Bedienung einer Getränkemischmaschine über Sprachbefehle

Studienarbeit

für die Prüfung zum

Bachelor of Science

des Studienganges Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Felix Manuel Gervasi

und

Alena Sutiagina

Abgabedatum 31. März 2023

Bearbeitungszeitraum	6 Monate
Matrikelnummer	1052491
Kurs	TINF20B4
Ausbildungsfirma	
Betreuer der Studienarbeit	Prof. Dr. Jörn Eisenbiegler

Erklärung

Ich versichere hiermit, dass ich meine Studienarbeitmit dem Thema: »Bedienung einer Getränkemischmaschine über Sprachbefehle« selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort Datum

Unterschrift

Zusammenfassung

TODO

Abstract

TODO

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

Kapitel 1

Einleitung

Die Informationstechnik versteckt sich heutzutage fast überall - selbst dort, wo sie von den meisten Menschen nicht vermutet werden würde. Beispiele hierfür sind Autos, Kaffeemaschinen, Zahnbürsten, Rasierer, Küchengeräte und vieles mehr. Grund dafür ist die fortschreitende Möglichkeit der Miniaturisierung von Computern, sodass diese nahezu überall verbaut werden können. Beispielsweise können Mikrochips in der Kaffeemaschine dafür sorgen, dass die richtige Menge an Kaffee serviert wird oder der Füllstand der einzelnen Behälter angezeigt werden kann. Solche Systeme, die Informationen mit Hilfe eines Computers verarbeiten und dabei mit ihrer Umgebung derartig „verschmelzen“, nennt man auch *embedded systems* (z. Dt. *eingebettete Systeme*) [Marwedel 2021].

Der aktuelle Trend des Internet der Dinge (IoT) führt zu einem noch größeren Anstieg eingebetteter Systeme im Alltag. Im IoT geht es speziell um eingebettete Systeme, die internetfähig (vernetzt) sind. Nach Schätzungen des Marktforschungsunternehmens *Gartner* gab es im Jahr 2017 8,4 Milliarden solcher vernetzten Geräte weltweit [Jansen 2017]. Das die Menge der vernetzten Geräte als Teilmenge der eingebetteten Systeme betrachtet werden kann ist damit zu rechnen, dass deren Anzahl sogar weit größer ausfällt.

1.1 Aufgabenstellung

Im Rahmen dieser Arbeit geht es um die Sprachsteuerung einer Getränkemischmaschine, die in diesem Fall als eingebettetes System zu verstehen ist und in einer vorangegangenen Arbeit bereits konzipiert und gebaut wurde [Götz, Höckele und Lobert 2022]. Sie verfügt derzeit über ein Touch-Display zur Bedienung durch den Benutzer. Ziel der Arbeit ist es zusätzlich eine natürlichsprachliche Interaktion mit der Maschine zu ermöglichen, die mindestens den Funktionsumfang besitzt, der aktuell über die Bildschirmeingabe möglich ist. Dabei soll die Maschine nicht nur in der Lage sein die natrüliche Sprache des Benutzers in ein geeignetes Format umzuwandeln, sodass die Maschine den korrekten Befehl ausführt. Sie soll auch in der

Lage sein dem Benutzer zu Antworten, sodass die Illusion einer Konversation mit der Maschine entsteht.

1.2 Vorgehen

Zunächst müssen die Sprachverarbeitung und Spracherkennung betrachtet werden. Die Sprachverarbeitung dient der Interpretation des Gesprochenen um eine geeignete Antwort auszugeben und dem Übersetzen in einen Maschinenbefehl. Im Rahmen dieser Arbeit sollen dafür Verfahren und Techniken der Künstlichen Intelligenz (KI) und des Machine Learning (ML) eingesetzt werden. Die Spracherkennung beschäftigt sich mit der Aufnahme des Tonsignals bzw. der Schallwellen (bspw. über ein Mikrofon) und dem Umwandeln dieser Signale in Text, sodass dieser an das KI-Modell weitergereicht werden kann.

Bei der Arbeit mit eingebetteten Systemen muss man sich der vorhandenen Hardwareleistung und den benötigten Hardwareanforderungen bewusst sein, da diese meist sehr begrenzt ist. Deshalb werden im Rahmen dieser Arbeit verschiedene Ansätze diskutiert, wie und wo die einzelnen Schritte und Berechnungen ablaufen sollen (s. Kapitel 4).

Kapitel 2

Anforderungen

Im Folgenden sollen die Anforderungen an das Ergebnis der Arbeit konkretisiert werden.

2.1 Antwortzeit

Diese Eigenschaft beschreibt die Zeitdauer vom registrieren eines Sprachbefehls bis zur Ausführung des Befehls durch die Mischmaschine und das Zurückgeben einer Antwort an den Benutzer. Die Antwortzeit spielt eine große Rolle bei der Bedienbarkeit eines interaktiven Systems, um das es sich bei der Mischmaschine handelt. Zu lange Antwortzeiten können dazu führen, dass der Benutzer seine ursprünglichen Ziele vergisst oder in Stress gerät, da in den aller meisten Fällen der Grund für eine lange Antwortzeit vor dem Benutzer verborgen bleibt. Umgekehrt können zu kurze Antwortzeiten ebenfalls zu Stress und Fehlbedienung seitens des Benutzers führen. Dies liegt unter anderem daran, dass kurze Antwortzeiten den Benutzer dazu veranlassen weniger über seine Aktionen und deren Folgen nachzudenken. Als eine für viele Anwendungen geeignete Antwortzeit werden zwei bis vier Sekunden genannt [Herczeg 2018]. Als maximale tolerierbare Antwortzeit werden für dieses Projekt sechs Sekunden festgelegt. Diese vergleichsweise lange Zeitdauer lässt sich zum Einen mit den langwierigen aber notwendigen Berechnungen begründen, die für die Spracherkennung und -verarbeitung benötigt werden. Zum Anderen werden die Auswirkungen einer langen Antwortzeit als gering eingeschätzt, da der Benutzer für diese Anwendung keine Teilarbeitsschritte o. ä. im Gedächtnis behalten muss. Das Ziel des Benutzers sich ein Getränk zubereiten zu lassen ist nach dem Eingang des Sprachbefehls bereits erfüllt.

