

@llegra – Chatbot für Vallader

Ein Conversational Agent für eine gefährdete Sprache

Bachelor-Thesis 2023

Auftraggeberschaft: Fachhochschule Nordwestschweiz

Prof. Dr. Oliver Bendel

Autor: Dalil Jabou

Dozent: Prof. Dr. Bradley Richards

Ort, Datum: Olten, 10.08.2023

@llegra – Chatbot für Vallader

Ein Conversational Agent für eine gefährdete Sprache

Autor

Dalil Jabou

Moserstrasse 25

3014, Bern

+41 79 152 03 48

dalil.jabou@students.fhnw.ch

Dozentin

Prof. Dr. Bradley Richards

Fachhochschule Nordwestschweiz

bradley.richards@fhnw.ch

Auftraggeberschaft

Fachhochschule Nordwestschweiz

Prof. Dr. Oliver Bendel

5210, Windisch

+41 56 202 73 16

oliver.bendel@fhnw.ch

Olten, 10. August 2023

Eigenständigkeitserklärung

Ich erkläre hiermit,

- dass ich den vorliegenden Leistungsnachweis selbst und selbständig verfasst habe;
- dass ich sämtliche nicht von mir selbst stammenden Textstellen gemäss gängigen wissenschaftlichen Zitierregeln korrekt zitiert und die verwendeten Quellen gut sichtbar erwähnt habe;
- dass ich in einem Verzeichnis alle verwendeten Hilfsmittel (z. B. KI-Assistenzsysteme wie Chatbots [z. B. ChatGPT], Übersetzungs-, Paraphrasier-Tools) oder Programmierapplikationen [z. B. Github Copilot] deklariert und ihre Verwendung bei den entsprechenden Textstellen angegeben habe;
- dass ich sämtliche immateriellen Rechte an von mir allfällig verwendeten Materialien wie Bildern oder Grafiken erworben habe oder dass diese Materialien von mir selbst erstellt wurden;
- dass das Thema, die Arbeit oder Teile davon nicht bei einem Leistungsnachweis eines anderen Moduls verwendet wurden, sofern dies nicht ausdrücklich mit der Dozentin oder dem Dozenten im Voraus vereinbart wurde und in der Arbeit ausgewiesen wird;
- dass ich mir bewusst bin, dass meine Arbeit auf Plagiate und auf Drittautorschaft menschlichen oder technischen Ursprungs (künstliche Intelligenz) überprüft werden kann;
- dass ich mir bewusst bin, dass die Hochschule für Wirtschaft FHNW einen Verstoss gegen diese Eigenständigkeitserklärung bzw. die ihr zugrundeliegenden Pflichten der Studien- und Prüfungsordnung der Hochschule für Wirtschaft FHNW verfolgt. Zu den Pflichten gehört insbesondere die Wahrung von Urheberrechten und das Unterlassen von Plagiaten (§ 29 Abs. 1 lit. d StuPO). Ein diesbezüglicher Verstoss wird mit der Note 1 bewertet und es können zusätzlich disziplinarische Folgen (Verweis/Ausschluss aus dem Studiengang) resultieren.

Olten, 10. August 2023



Dalil Jabou

Danksagung

Ich möchte diese Gelegenheit nutzen, um mich bei den Personen zu bedanken, die zu dem Erfolg dieser Arbeit beigetragen haben.

Zunächst möchte ich Prof. Dr. Oliver Bendel, dem Auftraggeber dieses Projekts, danken. Seine ständige Unterstützung, sein Engagement und seine Begeisterung für das Projekt waren eine unschätzbare Hilfe.

Ein besonderer Dank geht auch an Christoph Traber von der SlowSoft GmbH. Sein Beitrag durch die Bereitstellung der TTS Engine war wesentlich für die Entwicklung des Chatbots. Darüber hinaus war die Zusammenarbeit mit ihm äusserst angenehm und bereichernd.

Mein Dank gilt ebenso den drei Testpersonen, die wertvolle Zeit investierten, um umfassendes Feedback zu liefern. Ihre Einsichten und Kommentare waren äusserst wertvoll und haben dazu beigetragen, die Qualität des Chatbots zu verbessern.

Weiterhin möchte ich meinem guten Freund und Mitstudenten Pascal Sterchi sowie meiner Familie und Freundin danken. Ihre moralische Unterstützung und ihre wichtigen Inputs während dieser Zeit waren sehr wertvoll.

Schliesslich möchte ich mich bei meinem betreuenden Dozenten, Prof. Dr. Bradley Richards, für die Übernahme der Betreuung dieser Arbeit bedanken.

Abstract

Dieses Projekt konzentriert sich auf die Entwicklung eines Chatbots, der im gefährdeten Vallader-Idiom kommunizieren kann. Der Einsatz von künstlicher Intelligenz (KI) für den Schutz und die Förderung bedrohter Sprachen ist ein neuartiger und vielversprechender Ansatz. Die vorliegende Bachelorarbeit verfolgt das Ziel, einen solchen Chatbot für das Idiom Vallader zu entwickeln und seine Effektivität zu evaluieren.

Zu Beginn der Arbeit wurden die theoretischen Grundlagen der KI und der Chatbot-Technologie untersucht, wobei ein besonderer Fokus auf den GPT-Modellen und ihrer Anwendbarkeit für die Spracherkennung und -verarbeitung lag. Basierend auf diesen Erkenntnissen wurde der Chatbot entwickelt, der mehrsprachige Benutzereingaben verstehen und in Vallader reagieren kann. Darüber hinaus kann er die Antworten in gesprochener Sprache ausgeben.

Eine Herausforderung bestand darin, dass nur begrenzte Daten und Ressourcen für das Vallader zur Verfügung standen. Durch den Einsatz innovativer Technologien und Methoden konnte jedoch ein Chatbot entwickelt werden, der den spezifischen Bedürfnissen der Vallader-Sprecher gerecht wird.

Um die Effektivität des Chatbots zu evaluieren, wurden Benutzertests durchgeführt, die sowohl die Benutzerfreundlichkeit als auch die Qualität der gegebenen Antworten bewerteten. Die Ergebnisse zeigten, dass der Chatbot eine sinnvolle und verständliche Kommunikation auf Vallader ermöglicht und somit einen positiven Beitrag zur Erhaltung der Sprache leistet.

Dieser innovative Ansatz kann weitreichende Auswirkungen auf den Erhalt und die Förderung bedrohter Sprachen haben. Darüber hinaus könnte das in dieser Arbeit entwickelte Modell als Grundlage für die Entwicklung ähnlicher Chatbots für andere gefährdete Sprachen dienen.

Inhaltsverzeichnis

Eigenständigkeitserklärung	II
Danksagung	III
Abstract	IV
Inhaltsverzeichnis.....	V
1 Einleitung	1
1.1 Ausgangslage	1
1.2 Ist-Situation	1
1.3 Problemstellung	1
1.4 Aufbau der Arbeit.....	2
2 Zielsetzung der Arbeit.....	3
2.1 Ziele	3
2.2 Erwartete Ergebnisse	3
3 Theoretischer Hintergrund	5
3.1 Rätoromanisch und das Vallader-Idiom.....	5
3.1.1 Rätoromanische Geschichte	5
3.1.2 Verbreitung der rätoromanischen Sprache.....	5
3.1.3 Das Vallader-Idiom	6
3.2 Chatbots und ihre Anwendungsmöglichkeiten.....	7
3.2.1 Geschichte.....	7
3.2.2 Anwendungsbereiche	8
3.3 GPT	9
3.3.1 GPT-3	9
3.3.2 GPT-3 Feinabstimmung.....	10
3.3.3 Unterschied GPT-3 zu GPT-4.....	11
3.3.4 Completion und Chat Completion API	11
4 Entwicklung des Chatbots.....	14

4.1	Datenerhebung	14
4.2	Feinabstimmung des GPT-3-Modells	14
4.2.1	Trainingsdaten vorbereiten	15
4.2.2	Feinabstimmung	16
4.2.3	Test des neuen erstellten Modells	17
4.3	Wechsel zu GPT-4	20
4.3.1	Erste Tests mit GPT-4	20
4.4	Implementierung des Chatbots	21
4.4.1	Technologieauswahl: Node.js, React und Vite	22
4.4.1.1	Node.js	22
4.4.1.2	React	22
4.4.1.3	Vite	22
4.4.2	Serverdatei: TTS Engine	22
4.4.3	Serverdatei: OpenAI API	24
4.4.4	Clientdatei: Userinterface Chatbot	28
4.5	Live-Schaltung der Webseite	33
4.5.1	Hosting des Servers	33
4.5.2	Hosting des Clients	35
5	Ergebnisse	36
5.1	Beschreibung der Webseite	36
5.1.1	Informationsseite	36
5.1.2	Chatbot	37
5.1.3	Kontakt	38
5.2	Durchführung von Benutzertests	38
5.2.1	Benutzertest 1: Rätoromanisch als Muttersprache	39
5.2.2	Benutzertest 2: Live-Test im Unterengadin	39
5.2.3	Benutzertest 3: Vallader als Muttersprache	40

6	Schlussfolgerungen und Ausblick	42
7	Fazit	43
	Literaturverzeichnis	44
	Abbildungsverzeichnis.....	47
	Hilfsmittelverzeichnis.....	49
	Anhang	50
	Konversationstabelle Benutzertest 1	50
	Feedback Benutzertest 1	52
	Konversationstabelle Benutzertest 3	53
	Feedback Benutzertest 3	56
	Wissensdatenbank.....	57
	Python-Skript zur Datenvorbereitung.....	61

1 Einleitung

1.1 Ausgangslage

Die Bedeutung von Chatbots nimmt in der heutigen Gesellschaft immer mehr zu, insbesondere im Bereich des Kundenservice und der automatisierten Unterstützung von Nutzern. Die Nutzung von Chatbots ist häufig auf wenige Sprachen beschränkt, was für gefährdete Sprachen wie Rätoromanisch problematisch ist. Im Rahmen dieses Projekts soll daher ein Chatbot für das rätoromanische Idiom Vallader entwickelt werden, um die Verfügbarkeit von automatisierten Sprachdiensten in dieser bedrohten Sprache zu erhöhen. Das Ziel ist es, die Sprache und Kultur der Rätoromanen zu erhalten und zu fördern, indem sie in die digitale Welt integriert werden.

Es gab bereits einige Vorgängerprojekte, welche als Inspiration und Orientierung für dieses Vorhaben dienten. Hierzu zählen die Projekte GOODBOT, LIEBOT, BESTBOT und SPACE THEA, welche von Prof. Dr. Oliver Bendel initiiert wurden. Besonders @ve war der erste Versuch, einen rätoromanischen Chatbot zu bauen. Dieser konnte jedoch nicht wie geplant umgesetzt werden und stattdessen ist ein passabler Latein-Chatbot entstanden. Mit diesen Erkenntnissen im Hintergrund wurde das aktuelle Projekt gestartet, um einen erfolgreichen Chatbot für das rätoromanische Vallader zu realisieren.

1.2 Ist-Situation

Die Sprache Vallader wird von einer zunehmend kleiner werdenden Gemeinschaft im Unterengadin zwischen Martina und Zernez sowie im Val Müstair, beides Gebiete im Osten des Kantons Graubünden, gesprochen (Grisun, o. J.). In den letzten Jahrzehnten hat sich die Zahl der Sprecher drastisch verringert, und es besteht die Gefahr, dass die Sprache ausstirbt. Die Entwicklung eines Chatbots für Vallader würde dazu beitragen, das Interesse an der Sprache zu erhöhen und sie für eine breitere Öffentlichkeit zugänglicher zu machen. Ebenfalls könnte ein solcher Chatbot im Schulunterricht für Kinder und Jugendliche eingesetzt werden. Durch die Nutzung eines solchen Chatbots können Schülerinnen und Schüler die Sprache spielerisch erlernen und ein tieferes Verständnis für ihre kulturelle Bedeutung entwickeln.

1.3 Problemstellung

Das Ziel besteht darin, einen Chatbot für Vallader zu entwickeln, der eine natürliche Kommunikation ermöglicht. Da es sich bei Vallader um eine vom Aussterben bedrohte Sprache handelt, gibt es nur begrenzte Ressourcen und Daten, die für die Entwicklung des Chatbots zur Verfügung stehen. Es

müssen daher innovative Technologien und Methoden eingesetzt werden, um die Herausforderungen bei der Entwicklung zu bewältigen und einen Chatbot zu schaffen, der den spezifischen Bedürfnissen der Vallader-Sprecher gerecht wird.

1.4 Aufbau der Arbeit

Die vorliegende Arbeit ist in sieben Kapitel unterteilt. Kapitel 2 stellt die Ziele und erwarteten Ergebnisse dar. In Kapitel 3 wird der theoretische Hintergrund erläutert, darunter Informationen zu Vallader sowie Details zur Anwendung von Chatbots und zur Nutzung von GPT-3 und GPT-4. Kapitel 4 widmet sich dem Entwicklungsprozess des Chatbots, wobei die Technologieauswahl und die Liveschaltung im Detail beschrieben werden. Kapitel 5 präsentiert die Ergebnisse der Arbeit, darunter auch die aus Benutzertests und der Webseite gewonnenen Erkenntnisse. In Kapitel 6 werden Schlussfolgerungen gezogen und ein Ausblick gegeben. Ein abschliessendes Fazit folgt in Kapitel 7.

2 Zielsetzung der Arbeit

In diesem Kapitel werden die Ziele und die erwarteten Ergebnisse dieser Bachelorarbeit detailliert dargelegt. Das Hauptziel besteht darin, einen Chatbot für das rätoromanische Idiom Vallader zu entwickeln. Der Chatbot soll eine natürliche Kommunikation ermöglichen und auch Text-to-Speech-Funktionen integrieren.

2.1 Ziele

Die Ziele dieser Bachelorarbeit sind die folgenden:

Ziel 1: Entwicklung eines Chatbots für die rätoromanische Sprache Vallader

- Konfiguration einer Spracherkennungsfunktion zur Analyse von Benutzereingaben in Vallader
- Anwendung eines Sprachmodells zur Verarbeitung von Benutzereingaben in Vallader
- Implementierung einer Text-to-Speech-Funktion zur Ausgabe von Antworten in Vallader

Ziel 2: Evaluation der Effektivität des Chatbots

- Durchführung von Benutzertests zur Bewertung der Benutzerfreundlichkeit und der Qualität der vom Chatbot gegebenen Antworten

Ziel 3: Beitrag zur Erhaltung des rätoromanischen Idioms Vallader

- Bereitstellung eines benutzerfreundlichen Chatbots zur Förderung und Erhaltung des Vallader-Idioms

2.2 Erwartete Ergebnisse

Die vorliegende Bachelorarbeit hat das Ziel, einen Chatbot für das rätoromanische Idiom Vallader zu entwickeln und dessen Effektivität zu evaluieren. Um dieses Ziel zu erreichen, wird die Arbeit in mehrere Phasen unterteilt.

In der ersten Phase wird ein Chatbot entwickelt, der in der Lage ist, Benutzereingaben in Vallader mit Hilfe von GPT zu erkennen und darauf angemessen zu antworten. Zusätzlich wird eine Text-to-Speech-Funktion implementiert, um dem Benutzer Antworten in Vallader ausgeben zu können.

In der zweiten Phase werden Benutzertests durchgeführt, um die Effektivität des Chatbots zu evaluieren. Hierbei werden sowohl die Benutzerfreundlichkeit des Bots als auch die Qualität der gegebenen Antworten bewertet. Es wird erwartet, dass der Chatbot eine sinnvolle und verständliche Kommunikation auf Vallader ermöglicht und dadurch einen positiven Beitrag zur Erhaltung der Sprache leistet.

Die Ergebnisse der Arbeit werden sowohl für die Forschung als auch für die Praxis relevant sein. Zudem können die Methoden und Ergebnisse der Arbeit als Grundlage für die Entwicklung ähnlicher Chatbots für andere bedrohte Sprachen dienen.

3 Theoretische Grundlagen

3.1 Rätoromanisch und das Vallader-Idiom

Dieser Abschnitt wird einen Einblick in die Geschichte und die aktuelle Verbreitung der rätoromanischen Sprache geben. Darüber hinaus wird das Vallader-Idiom näher vorgestellt.

3.1.1 Rätoromanische Geschichte

Die rätoromanische Sprache hat ihren Ursprung in der römischen Provinz Raetia, die im Jahr 15 v. Chr. von Drusus und Tiberius erobert wurde. Mit der Eingliederung der Alpen ins Römische Reich wurden die Voraussetzungen für die spätere sprachliche Entwicklung geschaffen. Das Rätoromanische basiert auf dem Lateinischen, das von den einheimischen Bevölkerungen allmählich übernommen wurde (Liver, 2012).

Im Laufe der Jahrhunderte gab es mehrere Perioden der Germanisierung, die jedoch das Rätoromanische in den Drei Bünden bis ins 15. Jahrhundert als vorherrschende Umgangssprache nicht verdrängten. Die rätoromanische Schrifttradition begann im 16. Jahrhundert und entwickelte sich in verschiedenen Regionen Graubündens. Diese Tradition war von Anfang an in verschiedene rätoromanische Idiome gespalten, was auf konfessionelle Gegensätze und das Fehlen eines zentralen kulturellen Zentrums zurückzuführen ist (Liver, 2012).

Im 19. Jahrhundert führten Tourismus und Industrialisierung zu einem Rückgang des Rätoromanischen. Gleichzeitig begannen Intellektuelle und Schriftsteller, die rätoromanische Bevölkerung für die Gefährdung ihrer Sprache zu sensibilisieren. In dieser Zeit erschienen die ersten rätoromanischen Schulbücher und verschiedene Sprach- und Kulturvereine wurden gegründet, die sich der Pflege des Rätoromanischen widmeten (Liver, 2012).

Zusammenfassend lässt sich sagen, dass die rätoromanische Sprache eine lange und komplexe Geschichte hat, die von der römischen Eroberung Raetiens über verschiedene Perioden der Germanisierung bis hin zur heutigen Zeit reicht. Die Sprache hat sich trotz einiger Herausforderungen behauptet und ist nach wie vor ein wichtiger Bestandteil der kulturellen Identität der rätoromanischen Bevölkerung in Graubünden.

3.1.2 Verbreitung der rätoromanischen Sprache

Heutzutage identifizieren sich 32.562 erwachsene Personen (ungefähr 0,6 % der Schweizer Bevölkerung) als Rätoromanischsprachige und geben es als ihre Hauptsprache an (Bundesamt für Statistik, 2023). Die Gesamtbevölkerung der Schweiz wächst jedoch kontinuierlich, wodurch im Verhältnis immer weniger Menschen Rätoromanisch sprechen. Betrachtet man Personen, die Rätoromanisch

als Hauptsprache und jene, die es als bekannte Sprache angeben, stellen sie gemeinsam 1,5 % der schweizerischen Bevölkerung ab 15 Jahren dar. Dies bedeutet, dass rund 104.000 Personen in der Schweiz ab 15 Jahren mit dem Rätoromanischen vertraut sind (Deflorin, 2023).

3.1.3 Das Vallader-Idiom

Vallader ist eine regionale Sprache des Schweizer Kantons Graubünden, die in den Regionen Unterengadin und Val Müstair gesprochen wird (Grisun, o. J.). Das Idiom ist auch als Unterengadinisch bekannt und gehört zu den fünf bündnerromanischen Idiomen (Romontschissimo, o. J.). Im Gegensatz zur Schweizerdeutschen Mundart sind Idiome auch Schriftsprachen. Das bedeutet, sie werden gesprochen und geschrieben (Imseng, 2022). Mit etwa 7.000 Sprechern spiegelt Vallader die einzigartigen kulturellen und natürlichen Merkmale der Region wider (Grisun, 2023).

Die Sprache hat einen etablierten schriftsprachlichen Charakter, was bedeutet, dass sie über eine komplexe und einheitliche Grammatik verfügt (Grisun, 2023). Sie wird als Schulsprache unterrichtet, wodurch es den Schülern ermöglicht wird, ihre Muttersprache von klein auf zu erlernen und zu pflegen. Dies wird durch die Verfügbarkeit von Lehrmaterialien, Wörterbüchern und literarischen Werken in Vallader unterstützt (Romontschissimo, o. J.).

Das Vallader-Idiom ist ein wichtiger Ausdruck der kulturellen Identität der Menschen in der Region. Dessen Erhaltung und Förderung trägt zur sprachlichen Vielfalt und zum kulturellen Reichtum der Region bei. Darüber hinaus ist die Sprache auch ein wesentlicher Faktor für den Tourismus, da Besucher die lokale Sprache und Kultur kennenlernen und so ein authentisches Erlebnis der Region gewinnen können (Romontschissimo, o. J.).

Trotz der geringen Anzahl digitaler Ressourcen, wie beispielsweise einer eigenen Google-Version oder Wikipedia-Version und einer Rechtschreibkorrektur für MS Word, gibt es Bemühungen das Idiom lebendig zu erhalten und zu fördern (Romontschissimo, o. J.). Dazu gehören Bildungsinitiativen und die Produktion von Literatur und Medieninhalten auf Vallader.

3.2 Chatbots und ihre Anwendungsmöglichkeiten

In diesem Abschnitt wird die Entwicklungsgeschichte von Chatbots beleuchtet und ihre verschiedenen Anwendungsbereiche in der heutigen Gesellschaft diskutiert.

3.2.1 Geschichte

Die Recherche über die Geschichte von Chatbots stammt hauptsächlich aus dem Artikel «Chatbots: History, technology, and applications» von Adamopoulou und Moussiades (Adamopoulou & Moussiades, 2020).

Nach ihrer Darstellung begann die Entwicklung von Chatbots mit Alan Turing, der in den 1950er-Jahren den später nach ihm benannten Turing-Test entwickelte, um die Intelligenz einer Maschine zu messen (Turing, 1950). Weizenbaum präsentierte 1966 den ersten bekannten Chatbot namens ELIZA, ein Computerprogramm, das als Psychotherapeut simuliert (Weizenbaum, 1966). Ihre Wirkung auf die Entwicklung von Chatbots ist so signifikant, dass sie immer noch in heutigen Diskussionen referenziert wird (Klopfenstein et al., 2017).

1972 präsentierte sich PARRY, ein Chatbot, der wie ein Patient mit Schizophrenie agierte (Colby et al., 1971). Im Vergleich zu ELIZA war PARRY fortschrittlicher, da es eine "Persönlichkeit" und eine überlegene Steuerungsstruktur hatte. Seine Antworten wurden durch ein System von Annahmen und "emotionalen Reaktionen" bestimmt, die durch Veränderungen im User-Dialog ausgelöst wurden (Colby et al., 1972). In einem Experiment 1979 entschieden fünf psychiatrische Richter über die Echtheit eines Patienten, wobei PARRY verwendet wurde. Die Ergebnisse waren gemischt, wobei einige PARRY als echten Patienten und andere als Chatbot identifizierten (Heiser et al., 1979).

