Gemma 3:27b deutlich schwächer – vor allem, wenn es um genaues Verstehen von Kontext ging.

Das dritte Modell, Gemma 3:1b, gehört zwar zur gleichen Modellfamilie wie das erste, ist aber viel kleiner – es benötigt nur etwa 815MB Speicher (während Gemma 3:27b rund 17GB braucht). In den Tests war es das schwächste Modell: Es konnte nur 4 von 10 Fragen korrekt beantworten. Die restlichen Antworten waren oft ungenau oder haben wichtige Informationen weggelassen.

Aus diesen Ergebnissen kann man erkennen, dass die Qualität der Antworten nicht nur

vom Embedding-Modell, sondern stark auch von der Größe und Leistungsfähigkeit des Sprachmodells abhängt. Besonders bei schwierigen oder unklar formulierten Fragen reicht ein kleines Modell oft nicht aus, um gute und verlässliche Antworten zu geben.

5.2. Ergebnisse

Im Folgenden sind beispielhafte Antworten der getesteten Modelle dargestellt, so wie sie im User Interface des Chatbots angezeigt werden:

Figure 5.6.: Modellvergleich: LLM Gemma 3:27b, vollständig korrekte Antwort

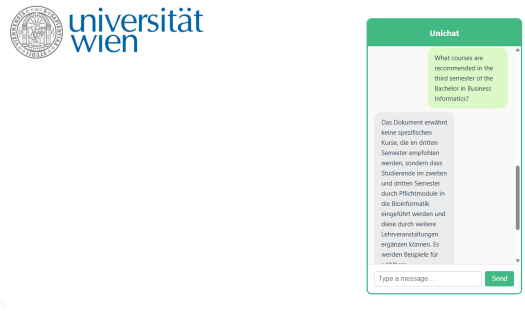

Figure 5.7.: Modellvergleich: LLM Gemma 3:27b, falsche Antwort

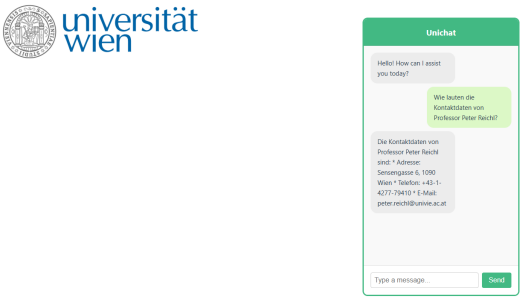

Figure 5.8.: Modellvergleich: LLM Llama3.1:8b, vollständig korrekte Antwort

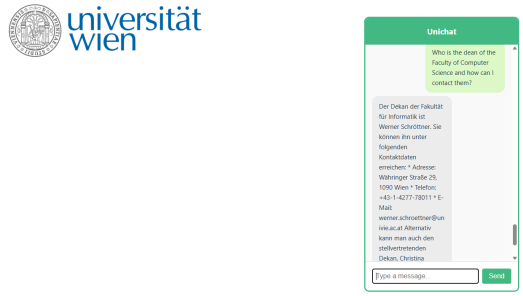

Figure 5.9.: Modellvergleich: LLM Llama3.1:8b, falsche Antwort

5.3. Die Erweiterung einer Wissensdatenbank (Knowledge Base)

Die Ergebnisse aus dem Evaluierungsteil beziehen sich auf den Zeitpunkt, zu dem die ursprüngliche Knowledge Base erstellt wurde. Diese Ergebnisse wurden im vorherigen Kapitel vorgestellt.

Im weiteren Verlauf der Forschung wurde aus experimentellen Gründen entschieden, die Knowledge Base zu erweitern. Zusätzliche Informationen zu Kursdetails (wie Ablauf, Terminplan, Standort und Lehrende) wurden hinzugefügt. Diese Informationen lagen im XML-Format vor. Da dieses Format bereits erfolgreich extrahiert und in das bestehende Datenwissen integriert wurde, wurden dabei keine größeren Schwierigkeiten erwartet.

Überraschenderweise konnte das Embedding-Modell (bge-m3) die neuen Informationen nicht vollständig verarbeiten. Während Angaben zu Standort und Lehrpersonen korrekt wiedergegeben wurden, war die Leistung bei Fragen zum Zeitplan und zu Terminen deutlich schlechter.

Zusätzlich zeigte sich ein weiterer Nachteil: Die Gesamtleistung anhand der Evaluierungsfragen verschlechterte sich. Fragen, die zuvor zuverlässig und korrekt beantwortet wurden, lieferten mit der erweiterten Knowledge Base teilweise schlechtere Ergebnisse.

Es wird vermutet, dass eine zu große Knowledge Base die Effizienz des Modells beeinträchtigen kann. Möglicherweise ist es notwendig, bei der Arbeit mit umfangreichen Daten andere Strategien zu verwenden – etwa durch den Einsatz von zwei getrennten Vektordatenbanken (z.B. ChromaDB). Eine Datenbank könnte bevorzugt abgefragt werden, während die zweite nur bei Bedarf berücksichtigt wird, wenn in der ersten keine passende Information gefunden wird.

Dies stellt einen möglichen nächsten Schritt in der Forschung dar und ist Teil der weiterführenden Fragestellung: „Verwendung mehrerer Vektordatenbanken (ChromaDB) für RAG-Systeme“.

6. Diskussion und zukünftige Arbeit

6.1. Herausforderungen bei der Implementierung

Bei der Entwicklung des Chatbots sind im Laufe des Projekts mehrere technische Hürden aufgetreten, insbesondere im Umgang mit Datenformaten und bei der Sicherstellung der Antwortqualität.

Ein zentrales Problem bestand in der Vielfalt der Datenquellen, aus denen die Wissensbasis aufgebaut wurde. Diese kamen aus ganz unterschiedlichen Formaten. Vor allem PDFs haben sich als besonders problematisch erwiesen. Lokale Sprachmodelle, besonders kleinere Varianten, hatten große Schwierigkeiten, Tabelleninhalte korrekt zu interpretieren und daraus sinnvollen Kontext zu ziehen.

