Generierung von domänenspezifischen Trainingsdaten für zielorientierte Chatbots

Michaela Pfleger

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2018

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 25. September 2018

Michaela Pfleger

Inhaltsverzeichnis

Kurzfassung

Chatbots ermöglichen Nutzern die Interaktion mit Maschinen mittels natürlicher Sprache. Eingehende Anfragen werden vom Chatbot in Einzelteile zerlegt, analysiert und bestehenden Intents zugeordnet. Zur Zuordnung dieser Nutzeranfragen werden Trainingsdaten benötigt. Das Erstellen dieser Datensätze ist für Chatbot-Entwickler mit hohem Zeitaufwand verbunden.

Die benötigten Trainingsdaten werden von verschiedenen Faktoren, wie gewünschte Domäne, Zielgruppe, Sprache, vorhandenen Datenstrukturen sowie Zielen des Chatbots beeinflusst. Oftmals wird eine Vielzahl an Daten, wie Gesprächsprotokolle und Supportanfragen benötigt, um daraus passende Trainingsdaten entwickeln zu können. Diese Daten sind allerdings nicht immer vorhanden oder liegen nur in unstrukturierter Weise vor.

In dieser Arbeit wird eine einfache Methode zur semiautomatischen Generierung von domänenspezifischen Trainingsdaten für zielorientierte Chatbots präsentiert. Dabei wird lediglich die strukturierte Darstellung der gewünschten Domäne benötigt. Hierzu kann auf bestehende Auszeichnungen zurückgegriffen, oder Beispielstrukturen von Schema.org verwendet werden. Anhand dieser Struktur der Domäne und deren Eigenschaften werden mittels Slot-Filling Trainingsdaten generiert.

Durch die präsentierte Methode kann im Entwicklungsprozess des Chatbots eine erhebliche Zeitersparnis gegenüber der manuellen Erstellung von Trainingsdaten erreicht werden. Dass sich die generierten Trainingsdaten auch im realen Einsatz in einem Chatbot bewähren können, wurde in der durchgeführten Benutzerstudie bestätigt. Durch die präsentierten Verbesserungsmöglichkeiten kann dieser Wert weiter gesteigert werden.

Abstract

Chatbots allow users to interact with machines via natural language. Incoming user requests are broken down to their individual parts, analyzed and mapped to existing intents of the chatbot. To be able to map these user requests suitable training data is required. Creating such datasets is a challenging task for chatbot developers.

The required training data is influenced by various factors, such as desired domain, target group, language, existing data structures and goals of the chatbot. Often a large amount of data, such as call logs and support requests, is required to develop suitable training data. However, this data is not always available or is only available in an unstructured form.

This paper presents a simple method for generating domain-specific training data for goal-based chatbots in a semi-automatical way. For this purpose, existing structures of the website or available example structures from Schema.org can be used. Based on this structure of the domain and its properties the required training data is generated by using slot filling.

When using the presented method, the development process of a chatbot can be considerably reduced compared to the manual creation of training data. The conducted user study confirmed that these generated training data can also assert themselves against other methods in real use in a chatbot. This value can be further increased by the presented possibilities to improve the underlying method.

Kapitel 1

Einleitung

Mit zunehmender Verbreitung von Messaging-Plattformen steigt auch die Beliebtheit in der Entwicklung und Verwendung von Chatbots enorm an. Chatbots ermöglichen die Interaktion zwischen Mensch und Maschine mittels natürlicher Sprache. Dazu werden eingehende Nutzeranfragen in ihre Einzelheiten zerlegt und nach definierten Mustern durchsucht. Hierbei werden unterschiedliche Arten von künstlicher Intelligenz angewendet.

Mittlerweile wird eine Vielzahl von alltäglichen Aufgaben, wie Reisebuchungen, Essensbestellungen und Tischreservierungen von Chatbots übernommen. Dem Anwender soll geholfen werden, Zeit zu sparen. Doch auch für Unternehmen bieten sich durch die Verwendung von Chatbots neue Möglichkeiten in der Kundenkommunikation. Chatbots können rund um die Uhr auf die Anfragen der Kunden reagieren und somit schnell Antworten liefern. Dadurch sparen Unternehmen nicht nur Zeit ein, sondern auch Kosten für Support-Mitarbeiter.

Durch die technische Entwicklung in den letzten Jahren konnte die Funktionalität von Chatbots verbessert werden. Außerdem profitieren Chatbots vom regen Interesse von führenden Software- und Online-Unternehmen. Nachdem die neue Bot-Entwicklerplattform für den Facebook-Messenger im April 2016 auf der F8-Konferenz vorgestellt wurde, stieg die Anzahl der dort verfügbaren Chatbots von 11.000 auf über 100.000 an [39, S. 3]. Auch weitere einflussreiche Unternehmen wie Microsoft und Google stellen bereits eigene Bot-Frameworks zur Verfügung. Mittlerweile können einfache Chatbots ohne technische Kenntnisse in wenigen Minuten erstellt und veröffentlicht werden.

Laut einer Umfrage von LivePerson mit internationalen Teilnehmern sehen 38 % der Befragten der Verwendung von Chatbots positiv entgegen, 51 % bewerten ihre Wahrnehmung als neutral [46, S. 3]. Für die Zukunft wird die Ablöse klassischer Apps und Webseiten durch Chatbots erwartet [5].

Trotz des großen Interesses und der stetigen technischen Weiterentwicklung, erschweren einige Faktoren die Entwicklung von Chatbots. Eines dieser Probleme ist die Notwendigkeit, geeignete Trainingsdatensätze für zielorientierte Chatbots zu erstellen, um so die Bedürfnisse der Nutzer in deren Anfragen richtig interpretieren zu können.

1.1 Zielsetzung

Im Rahmen dieser Masterarbeit soll die Entwicklung von zielorientierten Chatbots genauer betrachtet werden. Der Fokus liegt auf den zu erstellenden Trainingsdaten, die für die Zuordnung von Nutzeranfragen an den Chatbot benötigt werden.

Als praktisches Beispiel wird ein Chatbot zum Suchen und Buchen von Unterkünften entwickelt. Dieser Chatbot soll zeigen, wie zeitaufwendig die manuelle Erstellung von Trainingsdaten ist. Anhand dieser Erfahrungen wird eine Methode vorgestellt und implementiert, bei der entsprechende Trainingsdaten semiautomatisch generiert werden können.

Ziel dieser Arbeit ist es, zu klären, ob ein Entwickler durch die definierte Methode beim Entwickeln von geeigneten Trainingsdaten für zielorientierte Chatbots unterstützt werden kann. Dabei wird auch untersucht, ob dieser Prozess so weit verallgemeinert werden kann, dass er auf eine Vielzahl von zielorientierten Chatbots verschiedener Domänen anwendbar ist.

1.2 Aufbau der Arbeit

In Kapitel 2 dieser Arbeit werden die technischen Grundlagen für das weitere Verständnis der Arbeit erläutert. Dabei werden wichtige Begriffe im Bereich der künstlichen Intelligenz geklärt und auf die Geschichte und Funktionsweise von Chatbots, sowie deren Einsatzgebiete und Limitationen eingegangen. Danach wird in Kapitel 3 der aktuelle Stand der Technik in der Entwicklung von Chatbots genauer betrachtet. Außerdem werden bestehende Ansätze zur Erstellung von Trainingsdaten vorgestellt und deren Vorteile aufgezeigt.

Die Relevanz von Trainingsdaten und deren problematische Erstellung werden in Kapitel 4 erläutert. Anschließend wird eine Methode zur semiautomatischen Generierung von domänenspezifischen Trainingsdaten für zielorientierte Chatbots beschrieben. Diese kann für alle benötigten Domänen verwendet werden.

Kapitel 5 soll Einblicke in die Implementierung des Chatbots sowie des Trainingsdaten-Generators geben. Um die Relevanz des entwickelten Generators zu testen, wird eine zweiteilige Benutzerstudie durchgeführt. Die Ergebnisse dieser Studie werden in Kapitel 6 dieser Arbeit präsentiert und interpretiert. Das letzte Kapitel bildet mit einer kompakten Zusammenfassung von Chatbots und deren Zukunft sowie einem Fazit des entwickelten Trainingsdaten-Generators den Abschluss dieser Arbeit.

Kapitel 2

Technische Grundlagen

Dieses Kapitel beschäftigt sich mit den Grundlagen, die für das technische Verständnis von Chatbots bedeutend sind. Einerseits werden Funktionsweise, Geschichte und Einsatzgebiete von Chatbots näher beleuchtet, andererseits wird auch auf die Relevanz von künstlicher Intelligenz eingegangen.

2.1 Künstliche Intelligenz

Das Interesse an künstlicher Intelligenz war bereits im Jahr 1956 enorm groß. John McCarthy wählte für seine Konferenz zu diesem Thema den Titel *Artifcial Intelligence* [14]. Selbst unter den KI-Forschern gibt es keine einheitliche Definition von künstlicher Intelligenz. Die Grundaussage ist jedoch bei allen Definitionen ähnlich: Künstliche Intelligenz versucht, menschenähnliche Intelligenz nachzubilden und dieses Verhalten dem Computer zu erlernen [8, S. 231ff]. Der große Durchbruch auf diesem Gebiet blieb jedoch lange Zeit aus. Erst in den letzten Jahren durchlebte die künstliche Intelligenz eine enorme Entwicklung. Durch die gestiegene Rechenleistung konnten die komplexen Methoden parallelisiert und verstärkt werden. Dadurch ist auch die Anwendbarkeit für Unternehmen in der Praxis gelungen [7, S. 16ff].

Das vielschichtige Gebiet der künstlichen Intelligenz umfasst verschiedene Ansätze und Unterdisziplinen. In dieser Arbeit werden nur die für die Entwicklung von Chatbots relevanten Disziplinen exemplarisch behandelt.

2.1.1 Turing Test

Bereits in den 1950er Jahren stellte Alan Turing, einer der ersten Forscher auf dem Gebiet der Computerintelligenz, einen Versuch vor, der die Intelligenz von Maschinen im Vergleich zu Menschen untersucht. Dazu kommunizieren Testpersonen in zwei kurzen textuellen Unterhaltungen mit einem Menschen sowie einer Maschine. Ist diese Testperson anschließend nicht in der Lage, den menschlichen Gesprächspartner zu identifizieren, kann der Maschine eine menschenähnliche Intelligenz zugeschrieben werden [18]. Dieser Versuch wird heute als Turing-Test bezeichnet und findet seit 1991 jährlich im Rahmen eines Wettbewerbes statt. Die Gewinner des Wettbewerbes werden mit dem Loebner-Preis ausgezeichnet. Im Jahr 2014 konnte eine russische Software erstmals den Turing-Test bestehen [7, S. 16ff].

2.1.2 Machine Learning

Machine Learning beschäftigt sich mit der technischen Generierung von Wissen aus Erfahrung und bietet vielfältige Anwendungsmöglichkeiten und hohes Grundlagenforschungspotential. Ein Schach-Computerprogramm kann beispielsweise seine Leistung im Spiel verbessern, indem es möglichst viele Partien spielt, diese analysiert und daraus Erfahrungen sammelt. Außerdem wurde Machine Learning bereits früh zum Herausfiltern von Spam-E-Mails verwendet, denn mit jeder neuen E-Mail konnte der Algorithmus dazulernen und seine Leistung eigenständig verbessern. Machine Learning definiert verschiedene Arten des Lernens, die sich teilweise maßgeblich voneinander unterscheiden. Diese sind überwachtes Lernen (Supervised Learning), nicht überwachtes Lernen (Unsupervised Learning) und verstärkendes Lernen (Reinforcement Learning) [7, S. 37ff].

2.1.3 Natural Language Processing

Natural Language Processing (NLP) ist eine Schnittstelle zwischen Informatik und Sprachwissenschaften und befasst sich mit der Analyse und Interpretation menschlicher Sprache durch Computersysteme. NLP ist ein bedeutender Bestandteil von Chatbots, da aus dem gesprochenen oder geschriebenen Input des Nutzers eine Intention abgeleitet wird [12, S. 232ff]. In den letzten Jahren konnte NLP enorm weiterentwickelt und verbessert werden. Dennoch stellt die menschliche Sprache Maschinen heutzutage immer noch vor große Herausforderungen, da die kommunizierte Bedeutung des Gesprochenen oft in der Semantik liegt. Die Semantik verschiedener Sprachen ist nicht immer einheitlich und lässt sich somit nicht problemlos für Maschinen formalisieren. Auch die Vielzahl an Synonymen und Mehrdeutigkeiten der menschlichen Sprache lässt NLP oft an seine Grenzen kommen. Deshalb wird NLP als sogenanntes „hartes Problem“ der Informatik eingestuft. Ein aktuelles Beispiel in diesem Zusammenhang ist die Spracherkennung. Diese beschäftigt sich mit der Transkription menschlicher Sprache. Vor allem für das Endkundengeschäft ist diese Thematik enorm wichtig und lukrativ, denn beliebte Geräte, wie Amazon Echo, werden ausschließlich per Spracheingaben gesteuert [7, S. 31ff].

2.1.4 Natural Language Understanding

NLU steht für Natural Language Understanding und ist ein Teilbereich von NLP. Die Bedeutung einer Aussage soll im Detail analysiert und interpretiert werden. Dabei werden auch semantische und kontextsensitive Techniken verwendet. Somit können komplexe Satzfolgen interpretiert werden, ohne die Verwendung der relevanten Schlüsselwörter in jedem einzelnen dieser Sätze [36]. NLU wird häufig bei sprachgesteuerten Assistenten, Social Media Analysen oder automatischen Inhaltszusammenfassungen eingesetzt.

2.2 Chatbots

Chatbots ermöglichen dem Menschen die Interaktion mit Maschinen mittels natürlicher Sprache. Diese Interaktion kann sowohl per Tastatur, als auch in gesprochener Form stattfinden. Ihr Name setzt sich aus den englischen Wörtern *chat* und *bot* zusammen,

also ein Roboter, mit dem sich unterhalten werden kann [3, S. 21ff]. Um diese Unterhaltungen zu ermöglichen, wird künstliche Intelligenz in verschiedenen Formen verwendet. Chatbots sind vorwiegend in Messaging-Plattformen, wie Facebook Messenger, Slack oder Telegram intergiert. Sie können aber auch per E-Mail, direkt auf Webseiten oder durch persönliche intelligente Assistenten bedient werden. Nachdem sie einmal initiiert wurden, können die Bots ihre Dienste weitgehend selbstständig und ohne menschliche Hilfestellungen erbringen. Deshalb sind Chatbots besonders gut für wiederkehrende Aufgaben geeignet [12, S. 232ff].

Grundsätzlich lassen sich die meisten Chatbots in zwei unterschiedliche Kategorien einteilen. Soll der Chatbot eine bestimmte Absicht erfüllen, wird er als zielbasiert bezeichnet. Diese Absicht kann zum Beispiel das Buchen eines Urlaubes sein. Erfüllt der Chatbot allerdings keinen besonderen Nutzen, sondern dient lediglich zum Gespräch mit dem Nutzer, wird er als dialogorientiert bezeichnet.

Chatbots können personifiziert sein, das bedeutet, sie werden dem Nutzer als Person vorgestellt. Diese Person kann einen Menschen, ein Tier oder ein Fabelwesen darstellen [3, S. 21ff] und soll eine persönliche Bindung zum Nutzer aufgebauen. Viele Unternehmen haben ihre Chatbots mit prägnanten Charakteren ausgestattet, wie zum Beispiel der deutsche Luftfahrtkonzern Lufthansa. Eine zweisprachige Assistentin namens *Mildred*[1] informiert die Nutzer per Facebook Messenger über Flugpreise [41, S. 16].

2.2.1 Geschichte

Derzeit sind Chatbots in aller Munde, und das obwohl die Idee dazu schon seit über 50 Jahren existiert. Der erste Chatbot wurde eingeführt, noch bevor der erste PC entwickelt wurde. Eliza wurde 1966 am MIT Artificial Intelligence Laboratory von Professor Joseph Weizenbaum entwickelt und stellt eine Psychotherapeutin dar [20]. Eliza ermöglichte die natürlichsprachliche Konversation zwischen Mensch und Computer. Die grobe Vorgehensweise des Programms war relativ einfach. Während der Konversation mit Eliza tippte der Benutzer eine oder mehrere Aussagen ein. Diese Aussagen wurden von Eliza eingelesen und auf Schlüsselwörter überprüft. Wurde ein solches Schlüsselwort gefunden, wurde die Aussage anhand einer mit diesem Schlüsselwort verknüpften Regel transformiert und als Antwort ausgegeben. Wurde kein Schlüsselwort gefunden, wurde eine inhaltsfreie Bemerkung ausgegeben. Danach war wieder der Benutzer am Zug und konnte weitere Aussagen eintippen [20, S. 36].

Die Identifikation des wichtigsten Schlüsselwortes war eine der grundlegenden technischen Problemstellungen, mit denen sich Eliza beschäftigen musste. Schlüsselwörter wurden mit einer Gewichtung versehen. Bereits gefundene Schlüsselwörter konnten zugunsten anderer Wörter mit höherer Gewichtung überschrieben werden. Weitere Probleme waren die Erkennung eines minimalen Kontextes des gewählten Schlüsselwortes, und die Wahl geeigneter Transformationsregeln. Diese Mechanismen zur Transformation dienten dazu, die Aussagen zu zerlegen und nach bestimmten Spezifikationen wieder zusammenzusetzen. Dabei wurden beispielsweise alle persönlichen Instanzen ausgetauscht. Anhand von einzelnen Satzbausteinen und definierten Regeln wurde danach überprüft, wie Eliza antworten könnte. Durch die Wiederholung dieser Muster über den ganzen

[1] https://www.lufthansa.com/us/en/lufthansa_bot

Dialog hinweg, erweckte Eliza den Eindruck von Intelligenz [20, S. 37ff]. Jedoch kamen Menschen, die mit Eliza kommunizierten, schnell zu dem Entschluss, dass es sich dabei um eine Maschine handelte. Trotzdem erwies sich das System später als sehr bedeutsam für den Fortschritt von NLP [53] und gilt als einer der ersten Meilensteine in der Chatbot-Entwicklung [58].

Die Funktionsweise von Eliza mag trivial erscheinen, dennoch wird sie heute immer noch für die Erstellung von Chatbots verwendet. Nach dem Vorbild von Eliza wurde 1972 ein weiterer Chatbot von dem Psychiater Kenneth Colby entwickelt. Der Chatbot namens *Parry* simulierte dabei eine Person mit paranoider Schizophrenie [11, S. 2].

Im Jahr 1995 wurde A.L.I.C.E. oder ALICE, kurz für Artificial Linguistic Internet Computer Entity, von Richard Wallace entwickelt [19]. ALICE ist ebenfalls von Eliza inspiriert und verfügt über eine Wissensbasis, die sich aus etwa 41.000 Kategorien zusammensetzt. Jede Kategorie kombiniert eine Frage oder einen Reiz mit einer entsprechenden Antwort. Diese Paare werden als Muster oder Schablonen bezeichnet. Die Muster werden mithilfe von AIML, kurz für Artificial Intelligence Markup Language, als Baumstruktur gespeichert. AIML ist eine spezielle XML-Sprache und implementiert zusätzlich einen Musterspeicherungs- und Zuordnungs-Algorithmus [19, S. 4].

Die Art des Lernens wird bei ALICE als Supervised Learning bezeichnet, da der sogenannte Botmaster die Gespräche von ALICE überwacht und anschließend neue AIML-Inhalte erstellt, um die Antworten laufend zu verbessern [19, S. 3].

ALICE konnte bereits dreimal die Bronzemedaille des Loebner-Preises gewinnen. Die Bronzemedaille wird für das Programm vergeben, dass im jährlichen Wettbewerb die meisten menschlichen Juroren täuschen kann, sich also am menschenähnlichsten erweist [6, S. 282].

Im Jahr 2006 wurde Watson von IBM entwickelt, um Fragen in natürlicher Sprache zu beantworten. Die Maschine spricht und versteht acht verschiedene Sprachen und kann fast eine Milliarde Seiten pro Sekunde lesen. Dazu ist Watson mit 50 verschiedenen Technologien der künstlichen Intelligenz ausgestattet, von NLP bis hin zur Bildverarbeitung [17, S. 267]. Mittlerweile wird Watson dazu verwendet, Personen anhand ihrer Tweets zu analysieren und daraus zu lernen. Dies soll Unternehmen dabei helfen, Verbraucherverhalten vorherzusagen, um zu entscheiden, welche Produkte und Dienstleistungen sie zu welcher Zeit anbieten sollen [17, S. 258ff].

Der intelligente persönliche Assistent Siri wurde 2010 von Apple für iOS entwickelt. Siri steht für Speech Interpretation and Recognition Interface und wird als Grundstein für weitere intelligente persönliche Assistenten gesehen [53]. Das System ermöglicht den Nutzern Gespräche zu führen und dabei nützliche Informationen über das Wetter oder Nachrichten zu erhalten [11, S. 3]. 2014 stellte Amazon seinen intelligenten persönlichen Assistenten Alexa vor. Alexa wurde in Geräten wie Amazon Echo, Echo Dot und Echo Show eingebaut. Durch Sprachbefehle kann der Nutzer Alexa steuern und so unter anderem Informationen im Internet suchen, Produkte bestellen, Musik abspielen und sogar Smart-Home-Geräte im selben Netzwerk bedienen [53].

Was in den 1960er Jahren noch als Experiment und Phantasie angesehen wurde, hat sich über die Jahre zu einem festen Bestandteil des Alltags entwickelt. Die direkte Kommunikation in natürlicher Sprache mit einer Maschine wird laut Experten in Zukunft weiter zunehmen und so auch die klassischen Apps mit der Zeit ablösen [5, S. 40].

2.2.2 Funktionsweise

Heutzutage besteht ein Chatbot grundsätzlich aus einem Kanal, über den Input empfangen werden kann. Dies kann einerseits ein einfaches Inputfeld sein. Andererseits kann der Benutzer auch in gesprochener Form über ein Mikrofon mit dem Chatbot kommunizieren. Die Nachrichten, die über diesen Kanal eingegeben werden, werden mittels künstlicher Intelligenz auf eine Intention überprüft. Dazu wird der gesprochene oder geschriebene Satz mithilfe von NLP in seine Einzelteile zerlegt und nach Mustern analysiert. Wird eine Intention gefunden, stößt diese Intention eine zugehörige Aktion an. Diese Aktion ist in einer Wissensbasis hinterlegt und stellt das einfache Beantworten einer Frage oder das Ausführen einer bestimmten Aufgabe dar. Machine Learning kann dazu verwendet werden, dem Chatbot neue Vokabeln beizubringen. Somit versteht der Chatbot auch Abweichungen der definierten Trainingsdaten. Durch jede neue Interaktion wird der Chatbot intelligenter und entwickelt sich so ohne zusätzliches Eingreifen des Entwicklers selbstständig weiter [12, S. 233ff].

2.2.3 Einsatzgebiete

Chatbots werden bereits in vielen verschiedenen Branchen erfolgreich eingesetzt. Im Grunde lassen sich mit Chatbots Situationen am besten darstellen, bei denen einfache Nutzeranfragen beantwortet werden, die sich häufig wiederholen oder ähnlich sind [12, S. 238].

Oftmals werden Chatbots von Unternehmen dazu verwendet, dem Verbraucher Produkte und Dienstleistungen im Dialog zur Verfügung zu stellen. Diese E-Commerce-Chatbots dienen einerseits zur Produktsuche und bestenfalls zum Kauf, andererseits kann sich der Nutzer auch über Produkte und Services informieren. Ebenso kann der Customer-Support durch den Chatbot abgedeckt werden, beispielsweise das Abwickeln von Reklamationen und Rücksendungen [11, S. 99ff]. Aber auch zur Beratung von Kunden können Chatbots eingesetzt werden, zum Beispiel als persönlicher Styling-Assistent [52].

Chatbots werden inzwischen vermehrt im Bereich E-Government eingesetzt, aber auch im Gesundheitswesen wird die Verwendung immer beliebter. Mittlerweile wurde ein Chatbot entwickelt, der automatisiert Gesundheitsfragen von Patienten beantwortet. Der virtuelle Arzt verwickelt den Patienten in ein Gespräch, bei dem alle wichtigen Informationen zur Diagnosefindung abgefragt werden. Dem Patienten können je nach Diagnose benötigte Rezepte oder ärztliche Atteste direkt aufs Smartphone gesendet werden [13].

Auch im Banken- und Finanzwesen haben Chatbots bereits Einzug gehalten. Das Auffinden einer Bankfiliale in der Nähe und das Überprüfen des derzeitigen Guthabens sind ebenso möglich, wie das Beauftragen einer Überweisung auf ein anderes Konto[2]. Weitere Anwendungsfälle sind die Sperrung von gestohlenen Bankomat- oder Kreditkarten sowie das Anfordern einer neuen Karte [11, S. 100]. Auch der Kauf und Verkauf von Aktien und das Einholen von Informationen über aktuelle Kurse können per Dialog erledigt werden.

Weitere Anwendungsfälle für den Einsatz von Chatbots bietet die Tourismusbran-

[2] https://www.topbots.com/project/western-union-facebook-messenger-bot-review/

che. Dies sind unter anderem die Suche nach Restaurants und das Tätigen von Tischreservierungen[3] sowie Lieferdienste[4], bei denen Nutzer ihr Essen per Chatbot bestellen können. Außerdem werden immer mehr Reisebuchungs-Bots veröffentlicht. Dabei setzen Online-Reiseplattformen verstärkt auf Reiseagenten, die bei der Suche nach Flügen, Unterkünften oder Pauschalurlauben helfen sollen. Einer der wichtigsten Faktoren beim Buchen einer Reise oder eines Fluges ist für potenzielle Kunden oftmals der Preis. Chatbots können helfen, die Preise zu verfolgen, und bei der Unterschreitung eines bestimmten Wertes Benachrichtigungen an den Nutzer zu schicken und ihn so zum Buchen zu animieren. Online-Buchungsplattformen bieten oft nur die Möglichkeit, nach bestimmten Reisezielen zu suchen. Ist der potenzielle Kunde aber unschlüssig, wo seine nächste Reise hingehen soll, bieten Chatbots hier eine weitere Möglichkeit: Sie können Nutzer zu verschiedenen Reisen inspirieren. Dabei werden bestimmte Präferenzen, wie das gewünschte Klima angegeben, oder die Filterung ganz dem Bot überlassen [41, S. 14].

Während einer Reise kann der Chatbot den Kunden über Sehenswürdigkeiten und interessante Orte in der näheren Umgebung der Urlaubsdestination informieren, für ihn beim Zimmerservice bestellen oder Massagen und Stadtführungen buchen[5]. Weitere Anwendungsfälle sind das Umrechnen von Währungen, das Abfragen von Wettervorhersagen[6] oder das Bestellen von Taxis und Mitfahrgelegenheiten[7]. Auch für Fluglinien ist diese Art der Kommunikation interessant. Durch den Chatbot können wiederkehrende Fragen zum Buchen und Einchecken rasch beantwortet werden. Außerdem können zusätzliche Informationen, wie der Standort eines Fluges oder dessen Ankunftszeit für die Nutzer bereitgestellt werden[8].

Natürlich werden Chatbots auch zur Belustigung verwendet. So gibt es Bots, die anhand von vorhandenen Zutaten Cocktailrezepte vorschlagen[9], Nutzern beim Finden von neuen Freunden helfen[10] oder das Personalisieren von Produkten[11] ermöglichen.

2.2.4 Chancen und Limitationen

Chatbots bieten viele Vorteile gegenüber dem persönlichen Gespräch mit einem Mitarbeiter oder dem Besuch einer Filiale. Sie sind enorm effizient und können mehrere Anfragen simultan abarbeiten und somit einer größeren Anzahl an Kunden helfen. Außerdem ist der Chatbot immer erreichbar, da keine Öffnungszeiten eingehalten werden müssen. Dadurch werden die Anfragen und Bedürfnisse der Kunden zu jeder Zeit beantwortet, was sich positiv auf die Kundenbindung auswirken kann. Ebenso trägt die dargestellte Persönlichkeit der meisten Chatbots dazu bei. Dafür muss der Chatbot allerdings auch in der Lage sein, per Small-Talk mit dem Nutzer kommunizieren zu können. Ein Bot behandelt alle Kunden gleich, verspürt keinen Stress oder Druck, weshalb er immer

[3] https://chatobook.com/
[4] https://www.topbots.com/project/subway-food-ordering-bot-review/
[5] https://hijiffy.com/
[6] https://www.topbots.com/project/weather-channel-kik-bot-guide/
[7] https://www.topbots.com/project/uber-facebook-messenger-bot-guide/
[8] https://www.topbots.com/project/klm-facebook-messenger-chat-bot-guide/
[9] https://www.topbots.com/project/diageo-united-spirits-group-simi-chatbot-bartender/
[10] https://www.kik.com/bots/makefriends/
[11] https://www.topbots.com/project/nike-sneaker-style-facebook-bot-review/

freundlich mit den Nutzern interagiert. Obwohl Chatbots mittlerweile schon in unserem Alltag integriert sind, weckt die Verwendung bei vielen Nutzern immer noch Neugier und Entdeckergeist. Dies kann helfen, die ersten Berührungsängste zu überwinden [41, S. 10].

Bei umfangreichen Webseiten kann ein Chatbot den Nutzer dabei unterstützen, die gewünschten Antworten im Dialog schneller zu finden, als beim Durchforsten der Website nach einem bestimmten Begriff. Denn entdecken die Besucher die gewünschten Informationen nicht, oder erhalten sie zu viele Suchergebnisse auf ihre Anfrage, verlassen sie die Website oftmals in Richtung Konkurrenz [3, S. 32ff]. Ein weiterer relevanter Punkt ist die Gewinnung von Kundeninformationen, denn traditionelle Websites ermöglichen den Betreibern oft nicht, Auskunft über die Kundenzufriedenheit zu erhalten. Aus häufig aufgerufenen Dokumenten und der durchschnittlichen Verweildauer ist nicht zu erkennen, ob der Besucher die gewünschten Informationen erhalten hat und damit in vollem Ausmaß zufrieden ist. Auch hier bieten Chatbots einige Vorteile: Da die Konversationen mit dem Nutzer dokumentiert werden, kann einfach überprüft werden, ob der explizit geäußerte Informationsbedarf auch tatsächlich gestillt wurde. So können einerseits relevante Themen präsenter auf der Website platziert, und andererseits fehlende Bereiche ergänzt werden [3, S. 38].

