BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Fachbereich VI - Informatik und Medien
Studiengang Medieninformatik

Entwicklung eines Chatbots zur Unterstützung beim Erlernen einer Sprache

Abschlussarbeit

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

Eingereicht von: Karin Lampesberger
Matrikelnummer: 847835
Datum der Abgabe: 02.10.2019

Betreuerin: Prof. Dr. Agathe Merceron
Gutachter: Prof. Dr. Thomas Off

Kurzfassung

Der technische Fortschritt in den vergangenen Jahren in Bereichen wie maschinellem Lernen und Natural Language Processing (NLP) führte dazu, dass Chatbots erheblich an Popularität gewonnen haben und die Nachfrage an jenen gestiegen ist. Im Bildungssektor haben sich Chatbots bisher noch nicht etabliert, obwohl sie speziell im Zweitspracherwerb durch ihre ständige Verfügbarkeit, ihr interaktives Verhalten und ihre Möglichkeit zur Personalisierung ideal als Motivation wären.

Die vorliegende Masterarbeit befasst sich mit der Konzipierung und prototypischen Entwicklung eines Chatbots, mit dessen Hilfe die Konversationskenntnisse in Englisch durch das Führen schriftlicher Dialoge sowie dem Ausbau des Wortschatzes verbessert werden können. Der Schwerpunkt der Gespräche liegt auf Smalltalk und auf universitätsspezifischem Vokabular. Es wird untersucht, inwiefern sich das Open-Source Framework von Rasa oder Googles Plattform Dialogflow für die Entwicklung eigener Sprachlernapplikationen eignen und welche Vor- und Nachteile sich jeweils ergeben. In Interviews mit Sprachlehrenden werden die Anforderungen an einen Chatbot im Spracherwerb identifiziert und ein Konzept für den "EnglishBot" abgeleitet. Die prototypische Implementierung mithilfe des Rasa Stacks verfolgt einen Ansatz, der auf maschinellem Lernen basiert, und zeigt die Herausforderungen und Grenzen auf, welche bei Anwendungen auftreten, die natürliche Sprache verarbeiten. In einer abschließenden Evaluierung wird die Performance der Anwendung in einem Usability-Test erprobt. Die positive Bewertung des Prototyps zeigt, dass sich der Rasa Stack - unter der Bedingung einer eingegrenzten Thematik - durchaus eignet, um Sprachlernanwendungen zu entwickeln.

Abstract

Technological advances in recent years in areas such as machine learning and natural language processing (NLP) have led to a significant increase in the popularity and demand for chatbots. In education and especially in language acquisition chatbots have not yet become established, although their constant availability, interactive behaviour and ability to personalize make them ideal motivators for learning languages in particular.

This thesis deals with the conceptual design and protoypical implementation of a chatbot that helps to improve one's English conversational skills by conducting written dialogues. Conversations focus on smalltalk and on training vocabulary, specifically on the topic university. It is examined to what extent the open source framework by Rasa or Googles platform Dialogflow is suitable for the development of own second language learning applications and what their advantages and disadvantages are. In interviews with English teachers, the requirements for a chatbot in language acquisition are identified and a concept for the "EnglishBot" derived. The prototypical implementation using the Rasa stack follows a machine learning approach and shows the challenges and limitations that occur in applications that process natural language. In a final evaluation the performance of the application is being tested in a usability test. The positive evaluation of the prototype shows that the Rasa stack is suitable for developing own language learning applications. However, under the condition that the focus is on a limited topic.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 2. Oktober 2019

Karin Lampesberger

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

AIML Artificial Intelligence Markup Language.

API Application Programming Interface.

CRF Conditional Random Field.

DSGVO Datenschutz-Grundverordnung.

HTTP Hyper Text Transfer Protocol.

LSTM Long Short-Term Memory.

NER Named-Entity Recognition.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLTK Natural Language Toolkit.

NLU Natural Language Understanding.

NP-Chunking Noun-Phrase-Chunking.

POS Part-of-Speech.

SUS System Usability Scale.

UCD User-Centered Design.

URL Uniform Resource Locator.

1 Einleitung

1.1 Motivation

Sprachen sind wie Schlüssel zu anderen Welten. Sowohl im Privat- als auch im Berufsleben sind sie von großer Wichtigkeit. Beim Erlernen einer Sprache werden oft Hilfsmittel verwendet wie beispielsweise Applikationen für das Smartphone. Diese haben den Vorteil, dass orts- und zeitunabhängig gelernt werden kann [vgl. Sch14]. Es gibt zahlreiche Applikationen in den diversen App-Stores, mit deren Hilfe eine Sprache erlernt werden kann. Viele Anwendungen konzentrieren sich vor allem darauf, Vokabeln zu lernen oder die Grammatik zu üben. Lernende haben jedoch oft Schwierigkeiten dabei, tatsächliche Konversationen im Alltag zu führen. Hinzu kommt, dass in der Kommunikation mit Muttersprachlern oft Angst oder Scham vor Fehlern die Lernenden daran hindert, eine Sprache zu üben. Mithilfe eines Chatbots könnte dem abgeholfen werden. Durch das Führen von schriftlichen Dialogen und dem Erhalt von Feedback könnten Chatbots Lernende unterstützen.

1.2 Problemstellung

Beim Erlernen einer Sprache spricht man von vier Fertigkeiten, die in einer neuen Sprache trainiert werden sollen: Lesen, Schreiben, Hören und Sprechen. Mobile Geräte haben sich bereits im Bildungssektor etabliert. Gerade das Online-Chatten bringt einige Vorteile mit sich: es fördert die Autonomie, Kommunikationsfähigkeiten und soziale Kompetenzen und verringert Ängste der Lernenden ([vgl. SA04, S. 407-412] und [vgl. Gon03]). Mobile Anwendungen eignen sich für das Lernen von Sprachen, vor allem für das Training der vier genannten Fertigkeiten [vgl. MN12]. Insbesondere zur Verbesserung des Hör- und Sprachverständnisses ist der Einsatz von mobilen Geräten nützlich [vgl. Kuk12]. Es gibt bereits Chatbots, die die Konversationskenntnisse verbessern sollen. Duolingo[1] - einer der vorreitenden Anbieter von Sprachapplikationen - bietet seit Ende des Jahres 2016 auch Chatbots für ausgewählte Sprachen und Szenarien an, bisher allerdings nur für iOS-Geräte. Auch zur Erstellung eines Chatbots gibt es unzählige Anbieter und Plattformen. Beispielsweise stellen namhafte Unternehmen wie Google

[1] https://de.duolingo.com

(Dialogflow[2]), Microsoft (Azure Bot Service[3]) oder Facebook (Bots for Messenger[4]) Dienste zur einfachen Erstellung eines Bots bereit. Darüber hinaus gibt es weitere Chatbot-Frameworks wie Rasa[5], wit.ai[6] oder luis.ai[7].

1.3 Ziel der Arbeit

Das Ziel dieser Abschlussarbeit ist die prototypische Entwicklung eines Chatbots, mit dessen Hilfe die Konversationskenntnisse in Englisch durch das Führen schriftlicher Dialoge sowie dem Ausbau des Wortschatzes verbessert werden können. Es soll untersucht werden, inwiefern sich das Open-Source Framework von Rasa oder Googles Plattform Dialogflow für die Entwicklung eigener Sprachlernapplikationen eignen und was deren Vor- und Nachteile sind. In Interviews mit Sprachlehrenden sollen Anforderungen an eine Sprachlernanwendung identifiziert und anschließend in ein Konzept umgesetzt werden. Der Prozess der Anwendungsentwicklung soll von zukünftigen Nutzer*innen begleitet werden, um Probleme möglichst früh erkennen und beheben zu können. Die Zielgruppe der Anwendung sind Studierende, die ein Semester im Ausland verbringen und ihr Englisch verbessern möchten. Der Schwerpunkt der Anwendung liegt auf der Erweiterung des Wortschatzes mit Fokus auf universitätsspezifischem Vokabular sowie auf Smalltalk-Gesprächen.

1.4 Forschungsfragen

Aufgrund der dargelegten Ausgangslage ergeben sich die folgenden Forschungsfragen:

- Welche Anforderungen werden an einen Chatbot als Lernhilfe im Zweitspracherwerb gestellt?
- Inwiefern eignen sich bestimmte Frameworks zur Erstellung eines Chatbots als Hilfsmittel im Zweitspracherwerb?
  - Was sind die Vor- und Nachteile des Rasa Stacks im Vergleich zu Dialogflow?
  - Welche Herausforderungen ergeben sich bei der konkreten Implementierung in Bezug auf die beschriebene Zielerreichung?

[2] https://dialogflow.com
[3] https://azure.microsoft.com/de-de/services/bot-service
[4] https://developers.facebook.com/docs/messenger-platform/getting-started/sample-apps
[5] https://rasa.com
[6] https://wit.ai
[7] https://www.luis.ai

1.5 Gliederung und Aufbau

Die vorliegende Arbeit ist in acht Kapitel gegliedert:

1. Einleitung In der Einleitung werden die Motivation, die Problemstellung und das Ziel der Arbeit sowie die zu beantwortenden Fragestellungen beschrieben.

2. Chatbots Im 2. Kapitel werden die Grundlagen von Chatbots und Natural Language Processing (NLP) sowie dessen wichtigsten Methoden erläutert. Darüber hinaus werden Anwendungen für Chatbots im Spracherwerb und im universitären Bereich vorgestellt.

3. Chatbot Development Frameworks Im 3. Kapitel werden die Chatbot Building Frameworks Dialogflow und der Rasa Stack vorgestellt und anschließend verglichen.

4. Anforderungsanalyse In diesem Kapitel werden die Vorgehensweise der Anforderungsanalyse erläutert, die Interviews mit Sprachlehrenden zusammengefasst und die daraus abgeleiteten Anforderungen definiert.

5. Konzept des EnglishBot Dieses Kapitel behandelt das Konzept der Anwendung. Es werden die Use-Cases aufgelistet sowie der grundsätzliche Ablauf des Programms erklärt. Ferner wird näher auf die Lerninhalte und die Datenspeicherung sowie auf die Architektur der App eingegangen und es werden Mockups der Benutzerschnittstelle präsentiert.

6. Implementierung Im 6. Kapitel wird die praktische Anwendung des Konzepts dargestellt.

7. Evaluierung In diesem Kapitel wird der Prototyp im Rahmen von Usability-Tests untersucht und darauffolgend bewertet.

8. Fazit Abschließend werden die Erkenntnisse der Arbeit zusammengefasst. Darüber hinaus erfolgt ein Ausblick mit Erweiterungsmöglichkeiten der Anwendung.

2 Chatbots

Das folgende Kapitel behandelt die Grundlagen von Chatbots. Nach einer historischen Betrachtung erfolgt eine Klassifikation nach ihrem Antwortverhalten. Anschließend werden die einzelnen Komponenten eines Chatbots vorgestellt, der einen auf künstlicher Intelligenz basierenden Ansatz verfolgt. Im weiteren Verlauf wird näher auf das Natural Language Understanding (NLU) eingegangen und es werden in diesem Zusammenhang wichtige Methoden erklärt. Das Kapitel endet mit der Vorstellung bereits existierender Chatbots im Spracherwerb bzw. im universitären Umfeld.

2.1 Eine historische Betrachtung von Chatbots

Chatbots sind Computerprogramme, die Eingaben von Nutzer*innen in natürlicher Sprache verarbeiten, dazu relevante Antworten generieren und diese dann an die Nutzer*innen zurückschicken [vgl. KD18, S. 1]. Im Zusammenhang mit Chatbots fallen auch oft die Begriffe "Dialogsystem" und "Conversational Agent". Dialogsysteme beschreiben Systeme, die mit Nutzer*innen in natürlicher Sprache kommunizieren. Der Begriff des Conversational Agent wird synonym verwendet. Chatbots bilden eine Untergruppe von Dialogsystemen und sind darauf ausgerichtet, ausgedehnte textbasierte Konversationen zu führen, die die Interaktion zwischen Menschen imitieren sollen. Daneben gibt es Dialogsysteme, die das Lösen von bestimmten Aufgaben zum Ziel haben ("Aufgabenorientierte Dialogagenten"). [vgl. JM19, S. 422 - 423]

Vor allem in den letzten Jahren haben Chatbots enorm an Popularität gewonnen, allerdings reicht deren Geschichte bis in die 50er Jahre zurück. Alan Turing entwickelte 1950 den Turing-Test (englisch: Imitation Game). Bei diesem Intelligenztest für Maschinen wird geprüft, ob eine Maschine ein intelligentes Verhalten zeigen kann, welches dem eines Menschen gleicht [vgl. Raj19, S. 13]. 1966 stellte Joseph Weizenbaum mit Eliza den ersten Chatbot vor. Eliza täuschte dabei eine Konversation mit einem Psychotherapeuten vor. Das Ziel war es zu zeigen, dass Mensch und Computer in natürlicher Sprache miteinander kommunizieren können. Der eingegebene Satz wurde dabei auf Schlüsselwörter untersucht und darauf basierend wurde eine passende, grammatikalisch korrekte Antwort gegeben. Mit Eliza gelang es Joseph Weizenbaum, einen Meilenstein in der Entwicklung künstlich intelligenter Systeme zu setzen. [vgl. KD18, S. 2]

In den darauffolgenden Jahrzehnten folgten Chatbots wie Parry (1972), der das Verhalten eines paranoiden Schizophrenen imitierte oder Jabberwocky (1981), der eine

menschliche Kommunikation auf eine interessante und humorvolle Art und Weise zu simulieren anstrebte [vgl. Raj19, S. 14]. Ein weiterer Erfolg gelang Richard Wallace 1995 mit A.L.I.C.E. (Artificial Linguistic Internet Computer Entity). Das Wissen von A.L.I.C.E. war in Dateien im Artificial Intelligence Markup Language (AIML)-Format gespeichert [vgl. KD18, S. 2]. Auch wenn A.L.I.C.E. den Turing-Test nicht bestand, stellt sie bis heute einen der erfolgreichsten Chatbots dar und wurde mehrfach mit dem Loebner-Preis gewürdigt [vgl. KD18, S. 2]. Bei dem jährlich stattfindenden Wettbewerb werden Computerprogramme ausgezeichnet, die den Turing-Test am längsten oder gar völlig bestehen. Allerdings gelang letzteres bisher niemandem.

Anfang des 21. Jahrhunderts wurde SmarterChild von ActiveBuddy veröffentlicht [vgl. KD18, S. 3]. Es war der erste Versuch, einen Chatbot zu erstellen, der nicht nur unterhalten, sondern die Anwendenden auch mit Informationen zum aktuellen Wetter, zu Aktienkursen oder zu Sportergebnissen versorgen sollte. SmarterChild gilt als Vorläufer für Siri, die von Apple entwickelte persönliche Sprachassistentin. Siri ist Bestandteil des Apple Betriebssystems iOS und ermöglicht beliebige Konversationen mit ihren Nutzer*innen durch Spracheingabe. Überdies liefert sie aktuelle Informationen, zum Beispiel zum Wetter oder zu den Aktienkursen, wie es bereits SmarterChild tat. Unternehmen wie Google und Amazon folgten dem Trend und veröffentlichten wenige Jahre später mit Google Now und Alexa ebenfalls Sprachassistenten. [vgl. KD18, S. 3]

2.2 Architektur

Chatbots müssen in der Lage sein, natürliche Sprache zu verarbeiten und entsprechend zu reagieren, um mit ihren Anwender*innen kommunizieren zu können. Grundsätzlich sind dafür die folgenden drei Komponenten Natural Language Understanding (NLU), Dialogmanager und Natural Language Generation (NLG) verantwortlich [vgl. JM09, S. 2835]. Diese Architektur entspricht der eines Dialogsystems. Das Zusammenspiel der einzelnen Komponenten wird in der Abbildung 2.1 gezeigt.

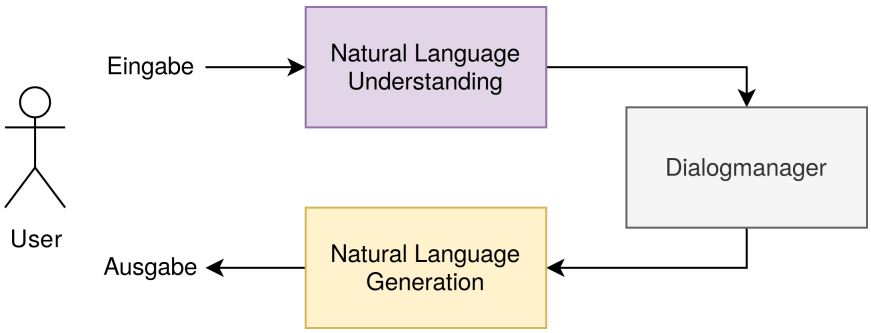


Abbildung 2.1: Architektur eines Dialogsystems (Quelle: in Anlehnung an [vgl. JM09, S. 835])

Die Aufgabe der NLU-Komponente ist die Analyse der Eingabe. Zum einen erfolgt eine syntaktische Analyse, die zum Ziel hat, die grammatikalischen Beziehungen zwischen den einzelnen Wörtern zu identifizieren. Zum anderen wird versucht, die Bedeutung der Eingabe festzustellen. Die NLU-Komponente bedient sich der Methoden des Natural Language Processing (NLP), auf welche im Kapitel 2.4.1 näher eingegangen wird. Der Dialogmanager verfolgt die Konversation und bestimmt, welche Aktionen - basierend auf den Ergebnissen des NLU - als nächstes durchgeführt werden. Schließlich folgt das Erstellen von Antworten durch die NLG-Komponente. [vgl. JM09, S. 837 - 842]

2.3 Klassifikation von Chatbots

Ein übliches Merkmal zur Klassifikation von Chatbots ist deren Antwortverhalten. Die Abbildung 2.2 visualisiert Ansätze, anhand derer Chatbots unterschieden werden können.

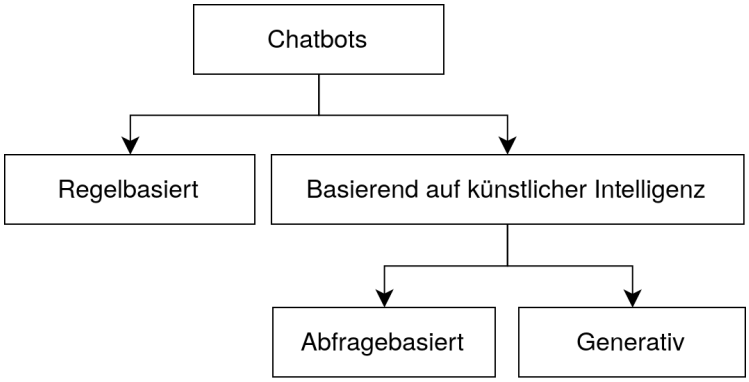


Abbildung 2.2: Klassifizierung von Chatbots nach ihrem Antwortverhalten (Quelle: in Anlehnung an [vgl. Sal19, S. 5159])

2.3.1 Regelbasierter Ansatz

Beim regelbasierten Ansatz, der bereits bei Chatbots wie Eliza oder Parry Anwendung fand, werden im Vorfeld Regeln festgelegt. Die Eingabe wird auf bestimmte Schlüsselwörter untersucht und fest definierte Entscheidungsbäume (englisch: decision trees) bestimmen den Gesprächsverlauf. Chatbots dieser Art stoßen bei Eingaben, die keine Schlüsselwörter enthalten, an ihre Grenzen.

2.3.2 Machine Learning Ansatz

Im Gegensatz zum regelbasierten Ansatz existieren auch auf künstlicher Intelligenz basierende Ansätze. Künstliche Intelligenz bezeichnet ein Gebiet der Informatik, welches untersucht, wie Maschinen ein dem Menschen ähnliches Verhalten imitieren können [vgl. Hay94]. Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz, welcher sich zum Ziel gesetzt hat, Vorhersagen auf der Grundlage vorheriger Beobachtungen zu treffen [vgl. GH17, S. 2]. Machine Learning macht dabei von neuronalen Netzen Gebrauch.
Ein neuronales Netz beschreibt ein mathematisches Modell, welches durch den Aufbau und die Arbeitsweise des menschlichen Gehirns inspiriert ist. Das menschliche Gehirn besteht aus miteinander verbundenen Neuronen, die Informationen verarbeiten und an andere Neuronen weiterleiten. Neuronale Netze versuchen dieses Verhalten nachzuahmen. Ein neuronales Netz besteht aus Schichten von Neuronen. Eingehende Daten werden auf der Eingabeschicht empfangen und gelangen durch eine oder mehrere Zwischenschichten zur Ausgabeschicht. Die einzelnen Neuronen sind miteinander verbunden. Diese Verbindungen sind unterschiedlich gewichtet und werden während des Trainings optimiert. [vgl. GH17, S. 41 - 42]
Supervised Learning (deutsch: überwachtes Lernen) - ein Ansatz des Machine Learning - bezeichnet das Lernen anhand von Beispielen, aus welchen allgemeine Aussagen abgeleitet werden [vgl. GH17, S. 13]. Es eignet sich unter anderem für die Klassifikation, bei der neue Eingaben bereits bestehenden Kategorien zugeordnet werden [vgl. GH17, S. 13].

Chatbots, die einen Machine Learning Ansatz verfolgen, erzeugen Antworten entweder abfragebasiert (englisch: retrieval-based) oder generativ (englisch: generative). Erstere werden auf eine vordefinierte Menge an möglichen Fragen und Antworten trainiert. Training bedeutet in diesem Fall ein statistisches Modell zu erstellen, das anhand dieses vordefinierten Sets lernt, neue Eingaben zu kategorisieren (*Supervised Learning*). Es wird die Antwort zurückgegeben, die am wahrscheinlichsten dem Intent der Nutzer*innen entspricht. Dabei werden keine Antworten generiert, sondern auf vordefinierte

zurückgegriffen. [vgl. JM19, S. 425 - 428]
Bei generativen Modellen werden die Antworten von Grund auf neu erstellt. Diese Vorgehensweise basiert auf großen Datenmengen und nutzt zur Erstellung von Antworten Deep Learning und maschinelle Übersetzung (englisch: maschine translation). [vgl. JM19, S. 428]

Der Fortschritt in den Bereichen der künstlichen Intelligenz und dem Machine Learning führte dazu, dass aktuelle Tools und Frameworks zur Entwicklung von Chatbots immer mehr auf künstliche Intelligenz, anstatt des klassischen regelbasierten Vorgehens, setzen [vgl. CCH17, S.25].

Machine Learning und im Speziellen *Supervised Learning* findet jedoch nicht nur bei der Antwortgenerierung Anwendung, sondern wird auch für das Natural Language Understanding genutzt.

2.4 Natural Language Processing

Natural Language Processing (NLP) ist ein Teilbereich der künstlichen Intelligenz und ermöglicht es Computern, menschliche Sprache zu analysieren und zu verstehen. Natural Language Understanding (NLU) wiederum ist ein Bereich des NLP und zielt darauf ab, die Bedeutung einer Eingabe im Detail zu erkennen. [vgl. KD18, S. 29]

2.4.1 Natural Language Understanding

Die NLU-Komponente hat zur Aufgabe, die *Domain* (deutsch: Domäne) zu klassifizieren, *Intents* (deutsch: Absichten) zu erkennen, *Entities* (deutsch: Entitäten) zu extrahieren und schließlich in *Slots* zu speichern. Die *Domäne* beschreibt einen Themenbereich, über den ein Bot mit den Anwendenden kommuniziert. Falls ein Chatbot ohnehin nur in einer Domäne agiert, so entfällt dieser Schritt. Im zweiten Schritt wird versucht, den *Intent* zu erkennen. Es soll die angestrebte Aufgabe oder das zu erreichende Ziel der Nutzer*innen herausgefunden werden. Im letzten Schritt werden die Parameter eines Satzes extrahiert. *Slots* speichern bereitgestellte Informationen und werden als das Gedächtnis eines Chatbots betrachtet. Ebenso erfolgt in diesem Schritt das Extrahieren von *Entitäten*, welche beispielsweise Eigennamen, Daten oder Zahlen sein können. [vgl. JM19, S. 434]

Im folgenden Beispiel soll der Ablauf dieser Schritte verdeutlicht werden. Die Eingabe lautet wie folgt:

> "Where can I find the International Office?"

Ein System könnte diese Eingabe folgendermaßen darstellen:

Domäne	university	
Intent	getLocation	
Slots	facility	International Office

Tabelle 2.1: Beispieldarstellung der Eingabe (Quelle: eigene Darstellung)

Basierend auf den extrahierten Informationen kann der Chatbot eine Antwort schicken. Im vorliegenden Fall könnte die Antwort folgendermaßen lauten:

"You can find the International Office in room C7 in the Grashof building."

Heutzutage bieten viele Unternehmen wie beispielsweise Google oder IBM eigene Dienste für das NLU an.

2.4.2 NLP-Methoden

NLP-Methoden helfen dabei, die Eingabe in Teile zu zerlegen und den Sinn zu verstehen. Zudem unterstützen sie bei der Überprüfung der grammatikalischen Korrektheit eines Satzes. Tools zur Rechtschreib- und Grammatikprüfung (wie beispielsweise LanguageTool, siehe Kapitel 6.3) machen häufig von NLP-Methoden Gebrauch.

Tokenisierung
Der erste Schritt bei der Verarbeitung einer Eingabe ist meist die Tokenisierung. Dabei wird der Text in sinnvolle Segmente (auch *Tokens* genannt) aufgeteilt, beispielsweise in Wörter und Satzzeichen. In Sprachen, bei denen ein Leerzeichen die einzelnen Wörter trennt, ist es üblich, jenes auch als Trennzeichen für die Tokenisierung zu verwenden. [vgl. Raj19, S. 59]

Part-of-Speech Tagging
Beim Part-of-Speech (POS) Tagging (deutsch: Wortartenerkennung) wird jedem Wort oder Token eines Textes sein zugehöriger POS Tag zugeordnet. POS Tags sind bestimmte Wortarten, wie zum Beispiel "Verb", "Nomen", "Präposition". Jede Sprache hat ihre eigenen POS Tags [vgl. Raj19, S. 37]. Der folgende Satz veranschaulicht ein Ergebnis des POS Tagging.

The	university	is	located	in	Germany
Artikel	Nomen	Verb	Verb	Präposition	Nomen

Tabelle 2.2: Beispiel für einen Satz und den dazugehörigen POS Tags (Quelle: eigene Darstellung)

Es gibt zwei Arten des Taggings: das regelbasierte Tagging, welches eine manuell erstellte Grammatik benötigt, und das stochastische Tagging, für das ein manuell annotierter

Trainingskorpus gebraucht wird [vgl. JM09, S. 15]. Ein Korpus ist eine große, strukturierte Sammlung von Text oder Sprache. Ein Beispiel dafür ist der Brown Corpus[1], welcher der erste Korpus für das amerikanische Englisch war.

POS Tagging spielt im Zusammenhang mit Chatbots eine wesentliche Rolle, da dadurch die Komplexität eines Textes verringert wird. Es ist notwendig, um Mehrdeutigkeiten aufzulösen. Eine Vielzahl an Wörtern kann mehrere Bedeutungen haben. Beispielsweise hat das Wort "book" in den Sätzen "I buy a book" und "I book a flight" unterschiedliche Bedeutungen. Im ersten Satz ist es ein Nomen, im zweiten allerdings ein Verb. [vgl. Raj19, S. 37]

Stemming und Lemmatisierung

Als Stemming bezeichnet man den Prozess der Reduktion eines Wortes auf dessen Wortstamm, indem das Ende des Wortes weggeschnitten wird. Es wird angenommen, dass der Rest des Wortes dem gesuchten Wort entspricht. Beispielsweise bleibt nach Entfernen des Endes des Wortes "talking" das Wort "talk" übrig. Allerdings funktioniert das einfache Wegschneiden nicht bei allen Wörtern. So würde beim Wort "presumably" nur "presum" übrig bleiben und somit fehlerhaft sein. [vgl. Raj19, S. 42]

Bei der Lemmatisierung wird - ähnlich wie beim Stemming - durch einen Algorithmus das Lemma (also die Grundform) eines Wortes bestimmt. Allerdings wird bei der Lemmatisierung nicht einfach das Ende weggeschnitten, sondern die Grundform durch Nachschlagen in einem elektronischen Wörterbuch bestimmt. Beispielsweise wird bei der Lemmatisierung für das Verb "is" die Grundform "be" zurückgegeben. Lemmatisierung hat im Vergleich zum Stemming den Vorteil, dass die gefundene Grundform in jedem Fall einem existierenden Wort entspricht, weil es einem Wörterbuch entstammt. [vgl. Raj19, S. 42]

Named-Entity Recognition

Named-Entity Recognition (NER) (deutsch: Eigennamenerkennung, auch Extraktion von Entitäten genannt) ist ähnlich wie POS Tagging, allerdings werden nur die Wörter oder Sequenzen an Wörtern extrahiert, die einer vordefinierten Kategorie entsprechen (z.B. Namen von Menschen oder von Unternehmen). Der Erfolg von NER ist stark abhängig von dem für das Training verwendeten Datensatz. [vgl. Raj19, S. 44]

[1] https://www.sketchengine.eu/brown-corpus

Stoppwörter

Stoppwörter sind Wörter, die besonders häufig vorkommen und entfernt werden sollen, bevor die eigentliche Verarbeitung eines Textes beginnt. Beispielsweise sind "a", "an", "the" oder "to" Stoppwörter in der englischen Sprache. Diese Wörter sind für die Bedeutung eines Textes, insbesondere wenn es sich um Informationsgewinnung handelt, nicht von bedeutender Relevanz. [vgl. Raj19, S. 47 - 49]

Dependency Parsing

Als Parser bezeichnet man ein Programm zur syntaktischen Analyse. Dabei wird geprüft, ob eine Eingabe gemäß den Regeln einer Grammatik zulässig ist. Dependency Parsing ist eine besondere Form des Parsing, bei der die grammatikalische Struktur eines Satzes analysiert wird und Beziehungen zwischen den einzelnen Wörtern hergestellt werden. [vgl. MR11, S. 141]

Das folgende Beispiel veranschaulicht die Relationen beim Dependency Parsing:

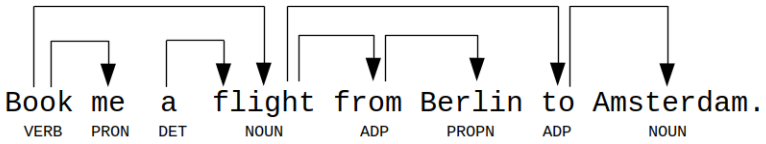


Abbildung 2.3: Beispiel Dependency Parsing (Quelle: in Anlehnung an [vgl. Raj19, S. 52])

Unterhalb der einzelnen Wörter des Satzes in Abbildung 2.3 stehen die jeweiligen Wortarten. Die Pfeile visualisieren die Relationen zwischen zwei Wörtern. An der Pfeilspitze steht das Wort, auf das sich das Wort am anderen Ende des Pfeils bezieht.