Ein weiteres zentrales Thema war die Performance des Systems – vor allem in Bezug auf die Antwortqualität im Zusammenspiel mit der verfügbaren Rechenleistung. Hier zeigte sich deutlich: Je größer das Modell (also je mehr Parameter und je höher die Dateigröße), desto besser war in der Regel auch die Qualität der gelieferten Antworten.

Allerdings führte die Verwendung größerer Modelle gleichzeitig zu einem neuen Problem: dem Ressourcenbedarf. Je umfangreicher das Modell, desto mehr GPU-Leistung wurde benötigt – was wiederum die Antwortzeiten stark verlängerte. Auf dem lokalen Rechner (ein Lenovo ThinkPad) konnte man Modelle bis ca. 4GB sinnvoll nutzen. Erst durch den Zugang zum Uni-Server war es möglich, auch großere Modelle zu testen.

Letztlich bestand die Herausforderung darin, ein gutes Gleichgewicht zu finden: Ein Modell zu wählen, das leistungsfähig genug ist, um qualitativ hochwertige Antworten zu liefern – aber gleichzeitig so kompakt, dass es auch mit begrenzten Ressourcen noch akzeptable Antwortzeiten ermöglicht.

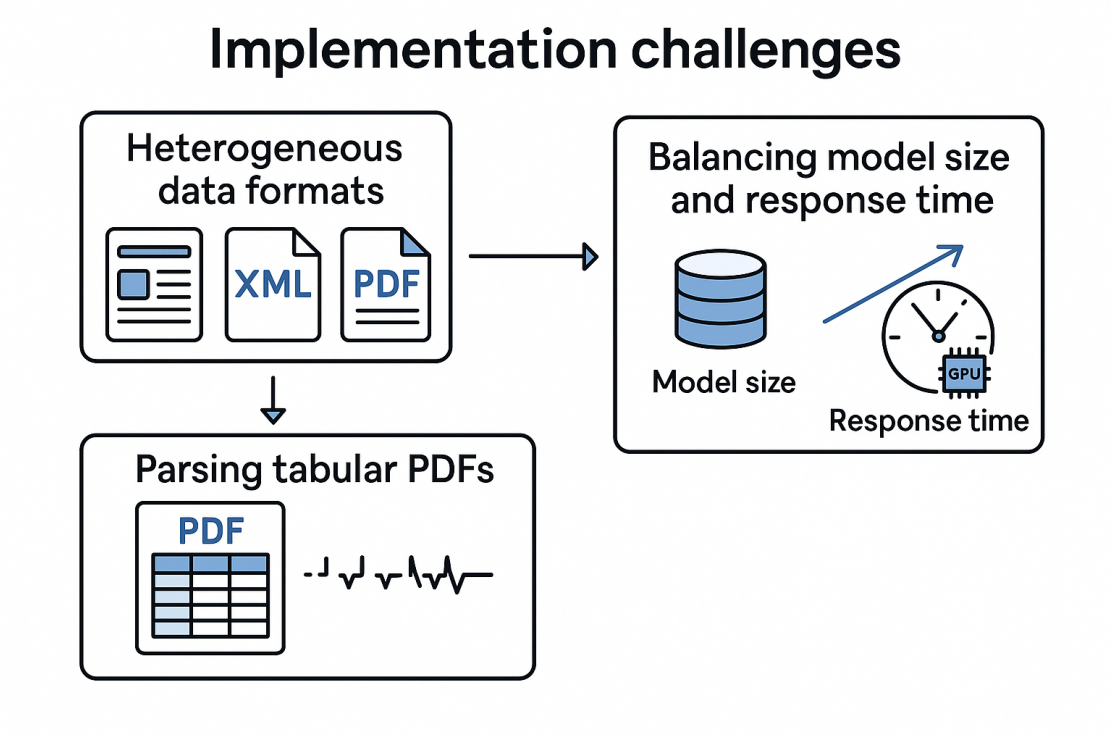

Figure 6.1.: Herausforderungen

6.2. Mögliche Erweiterungen

Im Verlauf des Projekts wurde intensiv darüber nachgedacht, ob eine Integration mit OpenAI sinnvoll wäre – sei es ergänzend zur lokalen Lösung oder als alternative Architektur. OpenAI bietet eine breite Palette an Optionen, die sich sowohl für kleinere Testszenarien als auch für professionelle Produktivsysteme eignen[Ope23a].

Einstieg mit OpenAI – für kleine Tests und Prototypen Für das Projekt – das als Beispielsystem konzipiert ist und über einen Zeitraum von etwa 4 Monaten entwickelt wurde – wäre der Kostenaufwand bei Nutzung von OpenAI sehr gering geblieben. Die Kombination aus kleiner Wissensdatenbank und moderatem Testvolumen lässt sich preislich nahezu vernachlässigen[Ope23b].

Parameter	**Wert**
Projektlaufzeit	4 Monate
Testfragen pro Woche	ca. 100
Anzahl Wochen	16
Gesamtanzahl Fragen	1.600
Durchschnittliche Anfragegröße	200 Tokens ($\tilde{1}$50–200 Wörter)
Durchschnittliche Antwortgröße	300 Tokens ($\tilde{2}$50 Wörter)
Tokens pro Frage+Antwort	ca. 500
Gesamt-Tokens	1.600 × 500 = **800.000**
Knowlegebase: Anzahl Chunks	3.000 á 1.000 Zeichen
Aufwand pro Retrieval-Aufruf	ca. 3.000 × 100 Tokens = 300k Tokens
Gesamtnutzung ca.	**1.1 Millionen Tokens**
Modell (z.B. `gpt-3.5-turbo`)	ca. \$0.0015 / 1.000 Tokens
Gesamtkosten (geschätzt)	**ca. 1.65 USD**

Professionelle Nutzung – OpenAI als robuste Cloud-Lösung Sollte das System produktiv eingesetzt werden – beispielsweise als studentischer Chatbot oder internes Recherchetool – ist OpenAI eine sehr attraktive Option: Modelle wie gpt-4-turbo liefern verlässlich qualitativ hochwertige Antworten, auch bei komplexen Themen. OpenAI kann bei Bedarf mehrere tausend Anfragen pro Stunde abdecken – ein entscheidender Vorteil gegenüber lokalen Systemen mit begrenzter Rechenkapazität. Dazu kommz auch keine Notwendigkeit zur Modellpflege oder Systemupdates – OpenAI übernimmt das alles.