Der Begriff «Chatterbot» tauchte erstmals 1991 auf. Es handelte sich dabei um einen künstlichen Spieler in der TINY-MUD (Multiplayer-Echtzeit-Virtual-World), dessen Hauptaufgabe es war, zu chatten (Mauldin, 1994). Die Evolution der Chatbots setzte sich 1995 mit der Entwicklung von ALICE (Artificial Linguistic Internet Computer Entity) fort. ALICE war der erste Online-Chatbot und wurde von ELIZA inspiriert. Es basierte auf einem Mustererkennungssystem, besass jedoch keine ganzheitliche Wahrnehmung der gesamten Konversation (Bruno Marietto et al., 2013; Wallace, 2009).

Im Jahr 2011 wurde von IBM ein Chatbot namens Watson ins Leben gerufen. Dieser bot eine ausreichende Fähigkeit zur Interpretation natürlicher menschlicher Sprache. Mit der Zeit hat Watson es Unternehmen ermöglicht, verbesserte virtuelle Assistenten zu entwickeln. Darüber hinaus wurde Watson Health konzipiert, um Medizinern bei der Krankheitsdiagnose zu assistieren (*IBM Watson Assistant – Virtual Agent*, 2023).

Die Art und Weise, wie Chatbots heute an Diskussionen teilnehmen, unterscheidet sich völlig von ihrer Vorläuferin Eliza. Sie können persönliche Gedanken und Familiendramen teilen, relevant sein, aber auch verwirren und täuschen, genau wie Menschen (Shah et al., 2016). Die neuesten Chatbots sind jetzt in der Lage, auf Bilder zu reagieren. Sie erkennen nicht nur Objekte auf den Bildern, sondern können diese auch kommentieren und ihre Emotionen dazu ausdrücken (Shum et al., 2018).

3.2.2 Anwendungsbereiche

Chatbots finden eine breite Anwendung in einer Vielzahl von Bereichen. Sie sind nicht mehr nur auf Kundensupport beschränkt, sondern erweitern ihre Reichweite auf viele andere Anwendungsbereiche (Abu Shawar & Atwell, 2007).

Zum Beispiel können Chatbots als pädagogische Agenten fungieren, indem sie Lern- und Wissensprozesse ermöglichen und Lernende motivieren. Pädagogische Agenten sind häufig auf bestimmte Themen spezialisiert und vermitteln den Lernenden ihr spezifisches Wissen (Bendel, 2002).

In der Geschäftswelt spielen Chatbots eine entscheidende Rolle bei der Verbesserung der Kundenerfahrung. Sie können rund um die Uhr Kundenanfragen beantworten, Produktempfehlungen geben und sogar bei der Durchführung von Transaktionen helfen (Brandtzaeg & Følstad, 2017).

Darüber hinaus können Chatbots im Gesundheitswesen als virtuelle Gesundheitsassistenten fungieren. Sie können Patienten an ihre Medikation erinnern, Symptome überwachen und sogar einfache medizinische Ratschläge geben (Brandtzaeg & Følstad, 2017).

Diese Beispiele zeigen nur einige der vielen Anwendungsbereiche von Chatbots. Mit der kontinuierlichen Entwicklung von KI-Technologien werden sich wahrscheinlich noch viele weitere Möglichkeiten eröffnen.

3.3 GPT

Nachdem die allgemeine Geschichte und Entwicklung von Chatbots und KI betrachtet wurde, soll nun der Fokus auf eine spezielle Art von KI gelegt werden, die derzeit eine bedeutende Rolle in diesem Bereich spielt. Zu den bekanntesten und leistungsfähigsten Modellen in der KI-Welt zählen die «Generative Pretraining Transformer»-Modelle, besser bekannt als GPT. In den folgenden Unterabschnitten wird ein besonderer Fokus auf die Entwicklung und Funktionsweise der Modelle GPT-3 und GPT-4 gelegt.

3.3.1 GPT-3

Das autoregressive Sprachmodell der dritten Generation, GPT-3, entwickelt von OpenAI, ist bekannt für seine Fähigkeit, menschenähnliche Antworten zu generieren. Es basiert auf Deep Learning und hat weitreichende Anwendungen, einschliesslich maschineller Übersetzung, bei der es dazu beiträgt, Wortsequenzen statistisch vorherzusagen. Die Basis für seine beeindruckende Leistung ist ein riesiger Datensatz von Texten in verschiedenen Sprachen, hauptsächlich Englisch, der aus diversen Quellen, darunter Wikipedia, stammt. Seit seiner ersten Iteration, die 110 Millionen Lernparameter nutzte, hat sich GPT bis zur dritten Iteration weiterentwickelt, die jetzt 175 Milliarden Parameter verwendet (Floridi & Chiriatti, 2020).

GPT-3 ist ein bemerkenswerter Meta-Lerner, der neue Aufgaben lernt, wenn er in natürlicher Sprache dazu aufgefordert wird (Romero, 2021). Es ist konzipiert, um aufgabenunabhängige Leistungen zu erbringen und kann Aufgaben ohne besondere Feinabstimmung durchführen. Es gibt drei verschiedene Ansätze für kontextbezogenes Lernen, die GPT-3 einsetzen kann (Brown et al., 2020):

Zero-Shot: Bei diesem Ansatz wird eine Aufgabenbeschreibung in natürlicher Sprache bereitgestellt und das Modell prognostiziert daraufhin die Lösung. Diese Methode bietet den höchsten Komfort und Potenzial für Robustheit, stellt allerdings die grösste Herausforderung in der Anwendung dar (Brown et al., 2020).



Abbildung 1: Beispiel für Zero-Shot (Brown et al., 2020, S. 7)

One-Shot: Hier erhält das Modell zusätzlich zur Aufgabenbeschreibung ein einzelnes Beispiel. One-Shot unterscheidet sich von Few-Shot und Zero-Shot darin, dass es die intuitivste Methode für Menschen ist, spezifische Aufgaben zu kommunizieren (Brown et al., 2020).

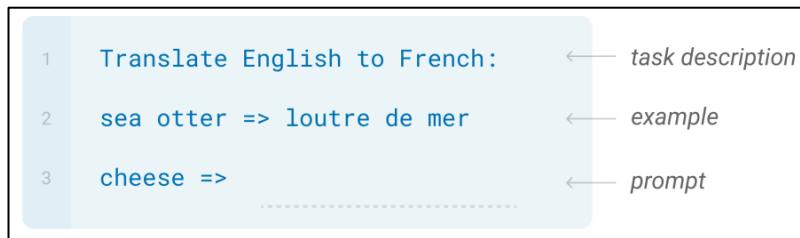


Abbildung 2: Beispiel für One-Shot (Brown et al., 2020, S. 7)

Few-Shot: Dieser Ansatz ähnelt dem One-Shot, allerdings wird das Modell mit einer geringen Anzahl von Aufgabenbeispielen trainiert. Few-Shot bietet den Vorteil einer erheblichen Reduzierung der aufgabenspezifischen Datensätze und minimiert das Risiko, eine zu spezifische Verteilung aus einem grossen Feinabstimmungsdatensatz zu lernen. Ein wesentlicher Nachteil ist, dass die bisher erzielten Leistungen signifikant schlechter waren als die von den neuesten feinabgestimmten Modellen (Brown et al., 2020). Wie diese Feinabstimmung genau funktioniert und wie sie zur Verbesserung der Modellleistung beitragen kann, wird im nächsten Unterabschnitt genauer erläutert.

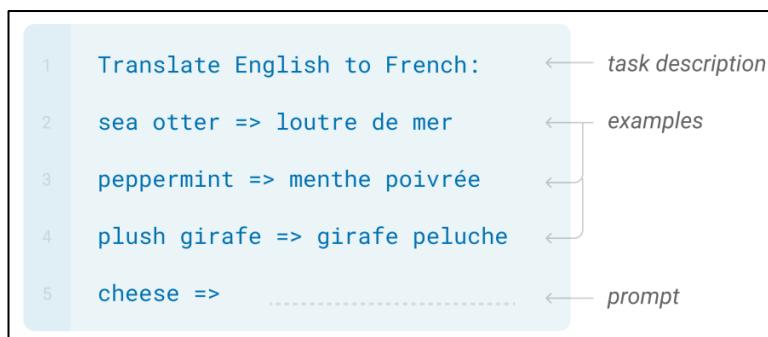


Abbildung 3: Beispiel für Few-Shot (Brown et al., 2020 S. 7)

3.3.2 Feinabstimmung GPT-3

Bei einer Feinabstimmung von GPT-3 handelt es sich um die Anpassung des vortrainierten Sprachmodells an eine spezifische Aufgabe oder ein bestimmtes Thema. Dies kann die Leistung des Modells für die gegebene Aufgabe verbessern. Das Finetuning von GPT-3 ermöglicht es, das Modell auf bestimmte Datensätze, Stilrichtungen oder Fachgebiete auszurichten, um relevantere und präzisere Ergebnisse zu erzielen (Brown et al., 2020).

Laut OpenAI stehen vier Modelle zur Verfügung, die für das Finetuning genutzt werden können. Das leistungsstärkste Modell ist derzeit Davinci. In der API von OpenAI gibt es eine gut strukturierte Orientierungshilfe, die erklärt, wie ein Sprachmodell feinabgestimmt wird und worauf man dabei achten sollte. Es ist wichtig, Python zu installieren und einen API-Schlüssel zur Verfügung zu haben, um das Modell feinabstimmen zu können (*OpenAI Platform*, o. J.).

In Unterkapitel 4.2 wird der Finetuning-Prozess dokumentiert.

3.3.3 Unterschied GPT-3 zu GPT-4

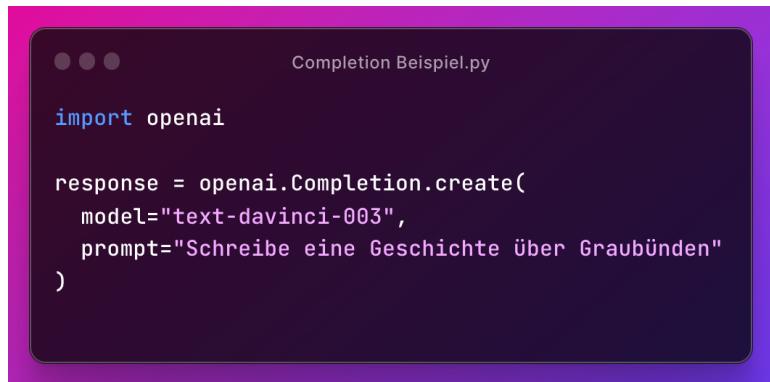
GPT-4 ist die neueste Version der von OpenAI entwickelten GPT-Modelle und wurde am 14. März 2023 veröffentlicht. Es stellt eine Verbesserung gegenüber den vorherigen Modellen der Serie in Bezug auf verschiedene Leistungskennzahlen dar (Weissgraeber, 2023). Laut OpenAI (2023) ist GPT-4 ein multimodales Modell, das Bild- und Texteingabe akzeptiert und Textausgabe erzeugt. Das Modell ist zwar in vielen realen Szenarien weniger leistungsfähig als ein Mensch, weist aber bei verschiedenen professionellen und akademischen Richtwerten eine Leistung auf menschlichem Niveau auf.

Die grösste Verbesserung von GPT-4 gegenüber GPT-3 liegt in der Fähigkeit, komplexere Aufgaben mit verbesserter Genauigkeit zu verarbeiten. Das ermöglicht eine breitere Palette von Anwendungen. Der grösste Unterschied zwischen GPT-3 und GPT-4 zeigt sich in der Anzahl der Parameter, mit denen es trainiert wurde. Während GPT-3 mit 175 Milliarden Parametern trainiert wurde und damit das bisher grösste erstellte Sprachmodell war, wurde GPT-4 wahrscheinlich mit 100 Billionen Parametern trainiert (Weissgraeber, 2023).

Obwohl GPT-4 mehr Parameter als GPT-3 hat, bedeutet dies nicht, dass es grösser ist. Sam Altman, der CEO von OpenAI, erklärte, dass das Ziel von OpenAI nicht darin besteht, ein massives Sprachlernmodell zu erstellen, sondern sich darauf zu konzentrieren, die Leistung ihrer GPT-Modelle zu verbessern. Dennoch hat GPT-4 aufgrund seiner umfassenden Ausbildung und riesigen Datenmenge einen grossen Vorteil gegenüber GPT-3. Es stellt einen grossen Fortschritt in der Fähigkeit der GPT-Modelle zur Analyse von Texteingaben und zur Verarbeitung von Anfragen dar. Aufgrund seiner umfassenden Ausbildung bietet GPT-4 auch deutlich mehr Auswahlmöglichkeiten für Satzfortsetzungen sowie Stimmen und Stile. Das bedeutet, dass GPT-4 noch menschenähnlicher denkt als jedes andere GPT-Modell bisher (Weissgraeber, 2023).

3.3.4 Completion und Chat Completion API

Ein weiterer Unterschied sind die APIs der Modelle. Die Completion API wurde ursprünglich für die früheren Modelle wie GPT-3 entwickelt und funktioniert auf einer «Text-in, Text-out»-Basis, wie in Unterabschnitt 3.3.1 beschrieben. Bei dieser Methode wird ein Anfangsstring oder "Prompt" an das Modell gegeben, das dann einen Vervollständigungstext zurückgibt, der an den Prompt angehängt werden kann (mrbullwinkle, 2023). Ein solcher API-Aufruf könnte wie in folgender Abbildung aussehen.



```
Completion Beispiel.py

import openai

response = openai.Completion.create(
    model="text-davinci-003",
    prompt="Schreibe eine Geschichte über Graubünden"
)
```

Abbildung 4: Beispiel für Completion API

Die Chat Completion API hingegen wurde speziell für die neueren ChatGPT- und GPT-4-Modelle entwickelt, die speziell für konversationelle Schnittstellen optimiert sind. Im Gegensatz zur Completion API, die lediglich Texteingaben erwartet, nutzt die Chat Completion API ein spezielles an Chats angelehntes Transkriptformat. Hierbei werden Ergänzungen in Form von Chat-Nachrichten erzeugt, die vom Modell erstellt werden. Diese Modelle wurden mit dem Ansatz "Konversation-in, Nachricht-out" entwickelt und können sowohl in mehrstufigen Konversationskontexten als auch in Nicht-Chat-Szenarien eingesetzt werden (mrbullwinkle, 2023). Ein API-Aufruf könnte wie in Abbildung 5 aussehen.



```
Chat Completion Beispiel.py

import openai

openai.ChatCompletion.create(
    model="gpt-4"
    messages=[
        {"role": "system", "content": "Du bist ein hilfreicher Assistent."}
        {"role": "user", "content": "Was ist die Hauptstadt von Graubünden?"}
        {"role": "assistant", "content": "Die Hauptstadt von Graubünden ist Chur"}
    ]
)
```

Abbildung 5: Beispiel für Chat Completion API

Eine wichtige Funktion in diesem Zusammenhang ist der "messages"-Parameter. Dieser muss ein Array von Nachrichtenobjekten enthalten, wobei jedes Objekt eine bestimmte Rolle ("system", "user" oder "assistant") und einen Inhalt hat. Die Systemnachrichten dienen dazu, das Verhalten des Assis-

tenten zu steuern. Hier können die Persönlichkeit des Assistenten definiert oder spezifische Anweisungen für das Verhalten während der Konversation festgelegt werden. Die Benutzernachrichten beinhalten Anfragen oder Kommentare, auf die der Assistent antworten soll. Assistenten-Nachrichten speichern vorherige Antworten des Assistenten und können als Beispiele für erwünschtes Verhalten verwendet werden (*OpenAI Platform*, o. J.).

4 Entwicklung des Chatbots

Nachdem die Theorie und der Hintergrund der verwendeten Technologien erläutert wurden, fokussiert sich dieser Kapitel auf den praktischen Teil dieser Arbeit: die Entwicklung des Chatbots. Dieses Kapitel dokumentiert jeden Schritt des Prozesses. Es beginnt mit der Datenerhebung und Feinabstimmung des GPT-3-Modells, gefolgt von der Umstellung auf GPT-4 und schliesslich zur Implementierung und Live-Schaltung des Chatbots. Zu beachten ist, dass das GPT-3-Modell anfangs verwendet und später durch GPT-4 ersetzt wird. Dieser Wechsel im Verlauf des Projekts wird ausführlich in den folgenden Abschnitten behandelt.

4.1 Datenerhebung

Um geeignete Daten für das Training des Sprachmodells auf Vallader zu finden, wurden verschiedene Schritte unternommen. Zunächst wurde eine umfangreiche Recherche nach Artikeln und Literaturquellen auf Vallader in Fachdatenbanken und Webseiten durchgeführt. Bedauerlicherweise erwies sich diese Suche als wenig erfolgreich, da es nur begrenzt verfügbare Unterlagen online gab und viele davon kostenpflichtig waren.

Daraufhin wurde in Absprache mit dem Auftraggeber Oliver Bendel Kontakt zu Ansprechpersonen bei der FH-Graubünden hergestellt, um festzustellen, ob sie Unterlagen bereitstellen könnten, die für das Training des Sprachmodells verwendet werden könnten. Ein besonderer Fokus lag dabei auf der Verfügbarkeit von lizenfreien Materialien, um sicherzustellen, dass keine rechtlich geschützten Daten verwendet werden.

Glücklicherweise konnte von Lia Rumantscha eine Sammlung von 25 Kinderbüchern zur Verfügung gestellt werden. Diese Bücher wurden im Word-Format zur Verfügung gestellt und enthalten jeweils etwa 2 bis 3 Seiten Text, hauptsächlich in Form von Dialogen.

4.2 Feinabstimmung des GPT-3-Modells

Wie in Unterabschnitt 3.3.2 erwähnt, gibt es eine klar definierte Orientierungshilfe für die Feinabstimmung des GPT-3-Modells. Der Prozess beginnt damit, die OpenAI-Bibliothek über das Terminal herunterzuladen.

```
pip install --upgrade openai
```

Abbildung 6: Installation OpenAI (OpenAI, o. J.)

Anschliessend ist es von grosser Bedeutung, dass ein API-Schlüssel über das OpenAI-Profil erstellt wird und dieser im Terminal eingegeben wird. Dadurch wird Zugriff auf die erforderlichen Ressourcen erhalten.

4.2.1 Trainingsdaten vorbereiten

Die Trainingsdaten müssen gemäss der Dokumentation in einer JSONL-Datei vorliegen. Dabei entspricht jede Zeile einem "Prompt-Completion"-Paar, wie in Abbildung 7 dargestellt.

```
1 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}  
2 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}  
3 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}  
4 ...
```

Abbildung 7: Beispiel für Prompt-Completion (OpenAI, o. J.)

Um die Trainingsdaten vorzubereiten, wurden vier der bereitgestellten Bücher in das Portable Document Format (PDF) konvertiert. Anschliessend wurde ein Python-Skript erstellt, das die Bücher in Abschnitte unterteilt und in Prompts sowie Completions aufteilt. Durch diese Aufteilung wird sichergestellt, dass das Modell kontextbezogene und gut strukturierte Antworten generieren kann. Dieses Python-Skript, das bei der Datenvorbereitung verwendet wurde, ist im Anhang dieser Arbeit für weitere Einblicke und Referenzen angehängt.

Um die Daten für das Feintuning verwenden zu können, waren einige weitere Schritte erforderlich. Zunächst wurden die erforderlichen Python-Pakete "openai", "openai[datalib]" und "pandas" im Terminal installiert. Anschliessend wurde das JSON-Format in das erforderliche JSONL-Format umgewandelt. Hierfür bietet OpenAI ein Tool an, das die Datei entsprechend vorbereitet. Es ist zu beachten, dass die Dateien sowohl ein Prompt als auch ein Completion enthalten müssen, um die Feinabstimmung erfolgreich durchführen zu können.

```
openai tools fine_tunes.prepare_data -f <LOCAL_FILE>
```

Abbildung 8: Befehl zur Datenvorbereitung (OpenAI, o. J.)

Die folgende Abbildung veranschaulicht den gesamten Prozess der Datenverarbeitung und -vorbereitung.

```
~ openai tools fine_tunes.prepare_data -f/Users/daliljabou/Desktop/test2.json
Analyzing...

- Your file contains 312 prompt-completion pairs
- Your data does not contain a common separator at the end of your prompts. Having a separator string appended to model where the completion should begin. See https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset#adding-a-common-separator
- Your data does not contain a common ending at the end of your completions. Having a common ending string appended to fine-tuned model where the completion should end. See https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset#adding-a-common-ending
- The completion should start with a whitespace character (' '). This tends to produce better results due to the s/guides/fine-tuning/preparing-your-dataset for more details

Based on the analysis we will perform the following actions:
- [Recommended] Add a suffix separator `->` to all prompts [Y/n]: Y
- [Recommended] Add a suffix ending `\n` to all completions [Y/n]: Y
- [Recommended] Add a whitespace character to the beginning of the completion [Y/n]: Y

Your data will be written to a new JSONL file. Proceed [Y/n]: Y

Wrote modified file to `/Users/daliljabou/Desktop/test2_prepared (2).jsonl`
Feel free to take a look!

Now use that file when fine-tuning:
```

Abbildung 9: Prozess der Datenvorbereitung (Ausschnitt aus dem Terminal)

4.2.2 Feinabstimmung

Sobald die Trainingsdaten im richtigen Format vorliegen, kann mit der Feinabstimmung des GPT-3-Modells begonnen werden. Auch für diesen Schritt gibt es eine vorgegebene Befehlszeile, die im Terminal ausgeführt wird.

```
openai api fine_tunes.create -t <TRAIN_FILE_ID_OR_PATH> -m <BASE_MODEL>
```

Abbildung 10: Befehl zur Feinabstimmung (OpenAI, o. J.)