NLU-Komponenten nutzen Dependency Parsing, um Relationen zu finden und damit die Absicht der Nutzer*innen besser zu verstehen. Auch wenn über mehr als einem Kontext gesprochen wird, so ist es durch Dependency Parsing möglich, den Unterschied zu erfassen. [vgl. Raj19, S. 54]

Chunking

Chunking stellt eine grundlegende NLP-Methode dar, bei der Wortgruppen aus einem Text extrahiert werden [vgl. Col11, S. 4]. Es erfolgt eine Einteilung der Wörter eines Satzes in Gruppen, die sich nicht überlappen. Beispielsweise sollen Wortgruppen wie "New York " oder "Los Angeles" als ein einziges Wort erkannt werden. Noun-Phrase-Chunking (NP-Chunking) ist eine besonders häufig verwendete Art des Chunkings, bei der Nominalphrasen aus einem Satz extrahiert werden [vgl. Col11, S. 4]. Nominalphrasen sind Wortgruppen, die sich auf ein Nomen beziehen und dieses näher beschreiben [vgl. BKL09, S. 265].

Nachfolgend wird ein Beispiel für NP-Chunking vorgestellt. Die mit NP markierten Bereiche des Satzes stehen für die Nominalphrasen.

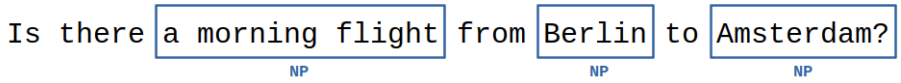


Abbildung 2.4: Beispiel NP-Chunking (Quelle: in Anlehnung an [vgl. JM19, S. 232])

2.5 Anwendungen von Chatbots

In der aktuellen Technik spielen Chatbots eine große Rolle. Sie kommen an vielen Stellen zum Einsatz. Viele Unternehmen haben Chatbots auf ihren Websites eingebaut, um Fragen von Besucher*innen zu beantworten. Aber auch Universitäten haben das Potential von Chatbots erkannt. So bietet beispielsweise die Georgia State University einen Chatbot zur Beratung von Studieninteressierten und Studierenden [PC14] an, der sogar nach mehreren Monaten im Einsatz nicht als Chatbot erkannt wurde. Auch an der Technischen Universität Berlin vermag der Chatbot "Alex" Fragen rund um Module und Kurse zu beantworten [MHW17]. Darüber hinaus bietet ein israelisches Technik-Unternehmen speziell für Universitäten und Hochschulen eine kostenpflichtige Chatbot-Plattform an. Es werden Vorlagen zur Verfügung gestellt, um schnell und einfach Chatbots für bestimmte Aufgaben erstellen zu können, beispielsweise für den Immatrikulationsprozess oder für die Betreuung von Studierenden [Uni18].

Im Bereich des mobilen Sprachenlernens existieren unzählige Applikationen und auch immer mehr Chatbots. Mondly, eine bekannte Sprachlernanwendung, veröffentlichte bereits im Jahr 2016 einen Chatbot, mit dem die Nutzer*innen über Sprache kommunizieren können [Ili16]. Duolingo, eine ebenfalls sehr erfolgreiche Applikation zum Lernen von Sprachen, bietet einen Chatbot für Spanisch, Französisch und Deutsch an, allerdings nur für iOS-Geräte. Die Anwendung "Andy the Chatbot" soll dabei helfen, Englisch zu lernen und zu üben. Andy stellt einen persönlichen Lehrer dar und richtet sich vor allem an Sprachanfänger [Pya17]. Asiatische Sprachen wie Koreanisch, Japanisch oder Chinesisch können mit dem Chatbot "Lanny" gelernt werden [Egg15].

2.6 Zusammenfassung

Chatbots sind keine neue Erfindung, sondern existieren bereits seit mehreren Jahrzehnten. Sie können regelbasiert arbeiten, d.h. es wird strikt vorgegebenen Regeln gefolgt, oder sie nutzen Machine Learning Modelle, welche entweder auf einer bestehenden Menge an Fragen und Antworten trainiert werden und anhand dieser lernen, neue Eingaben zu kategorisieren. Entsprechend kann eine vordefinierte Antwort zurückgegeben oder eine neue Antwort generiert werden. Die Architektur von Chatbots gleicht der eines Dialogsystems. Die NLU-Komponente ist verantwortlich für das Verständnis der Eingaben, während der Dialogmanager den Konversationsfluss kontrolliert und die NLG-Komponente für die Antworten zuständig ist.

Auf Basis dieses Kapitels können nun Chatbot Development Frameworks verglichen werden, mit denen Chatbots in der beschriebenen Architektur erstellt werden und die von den erklärten NLP-Methoden Gebrauch machen.

3 Chatbot Development Frameworks

In den vergangenen Jahren kamen immer mehr Tools und Frameworks zur Erstellung von Chatbots auf den Markt. Aufgrund der Vielzahl wurde eine Vorauswahl getroffen: im folgenden Kapitel werden die zwei Frameworks Dialogflow und Rasa vorgestellt. Dialogflow erweitert seit 2014 die Tools von Google. Rasa, sowohl Name des Herstellers als auch des Frameworks, stellt mit dem Open-Source Rasa Stack ein Machine Learning Framework für automatisierte text- oder sprachbasierte Konversationen bereit. Die beiden Tools werden auf die folgenden Merkmale [vgl. Pra18, S. 25] geprüft und abschließend verglichen.

Gebrauchstauglichkeit Die Gebrauchstauglichkeit beschreibt nach der DIN EN ISO 9241, wie leicht ein System zu erlernen ist, ob es intuitiv zu benutzen ist und ob es Zufriedenheit sicherstellt. Es soll erläutert werden, inwiefern technisches Vorwissen für die Benutzung des Frameworks notwendig ist.

Unterstützte Sprachen Es werden die Sprachen aufgelistet, die von dem Framework unterstützt werden.

Natural Language Understanding Natural Language Understanding kann unterschiedlich gelöst werden. Es wird beschrieben, wie das jeweilige Framework NLU umsetzt und welche Algorithmen dabei zum Einsatz kommen, sofern diese öffentlich sind.

Hosting und Verarbeitung Es wird verglichen, wo die Daten gehostet und verarbeitet werden.

Datensicherheit Es werden der Ort der Speicherung und Verarbeitung der Daten und die Bedingungen, welchen sie bei der Verwendung des jeweiligen Frameworks unterliegen, dargelegt.

Schnittstellen und Integration Die angebotenen Schnittstellen und die vom Framework unterstützten Dienste werden aufgelistet.

Sprache und Text Es wird untersucht, ob das Framework Text- und/oder Spracheingabe akzeptiert.

Lizenzen und Preise Das Preismodell und die Lizenzen werden erläutert.

Sonstiges In diesem Punkt werden Besonderheiten hervorgehoben, die das Framework von anderen unterscheidet.

3.1 Dialogflow

Dialogflow (früher API.AI) ist eine Google-Plattform für die Entwicklung von Google Assistenten und Chatbots, die in natürlicher Sprache mit den Nutzer*innen kommunizieren. Es wird eine webbasierte grafische Benutzeroberfläche (sogenannte "Dialogflow Console") zur Erstellung von Chatbots und Dialogsystemen angeboten [vgl. JM19, S. 422]. Dialogflow bedient sich des maschinellen Lernens, einem Teilbereich der künstlichen Intelligenz, und der Cloud Services von Google.

3.1.1 Konzepte von Dialogflow

Nachfolgend werden die Konzepte *Agents*, *Intents*, *Entities*, *Context* und *Fulfillment* erläutert.

Agents
Eine typische Konversation läuft in drei Schritten ab. Im ersten Schritt erfolgt die Eingabe durch die Nutzer*innen, welche auch als *Utterance* (deutsch: Äußerung) bezeichnet wird. Im zweiten Schritt versucht der *Agent*, die Nutzerabsicht herauszufinden. Dieser Vorgang wird auch Intent Matching genannt. Das Ergebnis wird im dritten Schritt als Antwort an die Nutzer*innen zurückgeschickt. Agents helfen dabei, die Eingabe der Nutzer*innen zu strukturieren und somit eine passende Antwort zurücksenden zu können. Agents können auch als NLU-Module betrachtet werden und überall dort eingebunden werden, wo Text- oder Spracheingaben in ausführbare Daten umgewandelt werden sollen. [vgl. Dia19a]

Intents
Als *Intent* bezeichnet man die Absicht der Nutzer*innen. Intents werden benötigt, um zu definieren, wie Konversationen funktionieren. Bei der Erstellung von Agents werden Antworten bestimmten Intents zugeordnet. Für jeden Intent werden Beispiele für Eingaben (Trainingsphrasen) aufgeführt, die den jeweiligen Intent auslösen können, und zugehörige Antworten definiert. Dialogflow bietet auch Welcome- und Fallback-Intents. Welcome-Intents dienen der Begrüßung zu Beginn einer Konversation. Fallback-Intents kommen zum Einsatz, wenn die Eingabe keinem vorhandenen Intent zugeordnet werden kann. [vgl. Dia19f]

Die folgende Grafik veranschaulicht die Erstellung eines Intents "name" mit Beispielfragen nach dem Namen.

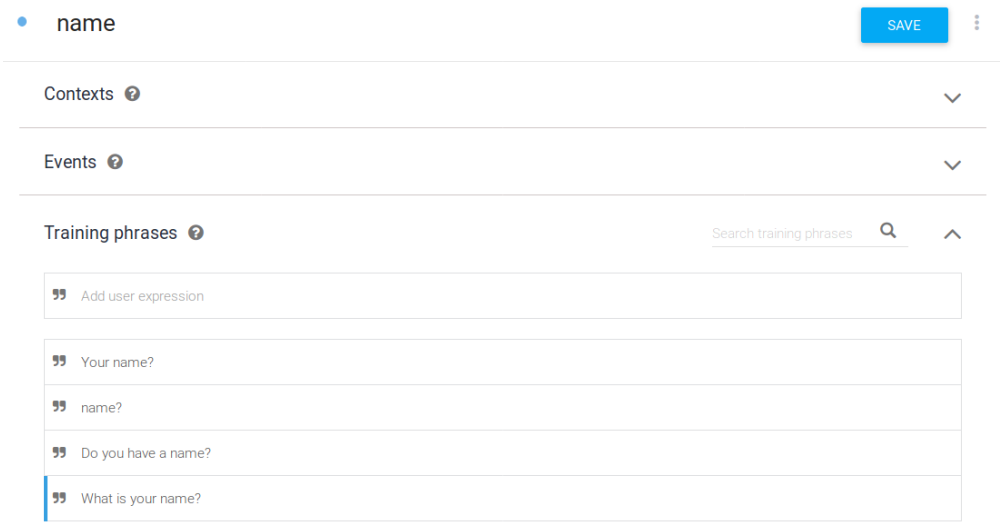


Abbildung 3.1: Beispiel Erstellung von Trainingsphrasen für einen Name-Intent (Screenshot der Dialogflow Console)

Zur Veranschaulichung wurden im Abschnitt "Responses" in der Abbildung 3.2 die folgenden möglichen Antworten festgelegt.

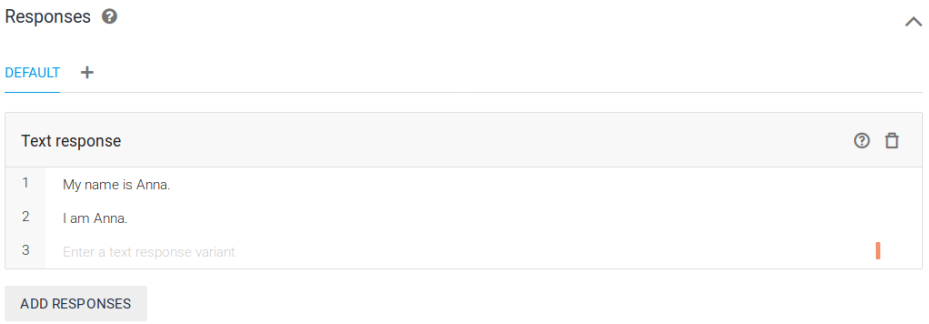


Abbildung 3.2: Beispielantworten (Screenshot der Dialogflow Console)

In einem Simulator können die Agents jederzeit getestet werden. In Abbildung 3.3 wird der Name-Intent getestet. Auf die Eingabe "What's your name?" wird schließlich korrekterweise "My name is Anna" geantwortet, obwohl die Eingabe von den in Abbildung 3.1 definierten Trainingsphrasen leicht abweicht.

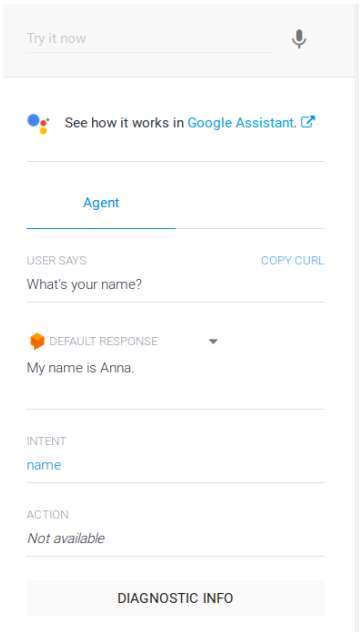


Abbildung 3.3: Test des Name-Intents (Screenshot der Dialogflow Console)

Zur Lösung dieses Problems nutzt Dialogflow Machine Learning, wobei für das Training des Modells die definierten Trainingsphrasen genutzt werden. Das Modell prüft, welcher Intent am ehesten zur Eingabe passt und erhält einen Wahrscheinlichkeitswert zwischen 0.0 (sehr unwahrscheinlich) und 1.0 (ganz sicher). Es wird angenommen, dass der Intent mit dem höchsten Wert der Absicht der Nutzer*innen entspricht. Liegt der höchste Wahrscheinlichkeitswert unter einem bestimmten Schwellenwert (englisch: threshold), wird der Fallback-Intent ausgelöst. [vgl. Dia19h]

Dialogflow bietet zwei Modi für das Intent-Matching an. Es kann entweder der hybride Modus gewählt werden, bei dem eine Kombination aus regelbasiertem Matching und Machine Learning zum Einsatz kommt. Dieser eignet sich vor allem für Agents mit wenigen Beispielen für Intents. Die zweite Variante basiert allein auf Machine Learning. [vgl. Dia19h]

Entities

Entities werden zur Identifikation und Extraktion brauchbarer Daten genutzt. Sie unterstützen bei der Erkennung spezifischer Informationen aus einer natürlichsprachlichen Eingabe, beispielsweise Straßen- oder Produktnamen. Für jede wichtige Information, die aus einer Eingabe extrahiert werden soll, existiert eine zugehörige *Entity*. [vgl. Dia19b]

Contexts

Contexts (deutsch: Kontexte) repräsentieren den aktuellen Status einer Anfrage und ermöglichen die Mitnahme von Informationen von einem Intent zum nächsten. Dadurch

kann die Reihenfolge von Intents beeinflusst und der Konversationsfluss kontrolliert werden. Das folgende Beispiel soll das Konzept verdeutlichen:

USER: How's the weather in Berlin?

CHATBOT: In Berlin it is currently sunny. The temperature is about 26° C.

USER: And in Paris?

CHATBOT: In Paris it is currently cloudy. The temperature is about 21° C.

Die Frage "And in Paris?" würde alleinstehend schwierig zu verstehen sein. Die Bestimmung des Kontexts ermöglicht dem Chatbot, die Frage richtig einzuordnen - nämlich in Bezug auf das Wetter - und eine entsprechende Antwort zu geben.

Dialogflow unterscheidet zwischen Ein- und Ausgabekontext und Follow-Up Intent. Ein Eingabekontext (englisch: input context) definiert einen aktiven Kontext als Bedingung für einen weiteren Intent. Ein Ausgabekontext (englisch: output context) bestimmt, dass ein Kontext nur dann aktiviert werden soll, wenn dieser noch nicht aktiv ist oder um einen Kontext beizubehalten, nachdem der passende Intent gefunden wurde. Standardmäßig erlöschen Kontexte nach fünf Anfragen oder 20 Minuten, nachdem ein Intent zugewiesen wurde. [vgl. Dia19e]
Follow-Up Intents sind eine einfache Möglichkeit, um Konversationen zu steuern, ohne Kontexte manuell kontrollieren zu müssen. Diese speziellen Intents werden innerhalb von übergeordneten Intents definiert und sollen voreingestellte Eingaben wie beispielsweise "Yes" oder "No" bearbeiten können. Sie erlöschen nach zwei Anfragen. [vgl. Dia19c]

Fulfillment
Als *Fulfillment* (deutsch: Erfüllung, Vollendung) wird die Integration von Fremdprodukten oder -datenbanken bezeichnet. Zur Nutzung muss für den jeweiligen Intent ein Webhook eingerichtet werden, welcher die Bereitstellung von Daten für andere Anwendungen in Echtzeit ermöglicht. Ist für einen Intent Fulfillment aktiviert, sendet Dialogflow eine Hyper Text Transfer Protocol (HTTP)-Anfrage an den jeweiligen Webhook, welcher nach Verarbeitung die entsprechende Antwort zurückschickt. Mithilfe von Fulfillment können beispielsweise dynamische Antworten erstellt werden, die auf Informationen einer Datenbank basieren. [vgl. Dia19d]

Knowledge Connectors
Knowledge Connectors (deutsch: Wissens-Konnektoren) ermöglichen es einem Agent, Anfragen zu beantworten, die auf vorgegebenen Wissensquellen wie Websites, Artikeln oder FAQ (Frequently Asked Questions) basieren. Knowledge Connectors parsen die Quellen, um automatisierte Antworten zu finden. Zurzeit sind sie nur in der Beta-Version

verfügbar, allerdings sind sie bereits in der Dialogflow Console integriert und können dort aktiviert werden. [vgl. Dia19g]

3.1.2 Eigenschaften von Dialogflow

Im Folgenden werden die Merkmale von Dialogflow zusammengefasst:

Gebrauchstauglichkeit Mit Dialogflow können Chatbots einfach und in kurzer Zeit erstellt werden. Durch die übersichtliche grafische Benutzeroberfläche, gute Dokumentation und Tutorials wird der Einstieg erleichtert. Nicht zuletzt gibt es mehr als 30 vorgefertigte Chatbots für besonders häufige Anwendungsszenarien. Über die Benutzeroberfläche kann eine große Anzahl an Funktionen freigeschaltet werden. Auch ohne Vorwissen ist es möglich, Chatbots zu entwickeln.

Unterstützte Sprachen Es werden mehr als 20 Sprachen unterstützt, darunter Englisch, Deutsch und Spanisch, aber auch Hindu oder Japanisch.

Natural Language Understanding Zur Klassifizierung der Intents wird Machine Learning genutzt. Allerdings können die verwendeten Algorithmen nicht eingesehen werden, da der Quellcode nicht öffentlich ist. Es ist jedoch möglich, den Schwellenwert festzulegen und zwischen dem hybriden Modus und dem Machine Learning Modus für das Intent Matching zu wählen.

Hosting und Verarbeitung Dialogflow läuft auf der Google Cloud Plattform[1] und jede Anfrage wird dort verarbeitet, wodurch die Nutzungsgeräte entlastet werden.

Datensicherheit Zur Nutzung der Dialogflow Console und der Erstellung eines Agents wird ein Google Account benötigt. Alle Daten, die mit dem Agent zusammenhängen, werden in der Google Cloud gespeichert. Die Nutzung von Dialogflow in der Enterprise-Version unterliegt den Datenschutz- und Sicherheitsbestimmungen der Google Cloud Plattform. Die kostenfreie Variante hingegen unterliegt den Nutzungsbedingungen von Googles Application Programming Interface (API)[2], wodurch es zum Beispiel zu Einschränkungen hinsichtlich der maximalen Anzahl von Intents oder Agents kommt. Google Nutzerdaten werden je Session der Konversation gespeichert. Die Daten können persistiert werden, indem eine externe Datenbank über einen Webhook eingebunden wird.

Schnittstellen und Integration Dialogflow bietet verschiedene SDKs (Software Development Kits)[3] für Programmiersprachen wie Java, Python oder auch Node.js. Mithilfe

[1] https://cloud.google.com
[2] https://developers.google.com/terms
[3] https://cloud.google.com/dialogflow/docs/reference/libraries/overview

des Integrationstools von Dialogflow kann ein erstellter Agent auf verschiedene Plattformen gelangen. Unterstützt werden dabei unter anderem Google Assistant, Amazon Alexa, Microsoft Cortana, Facebook Messenger, Slack, Skype oder Twitter.

Sprache und Text Es werden Ein- und Ausgaben in Form von Text und Sprache unterstützt. Außerdem kann Text durch Zugriff auf die Text-to-Speech API[4] in Audio umgewandelt werden.

Lizenzen und Preise Dialogflow kann kostenlos mit Einschränkungen in der Standard Edition oder kostenpflichtig in der Enterprise Edition (Essentials oder Plus) genutzt werden[5].

Sonstiges Die Plattform bietet einige Besonderheiten, die erwähnenswert sind. Für einen Agent kann das integrierte Smalltalk-Feature aktiviert werden. Dadurch können lockere Gespräche mit dem Chatbot geführt werden, die von "How are you?" über "What's your name?" bis hin zu Gefühlsäußerungen wie "I am bored" reichen. Die jeweiligen Antworten müssen festgelegt werden. Außerdem bietet Dialogflow eine automatische Rechtschreibkorrektur, die bei Aktivierung vor dem Intent Matching erfolgt.

3.2 Rasa Stack

Der Rasa Stack ist ein Open-Source Framework zum Erstellen von Conversational Agents bestehend aus zwei Modulen, nämlich Rasa NLU und Rasa Core. Während Rasa NLU für das Verständnis von Nachrichten sorgt, ist die Steuerung der Konversation Aufgabe von Rasa Core. Die beiden Module können sowohl unabhängig als auch in der Kombination verwendet werden. Der modulare Aufbau des Rasa Stacks begünstigt die Integration mit anderen Geräten. [vgl. Ras19f]

Die Abbildung 3.4 zeigt die grundlegenden Schritte eines Rasa Dialogsystems bei der Verarbeitung einer Nachricht. Die empfangene Nachricht wird zunächst an einen Interpreter weitergeleitet, der die Nachricht analysiert, und infolgedessen ein Dictionary mit der Originalnachricht, den Intents und den Entities erstellt. Anschließend wird dem Tracker die Information übergeben, dass eine neue Nachricht verfügbar ist. Dieser verwaltet den Stand aller geführten Konversationen. Im nächsten Schritt erhält die Policy-Komponente den aktuellen Stand und entscheidet, welche Action als nächste ausgeführt wird. Die ausgewählte Action wird vom Tracker protokolliert und es wird eine Nachricht an die benutzende Person geschickt. Tracker, Policies und Actions bilden den Dialogmanager. [vgl. Ras19b]

[4] https://cloud.google.com/text-to-speech/docs
[5] https://dialogflow.com/pricing

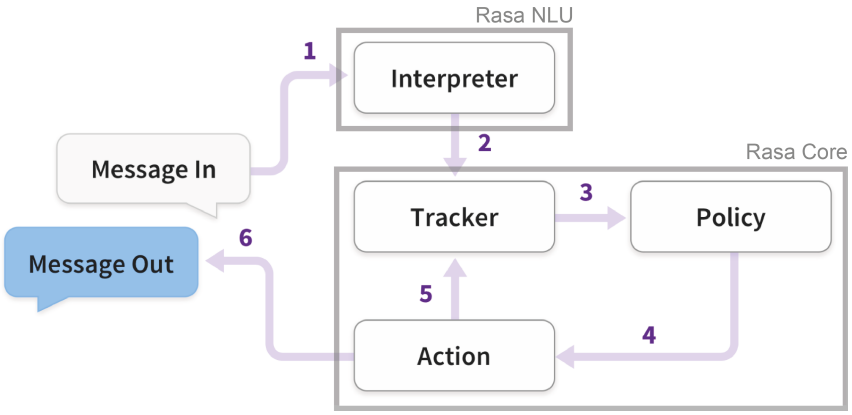


Abbildung 3.4: Nachrichtenverarbeitung des Rasa Stacks (Quelle: [Boc17, S. 3])

Der Rasa Stack verwendet - wie Dialogflow - Machine Learning Modelle, für deren Training vordefinierte Daten erforderlich sind. Für Rasa NLU wird eine Liste an Beispieleingaben benötigt, bei denen die Intents und Entities gekennzeichnet sind. Diese Liste kann entweder als JSON-Objekt oder im Markdown-Format bereitgestellt werden. Für Rasa Core sind die Trainingsdaten - sogenannte *Stories* - im Markdown-Format notwendig. [vgl. Boc17, S. 4]

Der Code ist in Python implementiert und sowohl Rasa NLU als auch Rasa Core können über eine HTTP-Schnittstelle angesprochen werden. [vgl. Ras19f]
Im Mai 2019 hat Rasa das neue Tool Rasa X[6] veröffentlicht, das dabei helfen soll, von echten Konversationen zwischen Mensch und Chatbot zu lernen und auf diese Weise bessere Daten für das Training zu erstellen.

3.2.1 Rasa NLU

Rasa NLU ist ein Open-Source Natural Language Processing Tool zur Intent Klassifizierung und Entity Extraktion und bündelt eine Vielzahl an NLP und Machine Learning Libraries in einer konsistenten API.