Allerdings steigen damit auch die Kosten[Ope23b] – je nach Nutzerzahl, Anfragevolumen und Modellwahl. Für produktive Szenarien wird meist mit folgenden Preisstufen gearbeitet:

Modell	**Preis pro 1K Tokens (Input)**	**Preis pro 1K Tokens (Output)**
gpt-3.5-turbo	\$0.0010	\$0.0020
gpt-4-turbo	\$0.01	\$0.03
gpt-4 (legacy)	\$0.03	\$0.06

Je nach Anwendungsfall können diese Kosten schnell signifikant werden.

Es wurde schließlich zu dem Schluss gekommen, dass für den Anwendungsfall ein lokaler Ansatz besser geeignet ist. So gibt es die Möglichkeit, verschiedene Modelle selbst zu testen, kennenzulernen und gezielt zu evaluieren. Bei der Nutzung von OpenAI hingegen sind viele Entscheidungen – wie etwa die Wahl der Embedding-Funktion – bereits vorgegeben, was die Flexibilität deutlich einschränkt.

6.3. Datenschutz und ethische Fragen

Bei der Entwicklung eines Chatbots, der mit Nutzerdaten arbeitet, stellt sich unweigerlich die Frage nach dem Datenschutz und der ethischen Verantwortung. Insbesondere der

Umgang mit personenbezogenen Daten wie Namen, Kontaktdaten oder anderen sensiblen Informationen muss sorgfältig abgewogen werden – sowohl in Bezug auf rechtliche Vorgaben (z.B. DSGVO) als auch auf technischer und ethischer Ebene. In dieser Arbeit wurden zwei technische Ansätze gegenübergestellt:
1. Lokale Verarbeitung mit Ollama Bei diesem Ansatz wird ein Large Language Model (LLM) lokal auf einem eigenen Server betrieben. Alle Daten – inklusive der vektorisierten Datenbank – bleiben vollständig unter eigener Kontrolle. Die Verarbeitung erfolgt offline bzw. ohne externe Datenübertragung. Damit bietet dieser Ansatz maximale Datensouveränität und Kontrolle. Besonders im Hinblick auf sensible Daten ist dies aus datenschutzrechtlicher Sicht von Vorteil, da keine personenbezogenen Daten an Dritte übermittelt werden.

2. Cloudbasierte Verarbeitung mit OpenAI Alternativ besteht die Möglichkeit, ein externes LLM wie GPT-4 von OpenAI zu nutzen. In diesem Fall werden Anfragen und gegebenenfalls auch personenbezogene Daten über das Internet an Server außerhalb der eigenen Infrastruktur übermittelt und dort verarbeitet. Dies wirft die Frage auf, ob es zulässig ist, solche Daten an einen Drittanbieter weiterzugeben – insbesondere, wenn dieser außerhalb der EU ansässig ist und somit nicht direkt der DSGVO unterliegt. Diese Entscheidung ist nicht trivial: Auf der einen Seite erfordert die Nutzung externer Dienste ein hohes Maß an Vertrauen in die Datensicherheit und Compliance des Anbieters. Auf der anderen Seite lässt sich argumentieren, dass viele der verwendeten Daten ohnehin bereits öffentlich im Internet zugänglich sind. Dennoch ist der Kontext entscheidend: Selbst öffentlich zugängliche Informationen können in bestimmten Zusammenhängen als schützenswert gelten, insbesondere wenn sie automatisiert verarbeitet, kombiniert oder langfristig gespeichert werden.

Ethisch betrachtet geht es daher nicht nur um die reine Verfügbarkeit von Daten, sondern auch um den verantwortungsvollen Umgang damit. Transparenz gegenüber den Nutzenden, klare Zweckbindung und der Grundsatz der Datensparsamkeit sollten bei jeder Lösung berücksichtigt werden – unabhängig davon, ob sie lokal oder in der Cloud umgesetzt wird. Die Universität Wien betont, dass bei der Nutzung externer Dienste zur Verarbeitung personenbezogener Daten bestimmte Verfahren eingehalten werden müssen, um Rechtskonformität gemäß der DSGVO sicherzustellen. Dazu gehören unter anderem die Einholung einer Datenschutzfolgeabschätzung und der Abschluss eines Auftragsverarbeitervertrags.[Uni24]

7. Fazit

7.1. Zusammenfassung der Ergebnisse

Wenn man die Erfahrungen und Ergebnisse zusammenfasse, kommt man zu dem Schluss, dass man auch mit sehr begrenzten Ressourcen und ausschließlich mit kostenlosen Tools, Open-Source-Modellen und frei verfügbaren Daten bereits relativ gute Resultate erreichen kann.

Im Bereich der Künstlichen Intelligenz – und besonders bei Chatbots – passiert im Moment sehr viel. Die Entwicklung geht sehr schnell, und es erscheinen fast jeden Monat neue Modelle oder aktualisierte Versionen, die auf anderen Daten basieren und dadurch ganz andere Ergebnisse liefern können.

Zum Glück gibt es viele offene Ressourcen, die Studierende unterstützen können: von wissenschaftlichen Artikeln, über YouTube-Videos, bis hin zu hilfreichen Communities. Besonders auffällig ist, dass ein großer Teil der relevanten wissenschaftlichen Arbeiten erst in den Jahren 2023 und 2024 veröffentlicht wurde. Das zeigt, dass viele dieser Technologien noch sehr neu sind und die Verbindung zwischen LLMs und neuen Architekturen gerade erst richtig erforscht wird.

Auf der anderen Seite wurde im Laufe des Projekts auch deutlich, dass die gewählte Infrastruktur – also kleine lokale Modelle auf begrenzter Hardware – nicht ausreichen würde, wenn der Chatbot später universitätsweit eingesetzt werden soll.