Dennoch gibt es bei der Verwendung, und auch bei der Entwicklung von Chatbots einige Einschränkungen und Limitationen zu beachten. Bei textuellen Chatbots, bei denen der User nicht spricht, sondern alle Anfragen eintippen muss, entsteht ein höherer Aufwand. Dies muss bei der Entwicklung berücksichtigt werden, um Interaktionen auch mit kurzen Befehlen oder Stichwörtern zu ermöglichen. Die Leistung beziehungsweise die Fähigkeit des Chatbots muss dem Nutzer klar kommuniziert werden, um Frustration bei fehlerhaften Ergebnissen oder Missverständnissen zu verhindern [41, S. 10]. Im schlimmsten Falle kann die Verwendung eines mangelhaft entwickelten Chatbots die Kunden enttäuschen und somit den Ruf des Unternehmens oder die Marke an sich schädigen [7, S. 85]. Außerdem sollte dem Nutzer deutlich gemacht werden, dass es sich bei seinem Gesprächspartner um einen Bot, und nicht um eine reale Person handelt. Um dies zu unterstreichen, sollte zu jeder Zeit des Gespräches die Möglichkeit zur Kontaktaufnahme mit einem menschlichen Service-Mitarbeiter bestehen.

Selbst wenn der Chatbot bereits produktiv eingesetzt wird, benötigt er ständige Überwachung und Optimierung. Aus den aufgezeichneten Unterhaltungen können Anzeichen für Usability-Probleme des Chatbots erkannt und dementsprechend behoben werden. Auch aus den abgebrochenen Konversationen können Rückschlüsse über die Dialogführung gezogen, und der Chatbot an diesen Stellen verbessert werden [21, S. 147ff].

Ebenso ist die Auswahl der Kommunikationsplattform für den Erfolg des Chatbots entscheidend. Verwendet die Zielgruppe die gewählte Plattform nicht, wird sich der Chatbot nicht durchsetzen können und potenzielle Kunden möglicherweise zur Konkurrenz abwandern. Gleichermaßen sollte die Marke zur gewählten Plattform passen [21, S. 147ff], sensible Finanztransaktionen werden die Nutzer beispielsweise nicht über Facebook oder WhatsApp abwickeln wollen.

Kapitel 3

Stand der Technik

In den letzten Jahren ist das Interesse an Chatbots und ihrer Entwicklung enorm gestiegen. Sie haben sich zu einer bevorzugten Plattform für die Benutzerinteraktion entwickelt. Die Verwendung von natürlichsprachlichen Benutzeroberflächen widerspricht allerdings dem bisherigen Ansatz von Forschern und Fachleuten, die sich auf die stetige Verbesserung von grafischen Benutzeroberflächen konzentrierten. Anstatt auf Buttons zu klicken, zu scrollen und zu wischen, verwenden Chatbots gesprochene Wörter oder Textfolgen [5, S. 38].

3.1 Vergleichbare Chatbots

Wie bereits in Abschnitt 2.2 erwähnt, eignen sich Chatbots vor allem für wiederholende, ähnliche Tätigkeiten, wie das Suchen und anschließende Erwerben von Produkten oder Dienstleistungen. Ein Bereich, der auf diese Eigenschaften zutrifft ist die Tourismusbranche. Touristeninformationsbüros werden häufig mit ähnlichen Fragen konfrontiert. Um Zeit und Ressourcen zu sparen, können hier Chatbots eingesetzt werden. Einerseits zur Information für den Urlauber, andererseits als Buchungsmöglichkeit für potenzielle Kunden.

In der Tourismusbranche gibt es bereits erfolgreiche Ansätze, die Vorteile von Chatbots zu nutzen. KAYAK[1], ein US-amerikanisches Unternehmen, das eine Reisesuchmaschine für Flüge, Hotels, Mietwagen und Pauschalurlaube zur Verfügung stellt, veröffentlichte einen eigenen Reiseberater-Chatbot[2]. Die deutsche Version dieses Chatbots ist in Facebook Messenger integriert. Nutzern soll geholfen werden, Reisen zu suchen, diese zu buchen, zu planen und zu verwalten. Die englische Version des Chatbots kann auch in Slack verwendet werden. Dort hilft der Bot entweder einzelnen Personen oder Gruppen bei der Reiseplanung. Der Bot ist ebenso über Amazons Alexa erreichbar, wo auf Daten von KAYAK zugegriffen wird, um dem Nutzer Fragen zur Reiseplanung zu beantworten [60]. Nutzer können gezielt nach Destinationen suchen oder sich vom Bot inspirieren lassen. Der Chatbot bietet nicht nur die Möglichkeit, sich über Preise zu informieren, sondern auch Mietwagen zu buchen und sich Auskünfte über Freizeitaktivität am Urlaubsort einzuholen. Der Charakter des Chatbots steht im Hintergrund, er stellt

[1] https://www.kayak.com/

[2] https://www.topbots.com/project/kayak-bot-facebook-messenger-slack-alexa/

keine bestimmte Persönlichkeit dar, die Kommunikation ist direkt und sachlich [60].

Ein weiterer Chatbot im Tourismus-Bereich ist Olympia, eine digitale Assistentin vom Tourismusverband Olympiaregion Seefeld. Olympia hilft auf der offiziellen Website des Tourismusverbandes[3] den Besuchern bei der Suche nach Hotels und Events und beantwortet Wetterfragen. Olympia stellt eine eigene Persönlichkeit dar, sie wirkt freundlich und hilfsbereit. Dies wird auch durch den Gebrauch von Emojis unterstützt. Die Interaktion im Chatbot erfolgt großteils über Buttons und Slider. Dadurch wird der Nutzer in der Dialogführung zwar eingeschränkt, die benötigte Zeit zum Eintippen von Anfragen kann so allerdings verkürzt werden und die Konversation weicht seltener von der geplanten Unterhaltung ab. Außerdem werden immer wieder nützliche Links zu anderen Webseiten angeboten. Der Verlauf der Unterhaltung wird auf der Website gespeichert, um dem Nutzer die Möglichkeit zu geben, auf alte Nachrichten zugreifen zu können.

3.2 Tools zur Erstellung von Chatbots

Viele einflussreiche Unternehmen, wie Microsoft, Google und Facebook, haben bereits eigene Bot-Frameworks veröffentlicht, die den Nutzern das Entwickeln von Chatbots ermöglichen und erleichtern. Die einzelnen Frameworks unterscheiden sich zwar im Detail, erfordern aber alle ein gewisses Maß an Programmierkenntnissen. Im Gegensatz dazu existieren Bot-Plattformen, die als Online-Ökosysteme angesehen werden können, und für Anfänger und technisch nicht versierte Benutzer geeignet sind [48]. Bot-Frameworks bieten eine Reihe von vordefinierten Funktionen und Klassen, um die Entwicklung von Chatbots von Beginn an zu vereinfachen. Sie unterstützen oft den Ansatz *write once run everywhere*. Technisch gesehen ist dies zweifellos möglich, dennoch sollten individuelle Chatbots für jede gewünschte Messaging-Plattform erstellt werden, da diese unterschiedliche Bedienelemente und Interaktionen verwenden, mit denen die Nutzer bereits vertraut sind [48].

3.2.1 Chatfuel

Wie bereits erwähnt, können auch Personen ohne Programmierkenntnisse Chatbots mit Bot-Plattformen, wie zum Beispiel Chatfuel, erstellen. Diese Plattform ist solange kostenlos, bis der Chatbot 30.000 Benutzer erreicht. Chatfuel zielt hauptsächlich auf die Verwendung in Facebook Messenger ab, dennoch sind auch andere Integrationen möglich. Auf der offiziellen Website wird damit geworben, dass Nutzer einen voll funktionsfähigen Chatbot in nur sieben Minuten veröffentlichen können. Dies soll durch das einfache Hinzufügen und Anpassen von Inhalten im Sinne des *what you see is what you get*-Ansatzes möglich sein [24].

Bekannte Unternehmen wie Adidas[4], Reuters[5] und Netflix[6] nutzen Chatbots, die mit Chatfuel entwickelt wurden.

[3] https://www.seefeld.com/
[4] https://chatfuel.com/bot/adidasWomenUK
[5] https://chatfuel.com/bot/Reuters
[6] https://chatfuel.com/bot/netflixfrance

3.2.2 Microsoft Bot-Framework

Anfang 2016 präsentierte Microsoft seine Vision von Gesprächsplattformen. Künstliche Intelligenz und natürlichsprachliche Interaktion sollten es dem Benutzer ermöglichen, neue interaktive Systeme zu erleben [5, S. 40]. Microsoft stellt ein Bot-Framework zur Verfügung, das sowohl über natürliches Sprachverständnis, als auch über eine Integrationskomponente verfügt. Die Integrationskomponente kann mit Messaging-Plattformen wie Slack, Facebook Messenger, Skype und sogar mit SMS verknüpft werden. Microsoft bietet kognitive Services an, um beispielsweise Benutzer in Fotos zu erkennen, intelligente Empfehlungen auszusprechen oder Sprachen zu übersetzen [50].

Dixons Carphone[7], ein europäischer Elektronik- und Telekommunikationsanbieter und Dienstleister [49], nutzt die Dienste von Microsoft ebenso wie UPS[8], ein Paketdienst und Logistikunternehmen [51].

3.2.3 Facebook Bot-Engine

Facebook trat der Bot-Entwicklung im Jahr 2016 offiziell bei, als das Unternehmen seine eigene Bot-Engine auf Basis der Wit.ai-Technologie veröffentlichte. Wit.ai unterstützt Entwickler bei der Erstellung von Anwendungen und Geräten, die natürliche Sprache als Input verwenden, sowohl in gesprochener, als auch in schriftlicher Form. Die Facebook Bot-Engine nutzt Machine Learning. Entwickler befüllen das Framework mit Testdatensätzen in Form von Anweisungen, Fragen und Antworten, die Bot-Engine ist anschließend in der Lage, verschiedene Variationen dieser Beispieldaten zu verarbeiten, um die Intention oder das Ziel des Chabot-Benutzers zu extrahieren. Die entwickelten Bots können ausschließlich in Facebook Messenger integriert werden [48].

3.2.4 Dialogflow

Dialogflow ist ein webbasiertes Bot-Framework, das von Google unterstützt wird und früher unter dem Namen Api.ai bekannt war. Um die Intention der Nutzer zu verstehen, wird Machine Learning genutzt. Der Entwickler muss dafür Beispieldaten liefern, die der Benutzer während der Interaktion mit dem Chatbot verwenden könnte. Mit Domänenkenntnissen und natürlichem Sprachverständnis analysiert Dialogflow die Intention des Benutzers und hilft dem Entwickler, entsprechend darauf zu reagieren [28].

Dialogflow stellt eine umfangreiche Dokumentation und eine Vielzahl von Tutorials zur Verfügung. Das Framework erlaubt einen einfachen Einstieg in die Entwicklung von Chatbots, sowohl für versierte Programmierer, als auch für Anfänger. Dialogflow bietet die Integration in 14 verschiedene Plattformen, über die sogenannte *One-Click-Integration*, an. Dazu gehören unter anderem Facebook Messenger, Skype und Telegram, aber auch plattformübergreifende Entwicklung ist möglich. Endgeräte wie Smartphones, Wearables oder intelligente Assistenten können mit diversen unterstützten Software Development Kits angesteuert werden. Dialogflow bietet außerdem mehr als 20 Sprachen für die Entwicklung von Chatbots [28].

Das Konzept von Dialogflow besteht aus folgenden Bereichen:

[7] http://www.dixonscarphone.com/
[8] https://www.ups.com/

Entities

Entities, also Entitäten, werden verwendet, um Parameterwerte aus den natürlichsprachlichen Eingaben zu extrahieren. Alle relevanten Daten, die der Benutzer während der Interaktion mit dem Chatbot angibt, werden einer entsprechenden Entity zugeordnet. Dialogflow stellt eine Vielzahl an vordefinierten Entities, wie Datum, Ort oder Währungseinheit zur Verfügung. Der Entwickler kann zusätzlich beliebig viele eigene Entities festlegen und mit den gewünschten Daten befüllen [29].

Contexts

Contexts werden verwendet, um den aktuellen Kontext der Konversation darzustellen. Dies ist enorm wichtig, um zwischen allgemeinen und kontextabhängigen Intentionen unterscheiden zu können oder um Rückfragen zu handhaben. Bei der Entwicklung von Chatbots ist die korrekte Verwaltung des Kontextes von entscheidender Bedeutung. Der Entwickler kann für jeden Intent individuelle Input- und Output-Kontexte definieren und deren Lebensdauer festlegen. Grundsätzlich sind Kontexte an die User-Session gebunden. Output-Kontexte werden dazu verwendet, die jeweiligen Parameter und Daten des aktuellen Intents zu speichern. Input-Kontexte hingegen begrenzen Intents, sodass diese nur dann einer Nutzeranfrage zugeordnet werden können, wenn bestimmte Kontexte in der aktuellen Unterhaltung vorhanden sind [27].

Intents

Intents, die Intentionen bzw. Absichten des Benutzers, repräsentieren die Zuordnung zwischen dem, was die Benutzer sagen, und der entsprechenden Aktion, die von der Software ausgeführt werden soll. Das Intent-Interface besteht aus verschiedenen Bereichen: der Action, den Trainingsphrasen, der Response und dem Context. Intents können priorisiert werden, um ihnen mehr Gewicht zu verleihen. Sollte eine Eingabephrase des Nutzers mit mehreren Intents übereinstimmen, kann die gewählte Priorität den Ausschlag geben, welcher Intent schlussendlich ausgelöst wird. Zusätzlich zu den vom Entwickler definierten Intents, stellt Dialogflow auch noch sogenannte Fallback-Intents zur Verfügung. Diese werden automatisch ausgelöst, wenn eine Anfrage keinem anderen Intent zugeordnet werden kann. Sobald ein Intent vom Entwickler gespeichert wird, beginnt Dialogflow das Training des Chatbots mit den neu hinzugefügten Daten [31].

Actions

Actions legen fest, welche Schritte die Anwendung durchführt, sobald ein bestimmter Intent durch eine Benutzereingabe ausgelöst wurde. Diese Actions können Parameter besitzen, die über Entities beschrieben werden. Sowohl der Action-Name, als auch die benötigten Parameter werden vom Entwickler festgelegt. Mittels Slot-Filling können alle relevanten Informationen vom Benutzer abgefragt werden. Die Parameter können als optional oder zwingend erforderlich gekennzeichnet werden. Stellt der Benutzer die erforderlichen Daten in seiner Eingabephrase nicht zur Verfügung, werden diese einzeln von ihm abgefragt, indem eine vom Entwickler gewählte Aufforderungsphrase angezeigt wird. Somit kann sichergestellt werden, dass alle relevanten Daten gesammelt wurden, bevor der Intent eine bestimmte Aktion durchführt. Parameter können nicht nur einzelne Werte annehmen, sondern auch als Liste definiert werden. Dies ist ideal für Situationen, in denen die

Eingabe des Benutzers eine unbegrenzte Aufzählung von Werten innerhalb einer Entity darstellt [26].

Trainingsphrasen

Trainingsphrasen werden dazu verwendet, die Absichten des Benutzer zu erkennen. Um dies zu erreichen, muss der Entwickler für jeden Intent Beispiele definieren, welche der Benutzer während der Interaktion gebrauchen könnte. Diese Daten werden anschließend verwendet, um den Chatbot zu trainieren, sodass auch leicht abweichende Eingaben mit gleicher Bedeutung erkannt und richtig zugeordnet werden können. Jede Trainingsphrase wird in natürlicher Sprache verfasst und mit Anmerkungen, den Annotations, versehen. Durch diesen Prozess werden einzelne Wörter der Phrase mit einer bestimmten Entity verknüpft und können in den späteren Nutzereingaben jeden beliebigen Wert dieser Entity annehmen [31].

Responses

Reponses sind Antworten auf bestimmte Absichten des Benutzers, welche der Entwickler zur Verfügung stellen muss. Diese Antworten sind für jeden Intent individuell und können aus einfachem Text bestehen. Um die Eloquenz des Chatbots zu verbessen, empfiehlt sich, mehrere Variationen einer Textantwort zu hinterlegen. Wird der gleiche Intent innerhalb einer Konversation mehrmals ausgelöst, werden diese Variationen wechselweise angezeigt. Dies soll dazu beitragen, die Agentensprache menschlicher erscheinen zu lassen. Innerhalb der Antworten kann auf alle vorhandenen Parameter aus Intent und Contexts zugegriffen werden. Diese Werte können als Referenzen verwendet werden, um beispielsweise die Begrüßung mit dem Namen des aktuellen Nutzers zu versehen [31]. Wird der Chatbot in eine Messaging-Plattform integriert, können Antworten auch aus sogenannten Rich Messages bestehen. Diese bieten vordefinierte Elemente wie Bilder, Slider oder Karten [32].

Fulfillment

Fulfillment wird dazu verwendet, den Chatbot mit einem Webservice zu verbinden. Dies geschieht über einen sogenannten Webhook, einem URL, der vom Entwickler definiert wird. Dieser Webhook ermöglicht, relevante Informationen aus einem bestimmten Intent an einen Webservice zu übertragen. Im Webservice kann nachfolgend die notwendige Logik zur Erfüllung des Intents ausgeführt und schriftliche, visuelle oder gesprochene Antworten an Dialogflow zurückgesendet werden [30].

Dialogflow bietet zwei unterschiedliche Editionen zum Entwickeln an, eine Standard- und eine Enterprise-Edition. Beide bieten dem Entwickler die nahezu gleichen Funktionalitäten. Bei der Standard-Edition werden die Interaktionen durch Nutzungskontingente begrenzt und der Support erfolgt per E-Mail und durch die Community. Die kostenpflichtige Enterprise-Edition bietet unbegrenzte Text- und Sprachinteraktionen und Unterstützung durch den Google Cloud Support [47].

Dialogflow wird unter anderem von KLM Royal Dutch Airlines[9] für den Customer-Support verwendet. Die niederländische Fluggesellschaft bietet ihren Kunden einen Buchungs- und Packhilfen-Chatbot namens *BB*[10], der in Facebook Messenger integriert wurde. Der Bot kann Buchungsbestätigungen senden, Kunden an den Check-In erinnern

[9] https://www.klm.com/home/at/de
[10] https://bb.klm.com/en

und sie über Verspätungen oder Änderungen informieren. Der Kunde hat außerdem die Möglichkeit, bei Bedarf seinen Sitzplatz umzubuchen. Über den Bot erhält KLM durchschnittlich fünf Anfragen pro Minute [12, S. 238ff].

Auch die Schnellrestaurantkette Domino's[11] und Ticketmaster[12], ein Anbieter für Ticketlösungen, verwenden mit Dialogflow erstellte Chatbots.

3.3 Methoden zur Erstellung von Trainingsdaten

Für die Entwicklung von Chatbots mithilfe der zuvor beschriebenen Bot-Frameworks werden Dialoge benötigt, welche aus Nutzeranfragen sowie zugehörigen Antworten des Chatbots bestehen. Die eingehenden Anfragen werden durch Trainingsdaten beschrieben und helfen somit, die Anliegen der Chatbot-Nutzer zu verstehen und richtig zuordnen zu können.

Dass die Erstellung von nützlichen Dialogen eine relevante Problemstellung in der Entwicklung von Chatbots bildet, zeigt die große Anzahl an Tutorials, bewährten Vorgehensweisen und Hilfestellungen, die im Internet angeboten wird. Auch die effiziente Erstellung von Trainingsdaten wurde bereits in einigen wissenschaftlichen Arbeiten beschrieben, siehe Negi u. a. [15] und Abushawar und Atwell [1]. Umgesetzt und als Produkt verkauft wurden allerdings nur wenige davon. Einige der Methoden, die umgesetzt wurden, sind durch die rasche technische Entwicklung in den letzten Jahren hingegen bereits obsolet.

Der einfachste Weg Trainingsdaten zu entwickeln, ist die Verwendung von bereits vorhandenen Daten. Für die Erstellung eines Customer-Service-Chatbots kann auf bestehende Inhalte von FAQ-Seiten und Support-Anfragen zurückgegriffen werden. Außerdem sollte persönliches Wissen in die Trainingsdaten miteinfließen [44]. Somit kann der Chatbot häufig angefragte Themen zwar abdecken, der Aufwand und die benötigte Zeit zum Erstellen der Trainingsdaten sind allerdings sehr hoch.

Eine weitere Methode um Trainingsdaten für zielorientierte Chatbots zu entwickeln, ist das automatische Extrahieren von Dialogmodellen aus Gesprächsprotokollen. Dabei werden im ersten Schritt semantisch ähnliche Äußerungen innerhalb dieser Daten zusammengefasst. Anschließend werden daraus Sub-Tasks gesammelt, die vom Nutzer des Chatbots abgearbeitet werden müssen, um das gewünschte Ziel erreichen zu können. Im letzten Schritt werden diese Sub-Tasks als AIML-Konstrukte abgebildet. Mithilfe von AIML-Interpretern können so zielorientierte Chatbots gebaut werden [15, S. 890ff].

Ein ähnlicher Ansatz beschäftigt sich mit der automatischen Extrahierung von Trainingsdaten aus natürlichen Dialogkorpora. Als Korpus wird eine Sammlung von Texten und sprachlichen Äußerungen in begrenzter Anzahl bezeichnet. Dabei werden die maschinenlesbaren Texte in ein bestimmtes Format gewandelt, das zum Trainieren von Chatbots verwendet werden kann [1, S. 29ff].

Vorteile dieser Methoden sind die Zeiteinsparung und die große Anzahl an generierten Trainingsdaten. Da die vorhandenen Daten automatisch verarbeitet werden, muss der Entwickler keine manuellen Trainingsdaten mehr erstellen. Außerdem können menschliche Fehler bei der Erstellung damit vermieden werden.

[11] https://dialogflow.com/case-studies/dominos/
[12] https://dialogflow.com/case-studies/ticketmaster/

Kapitel 4

Eigener Ansatz

In diesem Kapitel wird die auftretende Problematik bei der Erstellung von Trainingsdaten erläutert. Ebenso werden grundlegende Funktionalitäten und Anforderungen an den zu implementierenden Chatbot gestellt. Abschließend wird eine eigene Methode zur semiautomatischen Generierung von domänenspezifischen Trainingsdaten für zielorientierte Chatbots vorgestellt.

4.1 Problemstellung

Obwohl bereits viele Technologien für die Entwicklung von Chatbots veröffentlicht wurden, werden alle Entwickler mit dem gleichen Problem konfrontiert: Chatbots benötigen Daten, um die Anfragen der Nutzer zu verstehen und richtig zuordnen zu können. Außerdem sollten für diese Anfragen auch Daten verfügbar sein, die den Nutzern Antworten auf ihre Anliegen liefern.

Um eine gute Klassifizierungsgenauigkeit der Benutzereingaben zu erreichen, ist es wichtig, dem Chatbot ausreichend Daten zur Verfügung zu stellen. Je mehr Trainingsphrasen in einem Intent vorhanden sind, desto höher ist die Klassifizierungsgenauigkeit, also die Wahrscheinlichkeit, dass die Benutzereingabe entsprechend zugeordnet, und der richtige Intent ausgelöst wird [33]. Dennoch gilt hier Qualität vor Quantität. Unterscheiden sich die Trainingsphrasen eines Intents zu sehr voneinander, wird der Chatbot Probleme bei der Klassifizierung haben. Ebenso sollten zwei Phrasen unterschiedlicher Intents nicht zu ähnlich zueinander sein.

Vor der Erstellung der Trainingsdaten sollten zuerst einige grundlegende Anforderungen geklärt werden. Für gewöhnlich sind Chatbots auf eine oder mehrere spezielle Zielgruppen abgestimmt [45]. Dies hilft nicht nur, eine entsprechende Sprache für die Trainingsdaten zu finden, sondern auch die geeignete Integrationsplattform auszuwählen. Mittels der definierten Zielgruppen werden vor der Entwicklung der Trainingsdaten außerdem Ziele festgelegt, welche durch die Konversation mit dem Chatbot erreicht werden können. Alle dabei benötigten Zwischenschritte müssen in den Trainingsdaten vorhanden sein. Außerdem müssen zum Erreichen der festgelegten Ziele entsprechende Daten in einer Datenbank existieren [45].

Anhand dieser präzisierten Informationen können Trainingsdaten erstellt werden, die auf die definierten Ziele der Nutzer abgestimmt sind. Hierbei sollten möglichst vie-

le verschiedene Formulierungen und unterschiedliche Lösungen für die einzelnen Ziele abgedeckt werden. Auch Hilfestellungen für den Fall, dass Nutzer von der planmäßigen Konversation abweichen, können in den Trainingsdaten enthalten sein.

Ein großer Vorteil von Chatbots ist die relativ rasche und einfache Entwicklung. Mit den in Abschnitt 3.2 beschriebenen Frameworks und Plattformen können mühelos einfache Chatbots erstellt und veröffentlicht werden. Ein Großteil der verfügbaren Chatbots ist allerdings nicht in der Lage, die Nutzer richtig zu verstehen. Dies liegt einerseits an zu wenigen oder mangelhaften Trainingsdaten, andererseits kann auch fehlendes Training Probleme bei der Zuordnung der Anfragen verursachen. Gründe für die geringe Anzahl an Trainingsdaten und die inhaltlichen Unzulänglichkeiten sind oftmals fehlendes Wissen und Zeitmangel bei der Entwicklung des Dialogs. In diesem Abschnitt der Entwicklung sollte allerdings genügend Arbeit investiert werden, da Chatbots nur dann richtig funktionieren, wenn sie ausreichend gutes Training erhalten. Grundsätzlich sollte jedem Entwickler bewusst sein, dass das Training des Chatbots nicht mit seiner Veröffentlichung abgeschlossen ist. Um den Chatbot aktuell und intakt zu halten, müssen regelmäßige Analysen durchgeführt werden. Nur so kann aus bisherigen Nutzerinteraktionen und Rückmeldungen gelernt, und der Chatbot stetig verbessert werden. Diese Verbesserungen können einerseits die verwendete Sprache betreffen, andererseits auch Konversationsabläufe oder die Funktionalität des Chatbots [61].

Sowohl die Erstellung der Trainingsdaten, als auch deren Instandhaltung und Verbesserung nehmen viel Zeit in Anspruch. Deshalb empfiehlt sich eine effiziente und strukturierte Arbeitsweise.

4.2 Funktionalität und Anforderungen

Die grundlegende Funktionalität des zu entwickelnden Chatbots ist die einfache Suche nach Unterkünften. Um diese Suche im Dialog so natürlich wie möglich zu gestalten, soll dem Nutzer die Möglichkeit gegeben werden, mit den Ergebnissen zu interagieren und Fragen zu stellen. Dadurch unterscheidet sich der Chatbot von der Konkurrenz, da diese oftmals nur wenig Interaktion erlaubt. Im Idealfall entscheidet sich der Nutzer nach der Suche für eine Unterkunft und wird per Link auf die Buchungsseite geleitet. Dort werden anschließend alle relevanten, personenbezogenen Daten abgefragt und die Unterkunft kostenpflichtig gebucht.

Im Unterschied zur traditionellen Suche nach Unterkünften kann der Nutzer über den Chatbot auch für ihn relevante Zusatzinformationen erhalten, welche auf der Website aufgrund von Designentscheidungen oder Platzmangel nicht angezeigt werden. Der Nutzer muss nicht zwischen Übersichts- und Detailseiten navigieren, sondern erhält alle Daten gesammelt direkt im Chatbot. Außerdem soll durch eine klare Abgrenzung der Wünsche die Suche so genau wie möglich gestaltet, und dem Nutzer nur eine begrenzte Anzahl an Ergebnissen angezeigt werden.

Durch Neugier kann der Chatbot zusätzlich das generelle Interesse an der Website und an der Buchung einer Unterkunft steigern. Laut dem Mobile Travel Report von KAYAK aus dem Jahr 2017 haben 77 % der befragten Deutschen noch nie einen Chatbot benutzt. Dennoch könnten sich 27 % vorstellen, einen Chatbot zu nutzen, um Flüge, Hotels und Mietwagen zu suchen und zu buchen. Und auch von den 11 % der Befragten, die bereits einen Chatbot verwendet haben, nutzten ihn 23 % zu diesem Zweck [42, S.

25ff].

Um möglichst viele potenzielle Kunden zu erreichen, sollte die Wahl der Integrationsplattform des Chatbots gut überlegt sein. Dazu müssen zuerst die Zielgruppe und deren Bedürfnisse analysiert werden. Wird der Chatbot in eine Plattform integriert, die die Zielgruppe nicht verwendet, wird sich der Aufwand der Entwicklung nicht lohnen. Bestehende Chatbots, die ebenfalls in der Tourismusbranche verwendet werden, wurden in Facebook Messenger[1], Telegram[2] und Slack[3] integriert. In diesem Projekt wird der Chatbot in eine bestehende Website integriert, um den Chatbot einem möglichst breiten Publikum zur Verfügung stellen zu können.

Bei der raschen und effizienten Entwicklung des Chatbots soll der im Folgenden beschriebene Trainingsdaten-Generator helfen. Dieser unterstützt Entwickler einerseits bei der Erstellung der benötigten Trainingsdaten, andererseits soll so auch die Weiterentwicklung und Instandhaltung des Chatbots erleichtert werden.

4.3 Erstellung von Trainingsdaten

Die in Abschnitt 3.3 angeführten bestehenden Ansätze zur Erstellung von Trainingsdaten benötigen eine große Anzahl an vorhandenen Daten in speziellen Strukturen. Im Folgenden wird eine Methode beschrieben, bei der domänenspezifische Trainingsdaten für verschiedene zielorientierte Chatbots generiert werden können. Dazu werden ausschließlich strukturierte Daten benötigt.

Die zu implementierende Methode basiert auf Slot-Filling mithilfe von Natural Language Understanding (NLU) unter der Verwendung von Ontologien. In der Informatik bezeichnet eine Ontologie einen gesetzten oder abgeleiteten Kontext, welcher zur Wissensrepräsentation und formalen Begriffsanalyse dient. Mithilfe von Ontologien kann eine Menge an Begrifflichkeiten in Beziehung zueinander dargestellt werden [9, S. 281ff].

Für die gewünschte Domäne werden unterschiedliche Ziele definiert. Als Domäne wird der grobe Themenbereich des Chatbots bezeichnet. Um das definierte Ziel zu erreichen, muss der Nutzer festgelegte Informationen angeben. Diese Informationen werden durch die sogenannten Slots repräsentiert, welche der Reihe nach abgearbeitet werden müssen. Dieser Vorgang wird als Slot-Filling bezeichnet. Für jeden Slot wird ein spezieller Typ festgelegt, welcher die erlaubten Werte, die innerhalb des Slots verwendet werden dürfen, definiert. Tabelle 4.1 zeigt mögliche Slots für das Suchen einer Unterkunft, mit allen dazu erlaubten Typen. Diese Architektur wurde bereits 1977 in einem experimentellen Computersystem namens *GUS* (Genial Understander System) vorgestellt und als Reisevermittler verwendet [2].