Eine sogenannte *Pipeline* definiert die Folge von Verarbeitungsschritten, welche bei der Bearbeitung eines Eingabetextes durch verschiedenen Komponenten eingehalten wird. Es gibt zum Beispiel Komponenten für Entity-Extraktion, Intent-Klassifizierung oder Pre-Processing. Mit Rasa NLU können zudem benutzerdefinierte Komponenten erstellt werden, die standardmäßig nicht geboten werden. In jedem Schritt der Pipeline wird die Eingabe bearbeitet und als Input für den nächsten Verarbeitungsschritt bereitgestellt. Die Pipeline kann individuell eingerichtet werden oder es kann auf von Rasa

[6] http://rasa.com/docs/rasa-x

NLU standardmäßig vordefinierte Pipelines zurückgegriffen werden, die für die meisten Anwendungsfälle gut funktionieren [vgl. Boc17, S. 3]. Die zwei wichtigsten von Rasa angebotenen Pipelines sind `supervised_embeddings` und `pretrained_embeddings_spacy` [vgl. Ras19c] und unterscheiden sich hinsichtlich sogenannter Wortvektoren. Die Idee von Wortvektoren ist es, ähnliche Wörter in einem n-dimensionalen Vektorraum zu repräsentieren. Das Training eines neuronalen Netzes resultiert darin, dass ähnliche Wörter möglichst nah beieinander liegen [vgl. GPJ18, S. 75]. Während die Pipeline `pretrained_embeddings_spacy` vortrainierte Wortvektoren von GloVe[7] oder fastText[8] nutzt, werden bei `supervised_embeddings` die Wortvektoren auf das eigene Datenset angepasst. Bei einer geringen Anzahl an Trainingsdaten wird empfohlen, die Pipeline `pretrained_embeddings_spacy` zu benutzen. Bei mehr als tausend Trainingsdaten hingegen kann die Pipeline `supervised_embeddings` verwendet werden, da die Wortvektoren an die eigene Domäne angepasst werden. [vgl. Ras19c]
Ein großer Vorteil von Rasa NLU besteht darin, dass grundsätzlich jede beliebige Sprache benutzt werden kann. Insbesondere mit der Pipeline `supervised_embeddings` kann jede beliebige Sprache verarbeitet werden, da die Wortvektoren entsprechend der eigenen Domäne gebildet werden. Die Pipeline `pretrained_embeddings_spacy` nutzt das NLP-Framework spaCy[9], welches ebenso über hundert Sprachen unterstützt. [vgl. Ras19g]

3.2.2 Rasa Core

Rasa Core ist ein Dialogmanager, der bestimmt, welche *Action* - basierend auf der Eingabe der Nutzer*innen - als nächste ausgeführt wird. Die Domäne sowie die *Intents*, *Entities*, *Slots* und *Actions* des Chatbots werden in einer Konfigurationsdatei erfasst. Außerdem werden Templates als Antworten des Chatbots festgelegt. [vgl. Ras19d]

Das Problem des Dialogmanagers sieht Rasa als Klassifikationsproblem [vgl. Boc17, S. 3]. Bei jeder Iteration (siehe Abbildung 3.4) versucht Rasa die nächste Action aus einer vordefinierten Liste vorherzusagen. Eine *Action* kann das Senden einer einfachen Nachricht sein, das Ausführen einer bestimmten Funktion, eines externen API-Calls oder einer externen Datenbank. [vgl. Ras19a]

Slots sind das Gedächtnis des Chatbots. In einem Key-Value Store können Information gespeichert werden, die während einer Konversation weiter genutzt werden. Slots können von der Rasa NLU (siehe Codeauszug 3.1), durch einen Klick auf einen Button oder von Actions gesetzt werden und können - abhängig vom Slot-Typen - Einfluss auf

[7] https://nlp.stanford.edu/projects/glove
[8] https://fasttext.cc
[9] https://spacy.io

die Vorhersage haben. Slots vom Typ `categorical` oder `bool` werden verwendet, wenn das Vorhandensein des Inhalts wichtig ist. Beim einfachen Speichern von Daten, die den Konversationsfluss nicht beeinflussen sollen, werden Slots vom Typ `unfeaturized` empfohlen. [vgl. Ras19i]
Falls mehrere Informationen gesammelt werden sollen, empfiehlt Rasa *Forms* zu verwenden. Forms sind einzelne Actions, die über die notwendigen Slots iterieren [vgl. Ras19e].

Anstatt einer Reihe von if/else Anweisungen, nutzt Rasa Core ein Machine Learning Modell, das basierend auf Beispielkonversationen trainiert wird. Ein Trainingsbeispiel wird von Rasa als *Story* bezeichnet und im Markdown-Format bereitgestellt. [vgl. Ras19j]

```
## story_greet
* greet
    - utter_name
* name{"name": "Anna"}
    - utter_greet
```

Codeauszug 3.1: Beispiel für eine Rasa-Story

Der Codeauszug 3.1 zeigt ein Beispiel für eine Story. Jede Story beginnt mit ## gefolgt von einem willkürlich gewählten Namen, der jedoch hilfreich beim Debugging sein kann. Nachrichten, die von Nutzer*innen geschickt wurden, sind mit * gekennzeichnet und entsprechen dem Format `intent{"entity1": "value", "entity2": "value"}`. Im Beispiel vom Codeauszug 3.1 wird zunächst angenommen, dass die Eingabe eine Begrüßung war. Mit `utter_name` wird nach dem Namen gefragt. Die Antwort "Anna" wird in dem Slot "name" gespeichert. Anschließend wird die Nutzerin mit ihrem Namen begrüßt. Das Ende einer Story wird mit einer Leerzeile gekennzeichnet.

3.2.3 Eigenschaften von Rasa

Im weiteren Verlauf werden die Merkmale des Rasa Stacks dargelegt:

Gebrauchstauglichkeit Rasa zielt mit dem Rasa Stack auf ein einfach benutzbares, aber auch anpassbares Framework ab. Für Anwender*innen ohne technisches Vorwissen stellt es eine Herausforderung dar, mit Rasa einen Chatbot zu erstellen, da es keine grafische Benutzeroberfläche gibt. Allerdings ist gerade die Möglichkeit der individuellen Anpassbarkeit ein Vorteil für spezifische Chatbots.

Unterstützte Sprachen Mit dem Rasa Stack können grundsätzlich Chatbots für jede beliebige Sprache erstellt werden, wobei die gewählte Pipeline die Anzahl der angebotenen Sprachen bestimmt. Mit der Pipeline `supervised_embeddings` kann jede beliebige Sprache gewählt werden. Die Pipeline `pretrained_embeddings_spacy` unterstützt hingegen nicht jede beliebige, allerdings auch über hundert Sprachen.

Natural Language Understanding Wie bei Dialogflow wird für das Natural Language Understanding Machine Learning benutzt. Die individuell konfigurierbaren Pipelines haben eine volle Flexibilität zur Folge. Mit den Pipelines `pretrained_embeddings_spacy` und `supervised_embeddings` können gute Ergebnisse für die meisten Anwendungsfälle erzielt werden.

Hosting und Verarbeitung Zur Nutzung des Rasa Stacks wird ein eigener lokaler oder Cloud Server benötigt.

Datensicherheit Der Rasa Stack garantiert volle Kontrolle über die Daten. Rasa NLU und Rasa Core können als HTTP-Server betrieben werden. Für die Sicherheit des Servers sind die Betreiber*innen zuständig. Rasa empfiehlt grundsätzlich, den Server nicht öffentlich zugänglich zu machen, sondern über das Backend eine private Verbindung herzustellen.

Schnittstellen und Integration Es gibt eine SDK für Python und eine HTTP-Schnittstelle. Außerdem können die erstellten Chatbots für verschiedene Plattformen und Messenger, unter anderem für den Facebook Messenger, Slack oder Telegram, verfügbar gemacht werden.

Sprache und Text Rasa akzeptiert standardmäßig nur Textein- und -ausgaben, jedoch können Systeme zur Spracherkennung über die HTTP-Schnittstelle eingebunden werden.

Lizenzen und Preise Der Rasa Stack ist Open-Source und kostenlos nutzbar. Rasa X kann ebenfalls kostenlos genutzt werden, ist jedoch nicht Open-Source. Die Nutzung von Rasa X für eigene kommerzielle Chatbots ist erlaubt, es ist hingegen verboten, den Quellcode zu verändern oder Rasa X zu hosten. Für Unternehmen und großtechnische Lösungen bietet Rasa mit Rasa X eine kostenpflichtige Variante.

Sonstiges Die volle Kontrolle über die Daten und die Möglichkeit zur Anpassung an die eigenen Bedürfnisse machen den Rasa Stack besonders interessant. Darüber hinaus ist der Stack Open-Source. Mit dem Interactive Learning Modus[10] kann dem Chatbot interaktiv Feedback gegeben werde, wodurch ein Fehlverhalten direkt angepasst werden kann.

[10] https://rasa.com/docs/rasa/core/interactive-learning

3.3 Zusammenfassung

Im Folgenden werden die Eigenschaften der vorgestellten Tools Dialogflow und Rasa Stack zusammengefasst und verglichen.

	Dialogflow	**Rasa Stack**
Anbieter	Google	Rasa
Programmiersprache	Nicht bekannt	Python
Gebrauchstauglichkeit	Sehr gut	Moderat
GUI zur Dialogerstellung	Ja	Nein
Natural Language Understanding	Ja	Ja (Rasa NLU)
Machine Learning Modelling	Ja	Ja
Hosting	Cloud Hosting, lokal nicht möglich	Lokale Installation oder auf eigenem Cloud Server
Datensicherheit	Daten sind in der Google Cloud gespeichert	Daten werden auf eigenem lokalen Server oder privatem Cloud Server gespeichert
Integration mit anderen Diensten	Ja	Ja
Schnittstellen	HTTP-API und SDK für u.a. Java, Python, Node.js	HTTP-API und SDK für Python
Anzahl unterstützter Sprachen	Mehr als 20 (darunter Englisch)	Abhängig von gewählter Pipeline, aber grundsätzlich wird jede gewünschte Sprache unterstützt
Open-Source	Nein	Ja
Integriertes Smalltalk-Modul	Ja	Nein
Interactive Learning Modus	Nein	Ja

Tabelle 3.1: Vergleich von Dialogflow und Rasa Stack

Aus der Tabelle 3.1 ist zu entnehmen, dass sowohl Dialogflow als auch der Rasa Stack Vor- und Nachteile bergen. Dialogflow punktet hinsichtlich der einfachen Bedienbarkeit für Nutzer*innen ohne technisches Hintergrundwissen, da keine lokale Installation notwendig ist und die Steuerung über eine grafische Benutzeroberfläche erfolgt. Überdies können erstellte Chatbots unkompliziert auf anderen Plattformen verfügbar gemacht werden. Zudem unterstützt Dialogflow eine Vielzahl von Sprachen und erlaubt standardmäßig die Ein- und Ausgabe in Form von Text und Sprache. Es macht, wie der Rasa Stack, von Machine Learning Modellen beim Natural Language Understanding

Gebrauch. Allerdings ist es nicht möglich, einen eigenen Server zu betreiben. Die Daten werden in der Google Cloud gespeichert und verarbeitet.

Beim Rasa Stack ist vor allem hervorzuheben, dass er quelloffen ist und Betreiber*innen die volle Kontrolle über die Daten haben. Ferner ist der Stack modular aufgebaut, sodass er einfach mit anderen Systemen integriert werden kann. Ein weiterer Vorteil sind die Anpassungsmöglichkeiten, wodurch mit dem Rasa Stack Chatbots für spezifische Anwendungsfälle erstellt werden können. Für den Betrieb wird jedoch technisches Knowhow vorausgesetzt, da es keine grafische Benutzeroberfläche gibt und ein Server lokal oder in einer privaten Cloud betrieben werden muss.

4 Anforderungsanalyse

Das Ziel der Anforderungsanalyse besteht darin, die Anforderungen an den Chatbot zu identifizieren und zu definieren. Im folgenden Kapitel wird zunächst ein fiktives Szenario beschrieben, in dem ein solcher Chatbot sinnvoll eingesetzt werden kann. Darauffolgend werden die Vorgehensweise der Anforderungsermittlung erläutert und die funktionalen und nicht-funktionalen Anforderungen definiert.

4.1 Anwendungsszenario

Alice möchte ihre Englischkenntnisse verbessern, weil sie demnächst ein Semester im Ausland studieren wird. Ihr Sprachniveau liegt etwa bei B1. Sie hat das Gefühl, dass sie zwar bereits ein gutes Basisvokabular besitzt und sie vieles in der Sprache versteht, allerdings fällt ihr das Führen von Konversationen schwer, weil ihr die Übung fehlt. Sie hat eine Plattform gefunden, welche Sprachtandems vermittelt. Ein Sprachtandem beschreibt zwei Personen, die die Sprache des*der jeweils anderen in persönlichen Zusammentreffen lernen möchten. Nach dem ersten Treffen kommt sie etwas ernüchtert nach Hause. Sie hat sich unwohl gefühlt, weil sie ständig Angst hatte, Fehler in der Anwesenheit eines Muttersprachlers zu machen. Für einen Sprachkurs fehlt ihr die Zeit. Sie ist Studentin und hat einen vollen Stundenplan. Als sie den Sprachlern-Chatbot "EnglishBot" findet, erhofft sie sich, dadurch zeit- und ortsunabhängig lernen zu können. Des Weiteren möchte sie gern in unregelmäßigen Abständen selbstständig vom Chatbot kontaktiert werden, damit sie das Lernen nicht vergisst. Ihr Ziel ist es, sich in den Konversationen so sicher zu fühlen, dass sie ohne Bedenken an Sprachtandems teilnehmen kann.

4.2 Anforderungsermittlung

Zur Ermittlung der Anforderungen an den Chatbot wurden Interviews mit Englischlehrenden geführt. Die Interviews zielten auf die Identifikation der Einsatzbereiche sowie der Eigenschaften und Funktionen eines Chatbots im Zweitspracherwerb ab (siehe Interviewleitfaden im Anhang B). Es wurden insgesamt fünf Interviews mit Sprachlehrer*innen im Alter von 25 - 64 Jahren geführt. Drei der fünf Lehrenden unterrichten an einer Hochschule fachspezifisches Englisch, die anderen sind in der Erwachsenenbildung tätig. Der Großteil der Lehrenden hat bisher noch nie bewusst einen Chatbot benutzt,

allerdings wird einheitlich eine große Chance in der Nutzung von technischen Geräten in der Bildung und insbesondere beim Lernen einer Sprache gesehen.

Im Folgenden werden die in den Interviews gewonnenen Erkenntnisse zusammengefasst. Die Transkriptionen der Interviews sind auf dem der Arbeit beiliegenden Speichermedium im Ordner `interviews` zu finden.

Mögliche Anwendungsbereiche
Chatbots können vor allem beim Lernen von Vokabeln und Grammatik sinnvoll einsetzbar sein. Darüber hinaus ist es vorstellbar, dass Chatbots als digitale Tandempartner*innen agieren. Auch für die schriftliche Kommunikation, wie zum Beispiel Messaging, können Chatbots interessant sein. Durch die Simulation von Situationen aus dem echten Leben können Konversationen geübt werden. Kritisch gesehen wird die mündliche Kommunikation mit Chatbots, da dabei wesentliche Bestandteile, wie beispielsweise Mimik und Gestik, fehlen. Zudem wird die Aneignung der Aussprache durch die alleinige Nutzung eines Chatbots als schwierig angesehen. Das Anhören von Sprachnachrichten kann sich positiv auswirken, jedoch kann die produktive Fertigkeit des Sprechens in Bezug auf die Aussprache nur mit Feedback und Korrektur von Sprecher*innen auf C2-Niveau im direkten Kontakt erlangt werden.

Eigenschaften und Funktionen
Als besonders wichtig werden die natürliche Kommunikation und möglichst menschennahe Erscheinung des Chatbots erachtet. Smalltalk-Fähigkeiten können sich hinsichtlich der Sympathie und Menschlichkeit positiv auswirken. Wie bereits im vorangegangenen Punkt erwähnt, wird zwar die Eingabe durch Sprache kritisch gesehen, die Sprachausgabe kann hingegen einen lernförderlichen Effekt haben.

Eine Korrektur von Fehlern macht insbesondere dann Sinn, wenn diese Fehler zu Unverständnis führen. Bei fehlerhaften oder sprachlich verbesserungsfähigen Formulierungen kann es hilfreich sein, wenn weitere Formulierungsmöglichkeiten vorgeschlagen werden. Feedback in Form von Alternativformulierungen kann der Erweiterung des Wortschatzes von Lernenden dienen. Vereinzelt wird es gut geheißen, die Häufigkeit des Feedbacks individuell festlegen zu können.

Die Einbindung eines einsprachigen Wörterbuchs wird einheitlich für hilfreich befunden, da bei unklaren Vokabeln die Anwendung nicht gewechselt werden muss. Drei der fünf befragten Personen fänden es wünschenswert, dass das gesuchte Wort im Kontext mit einem Beispielsatz erklärt wird. Lediglich zwei der fünf interviewten Personen können sich ein zweisprachiges Wörterbuch als hilfreich vorstellen, und auch nur in den Fällen, in denen sich Lernende auf einem niedrigen Sprachniveau befinden. Grund hierfür ist, dass die Zielsprache beim Erlernen einer Sprache möglichst nicht verlassen werden soll. Dies würde bei der Nutzung eines zweisprachigen Wörterbuchs ständig eintreten.

Ferner legt der Großteil der Personen Wert auf eine Individualisierbarkeit, etwa durch Anpassung der Anwendung an den Lernfortschritt der Nutzer*innen oder durch die Festlegung eines täglichen Lernziels. Darüber hinaus soll laut vier Personen der Lernfortschritt transparent dokumentiert werden, damit sich die Lernenden - insbesondere als Ansporn - dessen bewusst sind. Regelmäßige Benachrichtigungen des Chatbots durch Kontaktieren der Nutzer*innen werden mehrheitlich ebenfalls als motivierend erachtet, jedoch sollte die Häufigkeit individuell festgelegt werden können.

Über dem Chatten hinaus wird von der Gesamtheit der Befragten ein Lernmodus für Vokabeln bzw. für Grammatik für sinnvoll gehalten. Vokabeln und Grammatik könnten thematisch gegliedert werden und einen weiteren Anreiz für Lernende darstellen. Eine befragte Lehrkraft fände die Integration eines Verbkonjugators in die Anwendung nützlich.

Die Abbildung 4.1 fasst die identifizierten Eigenschaften und Funktionen zusammen, welche nach der Häufigkeit der Nennung sortiert werden. Die Länge des Balkens zeigt, von wie vielen Lehrenden diese Anforderung genannt wurde.

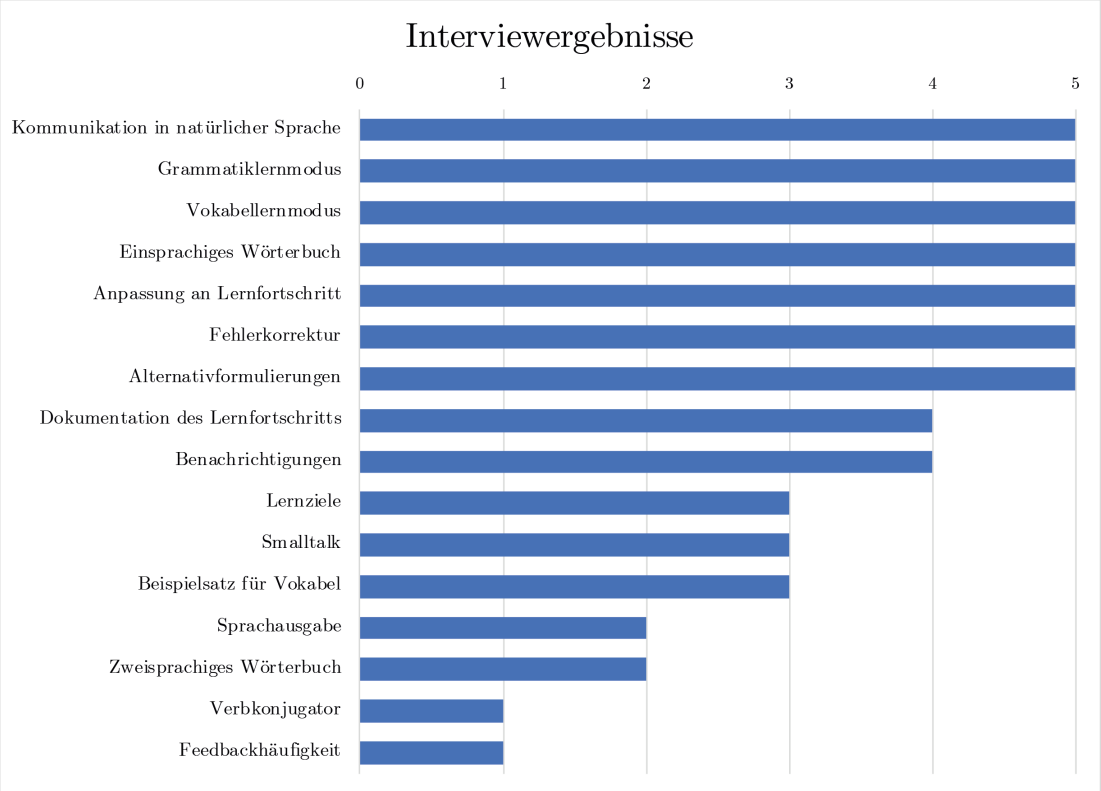


Abbildung 4.1: Übersicht der Interviewergebnisse (Quelle: eigene Darstellung)

Für das weitere Vorgehen werden die Anforderungen ausgewählt, die von der Mehrheit der Lehrenden für gut befunden wurden. Damit werden die folgenden Eigenschaften

und Funktionen nicht weiter berücksichtigt: Sprachausgabe, zweisprachiges Wörterbuch und Verbkonjugator. Eine Ausnahme gilt für die Einstellung der Häufigkeit des Feedbacks, die trotz der niedrigen Nennung weiter berücksichtigt wird. Die Individualisierbarkeit ist ein Grundsatz der Dialoggestaltung nach der Norm ISO 9241-110, welche Prinzipien für die Mensch-Computer-Interaktion beschreibt. Der Grammatiklernmodus wurde zwar von allen Lehrenden für allgemeine Sprachlernanwendungen als hilfreich eingeschätzt. Da sich die Anwendung jedoch auf die Konversationskenntnisse und damit auf die Erweiterung des Wortschatzes fokussiert, wird dieser ebenfalls nicht weiter berücksichtigt.

4.3 Anforderungen

Basierend auf den Ergebnissen der Interviews werden im weiteren Verlauf das Anwendungsziel, die Rahmenbedingungen und die Anforderungen definiert. Die Gewichtung der einzelnen Anforderungen richtet sich nach der Anzahl der Nennung in den geführten Interviews. Eine häufige Nennung resultiert somit in einer höheren Gewichtung. Die Skala der Gewichtungsstufen reicht von "niedrig" über "mittel" bis zu "hoch".

Anwendungsziel
Die Applikation richtet sich vor allem an Englisch-Lernende auf dem Sprachniveau B1 des Europäischen Referenzrahmens[1]. Durch die regelmäßige Nutzung der Anwendung sollen die Englisch-Sprachkenntnisse gefestigt und verbessert werden. Die wesentlichen Ziele sind die Erweiterung des Wortschatzes durch das Training von Vokabeln sowie dem Üben von Konversationen zu bestimmten Themen. Regelmäßige Gespräche mit dem Chatbot sollen Hemmungen und Ängste abbauen, die in der Kommunikation in realen Situationen auftreten können.

Rahmenbedingungen
Die Rahmenbedingungen legen die organisatorischen und technischen Einschränkungen der Anwendung fest. Die Applikation dient dem Selbststudium und zielt auf eine spezielle Gruppe an Nutzer*innen ab (siehe Anwendungsziel). Die Eingabe findet rein textbasiert statt. Darüber hinaus können keine beliebigen Konversationsthemen umfasst werden. Vielmehr liegt der Fokus auf dem Thema "University: Get ready for an exchange year". Es besteht die Möglichkeit der beliebigen Erweiterung von Konversationsthemen. Die Anwendung ist nur in Englisch verfügbar und benötigt einen Zugang zum Internet.

[1] https://www.coe.int/en/web/common-european-framework-reference-language

Funktionale Anforderungen

Funktionale Anforderungen legen fest, welche Funktionen oder Services vom Softwaresystem bereitgestellt werden [vgl. Bal09, S. 456]. Sie können in Anforderungen, welche die Statik, die Dynamik oder die Logik des Systems beschreiben, gegliedert werden. Die Anforderungen dieser Arbeit orientieren sich an einer Schablone für natürlichsprachliche Anforderungen [vgl. Bal09, S. 484]. Die Verwendung solcher Schablonen hilft dabei, syntaktische und semantische Mehrdeutigkeiten zu reduzieren. Weiterhin ist die Relevanz der jeweiligen Anforderung durch die Verwendung von Wörtern wie "muss", "kann" und "sollte in Zukunft" sofort ersichtlich. [vgl. Bal09, S. 481 - 483]

Jeder Anforderung wird eine eindeutige Identifikationsnummer sowie ein Kurztitel (fettgedruckt) zugewiesen, die der Abgrenzung von anderen Funktionen dienen. Überdies wird die aus den Interviews abgeleitete Gewichtung aufgezeigt.

Zunächst werden die Anforderungen aufgelistet, die für die Anwendung unabdingbar sind:

ID	Anforderung	Gewichtung
F01	**Vokabellernmodus:** Die Anwendung muss textbasierte Vokabelübungen zum Thema "University: get ready for an exchange year" anbieten.	hoch
F02	**Fehlerkorrektur:** Die Anwendung muss Rechtschreib- und Grammatikfehler korrigieren.	hoch
F03	**Alternativformulierungen:** Die Anwendung muss bei fehlerhaften Formulierungen Alternativformulierungen anbieten.	hoch
F04	**Smalltalk:** Die Anwendung muss Smalltalk beherrschen.	mittel
F05	**Einsprachiges Wörterbuch:** Die Anwendung muss im Stande sein, den Nutzer*innen unbekannte Begriffe zu erklären.	hoch

Tabelle 4.1: Funktionale *muss*-Anforderungen

In der Tabelle 4.2 werden die Anforderungen aufgezeigt, die zwar nicht zwingend notwendig sind, allerdings als wünschenswert und sinnvoll erachtet werden.

ID	Anforderung	Gewichtung
F06	**Benachrichtigungen:** Die Anwendung soll regelmäßige Benachrichtigungen an die Nutzer*innen schicken, wobei die Häufigkeit einstellbar sein soll.	hoch
F07	**Dokumentation des Lernfortschritts:** Die Anwendung soll den Lernfortschritt transparent dokumentieren.	hoch
F08	**Anpassung an Lernfortschritt:** Die Anwendung soll sich an den Lernfortschritt der Nutzer*innen anpassen.	hoch
F09	**Lernziele:** Die Anwendung soll ein einstellbares tägliches Lernziel haben.	mittel

Tabelle 4.2: Funktionale *soll*-Anforderungen

Schließlich beschreiben die nachfolgenden Anforderungen Möglichkeiten zur Erweiterung der Anwendung.

ID	Anforderung	Gewichtung
F10	**Feedbackhäufigkeit:** Die Anwendung sollte in Zukunft eine Einstellung für die Häufigkeit des Korrekturfeedbacks ermöglichen.	niedrig
F11	**Beispielsatz für Vokabel:** Die Anwendung sollte in Zukunft unbekannte Begriffe mit einem Beispielsatz erklären können.	niedrig

Tabelle 4.3: Funktionale *kann*-Anforderungen

Nicht-funktionale Anforderungen

Die in der Tabelle 4.4 dargestellten nicht-funktionalen Anforderungen beschreiben die Eigenschaften der Anwendung. Sie betreffen typischerweise mehrere oder alle funktionalen Anforderungen [vgl. Bal09, S. 463] und werden ebenfalls fortlaufend mit einer eindeutigen Identifikationsnummer gekennzeichnet.

ID	Anforderung	Gewichtung
NF01	**Natürliche Sprache:** Die Anwendung muss mit den Nutzer*innen in natürlicher Sprache in Englisch agieren.	hoch
NF02	**Intuitive Benutzeroberfläche:** Die Benutzeroberfläche der Anwendung muss intuitiv gestaltet sein.	hoch
NF03	**Unerwartete Eingaben:** Die Anwendung muss mit unerwarteten Eingaben umgehen können.	hoch
NF04	**Modulare Lerninhalte:** Lerninhalte müssen modular aufgebaut und erweiterbar sein.	hoch
NF05	**Hilfestellung:** Die Anwendung soll bei der Benutzung Hilfestellung leisten.	mittel
NF06	**Dialogspeicherung:** Die Dialoge sollen anonymisiert zur Verbesserung des Systems gespeichert werden.	hoch

Tabelle 4.4: Nicht-funktionale Anforderungen

4.4 Zusammenfassung

In Interviews mit Sprachlehrenden konnten mögliche Anwendungsbereiche sowie Eigenschaften und Funktionen eines Chatbots zum Lernen einer Sprache herausgearbeitet werden. Das Ergebnis der Anforderungsanalyse sind funktionale und nicht-funktionale Anforderungen, die die Basis für das Konzept der Anwendung schaffen.

5 Konzept des EnglishBot

Das folgende Kapitel befasst sich mit dem Konzept der Anwendung, welches sich aus den in Kapitel 4 identifizierten Anforderungen ableitet. Zunächst werden Use-Cases beschrieben. Im Anschluss daran wird der grundsätzliche Ablauf der Anwendung dargelegt, die Lerninhalte und die Datenhaltung spezifiziert sowie ein Entwurf der Architektur und Benutzeroberfläche ausgearbeitet. Bei der Konzipierung des EnglishBot wurde nach der User-Centered Design (UCD) Methode vorgegangen, bei der drei zukünftige Nutzer*innen bereits in den Entwicklungsprozess miteinbezogen wurden. Das Ziel von UCD besteht darin, eine Anwendung zu erstellen, die den Bedürfnissen und Erwartungen jener entspricht [vgl. Low13, S. 5].

5.1 Use-Cases

Die Anwendung "EnglishBot" zielt auf Studierende ab, die ihre Englischkenntnisse verbessern möchten. Der Fokus liegt dabei auf der Erweiterung des Wortschatzes, da dieser die Basis jeglicher Kommunikation schafft. In die Anwendung integriert ist ein Chatbot, welcher in englischer Sprache kommuniziert. Die Abbildung 5.1 fasst die Use-Cases des EnglishBot zusammen.