Die Universität Wien hat pro Jahr etwa 85.000 Studierende und über 10.000 Mitarbeiterinnen und Mitarbeiter[Uni25g]. Dazu kommen noch Studieninteressierte und externe Anfragen. Für so eine große Nutzergruppe wäre es schwierig, die Systemleistung und Antwortqualität allein mit kleinen Modellen dauerhaft sicherzustellen.

Aus den gewonnenen Erkenntnissen lässt sich ableiten, dass ein KI-gestützter Chatbot auf Fakultätsebene bei ausreichender Serverkapazität, zuverlässiger Wartung und sorgfältiger Systempflege bereits sehr leistungsfähig und praxistauglich betrieben werden kann. In diesem Rahmen kann eine lokale Infrastruktur durchaus ausreichen, um eine stabile, datenschutzkonforme und ressourcenschonende Lösung bereitzustellen.

Sollte ein solches System jedoch auf Universitätsebene zum Einsatz kommen, wären deutlich höhere Anforderungen an Skalierbarkeit, Rechenleistung und Infrastruktur zu erwarten. In diesem Fall müsste auch der technische Betrieb entsprechend professionell organisiert sein. Eine cloudbasierte Lösung – etwa über spezialisierte Anbieter wie OpenAI – könnte hier eine sinnvolle Alternative darstellen, da sie Zugriff auf leistungsstärkere Modelle sowie flexible Skalierungsmöglichkeiten bietet.

Voraussetzung für einen solchen Cloud-Einsatz wäre jedoch eine klare Klärung der datenschutzrechtlichen und rechtlichen Rahmenbedingungen. Werden diese Anforderungen

erfüllt, kann eine Cloud-basierte Lösung eine effektive und zukunftsfähige Option für den universitätsweiten Einsatz eines intelligenten Chatbots darstellen.

7.2. Bedeutung für Universitäten und Studierende

Die Bedeutung eines solchen Projekts hat sich in den letzten Jahren deutlich verändert. Hätte man im Jahr 2022 die Frage gestellt, ob ein KI-gestützter Chatbot für Universitäten nützlich ist, hätten viele wahrscheinlich geantwortet: „Nice to have“ – aber nicht wirklich notwendig. Zu dieser Zeit hatten viele Menschen eher schlechte Erfahrungen mit Chatbots gemacht, und ihr praktischer Nutzen war begrenzt. Kaum jemand hat sie regelmäßig genutzt.

Mit dem Aufkommen von ChatGPT[Ope25] hat sich das jedoch grundlegend verändert. Im Jahr 2025 ist es für viele Studierende kaum noch vorstellbar, ihren Studienalltag ohne KI-gestützte Werkzeuge zu organisieren. Solche Tools werden mittlerweile täglich eingesetzt – sei es zur Informationssuche, zum Verfassen von Texten, zur Planung oder zum Lernen.

Diese Entwicklung betrifft nicht nur die Hochschulwelt. Auch in der Arbeitswelt lässt sich eine klare Tendenz erkennen: Viele Unternehmen setzen inzwischen KI-gestützte Chatbots ein, die speziell auf die internen Arbeitsprozesse und Fragestellungen der Mitarbeiterinnen und Mitarbeiter abgestimmt sind. Dadurch entstehen enorme Effizienzgewinne – sowohl im Support als auch in der internen Kommunikation.

Für den universitären Bereich bedeutet das: Ein Chatbot, der gezielt auf hochschulrelevante Daten trainiert wurde und die Bedürfnisse von Studierenden, Lehrenden und Verwaltungspersonal kennt, könnte eine sehr große Unterstützung sein. Ein gut umgesetztes und kontinuierlich gepflegtes System dieser Art kann potenziell zu einer spürbaren Entlastung und Verbesserung für die gesamte Universitätsgemeinschaft beitragen.

7.3. Abschlussbemerkungen & mögliche nächste Schritte

Die Entwicklung des Projekts stellte einen intensiven und lehrreichen Prozess dar, bei dem technisches Wissen im Bereich künstlicher Intelligenz, Webentwicklung sowie Systemarchitektur vertieft wurde. Im Verlauf der Arbeit wurden verschiedene KI-Modelle direkt miteinander verglichen und praxisnahe Erfahrungen in den Bereichen Datenverarbeitung, Systemperformance und Benutzerfreundlichkeit gesammelt.

Obwohl der entwickelte Chatbot ursprünglich als Prototyp im Rahmen einer Bachelorarbeit konzipiert wurde, zeigt das System Potenzial für den praktischen Einsatz im Hochschulkontext. Es könnte künftig dazu beitragen, Studierende im Studienalltag effektiv zu unterstützen und den Zugang zu Informationen zu erleichtern.

Für die nächsten Schritte sehe ich folgende mögliche Erweiterungen:

- Ein nächster Schritt in der Forschung wird die mögliche Erweiterung der Systemarchitektur um zusätzliche Vektordatenbanken sein. Ziel ist es, die Effizienz und Antwortqualität von RAG-Systemen bei wachsendem Datenvolumen zu verbessern.

- Integration in bestehende Uni-Systeme, z.B. über eine API-Schnittstelle zu u:space oder Moodle.
- Feintuning des Modells mit echten Nutzungsdaten, um noch spezifischer auf typische Fragen reagieren zu können.
- Verbesserung der Antwortgeschwindigkeit.
- Einsatz als Mobile App oder Web-Widget, um den Zugang noch einfacher zu machen.

Langfristig wäre es interessant zu untersuchen, wie ein solcher Chatbot im täglichen Betrieb angenommen wird und wie er das Informationsverhalten von Studierenden verändert. Auch Fragen zur Verantwortung beim Einsatz von KI an Bildungseinrichtungen wären dabei ein wichtiges Thema.

Bibliography

[BAEV+22] Juliana Binfoh Abuaa, Harry Essel, Dimitrios Vlachopoulos, Akosua Tachie-Menson, Esi Johnson, and Papa Kwame. The impact of a virtual teaching assistant (chatbot) on students' learning in ghanaian higher educationthe impact of a virtual teaching assistant (chatbot) on students' learning in ghanaian higher education'. *International Journal of Educational Technology in Higher Education*, 19, 11 2022.