2.2 Offline-Funktionalität

Die Mischmaschine sollte für die Sprachsteuerung keine Verbindung zum Internet benötigen, da dies die möglichen Einsatzorte der Maschine deutlich einschränken würde. Diese Anforderung schränkt die möglichen, einzusetzenden Technologien zur Spracherkennung und -verarbeitung stark ein, da keine Cloud-Services wie bspw. *Google Cloud Speech* eingesetzt werden können

[Google o. D.] Eine weitere Herausforderung die dadurch entsteht ist, dass Berechnungen die unter Umständen sehr aufwendig sein können nicht ausgelagert sondern auf der Hardware innerhalb der Mischmaschine ausgeführt werden müssen.

2.3 Lautstärke

Die Lautstärke der, von der Mischmaschine zurückgegebenen Antwort, muss laut genug sein, sodass sie vom Benutzer gut verstanden werden kann. Diese Eigenschaft schränkt die Art und Weise wie die Hardware (Computer und Mikrofon) in die Mischmaschine eingebaut werden kann ein und welche Art von Hardware überhaupt verwendet werden kann.

2.4 Entfernung

Mit dieser Eigenschaft ist die Entfernung des Anwenders zu der Mischmaschine gemeint. Es muss dem Anwender ermöglicht werden aus einer moderaten Entfernung mit der Mischmaschine über die Sprachsteuerung zu interagieren. Sowohl die Eingabe eines Befehls über die Sprachsteuerung als auch die zu hörende Antwort sollte mindestens aus einer Entfernung von einem Meter möglich sein. Dafür müssen die Lautsprecher eine bestimmte Lautstärke erreichen können und das Mikrofon eine moderate Empfindlichkeit aufweisen.

2.5 Antworten

Die Antworten, die durch die Mischmaschine an den Benutzer zurückgegeben werden, sollen mit Hilfe eines eigens erstellten Sprachmodells auf Basis von künstlicher Intelligenz und ML erfolgen. Die Antworten der Mischmaschine sollen außerdem bissiger bzw. sarkastischer Natur sein was, je nach verwandter Technik, bei der Auswahl der Trainingsdaten eine große Rolle spielt.

Des Weiteren bestehen die Anforderungen, dass der Benutzer auf Deutsch mit der Mischmaschine kommunizieren können muss und die Antworten der Mischmaschine kontrollierbar sein müssen. Mit der Kontrollierbarkeit ist gemeint, dass Vorhersagen darüber gemacht werden können, was die Mischmaschine in Etwa auf eine bestimmte Frage oder sonstige Benutzereingabe antworten wird. Dies soll verhindern, dass der Benutzer von unerwarteten Reaktionen seitens der Maschine überrascht wird und die Antwroten der Maschine den Benutzer nicht beleidigen (aufgrund der sarkastischen bzw. humorvollen Art und Weise, wie die Maschine antworten soll).

2.6 Kosten

Die Materialkosten sollten einen gewissen Maximalbetrag nicht überschreiten. Zu den benötigten Materialien zählen ein Mikrocomputer zur Durchführung der Spracherkennung und -verarbeitung,

ein Mikrofon zur Aufnahme der Sprache und Lautsprecher zur Tonausgabe. Das Ziel besteht darin einen Betrag von 200€ nicht zu überschreiten.

2.7 Verbrauch von Arbeits- und Festplattenspeicher

Diese Anforderung korreliert mit der Anforderung nach Offline-Funktionalität. Diese bedingt, dass aufwendige, rechen- oder speicherintensive Operationen nicht auf entfernte Rechner ausgelagert werden können sondern, alles auf „kleiner“ Hardware innerhalb der Mischmaschine ablaufen muss. Diese Einschränkung soll anhand eines konkreten Beispiels verdeutlicht werden. Die vierte Version der bekannten Mikrocomputerreihe *Raspberry Pi* umfasst im Modell B maximal acht Gigabyte Read Only Memory (RAM) und einen Micro-SD-Karten-Steckplatz [Ltd, Raspberry Pi o. D.] Zwar sind Micro-SD-Karten von bis zu mehreren hundert Gigabyte erhältlich, jedoch muss dabei der Kostenfaktor mit beachtet werden. Dadurch sind sowohl Primär- als auch Sekundärspeicher stark beschränkt.

Auch Prozessorgeschwindigkeit und Grafikkartenleistung können bei der Spracherkennung- und verarbeitung eine Rolle spielen. Diese ist bei Mikrocomputern ebenfalls eingeschränkt und korreliert negativ mit der Anforderung an Geschwindigkeit. Einen ebenso negativen Einfluss auf die Geschwindigkeit des Gesamtsystems haben die virtuelle Vergrößerung des RAM durch Swapping oder die SD-Karte selbst, welche im Vergleich mit anderen Speichertechnologien in Sachen Geschwindigkeit deutlich das Nachsehen hat.

In anbetracht der geschilderten Herausfoderungen und Kostenbetrachtung wird der maximal zu verbrauchende Arbeits- und Festplattenspeicher für dieses Projekt auf acht Gigabyte RAM und 20 Gigabyte Festplattenspeicher festgelegt.

2.8 Anpassungsfähigkeit

Diese Anforderung beschreibt den Grad der Einfachheit bei Änderung der Anforderungen oder Umwelt die Sprachsteuerung der Mischmaschine an diese neuen Begebenheiten anzupassen. Ein einfaches Beispiel für die Anpassungsfähigkeit des Systems wäre das Hinzukommen eines Behälters innerhalb der Mischmaschine. Tritt dieser Fall ein sollte es leicht möglich sein die Sprachsteuerung so anzupassen, dass der Benutzer auch die Möglichkeit bekommt aus dem neuen, fünften Behälter Getränke zu Mischen und zu Bestellen.

Kapitel 3

Stand der Technik

3.1 Getränkemischmaschine

Ziel dieser Arbeit ist die Implementierung einer Sprachsteuerung für eine Getränkemischmaschine. Die Getränkemischmaschine wurde bereits in einem vorangegangenen Projekt erstellt [Götz, Höckele und Lobert 2022]. In diesem Abschnitt wird darauf eingegangen, um was für eine Art von Maschine es sich dabei handelt und es werden ihre Funktionsweise und ihr Aufbau beschrieben.

Das Mischen von Getränken bzw. Flüssigkeiten ist der Anwendungsfall für den die Maschine konzipiert wurde. Dazu besitzt die Mischmaschine fünf Behälter zu je einem Liter. Jedem Behälter ist eine Pumpe zugeordnet, die separat angesteuert werden kann und dafür sorgt, dass die Flüssigkeit aus dem Behälter zur Getränkeausgabe gelangt. Kurz vor dem Ausgang werden die Schläuche der Behälter zusammengeführt, wodurch letztlich die Mischung der verschiedenen Getränke erzielt wird. Sogenannte „Rückschlagventile“ sorgen dafür, dass ein Zurückfließen der Getränkemischung in die Behälter verhindert wird.