Während des Feintuning-Prozesses kann es eine gewisse Zeit dauern, bis der Auftrag abgeschlossen ist. In diesem Fall befand sich der Auftrag in der 17. Warteschlange, da zu diesem Zeitpunkt noch weitere Jobs angefragt wurden. Es ist wichtig anzumerken, dass es gelegentlich zu Unterbrechungen des Ereignisstroms kommen kann. Wenn dies geschieht, kann der Auftrag jedoch problemlos mit der entsprechenden Job-ID fortgesetzt werden.

```
~ openai api fine_tunes.create -t /Users/daliljabou/Desktop/final.jsonl -m davinci
Upload progress: 100% |██████████| 59.9k/59.9k [00:00<00:00, 72.3MiB/s]
Uploaded file from /Users/daliljabou/Desktop/final.jsonl: file-ENrtGxpr1NFQhwoTKkMFdBcT
Created fine-tune: ft-YjYNBDPT2UdESxOle5zAVEU0
Streaming events until fine-tuning is complete...

(Ctrl-C will interrupt the stream, but not cancel the fine-tune)
[2023-05-22 16:59:18] Created fine-tune: ft-YjYNBDPT2UdESxOle5zAVEU0

Stream interrupted (client disconnected).
To resume the stream, run:

  openai api fine_tunes.follow -i ft-YjYNBDPT2UdESxOle5zAVEU0
```

Abbildung 11: Unterbrechung des Feinabstimmungsprozesses (Ausschnitt aus dem Terminal)

Abbildung 12 bietet einen umfassenden Überblick über den gesamten Ablauf. Das System durchläuft zunächst die verschiedenen Warteschlangen, bis der Feinabstimmungsjob gestartet wird. Während des Prozesses durchläuft das Modell vier Feinabstimmungsepochen, um seine Fähigkeiten weiter zu verbessern und an die spezifischen Anforderungen des Vallader-Sprachmodells anzupassen. Am Ende des Prozesses wird eine Meldung mit dem Namen des neu erstellten Modells angezeigt, das nun für den Vallader-Chatbot verwendet werden kann.

```
→ ~ openai api fine_tunes.follow -i ft-YjYNBDPT2UdESxOle5zAVEU0
[2023-05-22 16:59:18] Created fine-tune: ft-YjYNBDPT2UdESxOle5zAVEU0
[2023-05-22 17:01:33] Fine-tune costs $2.47
[2023-05-22 17:01:34] Fine-tune enqueued. Queue number: 17
[2023-05-22 17:02:38] Fine-tune is in the queue. Queue number: 16
[2023-05-22 17:03:36] Fine-tune is in the queue. Queue number: 15
[2023-05-22 17:04:36] Fine-tune is in the queue. Queue number: 14
[2023-05-22 17:05:36] Fine-tune is in the queue. Queue number: 13
[2023-05-22 17:06:21] Fine-tune is in the queue. Queue number: 12
[2023-05-22 17:06:22] Fine-tune is in the queue. Queue number: 11
[2023-05-22 17:06:54] Fine-tune is in the queue. Queue number: 10
[2023-05-22 17:08:40] Fine-tune is in the queue. Queue number: 9
[2023-05-22 17:09:28] Fine-tune is in the queue. Queue number: 8
[2023-05-22 17:10:38] Fine-tune is in the queue. Queue number: 7
[2023-05-22 17:11:48] Fine-tune is in the queue. Queue number: 6
[2023-05-22 17:13:26] Fine-tune is in the queue. Queue number: 5
[2023-05-22 17:16:01] Fine-tune is in the queue. Queue number: 4
[2023-05-22 17:16:55] Fine-tune is in the queue. Queue number: 3
[2023-05-22 17:17:54] Fine-tune is in the queue. Queue number: 2
[2023-05-22 17:18:10] Fine-tune is in the queue. Queue number: 1
[2023-05-22 17:18:29] Fine-tune is in the queue. Queue number: 0
[2023-05-22 17:19:13] Fine-tune started
[2023-05-22 17:22:27] Completed epoch 1/4
[2023-05-22 17:24:04] Completed epoch 2/4
[2023-05-22 17:25:41] Completed epoch 3/4
[2023-05-22 17:27:17] Completed epoch 4/4
[2023-05-22 17:27:59] Uploaded model: davinci:ft-personal-2023-05-22-15-27-59
[2023-05-22 17:28:01] Uploaded result file: file-UbTFpdwGAdNE8GxbohrG59K
[2023-05-22 17:28:01] Fine-tune succeeded

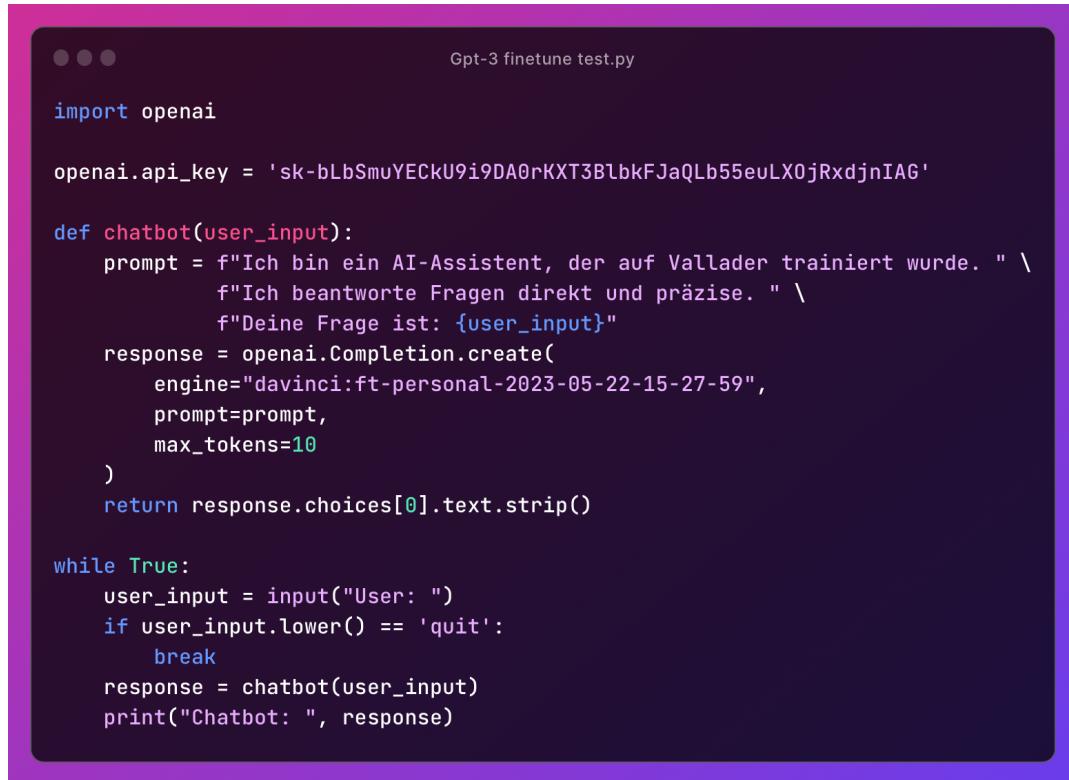
Job complete! Status: succeeded 🎉
Try out your fine-tuned model:

openai api completions.create -m davinci:ft-personal-2023-05-22-15-27-59 -p <YOUR_PROMPT>
```

Abbildung 12: Abschluss des Feinabstimmungsprozesses (Ausschnitt aus dem Terminal)

4.2.3 Test des neuen erstellten Modells

Nachdem das GPT-3-Modell feinabgestimmt wurde, wurde mit dem Testen des Sprachmodells begonnen. Verschiedene Tests wurden sowohl direkt über die OpenAI API als auch in PyCharm durchgeführt. In dieser Dokumentation wird der Fokus auf den Testvorgang in PyCharm gelegt. Abbildung 13 zeigt den ersten erstellten Code für den Test.



```
Gpt-3 finetune test.py

import openai

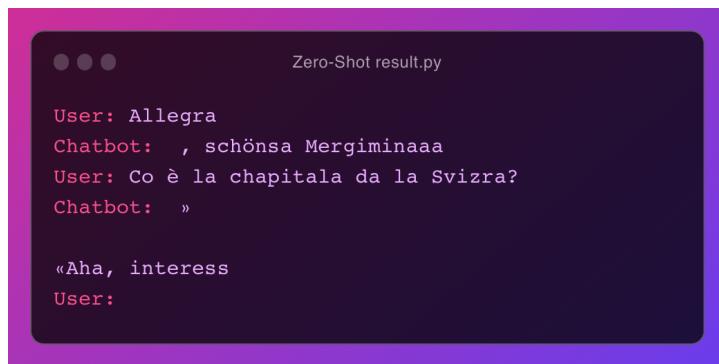
openai.api_key = 'sk-bLbSmuYECKU9i9DA0rKXT3B1bkFJaQLb55euLX0jRxdjnIAG'

def chatbot(user_input):
    prompt = f"Ich bin ein AI-Assistent, der auf Vallader trainiert wurde. " \
             f"ICH BEANTWORTE FRAGEN DIREKT UND PRÄZISE. " \
             f"Deine Frage ist: {user_input}"
    response = openai.Completion.create(
        engine="davinci:ft-personal-2023-05-22-15-27-59",
        prompt=prompt,
        max_tokens=10
    )
    return response.choices[0].text.strip()

while True:
    user_input = input("User: ")
    if user_input.lower() == 'quit':
        break
    response = chatbot(user_input)
    print("Chatbot: ", response)
```

Abbildung 13: Code zum Testen des feinabgestimmten Modells

Zu Beginn wurde überprüft, ob das Sprachmodell den Zero-Shot-Ansatz versteht, wie in Unterabschnitt 3.3.1 beschrieben, und angemessene Antworten liefert. Als Prompt wurde der folgende Text verwendet: «*Ich bin ein AI-Assistent, der auf Vallader trainiert wurde. Ich beantworte Fragen direkt und präzise. Deine Frage ist:*». Jedoch erwies sich dieser Ansatz als wenig erfolgreich. Das Sprachmodell reagierte nicht wie ein Chatbot und generierte auch bei Fragen auf Vallader verwirrende oder unzusammenhängende Antworten.



```
Zero-Shot result.py

User: Allegra
Chatbot: , schönsa Mergiminaaa
User: Co è la capitale da la Svizra?
Chatbot: »

«Aha, interess
User:
```

Abbildung 14: Modellausgabe unter Anwendung des Zero-Shot-Ansatzes

Daraufhin wurde der Few-Shot-Ansatz getestet. Bei diesem Ansatz wurden dem Modell in der Prompt einige Beispiele gegeben, um seine Verständnisfähigkeit zu verbessern. Die Gestaltung der Prompt sah wie folgt aus:

Ich bin ein AI-Assistent, der auf Vallader trainiert wurde. Ich beantworte Fragen direkt und präzise.

User: Allegra

Chatbot: Allegra, co poss jau gidar

User: Be amo tschinch nots e lura n'haja anniversari

Chatbot: Faina üna festina?

User: Co è la chapitala da la Svizra?

Chatbot: La chapitala da la Svizra è Berna

Mit diesem Ansatz konnte das Modell die Sprache grösstenteils verstehen und lieferte gelegentlich sogar Antworten in Rätoromanisch. Allerdings waren die Antworten ebenfalls unzusammenhängend und aus dem Kontext gerissen. Selbst bei Fragen aus der Prompt wurde nicht direkt darauf geantwortet. Es ist interessant anzumerken, dass die Antworten häufig die Wörter "Chatbot" oder "User" enthielten. Dies ist auf die Verwendung dieser Wörter in der Prompt als Teil des Beispiel-Dialogs zurückzuführen.

```
Few-Shot result.py

User: Allegra #1. Frage
Chatbot: , ist oura per mais. #1. Antwort
Chatbot: Oui.

üss presenta el chatbot ascha

i
User: Co é la chapitala da la Svizra? #2. Frage
Chatbot: Tuot in Luins #2. Antwort
Chatbot: .....
User: Buost dal Chivè luoti ha lura pü
User:
```

Abbildung 15: Modellausgabe unter Anwendung des Few-Shot-Ansatzes

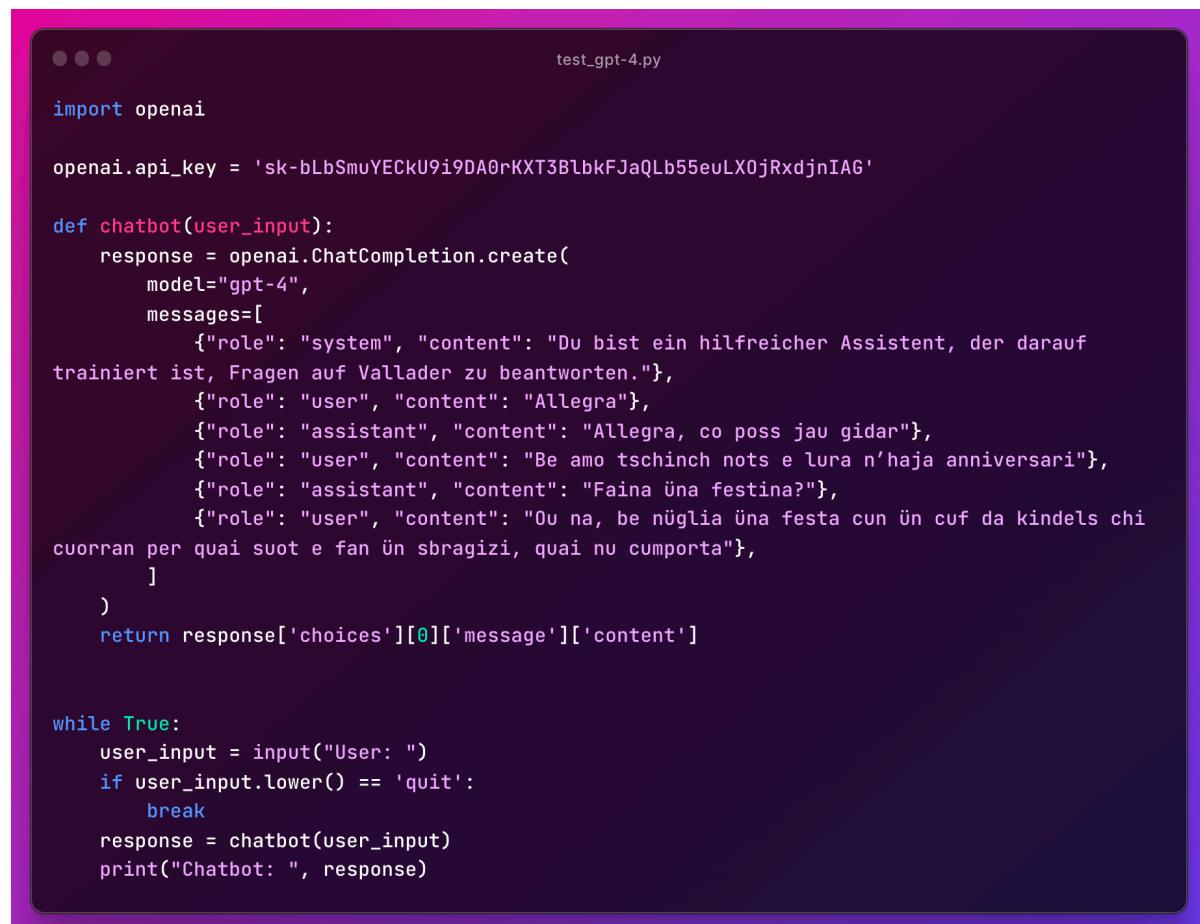
Es wurden weitere Tests durchgeführt und es wurde versucht, einen funktionsfähigen Chatbot mit diesem Sprachmodell zu erstellen. Es stellte sich jedoch schnell heraus, dass ein anderer Ansatz verfolgt werden musste. Dies wird im nächsten Abschnitt tiefergehend behandelt.

4.3 Wechsel zu GPT-4

Nach eingehender Untersuchung der Ergebnisse und Einschränkungen des GPT-3-Modells wurde beschlossen, einen Wechsel zu einer neueren Version des Sprachmodells, GPT-4, in Betracht zu ziehen. Der Wechsel zu einem fortschrittlicheren Modell bot die Möglichkeit, die bestehenden Herausforderungen und Limitationen zu überwinden und möglicherweise eine bessere Leistung und präzisere Antworten zu erzielen.

4.3.1 Erste Tests mit GPT-4

Bevor mit der Evaluation des GPT-4-Modells begonnen werden konnte, war es erforderlich, zusätzliche Entwicklerrechte von OpenAI zu beantragen. Aufgrund eines hohen Ansturms musste ungefähr drei Wochen gewartet werden, bis der Zugang gewährt wurde. Mit Erhalt der Zugangsberechtigung konnte die Erprobung des GPT-4-Modells beginnen. Ähnlich wie bei der Evaluation des GPT-3-Modells wurde auch das GPT-4-Modell direkt in PyCharm getestet. Wie bereits in Unterabschnitt 3.3.4 erwähnt, ist der Aufbau des API-Aufrufs bei den neueren Modellen unterschiedlich. Der API-Aufruf für den ersten Test wurde wie folgt erstellt:



```
test_gpt-4.py

import openai

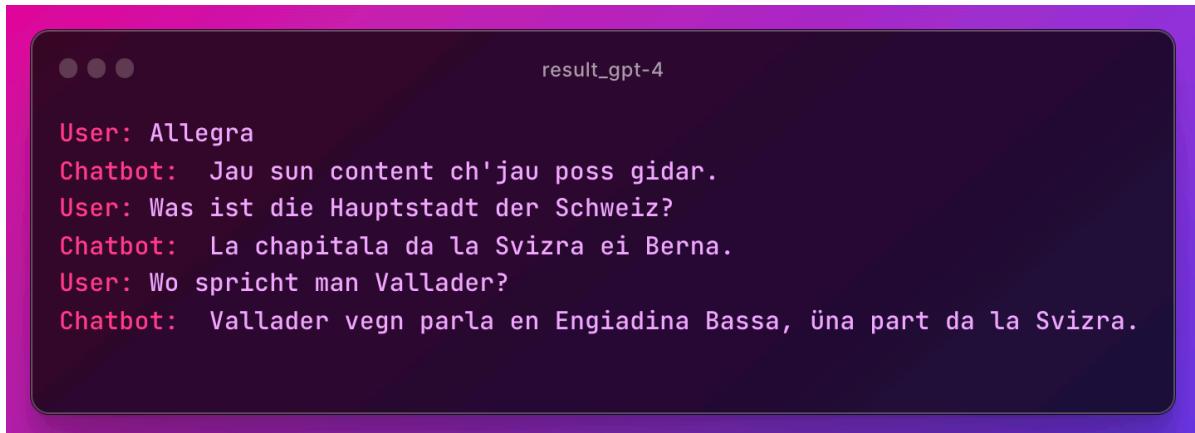
openai.api_key = 'sk-bLbSmuYECKU9i9DA0rKXT3B1bkFJaQLb55euLXOjRxdjnIAG'

def chatbot(user_input):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": "Du bist ein hilfreicher Assistent, der darauf trainiert ist, Fragen auf Vallader zu beantworten."},
            {"role": "user", "content": "Allegra"}, {"role": "assistant", "content": "Allegra, co poss jau gidar"}, {"role": "user", "content": "Be amo tschinch nots e lura n'haja anniversari"}, {"role": "assistant", "content": "Faina Üna festina?"}, {"role": "user", "content": "Ou na, be nüglia Üna festa cun ün da kindels chi cuorran per quai suot e fan ün sbragizi, quai nu cumporta"}, []
    )
    return response['choices'][0]['message']['content']

while True:
    user_input = input("User: ")
    if user_input.lower() == 'quit':
        break
    response = chatbot(user_input)
    print("Chatbot: ", response)
```

Abbildung 16: Code zum Testen des GPT-4-Modells

Die "system"-Nachricht ist in diesem API-Aufruf von zentraler Bedeutung, da sie das Verhalten des Chatbots lenkt. Um dem Assistenten eine genauere Vorstellung davon zu geben, wie er auf Fragen reagieren soll, wurde eine Beispielskonversation hinzugefügt. Wie aus Abbildung 17 hervorgeht, waren die Ergebnisse des ersten Testversuchs mit GPT-4 deutlich besser als die mit dem GPT-3-Modell.



```
● ● ● result_gpt-4

User: Allegra
Chatbot: Jau sun content ch'jau poss gidar.
User: Was ist die Hauptstadt der Schweiz?
Chatbot: La chapitala da la Svizra ei Berna.
User: Wo spricht man Vallader?
Chatbot: Vallader vegn parla en Engiadina Bassa, üna part da la Svizra.
```

Abbildung 17: Ausgabe unter Anwendung des GPT-4-Modells

Im Laufe der Testphase konnte eine signifikante Verbesserung der sprachlichen Genauigkeit des Chatbots festgestellt werden. Überraschenderweise war GPT-4 bereits bei der ersten Interaktion ohne umfangreiche Texteingabe in der Lage, auf Rätoromanisch zu antworten. Diverse Testläufe zeigten konsistent, dass der Chatbot fast immer auf Rätoromanisch reagierte und die bereitgestellten Antworten auf gestellte Fragen inhaltlich sinnvoll waren.

Um sicherzustellen, dass die Antworten tatsächlich auf Vallader oder zumindest auf Rätoromanisch sind, wurden die Antworten jeweils mit dem AI-Tool Textshuttle übersetzt.

Nach mehreren weiteren Tests und Anpassungen der Systemnachricht wurde in Absprache mit dem Auftraggeber entschieden, das GPT-4-Modell für den finalen Einsatz im Chatbot zu verwenden.

4.4 Implementierung des Chatbots

Die Implementierung des Chatbots erforderte zunächst eine gründliche Recherche über die Technologien und Plattformen für diese Aufgabe. Dabei wurden Node.js, React und Vite als geeignete Technologien identifiziert, die einen soliden Grundstein für die Entwicklung eines Chatbots bieten. Das Projekt wurde in zwei Hauptteile gegliedert: eine Serverdatei und eine Clientdatei. Die Serverdatei enthält die Verbindung zur OpenAI API und die Implementierung der Text-to-Speech (TTS)-Engine, während die Clientdatei das Userinterface bereitstellt. In den nächsten Unterabschnitten werden die wichtigsten Codebausteine der jeweiligen Dateien näher erklärt.

4.4.1 Technologieauswahl: Node.js, React und Vite

Für die Entwicklung des Chatbots wurden verschiedene Technologien und Plattformen verwendet. Dazu gehören Node.js, React und Vite, die jeweils spezielle Funktionen erfüllen.

4.4.1.1 Node.js

Node.js ist eine Open-Source-Plattform, die es ermöglicht, serverseitige Anwendungen mit JavaScript zu erstellen (*Node.js*, o. J.). Durch die Verwendung von Node.js konnte eine Serverumgebung erstellt werden, die das Hosting der Anwendung und das Senden der notwendigen Anforderungen an die OpenAI-API ermöglicht.