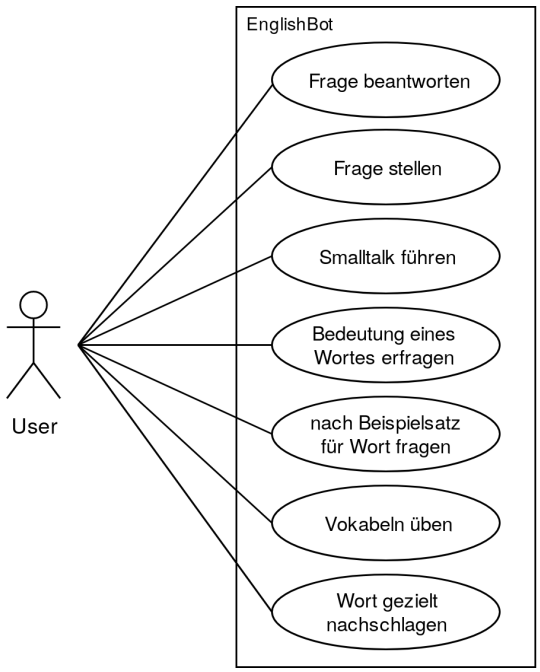


Abbildung 5.1: Anwendungsfälle EnglishBot (Quelle: eigene Darstellung)

Use-Cases: Frage stellen, Frage beantworten und Smalltalk führen
Die wesentliche Aufgabe des Chatbot ist die Kommunikation in Englisch. Nutzer*innen sollen Fragen in Englisch stellen können, die anschließend vom Chatbot beantwortet werden. Ebenso sollen sie Fragen beantworten können, die vom Chatbot gestellt wurden. Der Chatbot soll fähig sein, lockere Gespräche (Smalltalk) führen zu können. Überdies soll auf Eingaben wie "help" oder "I don't know what to do now" entsprechend reagiert und Hilfestellung geleistet werden.

Use-Case: Bedeutung eines Wortes erfragen ("Wörterbuchsuche")
Im Rahmen der Kommunikation mit dem Chatbot sollen natürlich formulierte Fragen nach einem Wort erkannt werden und die Bedeutung(en) des Wortes zurückgeliefert werden. Folgende oder ähnlich formulierte Fragen könnten zum Beispiel gestellt werden:

- What is the meaning of the word "compulsory"?
- What does "counsellor" mean?
- What is a degree?

Eine möglichst große Vielfalt an Fragemöglichkeiten ist erstrebenswert.

Use-Case: nach Beispielsatz für Wort fragen
Weiterhin soll es möglich sein, nach einem Beispielsatz für ein bestimmtes Wort zu fragen, wodurch das Lernen eines Wortes in dessen Kontext begünstigt wird. Die Frage nach einem Satz könnte beispielsweise folgendermaßen lauten: Give me a sentence with the word "schedule".

Use-Case: Vokabeln üben ("Vokabelquiz")
Vokabeln sollen während der Kommunikation mit dem Chatbot geübt werden können, indem durch bestimmte Aussagen wie zum Beispiel "I want to learn some new words" oder "Teach me some vocabulary" in das Vokabelquiz gewechselt werden kann. Es folgt eine Antwort mit einer Liste der möglichen Themen, die der Chatbot beherrscht. Nach Auswahl des Themas soll der Chatbot mit Fragestellungen, die die Bedeutung eines Wortes beinhalten, einzelne Wörter - ähnlich wie bei einem Quiz - nacheinander abfragen.

Use-Case: Wort gezielt nachschlagen
Zusätzlich zum Üben von Vokabeln im Chatmodus soll es die Möglichkeit geben, in einem gesonderten Modus Vokabeln thematisch geordnet nachzuschlagen und dadurch gezielt wiederholen zu können. Die Vokabeln einer Kategorie entsprechen einer alphabetisch geordneten Liste.

Ausgehend von den genannten Use-Cases kann in der Anwendung zwischen dem Chatmodus, in welchem die Kommunikation mit dem Chatbot und das Vokabelquiz stattfinden, und dem Vokabelmodus, in dem Wörter gezielt nachgeschlagen werden können, unterschieden werden.

Die Abbildung 5.2 visualisiert den grundsätzlichen Ablauf bei der Kommunikation mit dem Chatbot.

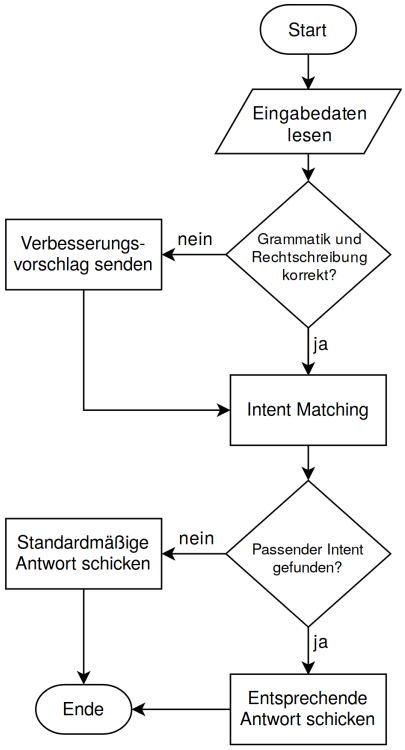


Abbildung 5.2: Grundsätzlicher Ablauf bei einer eingehenden Nachricht (Quelle: eigene Darstellung)

Zunächst wird der eingegebene Text auf Grammatik- und Rechtschreibfehler geprüft. Bei einer fehlerhaften Nachricht wird ein Verbesserungsvorschlag an die Nutzer*innen geschickt und erst anschließend auf Intents untersucht. Bei einer fehlerfreien Nachricht entfällt das Senden eines Verbesserungsvorschlags. Wird ein passender Intent gefunden, wird die entsprechende Antwort gesendet. Andernfalls wird eine standardmäßige Nachricht zurückgeschickt.

5.2 Lerninhalte

Da der Fokus auf der Erweiterung des Wortschatzes liegt, nimmt das Erlernen von Vokabeln eine zentrale Rolle ein. Vokabeln werden als digitale Karteikarten organisiert.

Jede Karteikarte enthält die dem Wort zugehörige Definition und Kategorie, einen POS Tag sowie Beispielsätze und Beispielfragen. Ein modularer Aufbau, der aus der Kategorisierung der Vokabeln resultiert, ermöglicht eine einfache Erweiterbarkeit, aber auch eine Struktur, anhand derer Lernerfolge gemessen werden können. Aus der Anforderungsanalyse ging insbesondere hervor, dass Lernziele ein wichtiges Mittel zur Motivation sind. Der Einsatz der Lernmethode "Spaced Repetition"(auch Distributed Practice genannt, deutsch: verteiltes Lernen) verhilft den Studierenden dabei, sich durch regelmäßiges Wiederholen Wörter zu merken. Dieser Methode liegt der sogenannte "Spacing Effect" zugrunde, der besagt, dass es effizienter ist, Wissen über einen längeren Zeitraum hinweg verteilt zu lernen, als konzentriert innerhalb eines kurzen Zeitraums [vgl. Sha77]. Diese Theorie ist angelehnt an die "Forgetting Curve" (deutsch: Vergessenskurve). Der deutsche Psychologe Hermann Ebbinghaus fand heraus, dass der Prozess des Vergessens von Wissen bereits kurz nach dem ersten Lernen beginnt. Diesem Effekt kann durch Wiederholen entgegengewirkt werden (siehe Abbildung 5.3).

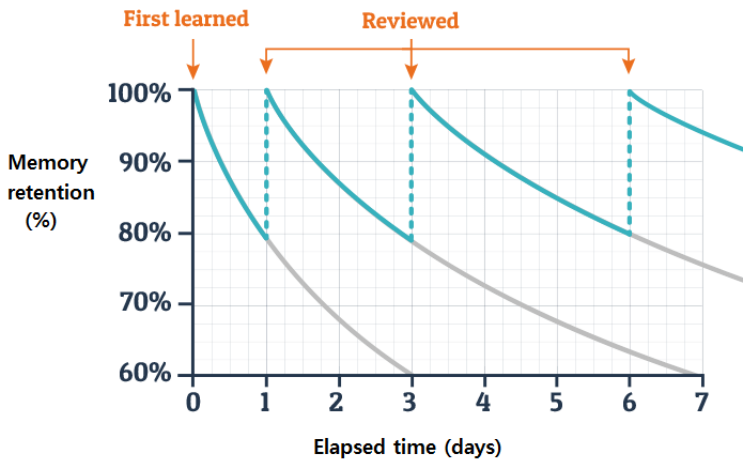


Abbildung 5.3: Abflachung der Vergessenskurve nach Ebbinghaus (Quelle: [CH18, S. 57])

Das sogenannte Leitner-System (siehe [Lei08]) ist eine vereinfachte Version des verteilten Lernens. Karteikarten werden auf einen Lernkarteikasten mit mindestens vier Boxen aufgeteilt. Jede Box wird an einem bestimmten Tag wiederholt. Karteikarten in der ersten Box werden täglich wiederholt, die in der zweiten Box jeden vierten Tag, die in der dritten Box jeden siebten Tag und die in der vierten Box alle zwei Wochen. Zu Beginn befinden sich alle Karteikarten in der ersten Box. Die erlernten Karteikarten wandern in die zweite Box, während nicht erlernte in der ersten Box verbleiben. Am zweiten und dritten Tag werden ebenfalls nur Wörter aus der ersten Box wiederholt. Am vierten Tag werden sowohl Karteikarten aus der ersten als auch aus der zweiten Box wiederholt. Karteikarten mit noch nicht beherrschten Inhalten wandern in die vorhergehende Box. [vgl. Sch15, S. 3]

Zahlreiche Sprachlernanwendungen nutzen das Leitner-System als Lernmethode, darunter auch Duolingo. Eine Anwendung wie der EnglishBot hat durch die Implementierung des Systems die Möglichkeit, die Vokabeln dem individuellen Lernfortschritt der Nutzer*innen anzupassen. Das Wissen wird messbar und kann zur Definition von Lernzielen herangezogen werden.

Im Rahmen dieser Arbeit wird der Schwerpunkt auf universitätsspezifisches Vokabular gelegt, da sich die Anwendung insbesondere an Studierende richtet, die ein Auslandssemester beabsichtigen. Die Wörter und Definitionen, welche aus unterschiedlichen Englisch-Lehrbüchern ([Red97], [MO08]) und Online-Quellen ([Exc19], [Tor19]) - darunter das Glossar der University of Toronto - stammen, wurden speziell für die Anwendung zusammengetragen (siehe `university_vocabulary.xlsx` auf beiliegendem Speichermedium).

5.3 Datenhaltung

Ein Datenmodell beschreibt, welche Daten einer Anwendung in einer Datenbank gespeichert werden, und wie deren Beziehungen untereinander sind [vgl. MK16, S. 25]. Grundsätzlich kann zwischen relationalen und nicht-relationalen Datenbanken unterschieden werden. Als bekanntestes Beispiel für relationale Datenbanken gilt MySQL[1]. Dabei werden Daten in Form von Tabellen gespeichert [vgl. MK16, S. 3]. Im Gegensatz dazu verfolgen nicht-relationale Datenbanken einen Ansatz ohne Tabellen. Daten werden beispielsweise in Spalten, Dokumenten oder Graphen gespeichert [vgl. MK16, S. 15]. Nicht-relationale Datenbanken überzeugen vor allem durch ihre Leistungsstärke und Flexibilität, da für diese im Vorfeld keine Struktur definiert werden muss. Relationale Datenbanken stoßen bei umfangreichen Datenmengen schnell an ihre Grenzen und sind unflexibler in der Datenverarbeitung als nicht-relationale Datenbanken [vgl. MK16, S. 222].
Bei einer Anwendung wie dem EnglishBot stehen Leistungsstärke und Flexibilität an oberster Stelle. Lerninhalte müssen einfach austausch- und erweiterbar sein. Darüber hinaus werden im Laufe der Benutzung der Anwendung immer mehr Daten pro Nutzer*in gespeichert, welche für den individuellen Lernfortschritt ausgewertet werden. Aus den genannten Gründen wird der Einsatz einer nicht-relationalen Datenbank zur Speicherung der Nutzer*innendaten und Lerninhalten als vorteilhafter bewertet. Eine weit verbreitete nicht-relationale Datenbank ist MongoDB[2], welche einen dokumentenorientierten Ansatz verfolgt, bei dem Daten in strukturierten Datensätzen gespeichert werden.

[1] https://www.mysql.com
[2] https://www.mongodb.com

Sie werden "Dokumente" genannt und bestehen aus einer Menge an Key-Value-Paaren, die beliebig verschachtelt sein können. [vgl. MK16, S. 229]

Speicherung der Lerninhalte

Vokabeln sollen in Form von digitalen Karteikarten gespeichert werden, die einer bestimmten Kategorie angehören. Der Codeauszug 5.1 zeigt, welche Daten eine Kategorie umfassen kann. Neben einer eindeutigen Identifikationsnummer *_id* wird die Bezeichnung der Kategorie (*name*) sowie die der Kategorie zugehörigen Vokabeln (*words*) gespeichert.

```
{
  "_id" : <object-id>,
  "name" : String,
  "words" : [ <flashcard-id>, <flashcard-id>, ... ],
}
```

Codeauszug 5.1: Dokument: *Category*

Der Codeauszug 5.2 zeigt die Inhalte einer Vokabel-Karteikarte (englisch: flashcard). Neben einer eindeutigen Identifikationsnummer *_id* wird der POS Tag (*pos*) gespeichert, welcher in Hinblick auf die Mehrdeutigkeit von Wörtern wichtig ist. Wörter können mehrere Bedeutungen haben und dabei unterschiedlichen Wortarten angehören. Überdies werden das Wort bzw. die Wörter (*words*), falls es mehrere Möglichkeiten gibt, und die zugehörige Definition (*definition*) abgebildet. Eine individuelle Festlegung von Fragen nach dem jeweiligen Wort (*questions*) sollen für Abwechslung im Quiz sorgen, während eine Auflistung von Beispielsätzen (*example_sentences*) die Anwendung des Wortes verdeutlichen soll.

```
{
  "_id" : <object-id>,
  "pos" :  String,
  "words" : [ String, String, String, ... ],
  "definition" : String,
  "questions" : [ String, String, String ... ],
  "example_sentences" : [ String, String, String, ... ]
}
```

Codeauszug 5.2: Dokument: *Flashcard*

Speicherung der Nutzer*innendaten

Im Folgenden werden die Dokumente aufgelistet, die für den EnglishBot zur Speicherung der Daten benötigt werden. Das Dokument *User* (siehe Codeauszug 5.3) umfasst alle Daten, welche die Nutzer*innen betreffen. Neben einer individuellen Identifikationsnummer *_id* und dem Namen des*der Nutzer*in (*name*) werden der letzte Kontakt

(*last_contact*) sowie die persönlichen Einstellungen (*settings*) gespeichert. Darüber hinaus ist die Dokumentation des Lernfortschritts vorgesehen.

Der Name zählt zu den personenbezogenen Daten. Diese sind besonders schutzwürdig, sodass sie zum Beispiel durch die Anwendung einer Hashfunktion pseudonymisiert werden müssen. Unter Pseudonymisierung versteht man nach Art. 4 Nr. 5 Datenschutz-Grundverordnung (DSGVO) "[...] die Verarbeitung personenbezogener Daten in einer Weise, dass die personenbezogenen Daten ohne Hinzuziehung zusätzlicher Informationen nicht mehr einer spezifischen betroffenen Person zugeordnet werden können [...]".

```
{
  "_id" : <individual-user-id>,
  "name" : gehashter String
  "last_contact" : Timestamp,
  "settings" : {
      "notify_daily" : Boolean,
      "notify_weekly" : Boolean,
      "error_correction_all" : Boolean,
      "error_correction_incomprehension" : Boolean
      "daily_dialogues" : Integer,
      "daily_words" : Integer,
  },
  "learning_model" : <learningmodel-id>,
}
```

Codeauszug 5.3: Dokument: *User*

Das Dokument *Learning Model* (siehe Codeauszug 5.4) umfasst alle Daten, die den individuellen Lernfortschritt betreffen. Es ist vorgesehen, geführte Dialoge und gelernte Wörter zu zählen. Ein Dialog entspricht einer gesendeten Nachricht, die vom Chatbot beantwortet wurde. Ein Wort hingegen ist dann erlernt, wenn es - gemäß dem Leitner-System - in die nächste Box gerückt wird. Im wöchentlichen Fortschritt wird die Summe an geführten Dialogen und gelernten Wörtern visualisiert. Jede*r Nutzer*in führt vier Leitner-Boxen *leitner_box_1* bis *leitner_box_4*, in denen die jeweiligen Vokabel-Karteikarten referenziert werden. Das Datum, an dem eine Karte hinzugefügt wurde, ist für die nächste Wiederholung ausschlaggebend. Darüber hinaus werden die Vokabeln, die für den bestimmten Tag vorgesehen sind, täglich durch eine Funktion ermittelt.

```
{
  "_id" : <object-id>,
  "words_total" : Integer,
  "dialogues_total" : Integer,
  "weekly_progress" : [ Integer, Integer, Integer, Integer, Integer, ↩
      Integer, Integer ],
  "daily_flashcards" : [ <flashcard-id>, <flashcard-id>, ... ],
  "leitner_box_1" : [ {
      "flashcard" : <flashcard-id>,
      "added": Timestamp },
      ...
  ],
  "leitner_box_2" : [ {
      "flashcard" : <flashcard-id>,
      "added": Timestamp },
      ...
  ],
  "leitner_box_3" : [ {
      "flashcard" : <flashcard-id>,
      "added": Timestamp },
      ...
  ],
  "leitner_box_4" : [ {
      "flashcard" : <flashcard-id>,
      "added": Timestamp },
      ...
  ],
}
```

Codeauszug 5.4: Dokument: *LearningModel*

Der Großteil der Daten soll serverseitig gespeichert werden, wodurch die Anwendung selbst schlank gehalten werden kann und Lerninhalte jederzeit verändert werden können. Clientseitig erfolgt eine Speicherung einer von der Anwendung generierten, eindeutigen Identifikationsnummer sowie des Chatverlaufs, um ihn beim Start der Anwendung laden zu können. Es ist ebenfalls eine Speicherung der Server-Logs vorgesehen. Falsche Vorhersagen bezüglich der Intents oder Actions können auf diese Weise erkannt und der EnglishBot dadurch zukünftig verbessert werden. Die Logs beinhalten die Eingaben der Nutzer*innen, ausgeführte Actions, Intent-Vorhersagen, aktuelle Slots und erkannte Entitäten. Bei der Speicherung und Verarbeitung der Daten wird auf die Einhaltung der DSGVO geachtet. Diese sieht nach Art. 13 vor, dass vor dem Speichern von personenbezogenen Daten das Einverständnis der betroffenen Person eingeholt werden muss und diese die Möglichkeit hat, diesem jederzeit zu widersprechen. Personenbezogene

Daten müssen nach Art. 5 Abs. 1a DSGVO "[...] für festgelegte, eindeutige und legitime Zwecke erhoben werden [...]". Darüber hinaus haben Nutzer*innen nach Art. 15 DSGVO das Recht, jederzeit Auskunft über die sie betreffenden erhobenen Daten zu erhalten. Art. 17 DSGVO sieht den Rechtsanspruch auf Löschung dieser Daten vor.

Eine clientseitige Deinstallation der Anwendung hat für die jeweilige Person zur Folge, dass alle Daten verloren gehen, unter anderem der Chatverlauf und die Lernerfolge. Es gibt keine Möglichkeit, über die Deinstallation der Anwendung auf einem Client informiert zu werden. Daher ist es notwendig, einen Mechanismus zu implementieren, der ungenutzte Dateien auf dem Server regelmäßig löscht. Nutzer*innen, deren letzter Kontakt mehr als ein halbes Jahr zurückliegt, sollen automatisch gelöscht werden.

5.4 Architektur

Die Wahl der Technologie für die Entwicklung des Chatbots ist ausschlaggebend für die Architektur der Anwendung. In Kapitel 3 wurde der Rasa Stack mit Dialogflow verglichen. Während Dialogflow hinsichtlich der Bedienbarkeit punktet, sticht der Rasa Stack vor allem durch seine Open-Source Eigenschaft hervor, wodurch individuelle Anpassungen möglich sind. Der Mehraufwand durch die Konfiguration eines eigenen Servers ist durch die einhergehende volle Kontrolle über die Daten gerechtfertigt. Aus diesen Gründen fällt die Wahl auf den Rasa Stack.

Die Abbildung 5.4 veranschaulicht die grundlegende Architektur der Anwendung, welche einer klassischen Client-Server-Architektur entspricht. Der Client bietet die Benutzeroberfläche, über die Nutzer*innen mit dem Server kommunizieren können. Die Kommunikation zwischen Client und Server ist über Websockets vorgesehen. Im Vergleich zu HTTP-Anfragen erlauben Websockets eine bidirektionale Kommunikation in Echtzeit, wodurch sie sich speziell für Chat-Anwendungen eignen [vgl. FM11]. Auf dem Rasa Core Server findet das Dialogmanagement statt. Rasa NLU ist für die Extraktion der Intents und Entities verantwortlich. Die Rechtschreibung und Grammatik werden durch ein Tool zur Erkennung von Fehlern geprüft. Ein Action-Server stellt Funktionen bereit, die auf bestimmte Intents folgen und durch den Dialogmanager bestimmt werden. Überdies können externe APIs aufgerufen werden. Zur Speicherung der Nutzer*innendaten und Lerninhalte ist eine nicht-relationale Datenbank vorgesehen. Darüber hinaus werden eine individuelle Identifikationsnummer, der Chatverlauf sowie die Zustimmung gemäß der DSGVO, welche für die Nutzung der Applikation unabdingbar ist, lokal gespeichert.

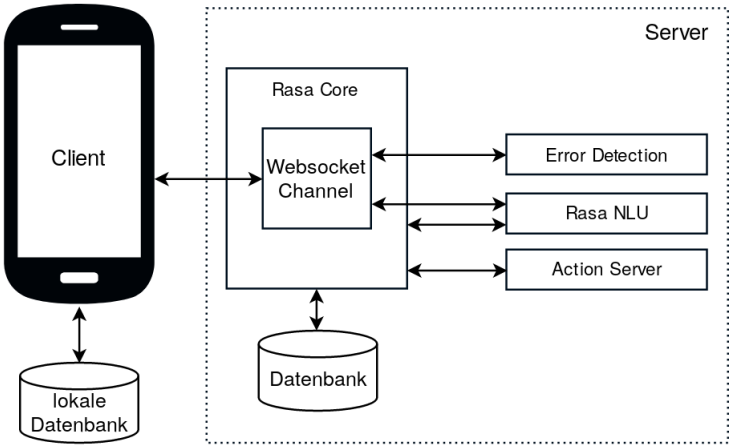


Abbildung 5.4: Überblick über die Softwarearchitektur (Quelle: eigene Darstellung)

Es sind die folgenden Services vorgesehen: Rasa Core, Rasa NLU, Error Detection, Rasa NLU, Action Server und eine Datenbank. Es wird eine möglichst modulare Architektur angestrebt, um die Komplexität der Gesamtanwendung zu minimieren. Jeder Service ist entkoppelt von den anderen und soll über HTTP-Anfragen mit JSON als Austauschformat angesprochen werden können. Dadurch ist es möglich, Services zu ändern, ohne die Funktionalität der gesamten Anwendung oder andere Services zu beeinflussen. Es können auch flexibel weitere Services hinzugefügt werden. Die einzelnen Services können auch als Container (beispielsweise Docker-Container[3]) bereitgestellt werden und so eine einfache Portabilität der Anwendung garantieren.

5.5 Benutzerschnittstelle

Die Benutzeroberfläche ist die Schnittstelle zwischen einer Anwendung und ihren Nutzer*innen und nimmt eine wichtige Rolle bei der Interaktion ein. Nach der DIN EN ISO 9241, welche Qualitätsrichtlinien zur Sicherstellung der Ergonomie interaktiver Systeme beschreibt, soll eine Applikation leicht zu erlernen und intuitiv zu benutzen sein, eine geringe Fehlerrate aufweisen und die Zufriedenheit sicherstellen. Die Miteinbeziehung der zukünftigen Nutzer*innen in den Entwicklungsprozess verhilft bei der Erfüllung dieser Anforderungen. Infolgedessen kann eine Anwendung mit einer hohen Usability und User Experience erstellt werden. Die User Experience umfasst die gesamte Erfahrung, die Nutzer*innen mit einem Produkt machen. Dazu zählen sowohl physische als auch psychische Reaktionen. [vgl. Low13, S. 13]

[3] https://www.docker.com

Bei der Entwicklung der Benutzerschnittstelle wurde das Feedback von zukünftigen Nutzer*innen eingeholt. Das Ergebnis ist in den Mockups in der Abbildung 5.5 zu sehen. Die Anwendung besteht neben dem Chatmodus in Abbildung 5.5 a) aus vier weiteren Ansichten. Die Navigation zwischen den einzelnen Ansichten ist durch eine Sidebar vorgesehen (siehe Abbildung 5.5 b)).

Abbildung 5.5: Mockups der einzelnen Screens (Quelle: eigene Darstellung)

Die Abbildung 5.5 c) veranschaulicht den individuellen Lernfortschritt. Es ist vorgesehen, die gelernten Wörter und geführten Dialoge sowie den wöchentlichen Fortschritt zu visualisieren. In einer weiteren Ansicht (siehe Abbildung 5.5 d)) können die Benachrichtigungshäufigkeit, die Fehlerkorrekturhäufigkeit und das tägliche Lernziel eingestellt werden. Überdies haben Nutzer*innen nach der DSGVO die Möglichkeit, die über sie gespeicherten Daten herunterzuladen oder zu löschen. Vor dem Löschen ist eine Information an die Nutzer*innen über dessen Auswirkungen, die unwiderrufliche Entfernung der dokumentierten Lernerfolge und sämtlicher Chatverläufe, beabsichtigt. Die Daten-

schutzerklärung und die rechtlichen Hinweise sind in einer weiteren Ansicht zu finden (siehe Abbildung 5.5 e)). Wörter können in einer eigenen Ansicht - dem Vokabelmodus - gezielt nachgeschlagen werden (siehe Abbildung 5.5 f) bis h)). Zunächst werden die verfügbaren Vokabelmodule aufgelistet, aus denen eine Kategorie ausgewählt werden kann. Anschließend werden die zugehörigen Vokabeln angezeigt. Durch das Tippen auf eine Vokabel erhalten Nutzer*innen deren Definition (siehe Abbildung 5.5 h)).

Im Rahmen der Besprechung des Konzepts wurde seitens der Nutzer*innen der Wunsch geäußert, Wörterbuchanfragen nicht nur über einen textuellen Befehl ("What does the word professor mean?") zu stellen, sondern hierfür eine effizientere Lösung zu finden. Ein Wischen auf dem Texteingabefeld soll einen schnellen Wechsel zwischen dem normalen Chatten und einer Wörterbuchanfrage ermöglichen. Somit kann lediglich das gesuchte Wort eingegeben werden. Die farbliche Änderung des Eingabefelds signalisiert den Nutzer*innen, in welchem Modus sie sich befinden. Die Abbildung 5.6 visualisiert die Wörterbuchsuche über die Wischfunktion.

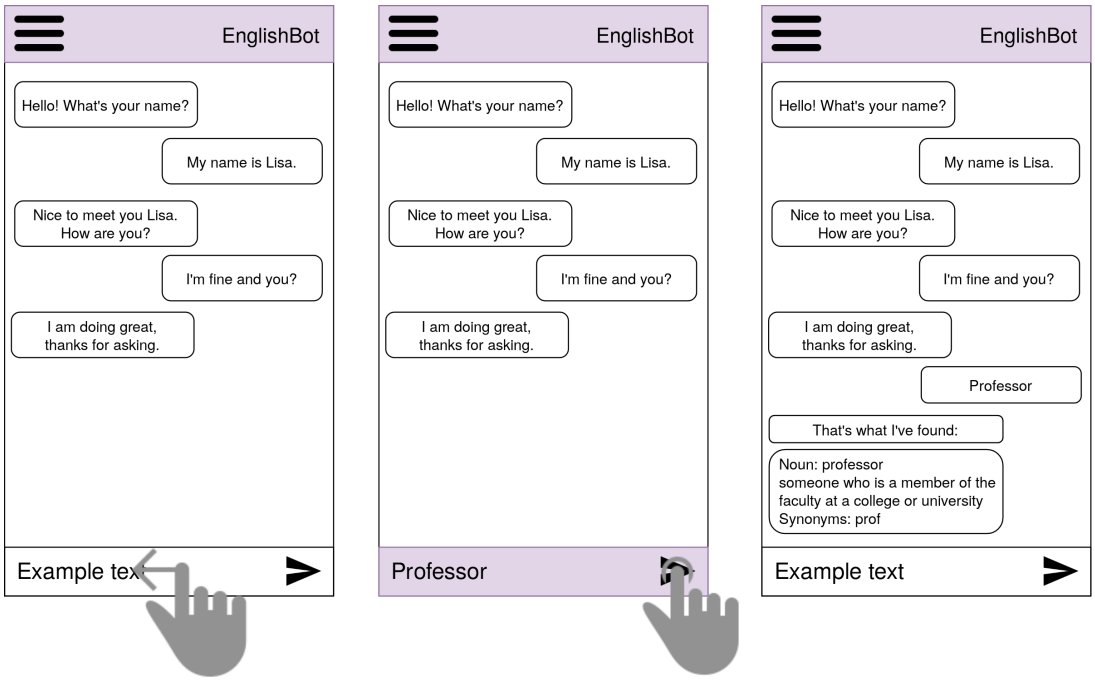


Abbildung 5.6: Wörterbuchsuche über die Wischfunktion (Quelle: eigene Darstellung)

5.6 Zusammenfassung

Aufbauend auf den Anforderungen aus Kapitel 4 wurde das Konzept definiert, die Datenhaltung spezifiziert sowie die Architektur und Benutzeroberfläche entworfen. Die folgende Tabelle 5.1 fasst die funktionalen Anforderungen zusammen und zeigt die Kapitel auf, in denen sie berücksichtigt wurden.