Zur Steuerung der Maschine durch einen Benutzer befinden sich an der Vorderseite zwei Knöpfe und ein Touch-Display. Einer der Knöpfe dient dem Anschalten der Maschine und ein weiterer der Ausgabe des Getränks. Über das Touch-Display kann der Benutzer die Mischung seiner Getränke konfigurieren und die Maschine administrieren.

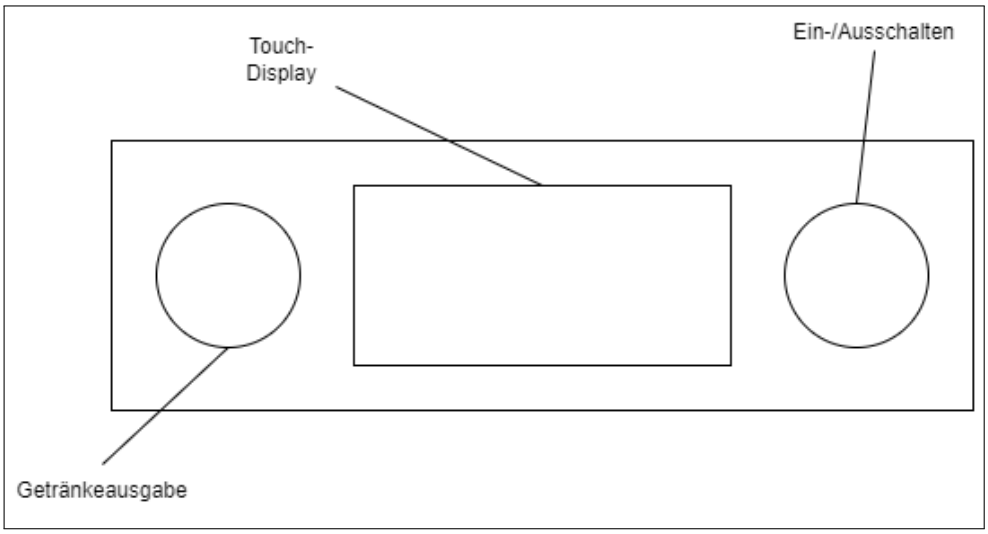

Abbildung 3.1: Schematischer Aufbau der Bedienungsschnittstelle

Abbildung 3.1 stellt den schematischen Aufbau der Bedienungsschnittstelle der Getränkemischmaschine für ein besseres Verständnis dar.

Die technische Umsetzung basiert auf der Kommunikation zwischen dem Touch-Display und einem, in der Getränkemischmaschine verbauten, Arduino, der anhand der Daten von der Benutzereingabe, die Pumpen steuert. Beim Drücken des Startknopfes werden das Display und der Arduino mit Strom versorgt, sodass sowohl das Display als auch der Arduino starten und mit der Ausführung des benutzerdefinierten Quelltextes beginnen. Auf dem Bediendisplay findet sich fünf Schieberegler - ein Schieberegler je Behälter - mit denen der Benutzer die Zusammensetzung seines Mischgetränks aus den fünf Behältern konfigurieren kann. Beim Drücken eines in der User Interface (UI) des Displays dargestellten Knopfes werden die Werte der Schieberegler an den Arduino übertragen. Dieser rechnet die Prozentwerte der Schieberegler in Durchsatzraten für die Pumpen um. Wird anschließend der Knopf für die Getränkeausgabe gedrückt gehalten steuert der Arduino die Pumpen mit ihrer jeweiligen Durchsatzrate an und der Anwender bekommt sein Mischgetränk ausgegeben.

3.2 Hardware

3.3 Sprachverarbeitung

Im Zuge des technologischen Fortschritts nutzen die Menschen heutzutage zunehmend Sprachassistenten für verschiedene Aufgaben. Einer der Hauptvorteile der Sprachsteuerung ist die Bequemlichkeit und Geschwindigkeit, mit der Aufgaben erledigt werden können, ohne dass man tippen oder mit der Maus klicken muss. Sprachassistenten nutzen die Verarbeitung natürlicher

Sprache, um Befehle zu erkennen und zu verstehen, die der Nutzer laut ausspricht.

Die Verarbeitung natürlicher Sprache - Natural Language Processing (NLP) - ist eine wichtige Technologie, die es Computern ermöglicht, die von Menschen verwendete natürliche Sprache zu verstehen. Diese Technologie ermöglicht es Computern, menschliche Sprache zu erkennen und zu interpretieren und Text und Sprache in natürlicher Sprache zu erzeugen. Im Bereich der Sprachsteuerung spielt NLP eine Schlüsselrolle bei der Erkennung von Sprache und dem Verstehen von Befehlen, die der Benutzer laut ausspricht. Es wird verwendet, um die Sprache des Benutzers in Text umzuwandeln, den ein Computer verstehen und verarbeiten kann. Um dies zu erreichen, verwendet NLP eine Vielzahl von Techniken und Technologien, darunter maschinelles Lernen, Tonanalyse, syntaktische Analyse und mehr. [Jurafsky u. a. 2009]

Im Rahmen dieses Projekts erfordert die Implementierung der Sprachsteuerung einer Getränkemischmaschine die Verarbeitung natürlicher Sprache, damit die Maschine Befehle verstehen kann, die der Benutzer laut ausspricht. Dieses Projekt ähnelt einem Chatbot, bei dem der Benutzer eine Frage stellen oder einen Befehl geben kann und der Chatbot führt die entsprechende Aktion aus. Die Verarbeitung natürlicher Sprache ist für die Entwicklung eines solchen Sprachsteuerungssystems unerlässlich und ermöglicht es der Maschine, Befehle in natürlicher Sprache zu verstehen und auszuführen.

Eine der wichtigsten Komponenten der Verarbeitung natürlicher Sprache ist die Spracherkennung und das Syntaxanalyseverfahren. Bei der Spracherkennung kommen Deep-Learning-Techniken zum Einsatz, die es einem Computer ermöglichen, Sprachlaute zu erkennen und in Text zu übersetzen. Anschließend wird das Syntaxanalyseverfahren verwendet, um die Satzstruktur zu bestimmen und Schlüsselwörter und -sätze hervorzuheben, die zur Bestimmung des Benutzerbefehls verwendet werden können. Auch die Tonwertanalyse ist ein wichtiger Bestandteil der Verarbeitung natürlicher Sprache. Mit Hilfe der Tonalitätsanalyse lässt sich die emotionale Färbung des Textes bestimmen, was für die Ermittlung der Absicht des Nutzers nützlich sein kann. Da die Tonwertanalyse im Rahmen dieser Arbeit nicht relevant ist, wird sie nicht weiter erörtert.