[Bak24] Thilini Bakmeedeniya. Leveraging conversational ai, specifically chatgpt, for enhanced learning experiences: Exploring challenges and proposing mitigation strategies. 9, 01 2024.

[BFPN17] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 2017.

[Chr23] Chroma. Chroma – the ai-native open-source vector database. `https://www.trychroma.com/`, 2023.

[CJC+24] Christina, Julia, Carina, Carmen, Tobias, and Brahim. Chatgpt – ein ki-tool, das die welt der studierenden komplett verändert hat?, 2024.

[Clo25] Google Cloud. Conversational ai on google cloud. `https://cloud.google.com/products/conversational-agents`, 2025.

[CXZ+24a] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[CXZ+24b] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[DLG+23] Yuhao Dan, Zhikai Lei, Yiyang Gu, Yong Li, Jia-Peng Yin, Jiaju Lin, Linhao Ye, Zhiyan Tie, Yougen Zhou, Yilei Wang, Aimin Zhou, Zeyang Zhou, Qin Chen, Jie Zhou, Liang He, and Xipeng Qiu. Educhat: A large-scale language model-based chatbot system for intelligent education. *ArXiv*, abs/2308.02773, 2023.

[Doc25] Docker Inc. Docker – accelerated container application development. https://www.docker.com, 2025.

[FA24] Bradley Freeman and Kumiko Aoki. Chatgpt in education: A comparative study of media framing in japan and malaysia. page 26–32, 2024.

[Fak25] Fakultät für Informatik, Universität Wien. Fakultät für informatik – universität wien. https://informatik.univie.ac.at/, 2025.

[Ful23] Wojtek Fulmyk. Text tokenization and vectorization in nlp. https://medium.com/@WojtekFulmyk/text-tokenization-and-vectorization-in-nlp-ac5e3eb35b85, 2023.

[Gom22] C K Gomathy. Artificial intelligence chatbot using python. 05 2022.

[GRCG24] Rui Guan, Mladen Rakovic, Guanliang Chen, and Dragan Gasevic. How educational chatbots support self-regulated learning? a systematic review of the literature. *Education and Information Technologies*, 30:4493–4518, 08 2024.

[HEH24] Achraf Hsain and Hamza El Housni. Large language model-powered chatbots for internationalizing student support in higher education. 03 2024.

[Hei23] Daniel Heinz. e5-base-sts-en-de: Multilingual e5-base model fine-tuned for semantic textual similarity. https://huggingface.co/danielheinz/e5-base-sts-en-de, 2023. Fine-tuned on German subsets of paraphrase and STS datasets using Multiple Negatives Ranking Loss and Cosine Similarity Loss.

[Hei24] Daniel Heinz. e5-base-sts-en-de: Multilingual e5-base model fine-tuned for semantic textual similarity. https://huggingface.co/danielheinz/e5-base-sts-en-de, 2024. Feinabgestimmt auf deutschen Datensätzen für semantische Textähnlichkeit.

[Ins23] ICT Institute. History of artificial intelligence: The turing test, 2023.

[K20] Ashok K. Smart college chatbot using ml and python. 12 2020.

[Kag25] Kaggle. Prompt engineering whitepaper. https://www.kaggle.com/whitepaper-prompt-engineering, 2025.

[Kum24] Shravan Kumar. Mastering rag: A deep dive into embeddings. https://medium.com/@shravankoninti/mastering-rag-a-deep-dive-into-embeddings-b78782aa1259, 2024.

[Lan24] LangChain. Langchain documentation – prompts module. https://python.langchain.com/v0.1/docs/modules/model_io/prompts/, 2024. Accessed: 2025-05-10.

[LGM23] Lasha Labadze, Maya Grigolia, and Lela Machaidze. Role of ai chatbots in education: systematic literature review. *International Journal of Educational Technology in Higher Education*, 20, 10 2023.

[Li24] Brian Li. Vue.js: The progressive javascript framework. `https://kinsta.com/blog/vue-js/`, 2024.

[LKH24] Fei Liu, Zejun Kang, and Xing Han. Optimizing rag techniques for automotive industry pdf chatbots: A case study with locally deployed ollama models, 2024.

[Mic25] Microsoft. Bot framework sdk. `https://github.com/microsoft/botframework-sdk`, 2025.

[MJ09] James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, volume 23. Pearson/Prentice Hall Upper Saddle River, 2009.

[MS24] Mashilo Modiba and Mahlatse Shekgola. Utilising artificial intelligence chatbots for student support at comprehensive open distance e-learning higher learning institutions in the fifth industrial revolution. *Journal of Education, Society & Multiculturalism*, 5:26–48, 06 2024.

[MTMR22] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. `https://arxiv.org/abs/2210.07316`, 2022.

[Notnd] NoteGPT. Pdf to markdown converter. `https://notegpt.io/pdf-to-markdown-converter`, n.d.

[Oll24a] Ollama. all-minilm. `https://ollama.com/library/all-minilm`, 2024.

[Oll24b] Ollama. Bge-m3. `https://ollama.com/library/bge-m3`, 2024.

[Oll24c] Ollama. mxbai-embed-large. `https://ollama.com/library/mxbai-embed-large`, 2024.

[Oll24d] Ollama. nomic-embed-text. `https://ollama.com/library/nomic-embed-text`, 2024.

[Oll25a] Ollama. shaw/dmeta-embedding-zh. `https://ollama.com/shaw/dmeta-embedding-zh`, 2025. Zugriff am 31. Mai 2025.

[Oll25b] Ollama Inc. Ollama – run large language models locally, 2025. Zugriff am 10. Mai 2025.

[Ope23a] OpenAI. Models - openai api documentation, 2023.

[Ope23b] OpenAI. Pricing - openai api documentation, 2023. Zugriff am 1. Juni 2025.

[Ope25] OpenAI. Chatgpt. `https://chatgpt.com`, 2025.

[Pal24] Pallets Projects. Flask documentation (version 2.x), 2024.

[Ras25] Rasa Technologies GmbH. Rasa – open source conversational ai. `https://rasa.com`, 2025.