ID	Art	Kurztitel	Konzept
F01	*muss*	Vokabellernmodus	Kapitel 5.1 - entspricht dem Vokabelquiz
F02	*muss*	Fehlerkorrektur	Kapitel 5.1, Kapitel 5.4
F03	*muss*	Alternativformulierungen	Kapitel 5.1, Kapitel 5.4
F04	*muss*	Smalltalk	Kapitel 5.1
F05	*muss*	Einsprachiges Wörterbuch	Kapitel 5.1
F06	*soll*	Benachrichtigungen	Kapitel 5.3
F07	*soll*	Dokumentation des Lernfortschritts	Kapitel 5.2, Kapitel 5.3 und Kapitel 5.5
F08	*soll*	Anpassung an Lernfortschritt	Kapitel 5.2 und Kapitel 5.3
F09	*soll*	Lernziele	Kapitel 5.2 und Kapitel 5.3
F10	*kann*	Feedbackhäufigkeit	Kapitel 5.2 und Kapitel 5.3
F11	*kann*	Beispielsatz für Vokabel	Kapitel 5.1

Tabelle 5.1: Im Konzept berücksichtigte funktionale Anforderungen

In der folgenden Tabelle 5.2 werden die nicht-funktionalen Anforderungen aufgelistet. In der dritten Spalte wird analog zur Tabelle 5.1 auf die einschlägigen Kapitel aus dem Konzept referenziert.

ID	Art	Kurztitel	Konzept
NF01	*nicht-funktional*	Natürliche Sprache	Kapitel 5.1
NF02	*nicht-funktional*	Intuitive Benutzeroberfläche	Kapitel 5.5
NF03	*nicht-funktional*	Unerwartete Eingaben	Kapitel 5.1, Abbildung 5.2
NF04	*nicht-funktional*	Modulare Lerninhalte	Kapitel 5.3
NF05	*nicht-funktional*	Hilfestellung	Kapitel 5.1
NF06	*nicht-funktional*	Dialogspeicherung	Kapitel 5.3

Tabelle 5.2: Im Konzept berücksichtigte nicht-funktionale Anforderungen

Das Konzept legt den Grundstein für die Implementierung des EnglishBot.

6 Implementierung

In diesem Kapitel wird die prototypische Implementierung des EnglishBot vorgestellt. Zunächst erfolgt eine Beschreibung des Prototyps und dessen Funktionsumfang. Im Anschluss daran werden die verwendeten Werkzeuge beschrieben. Zuletzt wird auf die Implementierung einzelner Module eingegangen, wobei Ausschnitte des Codes vorgestellt werden. Der vollständige Quellcode befindet sich auf dem dieser Arbeit beiliegenden Datenträger und in einem öffentlich zugänglichen Git-Repository[1]. Daneben verschafft das Video `englishbot.mp4` auf dem USB-Stick einen ersten Eindruck über die App.

6.1 Prototyp des EnglishBot

Das Ergebnis dieser Arbeit ist ein Android-Prototyp, der mit einem Python-Backend kommuniziert. Das Python-Backend wiederum stellt Anfragen zur Überprüfung der Rechtschreibung und Grammatik an einen Java-Server (LanguageTool-Server). Die Architektur entspricht grundsätzlich der im Kapitel 5.4 konkretisierten Softwarearchitektur mit dem Unterschied, dass die Anwendung bisher nur lokal läuft, die Kommunikation unverschlüsselt erfolgt und keine Daten serverseitig gespeichert werden. Die Abbildung 6.1 veranschaulicht die Architektur des Prototyps. Serverseitig sind drei Python-Server (Rasa Core, Rasa NLU und Action Server) sowie ein Java-Server (LanguageTool Server) aktiv.

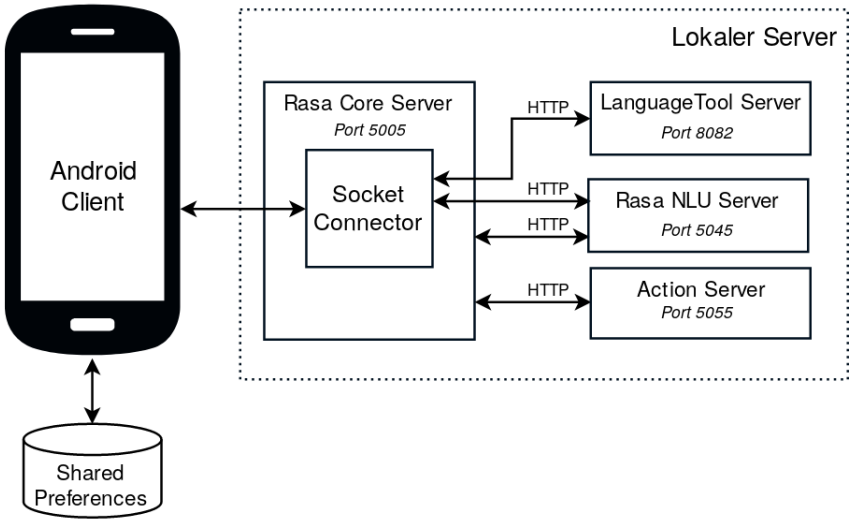


Abbildung 6.1: Softwarearchitektur des Prototyps (Quelle: eigene Darstellung)

[1] https://github.com/br00ks/englishbot

Beim Start der Android-App wird über Websockets eine Verbindung zum Server auf Port 5005 hergestellt, die bis zum Beenden der Anwendung aufrechterhalten wird. Die Kommunikation verläuft in Form von Events. Die Tabelle 6.1 listet alle Events auf, die in der Kommunikation zwischen Client und Server auftreten können.

Events	**Beschreibung**
connect	Herstellen der Verbindung zwischen Client und Server.
session_request	Übertragen der Session-ID, welche clientseitig erzeugt wird. Damit hat der Client die Kontrolle über die Verbindungsdauer.
user_uttered	Nachricht von Nutzer*in, welche aus dem eingegebenen Text, einer eindeutigen ID und der Session-ID besteht.
bot_uttered	Vom Chatbot erstellte Nachricht, die den Text enthält.
bot_error_message	Nachricht, welche die Liste von möglichen Rechtschreib- und Grammatikfehlern und Verbesserungsvorschlägen umfasst.
disconnect	Trennung der Verbindung zwischen Client und Server.

Tabelle 6.1: Liste der möglichen Events zwischen Client und Server

Die Abbildung 6.2 stellt den Ablauf bei Eingang einer Nachricht dar.

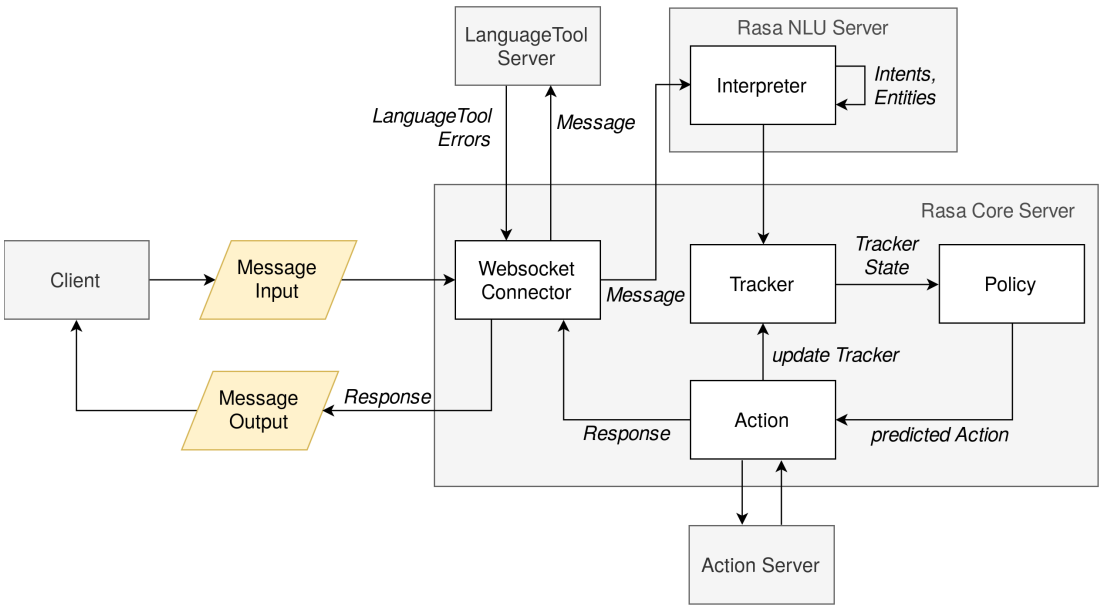


Abbildung 6.2: Visualisierter Ablauf bei Eingang einer Nachricht (Quelle: eigene Darstellung)

Zunächst erfolgt eine Überprüfung auf Rechtschreibung und Grammatik. Dafür ist ein lokaler Java-Server von LanguageTool im Einsatz, auf welchen im späteren Verlauf dieses Kapitels näher eingegangen wird. Gefundene Fehler werden an den Client geschickt. Als

Nächstes werden Intents und Entitäten von der Rasa NLU erkannt bzw. extrahiert und an den Rasa Core Server weitergeleitet, der zuständig für das Dialogmanagement ist. Die Antwort gelangt schließlich über den Websocket Connector an die Nutzer*innen.

Der Prototyp deckt nicht den vollständigen Funktionsumfang des Konzepts (siehe Tabelle 5.1 und Tabelle 5.2) ab. Der Fokus lag auf der Programmierung des Chatbots. Diese umfasste die Umsetzung der in der Anforderungsanalyse identifizierten *muss*-Anforderungen unter Berücksichtigung der nicht-funktionalen Anforderungen. Zudem stand bei der clientseitigen Programmierung der gesamtheitliche Eindruck im Mittelpunkt. So wurde neben der Chatbot-Ansicht auch die Navigationsleiste (Sidebar) implementiert, um den Nutzer*innen, die den Entwicklungsprozess begleiteten, einen möglichst vollständigen Eindruck über die Anwendung zu vermitteln. Bezüglich des Umfangs ist anzumerken, dass der Prototyp keine Datenspeicherung beinhaltet, jedoch bereits die Zustimmung zur DSGVO nach dem ersten Start der Anwendung umgesetzt wurde.

6.2 Android-Client

Bei der clientseitigen Android-Programmierung wurde auf die Einhaltung der Design Guidelines von Android[2] geachtet. Zur Gestaltung von nutzerfreundlichen Anwendungen wird unter anderem empfohlen, den Gestaltungsprinzipien von Material Design[3], einer von Google entwickelten Designsprache, zu folgen. Bei der Entwicklung von Android-Apps werden standardmäßig Komponenten bereitgestellt, die dem Material Design entsprechen.

Die erfolgte Implementierung ähnelt einem einfachen Chat. Die Abbildung 6.3 zeigt die Sidebar und Chatbot-Ansicht des EnglishBot. Die Sidebar besitzt in der aktuellen Version des Prototyps keinerlei Funktionalität, sondern dient lediglich dazu, einen vollständigen Gesamteindruck der App zu verschaffen. Der rechte Screenshot bildet die Information und Zustimmung zur DSGVO ab, die beim ersten Start der Anwendung angezeigt wird. Per Klick auf "Accept" wird die Zustimmung in den *Shared Preferences* des Geräts gespeichert, woraufhin eine Verbindung zum Server aufgebaut und der Einführungsdialog gestartet wird.

[2] https://developer.android.com/design
[3] https://material.io/design

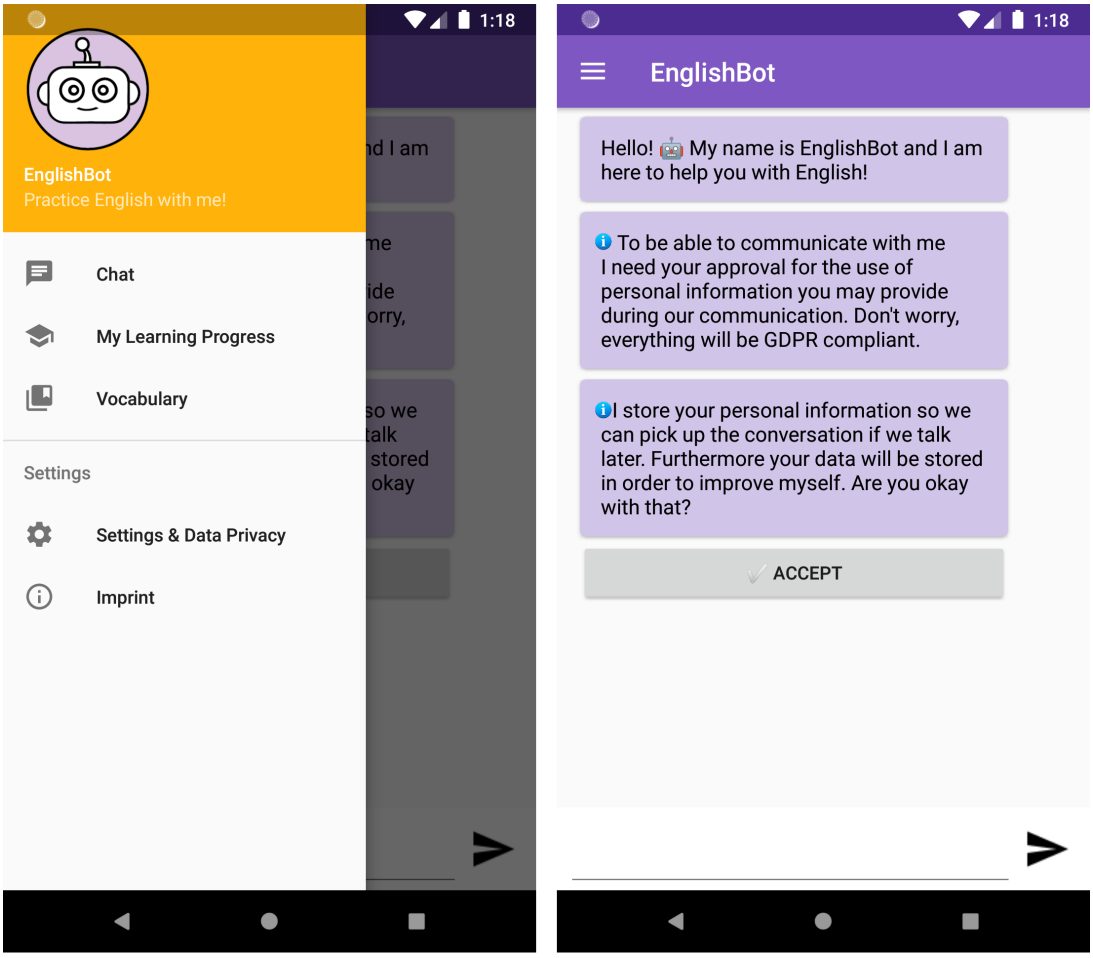


Abbildung 6.3: Sidebar (links) und Ansicht zur Zustimmung der DSGVO (rechts) (Quelle: eigene Darstellung)

Der Einführungsdialog (siehe Abbildung 6.4) gilt dem Kennenlernen der Nutzer*innen und dem Vertrautmachen mit der Applikation. Die Anwendenden werden nach ihrem Namen und Wohnort gefragt. Der Dialog endet mit der Vorstellung des Funktionsumfangs der Anwendung.

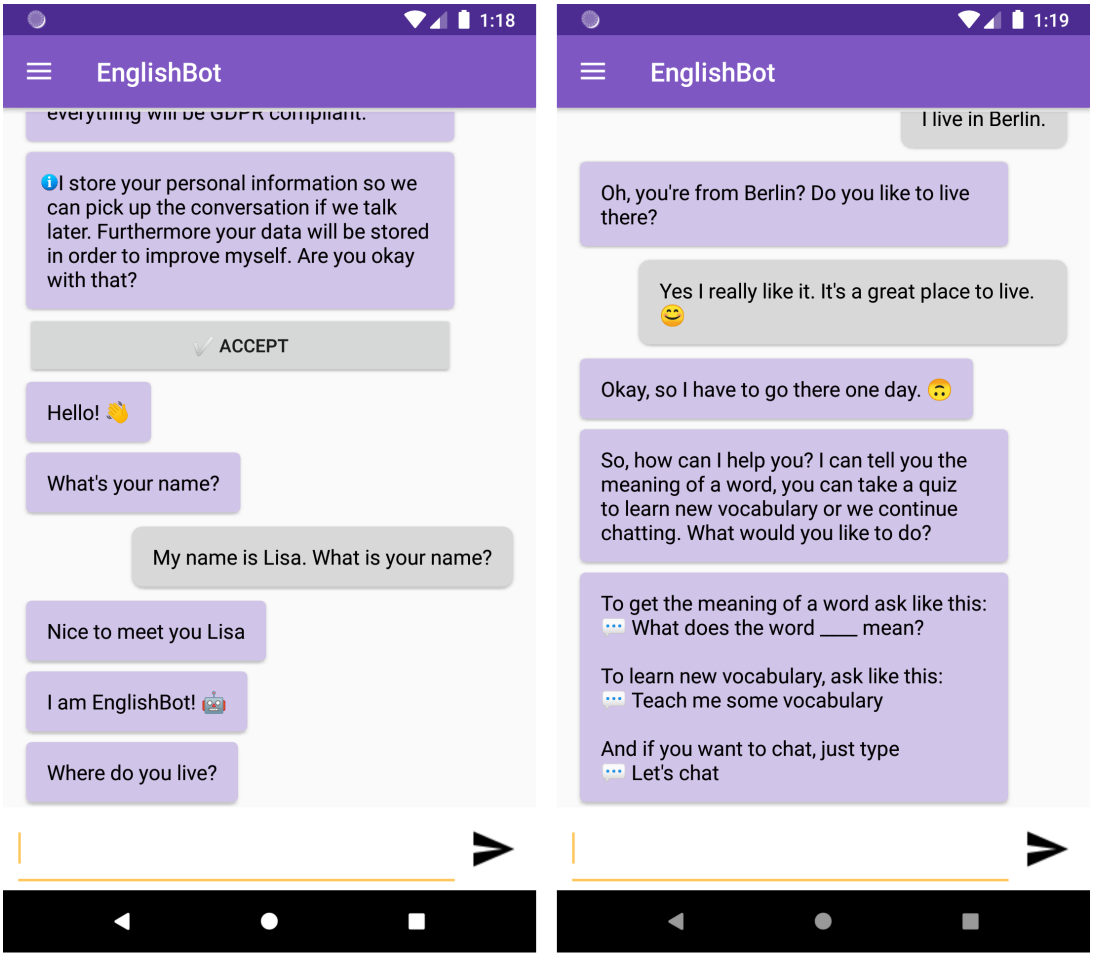


Abbildung 6.4: Einführungsdialog (Quelle: eigene Darstellung)

Das Diagramm in Abbildung 6.5 veranschaulicht die Klassen der Android-Anwendung. In der Klasse `MainActivity` wird die Chatbot-Ansicht definiert und die Socket-Verbindung aufgebaut. Ebenso findet die Verwaltung der ein- und ausgehenden Nachrichten statt. Die Klasse `MessageAdapter` führt eine Liste der Nachrichten und aktualisiert beim Eintreffen jeder neuen Nachricht, die einem Objekt der Klasse `Message` entspricht, die `View`. Die visuelle Darstellung einer Nachricht ist durch ihren Typ bedingt, welcher wiederum vom Socket-Event bestimmt wird. Neben der Nachricht selbst und der eindeutigen ID besitzt das Objekt ein Attribut zur Bestimmung der Nachrichtenart. Die Klasse `MessageType` deklariert die verschiedenen Nachrichtenarten. In der Klasse `Constants` werden lediglich konstante Werte - wie beispielsweise die IP-Adresse des Websocket Connectors - definiert.

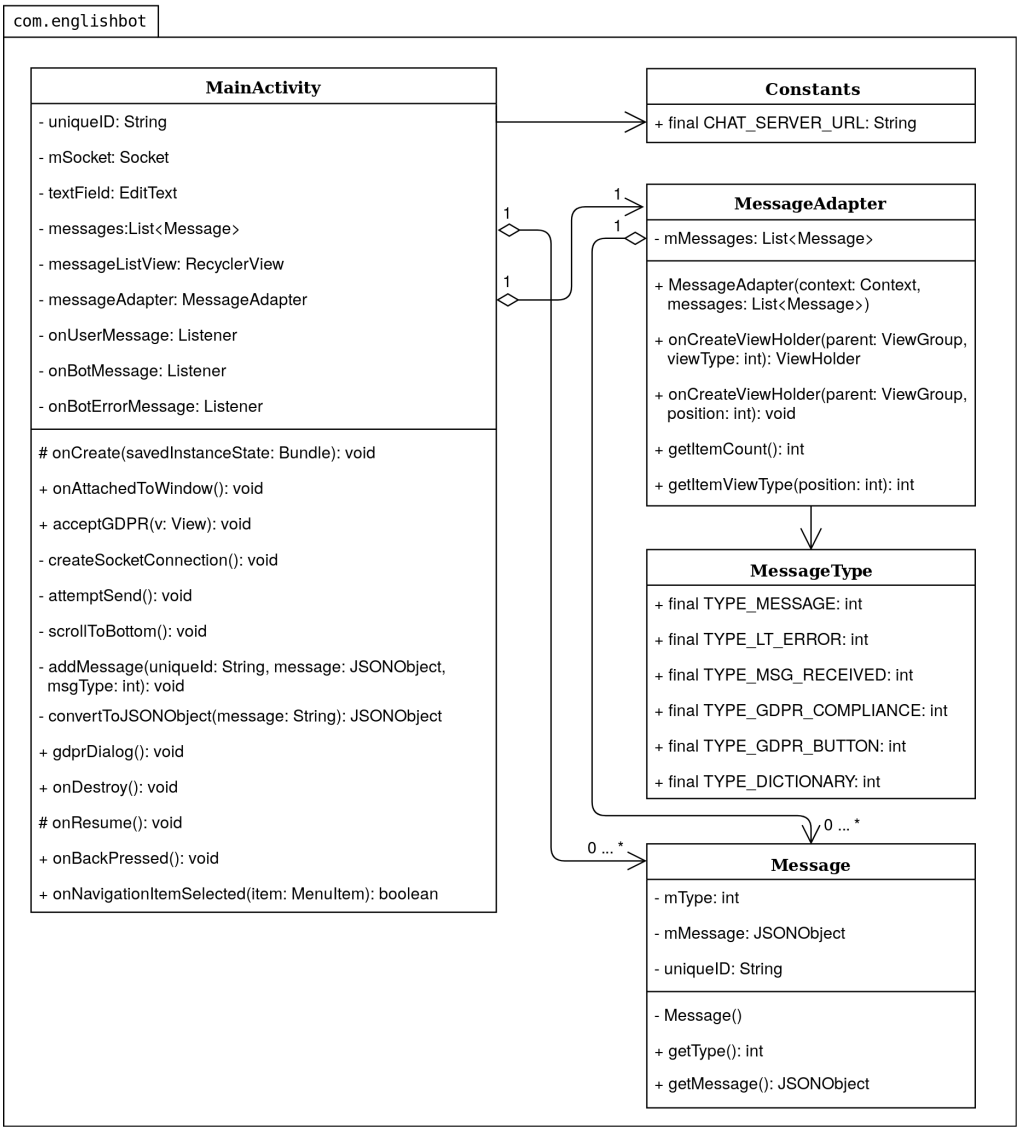


Abbildung 6.5: Klassendiagramm des Android-Client (Quelle: eigene Darstellung)

Im nächsten Schritt wird eine von den Nutzer*innen verschickte Nachricht vom Socket Connector entgegengenommen und vom LanguageTool Server auf Rechtschreibung und Grammatikfehler geprüft.

6.3 Grammatik- und Rechtschreibkorrektur

LanguageTool[4] ist ein Open-Source Tool zur Prüfung von Grammatik, Stil und Rechtschreibung und unterstützt 31 Sprachen. Entwickelt wurde es von Daniel Naber und

[4]https://languagetool.org

Marcin Milowski im Jahr 2005 [Nab03]. Wörter, Sätze und Texte werden dahingehend überprüft, ob sie bestimmten Regeln, welche in XML-Dateien definiert sind, entsprechen. Die Regeln beschreiben typische Fehler in verschiedenen Kategorien wie zum Beispiel Rechtschreibung oder Grammatik. Zunächst wird der eingegebene Text in Sätze aufgeteilt, daraufhin wird jeder Satz in seine einzelnen Wörter zerlegt und sodann wird jedem Wort ein POS Tag (siehe Kapitel 2.4.2) zugewiesen. Die Wörter und die Tags werden dafür genutzt, um nach Mustern zu suchen, die den definierten Regeln entsprechen. Bei einer Übereinstimmung ist davon auszugehen, dass der Satz fehlerhaft ist. [vgl. Lan19b]

LanguageTool eignet sich angesichts seiner individuellen Erweiterbarkeit - es können beliebige Regeln hinzugefügt werden - und seiner Quelloffenheit als Instrument der Grammatik- und Rechtschreibkorrektur [vgl. Lan19a]. Das Tool kann entweder über die öffentlich bereitgestellte API genutzt werden oder es wird ein eigener lokaler Server betrieben. Die erste Variante birgt den Nachteil, dass ein tägliches Limit die Anzahl an Serveranfragen beschränkt [vgl. Lan19e]. Für den Betrieb auf eigener Hardware wird ein Java-Server zum Download bereitgestellt [vgl. Lan19d], wobei zu berücksichtigen ist, dass für eine optimale Nutzung die Einbindung eines großen N-Gramm-Datensatzes notwendig ist [vgl. Lan19c].

Der EnglishBot macht von einem lokalen Server Gebrauch. Eingehende Nachrichten des Android-Client werden an den auf Port 8082 laufenden LanguageTool Server weitergeleitet. Das Ergebnis ist ein Objekt der Klasse `LanguageToolResponse`, welches den Eingabetext sowie eine JSON-Datei mit den gefundenen Fehlern und zugehörigen Verbesserungsvorschlägen enthält. Dabei wird an erster Stelle jener Verbesserungsvorschlag an die Nutzer*innen geschickt, welcher mit der größten Wahrscheinlichkeit zu einer Korrektur der fehlerhaften Eingabe führt [vgl. Lan19c].

Der Codeauszug 6.1 bildet eine beispielhafte JSON-Datei als Antwort auf die falsche Eingabe "No, I don't like it their" ab, die als Ergebnis an den Android-Client zurückgeschickt wird.

```
1 Errors:[ { 'start_pos': 20,
2        'end_pos': 25,
3        'error': 'their',
4        'suggestion': 'there',
5        'category': 'TYPOS',
6        'message': "Statistics suggests that 'there' (as in 'Is there an ↩
             answer?') might be the correct word here, not 'their' (as in ↩
             'It's not their fault.'). Please check. Did you mean 'there'?"
7        } ]
```

Codeauszug 6.1: LanguageTool Fehlerbeispiel

Anschließend beginnt die Verarbeitung der Eingabe durch den Rasa Stack.

6.4 Natural Language Understanding

Rasa NLU - als Teil des Rasa Stacks - ist für das Natural Language Understanding zuständig und hat die Klassifizierung von Intents und das Extrahieren von Entitäten zur Aufgabe. Eine Liste der gesamten Intents und Entitäten des Prototyps sind im Anhang D in den Tabellen D.1 und D.2 zu finden. Eingehende Nachrichten werden von unterschiedlichen Komponenten verarbeitet, die individuell zu einer NLU-Pipeline kombiniert werden. Jede Komponente verarbeitet die eingehende Nachricht und produziert einen Output für die nächste.