Darüber hinaus werden grammatik- und regelbasierte Technologien zur Verarbeitung natürlicher Sprache eingesetzt. Diese Technologien werden eingesetzt, um die korrekte Struktur des Benutzerbefehls zu bestimmen und Schlüsselwörter hervorzuheben, die zur Durchführung von Aktionen verwendet werden können. Außerdem werden Techniken des maschinellen Lernens eingesetzt, damit der Computer aus früheren Befehlen und Aktionen „lernen“ kann, was die Genauigkeit der Erkennung von Benutzerbefehlen verbessert.

Daher ist der Einsatz von Technologien zur Verarbeitung natürlicher Sprache für das Projekt der Sprachmischmaschine unerlässlich. Die Verarbeitung natürlicher Sprache wird es der Maschine ermöglichen, die Befehle zu verstehen und zu verarbeiten, die der Benutzer laut ausspricht und die entsprechenden Aktionen durchzuführen.

3.3.1 Verarbeitung natürlicher Sprache

Die Verarbeitung natürlicher Sprache ist ein Forschungsbereich der Informatik und der KI, der sich mit der Verarbeitung natürlicher Sprache befasst. Bei der Verarbeitung geht es in der Regel darum, natürliche Sprache in Daten zu übersetzen, die ein Computer nutzen kann, um Informationen über die Welt um ihn herum zu erhalten.

Die NLP-Pipeline, die zur Erstellung eines Dialogsystems erforderlich ist, erfordert vier Arten der Verarbeitung sowie eine Datenbank zur Speicherung vergangener Äußerungen und Antworten. Jeder dieser Schritte kann einen oder mehrere Verarbeitungsalgorithmen enthalten, die parallel oder sequentiell arbeiten [Lane, Howard und Hapke 2019]:

- Syntaktische Zergliederung - Extraktion von Merkmalen (strukturierte numerische Daten) aus natürlichem Text.
- Analyse - Erstellen und Kombinieren von Items, um Indikatoren für den Ton, die grammatikalische Korrektheit und die Semantik des Textes zu erhalten.
- Generierung - Generierung möglicher Antworten mit Hilfe von Mustern, Suchwerkzeugen oder Sprachmodellen.
- Ausführung - bereitet Aussagen vor, die auf der Geschichte und dem Zweck des Gesprächs basieren und wählt eine Folgeantwort.

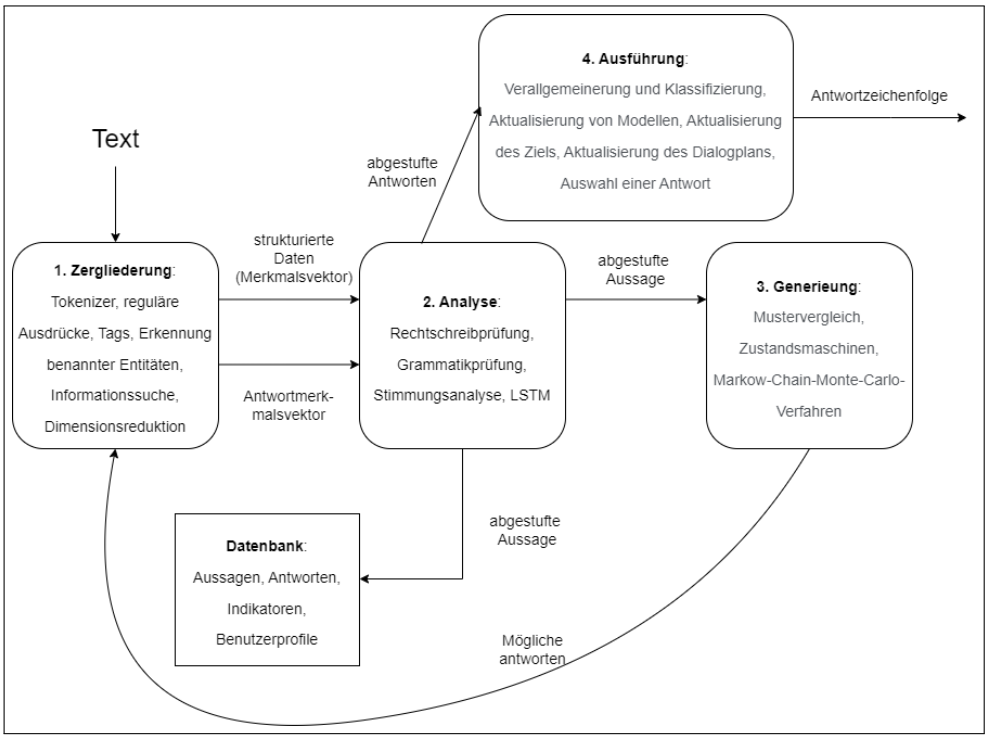

Abbildung 3.2: Rekurrente Chat-Bot-Pipeline

Die meisten Chatbots enthalten Elemente aus allen fünf Teilsystemen (die vier Verarbeitungsstufen sowie die Datenbank). Viele Anwendungen erfordern jedoch nur einfache Algorithmen, um viele dieser Schritte auszuführen. Ein Chatbot oder virtueller Assistent für Verbraucher wie Alexa oder Allo ist in der Regel so konzipiert, dass er äußerst sachkundig und leistungsfähig ist. Die Logik, die zur Beantwortung von Anfragen verwendet wird, ist jedoch oft oberflächlich und besteht aus einer Reihe von Codephrasen, die mit einer einzigen if-then-Entscheidungsverzweigung zur gleichen Antwort führen. Alexa (und die zugrundeliegende Lex-Engine) verhält sich wie ein einschichtiger, flacher Baum von Operatoren (if, elif, elif...). Andererseits stützt sich die Google Translate-Pipeline (oder jedes ähnliche maschinelle Übersetzungssystem) auf eine mehrstufige Hierarchie von Merkmalsextraktoren, Entscheidungsbäumen und Wissensgraphen, die Fragmente von Wissen über die Welt miteinander verbinden.