Umgesetzt wird die Methode mithilfe von Ontologien und einer Art Bausteinsystem. Die einzelnen Domänen werden durch JSON-Entitäten repräsentiert. Programm 4.1 zeigt die verwendete Struktur der Domäne Sehenswürdigkeit[4]. Diese enthält Eigenschaften, welche die einzelnen Slots symbolisieren. Die Werte dieser Eigenschaften beschreiben mögliche Informationen, die vom Nutzer angegeben werden können.

Der Chatbot-Entwickler kann für die Generierung der Trainingsdaten strukturierte

[1] https://www.messenger.com/
[2] https://telegram.org/
[3] https://slack.com/intl/de
[4] https://schema.org/TouristAttraction

Suche nach Unterkünften	
Slot	*Typ*
Anreise	Datum
Abreise	Datum
Personen	Zahl
Urlaubsort	Stadt
Unterkunftsart	Hotel, Frühstückspension, Ferienhaus
Einrichtung	Pool, Fitness-Studio, Tiefgarage
Klassifizierung	Sterne

Tabelle 4.1: Festgelegte Slots mit allen erlaubten Typen und Werten für die Suche nach Unterkünften, in Anlehung an Jurafsky und Martin [10, S. 10].

Programm 4.1: JSON-Entität für die Domäne Sehenswürdigkeit von Schema.org.

```
{
  "@context": "http://schema.org",
  "@type": ["TouristAttraction"],
  "name": "Disneyland Paris",
  "address":{
    "@type":"PostalAddress",
    "addressLocality":"Paris",
    "addressCountry":"FRA"
  },
  "description": "It's an amusement park in Marne-la-Vallee, near Paris, in France
    and is the most visited theme park in Europe.",
  "openingHours": ["Mo-Fr 10:00-19:00", "Sa 10:00-22:00", "Su 10:00-21:00"],
  "isAccessibleForFree": false,
  "currenciesAccepted": "EUR",
  "paymentAccepted": "Cash, Credit Card",
  "touristType": {
    "@type": "Audience",
    "audienceType": "children"
  },
  "availableLanguage":[
    "English",
    "French",
    "German"
  ],
  "telephone": ["+34983878011","+34659843463"],
  "url": "http://www.disneylandparis.fr"
}
```

Auszeichnungen verwenden, die bereits auf der Website existieren. Dadurch kann garantiert werden, dass für alle Eigenschaften, die in der JSON-Struktur vorkommen, Werte in einer Datenbank vorhanden sind. Aber auch bei Domänen, bei denen keine Daten oder bestehende Auszeichnungen verfügbar sind, ist diese Methode hilfreich. Der Entwickler erhält durch die vordefinierten JSON-Strukturen, die auf verschiedenen Webseiten heruntergeladen werden können, einen Überblick über wichtige Daten, die für den Dialog

Satzstrukturen					
Suchen	*Unterkunftsart*	*Personen*	*Ort*	*Anreise*	*Abreise*
Zeig mir	Hotels	für 2	in Wien	vom 1. Mai	bis zum 9. Mai
Zeig uns	Pensionen	für 2	in Wien	vom 1. Mai	bis zum 9. Mai
Ich suche	Ferienhäuser	für 5	in Linz	von morgen	bis Freitag
Wir suchen	Hotels	für 5	in Wien	von morgen	bis Freitag
Ich möchte	ein Zimmer	für 2	in Linz	vom 1. Mai	bis zum 6. Mai

Tabelle 4.2: Verwendete Satzstrukturen zum Abarbeiten der definierten Slots, abgewandelt von Jurafsky und Martin [10, S. 10].

des Chatbots benötigt werden.

Die gewählte JSON-Struktur kann nach dem Import in den Generator beliebig benannt werden. Hierbei können weitere Synonyme für die Bezeichnung der Domäne in Ein- und Mehrzahl angegeben werden. Diese Maßnahme hilft, ein möglichst breites Spektrum an potenziellen Formulierungen von Chatbot-Anfragen abzudecken. Auch die einzelnen Eigenschaften der Domäne, die benötigten Slots, werden vom Benutzer des Generators entsprechend benannt. Erforderliche Parameter, die nicht als Eigenschaft in der JSON-Struktur vorhanden sind, können nach dem Import in den Generator hinzugefügt werden, überflüssige entfernt werden.

Zum Generieren der Trainingsphrasen werden vordefinierte Satzstrukturen verwendet. Diese Strukturen sind in unterschiedliche inhaltliche Bereiche gegliedert und umfassen die Suche nach Ergebnissen, Änderungen der Suchparameter, Interaktionen mit den Ergebnissen und allgemeine Ausdrücke, wie Begrüßungen und Hilfestellungen. Diese inhaltlichen Bereiche repräsentieren die späteren Intents des Chatbots und erfüllen mittels Slot-Filling das definierte Ziel der Domäne. Tabelle 4.2 zeigt die Struktur bei der Suche nach Unterkünften mit den in Tabelle 4.1 festgelegten Slots und erlaubten Typen.

Die vordefinierten Satzstrukturen können an unterschiedlichen Positionen Platzhalter enthalten. Diese Platzhalter werden beim Generierungsvorgang mit den erlaubten Werten der einzelnen Slots befüllt. Die Slots stellen somit nicht nur die Eigenschaften der Domäne dar, sondern auch die benötigten Parameter der Action eines Intents. Die erlaubten Typen eines Slots werden in Dialogflow durch Entities repräsentiert und beim Generieren automatisch erstellt.

Grundsätzlich existieren zwei Ausprägungen von Platzhaltern, welche unterschiedlich bezeichnet werden. Einerseits sind in den Satzstrukturen Platzhalter für die möglichen Bezeichnungen der Domäne vorhanden. Diese werden als [TYPE] gekennzeichnet und abwechselnd mit den zuvor festgelegten Ausdrücken befüllt. Außerdem repräsentieren mit [PARAM] definierte Platzhalter die einzelnen Werte, die die importierten Parameter innerhalb der Sätze und Slots annehmen können. Die Platzhalter für die Parameter können an unterschiedlichen Positionen im Satz vorkommen, wie Tabelle 4.3 zeigt. Sie werden entweder vor oder nach der Bezeichnung der Domäne in die Satzstruktur eingefügt. Dabei wird zwischen Adjektiven und Substantiven unterschieden. Für Parameterwerte, die durch ein Adjektiv repräsentiert werden, müssen spezielle Anpassungen, sogenannte Deklinationen, vorgenommen werden, damit grammatikalisch

Zeig mir [PARAM] [TYPE] [PARAM].
Zeig mir Unterkünfte.
Zeig mir Hotels in Linz.
Zeig mir familienfreundliche Unterkünfte in Linz.
Zeig mir Sport-Hotels in Wien.
Zeig mir Wellness-Unterkünfte für Familien in Linz.
Zeig mir barrierefreie Design-Hotels in Wien mit Vollpension.
Zeig mir tierfreundliche Apartment-Unterkünfte für Familien in Linz mit Frühstück.
Ich suche Unterkünfte.
Ich suche Ferienhäuser in Wien.
Ich suche günstige Unterkünfte in Wien.
Ich suche Design-Hotels in Wien.
Ich suche Wellness-Unterkünfte für Paare in Wien.
Ich suche günstige Sport-Hotels in Wien mit Halbpension.
Ich suche tierfreundliche Design-Unterkünfte für Familien in Linz mit Frühstück.

Tabelle 4.3: Mögliche Ausprägungen der gegebenen Satzstrukturen.

korrekte Trainingsphrasen generiert werden können. Um dies zu erreichen, werden zusätzliche Angaben benötigt. Diese Angaben betreffen die Artikel der Domänenbezeichnungen und der Parameter sowie die Filterbarkeit und die Stellung der Parameter. Die Filterbarkeit eines Parameters ist gegeben, wenn das Vorhandensein dieses Wertes die Ergebnisse der Suche verändert. Beispiele für filterbare Parameter sind Adresse, Datum oder Preis. Nicht filterbare Parameter beeinflussen die Suche nach Ergebnissen hingegen nicht. Sie dienen lediglich zur Suche nach weiteren Informationen von bereits gefundenen Ergebnissen. Beispiele hierfür sind Kontaktdaten, Beschreibungen oder Anfahrtsinformationen.

Für jeden Parameter werden außerdem Werte benötigt, die in die definierten Satzstrukturen an den unterschiedlichen Positionen der Platzhalter eingesetzt werden können. Diese Ausprägungen repräsentieren die Werte innerhalb der zugehörigen Entities und müssen für jede mögliche Stellung innerhalb der Satzstruktur angeführt werden. Anhand der vorhandenen Informationen zum aktuellen Parameter und der Domäne werden diese Werte entsprechend in die Platzhalter eingesetzt. Um möglichst viele Variationen und Reihenfolgen der vorhandenen Parameter abzudecken, werden hierfür alle definierten Bezeichnungen und Synonyme der Domäne und der einzelnen Parameter verwendet.

Auch die Anzahl an Parametern innerhalb eines Satzes wird variiert, um allen denkbaren Nutzerangaben in den Trainingsdaten gerecht zu werden. In der Maximalausprägung des Satzes können somit an einer Platzhalter-Position alle gegebenen Parameter eingefügt werden, siehe Tabelle 4.3. In der Minimalausprägung wird der Platzhalter mit keinem Wert befüllt, sondern vollständig aus dem Satz entfernt. Außerdem ist zu beachten, dass ein Parameter an allen drei möglichen Positionen der Platzhalter vorkommen kann. So lassen sich in diesem Schritt der Generierung unzählige Variationen bilden. Dieser Vorgang wird für alle existierenden Satzstrukturen durchgeführt. Der Benutzer des Generators kann diese Satzstrukturen nach Belieben erweitern. Dabei müssen die

benötigten Platzhalter selbst gewählt und entsprechend gekennzeichnet werden. Satzstrukturen, die nicht in das Dialogkonzept des Chatbots passen, können entfernt werden.

Contexts werden beim automatischen Generieren der Trainingsdaten nicht gesetzt. Das Erkennen der Zusammenhänge und Abhängigkeiten zwischen den einzelnen Intents bleibt dem Entwickler des Chatbots überlassen. Responses und mögliches Fulfillment werden im Generierungsvorgang ebenfalls nicht abgedeckt. Die Verwendung von Fulfillment und die Antworten auf die Nutzeranfragen sind projektabhängig und können nicht allgemein formuliert werden. Denn Antworten werden einerseits statisch eingepflegt, oder anhand von Datenbankabfragen oder externen APIs zusammengestellt.

Die generierten Trainingsdaten können als formatiertes Dialogflow-Projekt heruntergeladen werden. Dabei werden alle benötigten Intents und Entities erstellt und mit den Trainingsphrasen und Daten befüllt. Der Entwickler kann dieses Projekt in die Weboberfläche von Dialogflow importieren und das Chatbot-Training starten.

Bei der Verwendung von anderen Chatbot-Plattformen können die generierten Trainingsphrasen auch als einfache Textdatei heruntergeladen werden. Die Phrasen sind innerhalb dieser Textdatei nach Intents sortiert. Der Entwickler muss diese Intents und alle benötigten Entities in der gewählten Entwicklungsplattform selbst erstellen und manuell mit den generierten Phrasen befüllen.

Auch die Weiterentwicklung des Chatbots soll mithilfe des Generators vereinfacht werden. Durch die Auswertung von nicht zugeordneten Nutzeranfragen können Rückschlüsse über mangelhafte Trainingsdaten und fehlende Funktionalitäten gezogen, und der Chatbot dahingehend verbessert werden. Dazu werden Anfragen, die der Chatbot nicht interpretieren kann, gespeichert und die wichtigsten Schlüsselwörter daraus extrahiert.

Die beschriebene Methode bietet gegenüber bereits existierenden Ansätzen einige Vorteile. Zur Generierung werden keine Unmengen an initialen Daten benötigt. Der Entwickler kann bereits bestehende strukturierte Auszeichnungen für die gewählte Domäne des Chatbots verwenden und die gewünschten Eigenschaften daraus extrahieren. Diese JSON-Struktur kann jederzeit angepasst werden, um fehlende Informationen zu ergänzen oder irrelevante Daten zu löschen.

Auch die gegebenen Satzstrukturen, in denen die Parameter eingesetzt werden, bieten ein hohes Maß an Wartbarkeit. Innerhalb der einzelnen inhaltlichen Bereiche können bestehende Strukturen erweitert beziehungsweise entfernt werden. Dadurch bietet sich die Möglichkeit, spezielle Ausdrucksformen, regionale Dialekte oder produktspezifische Formulierungen in den Trainingsdaten abzubilden.

Neben dem Export als Textdatei bietet der Generator auch die Möglichkeit, die Daten als komplettes Dialogflow-Projekt herunterzuladen. Somit benötigt der Entwickler keine zusätzliche Zeit, um die erforderlichen Intents und Entities zu erstellen und mit den generierten Trainingsdaten zu befüllen. Außerdem können dadurch Fehler beim manuellen Erstellen und Einpflegen der Daten verhindert werden.

Durch das simple Konzept des Generators kann die benötigte Gesamtzeit, von der initialen Suche nach Datenstrukturen, bis hin zum Generieren der Trainingsdaten und dem anschließenden Import in Dialogflow geringgehalten werden.

Kapitel 5

Implementierung

Dieses Kapitel zeigt die Implementierung des Chatbots zum Suchen und Buchen von Unterkünften sowie des semiautomatischen Trainingsdaten-Generators. Dazu werden die Technologien, die zur Umsetzung verwendet werden, vorgestellt und näher auf deren Verwendung im Projekt eingegangen.

5.1 Verwendete Technologien

Um die in Abschnitt 4.2 beschriebenen Funktionalitäten und Anforderungen bei der Umsetzung des Chatbots, und die in Abschnitt 4.3 erläuterte Methode zur Trainingsdatengenerierung realisieren zu können, werden verschiedenste Technologien verwendet. Die rasche Erstellung eines neuen Projektes und die Möglichkeit zur vielseitigen Integration des Chatbots standen bei der Auswahl dieser Technologien im Vordergrund.

5.1.1 Dialogflow

Die Entscheidung das Bot-Framework Dialogflow, beschrieben in Abschnitt 3.2.4, zu verwenden, wurde aufgrund mehrerer Faktoren getroffen. Einerseits ist Dialogflow ein beliebtes Mittel zur Erstellung von Chatbots, viele existierende Chatbots basieren darauf. Somit ist die Suche nach Problemlösungen in der Entwicklung einfacher, da in vielen Foren darüber diskutiert wird und Hilfestellungen angeboten werden. Dialogflow besitzt ein eigenes Forum[1], in dem Entwickler Fragen stellen, Features diskutieren und Feedback geben können. Andererseits bietet Dialogflow selbst eine beachtliche Dokumentation für Entwickler. Zahlreiche Anfängerbeispiele und Tutorials sollen den Einstieg in die Chatbot-Entwicklung erleichtern.

Außerdem ist Dialogflow eine Web-Applikation und kann somit überall und ohne zusätzliche Installation genutzt werden. Das Interface ist sehr übersichtlich und intuitiv. Der Chatbot kann zu jeder Zeit unkompliziert geändert, erweitert und neu trainiert werden. Die aktuelle Version wird immer automatisch in allen verwendeten Integrationen genutzt. Auch die genaue Aufzeichnung der Konversationen ist von Vorteil, da so Rückschlüsse über die Qualität des Chatbots gezogen werden können. Werden Eingaben des Nutzers vom Chatbot nicht erkannt und somit keinem Intent zugeordnet, kann

[1] https://productforums.google.com/forum/#!forum/dialogflow

der Entwickler diese Eingaben auswerten und den Chatbot dementsprechend anpassen. Dazu kann auch der Analyse-Bereich von Dialogflow verwendet werden. Jedes Mal, wenn eine Konversation mit dem Chatbot begonnen wird, werden die Uhrzeit und die Anzahl an Anfragen aufgezeichnet und als Diagramm im Analyse-Bereich dargestellt. So kann der Entwickler Trends erkennen und die Inanspruchnahme seitens des Nutzers beobachten. Auch die Zeit, die der Chatbot benötigt, um auf eine Anfrage zu reagieren wird gespeichert. Der Entwickler erhält Einblicke in den Ablauf, beziehungsweise in die Reihenfolge der einzelnen Intents, deren Aufrufhäufigkeit und die Ausstiegsquote der Nutzer.

All diese Gründe waren ausschlaggebend, Dialogflow zur Entwicklung des Chatbots zu verwenden.

5.1.2 Pimcore

Pimcore[2] ist eine Open-Source-Plattform für Unternehmen. Dabei werden Produktinformationsmanagement (PIM), User Experience Management (CMS/UX), Digital Asset Management (DAM) und eCommerce in einer webbasierten Benutzeroberfläche vereint. Pimcore wurde für Entwickler entworfen, das heißt, es ist kein *out-of-the-box*-Softwareprodukt, wie Wordpress[3] oder Magento[4]. Es bedarf Entwickler, um die gewünschten Projekte zu realisieren. Grundsätzlich basiert Pimcore auf PHP, MySQL und MariaDB und folgt dem MVC-Pattern [38]. Seit der im September 2017 veröffentlichten Version 5 baut Pimcore auf das PHP-Framework Symfony 3[5] auf. Somit ist es möglich, die Kernfunktionalität von Pimcore mithilfe von Bundles zu erweitern [37].

Pimcore erhielt bereits einige namhafte Auszeichnungen. Im Jahr 2016 wurde Pimcore der *Technology Innovation Leader*-Award von Frost & Sullivan verliehen [34] und auch beim CMS Critic Award 2017 konnte Pimcore überzeugen und erhielt die Auszeichnung *Best Enterprise CMS* [43].

Zu den Unternehmen, die Pimcore für ihre Projekte verwenden, zählen unter anderem Intersport Österreich[6] und IKEA Deutschland[7]. Aber auch viele heimische Tourismusbetriebe und Bergbahnen, wie Steiermark Tourismus[8] und Salzburg Tourismus[9] realisieren ihre Webauftritte und Buchungsportale mit Pimcore.

Um den Chatbot möglichst rasch auf diesen Webseiten einsetzen zu können, wird das Projekt als Pimcore-Bundle umgesetzt. Somit können sowohl der Chatbot, als auch der Trainingsdaten-Generator in allen versionskompatiblen Pimcore-Instanzen installiert und verwendet werden. Der Aufwand einer Neuintegration wird so gering wie möglich gehalten. Dazu wird auf eine modulare Vorgehensweise großen Wert gelegt. Alle Projekte, die das Chatbot-Bundle in Pimcore installieren können, verwenden die gleiche zugrundeliegende Datenstruktur. Alle Datenbankabfragen können ohne Anpassungen verwendet werden, die gelieferten Ergebnisse entsprechen einem definierten Format. Die

[2] https://pimcore.com/de
[3] https://de.wordpress.org/
[4] https://magento.com/
[5] https://symfony.com/
[6] http://www.intersport.at/
[7] https://www.ikea.com/de/de/
[8] https://www.steiermark.com/de
[9] https://www.salzburg.info/de

verwendeten Entities werden regelmäßig aus der Datenbank geladen und automatisch in Dialogflow importiert, damit der Chatbot immer auf dem aktuellen Stand bleibt.

5.1.3 JSON

Für die Speicherung der Daten, die für den Trainingsdaten-Generator notwendig sind, wird JSON verwendet. JSON steht für JavaScript Object Notation und ist eine Untergruppe von JavaScript. Als kompaktes Dateiformat in Textform eignet sich JSON für den Datenaustausch und zum Speichern von Daten. Das Format beschreibt Datenstrukturen als Schlüssel-Wert-Paare [16, S. 42ff].

Da die Daten für den Trainingsdaten-Generator leicht erweiterbar sein sollen, eignet sich JSON hier besonders gut. Über eine PHP-Funktion kann aus der JSON-Datei gelesen und deren Inhalt in ein JavaScript-Objekt konvertiert werden. Dieses Objekt kann anschließend verändert, und wieder in eine JSON-Zeichenkette zurückkonvertiert werden. Für den Export der Trainingsdaten zu Dialogflow wird JSON ebenfalls benötigt, da Dialogflow-Intents in einer JSON-Datei abgebildet werden. Auch sämtlicher Datenaustausch zwischen Dialogflow, Integrationsplattform und Webhook findet in JSON statt.

5.1.4 JSON-LD

JSON-LD steht für JavaScript Object Notation for Linked Data und wird zur Strukturierung von Daten verwendet. Das Format erweitert JSON und besteht aus maschinenlesbaren Daten, die auch für Menschen leicht zu lesen und zu schreiben sind. Die Daten können einfach zwischen Webanwendungen ausgetauscht und automatisiert verarbeitet werden [55].

JSON-LD wird hauptsächlich zur Suchmaschinenoptimierung verwendet, da die ausgezeichneten Daten interpretiert und somit in Verbindung zueinander gebracht werden können. Beispiele für solche semantischen Auszeichnungen können Personen, Adressen, Produkte und Veranstaltungen sein. Die Suchmaschine ist in der Lage, diesen Kontext zu erfassen und so bessere Ergebnisse zu liefern.

Eine Sammlung von verschiedenem JSON-LD-Vokabular ist unter Schema.org[10] zu finden. Dieser Auszeichnungs-Standard wird auch für den Trainingsdaten-Generator verwendet.

5.2 Chatbot-Implementierung

Nachdem die Entscheidung bezüglich verwendeter Technologien getroffen wurde, startet die eigentliche Entwicklung des Chatbots.

Der erste Schritt ist dabei die Abgrenzung der Funktionalität. Der Chatbot benötigt ein klares Ziel und sollte dies auch eindeutig kommunizieren können. Als nächstes muss die Beschaffung der Daten geklärt werden. Der Chatbot kann die Daten einerseits über eine externe API laden, um jederzeit aktuelle Werte zu erhalten, andererseits können aber auch statische Antworten verwendet werden. Wird eine API genutzt, müssen die Parameter, die diese API für die erfolgreiche Suche nach Daten benötigt, evaluiert

[10] http://schema.org/

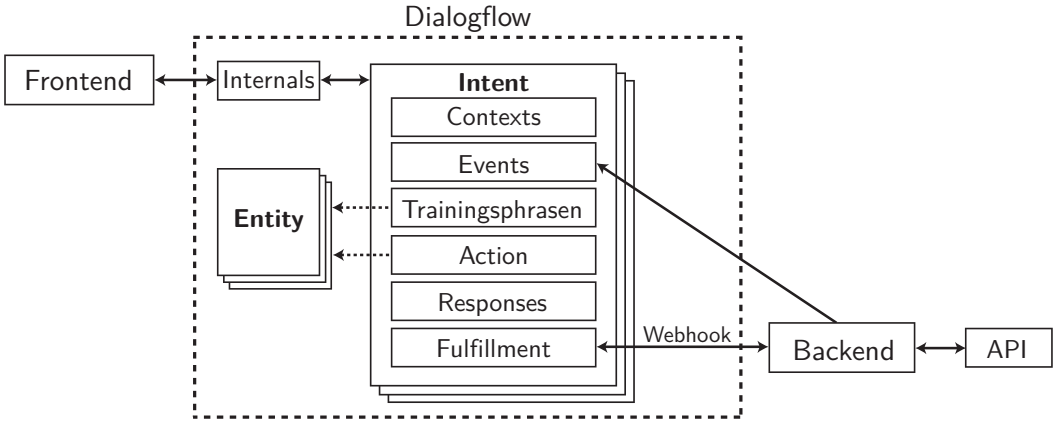


Abbildung 5.1: Zusammenspiel von Frontend, Backend und Dialogflow zur Beantwortung von Nutzeranfragen.

werden. Dabei sollte zwischen Pflichtparametern, die für die Suche zwingend erforderlich sind, und optionalen Parametern, die der Nutzer zusätzlich zur Verfügung stellen kann, um die Suche genauer zu gestalten, unterschieden werden. Auch das Format der benötigten Parameter muss beachtet werden. Dies ist vor allem bei zeitabhängigen Anfragen von Bedeutung, da unzählige Möglichkeiten existieren, ein und dasselbe Datum anzugeben.

Mit all diesen Erkenntnissen kann der nächste Schritt in der Entwicklung erfolgen: die Dialogfindung. Dabei gibt es, wie in Kapitel 3 beschrieben, unterschiedliche Vorgehensweisen. In diesem Schritt muss auch die Frage der unterstützten Sprachen geklärt werden. Soll der Chatbot mehrsprachig angeboten werden, muss der Dialog für jede benötigte Sprache entwickelt werden.

Sobald der Dialog finalisiert wurde, müssen die einzelnen Bereiche der Unterhaltung definiert werden. Jeder Bereich sollte einen eigenen Zweck erfüllen. In diesem Projekt werden die Begrüßung des Nutzers, der Start einer Suche oder das Anzeigen der Ergebnisse als solche Bereiche dargestellt und später als Intents umgesetzt. Danach kann mit der technischen Implementierung des Chatbots begonnen werden. Prinzipiell können dabei drei unterschiedliche Bereiche definiert werden: Dialogflow, Backend und Frontend. Diese Bereiche verfügen über abgegrenzte Zuständigkeiten und werden benötigt, um eine Nutzeranfrage entsprechend beantworten zu können. Abbildung 5.1 zeigt das Zusammenspiel dieser Bereiche.

Dialogflow bildet die grundsätzliche Logik des Chatbots und ist für den Dialog zuständig. Das Framework dient auch als Verbindung zwischen Frontend und Backend. Diese beiden Bereiche sind komplett eigenständig und können nicht direkt miteinander kommunizieren. Alle Interaktionen zwischen Frontend und Backend laufen über Dialogflow ab.

Das Frontend ist für die Übermittlung der Nutzereingaben und für die korrekte Darstellung der empfangenen Ergebnisse zuständig. Außerdem kann der Ablauf der Konversation im Frontend gesteuert werden. So können bestimmte Events durch die Interaktion des Nutzers, zum Beispiel beim Klick auf einen Button, ausgelöst werden.

Im Backend werden die Anfragen, die über den Webhook gesendet werden, entsprechend verarbeitet. Dazu wird die übergebene Action, die über die Ausführung der verschiedenen Funktionen entscheidet, herangezogen. Innerhalb dieser Funktionen können alle benötigten Informationen, die vom Nutzer angegebenen wurden, verwendet werden.

Grundsätzlich sollten diese drei Teile nicht als voneinander unabhängig betrachtet werden. Idealerweise wird nicht erst das gesamte Backend fertiggestellt, und anschließend mit der Darstellung im Frontend begonnen, sondern eine iterative Arbeitsweise gewählt. Dabei wird ein Feature nach dem anderen entwickelt. Zuerst wird der dafür benötigte Dialog erstellt und in Dialogflow umgesetzt. Danach wird die entsprechende Logik im Backend implementiert. Abschließend wird die Darstellung im Frontend erzeugt und das Feature vernünftig getestet, wobei auch fehlerhafte Eingaben verwendet werden, um den Chatbot auf Stabilität zu prüfen.

Erst wenn das erstellte Feature alle Tests besteht, wird der nächste Bestandteil mit der selben Vorgehensweise umgesetzt. Auch innerhalb der Features wird zuerst eine Basisversion implementiert, welche nach und nach erweitert wird. Durch dieses Konzept können Fehler schneller behoben werden, da nicht im gesamten Projekt nach ihnen gesucht werden muss. Außerdem können die einzelnen Features schneller getestet, und anhand des Feedbacks rasch adaptiert werden.

5.2.1 Ablauf einer Anfrage

Im Folgenden wird der Ablauf einer Benutzeranfrage an den Chatbot erläutert, wie in Abbildung 5.2 illustriert. Sobald der Nutzer über das Textfeld, welches im Frontend angezeigt wird, eine Anfrage an den Chatbot sendet, wird diese in Dialogflow interpretiert und dem entsprechenden Intent zugeordnet. Bei der Zuordnung wird nicht nur auf die definierten Trainingsphrasen und deren verknüpfte Entities geachtet, sondern auch auf den Eingangskontext.

Sobald ein Intent ausgelöst wurde, werden die übermittelten Werte überprüft und den entsprechenden Variablen zugeordnet. Sollten erforderliche Parameter nicht übermittelt worden sein, werden definierte Aufforderungen ausgegeben und die Anfrage wird erst weiterverarbeitet, sobald der Nutzer gültige Werte für diese Parameter angegeben hat. Danach wird die Antwort des Chatbots über den definierten Webhook eingeholt. Dazu werden die hinterlegte Action und die übergebenen Parameter an das Backend gesendet und dort anhand einer konkreten Funktion weiterverarbeitet. Die gelieferten Ergebnisse werden in eine zuvor gewählte Struktur gebracht, welche wieder an Dialogflow zurückgesendet wird. Dialogflow leitet diese Struktur als Antwort auf die getätigte Nutzeranfrage an das Frontend weiter. Anhand vordefinierter Funktionen wird diese Antwort im Frontend interpretiert und dem Nutzer schließlich angezeigt.

Um den beschriebenen Ablauf zu realisieren, werden die zuvor genannten Bereiche benötigt. Deren Implementierung wird in den folgenden Abschnitten genauer erläutert.