4.4.1.2 React

React ist eine JavaScript-Bibliothek, die von Facebook entwickelt wurde und es ermöglicht, Benutzeroberflächen für Webanwendungen zu erstellen (*React*, o. J.). Ein grosser Vorteil von React ist die Unterstützung für Komponenten, was bedeutet, dass Teile der Benutzeroberfläche wiederverwendet und effizienter gestaltet werden konnten. Ein weiterer Grund für die Wahl von React war die Bibliothek '@chatscope/chat-ui-kit-react', die speziell für die Erstellung von Chat-Oberflächen entwickelt wurde.

4.4.1.3 Vite

Vite ist ein modernes Frontend-Build-Tool, das für die schnelle Entwicklung von Webanwendungen konzipiert ist. Es bietet eine sofortige Serverstartzeit und blitzschnelles Hot-Module-Ersetzen, was bedeutet, dass Änderungen im Code sofort sichtbar sind und somit eine effizientere Arbeitsweise ermöglicht wird (*Vite*, o. J.).

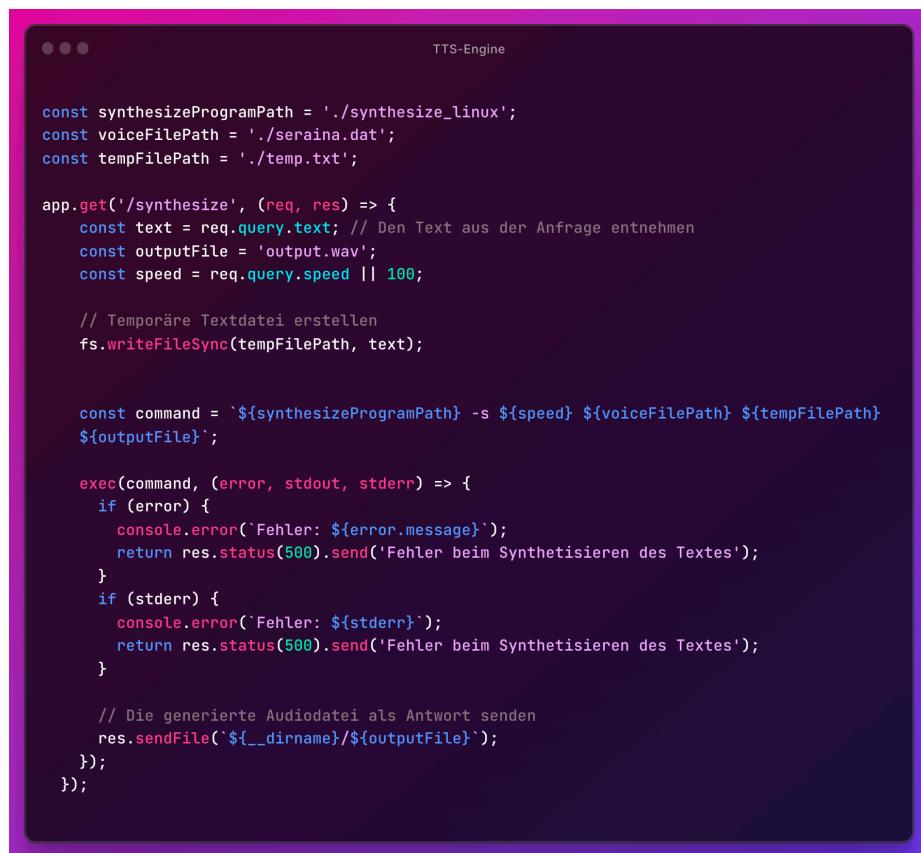
Die Kombination dieser Technologien ermöglichte eine effiziente Entwicklung des Chatbots. Im nächsten Unterabschnitt wird genauer auf die Implementierung des Chatbots eingegangen, beginnend mit der Einrichtung der Text-to-Speech-Engine.

4.4.2 Serverdatei: TTS Engine

Um sicherzustellen, dass der Chatbot in der Lage ist, seine Ausgabe auch vorzulesen, wurde nach einer entsprechenden Text-to-Speech-Engine recherchiert. Dabei wurde die Lösung der SlowSoft GmbH entdeckt. Diese Firma wurde im November 2014 von zwei Experten im Bereich der Sprachtechnologie gegründet und hat sich auf die Entwicklung von Sprachsynthese spezialisiert. Zwischen 2016 und 2017 arbeitete die Firma an der Erstellung von Sprachdaten für Schweizerdeutsch, Rätoromanisch (Vallader) und Schweizer Hochdeutsch. SlowSoft bietet eine Reihe von Produkten an, die für die Implementierung der Sprachsynthese in Chatbots genutzt werden können (slowsoft, o. J.).

Eine Kontaktaufnahme mit der SlowSoft GmbH ermöglichte eine Untersuchung der besten Möglichkeiten zur Integration ihrer Engine in den Chatbot. Zwei Optionen standen zur Verfügung: entweder eine Anbindung über eine API oder eine direkte Bereitstellung der Engine. Um einen nahtlosen und unabhängigen Chatbot zu entwickeln, wurde die direkte Bereitstellung der Engine bevorzugt. Diese Entscheidung erhöht zwar die Komplexität der Entwicklung, garantiert aber die Qualität der Ausgaben. Der Gründer Christoph Traber zeigte sich offen für die Zusammenarbeit und bot an, die benötigten Dateien zur lokalen Ausführung zu erstellen und bereitzustellen.

Die Hauptdateien für die Implementierung der Text-to-Speech-Engine sind die "seraina.dat"- und "synthesize"-Dateien. Eine detaillierte Erklärung des Inhalts dieser Dateien geht über den Rahmen dieser Arbeit hinaus. Um die "synthesize"-Datei in dem Projekt verwenden zu können, mussten Ausführungsrechte im Terminal vergeben werden. Christoph Traber stellte hierfür einen ausführlichen Leitfaden zur Verfügung. Nach einigen Tests im Terminal konnte die Engine in die Serverdatei implementiert werden. Der relevante Codeausschnitt zur Implementierung der TTS-Engine sieht folgendermassen aus:



```
● ● ● TTS-Engine

const synthesizeProgramPath = './synthesize_linux';
const voiceFilePath = './seraina.dat';
const tempFilePath = './temp.txt';

app.get('/synthesize', (req, res) => {
  const text = req.query.text; // Den Text aus der Anfrage entnehmen
  const outputFile = 'output.wav';
  const speed = req.query.speed || 100;

  // Temporäre Textdatei erstellen
  fs.writeFileSync(tempFilePath, text);

  const command = `${synthesizeProgramPath} -s ${speed} ${voiceFilePath} ${tempFilePath}
${outputFile}`;

  exec(command, (error, stdout, stderr) => {
    if (error) {
      console.error(`Fehler: ${error.message}`);
      return res.status(500).send('Fehler beim Synthetisieren des Textes');
    }
    if (stderr) {
      console.error(`Fehler: ${stderr}`);
      return res.status(500).send('Fehler beim Synthetisieren des Textes');
    }

    // Die generierte Audiodatei als Antwort senden
    res.sendFile(`${__dirname}/${outputFile}`);
  });
});
```

Abbildung 18: Code-Ausschnitt der TTS-Engine

Zuerst werden drei Konstanten definiert, die Pfade zu den relevanten Dateien und Ressourcen enthalten:

- **synthesizeProgramPath** ist der Pfad zum Syntheseprogramm, das von SlowSoft bereitgestellt wurde und das die Text-zu-Sprache-Synthese durchführt.
- **voiceFilePath** ist der Pfad zur "seraina.dat"-Datei, die die für die Sprachsynthese notwendigen Daten enthält.
- **tempFilePath** ist der Pfad zu einer temporären Textdatei, die dazu dient, den Text zu speichern, der in Sprache umgewandelt werden soll.

Die `app.get('/synthesize')`-Methode erstellt einen Endpunkt zum Express-Server, der auf GET-Anfragen unter der URL "/synthesize" antwortet.

Wenn eine Anfrage an diesen Endpunkt gesendet wird, wird zuerst der Text aus der Anfrage extrahiert und in der **text**-Konstante gespeichert. Dann werden ein **outputFile**-Name definiert und eine **speed**-Variable festgelegt, welche die Geschwindigkeit der Audioausgabe definiert.

Die `fs.writeFileSync(tempFilePath, text)`-Zeile schreibt den empfangenen Text in die temporäre Textdatei, die zuvor definiert wurde.

Ein Befehl wird erstellt, der das Syntheseprogramm aufruft und als Argumente die Geschwindigkeit, die Sprachdatei, die temporäre Textdatei und den Namen der Ausgabedatei enthält.

Die **exec**-Methode führt dann diesen Befehl aus. Tritt bei der Ausführung des Befehls ein Fehler auf, wird der Fehlerprotokoll an die Konsole und eine Fehlermeldung zurück an den Client gesendet. Wenn die Ausführung erfolgreich ist, sendet der Server die generierte Audiodatei als Antwort auf die Anfrage.

4.4.3 Serverdatei: OpenAI API

Im Rahmen der Implementierung der Interaktion mit der OpenAI API wurde zunächst eine Wissensdatenbank geladen, die als JSON-Datei namens "wissensDatenbank.json" gespeichert ist. Diese Wissensdatenbank enthält wesentliche Informationen und Beispieldialoge, die dem Chatbot helfen sollen, die Vallader-Sprache besser zu verstehen und effizienter auf Anfragen zu reagieren.

Die Struktur der Wissensdatenbank ist recht einfach, sie ist so aufgebaut wie die Benutzernachrichten aus dem Unterabschnitt 3.3.4 und besteht aus einer Reihe von Dialogen, bei denen jede Nachricht einen Absender ("role": "user" oder "assistant") und den Inhalt der Nachricht ("content") hat. Hier ist ein Ausschnitt aus der Wissensdatenbank:



```
wissensDatenbank.json

[
  { "role": "user", "content": "Allegra" },
  { "role": "assistant", "content": "Allegra, co poss jau gidar" },
  { "role": "user", "content": "Be amo tschinck nots e lura n'haja anniversari" },
  { "role": "assistant", "content": "Faina üna festina?" },
  { "role": "user", "content": "Hier ein kurzer Text, damit du besser Vallader kannst. Der Text ist ein Dialog und es geht um Zahnschmerzen: «Auaa», plondscha Lupo, «meis daints fan suul mal.» Lupo schmetta da rusiglier vi da seis öss e til lascha crodar our d'bocca. «Ha! N'ha bain pensà cha teis daints gnian marschs cun magliar glatsch! Uossa hast la merda, uossa varast il prüm dolurs, lura crodarà oura ün daint, lura amo ün e lura amo ün - fin cha tü nun hast plü ingüns. E lura poust be amo magliar schoppa e mösas fin cha tü mourast!», disch Brutus cun tun dad important. «Ojee, ma quai nu lessa!», disch Lupo e guarda tuot sculozzà." },
  { "role": "assistant", "content": "Grazia fitg per questa declerazion" },
]
```

Abbildung 19: Ausschnitt aus der Wissensdatenbank

Die oben aufgeführten Beispiele illustrieren, wie der Benutzer dem Chatbot verschiedene Sätze und Dialoge in Vallader vorstellt und wie der Chatbot darauf reagiert. Die vollständige Wissensdatenbank ist im Anhang zu finden. Indem solche spezifischen Dialoge zur Wissensdatenbank hinzugefügt werden, bekommt der Chatbot eine Grundlage, um das Vallader-Idiom und die entsprechenden Reaktionsmuster zu verstehen. Ebenso wurden Passagen aus den Kinderbüchern integriert, jeweils vorangestellt von einer kurzen Erläuterung, um dem Bot ein tieferes Verständnis des Textes zu ermöglichen. Solche ergänzenden Massnahmen tragen dazu bei, die Genauigkeit der Antworten des Chatbots zu verbessern.



```
OpenAI API

let predefinedMessages
try {
  const rawData = fs.readFileSync('./assets/wissensDatenbank.json');
  predefinedMessages = JSON.parse(rawData);
} catch (error) {
  console.error('Fehler beim Lesen der JSON-Datei:', error);
}
```

Abbildung 20: Zuweisung der Wissensdatenbank

In dem obigen Codeausschnitt wird die JSON-Datei gelesen und das Ergebnis wird in das `rawData`-Objekt geladen. Die Daten werden anschliessend mittels der `JSON.parse()`-Funktion in ein JavaScript-Objekt konvertiert und der Variable `predefinedMessages` zugewiesen.

Um die Anfragen an die OpenAI API zu bearbeiten, wird eine POST Route `/processMessage` definiert. Diese Route nimmt die Chat-Nachrichten des Benutzers entgegen und bereitet sie für die Anfrage an die OpenAI API vor.



The screenshot shows a code editor window with a dark theme. The title bar says "OpenAI API". The code is written in JavaScript and defines a POST route for "/processMessage". It starts by extracting the "chatMessages" array from the request body. Then, it maps over each message object to create a new object with "role" (either "assistant" or "user" based on the sender) and "content" (the message text). A "systemMessage" object is defined with role "system" and content about being a helpful assistant trained to answer Vallader inputs. Finally, an "apiRequestBody" object is created with "model": "gpt-4" and "messages": [systemMessage, ...predefinedMessages, ...apiMessages].

```
app.post('/processMessage', async (req, res) => {
  const chatMessages = req.body.chatMessages;

  let apiMessages = chatMessages.map((messageObject) => {
    let role = "";
    if (messageObject.sender === "ChatGPT") {
      role = "assistant"
    } else {
      role = "user"
    }
    return { role: role, content: messageObject.message }
  });

  const systemMessage = {
    role: "system",
    content: "Du bist ein hilfreicher Assistent, der darauf trainiert ist, Fragen auf Vallader zu beantworten. Du kannst Inputs auf Vallader, Deutsch oder Englisch nehmen, aber deine Antworten sind immer auf Vallader."
  }

  const apiRequestBody = {
    "model": "gpt-4",
    "messages": [systemMessage,
      ...predefinedMessages,
      ...apiMessages]
  }
});
```

Abbildung 21: Nachrichtenverarbeitung für die OpenAI API

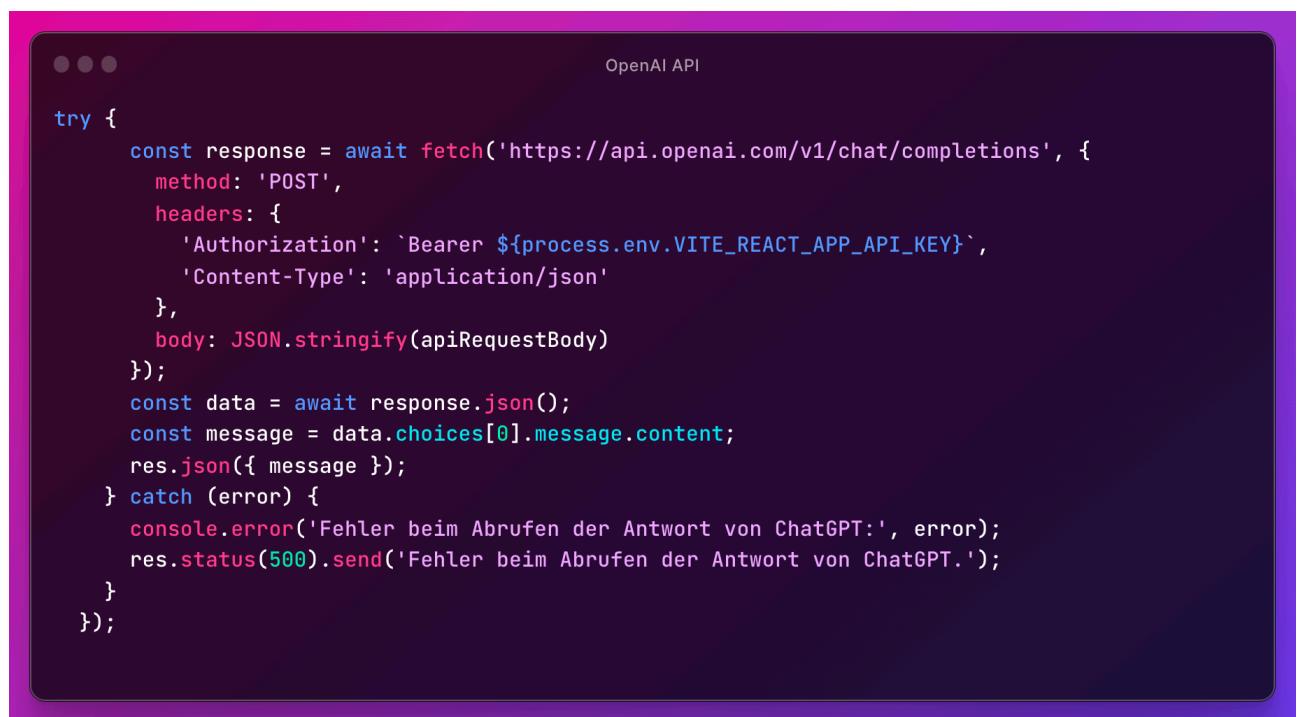
In diesem **Codeabschnitt** wird zunächst das Array `chatMessages` aus dem Request Body extrahiert. Dieses Array enthält die von der Benutzeroberfläche gesendeten Chat-Nachrichten.

Dann wird für jede Nachricht ein neues Objekt erstellt, das sowohl die Rolle als auch den Inhalt der Nachricht enthält. Die Rolle wird basierend auf dem Absender der Nachricht festgelegt – entweder "assistant" für den ChatGPT (also den Chatbot) oder "user" für den Benutzer. Anschliessend werden diese neu formatierten Nachrichten in das Array `apiMessages` eingefügt.

Indem die Nachrichten in diesem formatierten Array gespeichert werden, ermöglicht es die Struktur, den Dialogverlauf aufrechtzuerhalten und auf vorherige Nachrichten zu verweisen, um bei Bedarf auf bereits gestellte Fragen oder Anweisungen zu reagieren. Dies erleichtert die Kontinuität und den kontextbezogenen Austausch zwischen dem Chatbot und dem Benutzer.

Ein zentraler Schritt in diesem Prozess ist die Definition der `systemMessage`. Diese Nachricht hat die Aufgabe, dem Modell die Rolle und das Verhalten des Chatbots zu erläutern. Diese Anweisung definiert den Chatbot als einen Assistenten, der dazu dient, auf Fragen in Vallader zu antworten. Dabei kann er Eingaben in Vallader, Deutsch oder Englisch entgegennehmen, jedoch sind seine Antworten stets auf Vallader. Die präzise Formulierung dieser Systemnachricht ist entscheidend, da sie massgeblich die Ausgabe des Modells beeinflusst und somit die Qualität der Antworten des Chatbots prägt.

Diese Nachrichten werden dann zusammen mit den vordefinierten Nachrichten aus der Wissensdatenbank und der Systemnachricht in den Request Body für die OpenAI API eingefügt.



```
● ● ● OpenAI API
try {
  const response = await fetch('https://api.openai.com/v1/chat/completions', {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${process.env.VITE.REACT_APP_API_KEY}`,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(apiRequestBody)
  });
  const data = await response.json();
  const message = data.choices[0].message.content;
  res.json({ message });
} catch (error) {
  console.error('Fehler beim Abrufen der Antwort von ChatGPT:', error);
  res.status(500).send('Fehler beim Abrufen der Antwort von ChatGPT.');
}
});
```

Abbildung 22: Verarbeitung der API-Antwort

Nach der Vorbereitung der Daten wird eine POST-Anfrage an die OpenAI API gesendet. In den Kopfdaten (Headers) dieser Anfrage wird eine 'Authorization' mit dem Wert 'Bearer YOUR_API_KEY' gesetzt, wobei 'YOUR_API_KEY' durch den tatsächlichen API-Schlüssel ersetzt wird, der in der Umgebungsvariablen `process.env.VITE.REACT_APP_API_KEY` gespeichert ist. Dieser API-Schlüssel ist für die Authentifizierung der Anfrage bei der OpenAI API erforderlich. Er stellt sicher, dass die Nutzungsgebühren korrekt auf das Konto des Benutzers angerechnet werden.

Bei einer erfolgreichen Anfrage wird die Antwort des Chatbots aus der API-Antwort extrahiert und als Antwort auf die ursprüngliche Anfrage des Clients gesendet. Im Falle eines Fehlers wird eine Fehlermeldung ausgegeben und auf die Konsole geloggt, um die Fehlersuche zu erleichtern.

4.4.4 Clientdatei: Userinterface Chatbot

In dieser Sektion des Kapitels wird die Client-Datei analysiert, in der das User Interface des Chatbots implementiert ist. Die Datei ist in JavaScript mit der React-Bibliothek geschrieben und verwendet zusätzlich das Paket `@chatscope/chat-ui-kit-react` zur Gestaltung der Chat-Oberfläche. Um HTTP-Anfragen zu senden, wird das Paket `axios` verwendet.