Alle NLP-Merkmale müssen gut funktionieren, damit das Dialogsystem richtig funktioniert:

- Merkmalsextraktion (normalerweise zur Erstellung eines Vektorraummodells).
- Informationssuche zur Beantwortung von Sachfragen.
- Semantische Suche, um Informationen aus zuvor aufgezeichneten natürlichsprachlichen Texten oder Dialogen zu assimilieren.
- Generierung von natürlicher Sprache, um neue sinnvolle Aussagen zu verfassen.

3.3.2 Tokenisierung von Wörtern

Eine der wichtigsten Aufgaben der Verarbeitung natürlicher Sprache ist die Tokenisierung, d. h. die Zerlegung von Text in einzelne Wörter oder Phrasen, die so genannten Token. Dieser Prozess ist ein wichtiger Bestandteil der Verarbeitung natürlicher Sprache, da er es Programmen und Algorithmen ermöglicht, Texte anhand ihrer Bestandteile zu verstehen und zu analysieren. Die Tokenisierung ist für die Verarbeitung natürlicher Sprache von entscheidender Bedeutung, da sie es Programmen ermöglicht, Schlüsselwörter, Phrasen und semantische Einheiten in einem Text hervorzuheben. Die Tokenisierung kann auch dazu verwendet werden, nicht benötigte Textelemente wie Satzzeichen, Leerzeichen und andere Zeichen, die keine Bedeutung haben, zu entfernen. Dadurch können Programme und Algorithmen effizienter und genauer arbeiten und die Wahrscheinlichkeit von Fehlern und falscher Textverarbeitung verringern.

Im NLP ist die Tokenisierung eine besondere Art der Segmentierung von Dokumenten. Bei der Segmentierung wird der Text in kleinere Abschnitte (Segmente) mit engerem Informationsgehalt unterteilt. Die Segmentierung kann die Aufteilung eines Dokuments in Absätze, in Sätze, in Phrasen und in Token (Wörter) sowie Satzzeichen umfassen. Ein Tokenizer kann mit einem Scanner im Kompilierungsprozess verglichen werden. Token sind in diesem Fall die Endpunkte von kontextfreien Grammatiken - Context-free grammar (CFG) - zur Analyse von Programmiersprachen-Terminalen.

Die Tokenisierung ist der erste Schritt in der NLP-Pipeline und kann daher den Rest der Pipeline stark beeinflussen. Der Tokenizer zerlegt unstrukturierte Daten, d. h. natürlichsprachliche Texte, in Informationseinheiten, die als einzelne Elemente gezählt werden können. Die so gezählte Anzahl von Token-Vorkommen in einem Dokument kann direkt als Vektor verwendet werden, der dieses Dokument repräsentiert. Ein solcher Ansatz ermöglicht es, aus einer unstrukturierten Zeichenfolge (einem Textdokument) unmittelbar eine für das maschinelle Lernen geeignete numerische Datenstruktur zu gewinnen. Diese Werte können den Computer direkt dazu veranlassen, nützliche Aktionen durchzuführen und Reaktionen zu erzeugen.

Die einfachste Art, einen Satz zu tokenisieren, ist die Verwendung von Leerzeichen als Worttrenner in Zeichenketten. Dies ist jedoch nicht optimal, denn wenn ein Satz z. B. ein Satzzeichen enthält, wird es von einem der Tokenisierer erfasst. Optimierte Tokenizer sind in mehreren Python-Bibliotheken implementiert, die jeweils ihre eigenen Vor- und Nachteile haben: spaCy, Stanford CoreNLP und Natural Language Toolkit (NLTK). NLTK und StanfordCoreNLP sind die am längsten bestehenden und am häufigsten verwendeten Bibliotheken zum Vergleich von NLP-Algorithmen in wissenschaftlichen Artikeln. Obwohl die StanfordCoreNLP-Bibliothek eine Python-Application Programming Interface (API) hat, basiert sie auf dem Java 8 CoreNLP-Anwendungsteil, der separat installiert und konfiguriert werden muss. Daher wurde in dieser

Arbeit der NLTK-Tokenizer verwendet.

Ein wichtiges Konzept im Tokenisierungsprozess sind N-Gramme. Ein N-Gramm ist eine Sequenz mit bis zu n Elementen, die aus einer Sequenz dieser Elemente, in der Regel einer Zeichenkette, extrahiert wurden. Im Allgemeinen können die Elemente eines N-Gramms Buchstaben, Silben, Wörter oder sogar Symbole sein. N-Gramme sind notwendig, weil bei der Konvertierung einer Menge von Wörtern in einen Vektor eine Folge von Token einen Großteil der Bedeutung verliert, die in der Reihenfolge dieser Wörter verkapselt ist. Wenn das Token-Konzept auf Mehrwort-Token, N-Gramme, ausgedehnt wird, kann die NLP-Pipeline einen erheblichen Teil der Bedeutung, die in der Wortfolge dieser Äußerungen enthalten ist, beibehalten. So bleibt beispielsweise das Wort „kein“, das die Bedeutung umkehrt, neben den benachbarten Wörtern stehen, wo es hingehört. Ohne N-Gramm-Tokenisierung würde ein solches Wort an verschiedenen Positionen herumhängen und seine Bedeutung würde mit dem gesamten Satz oder Dokument assoziiert werden, anstatt mit benachbarten Wörtern. Das Bigramm „war nicht“ behält viel mehr Bedeutung der einzelnen Wörter „war“ und „nicht“ als die entsprechenden Singlegramme im Multigrammvektor. Durch die Verknüpfung eines Wortes mit seinen Nachbarn in einem Förderband kann ein Teil seines Kontexts erhalten bleiben. N-Gramme sind also ein Instrument zur Speicherung von Kontextinformationen, während die Daten die Pipeline durchlaufen.

Die Größe des Vokabulars spielt eine wichtige Rolle für die Leistung der NLP-Pipeline. Die Größe des Wörterbuchs bestimmt die Größe der Trainingsstichprobe, die benötigt wird, um eine Überanpassung an ein bestimmtes Wort oder eine bestimmte Phrase zu vermeiden und die Größe der Trainingsmenge bestimmt die Kosten der Verarbeitung. Eine Technik zur Verringerung der Größe des Wörterbuchs besteht darin, Token, die ähnliche Dinge bedeuten, in einer einzigen normalisierten Form zusammenzufassen. Eine solche Technik reduziert die Anzahl der gespeicherten Token und verbessert die Verbindungen zwischen den Bedeutungen von Phrasen mit unterschiedlicher „Schreibweise“ der Tokens und verringert die Wahrscheinlichkeit des Überlernens. Eine der Normalisierungsmöglichkeiten ist die Groß- und Kleinschreibung - die Kombination mehrerer Schreibweisen eines Wortes, die sich nur in der Groß- und Kleinschreibung unterscheiden. In diesem Fall wird die Groß-/Kleinschreibung ignoriert und zwei identische Wörter, von denen eines groß und das andere klein geschrieben wird, werden als dasselbe Token behandelt.