[Red23] Red Hat. Was ist retrieval-augmented generation (rag)?, 2023.

[RT23] Harald Ritz and Dogus Tansel. Entwicklung eines ki-basierten faq-chatbots für die hochschule im bereich prüfungsangelegenheiten. *Anwendungen und Konzepte der Wirtschaftsinformatik*, 17:81–92, 2023.

[Sch11] Michael Scharkow. *Automatisierte Inhaltsanalyse: Eine methodische Einführung für die Kommunikationswissenschaft*. PhD thesis, Universität der Künste Berlin, 2011.

[SML24] Odin Monrad Schei, Anja Mogelvang, and Kristine Ludvigsen. Perceptions and use of ai chatbots among students in higher education: A scoping review of empirical studies. *Education Sciences*, 14:922, 08 2024.

[Sno24] Snowflake. snowflake-arctic-embed. `https://ollama.com/library/snowflake-arctic-embed`, 2024. Zugriff am 31. Mai 2025.

[SSC+24] Ramteja Sajja, Yusuf Sermet, Muhammed Cikmaz, David Cwiertny, and Ibrahim Demir. Artificial intelligence-enabled intelligent assistant for personalized and adaptive learning in higher education. *Information*, 15:596, 09 2024.

[sta24] stanus74. e5-base-sts-en-de modell auf ollama. `https://ollama.com/stanus74/e5-base-sts-en-de`, 2024. Bereitgestellt über Ollama, basiert auf dem Modell von Daniel Heinz auf Hugging Face.

[Stu25a] Studienvertretung Informatik, Universität Wien. Faq – häufig gestellte fragen der studienvertretung informatik, 2025.

[Stu25b] Studienvertretung Informatik, Universität Wien. Stv & fv informatik – studienvertretung der fakultät für informatik, 2025.

[Stu25c] Studo GmbH. Studo – digitale lösungen für studierende und hochschulen, 2025.

[SV24] Tobias Seidl and Cornelia Vonhof. Studentische nutzung von ki-tools im hochschulalltag: Kontinuitäten und veränderungen, 2024.

[Tea24a] Gemma Team. Gemma 2: Improving open language models at a practical size, 2024.

[Tea24b] Gemma Team. Gemma: Open models based on gemini research and technology. `https://arxiv.org/abs/2403.08295`, 2024.

[The23] Ravi Theja. Evaluating the ideal chunk size for a rag system using llamaindex. `https://www.llamaindex.ai/blog/evaluating-the-ideal-chunk-size-for-a-rag-system-using-llamaindex-6207e5d3fec5`, 2023.

[TRP+24] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Lan, Sammy Jerome, and Alek Andreev. Gemma 2: Improving open language models at a practical size. 2024.

[Uni24] Universität Wien. DSGVO-Konformität – Hilfestellungen für Forschung und Lehre, 2024.

[Uni25a] Universität Wien. Ki in studium und lehre, 2025.

[Uni25b] Universität Wien. Mentoringprogramm der fakultät für informatik, 2025.

[Uni25c] Universität Wien. Studienprogrammleitung der fakultät für informatik, 2025.

[Uni25d] Universität Wien. Studienservicecenter (ssc) informatik, 2025.

[Uni25e] Universität Wien. u:find – vorlesungsverzeichnis und personenverzeichnis, 2025.

[Uni25f] Universität Wien. u:space – gastzugang und Überblick über services, 2025.

[Uni25g] Universität Wien. Zahlen, daten & broschüren. `https://www.univie.ac.at/ueber-uns/auf-einen-blick/zahlen-daten-broschueren/`, 2025.

[vGM25] Joerg von Garrel and Jana Mayer. Künstliche intelligenz im studium: Eine quantitative längsschnittstudie zur nutzung ki-basierter tools durch studierende (2023 & 2025), 2025.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017.

[Vue24] Vue.js Team. Vue.js official tutorial, 2024. Zugriff am 7. Mai 2025.

[YWP24] Xianjun Yang, Stephen D. Wilson, and Linda Petzold. Quokka: An open-source large language model chatbot for material science, 2024.

[Zen25a] Zentraler Informatikdienst der Universität Wien. Ki-software – u:ai, 2025.

[Zen25b] Zentraler Informatikdienst, Universität Wien. Zid-helpdesk – it-support für universitätsangehörige, 2025.

A. Anhang

A.1. Antwortprotokoll der Modelle

A.1.1. Embedding Model bge-m3

1. E-Mail: peter.reichl@univie.ac.at Telefon: +43-1-4277-79410
2. Öffnungszeiten: Mo-Do: 9:00-14:00, Fr: 9:00-13:00 Telefon: +43-1-4277-781 01 E-Mail: ssc.informatik(at)univie.ac.at Adresse: Währinger Straße 29, 1090 Wien
3. Für Informatik-Studierende sind SE1, NUM, EST und MM erfolgreich abgeschlossen sein. Für Wirtschaftsinformatik-Studierende sind StEOP, EIS, DSE, IDS, NET, VWL, EST, HCI, SE1 und MM erforderlich.
4. 6 ECTS.
5. Für die Bachelorstudien Informatik und Wirtschaftsinformatik gibt es ein mehrstufiges Aufnahmeverfahren. Es hängt von Staatsangehörigkeit, Ausstellungsland des Reifezeugnisses und Deutschkenntnissen ab.
6. Professor Peter Kalchgruber unterrichtet Modellierung, Information Management & Systems Engineering, Multimedia and Semantic Technologies, Digital Media Production und Softwarepraktikum mit Bachelorarbeit.
7. Für Studierende aus Nicht-EU/EWR-Ländern beträgt der Studienbeitrag 726,72 Euro pro Semester, zuzüglich 25,20 Euro ÖH-Beitrag, also insgesamt 751,92 Euro.
8. Das Dokument erwähnt keine spezifischen Kurse, die im dritten Semester empfohlen werden, sondern dass Studierende im zweiten und dritten Semester durch Pflichtmodule in die Bioinformatik eingeführt werden und diese durch weitere Lehrveranstaltungen ergänzen können. Es werden Beispiele für wählbare Ausprägungsfächer genannt, wie Numerical High Performance Algorithms, Gaming Technologies, Image Processing & Image Analysis, etc.
9. STEOP ist die Studieneingangs- und Orientierungsphase, bestehend aus den Modulen Programmierung 1, Technische Grundlagen der Informatik und Mathematische Grundlagen der Informatik 1 (insgesamt 18 ECTS). Sie ist wichtig, da man sie erfolgreich abschließen muss, um uneingeschränkten Zugang zu allen Lehrveranstaltungen zu erhalten.
10. Wilfried Gansterer ist der Dekan der Fakultät für Informatik. Sie können ihn/sie unter dekanat.informatik@univie.ac.at oder +43-1-4277-78001 erreichen.