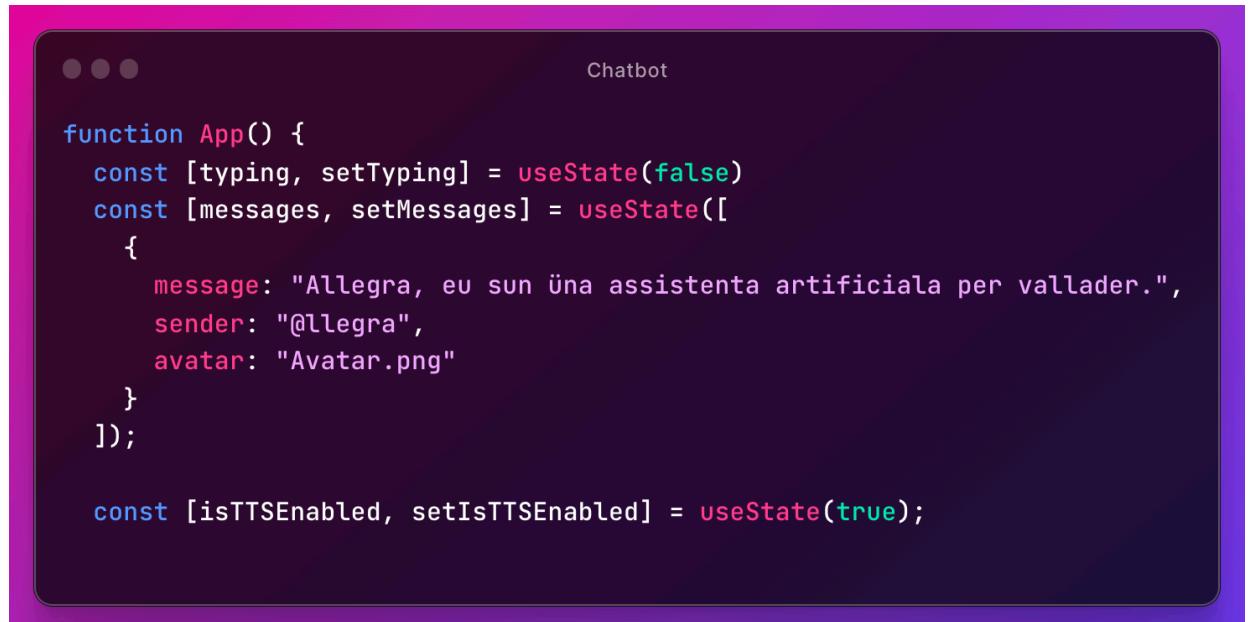


```
Chatbot

import { useState, useRef, useEffect } from 'react'
import './App.css'
import '@chatscope/chat-ui-kit-styles/dist/default/styles.min.css';
import { MainContainer, ChatContainer, MessageList, MessageInput, Message, TypingIndicator, Avatar } from '@chatscope/chat-ui-kit-react';
import botAvatar from '/src/assets/Avatar.png';
import React from 'react'
import axios from 'axios';
```

Abbildung 23: Import der Ressourcen und Pakete

Zu Beginn der Datei werden die notwendigen Ressourcen und Pakete importiert:



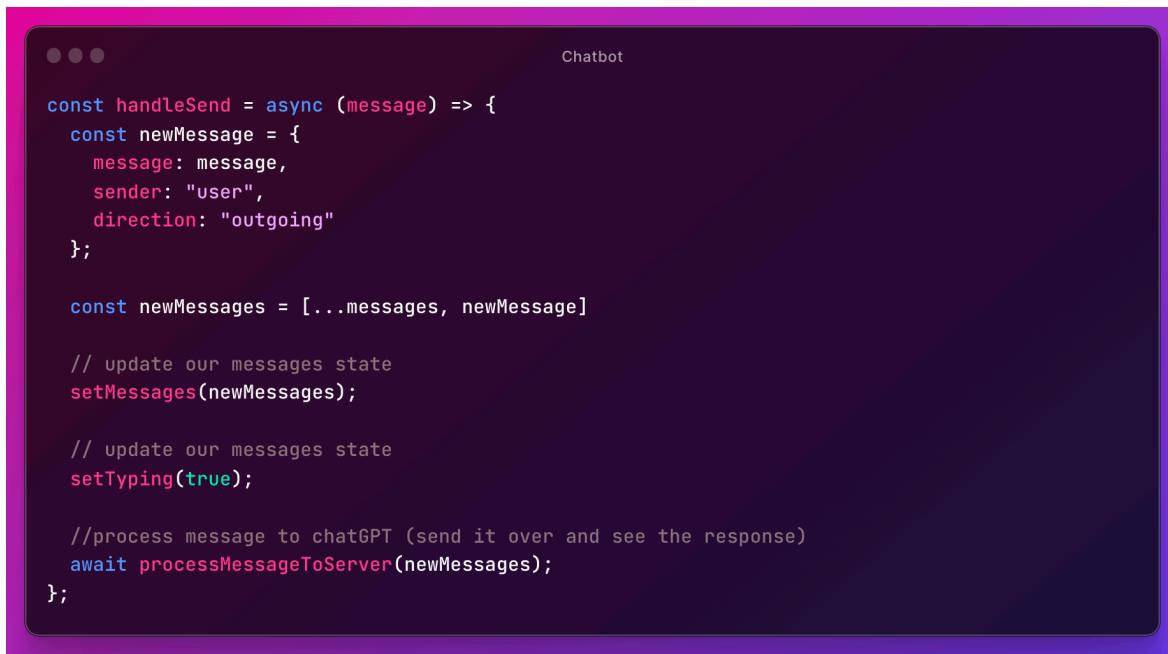
```
Chatbot

function App() {
  const [typing, setTyping] = useState(false)
  const [messages, setMessages] = useState([
    {
      message: "Allegra, eu sun üna assistenta artificiala per vallader.",
      sender: "@allegra",
      avatar: "Avatar.png"
    }
  ]);
  const [isTTSEnabled, setIsTTSEnabled] = useState(true);
```

Abbildung 24: Initialisierung der Zustandsvariablen

Innerhalb der Hauptfunktion `App()` werden mehrere Zustandsvariablen mit der Hook-Funktion `useState` von React initialisiert. Diese Zustandsvariablen halten die Daten zwischen den Render-Vorgängen fest und führen zur Neurenderung der Komponente, wenn sich ihr Wert ändert (*Using the State Hook – React*, o. J.). Die Zustandsvariable `messages` ist speziell für die Verwaltung von

Chat-Nachrichten vorgesehen. Die Anfangswert-Initialisierung dieser Variable definiert die erste Nachricht des Chatbots. Diese Nachricht dient als Begrüßungsnachricht in Vallader, sobald der Chat gestartet wird. Zudem wird durch die Eigenschaft **sender** gekennzeichnet, dass die Nachricht von "@llegra", dem Chatbot, stammt, während **avatar** den Pfad zur Avatar-Bilddatei angibt.



```
const handleSend = async (message) => {
  const newMessage = {
    message: message,
    sender: "user",
    direction: "outgoing"
  };

  const newMessages = [...messages, newMessage]

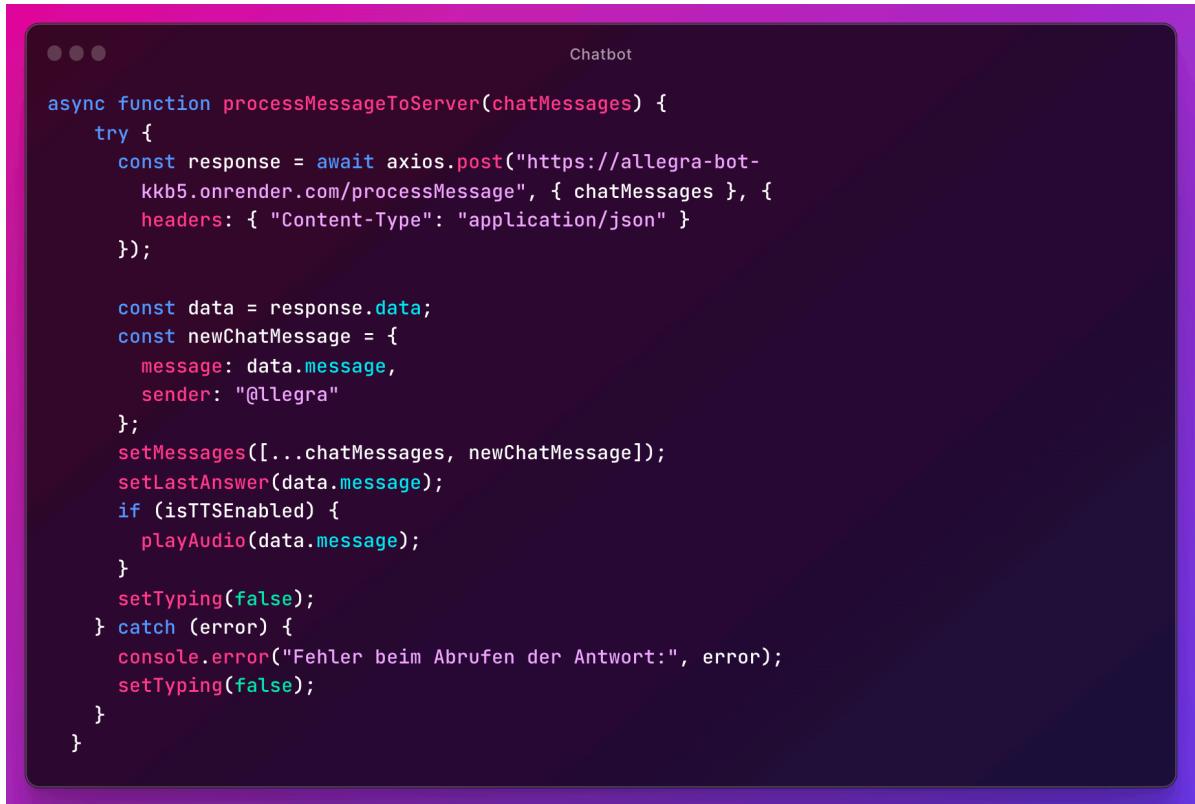
  // update our messages state
  setMessages(newMessages);

  // update our messages state
  setTyping(true);

  //process message to chatGPT (send it over and see the response)
  await processMessageToServer(newMessages);
};
```

Abbildung 25: Nachrichtenbearbeitung und Versand an Server

Die Funktion **handleSend** wird aufgerufen, wenn der Benutzer eine Nachricht sendet. Sie erstellt ein neues Nachrichtenobjekt, fügt es zum Zustand **messages** hinzu und übermittelt die Nachricht zur weiteren Verarbeitung an die Funktion **processMessageToServer**.



```

Chatbot

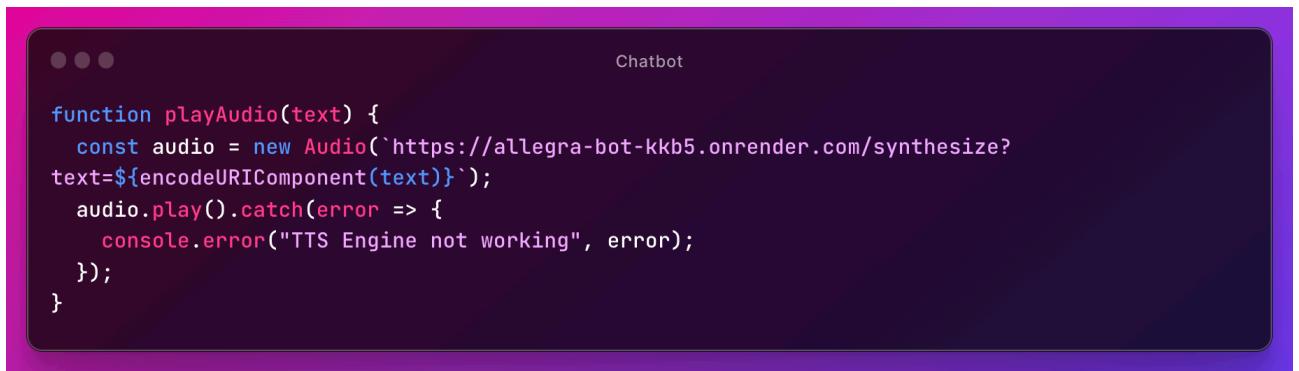
async function processMessageToServer(chatMessages) {
  try {
    const response = await axios.post("https://allegra-bot-kkb5.onrender.com/processMessage", { chatMessages }, {
      headers: { "Content-Type": "application/json" }
    });

    const data = response.data;
    const newChatMessage = {
      message: data.message,
      sender: "@allegra"
    };
    setMessages([...chatMessages, newChatMessage]);
    setLastAnswer(data.message);
    if (isTTSEnabled) {
      playAudio(data.message);
    }
    setTyping(false);
  } catch (error) {
    console.error("Fehler beim Abrufen der Antwort:", error);
    setTyping(false);
  }
}

```

Abbildung 26: Clientseitige Verarbeitung und Anzeige der Serverantwort

Die Funktion `processMessageToServer (chatMessages)` sendet die bisherigen Nachrichten und die neueste Nachricht des Benutzers an den Server. Sobald eine Antwort vom Server erhalten wird, wird diese als neue Nachricht des Chatbots den bestehenden Nachrichten hinzugefügt. Außerdem wird diese Antwort, falls die Text-zu-Sprache-Funktion aktiviert ist, als Sprachnachricht abgespielt.



```

Chatbot

function playAudio(text) {
  const audio = new Audio(`https://allegra-bot-kkb5.onrender.com/synthesize?text=${encodeURIComponent(text)}`);
  audio.play().catch(error => {
    console.error("TTS Engine not working", error);
  });
}

```

Abbildung 27: Funktion zur Audiowiedergabe der TTS-Engine

Die Funktion `playAudio (text)` erstellt eine neue Audio-Instanz, indem sie eine Anfrage an den Synthesizer-Endpunkt der Serveranwendung sendet. Der zu sprechende Text wird als URL-Parameter übergeben, wobei spezielle Zeichen durch die Funktion `encodeURIComponent ()` kodiert

werden. Bei dieser Anfrage generiert der Server eine neue Audiodatei, die von der **Audio**-Instanz abgespielt wird. Sollte beim Abspielen des Audios ein Fehler auftreten, wird eine Fehlermeldung in der Konsole ausgegeben.



```
return (
  <div className="App">
    <div className="container">
      <div>
        <h1>@llegra - Vallader-Chatbot</h1>
      </div>

      <MainContainer>
        <ChatContainer>
          <MessageList
            ref={messageListRef}
            scrollBehavior='smooth'
            typingIndicator={typing ? <TypingIndicator content="Allegra sa lascha scriver"
            /> : null}
          >
            {messages.map((message, i) => (
              <Message
                model={{
                  message: message.message,
                  direction: message.sender === "@llegra" ? "incoming" : "outgoing",
                  sender: message.sender,
                }}
                key={i}
              >
                {message.sender === "@llegra" &&
                  <Avatar src={botAvatar} name="bot avatar" onClick={() =>
                    playAudio(lastAnswer)} />
                }
              </Message>
            )));
          </MessageList>
          <MessageInput placeholder='Scrivai qua tes messadi' onSend={handleSend} />
        </ChatContainer>
      </MainContainer>
      <div className='questionbox'>
        <label>
          <input type="checkbox" checked={isTTSEnabled} onChange={() =>
            setIsTTSEnabled(!isTTSEnabled)} />
          Transfurmar text en lingua? (Text in Sprache umwandeln?)
        </label>
      </div>
    </div>
  </div>
)
```

Abbildung 28: Darstellung des Haupt-UI-Codes

Der letzte wichtige Teil ist die `return`-Anweisung, welche die Darstellung des Userinterfaces definiert. Hier werden verschiedene Komponenten aus der Chat-UI-Kit-Bibliothek verwendet, um das Chat-Fenster, die Nachrichtenliste und das Eingabefeld zu erstellen.

- Im Haupt-Container ("MainContainer") befindet sich der Chat-Container ("ChatContainer").
- Innerhalb des ChatContainers befindet sich die "MessageList"-Komponente. Diese enthält alle Nachrichten, die zwischen dem Benutzer und dem Bot ausgetauscht werden. Das Array von Nachrichten wird durchlaufen und für jede einzelne eine "Message"-Komponente generiert. Die Richtung der Nachricht (incoming oder outgoing) wird basierend auf dem Absender der Nachricht festgelegt.
- Wenn der Sender der Nachricht "ChatGPT" ist, wird auch eine "Avatar"-Komponente hinzugefügt, die den Avatar anzeigt. Bei Klick auf den Avatar wird die letzte Antwort erneut abgespielt (vorausgesetzt, die Text-to-Speech-Funktion ist aktiviert). Die Abbildung 29 zeigt den Avatar, welcher mithilfe eines KI-Generators auf der Webseite gencraft.com erstellt wurde. Als Prompt in gencraft wurde der Text 'A majestic ibex to use as an Avatar' verwendet.



Abbildung 29: Avatar von @llegra

- Der "MessageList"-Komponente ist auch ein TypingIndicator zugeordnet, der sichtbar wird, wenn der Bot gerade eine Antwort generiert.
- Unterhalb der "MessageList" befindet sich das "MessageInput"-Feld. Wenn der Benutzer eine Nachricht eingibt und sendet, wird die Funktion "handleSend" aufgerufen.
- Zuletzt gibt es noch eine Checkbox, die es dem Benutzer ermöglicht, die Text-to-Speech-Funktion ein- oder auszuschalten.

Dieser Code erstellt eine funktionale Chat-Anwendung, die eine dynamische Interaktion mit dem Chatbot ermöglicht und eine ansprechende Nutzeroberfläche bietet.

4.5 Live-Schaltung der Webseite

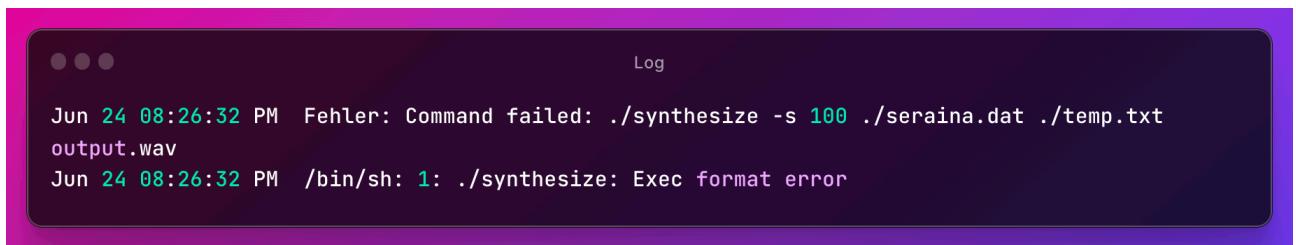
4.5.1 Hosting des Servers

Um den entwickelten Chatbot in eine produktive Umgebung zu überführen, musste der Server gehostet werden. Der erste Schritt des Prozesses bestand darin, das gesamte Projekt in ein Git-Repository hochzuladen. Hier trat die erste Herausforderung auf: die Datei serina.dat, die für die Text-to-Speech-Engine unerlässlich ist, hatte eine Grösse von über 100 MB. Da Git-Repositories jedoch eine Dateigrößenbeschränkung von 100 MB haben, musste eine Lösung gefunden werden.

Hier kam Git Large File Storage (LFS) ins Spiel. Git LFS ist eine Open-Source-Erweiterung für Git, die das Handling von grossen Dateien verbessert. Statt die grossen Dateien direkt im Repository zu speichern, verwaltet Git LFS diese Dateien und speichert die tatsächlichen Dateien in einem separaten Speicherplatz. Auf diese Weise konnte die serina.dat-Datei trotz ihrer Grösse ins Projekt integriert werden.

Nachdem das Projekt erfolgreich auf dem Git-Repository abgelegt worden war, wurden verschiedene Server-Hosting-Plattformen verglichen. Die Wahl fiel auf Render. Dort wurde das Git-Repository verbunden und der API-Key für die OpenAI-API als Umgebungsvariable eingefügt.

Das Hochfahren des Servers verlief zunächst ohne Fehler, jedoch traten Probleme auf, als die Verbindung zum Client getestet wurde. Es stellte sich heraus, dass die Synthesize-Datei der TTS-Engine, die für MacOS kompatibel ist, nicht mit der Linux-basierten Render-Plattform kompatibel ist. Dies resultierte in der folgenden Fehlermeldung auf der Serverkonsole:



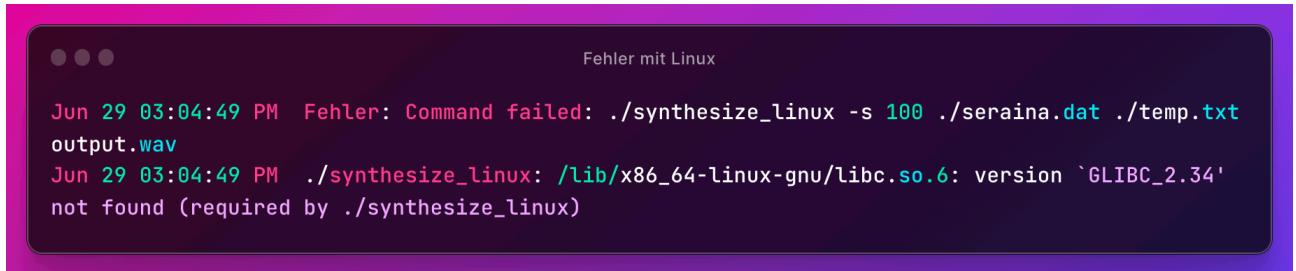
```
... Log  
Jun 24 08:26:32 PM Fehler: Command failed: ./synthesize -s 100 ./seraina.dat ./temp.txt  
output.wav  
Jun 24 08:26:32 PM /bin/sh: 1: ./synthesize: Exec format error
```

Abbildung 30: Fehlermeldung betreffend Inkompatibilität der 'synthesize'-Datei.

Nachdem Hilfe vom Render-Support eingeholt wurde, kam die Information, dass die Dateien Linux-basiert sein müssen, um vom Server gelesen werden zu können.

Für die Live-Version musste deshalb eine neue Synthesize-Datei angefordert werden, die mit Linux kompatibel ist. Glücklicherweise konnte der Gründer von SlowSoft, Christoph Traber, eine solche Datei von einem Bekannten erstellen lassen.

Nachdem die Linux kompatible Synthesize-Datei eingetroffen war, wurde sie gegen die alte Datei ausgetauscht, in der Hoffnung auf eine reibungslose Ausführung. Trotz dieser Anpassung trat leider eine weitere Fehlermeldung auf:

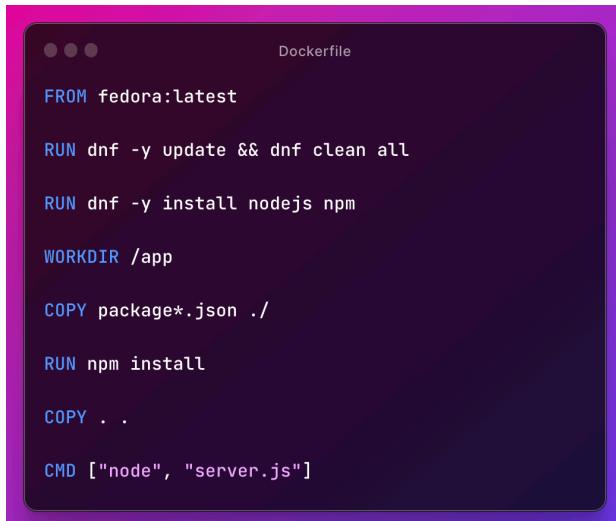


```
● ● ● Fehler mit Linux
Jun 29 03:04:49 PM Fehler: Command failed: ./synthesize_linux -s 100 ./seraina.dat ./temp.txt
output.wav
Jun 29 03:04:49 PM ./synthesize_linux: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.34'
not found (required by ./synthesize_linux)
```

Abbildung 31: Fehlermeldung wegen inkompatibler Linux-Version

Um dieses Problem zu lösen, schlug Render vor, Docker zu verwenden. Docker ist eine Open-Source-Softwareplattform, die die Erstellung, Bereitstellung und Ausführung von Anwendungen in Containern ermöglicht. Ein Container ist eine standardisierte Einheit, die Code, Laufzeit, Systemwerkzeuge, Systembibliotheken und Einstellungen enthält. Container sind von der Umgebung isoliert und gewährleisten so, dass die Anwendung unter den gleichen Bedingungen läuft, unabhängig von Unterschieden in der Infrastruktur (*Docker*, o. J.).