Ein weiterer wichtiger Schritt bei der Tokenisierung von Texten ist die Entfernung von Stoppwörtern, um den Umfang des Wörterbuchs zu verringern. Stoppwörter sind in jeder Sprache gebräuchliche Wörter, die sehr häufig vorkommen, aber sehr viel weniger aussagekräftige Informationen über die Bedeutung eines Satzes enthalten (z. B. a, an, the, this, of, on im Englischen oder der, die, das, diese im Deutschen). Stoppwörter können jedoch nützliche Informationen enthalten, so dass man sie nicht immer verwerfen sollte. Das NLTK-Paket für Python enthält

derzeit die umfassendste Liste kanonischer Stoppwörter in verschiedenen Sprachen.

Eine weitere gängige Methode der Normalisierung ist die Beseitigung kleiner semantischer Unterschiede im Zusammenhang mit Pluralendungen und Possessivendungen von Wörtern oder sogar unterschiedlichen Verbformen. Diese Methode der Normalisierung, bei der ein gemeinsamer Wortstamm für verschiedene Wortformen gefunden wird, wird als Stemming bezeichnet. Der gemeinsame Wortstamm von „Gehäuse“ und „Haus“ ist zum Beispiel „Haus“. Beim Stemming werden Suffixe von Wörtern entfernt, um Wörter mit ähnlicher Bedeutung unter einem gemeinsamen Stamm zu gruppieren. Der Wortstamm muss nicht unbedingt ein gültiges Wort sein, sondern kann auch nur ein Token oder eine Bezeichnung sein, das mehrere mögliche Schreibweisen repräsentiert. Einer der Hauptvorteile des Stemming besteht darin, die Anzahl der Wörter zu verringern, deren Bedeutung das Sprachmodell im Auge behalten muss. Durch diese Methode wird die Größe des Wörterbuchs reduziert, wodurch der Verlust nützlicher Informationen und Bedeutungen so weit wie möglich begrenzt wird. Das Stemming spielt eine wichtige Rolle bei der Suche nach Schlüsselwörtern oder Informationen. Zwei der bekanntesten Algorithmen sind Porter's Stemmer und Snowball.

Mit Informationen über die Beziehungen zwischen den Bedeutungen verschiedener Wörter ist es möglich, mehrere Wörter miteinander zu verknüpfen, auch wenn ihre Schreibweise sehr unterschiedlich ist. Eine solche erweiterte Normalisierung eines Wortes auf seine semantische Wurzel - ein Lemma - wird Lemmatisierung genannt. Die Lemmatisierung ist potenziell eine viel genauere Art der Normalisierung als das Stemming oder die Groß- und Kleinschreibung, da sie die Bedeutung des Wortes berücksichtigt. Der Lemmatisierer verwendet eine Wissensbasis von Synonymen und Wortendungen, um nur eng verwandte Wörter zu einem Token zu kombinieren. In Python kann die Lemmatisierung mit dem NLTK-Paket implementiert werden, das den *WordNetLemmatizer* enthält.

Wie bereits gezeigt wurde, ist die Tokenisierung der Prozess der Zerlegung von Text in einzelne Wörter oder Token. Für viele Anwendungen der Verarbeitung natürlicher Sprache ist jedoch nicht nur wichtig, welche Wörter im Text enthalten sind, sondern man muss auch in der Lage sein, diese Wörter als Zahlen darzustellen. Dadurch wird es möglich, maschinelles Lernen und andere Algorithmen, die mit Zahlen arbeiten, zur Textverarbeitung einzusetzen. Bei der Verarbeitung natürlicher Sprache wird dazu die Wortvektorisierung verwendet.

3.3.3 Vektoren von Wörtern

3.3.4 Syntaktische Analyse

3.3.5 Ansätze für die Erstellung eines Chatbots

Derzeit gibt es vier Hauptansätze für die Erstellung eines Chatbots [Lane, Howard und Hapke 2019]:

- Musterabgleich: Musterabgleich und Antwortvorlagen (vorgefertigte Antworten)
- Grounding: logische Wissensgraphen und das Ziehen von Schlussfolgerungen aus diesen basierend auf diesen Graphen
- Suche: Abrufen von Text
- Generierungsmethoden: Statistik und maschinelles Lernen

Die vier grundlegenden Ansätze zur Erstellung von Chatbots lassen sich kombinieren, was zu benutzerfreundlicheren Chatbots führt. Eine Vielzahl von Anwendungen nutzen alle vier grundlegenden Methoden. Hybride Chatbots unterscheiden sich hauptsächlich darin, wie genau sie diese Ansätze kombinieren und wie viel Gewicht auf jeden einzelnen Ansatz gelegt wird.

Musterabgleich

Bei den ersten Chatbots basierte die Antwort auf die Nachricht eines Benutzers auf einem Mustervergleich. Diese Chatbots suchen nach Mustern im eingehenden Text und geben eine feste (gemusterte) Antwort, wenn eine Übereinstimmung gefunden wird [Woudenberg 2014].

Solche rudimentären Dialogsysteme sind vor allem in automatisierten Benutzerunterstützungssystemen mit interaktiven Sprachmenüs nützlich, wo es möglich ist, das Gespräch an einen Menschen weiterzuleiten, wenn der Chatbot keine Antwortmuster mehr hat.

Da es viele NLP-Dienstprogramme in Python-Paketen gibt, ist es möglich, komplexere Chatbots auf der Grundlage von Mustervergleichen zu erstellen, indem man die Bot-Logik nach und nach direkt in Python mit regulären Ausdrücken und Suchmustern aufbaut.

1995 machte sich Richard Wallace daran, einen allgemeinen Rahmen für die Erstellung von Chatbots auf der Grundlage des Pattern-Matching-Ansatzes zu schaffen. Zwischen 1995 und 2002 schuf seine Entwicklergemeinschaft die AIML zur Beschreibung von Mustern und Chatbot-Antworten.