5.2.2 Logik des Chatbots

Um einen neuen Chatbot in Dialogflow zu erstellen, muss der Entwickler zunächst den gewünschten Namen des Agenten, so wie die bevorzugte Standardsprache angeben. Zum

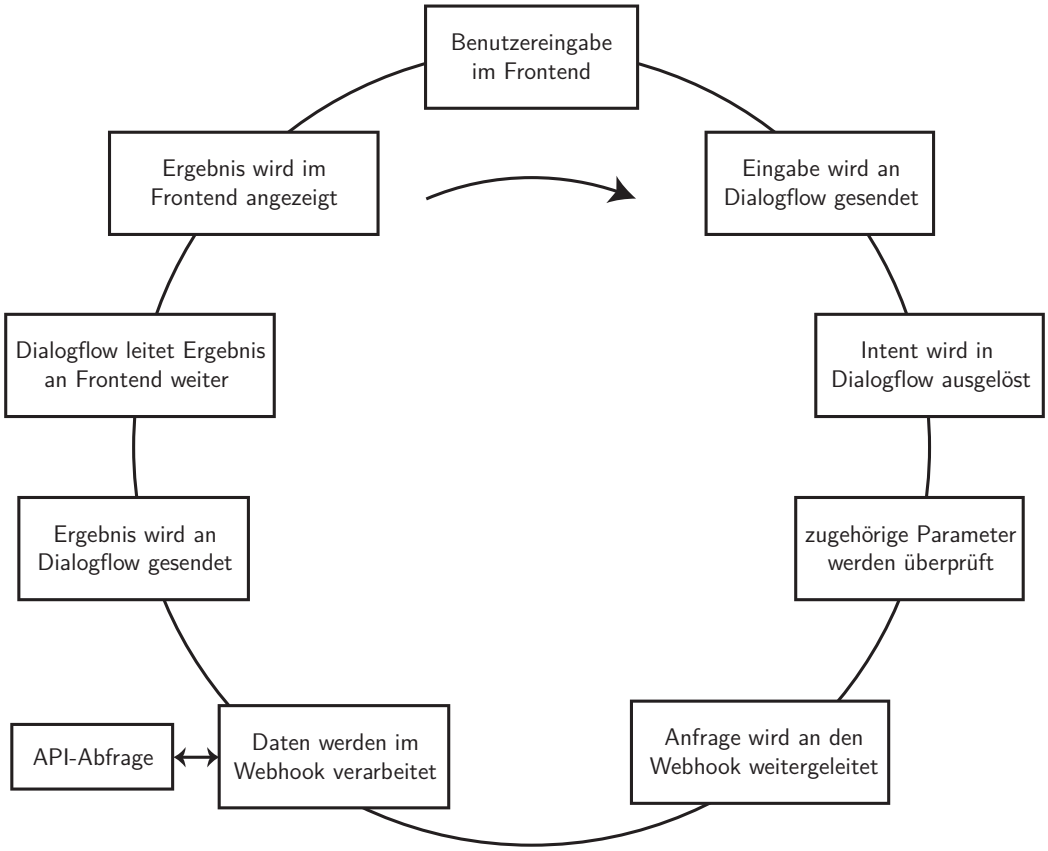


Abbildung 5.2: Ablauf einer Benutzeranfrage an den Chatbot.

Speichern des Agenten wird außerdem ein Google Cloud Platform[11]-Projekt benötigt. Zu Beginn werden die zuvor definierten Bereiche des Dialogs als eigenständige Intents im erstellten Projekt in Dialogflow umgesetzt. Dazu benötigt jeder Intent eine Vielzahl an Daten. Einerseits muss ein aussagekräftiger Name für den Intent gewählt werden. Andererseits müssen Eingangs- und Ausgangskontexte sowie deren Lebensdauer bestimmt werden. Der Kontext ist vor allem bei der Suche nach Unterkünften wichtig, um alle bereits eingegebenen Nutzerinformationen von einem Intent zum nächsten transportieren zu können. Beim Start einer neuen Suche sollte hingegen jeglicher Eingangskontext gelöscht werden, um keine veralteten Angaben zu nutzen. Möchte der Nutzer all seine bisherigen Informationen zurücksetzen, muss der Ausgangskontext dieses Intents gelöscht werden.

Um im Backend entsprechend auf die jeweiligen Intents reagieren zu können, sollten Actions definiert werden. Die Bezeichnungen dafür können frei gewählt werden und geben Aufschluss über die auszuführenden Tätigkeiten des Intents. Die Actions werden im Webhook verwendet, um zwischen den einzelnen Funktionalitäten unterscheiden zu können. Außerdem werden für einzelne Intents Events definiert, welche die Möglichkeit

[11] https://cloud.google.com/

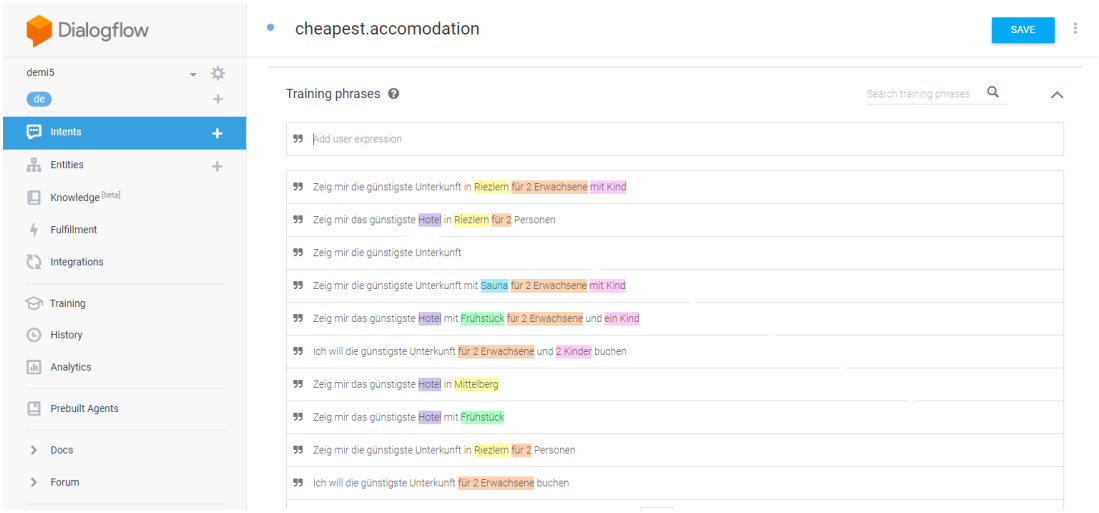


Abbildung 5.3: Trainingsphrasen innerhalb eines Intents zur Suche nach günstigen Unterkünften in Dialogflow.

bieten, den Intent ohne textuelle Eingaben des Nutzers im Backend auszulösen. Bei der Auswahl der bevorzugten Urlaubsdauer oder der Darstellung von alternativen Suchergebnissen werden die entsprechenden Intents über diese Events ausgelöst, sobald der Nutzer auf die dafür vorgesehenen Buttons klickt.

Des Weiteren benötigt jeder Intent Trainingsdaten, welche aus dem zuvor ausgearbeiteten Dialog verwendet werden. Innerhalb der Trainingsphrasen müssen Werte für die definierten Parameter verwendet werden, wie in Abbildung 5.3 ersichtlich. Dazu wird für jeden Parameter-Typ eine entsprechende Entity erstellt. Für den Buchungs-Bot werden Entities für die verschiedenen Unterkunftsarten, die Verpflegungstypen, die Orte der Urlaubsregion und für die Ausstattungen definiert. Alle gültigen Werte, die dieser Parameter innerhalb des Satzes annehmen kann, müssen in der Entity festgelegt werden. Dabei kann für jeden Wert eine Liste an Synonymen angeführt werden, wie Abbildung 5.4 zeigt. Hier sollten unterschiedliche Schreibweisen und gängige Abkürzungen von Wörtern beachtet werden. So kann der Nutzer beispielsweise die Anzahl der Reisenden mit *4 Personen* oder mit *vier Personen* angeben. Auch bei der gewünschten Ausstattung der Unterkunft sollten mehrere Synonyme definiert werden. Hier sucht der Nutzer möglicherweise nach einem Pool, welcher in der Datenbank aber als Hallenbad hinterlegt wurde.

Innerhalb der Entities wird Machine Learning grundsätzlich nicht verwendet. Nur Werte, die wortwörtlich in der Entity angegeben sind, können richtig zugeordnet werden. Für eine automatische Erweiterung der Liste kann allerdings eine zusätzliche Option aktiviert werden. Dadurch werden Werte, die nicht explizit definiert sind, automatisch erkannt und der entsprechenden Entity zugeordnet. Dies geschieht über Ähnlichkeiten in der Nutzeranfrage. Bei umfangreicheren Chatbots mit vielen gleichartigen Intents kann diese Ähnlichkeit allerdings nicht mehr eindeutig erkannt werden, und es kommt häufig zu Konflikten und ungewollten Zuordnungen der Werte.

Für die allgemeinen Parameter, wie das Anreise- und Abreisedatum, wird eine von

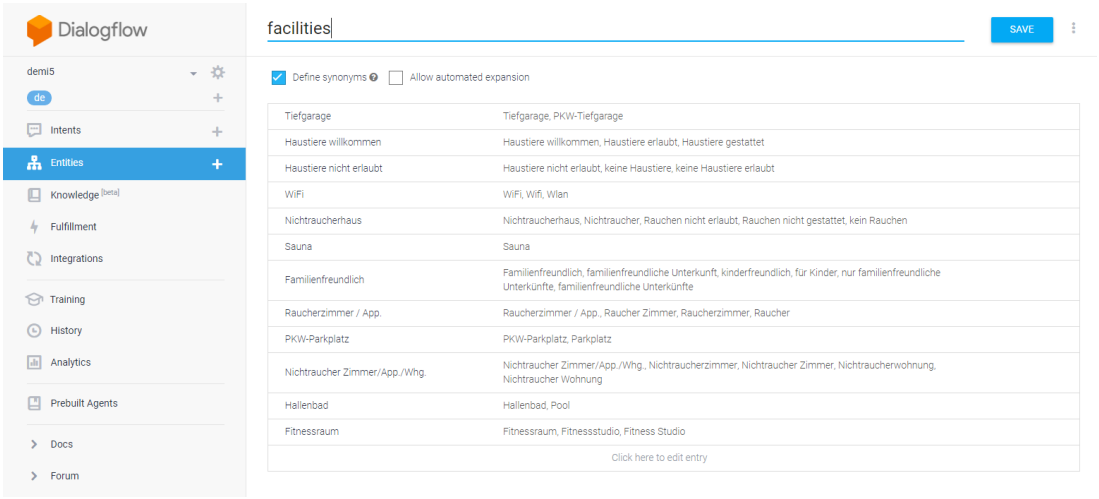


Abbildung 5.4: Darstellung von möglichen Werten und Synonymen einer Entity in Dialogflow.

Dialogflow definierte Entity genutzt. Diese verfügt bereits über alle gängigen Schreibweisen eines Datums und kann auch mit Eingaben wie *morgen*, *am Montag* oder *an Weihnachten* umgehen und diese einem gültigen Datumswert zuordnen.

Alle Entities, die innerhalb einer Trainingsphrase verwendet werden, stellen die Parameter der Action des jeweiligen Intents dar. Abbildung 5.5 zeigt alle möglichen Parameter für die Suche nach günstigen Unterkünften. Für jeden dieser Parameter muss ein einzigartiger Name gefunden werden. Außerdem sollte die Information bereitgestellt werden, ob der Parameter zwingend erforderlich ist oder nicht. Im Falle eines erforderlichen Parameters benötigt dieser eine oder mehrere Formulierungen, um den Nutzer auf die Eingabe dieses Wertes aufmerksam zu machen. Der Chatbot zeigt dem Nutzer so beispielsweise die Aufforderung, den gewünschten Urlaubsort anzugeben.

Parameter können zudem über einen Standardwert verfügen. Sollte der Nutzer den Parameter nicht explizit zur Verfügung stellen, wird dieser Wert ersatzweise herangezogen. Dabei kann auch auf Werte aus dem ausgelösten Event oder dem aktuellen Kontext zurückgegriffen werden. Im Projekt wird diese Maßnahme angewendet, wenn der Nutzer die Suchparameter ändert. Meist wird dabei nur ein Parameterwert angepasst. Alle anderen Parameter, denen kein neuer Wert zugewiesen wird, greifen auf den vorhandenen Wert im aktuellen Kontext zurück.

Zum Schluss benötigt der Intent eine Antwort auf die Anfrage des Nutzers. Diese kann statisch im jeweiligen Intent definiert werden und auf Werte aus der Benutzereingabe zurückgreifen. Außerdem kann für die Antwort auch der bereitgestellte Webhook verwendet werden. Dadurch können aktuelle Daten, wie zum Beispiel die gewünschten Unterkünfte oder Fotos abgefragt und dem Nutzer präsentiert werden. Für den Fall, dass der Webhook nicht erreichbar ist, werden die statischen Antworten als Fallback-Lösung verwendet.

Nachdem alle benötigten Intents und Entities erstellt wurden, muss der Chatbot mit den neuen Daten trainiert werden, damit diese in der Unterhaltung mit dem Nutzer richtig interpretiert werden können. Danach ist der Chatbot bereit zur Verwendung und

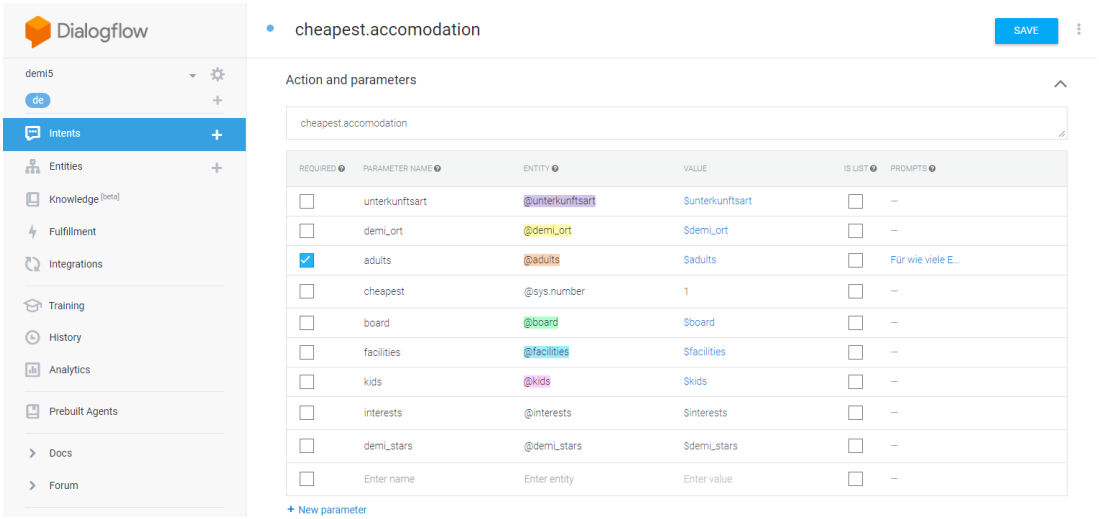


Abbildung 5.5: Übersicht über mögliche Parameter und deren zugehörigen Entities innerhalb eines Intents zur Suche nach günstigen Unterkünften.

kann entsprechend auf Anfragen reagieren.

Jede Unterhaltung wird in einer Session gespeichert und verfügt über eine eigene Session-ID. Alle Informationen sind grundsätzlich nur innerhalb dieser Session verfügbar, wird der Dialog für längere Zeit unterbrochen, läuft diese Session ab und jeglicher Kontext wird gelöscht. Der Nutzer muss eine erneute Anfrage starten, da der Chatbot nicht mehr auf bereits eingegebene Daten zurückgreifen kann.

5.2.3 Backend

Im Backend des Chatbots werden die empfangenen Daten verarbeitet und entsprechende Antworten an Dialogflow zurückgesendet. Dazu wird ein Webhook definiert, also der URL des Webservices. Dieser Webhook wird jedes Mal aufgerufen, sobald der entsprechende Intent ausgelöst wird und alle notwendigen Parameter vorhanden sind.

Dialogflow sendet einen HTTP POST-Request an den definierten Webservice. Zu diesem Zweck sollte der URL öffentlich zugänglich sein. Ist dies nicht möglich, können in Dialogflow zusätzliche Header und Authentifizierungen mitgeschickt werden. Da für jeden Chatbot nur ein Webhook angegeben werden kann, muss dieser je nach definierter Action die Verarbeitung der Daten koordinieren. Die empfangenen Daten weisen eine von Dialogflow definierte Struktur auf. Diese ist für jede Anfrage an den Webhook gleich, lediglich die Werte unterscheiden sich. In den Daten sind alle notwendigen Informationen über den aktuellen Stand der Konversation gespeichert. Diese Informationen beinhalten die verwendete Sprache, vorhandene Kontexte oder den ausgelösten Intent. Programm 5.1 zeigt die Struktur einer Anfrage über den Webhook.

Nachdem das Backend die Daten von Dialogflow empfangen hat, wird die JSON-kodierte Zeichenkette in eine PHP-Variable konvertiert. Anschließend wird diese Variable auf Gültigkeit überprüft, in dem die übergebene Action abgefragt wird. Ist dieser Wert nicht vorhanden, handelt es sich um keinen Aufruf von Dialogflow, und eine Feh-

Programm 5.1: Datenstruktur der Webhook-Anfrage.

```
id: "dca480a8-f5b6-4e5c-9106-beba1917b053"
lang: "de"
result:
    action: "booking.not.flexible"
    actionIncomplete: false
    contexts: [{...}]
    fulfillment: {speech: "", messages: [{...}]}
    metadata:
      intentId: "13119752-b246-48d7-8fd9-72e2de2cbedc"
      intentName: "check.booking.flexible - no"
      webhookForSlotFillingUsed: "false"
      webhookUsed: "true"
    parameters:
      adults: "zwei Personen"
      board: ""
      demi_ort: "egal"
      demi_stars: ""
      endDate: "2018-06-03"
      facilities: ""
      interests: ""
      kids: ""
      startDate: "2018-05-27"
      unterkunftsart: "Hotel"
    resolvedQuery: "egal"
    score: 1
    source: "agent"
    speech: ""
sessionId: "9e1a0aea-f4b9-92b1-ac8c-e7f427494be3"
status:
    code: 200
    errorType: "success"
timestamp: "2018-05-27T19:08:04.785Z"
```

lermeldung wird ausgegeben. Ist die Action jedoch vorhanden, wird der weitere Vorgang anhand einer Switch-Anweisung ermittelt. Dabei ist für jede Action eine andere Prozedur hinterlegt. Diese Prozedur kann zum einen das Abfragen der aktuellen Unterkunftsdaten sein, aber auch das Abfragen der bevorzugten Reisedauer oder das Einholen von Zusatzinformationen wie Fotos, Veranstaltungen oder Ausstattungen.

Für all diese Abfragen wird eine eigene API verwendet. Um diese nutzen zu können, müssen die von Dialogflow übergebenen Informationen in das entsprechende Format gebracht werden. Einerseits werden Start- und Enddatum der Reise vom ISO-8601-Format in einen Unix-Zeitstempel umgewandelt. Andererseits müssen jegliche Entities der entsprechenden Pimcore-Objekt-ID zugeordnet werden. Dazu werden Datenabfragen genutzt, die nach dem Typ und Wert der Entities suchen. Erst danach kann per HTTP-POST-Request eine Abfrage an die API getätigt werden. Diese liefert aktuelle Ergebnisse im JSON-Format zurück.

Nachdem ein API-Aufruf beendet wurde, werden die gelieferten Ergebnisse überprüft. Hierbei müssen zwei Szenarien beachtet werden. Einerseits ist es möglich, dass

die API die gewünschten Daten nicht findet, beziehungsweise keine aktuellen Ergebnisse liefern kann. In diesem Fall wird eine Fehlermeldung an Dialogflow zurückgesendet und der derzeitige Aufruf beendet.

Im Falle einer Abfrage bei der Suche nach buchbaren Unterkünften wird bei einem API-Aufruf ohne zutreffende Ergebnisse nicht sofort abgebrochen. Hier wird eine Alternativanfrage an die API getätigt. Dabei werden einige vom Nutzer gewählte Parameter gelöscht oder verändert, um abweichende Ersatzergebnisse zu erhalten. Ist diese Abfrage erfolgreich, wird eine Meldung zurückgegeben, welche den Nutzer über die erste erfolglose Suche, und über die gefundenen Ergebnisse mit den geänderten Parametern informiert.

Bei einem erfolgreichen API-Aufruf werden die Daten entsprechend weiterverarbeitet. Dabei wird vor allem auf die Anzahl der Ergebnisse geachtet. Ist diese zu groß, wird der Nutzer aufgefordert, seine aktuelle Auswahl weiter einzuschränken. Möchte er dies nicht, kann er den Chatbot auffordern, die Ergebnisse dennoch anzuzeigen.

Der vorletzte Schritt innerhalb des Webhooks ist die richtige Kodierung der Ergebnisse. Bereits im Backend muss die Struktur für die Anzeige der Daten im Frontend bestimmt werden. Diese ist von Integration zu Integration unterschiedlich. Wird der Chatbot in eine Messaging-Plattform integriert, muss sich der Entwickler an eine vorgegebene Struktur halten, damit die Plattform die Daten richtig verarbeiten und anzeigen kann. Wird der Chatbot in eine Website integriert, kann der Entwickler diese Struktur zum Teil selbst bestimmen. Lediglich einige vorgegebene Felder müssen in der Antwort an Dialogflow vorhanden sein. In diesem Projekt orientiert sich die Struktur an der Integration in Slack[12]. Diese ist sehr übersichtlich gehalten und eignet sich besonders für die Speicherung der verschiedenen Unterkunftsangebote. In Programm 5.2 werden die kodierten Ergebnisse einer solchen Unterkunftsanfrage dargestellt. Zeile 3 zeigt dabei den Wert, der für die weitere Verarbeitung im Frontend wichtig ist. Anhand dieses Wertes kann zwischen verschiedenen Szenarien unterschieden und dementsprechend darauf reagiert werden.

Sobald die Daten die richtige Struktur aufweisen, werden sie wieder in eine JSON-Zeichenkette konvertiert und an Dialogflow zurückgesendet. Danach wartet der Webhook auf die nächste Anfrage.

Außerdem wird das Backend dazu genutzt, die Entities des Chatbots aktuell zu halten. Dazu wird einmal pro Tag ein sogenannter Cronjob ausgeführt. Cronjobs starten Programme und Skripte zu einer bestimmten Uhrzeit automatisch am Server [54]. In diesem Projekt startet der Cronjob ein Exporter-Skript. Darin werden alle benötigten Objekte, die als Entities abgedeckt werden sollen aus der Datenbank geladen. Um den Chatbot aktuell zu halten, werden die unveröffentlichten und abgelaufenen Objekte ignoriert und nicht weiterverarbeitet. Aus den restlichen Objekten werden die Namen und mögliche Synonyme entnommen und in die benötigte JSON-Struktur gebracht. Danach wird diese Struktur an Dialogflow gesendet. Dazu wird der Entities-Endpoint der offiziellen Dialogflow-API V1 verwendet. Hierfür wird in den Pimcore-Einstellungen der Developer-Access-Token des Dialogflow-Projektes gespeichert. Dieser dient zur Autorisierung des Chatbots und muss bei jedem API-Aufruf an Dialogflow mitgesendet werden. Nach dem Export sind die aktuellen Entities im Chatbot verfügbar und können

[12] https://slack.com/intl/de

Programm 5.2: Auszug aus der Struktur der Daten, die an Dialogflow zurückgesendet werden.

```
fulfillment:
  data:
    action: "hotellist"
    attachments:
      0:
        address: "Leogang"
        board: "Frühstück"
        classification: []
        currency: "Euro"
        description: "Noch 12 Zimmer frei."
        description_sub: "Wählen Sie aus 4 Angeboten"
        detail_url: "/de/saalfelden/Hotel-PURADIES-Hotel-Leogang_ad-240678"
        facilities: ["PKW-Parkplatz", "WiFi"]
        id: 240678
        image_url: "/demi-image_240678_demi-chatbot_1527449100"
        meal: "Frühstück"
        price: "1.260,00"
        rating: {score: "4.8", text: "Ausgezeichnet"}
        stars: 4
        title: "Hotel PURADIES"
        type: "Hotel, Ferienwohnung / Appartement"
        vacancies: 12
    selection: "Deine aktuellen Kriterien: Hotel, zwei Personen, Anreisedatum:
    2018-05-27, Abreisedatum: 2018-06-03"
    text: "Folgende Unterkünfte treffen auf deine Kriterien zu. Wenn du eine
    Unterkunft auswählst, kannst du mir weitere Fragen dazu stellen."
```

entsprechend verwendet werden.

Das Exportieren der Entities zu Dialogflow macht vor allem bei der Suche nach Unterkünften Sinn, da sich die Parameter hier von Zeit zu Zeit ändern können. So ist es möglich, dass saisonabhängige Themen beziehungsweise Kategorien nur in gewissen Zeiträumen verfügbar sind. Folglich muss auch der Chatbot diese Zeiträume einhalten, um eine fehlerfreie Suche zu gewährleisten.

5.2.4 Frontend

Um den Chatbot innerhalb einer Website zu integrieren, muss der Entwickler auf der gewünschten Seite das Senden von Nuzteranfragen und das Empfangen von Antworten ermöglichen. In diesem Projekt wurde dafür eine vorhandene JavaScript-Bibliothek[13] verwendet und erweitert. Diese wird beim Aufruf der Seite geladen und verbindet sich per Client-Access-Token mit dem richtigen Dialogflow-Projekt.

Grundsätzlich verfügt die Website, auf der der Bot verwendet wird, über eine Art Chatfenster. Darin werden die gesamten Nachrichten der Konversation dargestellt. Über ein Textfeld kann der Nutzer mit dem Chatbot kommunizieren. Sobald die Enter-Taste gedrückt wird, wird ein Event ausgelöst und der aktuelle Wert des Textfeldes ausgele-

[13] https://github.com/dialogflow/dialogflow-javascript-client

Programm 5.3: Verwendung eines Promises zur Verarbeitung der Dialogflow-Antwort auf eine Benutzeranfrage.

```
1 sendText(value)
2   .then(function(response) {
3     result = response.result.fulfillment.speech;
4     setResponseJSON(response);
5     setResponseOnNode(result, responseNode);
6   })
7   .catch(function(err) {
8     setResponseJSON(err);
9     setResponseOnNode("An error occured.", responseNode);
10  });
```

sen. Über eine Funktion, die von der verwendeten JavaScript-Bibliothek zur Verfügung gestellt wird, wird der Text in ein JavaScript-Objekt gewandelt, per XHR-Request an Dialogflow gesendet und dort einem Intent zugeordnet. Um die zugehörige Antwort von Dialogflow erhalten zu können, wird ein sogenanntes Promise genutzt.

Promises sind JavaScript-Objekte, die bei asynchronen Operationen verwendet werden können. Sie werden häufig beim Laden externer Ressourcen eingesetzt. Das Promise repräsentiert dabei den Wert der Operation. Dieser Wert muss nicht zwingend sofort existieren, es wird aber garantiert, dass es zu einem Zeitpunkt in der Zukunft einen Wert geben wird. Das Promise wartet nicht auf das finale Ergebnis, sondern wird über einen Handler benachrichtigt, sobald die Operation abgeschlossen ist. In der Zwischenzeit befindet sich das Promise immer in einem der drei möglichen Zustände: pending (initialer Zustand), fulfilled (erfolgreich) oder rejected (gescheitert). Ist die Operation abgeschlossen, entweder erfolgreich oder fehlerhaft, wird das Promise als settled bezeichnet und darf seinen Zustand nicht mehr ändern. Ab diesem Zeitpunkt muss das Promise einen Wert besitzen, welcher ebenfalls nicht mehr veränderbar ist. Die then()-Methode dient als Handler und benötigt als Argument eine Funktion, die beim positiven Abschluss der Operation aufgerufen werden soll. Da sich das Promise zu diesem Zeitpunkt bereits im Zustand fulfilled befindet, kann innerhalb dieser Funktion der tatsächliche Wert des Promises verwendet werden. Für den Fall eines gescheiterten Promises wird die catch()-Methode verwendet. Über diesen Handler wird eine Funktion angegeben, die den Nutzer über die erfolglose Anfrage informiert und bei Bedarf eine Fehlermeldung protokolliert. Die beiden Methoden then() und catch() liefern wiederum ein Promise-Objekt zurück. Somit können sie aneinandergereiht werden, um mehrere voneinander abhängige, asynchrone Operationen zu ermöglichen [4].

Programm 5.3 zeigt den Aufruf der Funktion, die eine Benutzereingabe an Dialogflow sendet und ein Promise zurückliefert. Empfängt das Promise einen gültigen Wert von Dialogflow, wird der Handler in Zeile 2 darüber informiert und führt die übergebene anonyme Funktion zur Darstellung der Antwort aus. Tritt bei der Anfrage an Dialogflow ein Fehler auf oder kann keine Antwort empfangen werden, wird der Handler in Zeile 7 benachrichtigt und der Nutzer über eine Fehlermeldung darüber informiert.

Wie bereits erwähnt, kann die Struktur der Daten, die an das Frontend gesendet werden, bei der Webintegration zum Teil frei gewählt werden. Alle Daten, die über

Programm 5.4: Einfügen von verschachteltem HTML-Code in den DOM.

```
function createResponseNode() {
    var wrapper = document.createElement("div");
    wrapper.className = "chatbot-text-wrapper clearfix";
    var node = document.createElement("div");
    node.className = "chatbot-icon demi-icon-elements";
    wrapper.appendChild(node);
    var textNode = document.createElement("div");
    textNode.className = "left-align left chatbot-text";
    textNode.innerText = "...";
    wrapper.appendChild(textNode);
    resultDiv.appendChild(wrapper);
    return textNode;
  }
```

den Webhook an Dialogflow gesendet werden, kommen unverändert beim Frontend an. Diese Daten werden anhand des Action-Parameters, der bereits im Backend definiert wurde, analysiert. Somit kann zwischen den möglichen Prozeduren unterschieden und entsprechend reagiert werden.

Zu Beginn der Entwicklung wurden die empfangenen Daten in JavaScript verarbeitet und auf der Website angezeigt. Dazu wurden die benötigten HTML-Elemente in JavaScript generiert, mit Inhalt befüllt und anschließend in den DOM eingefügt, siehe Programm 5.4. Diese Vorgehensweise wurde im Laufe der Entwicklung immer unübersichtlicher und komplexer, da Repräsentation und Logik vermischt werden. Auch die richtige Reihenfolge, in der die HTML-Elemente in den DOM eingefügt werden müssen, wurde zunehmend schwieriger zu finden. Die Darstellung war nur schwer zu erweitern und zu warten, weshalb auf eine Darstellung mittels Templates umgestellt wurde. Um keine zusätzliche Bibliothek, wie etwa Mustache.js[14] oder Underscore.js[15] integrieren zu müssen, wird das neue HTML5-Element <template> verwendet [59]. Dazu werden als erstes alle HTML-Elemente aus der JavaScript-Datei entfernt und vor dem schließenden Body-Tag in der zugehörigen HTML-Datei eingefügt. Programm 5.5 zeigt die zu beachtende Struktur eines solchen Templates. Jedes dieser Elemente sollte über eine eindeutige ID verfügen, über die das Template im JavaScript-Code angesprochen und identifiziert werden kann. Zwischen dem öffnenden und dem schließenden Template-Tag kann beliebig viel verschachtelter HTML-Code eingefügt werden.