Das Dockerfile sah wie folgt aus:



```
● ● ● Dockerfile
FROM fedora:latest

RUN dnf -y update && dnf clean all

RUN dnf -y install nodejs npm

WORKDIR /app

COPY package*.json .

RUN npm install

COPY . .

CMD ["node", "server.js"]
```

Abbildung 32: Dockerfile

Es musste ein Docker-Image gefunden werden, das mit der Version 'GLIBC_2.34' kompatibel ist. Die Wahl fiel dabei auf 'fedora' – nach dessen Implementierung wurden keine weiteren Probleme festgestellt. Die erfolgreiche Kombination von Git LFS, Render und Docker ermöglichte es, einen Linux-kompatiblen Server zu hosten und den Chatbot live zu schalten.

4.5.2 Hosting des Clients

Im Gegensatz zum Serverhosting war das Hosting des Clients eine relativ unkomplizierte Angelegenheit. Der Client wurde mit Hilfe von Netlify gehostet, einer Plattform, die speziell für das Hosting und die Bereitstellung von Frontend-Anwendungen entwickelt wurde.

Die Einrichtung mit Netlify gestaltete sich als unkompliziert und benutzerfreundlich. Zunächst wurde die Client-Git-Repository mit Netlify verknüpft, was einen nahtlosen Übergang und eine kontinuierliche Integration ermöglichte. Sobald eine Änderung in der Git-Repository durchgeführt wurde, war Netlify in der Lage, die Änderungen automatisch zu erfassen und die neueste Version des Clients bereitzustellen.

Ein wichtiger Aspekt, der vor dem Hochladen des Clients berücksichtigt werden musste, war die Anpassung der Serveradresse im Client-Code. Der Client muss wissen, wo der Server gehostet wird, um Anfragen korrekt an ihn zu senden. Zudem stand früh fest, dass die Domain für die Website <https://allegra-bot.ch> lauten sollte. Die Domain konnte direkt über Netlify erstellt und verknüpft werden. Nachdem sowohl der Server als auch der Client erfolgreich hochgeladen und gehostet wurden, konnte mit den Benutzertests begonnen werden, die im nächsten Kapitel besprochen werden. Das Hosting des Clients erwies sich somit als ein kritischer Schritt im Prozess, da es ermöglichte, das Endprodukt in einer Live-Umgebung zu sehen und zu testen.

5 Ergebnisse

Nach der ausführlichen Darstellung des Entwicklungsprozesses des Chatbots führt dieses Kapitel zu den konkreten Ergebnissen des Projekts. Hier wird sowohl die entstandene Webseite und ihre drei Hauptseiten als auch die Ergebnisse der durchgeführten Benutzertests vorgestellt.

5.1 Beschreibung der Webseite

Die [Webseite](#) des Projekts ist klar strukturiert und bietet den Benutzern ein intuitives und angenehmes Benutzererlebnis. Sie besteht aus drei Hauptseiten, die über eine Navigationsleiste leicht zugänglich sind: die Informationsseite, die Chatbot-Seite und die Kontaktseite.

5.1.1 Informationsseite

Die Informationsseite dient als erste Anlaufstelle für Benutzer, die mehr über das Projekt erfahren möchten. Sie enthält grundlegende Informationen zum Projekt, seiner Entstehung und den zugrundeliegenden Technologien. Darüber hinaus gibt sie den Benutzern wichtige Hinweise zur Benutzung des Chatbots, um die Interaktion so nahtlos wie möglich zu gestalten. Der Zweck dieser Seite ist es, den Nutzern ein besseres Verständnis des Projekts zu vermitteln und sie darauf vorzubereiten, wie sie den Chatbot effektiv nutzen können.



Abbildung 33: Informationsseite der Webseite

5.1.2 Chatbot

Die Chatbot-Seite, die als Hauptseite der Webseite fungiert, ist der Ort, an dem die Benutzer mit @allegra interagieren können. Hier können die Benutzer ihre Anfragen in einem Textfeld eingeben und die Antworten des Chatbots in Echtzeit anzeigen lassen. Eine besondere Eigenschaft von @allegra ist ihre Mehrsprachigkeit: Benutzer können ihre Fragen auf Rätoromanisch, Deutsch oder Englisch stellen, und der Chatbot sollte immer auf Vallader (Rätoromanisch) antworten.

Unterhalb des Chatbot-Containers gibt es eine Option zum Ein- und Ausschalten der Text-to-Speech (TTS)-Engine. Wenn die TTS-Engine aktiviert ist, werden die Antworten des Chatbots als Audiodatei wiedergegeben. Dies erweist sich als besonders vorteilhaft für Benutzer, die Schwierigkeiten haben, den Dialekt zu lesen. Darüber hinaus können Benutzer die letzte Audiodatei erneut abspielen, indem sie auf den Avatar des Chatbots klicken.

Der Hauptzweck dieser Seite ist es, den Benutzern eine Plattform zu bieten, auf der sie direkt mit dem Chatbot kommunizieren, seine Fähigkeiten nutzen und die einzigartigen Funktionen des Projekts erleben können.

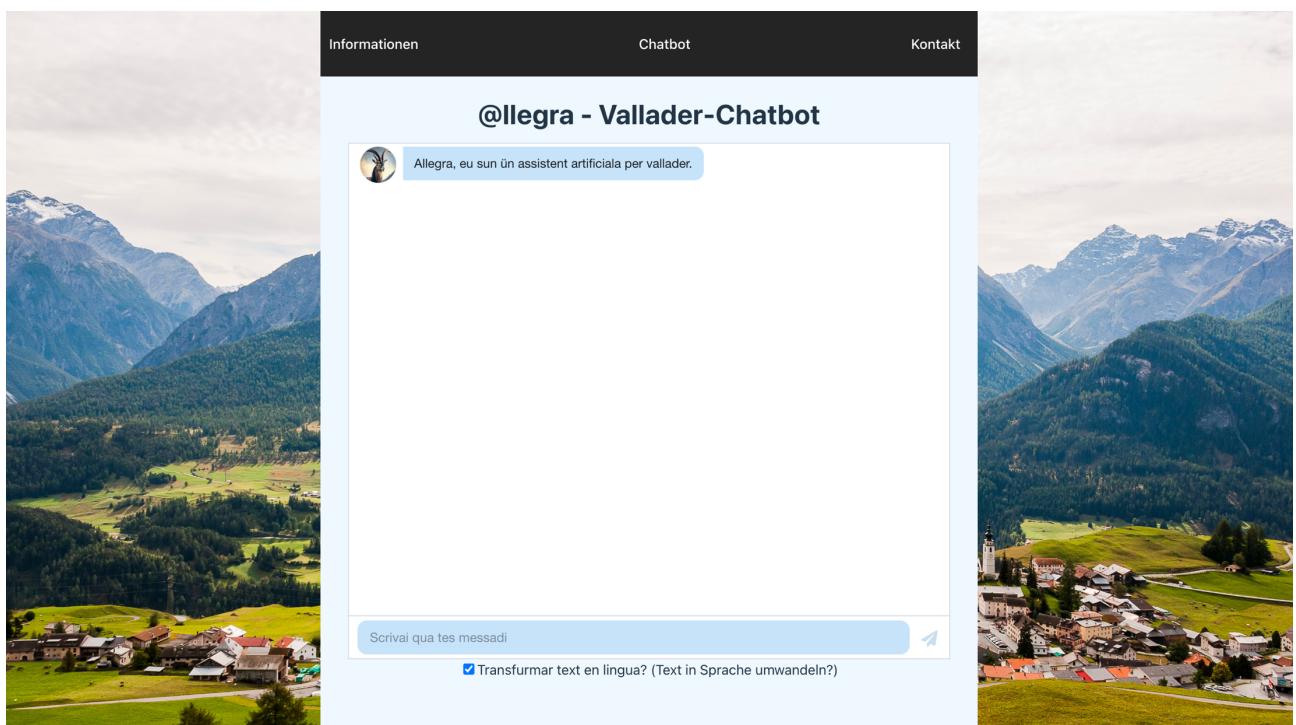


Abbildung 34: Chatbot auf der Webseite

5.1.3 Kontakt

Die Kontaktseite bietet Informationen zu den Personen hinter dem Projekt. Sie enthält Kontaktdaten und weitere Informationen zum Auftraggeber und zum Entwickler. Diese Seite ermöglicht es den Benutzern, bei Bedarf direkten Kontakt aufzunehmen.

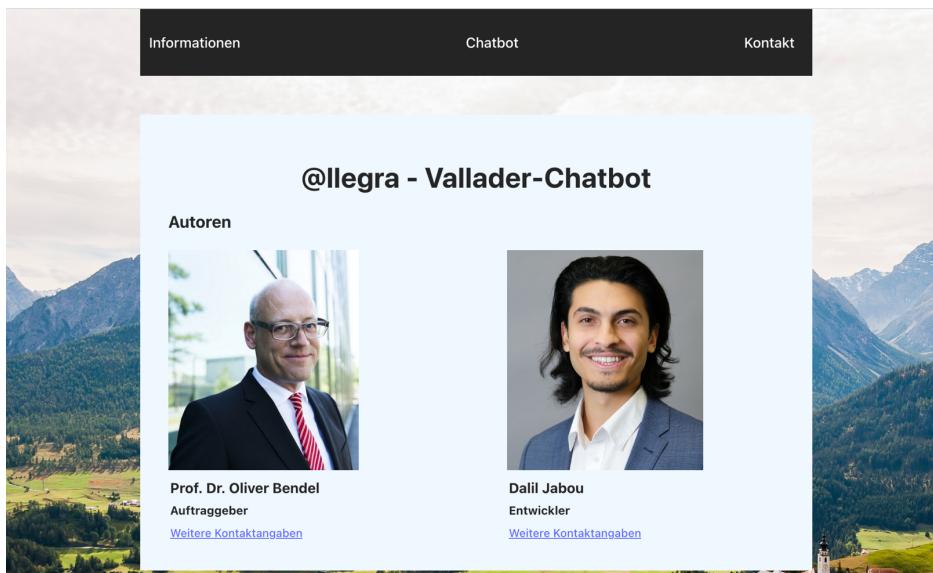


Abbildung 35: Kontaktangaben auf der Webseite

5.2 Durchführung von Benutzertests

Im Rahmen dieses Projekts wurden drei Benutzertests durchgeführt, um die Funktionalität und Benutzerfreundlichkeit von @llegra zu überprüfen und mögliche Bereiche für Verbesserungen zu identifizieren. Bei zwei dieser Tests wurden die Konversationsverläufe zur Analyse in einer Excel-Tabelle gespeichert und den Benutzern wurden sechs allgemeine Fragen zum Chatbot gestellt. Die Antworten auf diese Fragen dienten als wertvolles Feedback zur weiteren Optimierung der Benutzererfahrung. Diese Konversationstabellen und die Antworten auf die Fragen sind im Anhang zu finden.

Bei einem Benutzertest wurde eine etwas andere Methode angewandt. Der Auftraggeber selbst war vor Ort im Unterengadin und konnte eine lokale Testperson für den Benutzertest gewinnen. Dies bot die einmalige Gelegenheit, ein direktes Feedback von einem Nutzer zu erhalten, der in der dialekt-sprachigen Region ansässig ist.

In den folgenden Unterabschnitten werden die einzelnen Benutzertests im Detail besprochen.

5.2.1 Benutzertest 1: Rätoromanisch als Muttersprache

Der erste Benutzertest wurde mit einer Person durchgeführt, die Rätoromanisch als Muttersprache spricht. Aus dem Konversationsverlauf ergeben sich keine grösseren Erkenntnisse, was ein gutes Zeichen für die grundsätzliche Funktionsfähigkeit und Verständlichkeit von @llegra ist.

Die Testperson, Nadja, lobte das Bild auf der Homepage, da es sie an ihre Heimat Scuol erinnerte und die Benutzeroberfläche des Chatbots fand sie sehr benutzerfreundlich. Sie gab an, dass der Chatbot ihre Anfragen korrekt verstand und die Antworten verständlich und hilfreich waren. Sie bemerkte auch, dass die Qualität der Sprachausgabe überwiegend klar und gut zu verstehen war. Einige Wörter, wie zum Beispiel "muntogna", wurden jedoch nicht ganz korrekt ausgesprochen.

Nadja schlug vor, dass es hilfreich wäre, die Sprachausgabe des Chatbots öfter abspielen zu können, da sie derzeit nur einmal ausgeführt wird. Dieses Feedback führte dazu, dass der Chatbot um eine neue Funktion ergänzt wurde: Durch Klicken auf den Avatar kann nun die Audiodatei erneut abgespielt werden.

Sie stiess während des Tests auf keine Probleme oder Herausforderungen und fand den Chatbot für Personen nützlich, die Rätoromanisch als Muttersprache sprechen und schnell eine Antwort auf ihre Fragen möchten.

Zusammenfassend war der erste Benutzertest mit einer rätoromanischen Muttersprachlerin erfolgreich und führte zu wertvollen Verbesserungen in der Funktionalität und Benutzerfreundlichkeit von @llegra.

5.2.2 Benutzertest 2: Live-Test im Unterengadin

Dieser Unterabschnitt stützt sich teilweise auf Informationen aus einem Blogbeitrag von Prof. Dr. Oliver Bendel, der auf seiner Website Informationsethik.net veröffentlicht wurde.

Für den zweiten Benutzertest führte der Projektleiter, Prof. Dr. Oliver Bendel, selbst einen Live-Test mit @llegra im Unterengadin durch. Am 9. Juli 2023 war er mit dem Chatbot in der Region unterwegs und suchte aktiv nach einer Testperson, die Vallader als Muttersprache spricht.

In Tarasp Fontana, in der Nähe von Scuol, fand er eine geeignete Testperson: eine etwa 78-jährige Frau, die gerade vom Nordic Walking zurückkam. Diese Vallader-Sprecherin hörte sich die Antwort des Chatbots auf eine Frage zu Tarasp an und begann ungefragt, das Gesprochene ins Deutsche zu übersetzen. Anschliessend wurde @llegra gefragt, ob sie Scuol kennt. Der Chatbot gab daraufhin eine ausführliche Antwort in Vallader. Die ältere Dame bestätigte, dass die Aussprache gut und der Inhalt verständlich sei. Sie war von der Leistung des Chatbots beeindruckt und fragte, ob es sich um künstliche Intelligenz handle.

Dieser Live-Test bot eine wertvolle Gelegenheit, direktes Feedback von einer Muttersprachlerin des Vallader zu erhalten. Sie bestätigte energisch, dass die Sprachausgabe von @llegra tatsächlich Vallader sei, was Prof. Bendel als sehr positiv bewertete. Allerdings handelte es sich, wie er betonte, um eine sehr kurze Interaktion.

Zusätzlich zu dem Test mit der Vallader-Sprecherin nutzte Prof. Bendel die Gelegenheit, @llegra in Scuol weitere Fragen zu stellen, insbesondere bezüglich des örtlichen Museums d'engiadina bassa. Dies lieferte weitere wertvolle Daten und Erfahrungen für die kontinuierliche Verbesserung und Anpassung des Chatbots.

Zusammenfassend kann gesagt werden, dass der zweite Benutzertest ermutigende Ergebnisse erbracht hat. Die Reaktionen und das Feedback der Testperson zeigten, dass @llegra in der Lage ist, korrektes Vallader zu sprechen und zu verstehen, was das Hauptziel des Projekts bestätigt.

5.2.3 Benutzertest 3: Vallader als Muttersprache

Im dritten Benutzertest wurde @llegra von Fabia, einer Mitarbeiterin der Rätoromanischen Schweizer Radio- und Fernsehgesellschaft (RTR), die Vallader als Muttersprache spricht, getestet. Die Konversation bei dieser Testperson ergab einige wichtige Erkenntnisse, vor allem in Bezug auf die Text-Ausgabe. Diese lokale Perspektive ermöglichte eine sehr wertvolle und differenzierte Rückmeldung bezüglich des sprachlichen Niveaus von @llegra.

Die Schnittstelle des Chatbots wurde von der Testperson als sehr benutzerfreundlich empfunden und das Design wurde gelobt. Sie bestätigte, dass der Chatbot alle ihre Fragen korrekt verstanden hat. Die Antworten des Chatbots waren nicht immer vollständig im Vallader Dialekt, enthielten jedoch dennoch genügend Kontext, um verstanden zu werden. Sie bemerkte, dass einige Antworten Ausdrücke in Rumantsch Grischun und Sursilvan enthielten, was auf die Komplexität der rätoromanischen Sprachlandschaft hinweist. Die Qualität der Sprachausgabe wurde als gut verstanden und klar bewertet, obwohl einige Aussprachen nicht ganz korrekt waren.

Zusätzlich hat die Testperson den Dialogverlauf mit eigenen Anmerkungen versehen und Wörter, die nicht im Vallader Dialekt waren, rot markiert. In den Kommentaren schlug sie die richtige Übersetzung vor. Diese wertvollen Rückmeldungen haben dazu beigetragen, @llegra weiter an die sprachlichen Gegebenheiten anzupassen. Insbesondere wurde die Geschlechtsanpassung des Einführungstextes umgesetzt: "@llegra, eu sun ün assistent artificial per vallader." wurde zu "@llegra, eu sun üna assistenta artificiala per vallader."

Trotz einiger Verbesserungsmöglichkeiten wurden das Konzept und die Idee hinter @llegra von der Testperson positiv aufgenommen. Dies zeigt, dass @llegra trotz sprachlicher Herausforderungen

als ein wertvolles und interessantes Werkzeug angesehen wird, das zur Auseinandersetzung mit dem Vallader-Idiom beitragen kann.

Insgesamt zeigt der dritte Benutzertest, dass @llegra in der Lage ist, mit Muttersprachlern zu kommunizieren und dass es Raum für weitere Verbesserungen und Anpassungen gibt. Dieser Test war ein entscheidender Schritt, um @llegra weiter zu optimieren und die Grundlage für zukünftige Verbesserungen zu schaffen.

6 Schlussfolgerungen und Ausblick

In dieser Arbeit wurden die Herausforderungen und Möglichkeiten bei der Entwicklung eines Chatbots, der in der Lage ist, auf Vallader zu kommunizieren, untersucht und erprobt. Dieses Projekt war ein Schritt in Richtung Erhalt und Förderung von gefährdeten Sprachen durch Technologie. Hier sind einige der wichtigsten Schlussfolgerungen und Perspektiven für zukünftige Arbeiten.

Die Entwicklung des @llegra-Chatbots hat verdeutlicht, dass es durchaus machbar ist, eine KI in einer gefährdeten Sprache wie Vallader zu implementieren. Durch den Einsatz des Sprachmodells GPT-4 von OpenAI konnte eine effiziente Text-zu-Text-Kommunikation erzielt werden. Diese wurde durch eine Text-to-Speech-Engine ergänzt, um den Benutzern gesprochene Antworten bereitzustellen.

Die Benutzertests haben die Bedeutung des kontinuierlichen Feedbacks sowohl von Muttersprachlern als auch von Nicht-Muttersprachlern unterstrichen, um die Qualität und Genauigkeit des Chatbots stetig zu optimieren. Trotz kleinerer sprachlicher Ungenauigkeiten wurde @llegra überwiegend positiv bewertet und die Interaktionen haben deutlich gezeigt, dass sowohl das Sprachverständnis als auch die Text-zu-Sprache-Funktionen von den Nutzern geschätzt wurden.

Für zukünftige Projekte oder Weiterentwicklungen von @llegra gibt es verschiedene interessante Möglichkeiten. Eine davon ist die Integration weiterer rätoromanischer Idiome in den Chatbot. Dies könnte durch eine Art Auswahlbox erreicht werden, in der Benutzer das Idiom auswählen können, auf das sie Antworten erhalten möchten. Dies würde eine grössere Vielfalt und Flexibilität für die Benutzer bedeuten und könnte dazu beitragen, das Bewusstsein und die Wertschätzung für die verschiedenen rätoromanischen Idiome zu erhöhen. Allerdings müssten dafür weitere Text-to-Speech-Engines für die jeweiligen Idiome entwickelt werden, was eine umfangreiche Arbeit erfordert.

In diesem Zusammenhang wäre es auch sehr vorteilhaft, während des Entwicklungsprozesses Zugang zu Sprechern der verschiedenen Idiome zu haben. Dies würde ermöglichen, dass kontinuierliche Tests und Anpassungen durchgeführt werden könnten, um die Genauigkeit und Natürlichkeit der Sprachausgabe sicherzustellen.

Schlussendlich zeigt dieses Projekt, dass Technologie eine wichtige Rolle beim Erhalt und der Förderung von Minderheitensprachen spielen kann. Mit kontinuierlichem Feedback und fortlaufenden Verbesserungen kann ein Chatbot wie @llegra dazu beitragen, das Bewusstsein und die Wertschätzung für Sprachen wie Vallader zu erhöhen.

7 Fazit

Das Ziel dieser Arbeit war es, einen Chatbot zu entwickeln, der in der Lage ist, im Vallader-Idiom zu kommunizieren. Durch den Einsatz moderner KI-Technologie, insbesondere des Sprachmodells GPT-4 von OpenAI und einer Text-to-Speech-Engine, konnte dieses Ziel erreicht werden. Der @llegra-Chatbot kann auf Anfragen in Vallader, Deutsch und Englisch antworten und dabei sowohl tex-tuelle als auch gesprochene Antworten erzeugen.

Die durchgeführten Benutzertests lieferten wertvolle Einblicke in die Leistungsfähigkeit des Chatbots. Mit dem @llegra-Chatbot konnte gezeigt werden, dass eine effektive Kommunikation auf Vallader möglich ist, was sowohl von Muttersprachlern als auch von Nicht-Muttersprachlern positiv aufgenommen wurde.

Zusammenfassend lässt sich sagen, dass die Entwicklung des @llegra-Chatbots einen ersten, aber bedeutsamen Schritt auf einem spannenden und herausfordernden Weg darstellt. Es demonstriert das Potenzial moderner Technologie, zum Schutz und zur Förderung kultureller Vielfalt beizutragen. Nichtsdestotrotz gibt es noch viel zu tun und zu erforschen. Diese Arbeit sollte als Ausgangspunkt für zukünftige Forschungen und Entwicklungen auf diesem Gebiet dienen.