AIML ist eine deklarative Sprache, die auf dem eXtensible Markup Language (XML)-Standard

basiert, der die Sprachkonstrukte und Datenstrukturen einschränkt, die im Bot verwendet werden dürfen. [*AIML Foundation* o. D.] Ein Chatbot, der auf AIML basiert, sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?><aiml version="2.0">
<category>
  <pattern>Hi, Bot</pattern>
  <template>Hallo</template>
</category>
<category>
  <pattern>Wie geht es dir, bot?</pattern>
  <template>Gut</template>
</category>
</aiml>
```

Abbildung 3.3: AIML Chatbot

Eine der Einschränkungen von AIML ist die Art der Muster, die abgeglichen werden können und auf die reagiert wird. Der AIML-Kern (Pattern Matching Engine) reagiert nur auf Eingabetext, der einem vom Entwickler manuell vorgegebenen Muster entspricht. Unscharfe Suchanfragen, Smileys, Satzzeichen, Tippfehler oder falsch geschriebene Wörter sind nicht erlaubt, es findet kein automatischer Abgleich statt. In AIML müssen alle Synonyme manuell einzeln beschrieben werden.

Grounding

Die Grounding-Methode ist ein Ansatz zur Erstellung eines Chatbots auf der Grundlage logischer Wissensgraphen und der Durchführung von Schlussfolgerungen auf der Grundlage dieser Graphen. [Diana und Ismael 2011] Sie wird verwendet, um natürliche Sprache zu verarbeiten und sie dem Verständnis des Bots zuzuordnen. Das Wesentliche an der Grounding-Methode ist, dass der Chatbot nicht nur die Textnachrichten, sondern auch den Kontext und die Umgebung verarbeitet, um Anfragen besser zu verstehen und zu beantworten. Durch die Extraktion von Informationen wird ein Netz von Verbindungen oder Fakten geschaffen. Dieses Netz logischer Verbindungen zwischen Entitäten - ein Graph oder eine Wissensbasis - kann die Grundlage für die Antworten des Chatbots bilden.

Ein Beispiel für eine Grounding-Methode ist die Verwendung eines Wissensgraphen zur Beschreibung der Umgebung. Ein Wissensgraph enthält Informationen über die Objekte, mit denen

der Bot interagieren kann und die Beziehungen zwischen ihnen. Ein Wissensgraph könnte zum Beispiel Informationen über ein Glas auf einem Tisch und das darin befindliche Wasser enthalten. Wenn ein Benutzer eine Frage stellt, verwendet der Chatbot den Wissensgraphen, um den Kontext der Anfrage zu verstehen und die am besten geeignete Antwort abzuleiten. Wenn ein Benutzer zum Beispiel fragt: „Wie hoch ist die Temperatur des Wassers in dem Glas auf dem Tisch?“, kann der Chatbot Informationen aus dem Wissensgraphen verwenden, um die Frage zu beantworten.

Ein solcher Wissensgraph kann abgeleitet werden, um Fragen über die in dieser Wissensbasis enthaltene Welt zu beantworten und anschließend können auf der Grundlage der logischen Antworten die Werte der in den Antworten enthaltenen Template-Variablen ausgefüllt werden, um natürlichsprachliche Antworten zu erstellen. Ursprünglich wurden auf diese Weise Systeme zur Beantwortung von Fragen eingerichtet, wie z. B. der Watson-Bot von IBM (heutzutage wird für ähnliche Systeme jedoch die Informationssuchemethode verwendet). Der Wissensgraph stellt eine Art „Erdung“ des Chatbots in der realen Welt dar.

Die Erstellung von Chatbots auf der Grundlage von „Grounding“ eignet sich hervorragend für Chatbots, die Fragen generieren, bei denen das zur Beantwortung einer Frage erforderliche Wissen in einer umfangreichen Wissensbasis enthalten ist, die aus einer offenen Datenbank (z. B. Wikidata, Open Mind Common Sense oder DBpedia) bezogen werden kann.

Einer der Hauptvorteile der Grounding-Methode besteht darin, dass sie sich an ein sich veränderndes Umfeld anpassen kann. Wenn der Benutzer zum Beispiel ein Glas Wasser von einem Tisch auf einen anderen stellt, wird der Wissensgraph automatisch aktualisiert, um diese Änderung widerzuspiegeln.

Die Grounding-Methode hat jedoch auch ihre Grenzen. So kann es vorkommen, dass bei der Verarbeitung großer Informationsmengen Zusammenhänge nicht berücksichtigt werden und dem Bot möglicherweise verborgen bleiben.

Insgesamt ist die Grounding-Methode ein effektiver Ansatz zur Erstellung wissensbasierter Chatbots. Sie ermöglicht es dem Bot, Benutzeranfragen besser zu verstehen und eine genauere Antwort zu geben.

Suche

Die Informationssuchemethode ist eine der Methoden zum Aufbau von Chatbots, die auf der Extraktion von Informationen aus einer großen Menge von Textinformationen basiert. Die Hauptidee der Informationssuchemethode ist die Analyse des Eingabetextes (Benutzeranfrage), die

Auswahl von Schlüsselwörtern und Phrasen daraus und die anschließende Suche nach den relevantesten Informationen in der Wissensdatenbank oder in offenen Quellen. [Diana und Ismael 2011]

Die Wissensbasis kann auch eine Art „Gesprächsprotokoll“ sein, in Form von Aussage-Antwort-Paaren. Dabei sucht der Bot nach früheren Aussagen in den Protokollen früherer Unterhaltungen. Der Bot kann nicht nur in den Protokollen seiner eigenen Gespräche suchen, sondern auch in beliebigen Transkripten von Gesprächen zwischen Menschen, Gesprächen zwischen Menschen und Bots oder sogar Gesprächen zwischen Bots. Aber wie immer gilt: je besser die Eingabedaten, desto besser das Ergebnis. Daher ist es notwendig, die Datenbank früherer Gespräche sorgfältig zu säubern und zu organisieren, damit der Bot nach einem qualitativ hochwertigen Gespräch sucht und es dann imitiert.

Für die Umsetzung der Informationssuchemethode werden verschiedene Algorithmen und Techniken verwendet, z. B. Indizierung und Schlagwortsuche, Kontextsuche, Textanalyse mit Hilfe von maschinellen Lernverfahren usw. Die Informationssuchemethode kann in Python mit verschiedenen Bibliotheken und Tools wie NLTK, Scikit-learn und Gensim implementiert werden.