Die Template-Elemente werden beim Laden der Seite nicht sofort gerendert, sondern nur geparst und so zur späteren Verwendung im Dokument gespeichert [59]. Templates können in einem Dokument beliebig oft verwendet werden. Somit eignen sie sich besonders für repetitiven Inhalt, der nicht sofort, sondern erst beim Eintreten eines gewissen Ereignisses auf der Website sichtbar sein soll. Im Falle des Chatbots sind diese Ereignisse zum einen die Nutzereingaben. Sobald der Nutzer die Enter-Taste drückt, wird das Template per JavaScript aufgerufen, der Inhalt kopiert und in ein neues Node gespeichert. Danach kann dieses Node an einer beliebigen Stelle in den DOM eingefügt,

[14] https://mustache.github.io/
[15] http://underscorejs.org/

Programm 5.5: HTML-Struktur eines Templates.

```
1 <template id="response-template">
2     <div class="chatbot-text-wrapper clearfix">
3         <div class="chatbot-icon demi-icon demi-icon-elements"></div>
4         <div class="left-align left chatbot-text"></div>
5     </div>
6 </template>
```

Programm 5.6: Einfügen des befüllten Templates.

```
1 function createResponseNode() {
2     var tmpl = document.getElementById("response-template").content.cloneNode(true);
3     tmpl.querySelector(".chatbot-text").innerText = "...";
4     var textNode = tmpl.querySelector(".chatbot-text");
5     resultDiv.appendChild(tmpl);
6     return textNode;
7 }
```

und auf der Website dargestellt werden. Programm 5.6 zeigt den dafür notwendigen JavaScript-Code.

In diesem Projekt werden die Templates nicht nur für die Benutzereingaben verwendet, sondern auch für die Antworten des Chatbots. Diese können einerseits textueller Natur sein, andererseits aber auch aus Bildern, Listen, Links oder Kombinationen mehrerer Elemente bestehen.

Der Template-Tag wird von allen gängigen Browsern, ausgenommen Internet Explorer, unterstützt. Für diesen Browser muss eine Fallback-Lösung implementiert werden, da die Templates bereits beim Laden der Seite angezeigt werden.

Da die Templates nicht als Teil des Dokumentes angesehen werden, kann nicht mit herkömmlichen Methoden auf ihre Inhalte zugegriffen werden. Um diese Inhalte zu ändern, müssen sie per querySelector angesprochen werden, siehe Zeile 3 in Programm 5.6. Danach können alle üblichen Operationen durchgeführt, und die relevanten Informationen aus den empfangenen Daten eingefügt werden. So wird bei der Anzeige der Unterkunftsliste eine Schleife über die vorhandenen Daten aus dem Webhook gelegt. Für jedes dieser Elemente, also für jede Unterkunft, wird das zugehörige Template geladen, kopiert, mit den entsprechenden Werten befüllt und angezeigt. Dabei werden mehrere Templates kombiniert. Einerseits wird ein Template verwendet, das lediglich als Hülle für die gesamten Unterkünfte dient. Andererseits wird ein eigenes Template für die Unterkunft an sich verwendet. Und auch innerhalb der Unterkunft werden weitere Templates genutzt, zum Beispiel um die Anzahl der Sterne und die Bewertungen darzustellen.

Um dem Nutzer des Chatbots unnötiges Tippen zu ersparen, können neben offenen Fragen, auch solche mit einer begrenzten Anzahl an vordefinierten Antwortmöglichkeiten dargestellt werden. Dabei werden die einzelnen Antworten als Buttons angezeigt und mit einem Klick-Event hinterlegt. So kann dem Nutzer eine Art Hilfestellung gegeben werden. Für den Buchungs-Chatbot wurde diese Maßnahme bei der Auswahl des gewünschten Urlaubszeitraums verwendet. Wählt der Nutzer eine der gegebenen

Programm 5.7: Auslösen eines Dialogflow-Events im Frontend.

```
function setSubstitutes(params, node) {
  button.innerText = "alternative Ergebnisse anzeigen";
  button.onclick = function () {
    triggerEventWithParams("substitutes", params)
    .then(function (eventResponse) {
      hotelListOnNode(eventResponse.result.fulfillment.data.attachments,
    createResponseNode());
    }).catch(function (error) {
       console.log(error);
    });
  }
}
```

Möglichkeiten aus, wird das zugehörige Event ausgelöst und die gewählte Antwort an Dialogflow gesendet. Dazu wird eine bestehende Methode der JavaScript-Bibliothek umgeschrieben, die es ermöglicht, ein bestimmtes Event in Dialogflow auszulösen. In diesem Fall wird das Event für die Auswahl der bevorzugten Urlaubsdauer mit dem benötigten Parameter ausgelöst. Der Parameter der Methode, die Anzahl der Nächte, wird per JavaScript-Objekt übergeben und kann so von Dialogflow verwendet werden. Die Antwort auf dieses ausgelöste Event wird über ein Promise verarbeitet und bei Erhalt angezeigt.

Eine ähnliche Logik wird verwendet, wenn die Suche des Nutzers keine Ergebnisse liefert. Wie bereits beschrieben, wird in diesem Fall noch eine weitere, leicht veränderte Anfrage an die API gesendet. Werden dabei Ergebnisse gefunden, wird der Nutzer darüber informiert und hat die Möglichkeit, diese alternativen Ergebnisse über den Klick auf einen Button anzuzeigen. Dadurch wird wieder eine Anfrage an Dialogflow gesendet, die das zugehörige Event auslöst. Als Parameter werden in diesem Fall die geänderten Werte der API-Anfrage übergeben, welche die alten Werte in Dialogflow überschreiben. Mit diesen Werten wird dann eine erneute API-Anfrage gestartet, deren Ergebnisse über ein Promise verarbeitet werden. Dieser Vorgang wird in Programm 5.7 dargestellt.

Nachdem die Anfrage verarbeitet und dem Nutzer das Ergebnis angezeigt wurde, wartet das Frontend auf die nächste Benutzereingabe.

5.3 Trainingsdaten-Generator

Für die Implementierung des Trainingsdaten-Generators wird ebenfalls Pimcore verwendet. Dazu wird ein neuer Controller erstellt, welcher die zuvor definierten Aufgaben übernimmt. Wie bereits erwähnt, erfüllt der Generator drei Aufgaben. Zum einen können im Verwaltungsbereich die entsprechenden Domänen erstellt und bearbeitet werden. Außerdem werden im Analysebereich nicht zugeordnete Chatbot-Anfragen untersucht und können so Aufschluss über die Qualität des Chatbots geben. Im dritten Bereich werden aus den erstellten Domänen die Trainingsdaten generiert und im gewünschten Format exportiert. Abbildung 5.6 zeigt den schematischen Aufbau des Generators mit dem entsprechenden In- und Output für die drei definierten Bereiche.

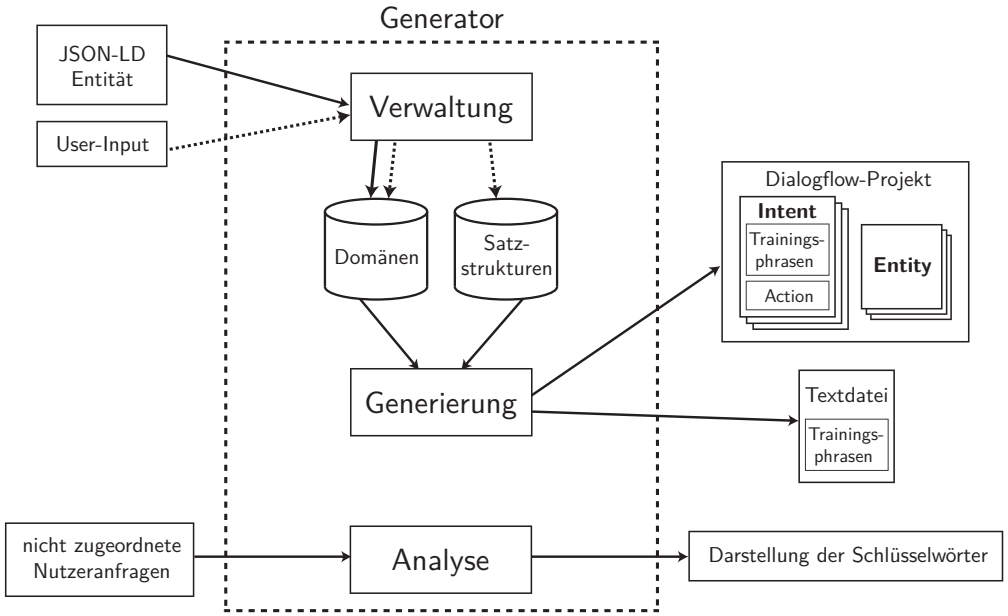


Abbildung 5.6: Schematischer Aufbau des Trainingsdaten-Generators mit In- und Output für die drei definierten Bereiche.

Im Verwaltungsbereich können neue Domänen hinzugefügt werden. Dies geschieht über ein Textfeld, in welches die entsprechende JSON-LD-Entität kopiert wird. Nach dem Absenden des Formulars werden vorhandene HTML-Tags aus der Entität entfernt. Danach wird der Code validiert, um die Korrektheit der Daten zu gewährleisten. Dazu wird auf spezifische JSON-LD-Merkmale geprüft. Handelt es sich um ein valides Objekt, wird dieses in eine PHP-Variable konvertiert. Danach wird anhand des Namens überprüft, ob die gegebene Domäne bereits existiert. Ist dies der Fall, wird sie nicht erneut eingefügt. Der Nutzer wird mittels Fehlermeldung über das Vorhandensein informiert und die Anfrage ist beendet.

Im Falle einer neuen Domäne werden alle Eigenschaften der JSON-Entität als Parameter in ein Array gespeichert. Zu jedem dieser Parameter werden Informationen abgefragt, die für die korrekte Grammatik bei der späteren Generierung der Trainingsdaten notwendig sind, siehe Abbildung 5.7. Diese Informationen beinhalten den konkreten Namen und das Geschlecht des Parameters. Außerdem muss definiert werden, ob der jeweilige Parameter zur Suche innerhalb der Domäne verwendet werden kann oder lediglich als Zusatzinformation dient. Für den Fall eines filterbaren Parameters werden weitere Angaben zur möglichen Satzstellung abgefragt. Abschließend gibt der Nutzer des Generators Beispiele für mögliche Werte dieses Parameters an. Auch für die Domäne selbst müssen korrekte Bezeichnungen in Ein- und Mehrzahl angeführt werden. Alle diese Daten werden dann per Formular abgeschickt und im Controller verarbeitet.

Für jede Domäne wird ein neues JavaScript-Objekt erstellt. Dieses Objekt verfügt über verschiedene Eigenschaften, die anhand der gegebenen Parameter mit Werten befüllt werden. So kann bei der Generierung der Trainingsdaten zwischen filterbaren, nicht

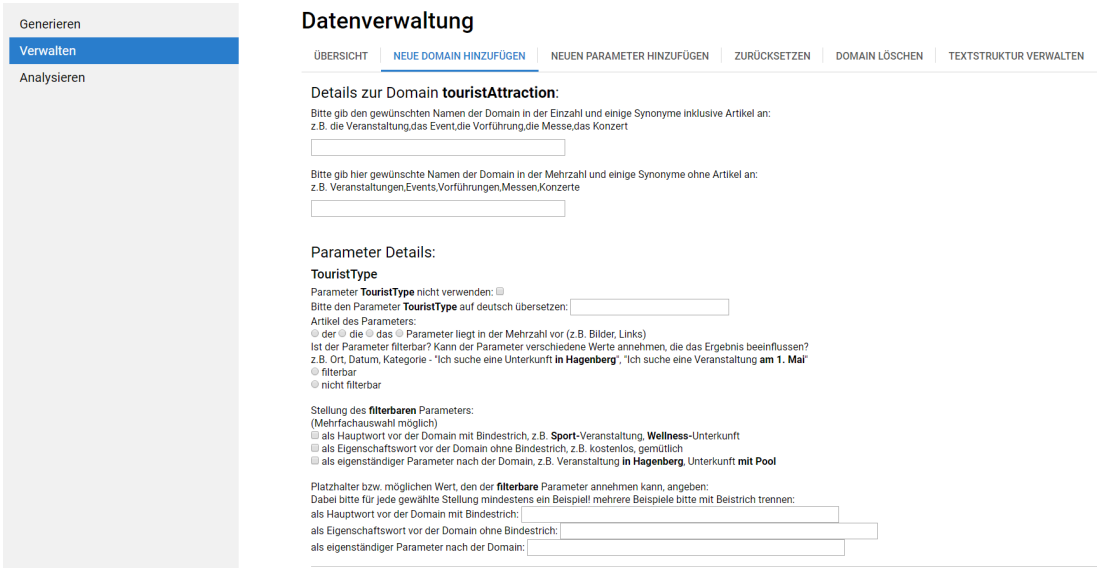


Abbildung 5.7: Benötigte Angaben zu den einzelnen Parametern beim Einfügen einer neuen Domäne in den Trainingsdaten-Generator.

Abbildung 5.8: Auswahl der gewünschten Parameter und des bevorzugten Export-Formats für die zuvor eingefügte Domäne.

filterbaren, weiblichen, männlichen und sächlichen Parametern unterschieden werden. All diese Angaben sind für die korrekte Grammatik des generierten Satzes wichtig. Auch die möglichen Positionen des jeweiligen Parameters innerhalb des Satzes werden als Array an der entsprechenden Stelle in das JavaScript-Objekt eingetragen. Weiters werden die Platzhalter für die Parameter als Array eingefügt. Sie werden ebenfalls anhand ihrer Position im Satz unterschieden. Somit existiert für jede mögliche Position der Platzhalter ein eigenes Array innerhalb des jeweiligen JavaScript-Objektes.

Nachdem alle Parameter an der richtigen Stelle in das JavaScript-Objekt eingefügt wurden, wird das Objekt in eine JSON-Zeichenkette konvertiert und in die entsprechende Datei geschrieben. Somit steht diese Domäne zur Generierung der Trainingsdaten bereit, wie Abbildung 5.8 zeigt. Auch das Einfügen eines neuen Parameters in eine vorhandene Domäne und das Bearbeiten der Strukturdaten erfolgt nach dem gleichen Prinzip. Über ein Formular werden alle benötigten Daten abgefragt und im Controller an der entsprechenden Stelle in die JSON-Datei geschrieben.

Um aus den zuvor erstellten Domänen die benötigten Trainingsdaten zu generieren, wird die in Kapitel 4.3 beschriebene Methode angewendet. Dazu wird eine weitere

Programm 5.8: Unterschiedliche Satztypen mit Platzhaltern.

```
"search": [
  "Ich suche [TYPE]",
  "Ich möchte [TYPE] [PARAM] sehen",
  "Kannst du mir [TYPE] empfehlen?",
  "Zeig mir [TYPE] [PARAM]"
],
"interaction": [
  "Gibt es [SEARCHTYPE] [ARTICLE] [PARAM]?",
  "Hat [SEARCHTYPE] [ARTICLE] [PARAM]?"
],
"adaption": [
  "Ändere [ARTICLE] [PARAM] auf [PLACEHOLDER]",
  "Kann ich [ARTICLE] [PARAM] [SEARCHTYPE] auf [PLACEHOLDER] ändern?",
  "Korrigiere [ARTICLE] [PARAM]"
]
```

JSON-Datei erstellt, in welcher die Struktur der Sätze definiert wird. Dazu werden unterschiedliche Satztypen erstellt, welche unter anderem zur Suche nach Ergebnissen dienen, aber auch zur Interaktion mit diesen Ergebnissen oder zur Änderung der Suchparameter. Innerhalb dieser Satztypen werden Platzhalter verwendet, welche beim Generieren mit den möglichen Parameterwerten befüllt werden. Auch die unterschiedlichen Bezeichnungen für die Domäne und deren zugehörige Artikel werden eingesetzt. Die Platzhalter sind in eckigen Klammern dargestellt, wie in Programm 5.8 zu sehen ist. In dieser JSON-Datei werden außerdem allgemeine Informationen zu den unterschiedlichen Artikeln und Fällen der Parameter-Typen gesammelt.

Nachdem die Struktur-Datei mit allen notwendigen Informationen befüllt wurde, können die Trainingsdaten generiert werden. Alle verfügbaren Domänen werden mit den zugehörigen Parametern aus der JSON-Datei geladen und dem Benutzer in der Weboberfläche des Trainingsdaten-Generators angezeigt, siehe Abbildung 5.8. Die gewünschte Domäne, alle selektierten Parameter und der bevorzugte Export-Typ werden über ein Formular an den Controller gesendet. Dort werden alle Eigenschaften der gewählten Domäne aus der JSON-Datei in Arrays geladen und eine neue Datei am Server erstellt, in die die generierten Sätze gespeichert werden.

Die einzelnen Sätze der Struktur-Datei werden ebenfalls in ein Array geladen und mit Hilfe einer Schleife durchlaufen. In jedem Durchlauf wird geprüft, ob der aktuelle Satz einen Platzhalter enthält. Ist dies der Fall, wird zunächst der Name der Domäne in den entsprechenden Platzhalter [TYPE] eingefügt. Dies geschieht für jeden möglichen Namen der Domäne, egal ob Ein- oder Mehrzahl. Sind im aktuellen Satz keine weiteren Platzhalter vorhanden, werden die befüllten Sätze in die Textdatei geschrieben. Befinden sich jedoch [PARAM]-Platzhalter im aktuellen Satz, müssen auch diese mit Werten befüllt und anschließend in die Textdatei gespeichert werden. Diese Vorgehensweise wird für alle vorhandenen Satzstrukturen und alle gewählten filterbaren Parameter wiederholt.

Klarerweise können die Chatbot-Nutzer in ihren Anfragen mehrere Parameter in einem Satz verwenden. Um zu garantieren, dass all diese Nutzeranfragen von Dialogflow

erkannt werden, sollten die Trainingsdaten möglichst viele unterschiedliche Parameter-Reihenfolgen aufweisen. Die Werte für die jeweiligen Parameter werden zunächst in der Reihenfolge, in der sie im Parameter-Array gespeichert sind, angeordnet und nacheinander in die Sätze integriert. Zugleich wird die aktuelle Reihenfolge in einem weiteren Array abgelegt. Dieser Vorgang wird nun für alle Parameter wiederholt. Allerdings wird die Reihenfolge der Parameter bei jedem Durchlauf zufällig neu angeordnet. Bevor die Parameterwerte in die Sätze eingefügt werden, wird überprüft, ob die generierte Reihenfolge bereits im Array gespeichert ist. Ist dies der Fall, werden die Parameter nochmals geordnet, so lange bis eine neue Reihenfolge auftritt. Anhand dieser werden die Parameterwerte schließlich in den Satz eingefügt und die aktuelle Reihenfolge in das Array gespeichert. So soll verhindert werden, dass sich bestimmte Reihenfolgen wiederholen und identische Sätze generiert werden.

Neben der Anzahl der Parameter muss auch deren Position innerhalb der Sätze beachtet werden. Wie bereits erwähnt, kann ein Parameter einerseits nach der Bezeichnung der Domäne stehen, zum Beispiel als *Unterkunft mit Pool*. Andererseits können die Parameter auch vor der Domäne auf zwei unterschiedliche Weisen eingefügt werden. Sowohl als Substantiv mit Bindestrich, wie bei *Selbstversorger-Unterkunft* oder als Adjektiv mit Leerzeichen, wie bei *barrierefreie Unterkunft*. Auch diese Parameter können wieder kombiniert werden. So werden in einen Satz beispielsweise alle Parameter, die vor der Domäne stehen eingefügt, in einen anderen alle Parameter nach der Domäne. Die Anzahl sollte dabei ebenfalls variabel sein, in einen Satz werden beispielsweise alle Parameter vor der Domäne eingefügt und ein zusätzlicher Wert nach der Domäne. Für alle Varianten wird mindestens ein Satz generiert, um möglichst viele unterschiedliche Reihenfolgen und Kombinationen der Parameter abzudecken.

Dabei muss allerdings auch beachtet werden, dass ein Parameter an allen drei möglichen Positionen exisitieren kann. Beispiele dafür sind *asiatisches Restaurant*, *Asia-Restaurant* und *Restaurant mit asiatischer Küche*. In diesem Fall sollten nicht alle Werte gleichzeitig im selben Satz vorkommen. Deshalb darf der aktuelle Parameter nicht in allen Parameter-Arrays vorhanden sein. Wird der Parameter als Eigenschaftswort vor der Domäne eingefügt, muss er temporär aus den Arrays für die Substantive entfernt werden. Im nächsten Umlauf wird der Parameter wieder ins Array eingefügt und als Substantiv vor der Domäne verwendet. Dazu wird er aus den restlichen beiden Arrays entfernt. So kann sichergestellt werden, dass der Parameter nur einmal pro Satz verwendet wird.

Neben den generierten Phrasen zur Suche innerhalb der Domäne, werden auch Trainingsdaten für Parameteränderungen und möglichen Fragen zu den Ergebnissen, beziehungsweise Interaktionen mit den Ergebnissen erstellt. Dabei wird das gleiche Prinzip angewendet. Lediglich bei nicht filterbaren Parametern muss ein Unterschied beachtet werden. So kann der Nutzer des Chatbots zum Beispiel die Telefonnummer der zuvor ausgewählten Unterkunft abfragen. Der Parameter *Telefonnummer* verfügt allerdings über keine eigenständigen Werte und wird deshalb direkt mit seinem Namen in den Satz eingefügt.

Alle generierten Sätze werden in die erstellte Textdatei am Server geschrieben. Diese Datei kann vom Entwickler sofort heruntergeladen und in den jeweiligen Dialogflow-Intent eingepflegt werden. Die verwendeten Parameter und zugehörigen Entities müssen in diesem Fall eigenhändig erstellt und verknüpft werden.

Zusätzlich ist auch der Export als vollständiges Dialogflow-Projekt möglich. Dafür werden alle benötigten Intents und Entities als JSON-Datei erstellt und per Zip-Archiv zur Verfügung gestellt.

Grundsätzlich sind die einzelnen Trainingssätze innerhalb der Textdatei, die auch als Grundlage für den Dialogflow-Export dient, bereits nach Intents geordnet. Diese werden durch Rauten gekennzeichnet, die Werte der verwendeten Parameter werden in eckige Klammern geschrieben. Dadurch können sie für den Dialogflow-Export leichter verarbeitet werden. Dazu werden zuerst alle Sätze der Datei ausgelesen und in ein Array gespeichert. Dieses Array wird dann in einer Schleife durchlaufen. Beginnt der aktuelle Satz mit einer Raute, handelt es sich um den Anfang eines Intents. Dementsprechend wird eine neue JSON-Datei erstellt, die den Namen dieses Intents trägt. Alle folgenden Sätze stellen die Trainingsdaten dieses Intents dar. Programm 5.9 zeigt die Struktur für einen Trainingsdatensatz. Darin ist zu sehen, dass der eigentliche Text von den verwendeten Parametern getrennt gespeichert wird. Um dies zu erreichen, werden alle Wörter des aktuellen Satzes in ein weiteres Array geladen, welches ebenfalls per Schleife durchlaufen wird. Dabei wird eine Variable für den Text angelegt und solange mit den einzelnen Wörtern befüllt, bis eine öffnende, eckige Klammer auftritt. Diese Klammer signalisiert den Beginn eines Parameters. Der aktuelle Wert der Text-Variable wird in die Trainingsdatenstruktur eingefügt, siehe Zeile 12 in Programm 5.9 und eine neue Parameter-Variable erstellt und mit dem aktuellen Wort befüllt.

Die folgenden Wörter werden so lange in diese Variable gespeichert, bis eine schließende, eckige Klammer auftritt, welche das Ende des Parameters signalisiert. Der aktuelle Wert der Variable wird wieder in die Trainingsdatenstruktur eingefügt, diesmal allerdings als Parameter, zu sehen in den Zeilen 15–18 in Programm 5.9. Dazu muss neben dem Wert des Parameters auch der entsprechende Typ angegeben werden, welcher die verwendete Entity definiert. Hierfür wird der Name des Parameterwertes herangezogen, welcher im Platzhalter-Array gespeichert ist. Nachdem der Parameter in die Datenstruktur eingefügt wurde, beginnt die beschriebene Prozedur von neuem. Bei generierten Phrasen ohne Parameter, wird der gesamte Satz als Text in die Datenstruktur eingefügt. Dieses Verfahren wird für alle Sätze der Textdatei und für jeden Intent wiederholt.

Zusätzlich zu den Intents müssen auch die benötigten Entities als JSON-Datei erstellt werden. Dazu werden alle verwendeten, filterbaren Parameter in einer Schleife durchlaufen. Für jeden dieser Parameter-Typen wird eine eigene JSON-Datei erstellt. Diese beinhaltet neben dem Namen des Parameters, welcher die Bezeichnung der Entity definiert, auch die angegebenen Platzhalter, die als Werte gespeichert werden. Die verwendete Struktur ist in Programm 5.10 ersichtlich. Für nicht filterbare Parameter wird keine Datei angelegt, da innerhalb der Trainingsdaten keine Werte für sie verwendet werden können.

Abschließend werden alle erstellten JSON-Dateien in das Zip-Archiv gespeichert und zum Download bereitgestellt. Danach werden die Daten wieder vom Server entfernt, um Speicherplatz zu sparen. Das heruntergeladene Zip-Archiv wird über die Dialogflow-Weboberfläche in das gewünschte Projekt importiert. Nach erfolgreichem Training können alle generierten Daten verwendet werden.

Der dritte Bereich des entwickelten Trainingsdaten-Generators dient zur Analyse des erstellten Chatbots. Hier werden jegliche Nutzereingaben, die vom Chatbot keinem

Programm 5.9: Struktur der generierten Trainingsdatensätze in Dialogflow.

```
"userSays": [
    {
      "data": [
        {
          "text": "Ich suche ein Restaurant "
        }
      ]
    },
    {
      "data": [
        {
          "text": "Ich suche ein "
        },
        {
          "alias": "Kochrichtung",
          "meta": "@Kochrichtung",
          "text": "asiatisches ",
          "userDefined": true
        },
        {
          "text": "Restaurant "
        }
      ]
    }
]
```

Programm 5.10: Struktur einer generierten Entity in Dialogflow.

```
{
   "name":"Kochrichtung",
   "isOverridable":true,
   "isEnum":false,
   "automatedExpansion":false,
   "entries":[
      {
         "value":"asiatisch"
      },
      {
         "value":"Asia-"
      },
      {
         "value":"Asia"
      },
      {
         "value":"mit asiatischer Küche"
      }
   ]
}
```

Intent zugeordnet werden konnten, angezeigt. Um diese Daten zu erhalten, wird der Dialogflow-Fallback-Intent im Backend überwacht und alle Eingaben in einer Textdatei am Server mitprotokolliert. Diese Eingaben werden anschließend über eine Github-Bibliothek namens *TextRank*[16] ausgewertet. Die häufigsten Nutzereingaben, die derzeit durch keinen vorhandenen Intent abgedeckt worden sind oder nicht in den aktuellen Trainingsdaten existieren, werden anhand ihrer Relevanz sortiert dargestellt. Dazu werden zuerst alle gespeicherten Anfragen aus der Server-Datei geladen und die darin vorhandenen Stoppwörter entfernt. Diese Wörter haben grundsätzlich keinen erheblichen Einfluss auf den Inhalt des Satzes und können somit bei der Extrahierung der Schlüsselwörter ignoriert werden. Meist handelt es sich dabei um die am häufigsten vorkommenden Wörter innerhalb einer Sprache, wie Artikel oder Konjunktionen [40]. Bei der Intent-Zuordnung in Dialogflow und bei der Suchmaschinen-Indexierung wird die Vorgehensweise der Stoppwort-Entfernung ebenfalls angewandt.

Nach der Entfernung der Stoppwörter wird einem Parser eine minimale Wortlänge übergeben. Alle Wörter, die diese Länge unterschreiten, werden ebenfalls aus dem Satz entfernt. Der restliche Satz wird analysiert und für jedes Wort eine Bewertung ermittelt. Diese Bewertung, ein Wert zwischen 0 und 1, gibt die Relevanz des Wortes innerhalb des Satzes an.

Die extrahierten Schlüsselwörter geben einerseits Aufschluss über die Bedürfnisse der Chatbot-Nutzer. Der Entwickler kann fehlenden Themenbereiche leichter erkennen und die dafür passende JSON-Struktur auswählen. Diese kann anschließend im Verwaltungsbereich als neue Domäne angelegt, und dazu passende Trainingsdaten generiert werden. Andererseits kann die Analyse auch Aufschluss über fehlende Parameter und somit fehlende Trainingsdaten in einer bereits bestehenden Domäne geben. Diese Parameter können ebenfalls im Verwaltungsbereich hinzugefügt werden.

Nach erfolgreicher Auswertung dieser Daten, und der entsprechenden Weiterentwicklung des Chatbots, können die gespeicherten Einträge gelöscht werden, um das Ergebnis bei der nächsten Analyse nicht zu verfälschen.

[16] https://github.com/DavidBelicza/PHP-Science-TextRank

Kapitel 6

Evaluierung

Um die Relevanz und die Qualität des Trainingsdaten-Generators zu testen, ist eine entsprechende Evaluierung notwendig. Anhand der erhaltenen Ergebnisse sollen Fehler in der Entwicklung behoben und Probleme in der Bedienung erkannt werden.

Dieses Kapitel beschreibt die gewählte Vorgehensweise der Evaluierung und die erhobenen Daten. Abschließend werden anhand der Ergebnisse mögliche Verbesserungsmöglichkeiten des Generators aufgezeigt.

6.1 Vorgehensweise

Grundsätzlich soll der implementierte Trainingsdaten-Generator mit der manuellen Erstellung von Chatbots verglichen werden, sowohl in Bezug auf die benötigte Zeit, als auch auf die Nutzerzufriedenheit bei der Anwendung. Um diese Daten zu erhalten, muss eine zweckmäßige Vorgehensweise gewählt werden.

Dazu wird eine Benutzerstudie in zwei Abschnitten durchgeführt. Im ersten Abschnitt werden Trainingsdaten zum einen manuell erarbeitet und zum anderen mit dem Generator erstellt. Danach werden aus diesen Trainingsdaten Chatbots entwickelt. Dies soll Aufschluss über mögliche Zeitunterschiede der unterschiedlichen Methoden geben.

Im zweiten Abschnitt der Studie wird die Qualität der erstellten Trainingsdaten ermittelt. Dazu werden die unterschiedlich entwickelten Chatbots von Anwendern getestet und anschließend bewertet.

6.1.1 Erstellen der Trainingsdaten

Zeit bildet einen wichtigen Faktor bei der Entwicklung von Chatbots. Um die Relevanz des Generators zu evaluieren, werden Testpersonen zuerst gebeten, manuell und ohne Zuhilfenahme des Generators Trainingssätze für einen einfachen Chatbot zu erstellen. Um das spätere Einpflegen in Dialogflow zu erleichtern, werden die Testpersonen angewiesen, die Trainingssätze nach Intention des Endnutzers zu gruppieren. Der gewünschte Chatbot soll die Funktionalität bieten, nach Sehenswürdigkeiten und zugehörigen Informationen zu suchen. Dabei wird eine Beispiel-Objektstruktur vorgegeben, die alle verfügbaren Dateneigenschaften aufzeigt. Auch die Information, welche Eigenschaften davon filterbar sind, wird gegeben. Die Zeit, die die Testpersonen zur Dialogfindung benötigen, wird genau mitprotokolliert. Anschließend werden die Trainingssätze, entspre-

chend der in Abschnitt 5.2.2 beschriebenen Vorgehensweise, in Dialogflow eingepflegt. Die dabei gemessene Dauer wird ebenfalls zur protokollierten Zeit addiert.