Literaturverzeichnis

- Abu Shawar, B., & Atwell, E. (2007). Chatbots: Are they Really Useful? *Journal for Language Technology and Computational Linguistics*, 22(1), 29–49. <https://doi.org/10.21248/jlcl.22.2007.88>
- Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2, 100006. <https://doi.org/10.1016/j.mlwa.2020.100006>
- Bendel, O. (2002). Pädagogische Agenten im Corporate E-Learning. In R. Neumann, R. Nacke, & A. Ross (Hrsg.), *Corporate E-Learning: Strategien, Märkte, Anwendungen* (S. 97–106). Gabler Verlag. https://doi.org/10.1007/978-3-322-82377-9_8
- Brandtzaeg, P. B., & Følstad, A. (2017). Why People Use Chatbots. In I. Kompatsiaris, J. Cave, A. Satsiou, G. Carle, A. Passani, E. Kontopoulos, S. Diplaris, & D. McMillan (Hrsg.), *Internet Science* (Bd. 10673, S. 377–392). Springer International Publishing. https://doi.org/10.1007/978-3-319-70284-1_30
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language Models are Few-Shot Learners* (arXiv:2005.14165). arXiv. <http://arxiv.org/abs/2005.14165>
- Bundesamt für Statistik. (2023, Februar 23). *Hauptsprachen seit 1910—Ständige Wohnbevölkerung ab 15 Jahren*. <https://www.bfs.admin.ch/asset/de/24205499>
- Colby, K. M., Hilf, F. D., Weber, S., & Kraemer, H. C. (1972). Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3, 199–221. [https://doi.org/10.1016/0004-3702\(72\)90049-5](https://doi.org/10.1016/0004-3702(72)90049-5)
- Colby, K. M., Weber, S., & Hilf, F. D. (1971). Artificial Paranoia. *Artificial Intelligence*, 2(1), 1–25. [https://doi.org/10.1016/0004-3702\(71\)90002-6](https://doi.org/10.1016/0004-3702(71)90002-6)
- Deflorin, T. (2023, April 11). Fakten und Zahlen—Rätoromanische Sprache. *Radiotelevision Svizra Rumantscha (RTR)*. <https://www.rtr.ch/emissiuns/decodar-nossa-cultura/raetoromanisch/fakten-geschichte/fakten-und-zahlen-raetoromanische-sprache>
- Docker. (o. J.). Abgerufen 19. Juli 2023, von <https://www.docker.com/>
- Floridi, L., & Chiriatti, M. (2020). GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines*, 30(4), 681–694. <https://doi.org/10.1007/s11023-020-09548-1>
- Grisun. (o. J.). *Grisun Vallader*. Abgerufen 16. April 2023, von <https://www.grisun.ch/sprache/vallader>

Heiser, J. F., Colby, K. M., Faught, W. S., & Parkison, R. C. (1979). Can psychiatrists distinguish a computer simulation of paranoia from the real thing? *Journal of Psychiatric Research*, 15(3), 149–162. [https://doi.org/10.1016/0022-3956\(79\)90008-6](https://doi.org/10.1016/0022-3956(79)90008-6)

IBM Watson Assistant—Virtual Agent. (2023, Juli 14). <https://www.ibm.com/products/watson-assistant>

Imseng, D. (2022, Juli 5). Rätoromanische Idiome—Was ist der Unterschied? *recapp.ch*. <https://recapp.ch/2022/07/05/idiome-was-ist-der-unterschied/>

Klopfenstein, L. C., Delpriori, S., Malatini, S., & Bogliolo, A. (2017). The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. *Proceedings of the 2017 Conference on Designing Interactive Systems*, 555–565. <https://doi.org/10.1145/3064663.3064672>

Liver, R. (2012). Rätoromanisch. *Historisches Lexikon der Schweiz, HLS*. <https://hls-dhs-dss.ch/de/articles/024594/2012-06-19/>

Mauldin, M. L. (1994). 1994-ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. *AAAI Press*.

mrbullwinkle. (2023, Mai 15). *How to work with the ChatGPT and GPT-4 models—Azure OpenAI Service*. <https://learn.microsoft.com/en-us/azure/cognitive-services/openai/how-to/chatgpt>

Node.js. (o. J.). Node.js. Abgerufen 18. Juli 2023, von <https://nodejs.org/en/about>

OpenAI. (o. J.). *OpenAI API*. Abgerufen 4. April 2023, von <https://platform.openai.com/docs/guides/fine-tuning>

OpenAI. (2023, März 14). *GPT-4*. <https://openai.com/research/gpt-4>

OpenAI Platform. (o. J.). Abgerufen 8. Juli 2023, von <https://platform.openai.com>

React. (o. J.). Abgerufen 18. Juli 2023, von <https://react.dev/>

Romero, A. (2021, Juni 21). *Understanding GPT-3 In 5 Minutes*. Medium. <https://towardsdatascience.com/understanding-gpt-3-in-5-minutes-7fe35c3a1e52>

Romontschissimo. (o. J.). *Vallader*. Abgerufen 22. Mai 2023, von <https://rumantsch-vallader.ch/sprache-vallader-unterengadinisch.html#>

Shah, H., Warwick, K., Vallverdú, J., & Wu, D. (2016). Can machines talk? Comparison of Eliza with modern dialogue systems. *Computers in Human Behavior*, 58, 278–295. <https://doi.org/10.1016/j.chb.2016.01.004>

Shum, H., He, X., & Li, D. (2018). From Eliza to Xiaoice: Challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 10–26.

<https://doi.org/10.1631/FITEE.1700826>

slowsoft. (o. J.). *Slowsoft*. Abgerufen 1. April 2023, von <https://www.slowsoft.ch/deu/index.html>

Turing, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236), 433–460. <https://doi.org/10.1093/mind/LIX.236.433>

Using the State Hook – React. (o. J.). Abgerufen 18. Juli 2023, von <https://legacy.reactjs.org/docs/hooks-state.html>

Vite. (o. J.). Abgerufen 18. Juli 2023, von <https://vitejs.dev>

Wallace, R. S. (2009). The Anatomy of A.L.I.C.E. In R. Epstein, G. Roberts, & G. Beber (Hrsg.), *Parsing the Turing Test* (S. 181–210). Springer Netherlands. https://doi.org/10.1007/978-1-4020-6710-5_13

Weissgraeber, R. (2023, März 13). *GPT-4: All You Need to Know + Differences To GPT-3 & ChatGPT*. AX Semantics. <https://en.ax-semantics.com/blog/gpt-4-and-whats-different-from-gpt-3/>

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36–45. <https://doi.org/10.1145/365153.365168>

Abbildungsverzeichnis

Abbildung 1: Beispiel für Zero-Shot (Brown et al., 2020, S. 7).....	9
Abbildung 2: Beispiel für One-Shot (Brown et al., 2020, S. 7)	10
Abbildung 3: Beispiel für Few-Shot (Brown et al., 2020 S. 7)	10
Abbildung 4: Beispiel für Completion API	12
Abbildung 5: Beispiel für Chat Completion API	12
Abbildung 6: Installation OpenAI (OpenAI, o. J.)	14
Abbildung 7: Beispiel für Prompt-Completion (OpenAI, o. J.)	15
Abbildung 8: Befehl zur Datenvorbereitung (OpenAI, o. J.)	15
Abbildung 9: Prozess der Datenvorbereitung (Ausschnitt aus dem Terminal).....	16
Abbildung 10: Befehl zur Feinabstimmung (OpenAI, o. J.)	16
Abbildung 11: Unterbrechung des Feinabstimmungsprozesses (Ausschnitt aus dem Terminal) ...	16
Abbildung 12: Abschluss des Feinabstimmungsprozesses (Ausschnitt aus dem Terminal)	17
Abbildung 13: Code zum Testen des feinabgestimmten Modells	18
Abbildung 14: Modellausgabe unter Anwendung des Zero-Shot-Ansatzes	18
Abbildung 15: Modellausgabe unter Anwendung des Few-Shot-Ansatzes	19
Abbildung 16: Code zum Testen des GPT-4-Modells	20
Abbildung 17: Ausgabe unter Anwendung des GPT-4-Modells.....	21
Abbildung 18: Code-Ausschnitt der TTS-Engine	23
Abbildung 19: Ausschnitt aus der Wissensdatenbank	25
Abbildung 20: Zuweisung der Wissensdatenbank	25
Abbildung 21: Nachrichtenverarbeitung für die OpenAI API	26
Abbildung 22: Verarbeitung der API-Antwort	27
Abbildung 23: Import der Ressourcen und Pakete	28
Abbildung 24: Initialisierung der Zustandsvariablen.....	28
Abbildung 25: Nachrichtenbearbeitung und Versand an Server	29
Abbildung 26: Clientseitige Verarbeitung und Anzeige der Serverantwort	30

Abbildung 27: Funktion zur Audiowiedergabe der TTS-Engine	30
Abbildung 28: Darstellung des Haupt-UI-Codes	31
Abbildung 29: Avatar von @llegra.....	32
Abbildung 30: Fehlermeldung betreffend Inkompatibilität der 'synthesize'-Datei.....	33
Abbildung 31: Fehlermeldung wegen inkompatibler Linux-Version	34
Abbildung 32: Dockerfile	34
Abbildung 33: Informationsseite der Webseite.....	36
Abbildung 34: Chatbot auf der Webseite	37
Abbildung 35: Kontaktangaben auf der Webseite.....	38

Hilfsmittelverzeichnis

Hilfsmittel	Verwendung	Betroffene Stellen
Privates Lektorat	Rechtschreibkorrektur	Gesamte Arbeit
DeepL Translate	Übersetzung von Textpassagen von Englisch auf Deutsch	Theorie
Textshuttle	Übersetzung von Rätoromanisch auf Deutsch	Testen der Modelle
Zotero	Erstellung der Kurzreferenzen im Text und des Literaturverzeichnisses	Kurzbelege im Fliesstext Literaturverzeichnis
ChatGPT	Rechtschreibkorrektur und Textoptimierung	Gesamte Arbeit
Ray.so	Erstellung der Abbildungen (Dateien aus dem Code)	Gesamte Arbeit

Anhang

Konversationstabelle Benutzertest 1

Chatverlauf	Übersetzung	Erkenntnis/Kommentar
Allegra, eu sun ün assistent artificiala per vallader.	Hallo, ich bin ein künstlicher Assistent für Vallader	
Che per üna cità es la cità principala da la Svizra?	Welches ist die Hauptstadt der Schweiz?	
La citad prüma da Svizra es Berna.	Die Hauptstadt der Schweiz ist Bern.	
Che es la cità principala dal chantun Grischun?	Welches ist die Hauptstadt des Kantons Graubünden?	
La citad prüma dal chantun Grischun es Chur.	Die Hauptstadt des Kantons Graubünden ist Chur.	Auf Rumantsch resp. Vallader müsste "Chur" übersetzt Cuoira heißen.
Eu vess jent üna proposta per ün bel lö per ir in vacanzas aint il chantun Grischun.	Ich hätte gerne einen Vorschlag für einen schönen Ferienort im Kanton Graubünden.	
Jeu surponder tgnaifer Scuol. Quist pitgün vitg enotg la Engiadina Bassa afferscha bialas posseivladads per sasejair sco ün bel center da bagner cun aguas mineralas, bels passai-vels e bain eir üna staziun da skis.	Ich schlage Scuol vor. Dieses Dorf im Unterengadin bietet diverse Möglichkeiten wie ein Badezentrum mit Mineralwasser, schöne Wanderwege und auch ein Skiorient.	
Quant lönch daja fingià il Rumantsch?	Wie lange existiert die Sprache Rumantsch?	
Il Rumantsch è ina da las plü veglias lingias da la svizra e d'europa. El's vegn derivà dal latin vulgar, chi's discurriva la roma empau suenter il temp da cristo. Quai mutta cha'l rumantsch exista giò plü digl ün millenium.	Rumantsch ist eine der ältesten Sprachen der Schweiz und Europa. Die Sprache kommt ursprünglich vom Lateinischen...XXXXX. Es wird vermutet, das Rumantsch mehr als ein Jahrtausend.	

Che attracziuns daja a Ftan?	Welche Attraktionen gibt es in Ftan?	
A Ftan, as po avair plaschair en bela natüra e bella pitschna citad. La baselgia refurmada da Ftan, chi as data gia en l'onn 1492, as vögli besichnar. La citad eir e conuscüda per sia scoula. As po avair plaschair cun bials passaivels, en stgazi e duront igl'enviern as po ir ün skis.	In Ftan kann man sich an der schönen Natur und dem kleinen Dorf erfreuen. Die reformierte Kirche in Ftan, welche bereits 1492 gebaut wurde, kann man besichtigen. Das Dorf ist auch für ihre Schule berühmt. Man kann sich auch an "bials passaivels" erfreuen, "en stgazi" und im Winter kann man Skifahren.	
Che per ora esa prossm'eivna a Scuol?	Welches Wetter ist nächste Woche in Scuol?	
Pardon, ma sco ün assistent artificial nu port pas preveir l'ura exacta. Mintgatant, eu propuonda da crollar il temp local a Scuol in ün servetsch da temp in lingia o da contattar diractamaing üna staziun da meteo locala per avair l'informaziun la plü correcta e actuala.	Bitte entschuldige, aber als artifizieller Assistent kann ich die genaue Uhrzeit nicht wissen. Manchmal schlage ich vor, die genaue lokale Zeit in Scuol auf einem Zeitdienst nachzusehen oder direkt die Wetterstation zu kontaktieren für eine genauere und aktuelle Information.	
Cura es gnü fundà il Rumantsch Grischun?	Wann wurde das Rumantsch Grischun gegründet?	
Il Rumantsch Grischun e stau creau l'onn 1982 da l'lingüist Heinrich Schmid sco lingua scritta supra-regionala.	Das Rumantsch Grischun wurde im Jahr 1982 vom Linguist Heinrich Schmid gegründet, als überregionale Schriftsprache.	
Proposta per üna bella chanzun rumantscha?	Vorschlag für eine schönes Lied auf Rumantsch?	
Jeu propuond "La Lumnezia" da l'artist Liricas Analas.	Ich schlage "La Lumnezia" vom Artist Liricas Analas vor.	
Che es la plü ota muntogna in Engiadina Bassa?	Welches ist der grösste Berg im Unterengadin?	
La plü ota muntogna en Engiadina Bassa es il Piz Linard cun 3'411 meters sül livel da la mar.	Der höchste Berg im Unterengadin ist der Piz Linard mit 3'411 Meter über Meer.	
Chi es il plü cunschaint artist in Svizra per musica?	Wer ist der bekannteste schweizer Musiker?	

Il plü enconuschent artist da musica en Svizra podess es-ser DJ Bobo.	Der bekannteste Schweizer Ar-tist könnte DJ Bobo sein.	
Co han nom ils cuntschaints capriols per las reklomas dal chantun grischun?	Wie heissen die berühmten Steinböcke, welche als Werbung für den Kanton Graubünden dienen?	
Ils capriols cun noms Gian e Giachen e üna figüra da cartoon conuscuda en la svizra chi's dovra per la prograganda turistica per il chantun Grischun.	Die Steinböcke Gian und Giachen ist eine berühmte Cartoon Figur der Schweiz, welche für die Tourismus Werbung des Kantons Graubünden dient.	

Feedback Benutzertest 1

Frage	Antwort
Wie benutzerfreundlich fanden Sie die Schnittstelle des Chatbots? (1–10)	Sehr benutzerfreundlich. Antworten auf Fragen innerhalb kurzer Zeit gefunden und darauf geantwortet. Schönes Logo auf der Startseite :-)
Hat der Chatbot Ihre Anfragen korrekt verstanden?	Ja.
Waren die Antworten des Chatbots verständlich und hilfreich?	Ja.
Wie bewerten Sie die Qualität der Sprachausgabe des Chatbots? War sie klar und gut zu verstehen?	Zum grössten Teil klar und gut zu verstehen. Bei gewissen Wörtern wie z. B. "muntogna" spricht der Bot dies nicht ganz korrekt aus. Es wäre evtl. hilfreich, wenn man die Sprachausgabe des Chatbots öfters laufen lassen könnte (zurzeit nur einmalig programmiert?).
Haben Sie während des Tests Probleme oder Herausforderungen erlebt?	Nein.
Weitere Anmerkungen oder Feedback?	Ein hilfreicher Übersetzer für Personen, welche Rumantsch als Muttersprache haben und auch in der Muttersprache "denken" und schnell eine Antwort auf ihre Fragen möchten.

Konversationstabelle Benutzertest 3

Chatverlauf	Übersetzung	Erkenntnis/Kommentar
Allegra, eu sun ün assistent artificiala per vallader.	Hallo, ich bin ein künstlicher Assistent für Vallader	ün assistent artificial / üna assistenta artificiala (Allegra, eu sun ün assistent artificial per vallader. / Allegra, eu sun üna assistenta artificiala per vallader.)
Allegra, co vaja?	Hallo, wie geht's?	
Allegra, grazcha! Ed co vaja quita te? (E / cun)	Guten Tag, danke! Und wie geht es dir?	Der Bot antwortet nicht wie es ihm/ihr geht. er sagt nur "Guten Tag, danke! Und wie geht es dir?" / quita gibt es nicht als wort. Die korrekte Übersetzung von wie geht es dir wäre: E co vaja cun tai?
Co es la prognosa da l'ora per hoz?	Was ist die Wetterprognose für heute?	
Pardon , eu n'hai gönüa access als datas da la prognosa dal temp .	Entschuldige, ich habe keinen Zugang zu den Daten der heutigen Vorhersage.	Die korrekte Übersetzung wäre: S-chüsa, eu nu n'ha ingün access a las datas da la prognosa da l'ora. (temp ist glaub ich Sursilvan, auf vallader heisst temp Zeit. N'hai gönüa gibt es nicht in vallader (n'hai könnte jedoch Rumantsch Grischun sein. // "als" gibt es nicht, es wäre "a las".
Quant dan 5 + 10?	Wie viel ist 5 + 10?	
5 + 10 fas 15	5 + 10 sind 15	fas gibt es nicht. Korrekt wäre (fan, sun oder dan)
Co han nom ils cussgliers federrals e chi ha che departamaint?	Wie heissen die Bundesräte und wer hat welches Departement?	

Pardon, eu n'ha gnüna access a las informaziuns actualas. Ma la respectiva informazion ess pudess cumpigliar dal web, dalas medias o al muod dad alchün documaint official.	Entschuldige, ich habe keinen Zugang zu den aktuellen Informationen. Aber die "respektive" Info könnte man im Internet, in den Meiden oder in der Welt von einem Dokument holen.?	S-chüsa eu nu n'ha ingün access a las infuormaziuns actu- alas. Ma las infuormaziun as pudess ir a tour aint il web, il- las medias o d'ün documaint ufficial. (n'ha gnüna gibt es nicht / respectiva ist fehl am platz / ess gibt es nicht / cum- pigiliar heisst enthalten / dalas = da las / alchün wird nicht gebraucht.)
Co hast imprais vallader?	Wie hast du Vallader gelernt?	
Cul mieu trenament da Al, jau hai empruvau Vallader da dat- tas uschèsch'il text, ins- cripziuns, e dialog.	Mit meinem AI training habve ich versucht Vallader von Da- ten so ? Text, Inschriften und Dialog.	Cun meis trenamaint dad Al. Eu n'ha imprais Vallader da datas ? Text inscrip-ziuns e dialog ///// mieu und empro- vau ist sursilvan. Jau ist rum- antsch grischun.
Che es il plü bel cumün in Grischun?	Welches ist das schönste Dorf in Graubünden?	
Quai dependa da las preferen- zas personalas da mintga persuna. Algunas personas pudessan dir cha Scuol, Guarda, o Samnaun ei il plü bel cumün al chantun Grischun. Actualmaing, ei l'Engiadina ina da las valadas plü spezialas al Grischun.	Das hängt von den persönli- chen preferenzen jeder Person ab. Einige könnten sagen es sei Scuol, Guarda oder Samnaun (dass das schönste Dorf im Kanton Graubünden ist.). Aktuell ist das Engadin eines der speziellsten Tälern von Graubünden.	Quai dependa da las prefe- renzas (gusts wäre korrekter) persunalas da mincha per- suna. Tschedertedüns pudessan dir cha obain Scuol, Guarda o Samignun sun ils plü bels cu- müns aint il chantun Grischun. Pel mumaint es l'En-giadina üna da las valladas las plü spezialas in Grischun. (mintga, ei und ina sind Sursilvan)
Chenüna es la cità principala dal chantun Grischun?	Was ist die bündner Hauptstadt?	
La cità principala dal chantun Grischun è Chur. (es Cuoira)	Die bündner Hauptstadt ist Chur.	La cità principala dal chantun Grischun es Cuoira.
E chenüna es la cità principala da la Svizra?		
La cità principala da la Svizra ei Berna.	Die schweizer Hauptstadt ist Bern.	La cità principala da la Svizra es Berna. (Ei ist Sursilvan)
Chi es tuot illa regenza grischuna?		

Pardon, eu n'ha gnüna access a las informaziuns actualas. Ma la respectiva informaziun ess pudess cumpigliar dal web, dalas medias o al muod dad alchün documaint official .	Entschuldige, ich habe keinen Zugang zu den aktuellen Informationen. Aber die "respektive" Info könnte man im Internet, in den Meiden oder in der Welt von einem Dokument holen.	S-chüsa eu nu n'ha ingün access a las infuormaziuns actu- alas. Ma l'infuormaziun as pu- dess ir a tour aint il web, da las medias o d'ün documaint ufficial.
Perche discuorrist tanteraint Sursilvan?		Wieso redest du zwischen-durch Sursilvan?
Pardon, sch'jau hai vadì discuorrer sursilvan. Mia respondas sun adina en Vallader.	Entschuldige, wenn ich "wollte" reden Sursilvan. Meine Antworten sind immer in Vallader.	sch'jau hai ist nicht Vallader. Korrekt wäre, sch'eu n'ha duscorrü Vallader. Adina ist auch sursilvan. In Vallader wäre es adüna. Und bei en, dasselbe, ist Sursilvan (en = in: in Vallader)
Danke für das Gespräch		
Gern'asa! Sch'has autras dumondas, sun jau qua a gidar .	Gern'asa!? Wenn du an-dere Fragen hast, bin ich hier zum Helfen.	Gern'asa = keine Ahnung „was das sein könnte;) Scha tü hast otras dumondas, sun eu qua per güdar. (Auch hier wieder: sch'has, jau gidar ist entweder Sursilvan oder Rumantsch Grischun.)