A.1.2. Embedding Model e5-base-sts-en-de

1. E-Mail: peter.reichl@univie.ac.at Telefon: +43-1-4277-79410
2. **Öffnungszeiten:** Mo-Do: 9:00-14:00, Fr: 9:00-13:00 **Telefon:** +43-1-4277-781 01 **E-Mail:** ssc.informatik@univie.ac.at **Adresse:** Währinger Straße 29-31
3. SE1, NUM, EST, MM.
4. 6 ECTS.
5. Für die Bachelorstudien Informatik & Wirtschaftsinformatik gibt es ein mehrstufiges Aufnahmeverfahren. Die Schritte hängen von der Staatsangehörigkeit und dem Ausstellungsland des Abschlusses ab. Für Masterstudiengänge hängt es davon ab, ob ein Abschluss der Universität Wien oder einer anderen Universität vorliegt.
6. Peter Kalchgruber unterrichtet Modellierung, Softwarepraktikum mit Bachelorarbeit, Information Management & Systems Engineering, Multimedia and Semantic Technologies, Praktikum Informatik 1 & 2, and Digital Media Production.
7. The text does not specify the study fee for students from non-EU countries. It states the fee for an *extraordinary* study is 363.36 Euro plus 25.20 Euro for the ÖH-Beitrag, totaling 388.56 Euro per semester, regardless of nationality. However, this is for an extraordinary study, not necessarily for all non-EU students.
8. Das Dokument erwähnt keine spezifischen Kurse, die im dritten Semester empfohlen werden. Es werden jedoch Pflichtmodule in der Bioinformatik im zweiten und dritten Semester erwähnt, ergänzt durch Lehrveranstaltungen aus mathematisch-naturwissenschaftlichen Disziplinen.
9. STEOP steht für Studieneingangs- und Orientierungsphase. Sie besteht aus den Modulen Programmierung 1, Technische Grundlagen der Informatik und Mathematische Grundlagen der Informatik 1 (insgesamt 18 ECTS). Sie ist wichtig, da man sie erfolgreich abschließen muss, um uneingeschränkten Zugang zu allen Lehrveranstaltungen zu erhalten.
10. Wilfried Gansterer ist der Dekan der Fakultät für Informatik. Man kann ihn/sie unter http://informatik.univie.ac.at erreichen.

A.1.3. Haupt-Model Gemma3:27b

1. E-Mail: peter.reichl@univie.ac.at Telefon: +43-1-4277-79410
2. Öffnungszeiten: Mo-Do: 9:00-14:00, Fr: 9:00-13:00 Telefon: +43-1-4277-781 01 E-Mail: ssc.informatik(at)univie.ac.at Adresse: Währinger Straße 29, 1090 Wien
3. Für Informatik-Studierende sind SE1, NUM, EST und MM erfolgreich abgeschlossen sein. Für Wirtschaftsinformatik-Studierende sind StEOP, EIS, DSE, IDS, NET, VWL, EST, HCI, SE1 und MM erforderlich.

4. 6 ECTS.

5. Für die Bachelorstudien Informatik und Wirtschaftsinformatik gibt es ein mehrstufiges Aufnahmeverfahren. Es hängt von Staatsangehörigkeit, Ausstellungsland des Reifezeugnisses und Deutschkenntnissen ab.

6. Professor Peter Kalchgruber unterrichtet Modellierung, Information Management & Systems Engineering, Multimedia and Semantic Technologies, Digital Media Production und Softwarepraktikum mit Bachelorarbeit.

7. Für Studierende aus Nicht-EU/EWR-Ländern beträgt der Studienbeitrag 726,72 Euro pro Semester, zuzüglich 25,20 Euro ÖH-Beitrag, also insgesamt 751,92 Euro.

8. Das Dokument erwähnt keine spezifischen Kurse, die im dritten Semester empfohlen werden, sondern dass Studierende im zweiten und dritten Semester durch Pflichtmodule in die Bioinformatik eingeführt werden und diese durch weitere Lehrveranstaltungen ergänzen können. Es werden Beispiele für wählbare Ausprägungsfächer genannt, wie Numerical High Performance Algorithms, Gaming Technologies, Image Processing & Image Analysis, etc.

9. STEOP ist die Studieneingangs- und Orientierungsphase, bestehend aus den Modulen Programmierung 1, Technische Grundlagen der Informatik und Mathematische Grundlagen der Informatik 1 (insgesamt 18 ECTS). Sie ist wichtig, da man sie erfolgreich abschließen muss, um uneingeschränkten Zugang zu allen Lehrveranstaltungen zu erhalten.

10. Wilfried Gansterer ist der Dekan der Fakultät für Informatik. Sie können ihn/sie unter dekanat.informatik@univie.ac.at oder +43-1-4277-78001 erreichen.

A.1.4. Haupt-Model Llama3.1:8b

1. Die Kontaktdaten von Professor Peter Reichl sind: * Adresse: Sensengasse 6, 1090 Wien * Telefon: +43-1-4277-79410 * E-Mail: peter.reichl@univie.ac.at

2. Die Öffnungszeiten des SSC der Fakultät Informatik sind: Mo - Do: 9:00 - 14:00 Fr: 9:00 - 13:00 Der Kontakt ist über Telefon (+43-1-4277-78101) oder E-Mail (ssc.informatik@univie.ac.at) möglich.