Die erstellten Trainingsdaten werden nicht von den Testpersonen selbst, sondern von einem versierten Dialogflow-Entwickler eingepflegt. Somit wird die gemessene Zeit nicht durch die unterschiedliche Erfahrung in der Verwendung von Dialogflow beeinflusst. Die erstellten Chatbots benötigen neben den Trainingsdaten, die die Anfragen der Nutzer darstellen, auch Antworten auf diese Anfragen, welche nicht von den Testpersonen entwickelt werden müssen. Da sie für alle Chatbots einheitlich sind, wird die benötigte Zeit für das Einpflegen dieser Antworten nicht zur Gesamtzeit addiert.

Die Aufgabe wird von mehreren technisch erfahrenen Web-Entwicklern durchgeführt. Aus den aufgezeichneten Zeiten wird die durchschnittlich benötigte Zeit zur Erstellung des Chatbots errechnet und zur Evaluierung herangezogen.

Um einen Referenzwert zu erhalten, wird die Aufgabe ein weiteres Mal mithilfe des Trainingsdaten-Generators wiederholt. Die gegebene Beispiel-Objektstruktur und die gewünschte Funktionalität des Chatbots bleiben dabei ident zur ersten Aufgabenstellung. Die Testperson muss den Generator zuerst mit den gewünschten Daten befüllen, um die Trainingsdaten daraus generieren zu können. Diese werden beim Generieren bereits in das notwendige JSON-Format gebracht, siehe Programm 5.9, um sie direkt in ein Dialogflow-Projekt importieren zu können. Auch hier wird die benötigte Zeit zum Erstellen der Trainingsdaten und zum Einpflegen in Dialogflow mitprotokolliert.

Nachdem alle Testpersonen die Aufgaben erfüllt haben, bewerten sie unterschiedliche Qualitätsmerkmale des Generators anhand einer Punkteskala von 0 bis 9. Diese Bewertungen beziehen sich unter anderem auf den Workflow, das Wording und die Zufriedenheit bei der Nutzung, und werden ebenso wie die protokollierte Zeit und die Bedienungsfehler bei der Generatornutzung für die Evaluierung herangezogen. Abschließend können die Testpersonen Verbesserungsvorschläge und Kritik äußern. Dadurch sollen fehlende Funktionalitäten oder Probleme in der Nutzeroberfläche ersichtlich werden.

6.1.2 Testen der Trainingsdaten

Im zweiten Teil der Studie wird die Qualität der zuvor erstellten Trainingsdaten überprüft. Die Trainingsdaten sollten im besten Fall die gesamten Nutzeranfragen abdecken und somit eine reibungslose Unterhaltung ermöglichen. Dazu nutzt eine weitere Gruppe von Testpersonen die entwickelten Chatbots. Anders als die Teilnehmer im ersten Abschnitt der Studie, besteht die zweite Gruppe von Testpersonen nicht nur aus Web-Entwicklern. Die Teilnehmer werden zufällig aus unterschiedlichen Berufsgruppen und Altersschichten ausgewählt und verfügen somit über abweichende Kenntnisse und Fähigkeiten in Bezug auf Chatbots.

Zu Beginn der Studie werden die Teilnehmer über den Ablauf des Testes informiert. Jeder Teilnehmer erhält zwei beziehungsweise drei Aufgabenstellungen, welche in Tabelle 6.1 dargestellt werden. Diese müssen mithilfe der im ersten Abschnitt der Studie entwickelten Chatbots bewältigt werden. Dabei testet ein Teilnehmer jeweils einen manuell erstellten und einen generierten Chatbot. Um die Ergebnisse der unterschiedlichen Erstellungsmethoden nicht zu beeinflussen, stammen beide Chatbots von ein und demselben Entwickler. Für den Fall eines fehlerhaft generierten Chatbots wird ein weiterer, optimal generierter Chatbot eines anderen Entwicklers von den Teilnehmern getestet.

Aufgabe	*Chatbot-Typ*	*Beschreibung*
A	2	Informiere dich über Museen in Salzburg, die für Kinder geeignet sind. Wähle ein Museum aus und suche die dazugehörige Telefonnummer.
B	3	Informiere dich über Museen, die günstige Eintrittspreise haben. Wähle ein Museum aus und suche die dazugehörigen Öffnungszeiten.
C	1	Informiere dich über Museen in Salzburg, die günstige Eintrittspreise haben. Wähle eine Sehenswürdigkeit aus und suche die dazugehörige E-Mail-Adresse.

Tabelle 6.1: Aufgabenstellung für einen Teilnehmer des zweiten Abschnittes der Studie.

Bei der Durchführung einer Aufgabe erhalten die Teilnehmer keinerlei Hilfestellungen. Die aktuelle Aufgabe gilt als beendet, wenn der Teilnehmer angibt, das geforderte Ergebnis gefunden zu haben, oder die Aufgabe vom Teilnehmer abgebrochen wird. Während jeder Aufgabe werden ausgewählte Daten gesammelt, welche Aufschluss über die Qualität des Chatbots geben sollen. Diese werden als Key Performance Indikatoren [23] bezeichnet und beziehen sich in dieser Studie ausschließlich auf den verwendeten Dialog. Das User Interface der Chatbots wird nicht analysiert.

Einer der wichtigsten Indikatoren ist die Genauigkeit des entwickelten Chatbots. Diese kann in nicht interpretierte und falsch interpretierte Anfragen eingeteilt werden. Nicht interpretierte Anfragen können keinem Intent zugeordnet werden und erhöhen die sogenannte Rate of Confusion. Diese beschreibt das Verhältnis zwischen nicht interpretierten und allen gesendeten Anfragen [23]. Eine hohe Rate of Confusion gibt Aufschluss über fehlende Trainingsdaten. Auf alle nicht interpretierten Anfragen erhält der Nutzer des Chatbots keine Antwort. Folglich ist eine geringe Rate of Confusion erstrebenswert.

Falsch interpretierte Anfragen können in zwei Gruppen eingeteilt werden. Vollständig falsch interpretiere Anfragen werden vom Chatbot einem Intent zugeordnet, welcher inhaltlich nicht zur Anfrage passt. Im Gegensatz dazu kann der Chatbot eine Anfrage zwar dem richtigen Intent zuordnen, aber nicht alle vom Nutzer eingegebenen Daten als Parameter interpretieren. Dem Nutzer werden Ergebnisse präsentiert, die nur teilweise auf seine Wünsche zutreffen. Das Problem von falsch interpretierten Anfragen tritt oft bei zu ähnlichen Trainingsdaten zwischen den Intents auf. Aber auch fehlende Trainingsdaten innerhalb eines Intents können dieses Verhalten des Chatbots verursachen. In dieser Arbeit wird das Verhältnis zwischen falsch interpretierten und allen gesendeten Anfragen als Rate of Failure bezeichnet. Für eine optimale Chatbot-Nutzung ist eine niedrige Rate of Failure von Vorteil.

Ein weiterer Key Performance Indikator ist die Anzahl an Konversationsschritten. Ein solcher Schritt besteht aus einer Nutzeranfrage und der dazugehörigen Antwort des Chatbots. Für den Nutzer kann es von großer Bedeutung sein, ein bestimmtes Ziel so schnell wie möglich zu erreichen. Eine Vielzahl an unnötigen Fragen und Interaktionen seitens des Chatbots ist in diesem Fall nicht sinnvoll. Deshalb sollte ein zielorientierter Chatbot in der Lage sein, das grundlegende Bedürfnis des Nutzers mit einer minimalen

Anzahl an Konversationsschritten erfüllen zu können. Alle Aufgabenstellungen in diesem Abschnitt der Studie, dargestellt in Tabelle 6.1, benötigen im geringsten Fall zwei Konversationsschritte [22].

Grundsätzlich existiert eine Vielzahl an weiterer Key Performance Indikatoren, die Aufschluss über die Qualität von Chatbots geben. Diese behandeln das Fehlermanagement, die Bindungsrate der Nutzer oder die Anzahl an Nutzerinteraktionen [23]. All diese Faktoren sind für die aktuelle Studie nicht von Bedeutung, oder können nicht erhoben werden, da die Daten über einen längeren Zeitraum gesammelt werden müssen.

Zusätzlich zur Aufzeichnung der messbaren Qualitätsmerkmale, wird jeder Chatbot nach Beendigung der Aufgabe von der Testperson anhand einer Punkteskala von 0 bis 9 bewertet. Die Kriterien der Bewertung reichen von der Nutzerzufriedenheit in Bezug auf benötigte Zeit und gefundener Information, über die Wiederverwendung, bis hin zu Problemen bei der Interaktion mit dem Chatbot. Maximal kann ein Chatbot bei dieser Bewertung 63 Punkte erreichen. Die erhobenen Daten werden ebenfalls zur Evaluierung herangezogen. Im Anschluss können die Testpersonen Verbesserungsvorschläge und Kritik zum getesteten Chatbot äußern. Dieses Prozedere wird für alle gegebenen Aufgabenstellungen wiederholt. Abschließend werden die getesteten Chatbots von den Teilnehmern nach Qualität geordnet, unabhängig der Punktebewertung. Dies soll zeigen, ob sich die vergebenen Punkte auch in der subjektiven Wahrnehmung der Teilnehmer wiederspiegeln.

Die beschriebene Vorgehensweise gilt für alle Teilnehmer des zweiten Abschnittes der Studie. Da allerdings zu erwarten ist, dass die Teilnehmer im Laufe der ersten Aufgabe gewisse Fertigkeiten im Umgang mit dem Chatbot erlernen, die sie bei den weiteren Aufgaben einsetzen können, kann dies die Ergebnisse der Studie beeinflussen. Um zu verhindern, dass ein bestimmter Chatbot dadurch tendenziell besser bewertet wird, wird die Reihenfolge der getesteten Chatbots nach jedem Teilnehmer geändert. Somit wird dieser ungewollte Lerneffekt bei der Nutzung auf alle zu testenden Chatbots verteilt. Die Reihenfolge der gegebenen Aufgabenstellungen hingegen bleibt über den gesamten Abschnitt der Studie gleich.

6.2 Ergebnisse

In diesem Abschnitt der Arbeit werden die erhobenen Daten der Studie analysiert und interpretiert. Dazu werden sowohl die Teilnehmerbewertungen, als auch die vorgestellten Key Performance Indikatoren herangezogen. Die Daten werden in unterschiedlichen Diagrammen visualisiert. Um die gleichzeitige Darstellung von zwei Variablen und deren Korrelation zueinander zu ermöglichen, werden Streudiagramme verwendet [35]. In den vorliegenden Daten können mehrere Werte ident sein. Diese werden im Diagramm übereinander gelagert und als ein Punkt dargestellt. Um die tatsächliche Anzahl an Werten abbilden zu können, wird in dieser Arbeit ein sogenannter Jitter verwendet. Dabei wird zu jedem vorliegenden Wert eine minimale Zufallszahl innerhalb eines definierten Bereiches addiert beziehungsweise subtrahiert. Alle Werte im Diagramm werden so leicht in ihrer Position verändert und überlappende Datenpunkte können einzeln wahrgenommen werden. Dabei ist allerdings zu beachten, dass somit keine exakte Darstellung der Daten mehr gegeben ist.

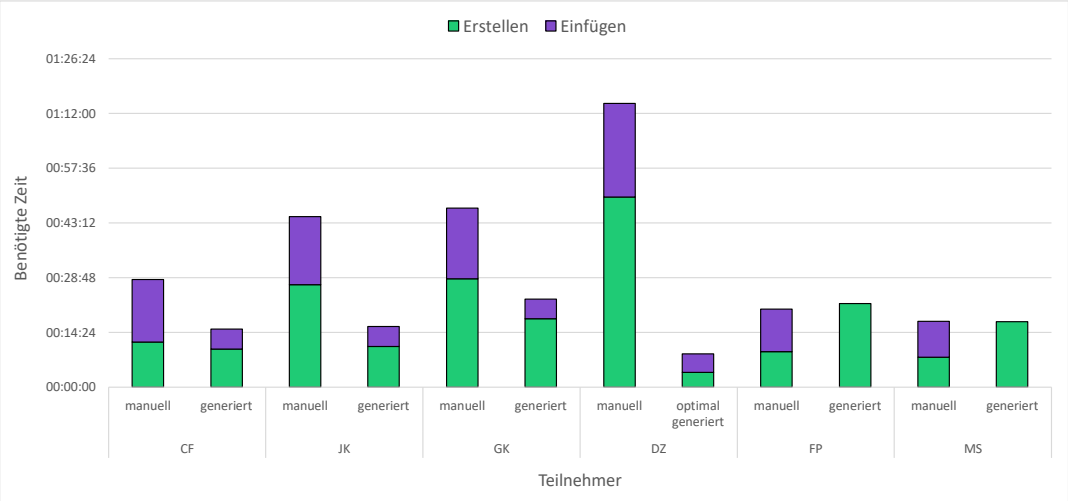


Abbildung 6.1: Zeitunterschiede bei der Erstellung der Trainingsdaten und dem anschließenden Einfügen in Dialogflow.

6.2.1 Erstellen der Trainingsdaten

Der erste Teil der Studie umfasst sechs Teilnehmer, die allesamt über ausreichende Chatbot-Kenntnisse verfügen, um die geforderten Trainingsdaten zu erstellen. Dennoch nutzten bisher nur zwei der Teilnehmer aktiv Chatbots, ein Teilnehmer entwickelte bereits einen Chatbot.

Die durchschnittlich benötigte Zeit zum manuellen Entwickeln der Trainingsdaten beträgt in dieser Studie 22,4 Minuten, siehe Abbildung 6.1. Dabei fiel die Vorgehensweise durchwegs unterschiedlich aus. Erfahrene Teilnehmer entwickelten eine Logik zur effizienten Erstellung der Trainingsdaten. Zuerst wurden Trainingsdaten für die allgemeine Suche nach Sehenswürdigkeiten formuliert, danach die konkrete Suche mit Parametern ausgearbeitet. Hierfür wurden die Parameter an unterschiedlichen Stellen und in abweichenden Reihenfolgen platziert. Diese Variationen wurden von vier der sechs Testpersonen erstellt und bilden den jeweils größten Teil der resultierenden Trainingsdaten. Andere Teilnehmer setzten wiederum auf eine einfachere Methode. Die gegebene Datenstruktur wurde Parameter für Parameter abgearbeitet und Fragen dazu formuliert. Diese Fragen wurden kopiert und im Wortlaut leicht verändert. Allerdings wurden dabei keinerlei Parameter-Kombinationen erstellt. Unabhängig der unterschiedlichen Vorgehensweisen wurden im Durchschnitt 77 Trainingsdatensätze formuliert. Dabei stuften alle Testpersonen die manuelle Entwicklung der Trainingsdaten als aufwendig und komplex ein.

Für die zweite Aufgabenstellung wurden die Trainingsdaten mithilfe des implementierten Generators erstellt. Die zum Generieren benötigte Zeit liegt bei durchschnittlich 13,7 Minuten und ist somit deutlich geringer als bei der manuellen Erstellung der Trainingsdaten. Der schnellste Teilnehmer konnte die Aufgabe in knapp unter 4 Minuten erledigen. Die Anzahl der generierten Trainingsdatensätze liegt bei durchschnittlich 636.

Grundsätzlich konnten alle Teilnehmer die geforderte Aufgabe erfüllen und Trai-

ningsdaten generieren. Dennoch waren teilweise erhebliche Probleme bei der Bedienung des Generators erkennbar. Vor allem der erste Arbeitsschritt der Aufgabe, das Hinzufügen einer neuen Domäne, war trotz vorhandener Hilfestellungen im Generator zu komplex. Die notwendigen Arbeitsschritte waren laut den Teilnehmern nicht intuitiv genug, daher wurden in diesem Abschnitt die meisten Bedienungsfehler begangen. Nur ein Teilnehmer konnte die Trainingsdaten fehlerfrei generieren. Am häufigsten wurden die benötigten Daten in einem falschen Format angegeben. Dies resultiert laut den Teilnehmern aus nicht ausreichend konkretisierten Anweisungen innerhalb des Generators. In der abschließenden Bewertung durch die Teilnehmer erhielt der Workflow beim Einfügen einer neuen Domäne deshalb nur durchschnittlich nur 4,5 von 9 möglichen Punkten.

Da alle notwendigen Werte zum Generieren der Trainingsdaten für jeden einzelnen Parameter definiert werden müssen, wurde die Zufriedenheit der Dauer von den Teilnehmern im Durchschnitt mit 5,2 Punkten bewertet. Weitaus weniger Probleme zeigten sich beim Vorgang des Generierens der Trainingsdaten aus der zuvor eingefügten Domäne. Der Arbeitsablauf wurde dabei mit durchschnittlich 7,7 von 9 möglichen Punkten bewertet, die Zufriedenheit bei der Dauer des Generierens mit 8,8 Punkten.

Abschließend wurden die Trainingsdaten in Dialogflow-Chatbots eingefügt. Bei den manuell erstellten Trainingsdaten wurden durchschnittlich 16,4 Minuten benötigt. Diese Zeit ergibt sich aus dem Anlegen der unterschiedlichen Intents, dem Einfügen der Trainingsdaten in die jeweiligen Intents und dem Anlegen aller benötigten Entities und deren Befüllung. Außerdem müssen voneinander abhängige Intents über Eingangs- und Ausgangskontexte verbunden werden. Da die generierten Trainingsdaten bereits als fertiges Dialogflow-Projekt exportiert werden können, werden bei dieser Methode lediglich Eingangs- und Ausgangskontexte gesetzt, und Synonyme für die bereits erstellten Entities eingefügt. Dafür wurden durchschnittlich fünf Minuten benötigt.

Für das Entwickeln der Chatbots ergibt sich in dieser Studie somit eine Gesamtzeit von durchschnittlich 38,8 Minuten bei manuell erstellten Trainingsdaten, und durchschnittlich 15,8 Minuten bei generierten Trainingsdaten. Das entspricht einer Zeitersparnis von 59 %. Bei der weiteren Unterteilung in optimal generierte Trainingsdaten kann eine Zeitersparnis von 77 % gegenüber den manuell erstellten Trainingsdaten erreicht werden. Die Gegenüberstellung der Werte ist in Abbildung 6.1 ersichtlich. Diese Zeitersparnis ist nicht nur auf die automatische Generierung der Sätze zurückzuführen, sondern auch auf die bereits erstellten und befüllten Intents und Entities. Somit muss der Entwickler die Trainingsdaten nicht von Hand eingeben. Die von den Teilnehmern im ersten Abschnitt der Studie als mühevoll und zeitintensiv beschriebene Entwicklung aller möglichen Variationen und Reihenfolgen der unterschiedlichen Parameter wird vom Generator problemlos erledigt.

Zwei der Teilnehmer hielten sich bei der Benutzung des Generators wiederholt nicht an die vorgegebene Datenstruktur. Auch die schriftlichen Anweisungen und Hilfestellungen des Generators wurden großteils nicht gelesen und die benötigten Daten dadurch nicht fehlerfrei angegeben. Folglich konnten keine grammatikalisch korrekten Trainingsdaten generiert werden. Der Import in Dialogflow war aufgrund einer unzureichenden JSON-Syntax, welche aus der lückenhaften Dateneingabe resultiert, nicht möglich. Infolgedessen wurden diese Trainingsdaten in keinem zu testenden Chatbot verwendet.

Aufgrund der geringen Teilnehmeranzahl im ersten Abschnitt der Studie lassen sich keine repräsentativen Aussagen über den Zusammenhang von Erfahrung in der Entwick-

lung von Chatbots und der benötigten Zeit zum Erstellen der Trainingsdaten treffen. Dennoch kann davon ausgegangen werden, dass vor allem bei der manuellen Erstellung der Trainingsdaten erfahrene Entwickler im Vorteil sind, da sie die üblichen Formulierungen und wichtigen Konzepte bereits kennen.

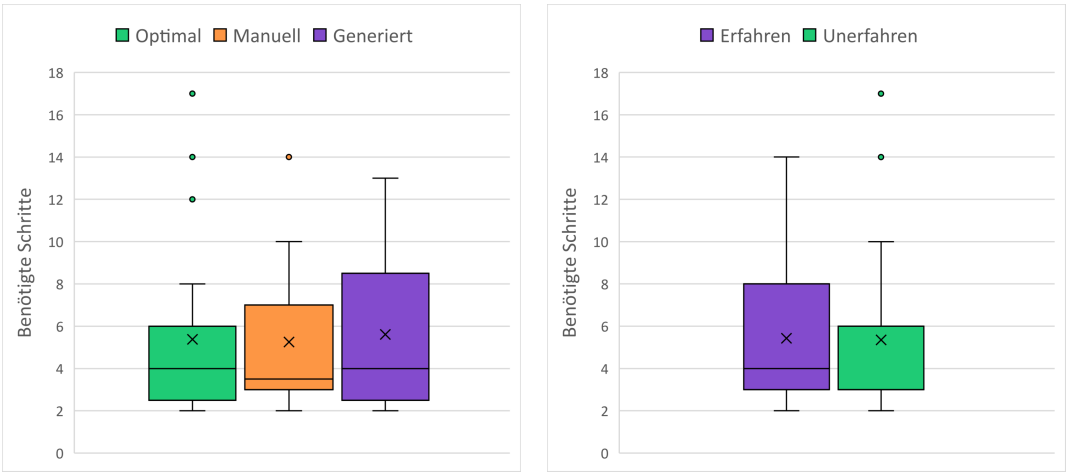
6.2.2 Testen der Trainingsdaten

Für den zweiten Abschnitt der Studie wurden 20 Teilnehmer zufällig ausgewählt, von denen 13 bereits über Erfahrung in der Nutzung von Chatbots verfügen.

Wie bereits erwähnt kann jede der drei gestellten Aufgaben in minimal zwei Schritten bewältigt werden. Die durchschnittlich benötigte Anzahl bei allen erfolgreich abgeschlossenen Testversuchen beträgt 5,4 Schritte. Betrachtet man diesen Durchschnittswert für die unterschiedlichen Erstellungsmethoden, ergibt sich ein ähnliches Bild. Aufgrund der Teilnehmeranzahl in diesem Abschnitt der Studie sind die erhobenen Werte allerdings von Ausreißern beeinflusst. Um einen besseren Überblick über die Verteilung der Daten zu erhalten und die unterschiedlichen Datensätze vergleichen zu können, werden die Daten in einem Boxplot-Diagramm visualisiert. Dieses zeigt in Abbildung 6.2 (a) die benötigten Schritte der erfolgreich abgeschlossenen Testversuche aller Teilnehmer nach der Erstellungsmethode sortiert. Dabei zeigen sich bei Median und Mittelwert keine gravierenden Unterschiede zwischen den Methoden. 75 % der Teilnehmer konnten die einzelnen Aufgaben bei der Verwendung von optimal generierten Trainingsdaten in maximal sechs Schritten erledigen. Bei den manuell erstellten liegt dieser Bereich bei maximal sieben Schritten, bei den fehlerhaft generierten bei knapp über acht Schritten. Unterteilt man die Teilnehmer in erfahren und unerfahren, ergibt sich unabhängig der Erstellungsmethode, dass 75 % der unerfahrenen Teilnehmer eine maximale Schrittanzahl von sechs aufweisen, die erfahrenen acht. Dieses durchaus erstaunliche Ergebnis wiederholt sich auch im höheren Median bei den Schritten der erfahrenen Teilnehmer, siehe Abbildung 6.2 (b). Eine mögliche Erklärung dafür ist die unterschiedliche Vorgehensweise bei den Testversuchen. Unerfahrene Teilnehmer nutzten großteils einfache und oftmals vorhersehbare Anfragen, während erfahrene Teilnehmer die Chatbots häufig mit unüblichen und komplexen Anfragen fordern und an ihre Grenzen bringen wollten.

In Abbildung 6.2 (a) ist deutlich zu erkennen, dass Chatbots mit optimal generierten Trainingsdaten die meisten Ausreißer aufweisen. Diese werden im Diagramm als Punkte dargestellt. Betrachtet man Abbildung 6.2 (b) so zeigt sich, dass diese Ausreißer auf unerfahrene Teilnehmer zurückzuführen sind.

Wie sich die benötigte Anzahl an Schritten auf die Bewertung der Teilnehmer auswirkt, zeigt Abbildung 6.3. Dabei ist eine negative Korrelation zwischen diesen beiden Variablen erkennbar, das bedeutet, je weniger Schritte der Teilnehmer benötigt, desto höher ist die anschließende Bewertung [56]. Der Korrelationskoeffizient, welcher die Stärke des linearen Zusammenhanges der beiden Variablen zwischen -1 und $+1$ angibt, beträgt in diesem Fall $-0,63$. Bei Werten unter $-0,6$ und über $+0,6$ wird von einem statistisch erkennbaren linearen Zusammenhang gesprochen [57]. Zwischen den unterschiedlichen Erstellungsmethoden sind dabei keine relevanten Unterschiede erkennbar. Die Erfahrung der Testpersonen beeinflusst das Ergebnis ebenfalls nicht, wie in Abbildung 6.4 ersichtlich. Dennoch sollte bei diesem Indikator auch beachtet werden, dass eine möglichst niedrige Schrittanzahl nicht das primäre Ziel der Chatbot-Nutzer sein



(a) (b)

Abbildung 6.2: Anzahl der benötigten Schritte anhand der unterschiedlichen Erstellungsmethoden (a) und sortiert nach der Erfahrung der Teilnehmer (b).

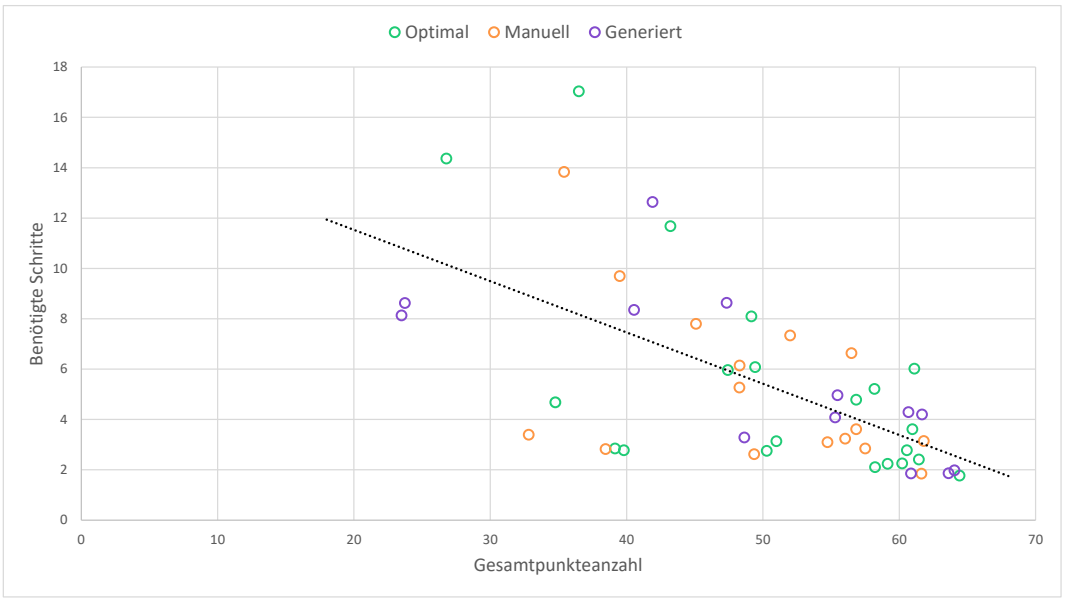


Abbildung 6.3: Auswirkung der benötigten Schritte auf die Bewertung durch die Teilnehmer, unterteilt anhand der Erstellungsmethode.

muss.

Weiters wird der Zusammenhang zwischen der benötigten Zeit zum Erfüllen der Aufgabe und der anschließenden Bewertung untersucht. Abbildung 6.5 zeigt die hierfür erhobenen Daten. Dabei sind zwischen den unterschiedlichen Erstellungsmethoden und der Erfahrung der Teilnehmer, siehe Abbildung 6.6, keine auffallenden Abweichungen

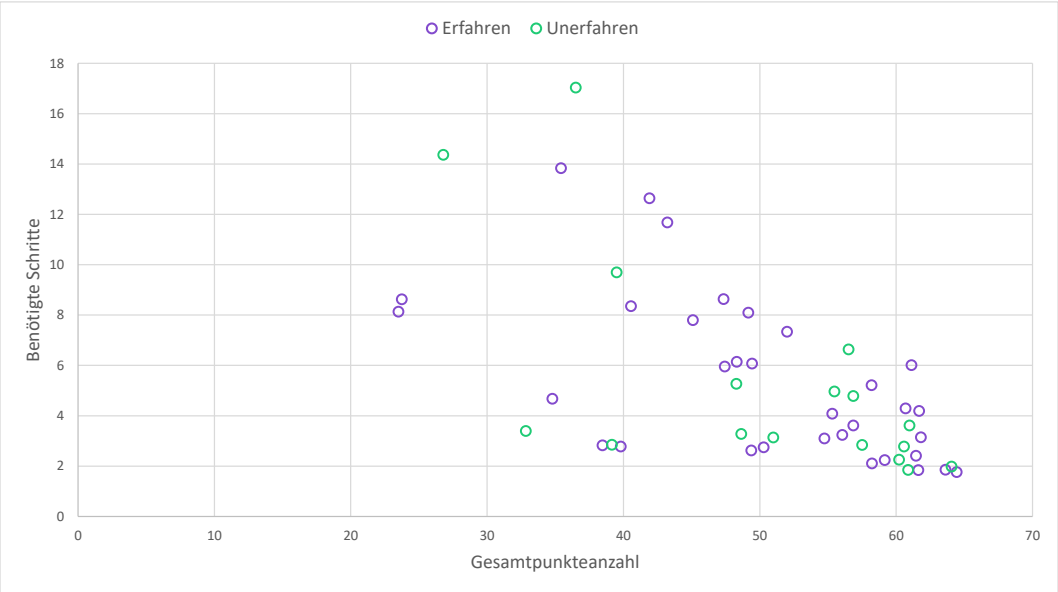


Abbildung 6.4: Auswirkung der benötigten Schritte auf die Bewertung, aufgeteilt in erfahrene und unerfahrene Teilnehmer.

erkennbar. Obwohl die Darstellung in Abbildung 6.5 eine negative Korrelation zwischen benötigter Zeit und zugehöriger Bewertung vermuten lässt, beträgt der Korrelationskoeffizient in diesem Fall lediglich $-0,48$. Somit kann von keinem statistisch erkennbaren Zusammenhang der beiden Variablen gesprochen werden. Dies ist auch auf die Tatsache zurückzuführen, dass die benötigte Zeit in erster Linie von den Testpersonen, und nicht von der Qualität der Trainingsdaten abhängig ist. Denn je länger die Nutzeranfrage an den Chatbot ist, desto mehr Zeit wird zum Eintippen in das Chatfenster benötigt. Auch bei kurzen Anfragen kann die benötigte Zeit durch langsames Tippen erhöht werden. Somit ergibt sich aus der benötigten Zeit zum Erfüllen der Aufgabe keinerlei konkreter Aufschluss über die Qualität der Trainingsdaten. Der relativ hohe Korrelationskoeffizient kann in diesem Fall von der benötigten Anzahl an Konversationsschritten beeinflusst werden, denn je mehr Schritte, desto mehr Zeit wird benötigt.