Der EnglishBot macht von der von Rasa vordefinierten Pipeline `supervised_embeddings` Gebrauch, weil diese es ermöglicht, in einer Eingabe auch mehrere Intents zu erkennen [vgl. Ras19c]. Der Codeauszug 6.2 bildet einen Ausschnitt der Datei `config.yml` ab und listet die verwendeten Komponenten der Pipeline auf, welche im weiteren Verlauf näher beschrieben werden.

```
1  # Configuration for Rasa NLU.
2  language: "en"
3
4  pipeline:
5    - name: "WhitespaceTokenizer"
6      case_sensitive: false
7    - name: "RegexFeaturizer"
8    - name: "CRFEntityExtractor"
9    - name: "EntitySynonymMapper"
10   - name: "CountVectorsFeaturizer"
11   - name: "EmbeddingIntentClassifier"
```

Codeauszug 6.2: Rasa NLU Pipeline

Eingangs werden die einzelnen Wörter durch den `WhitespaceTokenizer` getrennt, wobei für jede durch ein Leerzeichen getrennte Zeichenfolge ein Token erstellt wird. Nachfolgend legt die Komponente `RegexFeaturizer` eine Liste an regulären Ausdrücken an, die in den NLU-Trainingsdaten definiert wurden und als Hilfsmittel beim Erkennen von Intents und Entitäten dienen. Die Komponente `CRFEntityExtractor` bedient sich eines Conditional Random Field (CRF), welches ein probabilistisches Modell zur Segmentierung von Daten und Extraktion von Entitäten ist [vgl. LMP01]. Die wahrscheinlichste Menge an Entitäten gelangt an die nächste Komponente `EntitySynonymMapper`, welche diese daraufhin auf die in den Trainingsdaten definierten Synonyme abbildet. Der

`CountVectorsFeaturizer` ist für die Erstellung eines *Bag-of-Words* für die Eingabe und der Intents zuständig. Ein *Bag-of-Words* ist ein Algorithmus, der die Häufigkeit eines Wortes in einer Eingabe berechnet [vgl. JM09, S. 65] und dabei von der Funktion `CountVectorizer` der Python Machine Learning Bibliothek *scikit-learn*[5] Gebrauch gemacht, die im Rasa Stack integriert ist. Die Bag-of-Words-Darstellung ist für die letzte Komponente `EmbeddingIntentClassifier` von Bedeutung, da diese das Training von *Word Embeddings* vornimmt. *Embeddings* bezeichnen im Allgemeinen Vektoren zur Darstellung von Wörtern, da diese in einem bestimmten Vektorraum eingebettet sind [vgl. JM09, S. 107]. Das Ergebnis des NLU ist eine strukturierte Eingabe, wie im folgenden beispielhaften Codeauszug 6.3 zu sehen ist.

```
{
  'intent':{
    'name':'user.name+bot.name',
    'confidence':0.970967114
    },
  'entities':[
      {
        'start':11,
        'end':15,
        'value':'lisa',
        'entity':'name',
        'confidence':0.99928373,
        'extractor':'CRFEntityExtractor'
      }
    ],
  'intent_ranking':[
    {
      'name':'user.name+bot.name',
      'confidence':0.970967114
    },
    {
      'name':'ask_whatspossible',
      'confidence':0.2276818156
    },
  ],
  'text':'My name is Lisa and yours?'
}
```

Codeauszug 6.3: Beispielergebnis des NLU

Die Eingabe "My name is Lisa and yours?" aus dem Beispiel entspricht mit einer Wahrscheinlichkeit von rund 97 % dem Intent "user.name+bot.name". Es ist also davon

[5] https://scikit-learn.org

auszugehen, dass die Eingabe den Namen der Nutzerin beinhaltet und nach dem Namen des Bot gefragt wurde. Ebenso wird korrekterweise die Entität "name" mit dem Wert "Lisa" erkannt. Dieses Ergebnis wird nun im nächsten Schritt an den Dialogmanager weitergeleitet.

6.5 Dialogmanagement

Rasa Core als Dialogmanager kontrolliert den Verlauf der Konversation. Mithilfe eines trainierten Modells wird basierend auf den vergangenen Gesprächen eine Vorhersage getroffen, welche Action als nächstes ausgeführt werden soll. Als Trainingsdaten dienen beispielhafte Konversationen zwischen Nutzer*in und Chatbot (*Stories*). Der EnglishBot arbeitet hinsichtlich des Antwortverhaltens abfragebasiert (siehe Kapitel 2.3.2). Die möglichen Antworten sind in der `domain.yml` definiert und werden abhängig von der Vorhersage des Dialogmanagers zurückgeliefert.

Der Codeauszug 6.4 stellt die Policies der Anwendung dar, welche Maßnahmen festlegen, die bei jedem Schritt im Dialog durchzuführen sind. Rasa bietet verschiedene vordefinierte Policies an, aus denen ausgewählt werden kann. Jede Policy trifft eine Vorhersage für die als nächstes auszuführende Action, wobei schließlich die Policy mit dem höchsten *Confidence Score* die nächste Action festlegt. Der Confidence Score gibt die Sicherheit des Modells bezüglich der Zuordnung an [vgl. Raj19, S. 28]. Liegt dieser Confidence Score unter einem bestimmten Schwellenwert (im Fall des Prototyps bei weniger als 0.5), greift die `FallbackPolicy` [vgl. Ras19h]. Eine Action kann das Senden einer einfachen Nachricht an den Client beinhalten, das Ausführen von Code, der eine beliebige Anzahl an Nachrichten zurückschickt oder es wird eine standardmäßige Action, bei denen Rasa zwischen acht unterscheidet, ausgeführt [vgl. Ras19a]. Dazu zählt beispielsweise das Warten auf eine neue Eingabe (`action_listen`).

```
1  # Configuration for Rasa Core.
2
3  policies:
4    - name: MemoizationPolicy
5    - name: KerasPolicy
6    - name: FormPolicy
7    - name: "FallbackPolicy"
8      nlu_threshold: 0.5
9      core_threshold: 0.5
10     fallback_action_name: "action_default_fallback"
```

Codeauszug 6.4: Rasa Core Pipeline

Die `MemoizationPolicy` gleicht die Eingabe mit den Konversationen der Trainingsdaten ab und prognostiziert die nächste Action mit einem Confidence Score von 1.0, wenn die aktuelle Konversation sich mit einer der Trainingsdaten deckt.

Die `KerasPolicy` nutzt ein in Keras[6] - einer Python Deep Learning Bibliothek - implementiertes neuronales Netz, um die nächste Action auszuwählen. Die Architektur baut auf einem Long Short-Term Memory (LSTM)-Netzwerk - einer besonderen Art eines rekurrenten neuronalen Netzes - auf. Rekurrente neuronale Netze eignen sich für die Verarbeitung von Sprache, weil sie die Darstellung beliebig großer Eingaben in Vektoren fester Größe ermöglichen [vgl. GH17, S. 163]. Basierend auf Intents, Entitäten, Slots und vorhergegangen Actions wird eine Vorhersage für die nächste Action getroffen.

Die `FormPolicy` ist eine Erweiterung der `MemoizationPolicy` und übernimmt das Ausfüllen von Forms. Sobald eine `FormAction` aufgerufen wird, sagt die `FormPolicy` diese `FormAction` so lange voraus, bis alle erforderlichen Werte eingegangen sind. Forms bieten eine Möglichkeit, mehrere Informationen nacheinander zu sammeln. Im EnglishBot kommen sie an drei Stellen zum Einsatz: im Einführungsdialog, im Vokabelquiz und bei der Wörterbuchanfrage (siehe Tabelle D.5 im Anhang).

Die `FallbackPolicy` ruft in folgenden Fällen eine Fallback-Action auf:

- Der Confidence Score liegt bei der Erkennung des Intents unter einem bestimmten Schwellenwert (`nlu_threshold`).
- Der Intent mit dem höchsten und zweithöchsten Wert sind sich zu ähnlich und der Differenzwert liegt unter einem bestimmten Schwellenwert.
- Keine Vorhersage der Policies liegt über dem `core_threshold`.

Die Fallback-Action des EnglishBot führt letztlich dazu, dass der Bot eine Nachricht zurückschickt und um eine Umschreibung des Satzes bittet. Die möglichen Antworten wurden in der Datei `domain.yml` festgelegt. Eine Nachricht könnte beispielsweise lauten: "I am afraid I have trouble understanding. Please try to rephrase.".

6.6 Funktionen des EnglishBot

Nach der Erläuterung der grundsätzlichen Verarbeitung einer Nachricht werden im Folgenden die Funktionen des EnglishBot aufgeführt. Der Prototyp weist im Chatmodus im Wesentlichen drei Modi auf. Zum einen kann Smalltalk über eine begrenzte Menge an Themen geführt werden. Zum anderen kann die Bedeutung eines Wortes erfragt

[6] https://keras.io

werden und es können Vokabeln zum Thema "University: get ready for an exchange year" in einem Quiz geübt werden.

Smalltalk
Zum Umfang des Smalltalks im Rahmen der prototypischen Umsetzung zählen beispielsweise Fragen zum Namen, zu den Hobbies, zu den beherrschten Sprachen des EnglishBot und Gespräche bezüglich des Studiums/Berufs. Die vollständige Liste der Intents ist im Anhang D zu finden.

Wörterbuchsuche
Die Bedeutung eines Wortes kann mit einer natürlichsprachlichen Formulierung, wie zum Beispiel "What does the word teacher mean?", erfragt werden. Wird der Intent `word.meaning` erkannt, wird folglich die mithilfe von Natural Language Toolkit (NLTK) oder Wiktionary eingeholte Definition des extrahierten Wortes an den Client als JSON-Datei zurückgeschickt. Falls keine Bedeutung gefunden wird, sendet der Chatbot eine entsprechende Nachricht an den Client (siehe Abbildung 6.6).

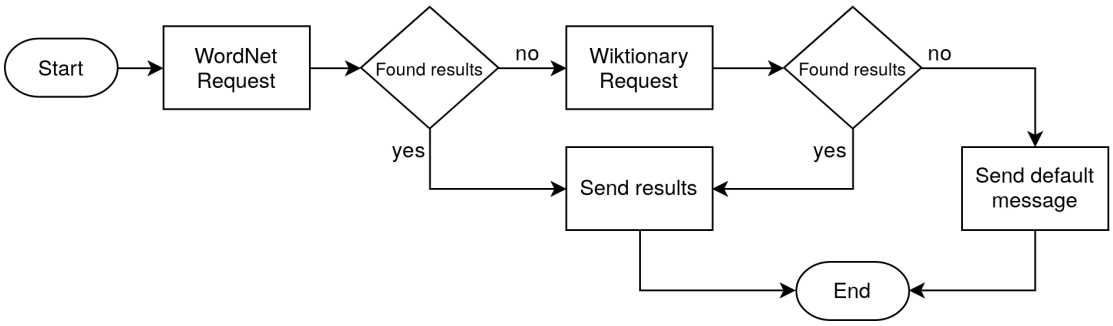


Abbildung 6.6: Wörterbuchanfrage (Quelle: eigene Darstellung)

NLTK, eine Python-Sammlung an Bibliotheken für NLP, besitzt eine Schnittstelle zu WordNet[7], einer lexikalischen Datenbank für die englische Sprache, die von der Princeton University erstellt wurde und kostenlos zur Verfügung steht [vgl. Mil95]. Wörter werden in Gruppen von Synonymen (sogenannten *Synsets*) zusammengefasst, die nach der statistischen Häufigkeit geordnet sind [vgl. JM09, S. 497].

Der beispielhafte Quellcode (siehe Codeauszug 6.5) veranschaulicht die Abfrage des Wortes "schedule", anhand dessen das Konzept von Synsets verdeutlicht werden soll.

```
1 from nltk.corpus import wordnet
2
3 synsets_word = wordnet.synsets("schedule")
```

Codeauszug 6.5: Synsets für das Wort "schedule"

[7] https://wordnet.princeton.edu

Die Variable `synsets_word` aus dem Codeauszug 6.5 umfasst die vier im Folgenden aufgelisteten Synsets. Die Namen der *Synsets* entsprechen dem gleichen Schema: [Name].[POS Tag].[Nummer]

- `Synset('agenda.n.01')`
- `Synset('schedule.n.02')`
- `Synset('schedule.v.01')`
- `Synset('schedule.v.02')`

Jedes Synset beinhaltet unter anderem den POS Tag, die Definition und gegebenenfalls auch Sätze, in denen das Wort vorkommt. Die Definition kann zum Beispiel mit der Funktion `definition()` ermittelt werden.

Falls das Wort nicht von WordNet erfasst ist, also die Liste an Synsets leer ist, wird eine HTTP-Anfrage an Wiktionary gestellt, einem frei verfügbaren Wörterbuch. Dieses stellt eine öffentliche API bereit, über welche Bedeutungen von Wörtern erfragt werden können. Das Ergebnis wird als JSON-Datei zurückgeliefert.

Vokabelquiz
Das Vokabelquiz wird mit Erkennung des Intents `learning.teach_vocabulary` gestartet, wobei zunächst nach der gewünschten Kategorie gefragt wird. Da der Prototyp bislang nur das Thema "studying" beherrscht, entfällt demnach die Frage und das Quiz startet mit einer einfachen Bestätigung.

Die Vokabelmodule sind im Prototyp in einer JSON-Datei organisiert. In der Datei `university.json` befinden sich insgesamt 89 Wörter, die in der prototypischen Implementierung in einer zufälligen Reihenfolge abgefragt werden. Die Vokabeln entsprechen dem im Konzept festgelegten Format einer Vokabel-Karteikarte (siehe Codeauszug 5.2). Hierfür ist im Codeauszug 6.6 ein Beispiel angeführt.

```
1 {
2   "POS": "Noun",
3   "words": ["syllabus", "study plan"],
4   "questions": [],
5   "definition": "a plan showing the subjects or books to be studied in a ↩
        particular course",
6   "example_sentences": ["The new syllabus allows students greater ↩
        freedom of choice."]
7 }
```

Codeauszug 6.6: Ausschnitt aus der university.json

Das Vokabelquiz ist mithilfe einer Form umgesetzt, einem Konzept von Rasa, welches die Vorhersage der nächsten Action durch den Dialogmanager beeinflusst. Aktivierte Forms werden erst dann deaktiviert, wenn definierte Informationen vorliegen.

Die Abbildung 6.7 visualisiert die einzelnen Ablaufschritte des Vokabelquiz. Aufgrund der Implementierung mit einer Form beschränkt sich die Reaktion auf folgende Intents: `word.meaning`, `learning.hint` und `learning.exit`. Infolgedessen kann nach der Bedeutung eines Wortes oder nach einem Tipp gefragt werden. Beim Vorliegen des Intents `learning.exit` wird der Slot `vocabulary_quiz_exit` gesetzt, die Form deaktiviert und folglich das Quiz verlassen.

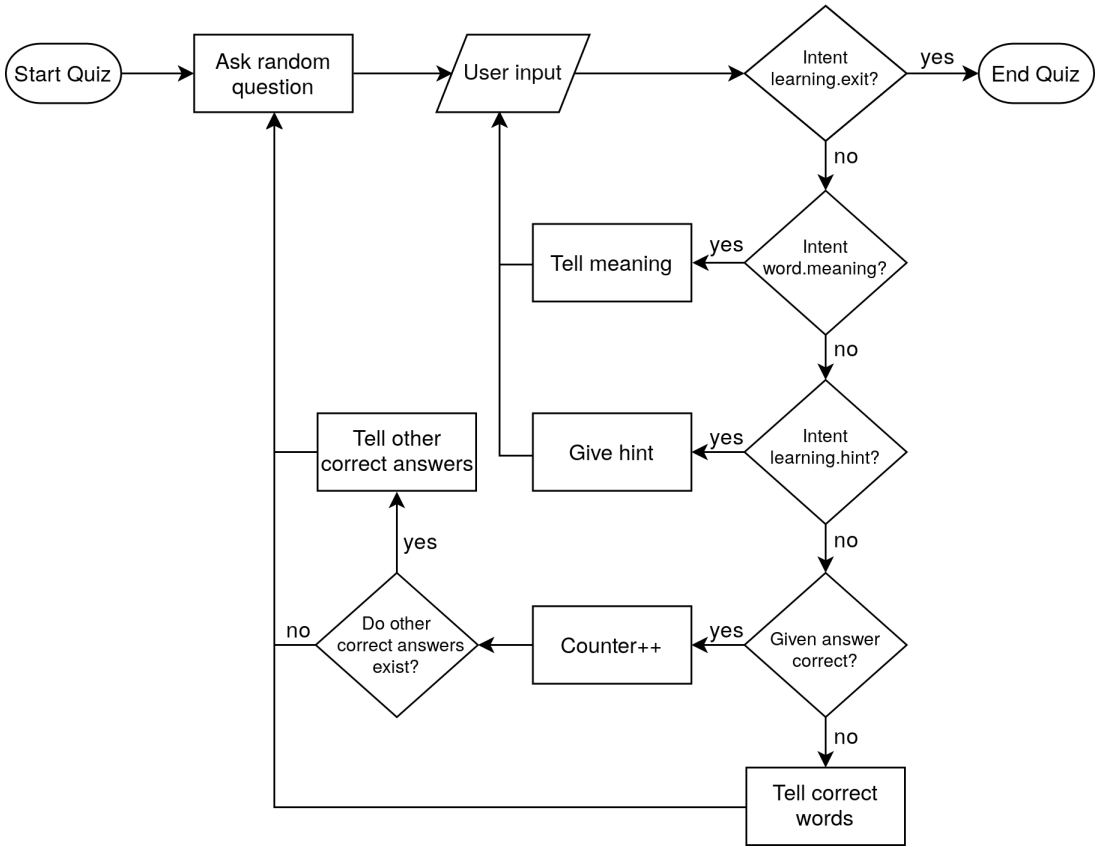


Abbildung 6.7: Vokabelquiz Ablaufdiagramm (Quelle: eigene Darstellung)

Nach Verlassen des Quiz erhalten die Nutzer*innen eine Meldung über ihren Erfolg. Ferner besteht die Möglichkeit, in den Smalltalk-Modus zu wechseln, in welchem eine Konversation über ein beliebiges Thema gestartet wird.

6.7 Zusammenfassung

Die prototypische Umsetzung des EnglishBot umfasst alle wesentlichen Komponenten für eine Kommunikation zwischen Chatbot und Nutzer*in. Die Android-Anwendung steht über Websockets mit den serverseitigen Diensten in Verbindung. Während LanguageTool für die Rechtschreib- und Grammatikprüfung zuständig ist, sorgt Rasa NLU für das Verständnis der natürlichsprachlichen Eingaben und Rasa Core ist für den Verlauf der Konversation verantwortlich.

Im Mittelpunkt der Implementierung standen die definierten funktionalen *muss*- und die nicht-funktionalen Anforderungen. Im schriftlichen Dialog mit dem Bot werden Fehler in den Eingaben der Nutzer*innen korrigiert und alternative Schreibweisen vorgeschlagen. Überdies ist der EnglishBot in der Lage, Smalltalk über bestimmte Themen zu führen und die Bedeutung von Wörtern zu erläutern. Das Vokabelquiz ist modular aufgebaut, sodass neue Vokabelmodule im entsprechenden JSON-Format jederzeit hinzugefügt werden können. Darüber hinaus kommt unter bestimmten Bedingungen (siehe im Einzelnen Kapitel 6.5) ein Fallback-Mechanismus zum Einsatz, der dafür sorgt, dass die Anwendenden zum Paraphrasieren aufgerufen werden.

Die Tabellen 6.2 und 6.3 fassen abschließend zusammen, welche funktionalen und nichtfunktionalen Anforderungen umgesetzt und in welchem Kapitel sie erörtert wurden.

ID	Art	Kurztitel	Implementierung
F01	*muss*	Vokabellernmodus	Kapitel 6.3
F02	*muss*	Fehlerkorrektur	Kapitel 6.3
F03	*muss*	Alternativformulierungen	Kapitel 6.3
F04	*muss*	Smalltalk	Kapitel 6.3
F05	*muss*	Einsprachiges Wörterbuch	Kapitel 6.3
F06	*soll*	Benachrichtigungen	-
F07	*soll*	Dokumentation des Lernfortschritts	-
F08	*soll*	Anpassung an Lernfortschritt	-
F09	*soll*	Lernziele	-
F10	*kann*	Feedbackhäufigkeit	-
F11	*kann*	Beispielsatz für Vokabel	-

Tabelle 6.2: In der Implementierung berücksichtigte funktionale Anforderungen

ID	Art	Kurztitel	Implementierung
NF01	*nicht-funktional*	Natürliche Sprache	Kapitel 6.1
NF02	*nicht-funktional*	Intuitive Benutzeroberfläche	umgesetzt durch UCD
NF03	*nicht-funktional*	Unerwartete Eingaben	Kapitel 6.5
NF04	*nicht-funktional*	Modulare Lerninhalte	Kapitel 6.6
NF05	*nicht-funktional*	Hilfestellung	umgesetzt durch Help-Intent
NF06	*nicht-funktional*	Dialogspeicherung	-

Tabelle 6.3: In der Implementierung berücksichtigte nicht-funktionale Anforderungen

7 Evaluierung

Das folgende Kapitel behandelt die Evaluierung des EnglishBot Prototyps. Das Ziel ist die Benutzerfreundlichkeit der Anwendung zu überprüfen und beurteilen zu können. Nach Erläuterung der Vorgehensweise werden die Ergebnisse des durchgeführten Usability-Tests präsentiert. Am Ende werden die gewonnenen Erkenntnisse und aufgedeckten Usability-Probleme diskutiert.

7.1 Vorgehensweise

Zur Evaluierung des EnglishBot wurden Usability-Tests mit anschließendem Fragebogen durchgeführt. Usability-Tests stellen eine gängige Methode im Bereich des User-Centered Designs zur Überprüfung der Gebrauchstauglichkeit dar. Testpersonen aus der Zielgruppe führen vordefinierte Aufgaben durch, um Probleme und positive Aspekte der Anwendung ausfindig zu machen [vgl. Low13, S. 95]. Ein Testleitfaden stellt die Grundlage für Usability-Tests dar. Während der Bearbeitung der Aufgaben sind Nutzer*innen angehalten, laut zu denken. Usability-Tests werden in der Regel durch Interviews oder Fragebögen ergänzt [vgl. Low13, S. 100].

Die Evaluierung fand mit sieben Personen aus der Zielgruppe "Studierende" statt. Die prototypische Anwendung war auf einem Android-Gerät vorinstalliert. Der Server lief auf einem lokalen Rechner, die Kommunikation zwischen Client und Server erfolgte über die lokale IP-Adresse oder einer durch NGROK[1] für den Testzeitraum bereitgestellten öffentlichen Uniform Resource Locator (URL), wodurch der Zugriff auf einen lokalen Server aus dem Internet möglich wurde.

Der durchgeführte Usability-Test bestand aus einem Testleitfaden und einem Fragebogen, welche im Anhang C zu finden sind. Nach einer kurzen Erklärung der Vorgehensweise und Beantwortung allgemeiner Fragen wurden die Studierenden darum gebeten, sich in das Einstiegsszenario hineinzuversetzen und sieben Testaufgaben durchzuführen. Diese decken die Funktionalität des Prototyps ab. Im anschließenden Fragebogen wurde zum einen überprüft, wie erfolgreich die Testaufgaben durchgeführt werden konnten. Zum anderen wurde die Usability anhand eines standardisierten Fragebogens, der System Usability Scale (SUS), messbar [Bro96]. Die SUS besteht aus zehn Aussagen, die abwechselnd positiv und negativ formuliert sind und nach der Likert-Skala [Jos15] bewertet werden. Die Skala der SUS reicht bei den standardmäßig zehn Aussagen von 0

[1] https://ngrok.com

bis 100 Punkten, wobei bei einem Wert ab 68 Punkten von einer "mindestens guten Gebrauchstauglichkeit" gesprochen wird [vgl. SL12, S. 204].

Der sogenannte SUS Score, also das Ergebnis der SUS, errechnet sich folgendermaßen: jede Aussage trägt zwischen null und vier Punkten zum Score bei. Bei positiv formulierten Aussagen wird der angekreuzte Wert minus eins $(x_i - 1)$ gerechnet, bei negativ formulierten Aussagen hingegen fünf minus den angekreuzten Wert $(5 - x_i)$. Der SUS Score ergibt sich aus der Summe dieser Werte multipliziert mit 2,5. [vgl. SL12, S. 198]

Der Fragebogen wurde um Aussagen ergänzt, die sich auf Chatbots spezialisierten. Es wurde beispielsweise gefragt, ob die Antworten zu den Eingaben passten oder ob der Chatbot einen sympathischen Eindruck hinterlassen hatte. In abschließenden Freitextfeldern konnten positive und negative Aspekte aufgelistet und sonstiges Feedback angemerkt werden.

7.2 Ergebnisse

Im weiteren Verlauf werden die Ergebnisse der Usability-Tests zusammengefasst. Die ausgefüllten Fragebögen sind auf dem dieser Arbeit beiliegenden Speichermedium zu finden (siehe Datei `ui_tests_scan.pdf` im Ordner `ui_tests`).

Allgemeine Angaben zur Befragtengruppe
Die Gruppe an Testpersonen bestand aus drei männlichen und vier weiblichen Studierenden im Alter von 24 bis 29 Jahren. Jede*r Befragte kann als technikaffin eingeordnet werden, da das Smartphone mindestens einmal täglich genutzt wird. Vier der Teilnehmenden nutzen das Smartphone mehrmals täglich. Fünf der sieben Personen verwenden Smartphone-Apps speziell zum Lernen von Sprachen. Zu den genutzten Anwendungen zählen Babbel, Duolingo, Dict CC Plus und Der Die Das. Außerdem haben vier der sieben Studierenden Erfahrung im Umgang mit Chatbots, allerdings vorwiegend im Bereich des Kundenservices.

Lösbarkeit der Testaufgaben
Die einzelnen Aufgaben wurden von den Proband*innen auf einer fünfstufigen Likert-Skala hinsichtlich ihrer Lösbarkeit bewertet. Die Skala reichte von "Stimme nicht zu" und "Stimme eher nicht zu" über "Weder noch" bis hin zu "Stimme eher zu" und "Stimme zu". Die Abbildung 7.1 fasst die einzelnen Aufgaben und deren Bewertung durch ihre Nutzer*innen zusammen. Die Farben grün, hellgrün, grau, rosa und braun visualisieren die Wertung und die Zahl steht für die Personen, welche die jeweilige Wertung zu den einzelnen Aussagen vorgenommen haben.

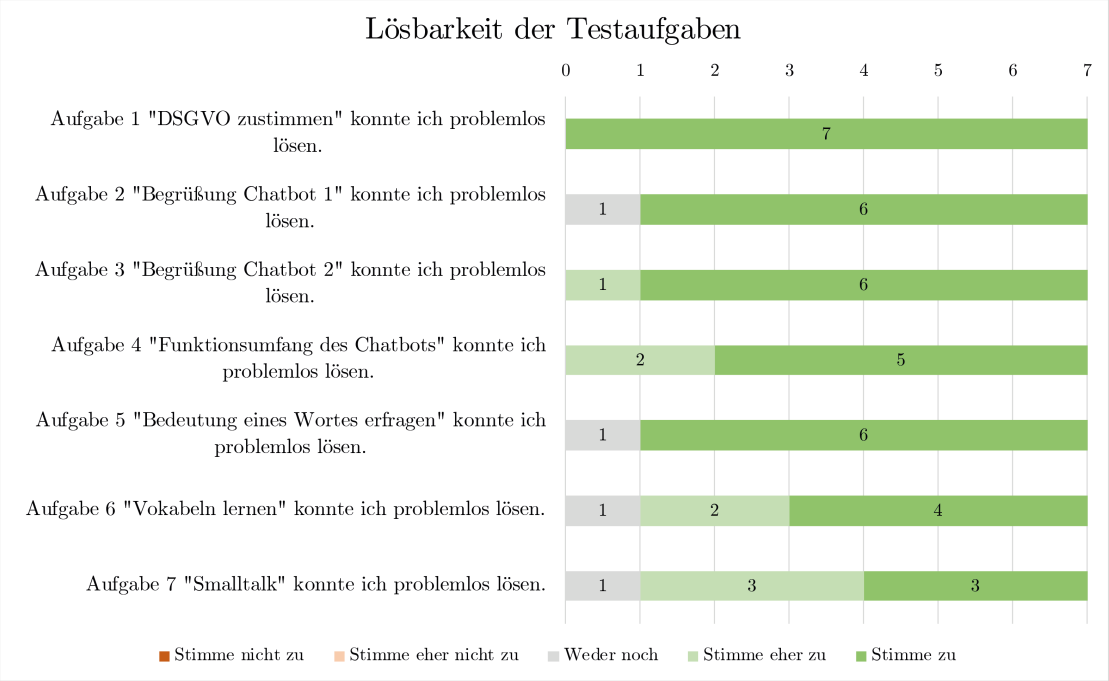


Abbildung 7.1: Ergebnisse der Testaufgaben

Die erste Aufgabe behandelte die Zustimmung zur DSGVO. Dabei wurde zunächst die Verständlichkeit des Textes in Bezug auf die Zustimmung zur DSGVO, welche für den Verbindungsaufbau zum Server notwendig ist, geprüft. Diese Aufgabe konnte von allen problemlos gelöst werden (siehe Abbildung 7.1).

In Aufgabe 2 wurden die Teilnehmer*innen vom Chatbot begrüßt und darum gebeten, sich namentlich vorzustellen. Größtenteils konnte diese Aufgabe problemlos gelöst werden, allerdings gab es vereinzelte Probleme bei der Extraktion der Entität "name", wenn einzig und allein der Name eingegeben wurde und dieser dem Chatbot nicht bekannt war. Mit Eingabe des Zusatzes "My name is ..." oder "I am ..." konnte dieses Problem umgangen werden.