3. Für das Softwarepraktikum mit Bachelorarbeit sind folgende Module erfolgreich abgeschlossen sein müssen: * Im Studiengang Informatik Allgemein: SE1, NUM, EST und MM * Im Studiengang Wirtschaftsinformatik: StEOP, EIS, DSE und IDS

4. Das Fach PR2, Programmierung 2, hat 6 ECTS-Punkte.

5. An der Fakultät für Informatik wird ein mehrstufiges Aufnahmeverfahren durchgeführt, das von der Staatsangehörigkeit, dem Ausstellungsland des Reifezeugnisses und dem Niveau der Deutschkenntnisse abhängt.

6. Professor Peter Kalchgruber lehrt folgende Kurse: * Modellierung (Modelling) * Softwarepraktikum mit Bachelorarbeit (Practical Software Course with Bachelor's Thesis) * Information Management & Systems Engineering * Multimedia and Semantic Technologies * Praktikum Informatik 1 und 2 (Practical Courses: Computer Science 1 und 2) * Multimedia Retrieval and Content-Based Search * Digital Media Production

7. Die Höhe des Studienbeitrags für Nicht-EU-Studierende an der Universität Wien beträgt 726,72 Euro pro Semester.

8. Im dritten Semester des Bachelor-Studiengangs Wirtschaftsinformatik werden die Pflichtmodule "Algorithmische Bioinformatik, Sequenz- und Vorhersagealgorithmen" empfohlen.

9. Die Studieneingangs- und Orientierungsphase (STEOP) ist eine Pflichtphase im ersten Semester, die aus drei Modulen besteht: PR1 - Programmierung 1, TGI - Technische Grundlagen der Informatik und MG1 - Mathematische Grundlagen der Informatik 1. Diese Module sollen einen Einblick in das Studium geben und sicherstellen, dass Studenten die Anforderungen für das Studium erfüllen können. Es ist wichtig, die STEOP erfolgreich abzuschließen, um uneingeschränkten Zugang zu weiteren Lehrveranstaltungen aller Semester zu bekommen.

10. Der Dekan der Fakultät für Informatik ist Werner Schröttner. Sie können ihn unter folgenden Kontaktdaten erreichen: * Adresse: Währinger Straße 29, 1090 Wien * Telefon: +43-1-4277-78011 * E-Mail: werner.schroettner@univie.ac.at Alternativ kann man auch den stellvertretenden Dekan, Christina Licayan, kontaktieren: * Adresse: Währinger Straße 29, 1090 Wien * Telefon: +43-1-4277-78002 * E-Mail: christina.licayan@univie.ac.at

A.1.5. Haupt-Model Gemma3:1b

1. Peter Reichl Univ.-Prof. Dipl.-Math. Dr. M https://ucrisportal.univie.ac.at/en/persons/342bc49c-6fd6-476c-ac4a-e91b27532317 You can find his contact information on the university website: peter.reichl@univie.ac.at

2. The SSC (StudienServiceCenter) of the Faculty of Computer Science at the University of Vienna is open: * **Monday to Thursday:** 9:00 AM - 14:00 PM * **Friday:** 9:00 AM - 13:00 PM * **Saturday:** 14:00 - 17:00 * **Sunday:** Closed You can reach them via email at: informatik.spl@univie.ac.at The SSC also has a Sprechstunden (appointments) available via email, and you can schedule appointments by email.

3. The Bachelorarbeit – Teilnahmevoraussetzungen im Bachelorstudium Informatik ist erforderlich.

4. 6 ECTS

5. There are multiple Aufnahmeverfahren at the Faculty of Computer Science (DoCS). The context states that "Alle Fristen & Details zum Aufnahmeverfahren" are listed.

6. Peter Kalchgruber teaches courses related to Computer Science, specifically Practical Course: Computer Science 1, Practical Course: Computer Science 2, and the LP Praktikum Informatik 1 Practical Course: Computer Science 1, Practical Course: Computer Science 2, and the VU Programmierung 1 Programming 1.

7. The tuition fee for non-EU students at the University of Vienna is 388,56 Euro pro semester.

8. The context doesn't explicitly state which courses are recommended in the third semester of the Bachelor in Business Informatics. It only mentions that the third semester will involve the Bioinformatik and the completion of the mandatory modules in the Bioinformatik.

9. The STEOP (Studieneingangs- und Orientierungsphase) is a three-module learning phase that consists of PR1, TGI, and MG1. It's crucial because it's the first step in completing the STEOP, which is essential for gaining access to further courses in all semesters. Successfully completing all three modules is required to be able to attend more courses. If not all three modules are completed, students can only attend additional courses from the first semester.

10. Christina Licayan is the dean of the Faculty of Computer Science. You can contact her at christina.licayan@univie.ac.at.

A.2. Fragen

Table A.1.: Evaluationsfragen für den Chatbot (Deutsch und Englisch)

Nr.	Frage	Sprache	Typ	Schwierigkeit
1	Wie lauten die Kontaktdaten von Professor Peter Reichl?	Deutsch	Personenbezogen	Einfach
2	Wie lauten die Öffnungszeiten und Kontaktdaten des SSC der Fakultät Informatik?	Deutsch	Administrativ	Mittel
3	Welche Teilnahmevoraussetzungen gibt es für das Softwarepraktikum im Rahmen der Bachelorarbeit?	Deutsch	Studienrechtlich	Hoch
4	Wie viele ECTS-Punkte hat das Fach PR2 (Programmieren 2)?	Deutsch	Curriculum	Einfach
5	Welche Aufnahmeverfahren gibt es generell an der Fakultät für Informatik?	Deutsch	Zulassung	Mittel
6	What courses does Professor Peter Kalchgruber teach?	Englisch	Personenbezogen	Einfach
7	How much is the tuition fee for non-EU students at the University of Vienna?	Englisch	Finanzen	Mittel
8	What courses are recommended in the third semester of the Bachelor in Business Informatics?	Englisch	Curriculum	Hoch
9	What is STEOP and why is it important?	Englisch	Studienbegriff	Mittel
10	Who is the dean of the Faculty of Computer Science and how can I contact them?	Englisch	Personenbezogen	Mittel