Wie bereits erwähnt, ist einer der wichtigsten Key Performance Indikatoren die Genauigkeit des Chatbots. Diese wird in nicht zugeordnete und falsch zugeordnete Anfragen eingeteilt. Die erhobenen Daten der durchgeführten Tests zeigen, dass die durchschnittliche Rate of Confusion bei den Chatbots mit manuell erstellten Trainingsdaten 36,5 % beträgt. Dies liegt vor allem an den fehlenden Variationen der Parameter und oftmals an nicht vorhandenen Konzepten wie Begrüßung, Hilfestellung oder Interaktion mit den Ergebnissen. Betrachtet man die gesamten Chatbots, welche mit generierten Trainingsdaten entwickelt wurden, so beträgt die Rate of Confusion hier nur 15 %. Teilt man diese Chatbots noch in fehlerhaft und optimal generierte Trainingsdaten, bleiben in dieser Studie lediglich 13 % aller Anfragen, die bei Chatbots mit optimal generierten Trainingsdaten nicht zugeordnet werden konnten. Dies entspricht einem Unterschied von 23 % von manuell erstellten zu optimal generierten Trainingsdaten. Zurückzuführen ist

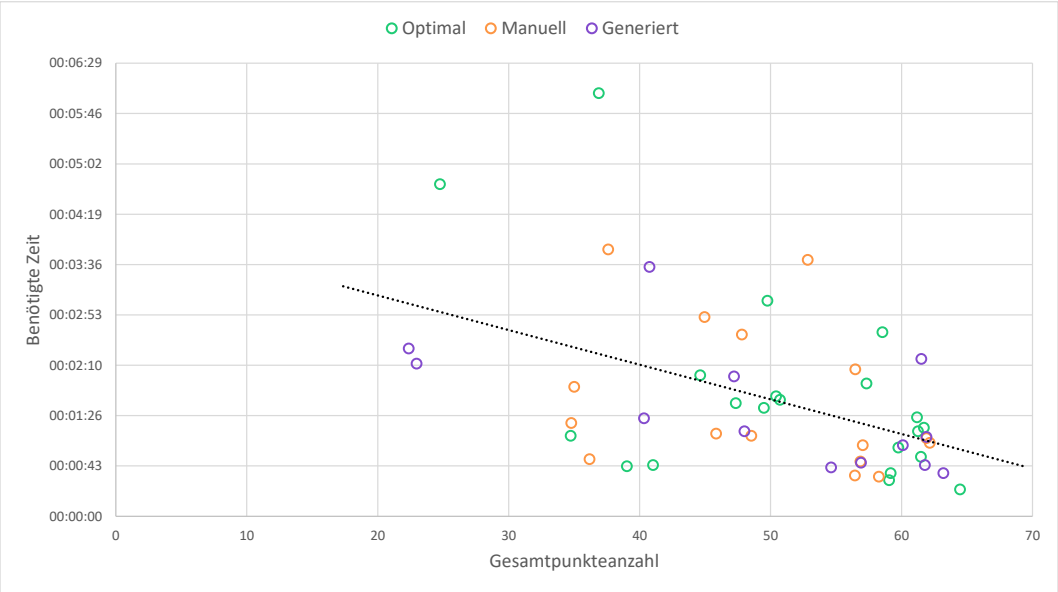


Abbildung 6.5: Auswirkung der benötigten Zeit auf die Bewertung, aufgeteilt nach den unterschiedlichen Erstellungsmethoden.

diese Differenz auch auf die Unterschiede in der Anzahl der verfügbaren Trainingsdaten. Je mehr Trainingsdaten im Chatbot vorhanden, desto größer ist die Wahrscheinlichkeit, dass eine Anfrage zugeordnet werden kann. Dies ist durch einen Korrelationskoeffizienten von $-0,4$ zwar nicht statistisch erkennbar, dennoch zeigt sich innerhalb der Testversuche ein Zusammenhang dieser beiden Variablen.

Die Auswirkung der Rate of Confusion auf die Punktebewertung wird in Abbildung 6.7 dargestellt. Dabei zeigt sich, dass zwischen diesen beiden Variablen eine negative Korrelation besteht. Je niedriger die Rate of Confusion, desto höher die anschließende Bewertung. Der Korrelationskoeffizient beträgt in diesem Fall allerdings nur $-0,4$, somit kann nicht von einem statistisch erkennbaren linearen Zusammenhang gesprochen werden. Dessen ungeachtet zeigt sich allerdings, dass Chatbots mit einer niedrigen Rate of Confusion meist zwischen 55 und 63 Punkten erreichen konnten. Der errechnete Korrelationskoeffizient kann auch in diesem Fall von der benötigten Anzahl an Konversationsschritten beeinflusst werden.

Dass die Zuordnung einer Anfrage nicht immer korrekt sein muss, zeigt die Rate of Failure eines Chatbots. In dieser Studie beträgt diese durchschnittlich 24,6 % bei fehlerhaft generierten Trainingsdaten. Auch bei der optimal generierten Variante zeigt sich eine Rate of Failure von 20,6 %. Bei Chatbots, die auf manuell erstellten Trainingsdaten beruhen, ergibt sich ein Wert von durchschnittlich 17,1 %. Zurückzuführen ist dieser Unterschied auf die Ähnlichkeit der generierten Trainingsdaten zwischen den unterschiedlichen Intents. Durch die Logik beim Generieren sieht die grundsätzliche Satzstellung und Formulierung bei allen Intents gleich aus. Somit fällt dem Chatbot die Zuordnung einer Anfrage schwerer. Bei den manuell erstellten Trainingsdaten hingegen ist diese Ähnlichkeit in einem geringeren Ausmaß gegeben und der Chatbot kann bes-

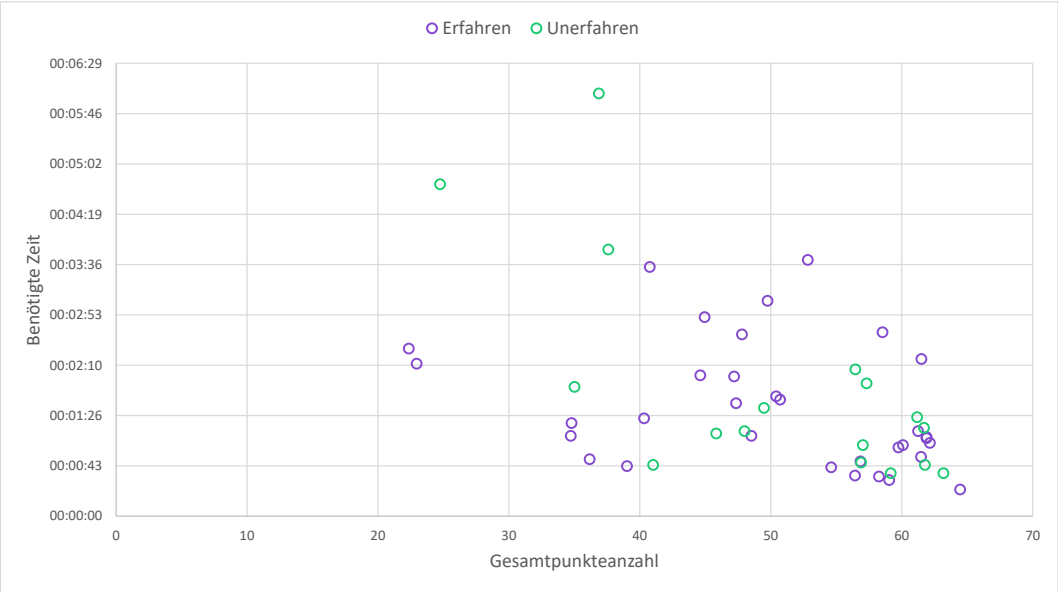


Abbildung 6.6: Auswirkung der benötigten Zeit auf die Bewertung, aufgeteilt in erfahrene und unerfahrene Teilnehmer.

ser zwischen den einzelnen Intents unterscheiden. Die erhobenen Daten zeigen, dass die Anzahl der Trainingsdaten dabei in keinem statistischen Zusammenhang mit der Rate of Failure steht, da nur ein Korrelationskoeffizient von 0,09 berechnet werden konnte.

Zusammenfassend konnten die manuell erstellten Trainingsdaten in diesem Bereich überzeugen. Allerdings ist die Rate of Failure während der Benutzung des Chatbots großteils kaum ersichtlich. Sofern der Chatbot eine inhaltlich passende Antwort auf die Anfrage bereitstellt, kann der Nutzer davon ausgehen, dass seine Anfrage richtig interpretiert wurde und die gezeigten Ergebnisse seinen Wünschen entsprechen. Welche Auswirkung die Rate of Failure auf die Teilnehmerbewertung hat, ist in Abbildung 6.8 erkennbar. Dabei ist zu sehen, dass sich ein konzentrierter Bereich von Datenpunkten bei einer Rate of Confusion von 0 % und einer hohen Punkteanzahl befindet. Der Korrelationskoeffizient von $-0,4$ zeigt allerdings, dass kein statistisch erkennbarer Zusammenhang zwischen diesen beiden Variablen besteht. Auch hier kann die Anzahl an Konversationsschritten diesen Zusammenhang beeinflussen.

Die Rate of Failure ist wichtig für die Wiederverwendung des Chatbots. Sucht ein Nutzer nach barrierefreien Museen und der Chatbot zeigt fälschlicherweise nicht barrierefreie Museen an, entsteht die Frustration beim Anwender nicht, wie bei der Rate of Confusion bereits während der Nutzung des Chatbots, sondern erst nach dem Kauf einer Eintrittskarte oder beim tatsächlichen Besuch des Museums.

Verringert werden kann die Rate of Failure durch aussagekräftige und abwechslungsreiche Trainingsdaten. Diese Möglichkeit der abwechslungsreichen Trainingsdaten ist beim entwickelten Generator aufgrund der verwendeten Logik nicht gegeben. Um den Nutzer dennoch über die falsch interpretierte Anfrage zu informieren, können die aktuellen Kriterien, die der Chatbot extrahieren konnte, aufgelistet werden. Somit kann

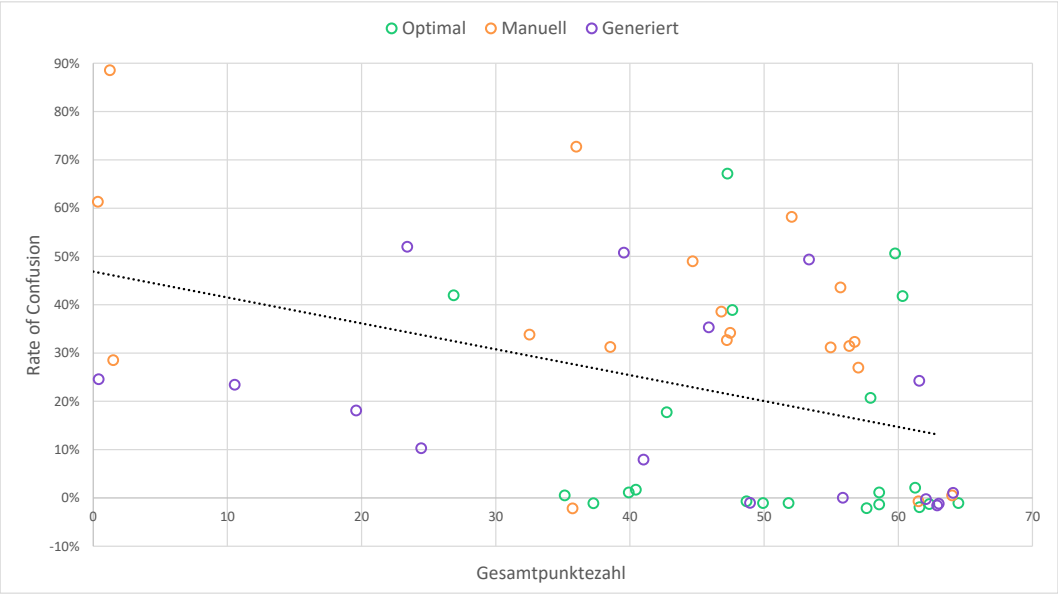


Abbildung 6.7: Zusammenhang zwischen Rate of Confusion und Teilnehmer-Bewertung, unterteilt in die drei unterschiedlichen Erstellungsmethoden.

verhindert werden, dass der Nutzer falsche Informationen des Chatbots übernimmt.

Die Bewertung der Teilnehmer, sortiert nach den unterschiedlichen Entwicklungsmethoden, ist in Abbildung 6.9 (a) anhand eines Boxplot-Diagrammes dargestellt. Dabei werden auch die abgebrochenen Testversuche miteinbezogen. Hierbei lässt sich erkennen, dass die Mediane der drei Methoden annähernd ident sind. Werden die Werte allerdings genauer betrachtet, können teilweise große Unterschiede abgelesen werden. 50 % der Testversuche, in denen Chatbots, die auf optimal generierten Trainingsdaten beruhen, verwendet wurden, wurden mit mindestens 51 von 63 möglichen Punkten bewertet. 25 % der Testversuche erreichten einen Wert von mindestens 60 Punkten. Die optimalen Chatbots verfügen außerdem über die geringste Spannweite der drei Methoden. Alle Bewertungen der Teilnehmer liegen zwischen 21 und 63 Punkten. Dadurch ergibt sich ein Mittelwert von 51 Punkten. Die manuell erstellten Trainingsdaten weisen die größte Spannweite auf, hier liegen alle Datensätze zwischen 0 und 63 Punkten. 50 % der Testversuche wurden mit maximal 48 Punkten bewertet. Drei davon erhielten lediglich null Punkte. Diese werden in Abbildung 6.9 zur besseren Sichtbarkeit gejittert und somit übereinander dargestellt. Die niedrigen Bewertungen beeinflussen auch den Mittelwert, welcher bei knapp unter 42 Punkten liegt. Bei den fehlerhaft generierten Trainingsdaten zeigt sich ein ähnliches Bild. Ein getesteter Chatbot wurde mit nur einem Punkt bewertet, somit umfasst die Spannweite 62 Punkte, weshalb der Mittelwert ebenfalls nur 42 Punkte beträgt. Grundsätzlich lässt sich allerdings erkennen, dass die Bewertungen bei dieser Erstellungsmethode stark variieren. 25 % der Testversuche wurden mit maximal 23 Punkten bewertet. Dennoch erhielten weitere 25 % eine Gesamtpunktezahl von mindestens 61,5. Die restlichen 50 % liegen zwischen diesen beiden Grenzen.

Somit lässt sich zusammenfassend sagen, dass optimal generierte Trainingsdaten

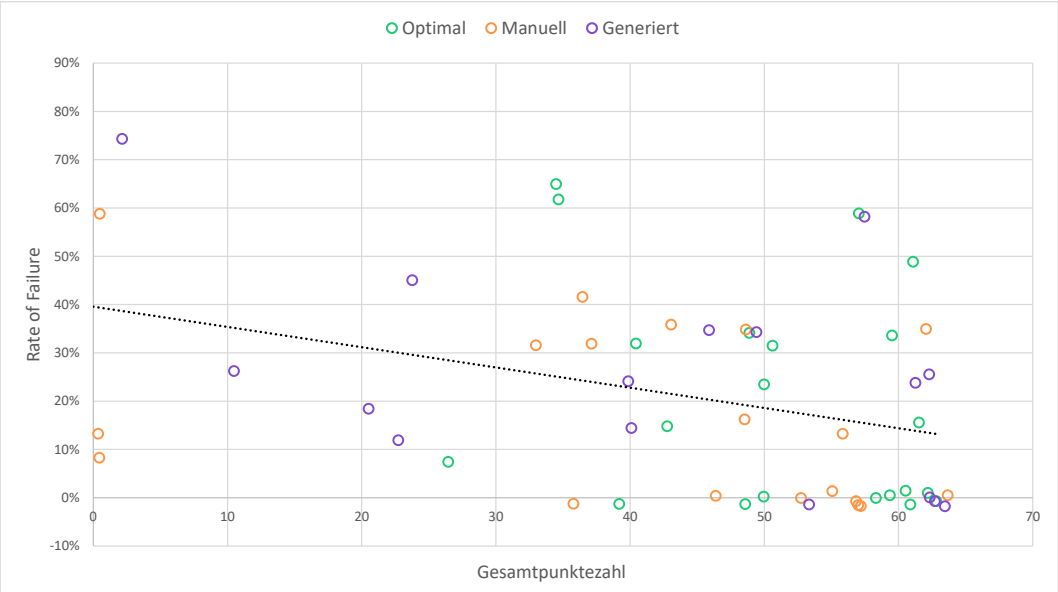


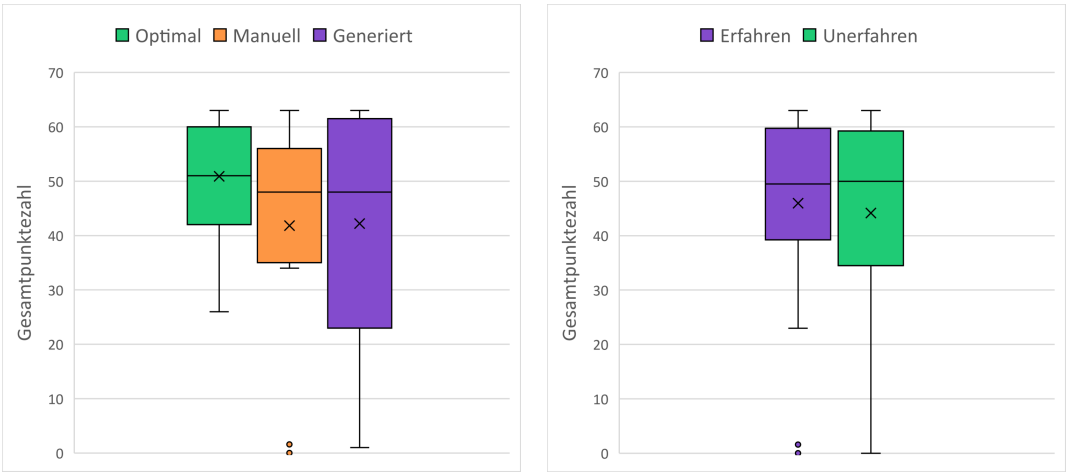
Abbildung 6.8: Zusammenhang zwischen Rate of Failure und Teilnehmer-Bewertung, aufgeteilt in die drei unterschiedlichen Erstellungsmethoden.

im Mittel die meisten Punkte erreichen konnten. Im Vergleich zu manuell erstellten Trainingsdaten erhielten die optimal generierten durchschnittlich 22 % mehr Punkte. Dies kann einerseits auf die große Anzahl an Trainingsdaten mit variierenden Parameter-Kombinationen zurückgeführt werden. Andererseits spielen in dieser Studie laut den Teilnehmern auch das Vorhandensein von Begrüßungen und Hilfestellungen sowie die geringe Rate of Confusion eine wichtige Rolle bei der Bewertung.

Vergleicht man die erstellten Chatbots der Teilnehmer vom ersten Abschnitt der Studie in Abbildung 6.10, so lassen sich dabei ebenfalls Unterschiede der beiden Methoden bei den Höchstbewertungen erkennen. Mit einer Ausnahme konnten alle generierten Chatbots mindestens einmal 63 Punkte erreichen. Die zugehörigen manuell erstellten Chatbots wurden mit maximal 58 Punkten bewertet. Im Durchschnitt ergibt das eine Steigerung der Höchstbewertungen von 8 %. Dies ist auf die zuvor beschriebenen Vorteile des Generators zurückzuführen.

Dass sich die Erfahrung der Teilnehmer kaum auf die Bewertung der Chatbots auswirkt, wird in Abbildung 6.9 (b) gezeigt. In beiden Fällen, also erfahren und unerfahren, wurden 50 % der Testversuche mit mindestens 49,5 beziehungsweise 50 Punkten bewertet. Die Spannweite ist ebenfalls ident. Der einzige Unterschied ist im unteren Quartil der Werte erkennbar. Hier wurden 25 % der Testversuche von erfahrenen Teilnehmern mit maximal 39,25 Punkten bewertet. Die maximale Punktebewertung im unteren Quartil der unerfahrenen Teilnehmer beträgt hingegen nur 34,5.

Zwischen der Anzahl an erstellten Trainingsdaten und der Punktebewertung durch die Teilnehmer besteht kein statistisch erkennbarer Zusammenhang, da der Korrelationskoeffizient in diesem Fall nur 0,2 beträgt. Auch für den Zusammenhang zwischen der benötigten Zeit zum Erstellen der Trainingsdaten und den Punktebewertungen kann ein



(a) (b)

Abbildung 6.9: Darstellung der Punktebewertungen aufgeteilt in die unterschiedlichen Erstellungsmethoden (a) und aufgeteilt in erfahrene und unerfahrene Teilnehmer (b).

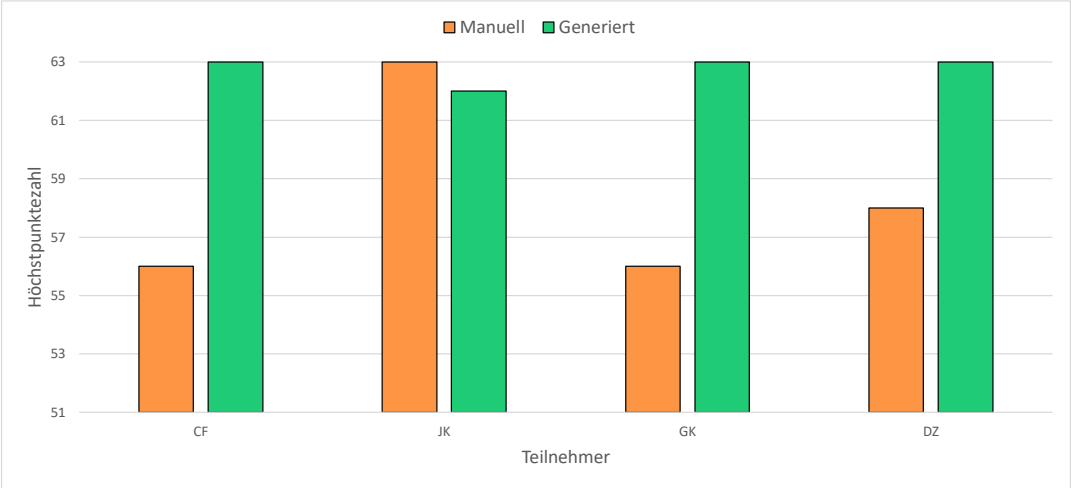


Abbildung 6.10: Vergleich der maximalen Punkteanzahlen der Chatbots mit unterschiedlichen Erstellungsmethoden.

Korrelationskoeffizient von $-0,2$ berechnet werden. Erstaunlicherweise hat die Anzahl an Bedienungsfehlern beim Benutzen des Generators ebenfalls keinen statistisch nachweisbaren Einfluss auf die Bewertung der Teilnehmer, und somit auf die resultierende Qualität der getesteten Trainingsdaten.

In diesem Abschnitt der Studie konnten die Teilnehmer die geforderten Aufgaben zu jedem Zeitpunkt abbrechen, um die Tests so realitätsgetreu wie möglich zu gestalten. Von den 20 Teilnehmern, denen Aufgabe A, siehe Tabelle 6.1, gestellt wurde, beendeten diese nur 19 erfolgreich. Ein Teilnehmer, welcher einen fehlerhaft generierten Chatbot

testete, brach die Aufgabe nach 3,5 Minuten ab, da der Chatbot bei 10 Konversationsschritten bereits eine Rate of Confusion von 40 % aufwies. Somit konnte der Chatbot fast jede zweite Anfrage nicht interpretieren. In diesen Anfragen wurde häufig nach Informationen zu einem präsentierten Ergebnis gesucht. Die gewählte Formulierung konnte vom Chatbot allerdings keinem Intent zugeordnet werden.

Für Aufgabe B konnten zwei manuell erstellte und zwei fehlerhaft generierte Chatbots die Anforderungen nicht erfüllen und wurden abgebrochen. Somit konnten bei 20 Teilnehmern nur 80 % der Testversuche abgeschlossen werden. Zum Abbruchzeitpunkt wiesen die manuell erstellten Chatbots eine durchschnittliche Rate of Confusion von 59 % auf, die fehlerhaft generierten lediglich 25 %.

Aufgabe C wurde aufgrund der gewählten Vorgehensweise von 16 Testpersonen durchgeführt und nur einmal abgebrochen. Der dabei verwendete Chatbot wurde manuell erstellt und konnte 60 % der gesendeten Anfragen nicht interpretieren. Die Testperson konnte keine verwertbaren Ergebnisse erhalten, da der Chatbot bereits die gewählten Formulierungen bei der initialen Suche nach Museen keinem bestehenden Intent zuordnen konnte. Auch die von der Testperson angefragten Hilfestellungen konnte der Chatbot nicht liefern. Aus diesem Grund wurde die Aufgabe nach 2,3 Minuten und nur drei richtig zugeordneten Anfragen abgebrochen.

Grundsätzlich konnten die getesteten Chatbots in den Aufgaben B und C weniger Nutzeranfragen zuordnen als in Aufgabe A. Dies liegt vor allem an der Formulierung dieser Anfragen. Viele Teilnehmer ließen sich von der Aufgabenstellung inspirieren und übernahmen diese großteils in ihre Anfrage. In den meisten Fällen waren diese Ausdrücke allerdings nicht in den Trainingsdaten vorhanden.

Darüber hinaus wurde in den Anfragen zu den Aufgaben B und C der Wunsch nach bestimmten Funktionalitäten der Chatbots deutlich. Eine Vielzahl an Teilnehmern bat um eine Art Preisvergleich der Museen oder forderte den Chatbot auf, die Ergebnisse anhand des Preises zu sortieren.

Aus den gesammelten Daten ergibt sich, dass bei 50 % aller abgebrochenen Aufgaben manuell erstellte Chatbots getestet wurden. Die restlichen 50 % der nicht beendeten Testversuche wurden mit fehlerhaft generierten Chatbots durchgeführt. Einzig der optimal generierte Chatbot konnte in allen Aufgaben der Studie bestehen. Generell ergibt sich daraus eine Abbruchrate von nur 5 %.

Von den insgesamt sechs abgebrochenen Testversuchen, wurde die Hälfte von erfahrenen Chatbot-Nutzern durchgeführt. Folglich spielt die Erfahrung der Teilnehmer in Bezug auf die Verwendung von Chatbots bei den abgebrochenen Testversuchen in dieser Studie keine Rolle.

Die Reihenfolge der getesteten Chatbots wurde nach jedem Teilnehmer verändert, um zu verhindern, dass bereits erworbenes Wissen in den darauffolgenden Aufgaben eingesetzt werden kann und somit die Studienergebnisse beeinflusst werden. Durch diese Maßnahme ist eine gleichmäßige Verteilung der Aufgaben über die unterschiedlich erstellten Chatbots gegeben. Abbildung 6.11 zeigt, dass Chatbots bei Aufgabe A mit durchschnittlich 52 Punkten die besten Bewertungen erhielten. 50 % der Teilnehmer vergaben eine Bewertung von mindestens 57,5 Punkten. Kein Teilnehmer bewertete die Chatbots in Aufgabe A mit weniger als 21 Punkten. Die Chatbots, die zur Bewältigung von Aufgabe B verwendet wurden, konnten durchschnittlich nur 41,3 Punkte erreichen. 25 % der Werte liegen unter 35 Punkten. Obwohl Aufgabe B und C ein ähnliches

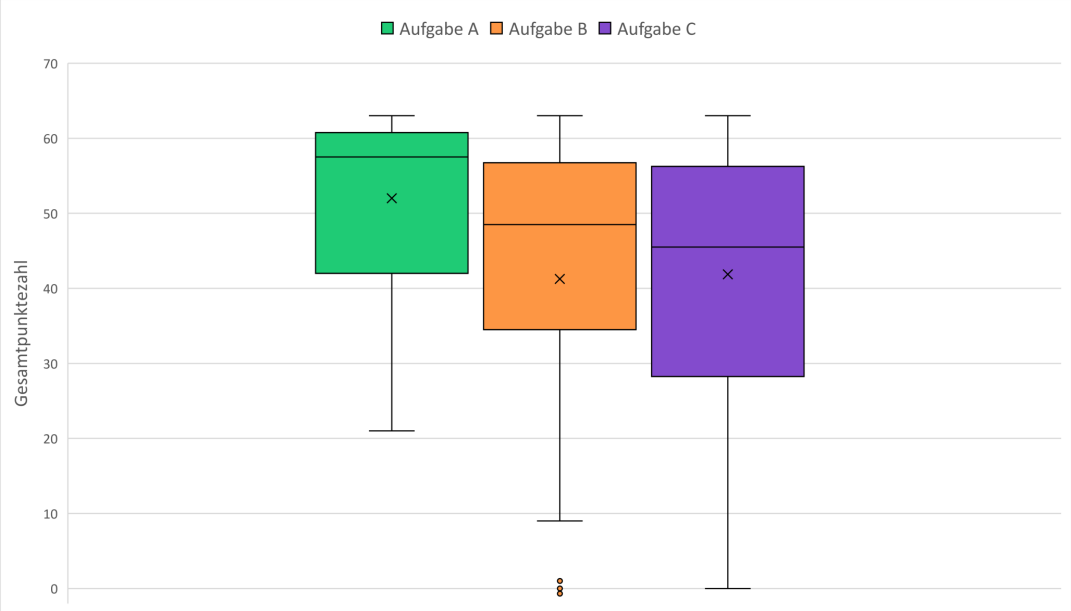


Abbildung 6.11: Darstellung der Punkteverteilung bei den drei gestellten Aufgaben der Benutzerstudie.