Der Einstiegsdialog wurde in Aufgabe 3 fortgeführt. Der EnglishBot fragte eingangs nach dem Wohnort und anschließend, wie es der Person gefällt, dort zu wohnen. Diese Aufgabe konnte von sechs Teilnehmenden problemlos bearbeitet werden, indes traten ähnliche Schwierigkeiten wie bei Aufgabe 2 auf. Bei einer Eingabe eines dem Chatbot unbekannten Wohnorts konnte dieser ohne Zusätze wie "I live in ..." nicht extrahiert werden.

Nach der dritten Aufgabe war der sogenannte Einführungsdialog abgeschlossen. In der Aufgabe 4 ging es um die Identifikation des Funktionsumfangs des Chatbots. Für fünf der sieben Personen war es auf Anhieb verständlich, wobei der Chatbot behilflich sein

konnte. Für die restlichen Teilnehmer*innen war der Unterschied zwischen "Teach me vocabulary" und "Tell me the meaning of a word" nicht ganz klar. Hierbei kann eine Anpassung der Dialoge Abhilfe abschaffen.

In den darauffolgenden Aufgaben wurden die drei Lernmodi auf ihre Funktionsfähigkeit geprüft. Zunächst mussten die Testpersonen in Aufgabe 5 die Bedeutung des Wortes "teaching assistant" herausfinden. Dies funktionierte bei sechs Personen ohne Weiteres. Eine Person benötigte zwei weitere Anläufe, da unklar war, wie die Eingabe erfolgen sollte. In Aufgabe 6 wurde das Vokabelquiz getestet. Die Mehrheit konnte diese Aufgabe mühelos lösen. Einige Teilnehmer*innen hatten Schwierigkeiten zu verstehen, dass die Vokabeln im Rahmen eines Quiz geübt werden. Wie bereits im Absatz zuvor erwähnt, könnte die Anwendung auch hier durch Anpassung der Dialoge verbessert werden. In der letzten Aufgabe wurden die Smalltalkfähigkeiten des Chatbots untersucht. Hierbei traten im Vergleich zu den anderen Aufgaben die meisten Probleme auf. Die Aufgabenstellung besagt, dass zunächst dem Gesprächsverlauf gefolgt werden soll und erst im Anschluss der Name, das Alter und der Wohnort des Chatbot herausgefunden werden soll. Nach Verlassen des Vokabelquiz wurde den Nutzer*innen die Möglichkeit gegeben, weiterhin mit dem EnglishBot zu chatten. Bei einer Zustimmung startete der Bot eine Konversation über ein zufälliges Thema. In der Version des Prototyps ist nur das Thema "studying" verfügbar, sodass ein Dialog darüber initiiert wurde. Anfänglich erkundigte sich der Chatbot, ob die Anwendenden studieren. Bei einer Bejahung folgten Fragen nach dem Studienfach, dem Semester und den Plänen nach Beendigung des Studiums. Vier der sieben Proband*innen versuchten die Fragen aus der Aufgabenbeschreibung zu stellen, ohne zunächst dem Gesprächsverlauf zu folgen und die vom Chatbot gestellten Fragen zum Studium zu beantworten. Der Chatbot befand sich demnach in einem Zustand, indem er die Fragen nicht beantworten konnte. Daher stimmten nur drei der sieben Personen der Aussage zu, die Aufgabe 7 problemlos gelöst zu haben (siehe Abbildung 7.1). Obgleich dieses Fehlverhalten unter anderem auf die Aufgabenstellung zurückzuführen ist, ist eine flexiblere Gestaltung der Dialoge in Konversationen über zufällige Themen wünschenswert.

Ergebnisse der SUS

Im Anschluss an die Testaufgaben wurden 16 Aussagen bewertet. Zehn Aussagen waren Teil der SUS sind und sechs bezogen sich explizit auf den Chatbot. Die Ergebnisse der SUS sind in der Abbildung 7.2 zusammengefasst. Die x-Achse zeigt die Kürzel (p1 - p7) der Teilnehmenden, auf der y-Achse ist die Punkteskala der SUS aufgeführt. Die genauen Berechnungen sind auf dem beiliegenden Speichermedium in der Datei `auswertung_ui_tests.xlsx` zu finden.

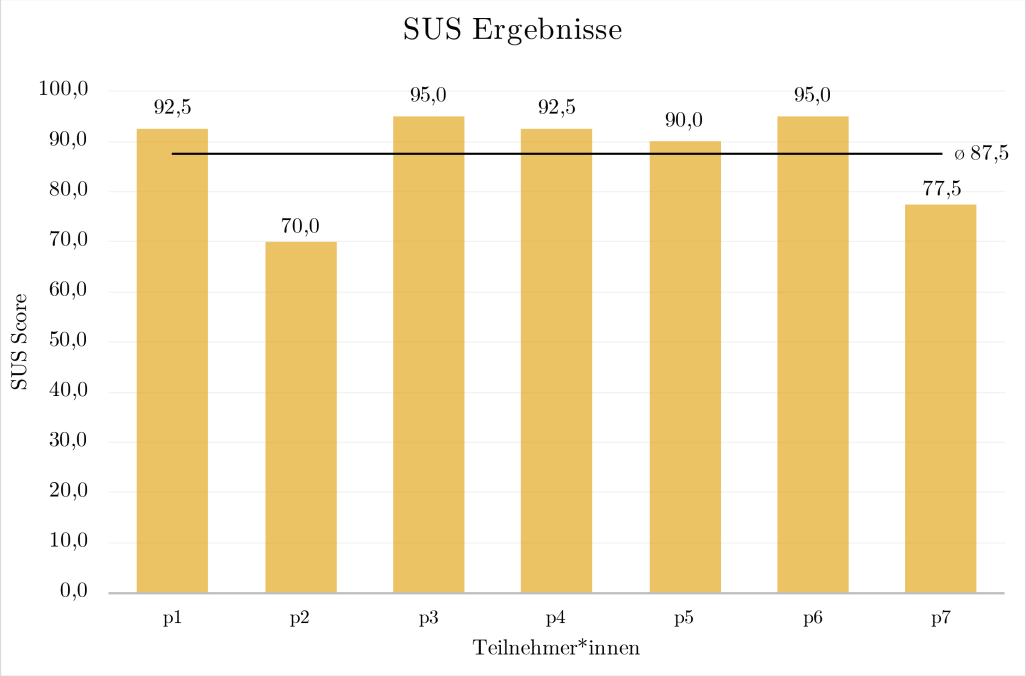


Abbildung 7.2: Ergebnisse der SUS

Durchschnittlich wurde ein SUS Score von 87,5 Punkten erreicht. Der niedrigste Score liegt bei 70 Punkten und der höchste Wert bei 95 Punkten. Der Mindestwert von 68 Punkten wurde zwar mit einem Durchschnittswert von 87,5 Punkten deutlich überschritten, jedoch ist die Anwendung noch weiter ausbaufähig. Der SUS Score stellt lediglich eine Tendenz dar. Da er keine exakten Rückschlüsse darauf gibt, in welchen konkreten Bereichen Entwicklungspotential besteht, wurde der Fragebogen um chatbotspezifische Aussagen und abschließende Freitextfelder ergänzt.

Chatbotspezifische Aussagen

Einen Überblick über die Ergebnisse der Bewertung der chatbotspezifischen Aussagen gibt die Abbildung 7.3.

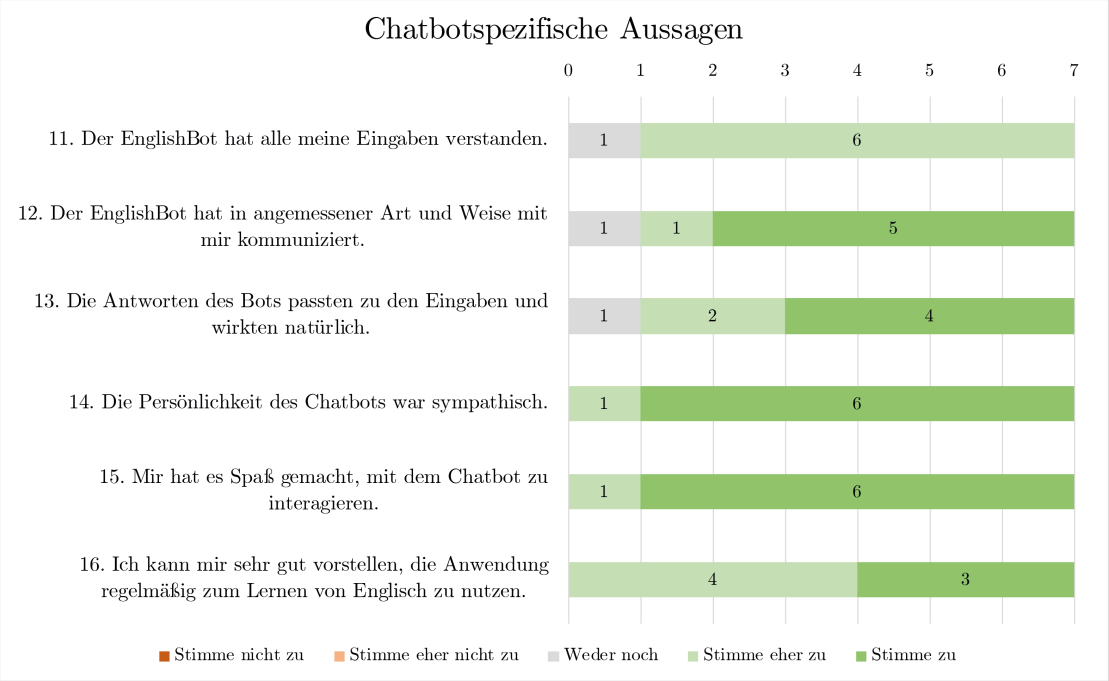


Abbildung 7.3: Ergebnisse der chatbotspezifischen Aussagen

Daraus ist zu entnehmen, dass der Chatbot insgesamt einen positiven Eindruck hinterlassen hat, da sich die Bewertungen nur im neutralen ("weder noch") bis positiven Bereich ("stimme eher zu", "stimme zu") befinden. Verbesserungspotential besteht vor allem im Verständnis der Eingaben und hinsichtlich der darauffolgenden Antworten. Alle Teilnehmer*innen stimmten der Aussage zu oder eher zu, die Anwendung regelmäßig zum Lernen von Englisch zu nutzen.

Sonstige Angaben

In den abschließenden Freitextfeldern konnten die Testpersonen die positiven und negativen Aspekte der Anwendung beschreiben. Positiv aufgefallen ist vor allem der Charakter des EnglishBot. Er wurde als "sehr sympathisch", "freundlich" und "lustig" empfunden. Seine Antworten wurden als "hilfreich" und "amüsant" erachtet. Überdies wurden die "Emoticons in den Antworten des Bot" sowie die "Natürlichkeit der Sprache des Bot" und dessen "schnelle Beantwortung und Reaktion" positiv bewertet. Das Vokabelquiz wurde von drei Personen für gut befunden, da dabei "spielerisch gelernt werden kann". Ebenso wurde es als lernförderlich empfunden, dass "die App nur in Englisch kommuniziert". Des Weiteren wurde die Konformität mit der DSGVO und die Personalisierung der Anwendung hervorgehoben.

Die Anwendenden wünschten sich neben einem besseren Verständnis ihrer Eingaben auch eine Erweiterung der Themen im Rahmen des Smalltalks. Darüber hinaus fänden zwei der sieben Personen Shortcuts oder Buttons nützlich, um schnell in andere Modi

wechseln zu können. Der Einführungsdialog signalisierte ihnen, dass sie sich strikt an die Beispielformulierungen für den Aufruf der Funktionen zu halten haben und nicht von dieser Eingabeform abzuweichen ist, obwohl der Chatbot auch den Beispielformulierungen ähnliche Eingaben verstehen kann. Ebenso ist aufgefallen, dass es Schwierigkeiten bei der Differenzierung der drei Modi des Chatbots gab. Die drei Modi wurden zwar erkannt, allerdings waren deren Unterschiede nicht von vornherein eindeutig. Eine Überarbeitung der konkreten Formulierung des Einführungsdialogs würde Abhilfe schaffen. Beim Vokabelquiz fänden es vier von sieben Personen sinnvoll, wenn es bei Nichtkenntnis des abgefragten Wortes die Möglichkeit gäbe, mehr als bloß einen Tipp zu erhalten. Eine der befragten Personen würde eine menschliche Namensgebung des Chatbots begrüßen. Im abschließenden Feedback-Feld wurde außerdem vorgeschlagen, die Themen des Vokabelmodus zu erweitern (beispielsweise Transport, Einkaufen oder Sport) und die Definitionen "in eigenen Sätzen zu verpacken, um menschlicher/persönlicher zu wirken".

7.3 Zusammenfassung

Insgesamt wurde die prototypische Anwendung des EnglishBot überwiegend positiv bewertet und erreichte einen SUS Score von durchschnittlich 87,5 Punkten. Die implementierten Funktionen konnten größtenteils ohne Schwierigkeiten angewandt werden. Folglich kann von einer "mindestens guten Gebrauchstauglichkeit" gesprochen werden. Weitere Entwicklungsmöglichkeiten sind vor allem im Bereich des Verständnisses der Eingaben zu sehen. Die Erweiterung von Gesprächsthemen im Rahmen des Smalltalks sowie die Ergänzung weiterer Vokabelmodule im Bereich des Quiz (zum Beispiel Vokabular zum Thema Transport oder Freizeitaktivitäten) könnten zudem den Chatbot noch abwechslungsreicher und dessen Nutzung attraktiver machen.

8 Fazit

Das letzte Kapitel fasst die gewonnenen Erkenntnisse dieser Arbeit zusammen. Es werden aufgetretene Schwierigkeiten und Herausforderungen im Zusammenhang mit der Implementierung mit dem Rasa Stack erläutert und abschließend ein Ausblick gegeben.

8.1 Zusammenfassung der Arbeit

Das Ziel dieser Arbeit war die prototypische Entwicklung eines Chatbot, mit dessen Hilfe die Konversationskenntnisse in Englisch durch das Führen schriftlicher Dialoge und dem Ausbau des Wortschatzes verbessert werden können. Dafür wurde zunächst der Begriff Chatbot definiert, dessen Architektur erläutert sowie eine Klassifizierung anhand seines Antwortverhaltens vorgenommen. Es wurde auf Natural Language Understanding und die wichtigsten NLP-Methoden näher eingegangen. Nach der Vorstellung von Chatbot-Anwendungen erfolgte die Untersuchung von Dialogflow und des Rasa Stacks, wobei Letzteres für die Entwicklung eines Chatbots im Zweitspracherwerb für geeigneter befunden wurde. Dialogflow – die Chatbot-Plattform des großen Technologie-Unternehmens Google – mag zwar hinsichtlich der Bedienbarkeit punkten, jedoch bietet sie keine Alternative zur Datenspeicherung und -verarbeitung in der Google Cloud. Der Rasa Stack ist nicht nur Open-Source und der Quellcode damit öffentlich einseh- und veränderbar, sondern ermöglicht durch einen lokalen Serverbetrieb die volle Kontrolle über die eigenen Daten.

Im Anschluss an den Vergleich von Dialogflow mit dem Rasa Stack wurden Interviews mit Sprachlehrenden geführt, aus denen hervorging, dass durchaus Potential in Chatbots als Hilfsmittel gesehen wird, da sie einen neuen Ansatz für die Wissensvermittlung darstellen und durch ihr interaktives Verhalten Nähe zu ihren Anwendenden schaffen. In Hinblick auf die vier Fertigkeiten beim Lernen von Sprachen wird in Chatbots vor allem eine Möglichkeit gesehen, das Hören, Lesen und Schreiben zu trainieren. Die Aneignung der Aussprache wird als schwierig empfunden, da diese nach Einschätzung der Sprachlehrenden nur im direkten Kontakt erlernt werden kann. Zur Verbesserung der Konversationskenntnisse könnten Chatbots als eine Art digitaler Tandempartner*innen agieren, die in natürlicher Sprache kommunizieren und Feedback zu Fehlern in der Rechtschreibung und Grammatik geben. Zudem kann ein integriertes einsprachiges Wörterbuch von Vorteil sein. Als besonders wichtig wurde außerdem erachtet, dass

Chatbot-Anwendungen Lernziele verfolgen und den Lernfortschritt klar dokumentieren, um die Motivation der Nutzer*innen zu stärken.

Von den Resultaten der Anforderungsanalyse ausgehend wurden die funktionalen und nicht-funktionalen Anforderungen definiert und der “EnglishBot” konzipiert und entworfen. Dabei wurden zukünftige Nutzer*innen im Rahmen des User-Centered Designs miteinbezogen. Im Ergebnis wurde ein Prototyp mit dem Rasa Stack implementiert, der die *muss*-Anforderungen sowie einen Großteil der nicht-funktionalen Anforderungen des Konzepts abdeckt. Es zeigte sich, dass sich der Rasa Stack durchaus eignet, um einen Chatbot für den Zweitspracherwerb zu entwickeln. Jedoch ist die Implementierung auch mit Herausforderungen verbunden, die zum einen Sprachlern-Chatbots im Allgemeinen betreffen und zum anderen dem Umstand geschuldet sind, dass Rasa als Open-Source zur Verfügung steht und einen Machine Learning Ansatz verfolgt.

Eine Anwendung zum Lernen einer Sprache soll einen möglichst großen und diversen Umfang an Thematiken haben, welcher in einem immensen Bedarf an Daten resultiert. Zum Zeitpunkt der Programmierung gab es keine Datensammlungen, die einer Anwendung wie dem EnglishBot genügten, sodass eine manuelle Erstellung von Modulen notwendig war.

Hervorzuheben ist auch, dass durch die Nutzung einer Open-Source Software Nachteile entstehen können, da diese abhängig von ihrer Community ist und Software-Dokumentationen oft nicht auf dem aktuellen Stand sind. Während des Verfassens dieser Arbeit wurde beispielsweise der Rasa Stack vollständig überarbeitet. Ein Umstieg auf die aktuellste Version bot sich wegen des Kommandozeilentools und weiteren Verbesserungen an. Allerdings wurde die Dokumentation erst sukzessive ergänzt, wodurch ein Debugging der Anwendung erschwert wurde.

Der EnglishBot verfolgt bei der Erkennung der Intents und der Vorhersage der nächsten Action einen Ansatz, der auf Machine Learning beruht. Dieser hat zwar den Vorteil, dass Eingaben der Nutzer*innen auch dann erkannt werden, wenn sie nicht vollständig mit den Trainingsdaten übereinstimmen. Gleichwohl ist die Erkennung der Intents ausschlaggebend für die Bewertung der Funktionsfähigkeit eines Chatbot. Demzufolge müssen genügend Trainingsdaten zur Verfügung stehen. Da der EnglishBot keinem standardisierten Chatbot entspricht, kann jedoch nicht auf vortrainierte Modelle zurückgegriffen werden. Demnach müssen die Trainingsdaten von Grund auf erstellt werden. Dies kann eine durchaus zeitintensive Aufgabe sein.
Intents, die zu ähnliche Trainingsdaten aufweisen, stellen ein weiteres Problem dar. Eine Wörterbuchabfrage, die mit "What’s" gefolgt von dem gesuchten Wort aufgerufen wird, ist beispielsweise schwer von der Frage "What’s up?" zu unterscheiden. Insgesamt steigt mit wachsender Anzahl an Intents auch die Komplexität der Anwendung und die

Wahrscheinlichkeit, dass sich Intents überschneiden. Eine Eingrenzung der Domäne, in denen der Chatbot agiert, ist unerlässlich.

Trotz der genannten Herausforderungen in der Implementierung konnte der EnglishBot in den abschließenden Usability-Tests mit sieben Studierenden in Bezug auf die Zielerreichung mit einem durchschnittlichen SUS Score von 87,5 Punkten überzeugen.

8.2 Ausblick

Es bleibt festzuhalten, dass der in dieser Arbeit entstandene Prototyp sich auf die Untersuchung der grundsätzlichen Eignung des Rasa Stacks für Anwendungen wie den EnglishBot beschränkt. Infolgedessen bestehen trotz der guten Funktionsfähigkeit noch Erweiterungsmöglichkeiten. Für den vollen Funktionsumfang und einer Ausschöpfung des gesamten Potentials des EnglishBot ist eine Umsetzung der *soll*- und *kann*-Anforderungen sowie der Datenspeicherung notwendig. Darüber hinaus ist es für die Vervollständigung der App unabdingbar, eine verschlüsselte Verbindung zwischen Client und Server einzurichten und zugleich eine serverseitige Dockerisierung vorzunehmen.

Nach aktuellem Stand des Prototyps bleibt ein trainiertes Modell während des Gesprächs mit den Nutzer*innen unverändert und kann ausschließlich auf Initiative der Entwickler*innen mit neuen Trainingsdaten erweitert werden. In Hinblick auf die Abhängigkeit der Chatbots von ihren Trainingsdaten wäre es erstrebenswert, als Verbesserungsmaßnahme einen Feedback-Mechanismus zu implementieren, welcher ein Training auch nach dem Deployment ermöglicht.
Nach einem Vorschlag von Hancock et al. kann ein Chatbot dahingehend befähigt werden, aus aktuellen Konversationen Trainingsdaten zu extrahieren. Die Vorhersage hinsichtlich der Zufriedenheit der Nutzer*innen mit den Antworten des Bot bestimmt die weitere Vorgehensweise. Entsprechen diese den Erwartungen der Nutzer*innen, werden die Eingaben jener nachgeahmt. Andernfalls wird nach Feedback gebeten, welches schließlich die neuen Trainingsdaten darstellt. [vgl. Han19]

Das Potential von Chatbots ist groß und deren Einsatzgebiete sind vielfältig. Im Zweitspracherwerb können sie nützliche Hilfsmittel sein. Der Fortschritt in den Bereichen des NLU und des maschinellen Lernens bedingt die Leistungsfähigkeit von Chatbots und könnte in Zukunft dazu führen, dass Themenfelder weniger stark eingegrenzt werden müssen, wodurch Chatbots dem weiten Spektrum einer Sprache noch besser genügen würden.

Literaturverzeichnis

[Bal09] Helmut Balzert. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. 3. Aufl. Heidelberg: Spektrum Akademischer Verlag, 2009. DOI: 10.1007/978-3-8274-2247-7.

[BKL09] Steven Bird, Ewan Klein & Edward Loper. *Natural Language Processing with Python*. 1. Aufl. Cambridge: O'Reilly Media, Inc., 2009.

[Boc17] Tom Bocklisch et al. „Rasa: Open Source Language Understanding and Dialogue Management". In: *Computing Research Repository (CoRR)* abs/1712.05181 (2017). URL: http://arxiv.org/abs/1712.05181 (besucht am 02. 06. 2019).

[Bro96] John Brooke. „SUS-A Quick and Dirty Usability Scale". In: *Usability evaluation in industry* 189.194 (1996). URL: https://www.crcpress.com/product/isbn/9780748404605.

[CCH17] Yun-Nung Chen, Asli Celikyilmaz & Dilek Hakkani-Tür. „Deep Learning for Dialogue Systems". en. In: *Proceedings of ACL 2017, Tutorial Abstracts*. Vancouver, Canada: Association for Computational Linguistics, 2017, S. 8–14. DOI: 10.18653/v1/P17-5004. (Besucht am 13. 09. 2019).

[CH18] Bo Ae Chun & Hae Ja Heo. „The Effect of Flipped Learning on Academic Performance As an Innovative Method for Overcoming Ebbinghaus' Forgetting Curve". In: *Proceedings of the 6th International Conference on Information and Education Technology*. ICIET '18. Osaka, Japan: ACM, 2018, S. 56–60. DOI: 10.1145/3178158.3178206.

[Col11] Ronan Collobert et al. „Natural Language Processing (almost) from Scratch". en. In: *The Journal of Machine Learning Research* 12 (2011), S. 2493–2537.

[Dia19a] Dialogflow. *Agents Overview*. 2019. URL: https://cloud.google.com/dialogflow/docs/agents-overview (besucht am 06. 08. 2019).

[Dia19b] Dialogflow. *Entities Overview*. 2019. URL: https://cloud.google.com/dialogflow/docs/entities-overview (besucht am 06. 08. 2019).

[Dia19c] Dialogflow. *Follow-up intents*. 2019. URL: https://cloud.google.com/dialogflow/docs/contexts-follow-up-intents (besucht am 16. 06. 2019).

[Dia19d] Dialogflow. *Fulfillment Overview*. 2019. URL: https://cloud.google.com/dialogflow/docs/fulfillment-overview (besucht am 06. 08. 2019).

[Dia19e] Dialogflow. *Input and output contexts*. 2019. URL: https://cloud.google.com/dialogflow/docs/contexts-input-output (besucht am 16. 06. 2019).

[Dia19f] Dialogflow. *Intents Overview*. 2019. URL: https://cloud.google.com/dialogflow/docs/intents-overview (besucht am 06. 08. 2019).

[Dia19g] Dialogflow. *Knowledge connectors*. 2019. URL: https://cloud.google.com/dialogflow/docs/knowledge-connectors (besucht am 16. 06. 2019).

[Dia19h] Dialogflow. *Machine Learning*. 2019. URL: https://cloud.google.com/dialogflow/docs/agents-settings (besucht am 16. 06. 2019).

[Egg15] Eggbun. *Learn Korean with Eggbun*. 2015. URL: https://web.eggbun.net/ (besucht am 23. 05. 2019).

[Exc19] ExcellentESL4U. *ESL University Vocabulary*. 2019. URL: https://www.excellentesl4u.com/esl-university-vocabulary.html (besucht am 08. 08. 2019).

[FM11] Ian Fette & Alexey Melnikov. „The WebSocket Protocol". In: *RFC* 6455 (2011), S. 1–71. DOI: 10.17487/RFC6455. URL: https://doi.org/10.17487/RFC6455.

[GH17] Yoav Goldberg & Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[Gon03] Dafne Gonzalez. „Teaching and learning through chat: A taxonomy of educational chat for EFL/ESL". In: *Teaching English with Technology* 3 (2003), S. 57–69.

[GPJ18] Palash Goyal, Sumit Pandey & Karan Jain. *Deep Learning for Natural Language Processing: Creating Neural Networks with Python*. 1. Aufl. Apress, 2018.

[Han19] Braden Hancock et al. „Learning from Dialogue after Deployment: Feed Yourself, Chatbot!" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, S. 3667–3684. DOI: 10.18653/v1/P19-1358.

[Hay94] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. 1. Aufl. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994.

[Ili16] Alexandru Iliescu. *Mondly launches first voice chatbot for learning languages*. 2016. URL: https://www.mondly.com/blog/2016/08/25/mondly-chatbot-press-release/ (besucht am 22. 05. 2019).

[JM09] Dan Jurafsky & James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. 2. Aufl. Prentice Hall series in artificial intelligence. Upper Saddle River, N.J: Pearson Prentice Hall, 2009.

[JM19] Dan Jurafsky & James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. 3. Aufl. Unpublished draft. 2019. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf (besucht am 21. 05. 2019).

[Jos15] Ankur Joshi et al. „Likert Scale: Explored and Explained". In: *British Journal of Applied Science & Technology* 7.4 (2015), S. 396–403. ISSN: 22310843. DOI: 10.9734/BJAST/2015/14975.

[KD18] Rashid Khan & Anik Das. *Build better Chatbots: a complete guide to getting started with chatbots*. 1. Aufl. New York: Apress, 2018.

[Kuk12] Agnes Kukulska-Hulme. „Mobile-assisted language learning". In: *The Encyclopedia of Applied Linguistics*. Hrsg. von Carol A. Chapelle. Blackwell Publishing Ltd., 2012. URL: http://oro.open.ac.uk/42093/.

[Lan19a] LanguageTool. *Developing robust rules - LanguageTool Wiki*. 2019. URL: http://wiki.languagetool.org/developing-robust-rules (besucht am 13. 09. 2019).

[Lan19b] LanguageTool. *Development Overview - LanguageTool Wiki*. 2019. URL: http://wiki.languagetool.org/development-overview (besucht am 13. 09. 2019).

[Lan19c] LanguageTool. *Finding errors using n-gram data - LanguageTool Wiki*. 2019. URL: http://wiki.languagetool.org/finding-errors-using-n-gram-data (besucht am 13. 09. 2019).

[Lan19d] LanguageTool. *HTTP Server - LanguageTool Wiki*. 2019. URL: http://wiki.languagetool.org/http-server (besucht am 13. 09. 2019).

[Lan19e] LanguageTool. *Public HTTP Proofreading API - LanguageTool Wiki*. 2019. URL: http://wiki.languagetool.org/public-http-api (besucht am 13. 09. 2019).

[Lei08] Sebastian Leitner. *So lernt man lernen: der Weg zum Erfolg*. ger. 16. Aufl. Herder-Spektrum 5060. Freiburg im Breisgau: Herder, 2008.

[LMP01] John D. Lafferty, Andrew McCallum & Fernando C. N. Pereira. „Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, S. 282–289.

[Low13] Travis Lowdermilk. *User-Centered Design*. 1. Aufl. Sebastopol: O'Reilly, 2013.

[MHW17] Thilo Michael, Stefan Hillmann & Benjamin Weiss. *Alex - Advosiry Artificial Conversational Agent*. 2017. URL: https://alex.qu.tu-berlin.de/ (besucht am 23. 05. 2019).