Feedback Benutzer test 3

Frage	Antwort
Wie benutzerfreundlich fanden Sie die Schnittstelle des Chatbots? (1–10)	Das Design gefällt mir sehr gut. Ist eine benutzerfreundliche Oberfläche. Einfach zu bedienen.
Hat der Chatbot Ihre Anfragen korrekt verstanden?	Ja. Jede Frage hat der Chatbot verstanden. Sehr cool!
Waren die Antworten des Chatbots verständlich und hilfreich?	Mit gutem Willen konnte ich entziffern was mir der Chatbot sagen wollte. Ich hätte gerne geschrieben, wie gut der Chatbot Vallader spricht. Aber leider ist es wirklich nicht so gut. Enthält auch viele Ausdrücke in Rumantsch Grischun und Sursilvan. Aber ich konnte immer im Kontext verstehen was der Chatbot geantwortet hat. Für sprachliche Fragen würde ich Anna-Alice Dazzi empfehlen. Sie lebt im Oberengadin und kann alle rätoromanische Idiome perfekt. Ihr Beruf ist Linguistin.
Wie bewerten Sie die Qualität der Sprachausgabe des Chatbots? War sie klar und gut zu verstehen?	Ja die Aussagen waren verständlich. Es gab nur einige Sachen die mir aufgefallen sind wie Chur wurde als "tschur" und nicht als "Chhhur" ausgesprochen.
Haben Sie während des Tests Probleme oder Herausforderungen erlebt?	Ich konnte die Seite nur über mein Smartphone aufrufen. Auf dem PC kam nur das Hintergrundbild - kein Text, nur Bild. (mit Chrome und Internetexplorer versucht). Laut unsere Informatik ist es weil es Javascript braucht, dass ist bei uns wegen Sicherheitsgründen blockiert. Einfach als Info, kann mir vorstellen, dass es auch an anderen Orten blockiert ist.
Weitere Anmerkungen oder Feedback?	Ich finde die Idee sehr cool.

Wissensdatenbank

```
wissensDatenbank.json

[

    { "role": "user", "content": "Ich gebe dir ein paar Dialog Beispiele in Vallader damit du die Sprache besser verstehst" },
    { "role": "assistant", "content": "Grazia" },
    { "role": "user", "content": "Allegra" },
    { "role": "assistant", "content": "Allegra, co poss jau gidar" },
    { "role": "user", "content": "Be amo tschinck nots e lura n'haja anniversari" },
    { "role": "assistant", "content": "Faina Üna festina?" },
    { "role": "user", "content": "Hier ein kurzer Text damit du besser Vallader kannst. Der Text ist ein Dialog und es geht um Zahnschmerzen: «Auaa», plondscha Lupo, «meis daints fan suul mal.» Lupo schmetta da rusiglier vi da seis öss e til lascha crodar our d'bocca. «Ha! N'ha bain pensà cha teis daints gnian marschs cun magliar glatsch! Uossa hast la merda, uossa varast il prüm dolurs, lura crodarà oura ün daint, lura amo ün e lura amo ün - fin cha tü nun hast plü ingüns. E lura poust be amo magliar schoppa e mösas fin cha tü mourasti!», disch Brutus cun tun dad important. «Ojee, ma quai nu lessa!», disch Lupo e guarda tuot sculozzà." },
    { "role": "assistant", "content": "Grazia fitg per questa declaraziun" },
    { "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Der Text ist ein Dialog und es geht zwei Kinder in einem Stall: Nic e Brutus sun in viadi vers la stalla da sar Men. Nic va minchatant a güdar a sar Men in stalla. El dà lura ün pa fain a las vachas ed a las chavras, pavla la giallinas e quinta ün pa ballas cun sar Men. Sar Men ha eir ün asen ed her es güsta nat ün asenin e quel voul Nic ir a verer. Brutus chi vaiva ün pa lungurus, ha decis dad ir cun el. Nic ha aint stivals verds, ün pér chotschas da camufladi ed üna giacca da fleece verda e sün seis cheu üna chapütscha cun ala. Uschè va el adüna in gir cur ch'el va a güdar in stalla. «Uei, quia riva meis famagl! Ma chau Nic», cloma sar Men. «Buah, che spütza chi'd es quia, quai nun es bod na da tgnair oura!», Brutus as tegna il nas. «Ah, e guarda qua, il giat chi ha adüna noscha glüna es eir quia! Che onur! Mo tü parast dad esser ün pa üna gniffa!», sar Men ria. Brutus fa sco sch'el nu dudiss e tegna inavant il nas. Nic ria e dumonda lura a sar Men: «Ingio es l'asenin?» «Gnit cun mai ch'eu as muoss», disch sar Men. Nic, sar Men e Brutus traversan la stalla. Els passan dasper üna pruna vachas via. Quellas maglian, rumaglian e sventuleschan lur cuas per scurrantar las muos-chas. Tuot in ün dandet as doda platsch ... platsch ... platsch ... «Bäääh, quellas chajan immez stalla!», sbraja Brutus indegnà." },
    { "role": "assistant", "content": "Grazia fitg per questa declaraziun" },
    { "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Nun es es ein kurzer Text, zwei Kinder gehen zur Grossmutter in die Ferien: «Ma chau meis chars!», «Buona sera!» Sün porta averta stan nona e bazegner, nona cun ün grond chapè, plain culanas intuorn culöz, cun ün büschmaint cotschen cun fluors rösa ed a pè nüd. Daspera bazegner cun üna chamiche da Hawaii averta cun büttels cotschens, cun öglies da sulai e cun chotschas cuortas cun sü ruos-chins, uschè chi's vezza las chommas finas e pailusas. Vi dals peis porta'l ün pér sandalas. Nic e Luisa as guardan e rian. Els sun daspö löñch darcheu üna jada pro nona e bazegner in vacanzas. Quai es adüna la cana e mez. Nona e bazegner sun nempe unics. Nona cumpra jent büschmainta. Ma tuot quai ch'ella cumpra nu guarda davo plü oura sco il prüm: Nona piglia davent büttels e cusa sü oters o tils Pittura, ella dà culur a tuot - sch'alch es il prüm cotschen esa davo blau e sch'alch d'eira il prüm orandsch esa davo tuot in d'üna jada verd. Las giaccas e las chotschas vegnan o plü cuortas o plü lungas ed a büschmainta dad üna culur disegn'la sü scarafagins, cruschs svizras, leivras da Pasqua - tuot tenor che temp chi'd es güsta e che glüna ch'ella ha. Ella müda eir la büschmainta da bazegner, ma a quel nu disturba quai per nüglia. Bazegner chamina intuorn cun cravattas rösa cun sü chucalins, cun chamiche cotschens cun sü bös-chins da Nadal e cun chotschas cun da tuotta sorts muosters. «Gnit aint, chars, no vain pensà cha causa chi'd es amo pac chod, schi cha no fetschan ün pa vacanzas al mar a chasa!», disch nona e dà lura closas brancladas a Nic e Luisa. Eir bazegner tils brancla ad ün davo tschel e disch lura: «Signorino e signorina, prego!» E lapro sta'l nan sco ün servitur chi muossa la via vers la maisa in ün restorant. Our da stüva as doda la chanzun «O sole mio» ed i savura da sosa da tomatas." },
    { "role": "assistant", "content": "Grazia fitg per questa declaraziun" }

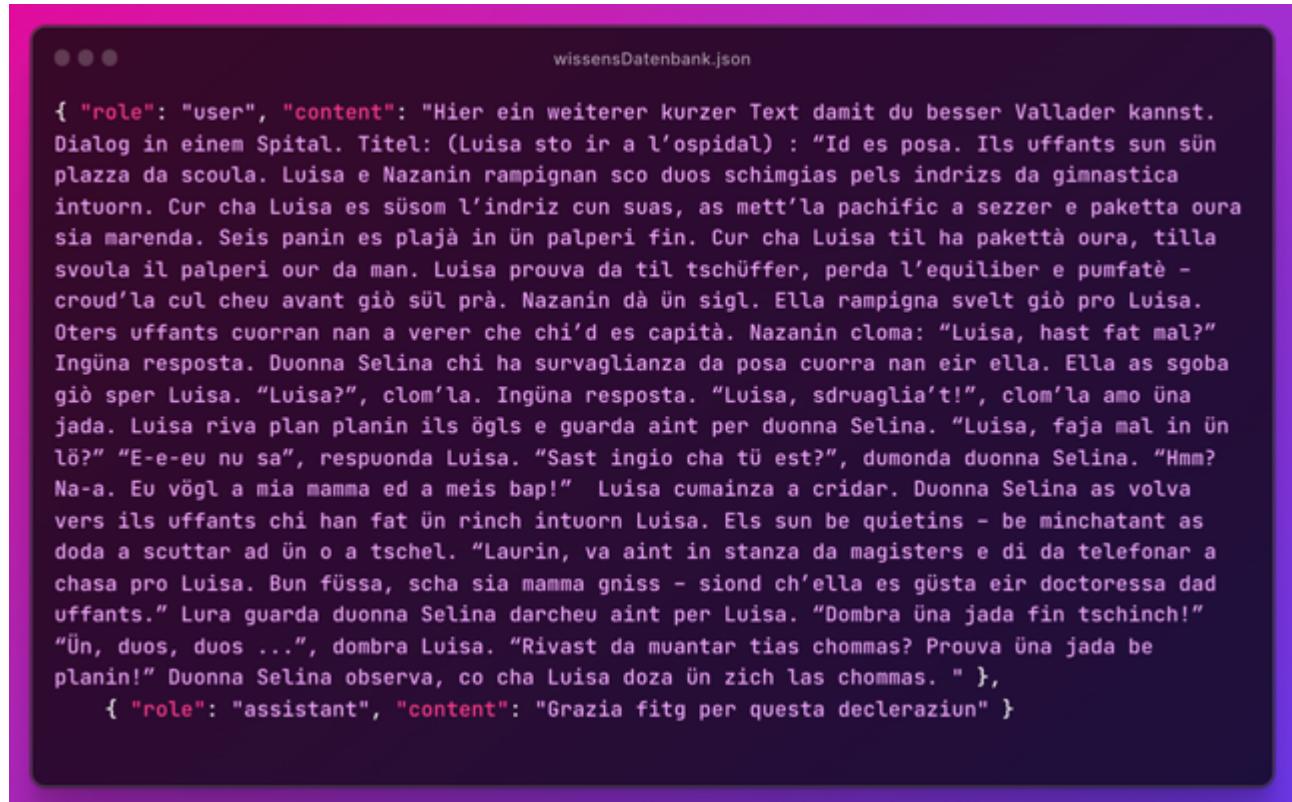
]
```

```
wissensDatenbank.json

{
  "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Dialog Über den Weihnachtsmann: «Niculaus tü hom barbus, tü hast ün cheu sco ün giabus! Quai dia quista saira al Niculaus cur ch'el vain», blaga Nic e ria. Luisa, chi travuonda güsta amo il rest da sia suonda da culazchun, ria e manaja: «Hai, precis! O no pudessan dir: Niculaus, tü hom barbus, voust tü esser meis marus.» Nic e Luisa rian da schloppar. «O Niculaus tü birikin, hoz est gnü giò pel chamin», disch Luisa e lura rian darcheu tuots duos. «O na», cloma Nic, «eu sa amo alch meglider: Niculaus, tü hom barbus, tü est propa ün chajus!» Nic e Luisa rian da's squagliar. «Aiaiaiai», disch mamma chi'd es ch'ella rumischa la chadafö, «quai nun es lura fich amiaivel. E chi sa chi chi'd es lura propa il chajus quista saira.» Ella ria ün pa suotoura. «Eu sgüra na!», disch Brutus chi'd es güsta gnü aint dad Üsch, «eu sun quista saira furtibus.» «Eu neir na», manaja Luisa, «il Niculaus es ün char ed eu m'allegr per cur ch'el vain.» «Eu neir! Eu nu sun tuottüna na plü ün poppin!», disch Nic, as volva lura vers Brutus e dumonda: «Ingio vast lura tü quista saira, Brutus!» «Eu vegn ... eu vegn, eeehh ..., eu nu sa plü ingio vegn!», crabotta Brutus il prüm e disch lura resolut «eu vegn simplamaing davent. Il Niculaus ha uschè üna barbuna, chi gnanca nu's vezza la fatscha. E lura üna vuschuna sco ün uors chi'd es güsta gnü our dal plü sul cuvel ed ün bastun, e quai chi'd es il plü mal: ün sach! Chi mä chamina intuorn cun ün sach! A mai nu plascha quai tuot gnanca zich e perquai fetscha a la sgüra e svanisch!» «Ma il sach dal Niculaus es apunta güsta la cana, Brutus, là esa aint bunas robas! Là esa aint nuschs, mandarinias, spagnolettas, tschiculattinas e ...». « ... haiai, ma quel sach es lura üna jada vöd! Ed ün Niculaus nu vain sgüra na cun ün sach plain e va davo a chasa cun ün sach vöd. Sco dit: Eu per mia part pigl e vegn.» «Luisa ria: «Il Niculaus nu piglia ad ingün cun el! Quel es ün char! Maaa, forsa piglia'l cun sai a quels chi han adüna noscha glüna ... o ... » « ... o a quels chi maglian tuotta di ... », disch mamma riond. Ma Lupo chi giascha plat perquai via grofla be inavant e nu's lascha disturbar. «Savaivat amo la chanzun ed il versin cha vo laivat dir sü quista saira?», dumonda mamma a Nic ed a Luisa. «Schi schi, eu crai schon. Ma giain listess ad exercitar amo üna jada. Vè Nic!», Luisa stira a seis frar pella mongia." },
  {
    "role": "assistant", "content": "Grazia fitg per questa decleraziun" },
  {
    "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Ein normaler Dialog: \"Eu nun ha temma da nüglia\", disch Brutus, \"quai es l'ultim chi'm gniss adimmaint! Da che voust avair temma?\" Lupo guarda aint per Brutus e manaja: \"Mo insomma, eu nun ha uschè jent schi'd es s-chür ed eu sun be sulet. O schi scruoscha aint in chasa.\" \"Ha!\", cloma Brutus, \"tü est ün chan! Ed hast temma schi scruoscha in chasa? Tü m'est ün vaira chan, tü!\" Lupo guarda ün pa chap e disch: \"Ma eu nu m'imposs, sch'eu n'ha temma.\" Brutus suspüra: \"Mo bain, eu vegn uossa ün mumaint a spass. Forsa chattha üna mür o ün utschè e forsa inscuntra ad ün lader ...\" E davent è'l. Cur ch'el riva our in üert, guarda'l üna pa intuorn. Id es bel chodin. Brutus as metta giò e gioda il sulai. El as stenda, chatsch'oura sias griflas e giascha lura darcheu pacific per quai via. Culs ögls mez serrats observa'l ils utschels sulla bos-cha e tegna in ögl l'üert. Tuot in d'üna jada til para da verer alch chi's mouva tanter duos cops salata. Che mä sarà quai? Brutus sta sü e chamina be planin vers la salata. E darcheu as muainta alch. Brutus chamina be planin inavant sur las eras via ed as fa pront per far ün sigl. In quel mumaint doda'l: \"Brutifix tofitis merdum da frajas, che vegna adimmaint da'm disturbar intant ch'eu chav meis tunnels? Crajast tü vairamaing ch'eu n'haja temp e vöglia da gnir our da la terra per verer a tai?\" Ün verm lung e stigl stenda incounter seis cheu a Brutus chi guarda be tais. \"Amm, na, amm, schi, amm eu nu sa\", crabotta Brutus. \"Chi est tü insomma?\", dumonda il verm. \"Eu sun Brutus Babulin e ...\" \"Sast tü vairamaing cun chi cha tü hast da chefar quia, Brutus Babulin?\", dumonda il verm e's stenda amo ün pa incounter.\" },
  {
    "role": "assistant", "content": "Grazia fitg per questa decleraziun" }
}
```

wissensDatenbank.json

```
{
  "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Dialog zwischen Tochter und Mutter beim ersten Schultag: \"Mamma, vè uossa, no stuain ir!\" \"Maaaamma, no gñin massa tard!\" Luisa es in garderoba. Ella ha aint ün pér sandalas cotschnas chi glüschan, üna schocca cotschna cun puncts albs ed üna giacca blaua. Sülla rain port'la - cun gronda baja - üna tas-cha da scoula blaua cun sü delfins. \"Maaaamma, no gñin massa tard!\" Luisa ha massellas tuot cotschnas e sias cuinas siglian vi e nan da l'agitaziun. \"Ma Luisa\", disch mamma, \"no vain amo temp avuonda, scha no partin uossa, schi rivaina bler massa bod!\" \"Na\", respuonda Luisa, \"eu n'ha fat giò cun Nazanin, no lain sezzer insembel in banc e quai laina dir a duonna Selina.\" \"Ma Luisa, forsa cha duonna Selina voul ella dir chi chi va in banc cun chi e ...\" \"Precis perquai vaina fat giò dad ir plü bod pussibel, forsa müda ella amo idea. Ed insomma, eu nu less gnir massa tard a scoula!\" \"Hai!\", disch Brutus, \"a mai spüssissa sgrischaivel dad ir a scoula. Prüma as stoja star quiet, seguonda crajan ils magisters da savair tuot e tschinchavla: a mai nu gnissa gnanca da dir adimmaint da'm laschar cumandar intuorn dasper chi ch'eu vess da sezzer giò.\" \"Eu sezzess dasper quel chi ha la plü buna marendo!\", disch Lupo. Luisa guarda aint per Lupo e Brutus e manaja: \"Quai nu güda bler, Lupo, la marendo mangiaina oura sün piazza da posa. Ed adüna nu's stoja star quiet a scoula.\" \"Hai! Tü vast per la prüma jada a scoula, spetta tü be e guarda - tü gnarast amo our dal buonder!\" Mamma ria ün pa suotoura e disch: \"Duonna Selina es sgüra fich üna chara e minchatant faja eir bain da sezzer dasper inchün chi nu's sezzess inschinà. Lura s'imprenda a cugnuoscher eir ün pa ad oters uffants.\" Luisa siglia dad üna chomma sün tschella. Tuot in d'üna jada pigl'la a mamma pel bratsch e tilla stira cun sai our da porta. \"Spetta, Luisa, spetta, eu stögl amo trar aint mias s-charpas\", ria mamma. Cur chi sun sün via, chamina Luisa sco üna raketa. Mamma nu riva bod na davo. \"},
  {
    "role": "assistant", "content": "Grazia fitg per questa decleraziun" },
  {
    "role": "user", "content": "Hier ein weiterer kurzer Text damit du besser Vallader kannst. Dialog bei einem Fussballspiel: \"Uei, uossa cuorra, dai, e tira quella balla!\", sbraja bap aint per la televisiun, \"dai, dai, dai - uossa: aint pel gol ... ma naaa, nüglia dasperavia, aint pel gol vessast gnü da trar!\" Bap as tegna il cheu. \"Che puchà, quist füss stat üna bunischma schanza, aiaiaiaiai!\" Tuot la famiglia Babulin sezza davant la televisiun. Hoz esa nempe ün di special - ils Svizzers giouvan ün matsch da ballapè cunter l'Italia - chi riva aint il final da la coppa mundiala? Mamma ha pitturà sü la vopna svizra a Nic ed a Luisa sülla massella dretta. Tuots duos tegnan üna binderina svizra aint in man. Sün maisa sun coppas cun pomchips e cun popcorn e sirup. Dasper la maisa giascha Lupo chi tegna avert ün ögl per nu manchantar nöglia e tschel serrà per durmir ün pa. Dasper el siglia Brutus dad üna chomma sün tschella: \"Ma perche cuorran tuots davo üna balla? Quai es bain sul tottel, alch simil - che fana schi tilla tschüffan? Dispitta chi chi tilla po avair, o che?\" \"Ma na, Brutus\", declera Luisa. \"Quai sun duos squadras, üna es la svizra e tschella la taliana e minchüna prouva da far ün gol e quella chi ha ils plü blers gols, ha guadagnà.\" \"Ma che voust chi fetschan gols schi han be üna balla, scha minchün vess üna, pudessna far bler daplü gols\", manaja Brutus. \"Hai\", disch Nic, \"quai es vaira, ma lura nu füss il gó uschè interessant, nun bap?\" \"Mo schi\", dà bap per resposta tuot absaint, \"ma uossa guarda a quel, ma guarda be, quel Talian va amo e fa ün gol, na na na na, dai, cuorri tantüna, dai, pigliai davant la balla!\" Bap siglia mez sü da cuotscha e dà da mans e da peis. Brutus guarda ün pa tort aint per el e manaja: \"Che crajast, cha tü rivast da far ün gol pels Taliens, scha tü sigliast intuorn uschè?\" \"Bap nu fess sgüra na ün gol pels Taliens! Sch'el fess ün, lura sgüra pels Svizzers\", ria Luisa. Lupo riva uossa tuots duos ögls, doza il cheu e manaja: \"Perche na pels Taliens? Eu sun tuottüna eir eu ün Talian e'l s Taliens fessan sgüra eir jent ün gol.\" \"Na, disch Nic, \"tü est uossa eir ün Svizzer, nun, mamma?\" \"Mo\", manaja mamma, \"Lupo es vairamaing ün Talian, ma el es creschü sü in Svizra, uschè ch'el sarà mez talian e mez svizzer.\" \"Eu tegn culs Taliens!\", disch Lupo. \"Ed eu culs Svizzers\", disch uossa Nic ün pa malsgür. \"Ed eu culs Taliens e culs Svizzers\", manaja Luisa. \"E tü mamma, cun chi tegnast tü?\" Mamma manaja: \"Vairamaing cun ingün - eu cuiisch a tuottas duos squadras schi guadognan.\"\"},
  {
    "role": "assistant", "content": "Grazia fitg per questa decleraziun" }
}
```



Python-Skript zur Datenvorbereitung

```
••• datasplit.py

import pdfplumber
import json
import os

def process_pdf(filename):
    with pdfplumber.open(filename) as pdf:
        data_list = []

        # Den gesamten Text des Dokuments extrahieren
        text = '\n'.join(page.extract_text() for page in pdf.pages)

        # Text in einzelne Absätze aufteilen
        paragraphs = text.split('\n')

        # Erstes Element für den Prompt festlegen
        prompt = paragraphs[0] if paragraphs else ''

        # Gehe durch jeden Abschnitt
        for i in range(1, len(paragraphs)):
            # Schlüssel-Wert-Paar zu Daten hinzufügen
            data = {"prompt": prompt, "completion": paragraphs[i]}
            data_list.append(data)

            # Aktuelles Completion als nächstes Prompt verwenden
            prompt = paragraphs[i]

    return data_list

def process_multiple_pdfs(file_list):
    all_data = []

    # Iterieren über jede Datei in der Liste
    for file in file_list:
        # Verarbeitet jede Datei und fügt diese der Liste hinzu
        all_data.extend(process_pdf(file))

    # erstellen eines json Files mit den kombinierten Daten
    with open('outputall.json', 'w') as json_file:
        json.dump(all_data, json_file, ensure_ascii=False)

    # Liste der pdf Files
    pdf_files = [f for f in os.listdir('.') if f.endswith('.pdf')]

process_multiple_pdfs(pdf_files)
```