Themengebiet abdecken, konnte das erworbene Wissen aus Aufgabe B nicht für Aufgabe C verwendet werden, da unterschiedliche Chatbots getestet wurden. Dies bestätigt auch die durchschnittliche Bewertung von nur 41,8 Punkten bei Aufgabe C. 25 % dieser Chatbots wurden mit maximal 28 Punkten bewertet. Grundsätzlich konnte in allen drei Aufgaben mindestens einmal die Höchstpunktezahl von 63 erreicht werden. In den Aufgaben B und C wurde mindestens einmal eine Bewertung von null Punkten abgegeben. Da die Verteilung der unterschiedlichen Chatbots innerhalb der Aufgaben annähernd gleich ist, kann davon ausgegangen werden, dass Aufgabe A, siehe Tabelle 6.1, für die Teilnehmer am leichtesten zu lösen war. Dies zeigt auch die niedrige Abbruchrate von nur 5 %.

Neben der Bewertung nach Punkten durch die Teilnehmer, wurde auch eine abschließende subjektive Reihung der getesteten Chatbots nach der wahrgenommenen Qualität durchgeführt. Dabei wurde in 45 % der Fälle der optimal generierte Chatbot als Favorit bezeichnet. 30 % der Teilnehmer bevorzugten einen fehlerhaft generierten und lediglich 25 % einen manuell erstellten Chatbot. Diese subjektive Bewertung deckt sich bei allen Teilnehmern mit dem Ergebnis der Punktebewertung.

6.3 Verbesserungsmöglichkeiten

Durch die Evaluierung der Daten, welche in der durchgeführten Studie erhoben wurden, wurden diverse Probleme in der Bedienung des Generators sowie in der grundsätzlichen Logik bei der Erstellung der Trainingsdaten ersichtlich.

6.3.1 Bedienung des Trainingsdaten-Generators

Wie bereits erwähnt, wurden die Arbeitsschritte des Generators beim Einfügen einer neuen Domäne als nicht intuitiv bewertet. Die Teilnehmer bemängelten die fehlenden Anweisungen innerhalb des Workflows. Dieses Problem kann durch aussagekräftigere Bezeichnungen der einzelnen Arbeitsschritte und konkreteren Angaben zum benötigten Format der einzugebenden Daten behoben werden. Mithilfe eines Tutorials oder einer Beispiel-Aufgabe kann dem Nutzer außerdem der Workflow beim Generieren der Trainingsdaten besser veranschaulicht werden.

Auch die Zufriedenheit bei der Dauer des Einfügens einer neuen Domäne wurde von den Teilnehmern der Studie mit nur durchschnittlich 5,2 von 9 möglichen Punkten beurteilt. Dies wurde vor allem durch die Notwendigkeit des wiederholten Eintippens der benötigten Daten für alle gewünschten Parameter begründet. Um hierbei Zeit zu sparen, kann der Generator um die Möglichkeit zur Wiederverwendung bereits bestehender Parameter aus anderen Domänen erweitert werden. Somit können häufig verwendete Parameter, wie Adresse, Datum und Preis kurzerhand kopiert werden, sodass der Benutzer diese nicht mehr explizit definieren muss. Dadurch kann der Zeitaufwand bei der Verwendung des Generators verringert werden.

6.3.2 Qualität der generierten Trainingsdaten

Neben den gezeigten Verbesserungsmöglichkeiten bei der Bedienung des Generators, kann auch die resultierende Qualität der semiautomatisch erstellten Trainingsdaten anhand der Ergebnisse der Studie optimiert werden.

In den Aufgaben B und C im zweiten Teil der Studie wurden die Teilnehmer aufgefordert, nach Museen mit günstigen Eintrittspreisen zu suchen. Wie die erhobenen Daten zeigen, formulierte eine Vielzahl der Teilnehmer dabei den Wunsch eines Preisvergleiches aller vorhandenen Museen. In den generierten Trainingsdaten existieren allerdings keinerlei Formulierungen hierfür. Um diese und ähnliche Anliegen des Nutzers erfüllen zu können, können die einzelnen Parameter beim Einfügen einer neuen Domäne um definierte Eigenschaften erweitert werden. Für jeden verwendeten Parameter können so, neben den bereits vorhandenen Angaben, auch Eigenschaften, wie orts-, zeit- oder preisabhängig zugeordnet werden. Somit können anschließend ergänzende Trainingsdaten generiert werden, welche speziell auf diese Zusatzinformationen abgestimmt sind.

Wie bereits erwähnt, wurden in den Anfragen zu den Aufgaben B und C von den Testpersonen oftmals andere Bezeichnungen für die bestehenden Parameter gewählt. So wurde statt Preisen nach Eintrittspreisen oder Ticketpreisen gesucht. Folglich konnten diese Anfragen vom Chatbot häufig keinem Intent zugeordnet werden. Um dieses Problem zu beheben, können die einzelnen Parameter um eine Liste von Synonymen erweitert werden. Diese Synonyme werden bei der Generierung der Trainingsdaten ebenso verwendet, wie die ursprünglichen Parameterbezeichnungen. Somit wird in den Intents ein breiteres Spektrum an Trainingsdaten abgedeckt.

Bei der Durchführung des zweiten Teils der Studie war auch erkennbar, dass einige Teilnehmer mehrere Anliegen in einer Anfrage kombinierten. So wurde beispielsweise nach der Telefonnummer vom günstigsten Museum in Salzburg gesucht. Diese Funktionalität wurde bei der Entwicklung der Logik für die Generierung nicht berücksichtigt. Durch das Hinzufügen von Kombinationen von filterbaren und nicht filterbaren Parame-

tern innerhalb der Trainingsdaten, können auch diese Anfragen vom Chatbot erkannt, und das Anliegen des Nutzers so mit nur einem Schritt erfüllt werden.

Um die Rate of Confusion des Chatbots zu verringern und die Zufriedenheit in der Verwendung zu steigern, können bei der Generierung der Trainingsdaten auch Hilfestellungen zur Bedienung des Chatbots erstellt werden. Bei wiederholt nicht zuordenbaren Anfragen kann der Chatbot somit Aufschluss über die gewünschte Formulierung des Nutzeranliegens geben. In der durchgeführten Studie wurden nicht vorhandene, beziehungsweise nicht zweckmäßige Hilfestellungen von einigen Teilnehmern bemängelt.

6.4 Fazit

Anhand der erhobenen Daten zeigt sich, dass der Entwicklungsprozess durch den Einsatz des Trainingsdaten-Generators um durchschnittlich 59 % verkürzt werden konnte. Die geforderte Aufgabe zur Generierung der Trainingsdaten konnte jedoch nur von einem Teilnehmer fehlerfrei absolviert werden. Zwei der Teilnehmer beendeten die Aufgabe zwar, die Trainingsdaten konnten aufgrund zu vieler Fehler allerdings nicht in Dialogflow importiert werden.

Bei der Durchführung der Testversuche im zweiten Abschnitt der Studie zeigte sich zwischen den unterschiedlichen Erstellungsmethoden eine relativ gleichmäßige Verteilung der durchschnittlich benötigten Schritte zur Erfüllung der einzelnen Aufgaben. Die Anzahl an Konversationsschritten war bei erfahrenen Teilnehmern im Mittel jedoch höher als bei unerfahrenen. Anhand der erhobenen Daten wurde eine negative Korrelation zwischen der Anzahl an benötigten Schritten und der Bewertung durch die Teilnehmer erkennbar. Weitaus weniger Einfluss auf die Bewertung hatte die benötigte Zeit zum Erfüllen der gestellten Aufgaben.

In den durchgeführten Tests konnte bei manuell erstellten Trainingsdaten mehr als jede dritte Anfrage vom jeweiligen Chatbot nicht zugeordnet werden. Die gesamten Chatbots, die auf generierten Trainingsdaten beruhen, konnten nur knapp jede zehnte Anfrage der Nutzer nicht zuordnen. Diese Zuordnungen waren bei den gesamten generierten Trainingsdaten allerdings in nur 78 % der Fälle korrekt, bei manuell erstellen Trainingsdaten hingegen in 83 % der Fälle.

In der Gesamtbewertung der Teilnehmer konnten die auf optimal generierten Trainingsdaten beruhenden Chatbots durchschnittlich die meisten Punkte erreichen. Auch die Spannweite aller Bewertungen war im Vergleich zu den anderen Erstellungsmethoden am geringsten. Am schlechtesten wurden die manuell erstellten Trainingsdaten bewertet. Durchschnittlich konnten die optimal generierten Trainingsdaten 22 % mehr Punkte erreichen als die manuell erstellten. In dieser Studie wirkte sich die Erfahrung der Teilnehmer in Bezug auf Chatbots nicht auf die anschließende Bewertung aus.

Bei einer Gesamtabbruchrate von 5 %, konnten lediglich bei der Verwendung von optimal generierten Trainingsdaten alle Testversuche erfolgreich beendet werden.

Abschließend lässt sich zusammenfassen, dass laut der subjektiven Reihung nach der Qualität der zwei beziehungsweise drei getesteten Chatbots, 45 % der Teilnehmer Chatbots mit optimal generierten Trainingsdaten bevorzugten. Nur 25 % der Teilnehmer favorisierten die manuell erstellten Chatbots. Diese Wahrnehmung deckt sich auch in der Punktebewertung der Teilnehmer.

Natürlich darf in dieser Studie auch die Tatsache nicht ignoriert werden, dass die

Ergebnisse aus einer Testsituation resultieren. Manche Teilnehmer formulierten ihre Anfragen absichtlich um, um keine Standardsätze zu verwenden und die Chatbots so an ihr Limit zu bringen und Fehler aufzuzeigen. Dieses Aufzeigen von Fehlern ist in dieser Studie zwar wünschenswert, dennoch können die Ergebnisse so nicht direkt auf die Realität übertragen werden.

Da die Testversuche unter Beobachtung durchgeführt wurden, ist anzunehmen, dass die Teilnehmer ihr Verhalten adaptierten und folglich bessere Leistungen erbracht wurden. So wurde bei nicht oder falsch interpretierten Anfragen oftmals eine Vielzahl an weiteren Formulierungen verwendet, um die Aufgabe erfolgreich beenden zu können. Diese Änderung im Verhalten der Teilnehmer und die daraus resultierende Leistungssteigerung während einer Testsituation ist allgemein als Hawthorne-Effekt[1] bekannt. Da diese Testsituation im realen Einsatz der Chatbots nicht gegeben ist, wäre die Abbruchrate mitunter deutlich höher, da für die Nutzer oftmals keine Notwendigkeit besteht, die Aufgabe mithilfe eines Chatbots zu lösen.

Dennoch wurde die Relevanz des Trainingsdaten-Generators in der durchgeführten Benutzerstudie bestätigt, da sowohl die benötigte Zeit zur Entwicklung reduziert, als auch die Qualität dieser generierten Trainingsdaten gesteigert werden konnte.

[1] https://en.oxforddictionaries.com/definition/us/hawthorne_effect

Kapitel 7

Schlussbetrachtung

In diesem Kapitel werden der theoretische Teil der Arbeit und die wichtigsten Erkenntnisse der Evaluierung zusammengefasst. Den Abschluss dieser Arbeit bildet ein Ausblick über die Zukunft von Chatbots und deren Entwicklung.

7.1 Zusammenfassung

Chatbots sind keineswegs ein neues Phänomen in der Kommunikation zwischen Mensch und Maschine. Bereits vor über 50 Jahren wurden erste Ideen präsentiert und umgesetzt. Seitdem wurde die Entwicklung stetig vorangetrieben. Durch die gestiegene Rechenleistung und die großen Fortschritte im Bereich der künstlichen Intelligenz, nahm die Relevanz von Chatbots in den letzten Jahren immer weiter zu.

Grundsätzlich eignen sich Chatbots, um Nutzeranfragen, die sich häufig wiederholen oder ähnlich sind, zu beantworten. Deshalb werden Chatbots von Unternehmen vor allem im Customer-Support eingesetzt. So können beispielsweise Standardauskünfte, Reklamationen oder Rücksendungen automatisiert beantwortet werden. Aber auch in der Tourismusbranche bieten sich unzählige Einsatzmöglichkeiten. Die Suche nach Urlauben, Unterkünften, Flügen, Restaurants sowie deren Buchung können durch die Verwendung von Chatbots abgedeckt werden. Diese Vielzahl an möglichen Anwendungen ist ein erheblicher Faktor für die Beliebtheit von Chatbots.

Eingehende Nutzeranfragen werden vom Chatbot in ihre Einzelteile zerlegt und nach festgelegten Mustern durchsucht. Existiert die Intention des Nutzers in diesen Mustern, wird eine zugehörige Aktion angestoßen und die passende Antwort angezeigt. Mittlerweile existieren unzählige Tools zur Erstellung von Chatbots. Diese wurden unter anderem von Microsoft, Google und Facebook veröffentlicht und bieten Anwendern mit unterschiedlichem technischen Know-how die Möglichkeit, rasch Chatbots zu entwickeln. Diese Tools unterstützen außerdem die Veröffentlichung dieser entwickelten Chatbots auf unzähligen Integrationsplattformen.

Auch bei der Verwendung von effizienten Tools bleibt der aufwendigste und zeitintensivste Prozess in der Entwicklung von Chatbots die Erstellung des Dialoges, respektive die Ausarbeitung der Trainingsdaten. Anhand der Ergebnisse der durchgeführten Benutzerstudie zeigt sich, dass der implementierte Generator die benötigte Zeit zum Entwickeln eines Chatbots um durchschnittlich 59 % reduzieren kann. Auch die Qua-

lität der generierten Trainingsdaten konnte mit einem Plus von 13 % im Vergleich zu manuell erstellten Trainingsdaten in dieser Studie überzeugen. Der Unterschied in der Bewertung von optimal generierten und manuell erstellten Trainingsdaten fällt dabei mit 22 % noch deutlicher aus. Durch die modulare Bauweise des Generators und die in Kapitel 6.3 erläuterten Verbesserungsmöglichkeiten, kann die Relevanz dieser Trainingsdaten weiter gesteigert werden.

Trotz der soliden Studienergebnisse wird der Generator keinen vollständigen Dialogverlauf entwickeln können. Zusätzlich zu den generierten Trainingsdaten sollten individuell erstellte Ausdrücke ergänzt werden. Dadurch kann die Zuordnung der Nutzeranfragen verbessert, und außerordentliche Funktionalitäten des Chatbots abgedeckt werden. Dennoch bietet der Generator Unterstützung bei der Entwicklung von Trainingsdaten für eine Vielzahl von zielorientierten Chatbots unterschiedlicher Domänen. Da der Generator auf bestehenden Ontologien basiert, werden zur grundlegenden Entwicklung der Trainingsdaten keine weiteren domänenspezifischen Daten benötigt und somit auch die erforderliche Zeit bis zum tatsächlichen Entwicklungsstart des Chatbots verkürzt.

7.2 Ausblick

Durch die vielseitigen Einsatzmöglichkeiten sind der Verwendung von Chatbots beinahe keine Grenzen gesetzt. Dank der simplen Integration in bereits bestehende Messaging-Plattformen begleiten sie die Nutzer in allen Bereichen des täglichen Lebens. Doch sie erleichtern nicht nur alltägliche Aufgaben, sondern dienen auch zur Unterhaltung der Nutzer.

Laut einer von Oracle durchgeführten Studie, planen 44 % der befragten Unternehmen bis 2020 Chatbots zu entwickeln, 36 % nutzen diese bereits [25]. Die entwickelten Chatbots sollen vor allem die Kundenkommunikation und Kundenbindung verbessern und den Nutzern rund um die Uhr Anliegen erfüllen.

Die Popularität von Chatbots spiegelt sich auch in der Weiterentwicklung von Dialogflow wieder. Während der Umsetzung des Projektes wurden mehrere Funktionsupdates und Re-Designs durchgeführt. Zusätzlich wurden eine neue API-Version und die kostenpflichtige Enterprise-Edition veröffentlicht. Durch diese ständigen Neuerungen zeigt sich das enorme Interesse an Chatbots und der Bedarf an geeigneten Entwicklungsinstrumenten. Auch die Teilnehmer der Studie bestätigen diese Einschätzung. Von den insgesamt 26 Teilnehmern gaben 15 an, bereits Chatbots genutzt zu haben, alle von ihnen waren zumindest mit der Thematik vertraut.

Durch die stetige Entwicklung im Bereich der künstlichen Intelligenz, vor allem Machine Learning, wird das Sprachverständnis von Chatbots verbessert werden können und die Relevanz von Chatbots auch in Zukunft weiter steigen. Somit werden allerdings immer mehr Methoden und Ansätze notwendig, um die Entwicklung und Instandhaltung von Chatbots so effizient wie möglich zu gestalten.

Anhang A

Inhalt der CD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 PDF-Dateien

Pfad: /

Pfleger_Michaela_2018.pdf Masterarbeit (Gesamtdokument)

A.2 Online-Quellen

Pfad: /online-references

*.pdf . . . . . Online-Quellen als PDF-Dateien

A.3 Projekt

Pfad: /sources

dialogflow.zip . . . . . Dialogflow-Projekt des Chatbots

pimcore.zip . . . . . Pimcore-Bundle zur Installation von Backend, siehe Abschnitt 5.2.3, Frontend, siehe Abschnitt 5.2.4 und Trainingsdaten-Generator, siehe Abschnitt 5.3

A.4 Abbildungen

Pfad: /images

*.pdf . . . . . Original Vektorgrafiken

*.PNG . . . . . Original Rasterbilder

Quellenverzeichnis

Literatur

[1] Bayan Abushawar und Eric Atwell. „Automatic Extraction of Chatbot Training Data from Natural Dialogue Corpora“. In: *Proceedings of RE-WOCHAT: Workshop on Collecting and Generating Resources for Chatbots and Conversational Agents-Development and Evaluation Workshop Programme*. (Portorož, Slowenien). SIGdial und COLIPS, 2016, S. 29–38 (siehe S. 15).

[2] Daniel G. Bobrow u. a. „GUS, a frame-driven dialog system“. *Artificial Intelligence* 8.2 (1977), S. 155–173 (siehe S. 18).

[3] Alexander Braun. *Chatbots in der Kundenkommunikation*. Springer, 2003 (siehe S. 5, 9).

[4] Ecma International. *ECMAScript 2015 Language Specification*. Standard ECMA-262. Juni 2015. URL: http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf (siehe S. 35).

[5] Asbjørn Følstad und Petter Bae Brandtzæg. „Chatbots and the new world of HCI“. *Interactions* 24.4 (2017), S. 38–42 (siehe S. 1, 6, 10, 12).

[6] Ulrich Furbach. „Turing und Künstliche Intelligenz“. *Informatik-Spektrum* 35.4 (Aug. 2012), S. 280–286 (siehe S. 6).

[7] Peter Gentsch. „Conversational Commerce: Bots, Messaging, Algorithmen und Artificial Intelligence“. In: *Künstliche Intelligenz für Sales, Marketing und Service: Mit AI und Bots zu einem Algorithmic Business – Konzepte, Technologien und Best Practices*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, S. 83–116 (siehe S. 3, 4, 9).

[8] Mark Hartmann. „Machine Learning und IT-Security“. *Datenschutz und Datensicherheit – DuD* 42.4 (Apr. 2018), S. 231–235 (siehe S. 3).

[9] Wolfgang Hesse und Hermann Engesser. „Ontologie“. *Informatik-Spektrum* 37.4 (Aug. 2014), S. 281–282 (siehe S. 18).

[10] Daniel Jurafsky und James Martin. „Dialog Systems and Chatbots“. *Speech and Language Processing*. (2017), S. 418–440 (siehe S. 19, 20).

[11] Rashid Khan und Anik Das. *Build Better Chatbots – A Complete Guide to Getting Started with Chatbots*. New York: Apress, 2017 (siehe S. 6, 7).

[12] Robert Kusber. „Chatbots – Conversational UX Platforms“. In: *Innovationen und Innovationsmanagement in der Finanzbranche*. Hrsg. von Remigiusz Smolinski u. a. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, S. 231–244 (siehe S. 4, 5, 7, 15).

[13] Adrian Lobe. „Chatbots könnten Arztbesuche verringern“. *Der Standard* (März 2017). URL: https://derstandard.at/2000053880580/Chatbots-koennten-Arztbesuche-verringern (siehe S. 7).

[14] John McCarthy u. a. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. 1955. URL: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html (siehe S. 3).

[15] Sumit Negi u. a. „Automatically Extracting Dialog Models from Conversation Transcripts“. In: *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*. (Miami, FL, USA). IEEE, Dez. 2009, S. 890–895 (siehe S. 15).

[16] Valentin Plenk. „Dateiformate: JSON“. In: *Angewandte Netzwerktechnik kompakt: Dateiformate, Übertragungsprotokolle und ihre Nutzung in Java-Applikationen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, S. 41–51 (siehe S. 25).

[17] Efraim Turban u. a. *Electronic Commerce 2018 - A Managerial and Social Networks Perspective*. Berlin, Heidelberg: Springer, 2017 (siehe S. 6).

[18] Alan Turing. „Computing Machinery and Intelligence“. *Mind* LIX.236 (1950), S. 433–460 (siehe S. 3).

[19] Richard S. Wallace. „The Anatomy of A.L.I.C.E.“ In: *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. Hrsg. von Robert Epstein, Gary Roberts und Grace Beber. Dordrecht: Springer Netherlands, 2009, S. 181–210 (siehe S. 6).

[20] Joseph Weizenbaum. „ELIZA – a Computer Program for the Study of Natural Language Communication Between Man and Machine“. *Communications of the ACM* 9.1 (Jan. 1966), S. 36–45 (siehe S. 5, 6).

[21] Thomas Wilde. „Customer Engagement mit Chatbots und Collaboration Bots: Vorgehen, Chancen und Risiken zum Einsatz von Bots in Service und Marketing“. In: *Künstliche Intelligenz für Sales, Marketing und Service: Mit AI und Bots zu einem Algorithmic Business – Konzepte, Technologien und Best Practices*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, S. 138–149 (siehe S. 9).

Online-Quellen

[22] Pulse Chat. *Chatbot Testing*. Okt. 2017. URL: https://chatbotslife.com/chatbot-testing-1f359b5459a (besucht am 17. 07. 2018) (siehe S. 49).

[23] Pulse Chat. *Must have KPIs for Chat bots*. Sep. 2017. URL: https://chatbotslife.com/must-have-kpis-for-chat-bots-b75b9efddfb9 (besucht am 17. 07. 2018) (siehe S. 48, 49).

[24] Chatfuel. *Chatfuel*. 2017. URL: https://chatfuel.com/ (besucht am 07. 02. 2018) (siehe S. 11).

[25] Oracle Corporation. *Can Virtual Experiences Replace Reality?* 2016. URL: https://www.oracle.com/webfolder/s/delivery_production/docs/FY16h1/doc35/CXResearchVirtualExperiences.pdf (besucht am 06. 08. 2018) (siehe S. 66).

[26] Dialogflow. *Actions and Parameters*. 2018. URL: https://dialogflow.com/docs/actions-and-parameters (besucht am 21. 09. 2018) (siehe S. 14).

[27] Dialogflow. *Contexts*. Jan. 2018. URL: https://dialogflow.com/docs/contexts (besucht am 16. 03. 2018) (siehe S. 13).

[28] Dialogflow. *Dialogflow*. 2018. URL: https://dialogflow.com/ (besucht am 07. 02. 2018) (siehe S. 12).

[29] Dialogflow. *Entities*. Feb. 2018. URL: https://dialogflow.com/docs/entities (besucht am 16. 03. 2018) (siehe S. 13).

[30] Dialogflow. *Fulfillment*. März 2018. URL: https://dialogflow.com/docs/fulfillment (besucht am 16. 03. 2018) (siehe S. 14).

[31] Dialogflow. *Intents*. Feb. 2018. URL: https://dialogflow.com/docs/intents (besucht am 15. 03. 2018) (siehe S. 13, 14).

[32] Dialogflow. *Rich Messages*. 2017. URL: https://dialogflow.com/docs/rich-messages (besucht am 21. 09. 2018) (siehe S. 14).

[33] Dialogflow. *Training*. Feb. 2018. URL: https://dialogflow.com/docs/training (besucht am 16. 03. 2018) (siehe S. 16).

[34] Britany DiCicco. *Frost & Sullivan: Technology Innovation Leader in CMS Market 2016*. Mai 2016. URL: https://pimcore.com/de/ressourcen/blog/frost-sullivan-technology-innovation-leader-in-cms-market-2016_a955 (besucht am 27. 04. 2018) (siehe S. 24).

[35] Alexander Engelhardt. *Streudiagramme*. Feb. 2015. URL: https://www.crashkurs-statistik.de/streudiagramme/ (besucht am 26. 07. 2018) (siehe S. 49).

[36] Kauz GmbH. *Was ist der Unterschied zwischen NLP und NLU?* Dez. 2016. URL: https://kauz.net/2016/12/30/was-ist-der-unterschied-zwischen-nlp-und-nlu/ (besucht am 06. 08. 2018) (siehe S. 4).

[37] Pimcore GmbH. *Bundle Developer's Guide*. 2018. URL: https://pimcore.com/docs/5.x/Development_Documentation/Extending_Pimcore/Bundle_Developers_Guide/index.html (besucht am 27. 04. 2018) (siehe S. 24).

[38] Pimcore GmbH. *MVC in Pimcore*. 2018. URL: https://pimcore.com/docs/5.x/Development_Documentation/MVC/index.html (besucht am 21. 09. 2018) (siehe S. 24).

[39] USU GmbH. *Chatbots - Revolution im Service?* Aug. 2017. URL: https://www.unymira.com/?eID=dumpFile&t=f&f=113909&n=chatbot-studie-2017 (besucht am 06. 08. 2018) (siehe S. 1).

[40] InfoWissWiki. *Stoppwort*. Sep. 2008. URL: https://wiki.infowiss.net/Stoppwort (besucht am 18. 05. 2018) (siehe S. 45).

[41] Julia Jung und Stefan Niemeyer. *Chatbots im Tourismus*. Jan. 2017. URL: http://labor.neusta-etourism.de/ebook/ebook-chatbots-neusta.pdf (besucht am 28. 08. 2018) (siehe S. 5, 8, 9).

[42] KAYAK. *Mobile Travel Report 2017*. 2017. URL: https://www.kayak.de/news/wp-content/uploads/2017/05/DE_Report-compressed-1.pdf (besucht am 02. 08. 2018) (siehe S. 17).

[43] Christian Kemptner. *CMS Critic Award 2017: 'Best Enterprise Content Management System'*. Nov. 2017. URL: https://pimcore.com/de/ressourcen/blog/cms-critic-award-2017-best-enterprise-content-management-system_a1213 (besucht am 27. 04. 2018) (siehe S. 24).

[44] Chris Knight. *How to Teach Your Chatbot With Training Data*. März 2018. URL: https://chatbotslife.com/how-to-teach-your-chatbot-with-training-data-46c58b873c31 (besucht am 02. 08. 2018) (siehe S. 15).

[45] Anna Kulawik. *Making Chatbots Talk - Writing Conversational UI Scripts Step by Step*. Feb. 2017. URL: https://uxdesign.cc/making-chatbots-talk-writing-conversational-ui-scripts-step-by-step-62622abfb5cf (besucht am 02. 08. 2018) (siehe S. 16).

[46] LivePerson. *How consumers view bots in customer care*. 2017. URL: https://docsend.com/view/826nkc4 (besucht am 07. 08. 2018) (siehe S. 1).

[47] Google LLC. *Dialogflow Editions*. Aug. 2018. URL: https://cloud.google.com/dialogflow-enterprise/docs/editions (besucht am 21. 09. 2018) (siehe S. 14).

[48] Maruti Techlabs Pvt. Ltd. *Complete guide on Bot Frameworks*. 2017. URL: https://www.marutitech.com/complete-guide-bot-frameworks/ (besucht am 15. 03. 2018) (siehe S. 11, 12).

[49] Microsoft. *Electronics retailer uses interactive artificial intelligence bot to boost customer engagement*. Juli 2017. URL: https://customers.microsoft.com/en-us/story/dixons-carphone-retail-cognitive-services (besucht am 11. 03. 2018) (siehe S. 12).

[50] Microsoft. *Microsoft Bot Framework*. 2017. URL: https://dev.botframework.com/ (besucht am 07. 02. 2018) (siehe S. 12).

[51] Microsoft. *UPS paves the way for better service with faster development and artificial intelligence*. Sep. 2017. URL: https://customers.microsoft.com/en-us/story/ups (besucht am 15. 03. 2018) (siehe S. 12).

[52] Onlim. *7 große Marken die bereits Chatbots einsetzen*. Aug. 2017. URL: https://onlim.com/7-grosse-marken-die-bereits-chatbots-einsetzen/ (besucht am 17. 03. 2018) (siehe S. 7).

[53] Onlim. *Die Geschichte und Entwicklung von Chatbots*. Okt. 2017. URL: https://onlim.com/die-geschichte-und-entwicklung-von-chatbots/ (besucht am 18. 03. 2018) (siehe S. 6).

[54] Heinrich Schwietering. *Cron*. Feb. 2018. URL: https://wiki.ubuntuusers.de/Cron/ (besucht am 16. 05. 2018) (siehe S. 33).

[55] Manu Sporny u. a. *JSON-LD 1.0*. URL: https://www.w3.org/TR/json-ld/ (besucht am 18. 05. 2018) (siehe S. 25).

[56] Statista. *Statistik-Lexikon: Definition Korrelation*. URL: https://de.statista.com/statistik/lexikon/definition/77/korrelation/ (besucht am 27. 07. 2018) (siehe S. 52).

[57] Statista. *Statistik-Lexikon: Definition Korrelationskoeffizient*. URL: https://de.statista.com/statistik/lexikon/definition/78/korrelationskoeffizient/ (besucht am 27. 07. 2018) (siehe S. 52).

[58] Björn Tantau. *Chatbots und ihre faszinierende Geschichte*. URL: https://bjoerntantau.com/chatbots-23122016.html (besucht am 18. 03. 2018) (siehe S. 6).

[59] *<template>: The Content Template element*. Apr. 2018. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template (besucht am 28. 04. 2018) (siehe S. 36).

[60] Topbots. *Kayak*. 2017. URL: https://www.topbots.com/project/kayak-bot-facebook-messenger-slack-alexa/ (besucht am 07. 02. 2018) (siehe S. 10, 11).

[61] Dean Withey. *Where to get Chatbot Training Data (and what it is)*. Juli 2017. URL: https://blog.ubisend.com/optimise-chatbots/chatbot-training-data (besucht am 02. 08. 2018) (siehe S. 17).