[Mil95] George A. Miller. „WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), S. 39–41. DOI: 10.1145/219717.219748.

[MK16] Andreas Meier & Michael Kaufmann. *SQL- & NoSQL-Datenbanken*. 8. überarbeitete und erweiterte Aufl. eXamen.press. Berlin Heidelberg: Springer Vieweg, 2016.

[MN12] Tayebeh Mosavi Miangah & Amin Nezarat. „Mobile-Assisted Language Learning". In: *International Journal of Distributed and Parallel Systems* 3 (2012), S. 309–319. DOI: 10.5121/ijdps.2012.3126.

[MO08] Michael McCarthy & Felicity O'Dell. *Academic vocabulary in use*. eng. 1. Aufl. Cambridge University Press, 2008.

[MR11] Rada Mihalcea & Dragomir Radev. *Graph-based natural language processing and information retrieval*. 1. Aufl. Cambridge ; New York: Cambridge University Press, 2011.

[Nab03] Daniel Naber. „A Rule-Based Style and Grammar Checker". Diplomarbeit. Universität Bielefeld, 2003. URL: http://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf (besucht am 01. 05. 2019).

[PC14] Lindsay Page & Ben Castleman. *Reduction Of Summer Melt - A Strategic Approach*. 2014. URL: https://success.gsu.edu/initiatives/reduction-of-summer-melt/ (besucht am 23. 05. 2019).

[Pra18] Angara Prashanti. „Towards a Deeper Understanding of Current Conversational Frameworks through the Design and Development of a Cognitive Agent“. Masterthesis. University of Victoria, 2018. URL: https://cloud.google.com/dialogflow/docs/contexts-input-output (besucht am 16. 09. 2019).

[Pya17] Andrey Pyankov. *Andy English Bot - Chat and Learn English with Robot*. 2017. URL: https://andychatbot.com/ (besucht am 23. 05. 2019).

[Raj19] Sumit Raj. *Building Chatbots with Python: Using Natural Language Processing and Machine Learning*. 1. Aufl. Berkeley, CA: Apress, 2019.

[Ras19a] Rasa. *Actions*. 2019. URL: http://rasa.com/docs/rasa/core/actions/ (besucht am 22. 06. 2019).

[Ras19b] Rasa. *Architecture*. 2019. URL: http://rasa.com/docs/rasa/user-guide/architecture (besucht am 21. 06. 2019).

[Ras19c] Rasa. *Choosing a Pipeline*. 2019. URL: http://rasa.com/docs/rasa/nlu/choosing-a-pipeline/ (besucht am 21. 06. 2019).

[Ras19d] Rasa. *Domains*. 2019. URL: http://rasa.com/docs/rasa/core/domains/ (besucht am 22. 06. 2019).

[Ras19e] Rasa. *Forms*. 2019. URL: http://rasa.com/docs/rasa/core/forms/ (besucht am 22. 06. 2019).

[Ras19f] Rasa. *Getting Started with Rasa*. 2019. URL: https://rasa.com/docs/getting-started (besucht am 29. 05. 2019).

[Ras19g] Rasa. *Language Support*. 2019. URL: http://rasa.com/docs/rasa/nlu/language-support/ (besucht am 22. 06. 2019).

[Ras19h] Rasa. *Policies*. 2019. URL: https://rasa.com/docs/rasa/core/policies (besucht am 21. 08. 2019).

[Ras19i] Rasa. *Slots*. 2019. URL: http://rasa.com/docs/rasa/core/slots/ (besucht am 22. 06. 2019).

[Ras19j] Rasa. *Stories*. 2019. URL: http://rasa.com/docs/rasa/core/stories/ (besucht am 22. 06. 2019).

[Red97] Stuart Redman. *English vocabulary in use: pre-intermediate and intermediate*. Cambridge, U.K: Cambridge University Press, 1997.

[SA04] Bayan Abu Shawar & Eric Atwell. „Accessing an Information System by Chatting“. In: *Natural Language Processing and Information Systems*. Hrsg. von Farid Meziane & Elisabeth Métais. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 407–412.

[Sal19] Anagha A. Salunke et al. „Designing Smart Agent Based Using Retrieval Model“. In: *International Journal For Technological Research In Engineering* 6.7 (2019), S. 5159–5163.

[Sch14] Martin Schmidt. *Sieben gute Gründe für mobiles Lernen*. Bitkom e.V. 2014. URL: https://www.bitkom.org/Bitkom/Publikationen/Sieben-gute-Gruende-fuer-mobiles-Lernen.html (besucht am 26. 04. 2019).

[Sch15] Ulf Schuetze. „Spacing techniques in second language vocabulary acquisition: Short-term gains vs. long-term memory“. en. In: *Language Teaching Research* 19.1 (2015), S. 28–42.

[Sha77] John J. Shaughnessy. „Long-Term Retention and the Spacing Effect in Free-Recall and Frequency Judgments“. In: *The American Journal of Psychology* 90.4 (1977), S. 587.

[SL12] Jeff Sauro & James R. Lewis. *Quantifying the user experience: practical statistics for user research*. 1. Aufl. Amsterdam; Waltham, MA: Elsevier/Morgan Kaufmann, 2012.

[Tor19] University of Toronto. *Glossary of University Terms | Future Students*. 2019. URL: `https://www.utm.utoronto.ca/future-students/parents-counsellors/glossary-university-terms` (besucht am 08. 08. 2019).

[Uni18] Chatbots für Universitäten und Hochschulen. *Chatbots für Universitäten und Hochschulen*. 2018. URL: `https://www.virtualspirits.com/de/chatbot-for-university.aspx` (besucht am 23. 05. 2019).

A Inhalte des Speichermediums

Der beiliegende USB-Stick weist den nachfolgenden Inhalt auf:

```
/
├── englishbot ..................Quellcode des EnglishBot.
│   ├── backend
│   │   └── ...
│   ├── frontend
│   │   └── ...
│   ├── images
│   │   └── ...
│   ├── languagetool
│   │   └── ...
│   └── README_INSTALL ..........Installationsanleitung EnglishBot.
├── englishbot_icons ............EnglishBot Logos.
├── interviews ..................Pseudonymisierte Transkriptionen
│   │                            der Interviews.
│   └── ...
├── masterthesis ................Diese Masterthesis im
│   │                            Latex-Format.
│   └── ...
├── ui_tests ....................Scans der für die Evaluierung
│   │                            durchgeführten Usability-Tests.
│   └── ...
├── auswertung_ui_tests.xlsx ....Auswertung der UI-Tests.
├── englishbot.mp4 ..............Screencast des EnglishBot.
├── interview_ergebnisse.xlsx ...Ergebnisse der Interviews.
├── interview_leitfaden.pdf .....Interview-Leitfaden der
│                                Anforderungsanalyse.
├── README ......................Beschreibung USB-Stick.
├── thesis_lampesberger.pdf .....Masterthesis im PDF-Format.
├── uitest.pdf ..................Leitfaden und Fragebogen der
│                                Evaluierung.
└── university_vocabulary.xlsx ...Vokabelsammung University.
```

B Interview-Leitfaden

Chatbots im Spracherwerb – Interviewleitfaden
Allgemeines
Name:
Beruf:
Geschlecht:

1. Einstieg:
- Begrüßung und Dank für die Zeit
- Einführung in das Thema
- Beschreibung Interviewablauf, ungefähre Dauer

2. Einstiegsfragen:
1. Haben Sie schon mal einen Chatbot benutzt?
2. Nutzen Sie technische Hilfsmittel im Unterricht?
   Wenn nein, warum nicht?
3. Wenn ja, was finden Sie an der Anwendung gut? Was würden Sie verbessern wollen?

--- Kernfragen ---

3. Einsatzbereiche:
1. Gibt es einen Bereich beim Englisch-Lernen, der sich besonders gut eignen würde um ihn mit einem Chatbot zu üben? (Aspekte: Wortschatz ausbauen, Grammatik üben)

2. Gibt es einen, der sich überhaupt nicht eignen würde aus pädagogischer Sicht? Warum nicht?

4. Eigenschaften und Funktionen:
1. Was denken Sie, sind die wichtigsten *Eigenschaften* eines solchen Chatbots?
   Beispiele:
   - Kommunikation über Sprache/Text oder beides
   - Möglichkeit sich Text ausgeben lassen (Sprachausgabe)
   - Möglichkeit sich Text übersetzen lassen zu können in Muttersprache
   - Anpassung an Lernfortschritt
   - mehrere Sprachlevel?
     - Wie kann festgestellt werden auf welchem Level sich jemand befindet? (Ideen?)

1. Was denken Sie, sind die wichtigsten *Funktionen* eines solchen Chatbots?
   Beispiele:
   - Wörterbuch (ENG-ENG, oder ENG-DE?)
   - Feedback bei Fehlern
   - Lernmaterialien enthalten
   - Erinnerung an das Lernen (Notifikation)
   - zu Beginn eine Levelfestellung

- Wie können insbesondere Konversationskenntnisse verbessert werden?
- Was würden Sie sagen, sind die wichtigsten drei Eigenschaften/Funktionen?

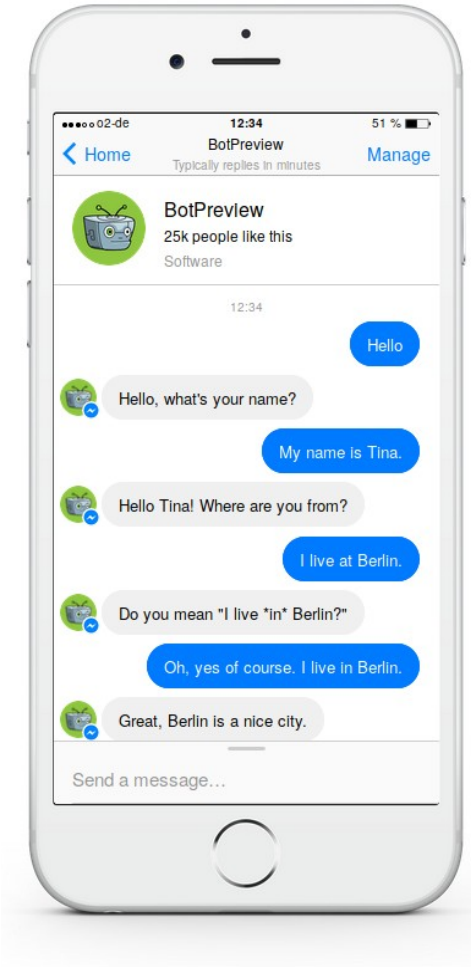
5. Feedback bei Fehlern:
- Wie sollte Feedback aussehen? Bei jedem kleinen Tippfehler? Oder nur bei groben Fehlern? (Nutzereinstellung?)
- Abhängig von Sprachlevel bzw. von der Anzahl an Fehlern, die pro (x-te) Minute gemacht werden und die levelabhängig ist
- Wie kann festgestellt werden, auf welchem Level sich jemand befindet?

6. Abschlussfrage:
- Können Sie sich vorstellen, einen solchen Sprachlern-Chatbot im Unterricht zu empfehlen?
- Haben Sie noch Anmerkungen, Ergänzungen, Ideen?

7. Rückblick und Ausblick
- Bedanken für die Zeit
- Information über Auswertung der Ergebnisse

Beispiel eines Dialogs zwischen Chatbot und Nutzer*in

Das folgende Bild zeigt eine beispielhafte Kommunikation mit einem Chatbot. Auf der rechten Seite ist die Eingabe des/der Nutzers/Nutzerin zu sehen. Der Chatbot antwortet abhängig von dieser Eingabe. Wenn Nutzer*innen Fehler machen, dann könnten diese von dem Chatbot korrigiert werden.



C Usability-Test

Usability-Test: EnglishBot

Persönliche Angaben:

Geschlecht: m ☐ w ☐ d ☐

Alter: _____

1. Wie häufig nutzt du das Smartphone?

zwei bis dreimal die Woche	täglich	zwei bis dreimal täglich	mehrmals täglich
☐	☐	☐	☐

2. Nutzt du auch Smartphone-Apps zu Lernen von Sprachen? ja ☐ nein ☐
   Wenn ja, welche?

3. Hast du Erfahrung im Umgang mit Chatbots? ja ☐ nein ☐
   Wenn ja, mit welchen?

Einstiegsszenario:

Du bist Student*in und wirst nächstes Semester ins Ausland gehen. Zur Vorbereitung suchst du eine mobile Anwendung, mit der du dein Englisch verbessern kannst. Insbesondere ist es wichtig für dich, universitätsspezifisches Vokabular zu lernen, um gut vorbereitet zu sein. Auf der Suche nach einer Anwendung stößt du auf die App „EnglishBot“. Die Anwendung ermöglicht Smalltalk mit dem EnglishBot zu üben, dabei gleichzeitig Rechtschreib- und Grammatikverbesserungen zu erhalten, nach Erklärungen für unbekannte Wörter zu fragen und universitätsspezifische Vokabeln zu üben.

Testaufgaben

Im Folgenden werden Testaufgaben vorgestellt, um die Funktionen der Anwendung testen. Bitte versuche die Aufgaben zu lösen und dabei bei jedem Schritt *laut zu denken*. Abschließend ist eine Beurteilung des Erfolgs beim Lösen der einzelnen Testaufgaben vorgesehen.

Aufgabe 1: DSGVO zustimmen

Starte die Anwendung. Du wirst um die Zustimmung der DSGVO gebeten. Bitte stimme der DSGVO zu.

Aufgabe 2: Begrüßung Chatbot 1

Du wirst zunächst vom Chatbot begrüßt und nach dem Namen gefragt. Stelle dich dem Chatbot namentlich vor.

Aufgabe 3: Begrüßung Chatbot 2

Anschließend wirst du nach dem Wohnort gefragt. Bitte teile dem Chatbot deinen Wohnort mit. Im Anschluss wirst du gefragt, wie es dir dort gefällt zu wohnen. Bitte teile auch das dem Chatbot mit.

Aufgabe 4: Funktionsumfang des Chatbots

Stell dir vor, du nutzt zum ersten Mal die Anwendung und möchtest wissen, wobei der Chatbot behilflich sein kann. Versuche herauszufinden, über welche drei Lernmodi der Chatbot verfügt.

Notiere sie bitte hier:

1.	2.	3.

Nun kannst du Aufgabe 5 lösen.

Aufgabe 5: Bedeutung eines Wortes erfragen

Versuche die Bedeutung des Wortes “teaching assistant” mithilfe des Chatbots herauszufinden.

Aufgabe 6: Vokabeln lernen

Du möchtest gerne Vokabeln lernen, die die Universität betreffen. Versuche den Chatbot dazu zu bringen, dir neue Wörter beizubringen. Ein Tipp: Falls du Hilfe benötigst, kannst du den Chatbot bitten, dir einen Tipp zu geben. Lerne drei Vokabeln und beende das Quiz.

Aufgabe 7: Smalltalk

Du wirst nun gefragt, ob du mit dem Chatbot chatten möchtest. Folge den Gesprächsverlauf und versuche im Anschluss herauszufinden, wie der Chatbot heißt, wie alt er ist und wo er wohnt. Bringe den Chatbot dazu, dir einen Witz zu erzählen.

Beurteilung der Testaufgaben:

Aufgabe 1 „DSGVO zustimmen“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 2 „Begrüßung Chatbot 1“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 3 „Begrüßung Chatbot 2“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 4 „Funktionsumfang des Chatbots“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 5 „Bedeutung eines Wortes erfragen“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 6 „Vokabeln lernen“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Aufgabe 7 „Smalltalk“ konnte ich problemlos lösen.

Stimme nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme zu
☐	☐	☐	☐	☐

Fragebogen:

Bitte kreuze an, inwiefern du den Aussagen zustimmst.

		Stimme nicht zu 1	Stimme eher nicht zu 2	Weder noch 3	Stimme eher zu 4	Stimme zu 5
1	Ich denke, dass ich den EnglishBot gerne häufig benutzen würde.	☐	☐	☐	☐	☐
2	Ich fand den EnglishBot unnötig komplex.	☐	☐	☐	☐	☐
3	Ich fand den EnglishBot einfach zu benutzen.	☐	☐	☐	☐	☐
4	Ich denke, dass ich technischen Support brauchen würde, um die Anwendung zu nutzen.	☐	☐	☐	☐	☐
5	Ich finde, dass die verschiedenen Funktionen der Anwendung gut integriert sind.	☐	☐	☐	☐	☐
6	Ich finde, dass es in der Anwendung zu viele Inkonsistenzen gibt.	☐	☐	☐	☐	☐
7	Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit dieser Anwendung sehr schnell lernen.	☐	☐	☐	☐	☐
8	Ich empfinde die Bedienung als sehr umständlich.	☐	☐	☐	☐	☐
9	Ich fühlte mich bei der Benutzung des Systems sehr sicher.	☐	☐	☐	☐	☐
10	Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.	☐	☐	☐	☐	☐
11	Der EnglishBot hat alle meine Eingaben verstanden.	☐	☐	☐	☐	☐
12	Der EnglishBot hat in angemessener Art und Weise mit mir kommuniziert	☐	☐	☐	☐	☐
13	Die Antworten des Bots passten zu den Eingaben und wirkten natürlich.	☐	☐	☐	☐	☐
14	Die Persönlichkeit des Chatbots war sympathisch.	☐	☐	☐	☐	☐
15	Mir hat es Spaß gemacht, mit dem Chatbot zu interagieren.	☐	☐	☐	☐	☐
16	Ich kann mir sehr gut vorstellen, die Anwendung regelmäßig zum Lernen von Englisch zu nutzen.	☐	☐	☐	☐	☐

Ein paar abschließende Fragen:

4. Was mochtest du am meisten an der Anwendung und warum?

5. Was würdest du an der Anwendung verbessern und warum?

6. Hast du noch weitere Anmerkungen, Feedback, Vorschläge?

Vielen Dank für deine Hilfe!

D Domäne des EnglishBot

Intents

Die Gesamtanzahl der Intents beläuft sich auf 46. Die Beispiele in der folgenden Tabelle sind lediglich ein Auszug der Daten. Die vollständige Liste an Trainingsbeispielen ist im Quellcode im Ordner `data\nlu_data` zu finden.

Intent	Beispiele
appraisal.thanks	thank you | you helped a lot thank you | thanks
ask_whatspossible	what can I do with this bot | can you do anything else
bot.age	how old are you? | what's your age?
bot.bad_joke	what a bad joke | ok that was a bad one
bot.be_clever	be smarter | you need to improve | you must learn
bot.gender	do you have a gender? | are you a boy? | are you a girl?
bot.hobbies	what are your hobbies? | how do you spend your time?
bot.joke	I would like to hear a joke | do you know jokes?
bot.knows_a_lot	you're so clever | you know a lot | very clever
bot.languages	do you speak German? | do you speak any other language?
bot.location	Where do you live? | what's your origin?
bot.name	how do people call you? | what's your name?
bot.real	are you a real person? | are you alive and real?
bot.stupid	you are useless | you're terrible | you are stupid
confirm.cancel	cancel | i want to cancel | forget
confirm.no	no | I don't think so | of course not
confirm.no+user.location.not_likes	no i don't like it there | not much, I would rather live in the city
confirm.yes	yeah | yes | alright | okay | I guess
confirm.yes+user.location.likes	yes i love to live there. | I absolutely do | it is nice
emotions.ha_ha	haha | lol | haha funny | LMAO
greet.goodbye	bye | see you | i have to go | goodbye
greet.hello	hello | hi there | hey | good morning
greet.how_are_you	how are you? | are you okay? | how do you do?
how_does_it_work	how can I learn vocabulary? | how should I ask you
learning.exit	exit | i want to stop | i have enough
learning.hint	can you give me a hint? | please some help
learning.intent_general	I want to learn english | i want to learn new stuff
learning.teach_vocabulary	i would like to learn new words | teach me some vocabulary
nice_to_meet_you	nice to meet you | it's lovely meeting you
out_of_scope	do you want to marry me? | i dont like bots | are you vegan
user_mood.bad	I am feeling bad | I am not feeling good | not very good
user_mood.good	I'm feeling great | I'm good | I am fine | Fine thanks

user_mood.good+greet.how_are_you	I am doing great and you? | Good and you? | I am good, how about you?
user.age	I'm 26 years old | 19 years | I turned 20 a few weeks ago
user.degree_program	I study Computer Science | I'm a medicine student | I'm studying psychology
user.degree_program.semester	i'm about to finish, in my last semester | 10th semester | ...
user.hobbies	in my spare time i like to go out | i like to do yoga | i like playing football
user.is_student	yes I'm a student | yes I'm a master student | yes I'm a bachelor student
user.location	I live in Berlin | in Paris | I come from Vienna
user.location+bot.location	I'm from Prague and you? | in Honduras and you
user.name	I am Lisa | My name is Christina | I'm Daniel
user.name+bot.name	My name is Leo and yours? | I am Lucy and who are you?
user.profession	i am a judge | I'm a professor | i'm working as a waiter | I am an IT consultant
user.wants_to_chat	I want to talk to you | let's talk | Come on, talk to me
what_time_is_it	what is the time | is it early? | do you know the time
word.meaning	tell me what blueberry means | what does enemy mean?

Tabelle D.1: Intents des EnglishBot

Entitäten

Entität	Beschreibung
degree_program	Studienfach Nutzer*in
location	Wohnort Nutzer*in
name	Name Nutzer*in
word	Wort für Wörterbuchanfrage

Tabelle D.2: Entitäten des EnglishBot

Actions

Die Anwendung weist insgesamt 79 Actions auf. In der rechten Spalte der Tabelle D.3 wird die jeweilige Action beschrieben oder - im Falle einer einfachen Nachricht - werden Antwortbeispiele aufgeführt. Die drei weiterführenden Punkte signalisieren, dass die Liste unvollständig ist. Der komplette Umfang an Antwortmöglichkeiten ist im Quellcode in der Datei `domain.yml` zu finden.

Action	Beschreibung oder Utterance Beispiel
action_ask_question	Stellen einer zufälligen Frage im Rahmen des Vokabelquiz

action_bot_questions	Now I asked you so many things - do you also have questions about me?
action_dictionary_meaning	Einholen der Definition eines Wortes
action_end_random_conversation	Beenden einer Konversation über zufälliges Thema
action_greet_user	Begrüßung
action_how_to	Erklärung, wie Chatbot zu benutzen ist und Funktionen aufgerufen werden
action_joke	Erzählen eines Witzes
action_react_to_degree_program	Reaktion auf Studienfach
action_save_user_location	Speichern des Wohnorts
action_save_user_name	Speichern des Namens
action_start_random_conversation	Beginn einer Konversation über zufälliges Thema
action_tell_time	Mitteilung der aktuellen Uhrzeit
action_user_degree_program	Setzen des Studienfach-Slots
utter_appraisal.welcome	You're welcome. | My pleasure. | ...
utter_ask_current_location	I don't know that place. Where is it? | Sorry, I didn't get that. Where are you from?
utter_ask_current_user_name	I didn't get that. What is your name?
utter_ask_for_name	What's your name?
utter_ask_for_name_again	Now you know my name. Would you let me know yours?
utter_ask_user_location	Where are you from? | Where do you live?
utter_ask_vocabulary_topic	Which topic would you like to talk about? Currently I can only talk about university. Would you like to start? | ...
utter_bad_joke	I'm sorry, but I told you. Do you want to hear another one? | ...
utter_bot_help	How can I help you? I can tell you the meaning of a word, you can take a quiz to learn new vocabulary or we continue chatting. What would you like to do? | ...
utter_bot.age	I am quite young. I was created recently, but I don't know my exact age. How old are you? | ...
utter_bot.and_you	And you {current_user_name}? | What about you {current_user_name}? | ...
utter_bot.be_clever	I'm certainly trying to improve myself, but it's not always easy as a chatbot. | ...
utter_bot.gender	Gender doesn't really apply to me as a chatbot. I guess I have no gender. | ...
utter_bot.hobbies	In my free time I surf the internet a lot. My favourite website is Wikipedia. | ...
utter_bot.languages	English is the only language I speak. | I only speak English | ...
utter_bot.location	I'm a chatbot, I live in the digital world. | ...
utter_bot.mood.good	I am doing great, thanks. | ...
utter_bot.name	My name is EnglishBot! | ...
utter_bot.questions	I asked you so many questions. Now it is your turn. Do you also have questions about me? | ...
utter_bot.real	I'm not a real person, but I certainly exist. | ...

utter_bot.stupid	I'm sorry. I can be trained to be more useful. My developer will keep training me. | ...
utter_bot.thanks_for_compliment	Thank you, I try my best to learn as much as I can. | ...
utter_cannot_help	I am afraid I can't help you with that.
utter_confirmation.no	Okay. | Alright. | ...
utter_confirmation.yes	Great! | Good. | ...
utter_confirmation.yes_questions	Great! Go ahead | Alright! I'm all ears.
utter_continue_chatting	We can just continue talking. What do you think? | ...
utter_continue_chatting_question	Would you like to continue chatting with me? | ...
utter_default	Sorry, I can't understand. Try to say it again. | ...
utter_emotions.ha_ha	Glad you think I'm funny. | ...
utter_end_chat	If you need anything, just let me know. I'm here 24/7 | ...
utter_explain_whatspossible	I can help you learn English. I can tell you the meaning of a word or you can learn some vocabulary.
utter_greet.ask_how_are_you	How are you? | How's it going? | ...
utter_greet.goodbye	Bye. | Bye {current_user_name}! | ...
utter_greet.hello	Hi there! | Hi! | Hey! | Hello!
utter_greet.hello.name	Hi there, {current_user_name}! | ...
utter_joke	I know jokes, but be prepared for bad ones | ...
utter_lets_start_vocabulary_quiz	Alright, let's get started! If you want to stop, type EXIT. If you need help, ask me for a hint | ...
utter_nice_to_meet_you	Nice to meet you {current_user_name}! | ...
utter_nice_to_meet_you_too	It's nice meeting you too! | ...
utter_show_interest	Sounds very interesting. | Pretty cool. | ...
utter_tell_me_word	Which word would you like to know? | ...
utter_user_what_now	What do you want to do now? I really enjoy chatting with you {current_user_name} | ...
utter_user.age	The best age to learn a language | The perfect age to learn a language
utter_user.hobbies	What do you like to do in your free time? | ...
utter_user.interesting	Sounds interesting! In which semester or year are you? | ...
utter_user.interesting_degree_program	{user_degree_program} sounds interesting! In which semester or year are you? | ...
utter_user.learning_exit	That was great. You got {vocabulary_counter} words correct. Hope to teach you some new vocabulary soon. | ...
utter_user.learning_exit_no_correct	Oh, that's a pity that you already want to stop. | ...
utter_user.learning_exit_one_correct	Good job! You got {vocabulary_counter} word correct! | ...
utter_user.location_again	You already told me you live in {current_location}, {current_user_name}. Did you move?
utter_user.location_go_there	Okay, so I have to go there one day. | ...
utter_user.location_good	Really, {current_location}? Do you like it there? | ...
utter_user.location_unknown	I don't know that place, where is it? | ...
utter_user.mood.happy	Great! Glad to hear that. | Happy to hear that. | ...

utter_user.mood.unhappy	I am sorry to hear that. How about learning some new words to cheer you up? | ...
utter_user.name_again	You already told me your name, {current_user_name}.
utter_user.new_location	Where do you live now?
utter_user.plans_after_studies	Cool, do you already know what you want to do when you finish your studies? | ...
utter_user.profession	So what do you do for a living? | ...
utter_user.studying	Are you a student? | ...
utter_user.studying.ask_subject	What are you studying? | ...
utter_user.wants_to_learn_english	Great, what would you like to learn? I can tell you the meaning of a word, you can take a quiz to learn new vocabulary or we continue chatting. Tell me what you prefer.
utter_vocabulary_quiz_correct	Great, that was correct! | Correct | ...
utter_vocabulary_quiz_correct_answer	The correct answer is {vocabulary_correct_answer} | ...
utter_vocabulary_quiz_not_correct	Sorry your last answer was not correct. | ...

Tabelle D.3: Actions des EnglishBot

Slots

Slot	Beispiele
chat_topic	Aktuelles Chatthema
current_location	Wohnort Nutzer*in
current_user_name	Name Nutzer*in
location	Extrahierte Entität Wohnort Nutzer*in (Zwischenspeicher)
name	Extrahierte Entität Name Nutzer*in (Zwischenspeicher)
user_degree_program	Studienfach Nutzer*in
vocabulary_correct_answer	Richtige Antwort für Frage im Vokabelquiz
vocabulary_counter	Zähler für korrekte Antworten in Quiz
vocabulary_quiz_exit	Vokabelquiz Form-Slot zum Beenden des Vokabelquiz
vocabulary_topic	Aktuelles Vokabelquiz-Thema
vocabulary_user_answer	Gegebene Antwort Nutzer*in
word	Wort für Wörterbuchanfrage

Tabelle D.4: Slots des EnglishBot

Forms

Form	Beispiele
dictionary_form	Form für Wörterbuchanfrage
introduction_form	Form für Einführungsdialog
vocabulary_quiz_form	Form für Vokabelquiz

Tabelle D.5: Forms des EnglishBot