Einer der ersten Schritte bei der Implementierung einer Informationssuchemethode in Python ist die Vorbereitung der Daten. Dies erfordert Tokenisierung, Lemmatisierung und die Entfernung von Stopp-Wörtern. Als nächstes muss ein Index auf der Grundlage von Schlüsselwörtern erstellt werden. Der Index kann auf der Grundlage von Bag-of-Words oder Term Frequency (TF) und Inverse Document Frequency (IDF) (TF-IDF-Modelle) erstellt werden. Sobald der Index erstellt ist, kann eine Stichwortsuche durchgeführt werden. Dazu muss die Benutzeranfrage in einen Vektor umgewandelt und mit den Dokumentvektoren im Index verglichen werden. Dies kann mit Hilfe der Scikit-learn-Bibliothek erfolgen [scikit-learn o. D.] Sobald die relevantesten Dokumente gefunden wurden, können sie in eine Rangfolge gebracht und als Antwort auf die Benutzeranfrage angezeigt werden.

Der Vorteil der Informationssuchemethode besteht darin, dass sie ein schnelles und genaues Auffinden der gewünschten Informationen ermöglicht, insbesondere wenn die Wissensbasis gut strukturiert ist und genügend Informationen enthält. Ein Nachteil dieser Methode ist jedoch, dass sie den Kontext der Anfrage nicht berücksichtigt und nicht immer eine vollständige und genaue Antwort auf die Frage des Nutzers liefert. Wenn die Aussage semantisch mit der vom Bot zu beantwortenden übereinstimmt, ist es möglich, die Antwort wortwörtlich und ohne Änderungen wiederzuverwenden. Aber selbst wenn die Datenbank alle möglichen Benutzeräußerungen enthält, wird der Bot die Persönlichkeiten der Personen widerspiegeln, die diese Äußerungen machen. Wenn die Antworten konsistent sind und von einer Vielzahl von Personen stammen, ist das gut. Problematisch wird es jedoch, wenn die Äußerung, auf die der Bot reagieren soll, vom

Gesamtkontext des jeweiligen Gesprächs oder von den Umständen in der Umgebung abhängt, die sich seit der Erstellung des Dialogkorpus geändert haben können.

Beispielsweise sollte der Bot auf die Frage „Wie spät ist es?“ nicht die von der Person gegebene Antwort, sondern die am besten geeignete Aussage aus der Datenbank verwenden. Diese Antwort funktioniert nur, wenn die Zeit, zu der die Frage gestellt wurde, mit der Zeit übereinstimmt, zu der die passende Äußerung aus der Datenbank aufgezeichnet wurde. Neben dem natürlichsprachlichen Text der Äußerung müssen auch ähnliche Informationen über die Zeit - der Kontext (Zustand) - erfasst und verglichen werden. Sie spielt vor allem dann eine wichtige Rolle, wenn die Semantik der Äußerung auf eine aktive Veränderung des im Kontext (Wissensbasis des Chatbots) erfassten Zustands hinweist.

Um den Zustand (Kontext) in einem Chatbot auf der Grundlage der Informationssuche zu berücksichtigen, kann etwas Ähnliches für einen Chatbot mit Musterabgleich durchgeführt werden, da die Auflistung einer Liste von Benutzeraussagen nur eine andere Art ist, ein Muster zu beschreiben. Dies auch ist der Ansatz von Amazon Lex [AMAZON o. D.] und Google Dialogflow [CHAWLA o. D.] Anstatt ein starres Muster zu beschreiben, um den Befehl des Benutzers zu erfassen, können der Dialogflow-Engine einfach ein paar Beispiele geliefert werden. So wie jedes Muster im Chatbot auf der Grundlage der Musterzuordnung einem Zustand zugeordnet wurde, muss auch hier nur die Aussage-Antwort-Beispielpaare mit dem genannten Zustand verknüpft werden.

Der suchbasierte Chatbot indiziert also den Korpus der Dialoge, so dass er leicht frühere Aussagen finden kann, die derjenigen ähnlich sind, auf die er antworten muss und antwortet dann mit einer der passenden Aussagen aus dem Korpus, die er sich „gemerkt“ und für eine schnelle Suche indiziert hat. Im Allgemeinen ist die Methode der Informationssuche eine der gängigsten und beliebtesten Methoden zum Aufbau von Chatbots, die in verschiedenen Bereichen wie Wirtschaft, Medizin, Tourismus und vielen anderen eingesetzt werden.

Generierungsmethoden

Generierungsmethoden sind einer der wichtigsten Ansätze bei der Entwicklung von Chatbots auf der Grundlage künstlicher Intelligenz. Sie ermöglichen es Chatbots, Textantworten auf der Grundlage der Analyse der eingehenden Nachricht und des Kontextes des Dialogs zu generieren. Die folgenden Generierungsmodelle sind nützlich, um einen kreativen Chatbot zu erstellen, der Dinge sagen kann, die noch niemand zuvor gesagt hat:

- Sequenz-zu-Sequenz-Konvertierungsmodelle: Modelle, die darauf trainiert sind, Antworten auf der Grundlage von Eingabesequenzen zu generieren;

- Restricted Boltzmann Machines (RBM): Markov-Ketten, die so trainiert werden, dass sie die „Energie“-Funktion minimieren [Nupur Sharma o. D.];
- Generative Adversarial Network (GAN): statistische Modelle, die darauf trainiert sind, einen Experten, der die Qualität eines Gesprächs bewertet, zu täuschen. [Li u. a. 2017]

Die Vorteile des Einsatzes der Generierungsmethoden:

- Flexibilität: Generative Methoden können für eine breite Palette von Aufgaben eingesetzt werden, einschließlich Texterstellung, Sprachübersetzung, Verarbeitung natürlicher Sprache und mehr.
- Automatisierung: Generative Methoden können auf großen Datensätzen trainiert werden, wodurch die Erstellung von Inhalten automatisiert werden kann.
- Qualität: Generative Methoden zeigen eine hohe Qualität bei der Textgenerierung, Sprachübersetzung und anderen Aufgaben der natürlichen Sprachverarbeitung, wenn sie auf einem ausreichend großen Datensatz trainiert werden.
- Schnelligkeit: Generative Methoden können schneller arbeiten als Menschen, was die Erstellung von Inhalten mit großer Geschwindigkeit ermöglicht.

Die Nachteile der generativen Methoden:

- Große Datenmengen für das Training: Generative Methoden benötigen große Datenmengen für das Training, was bei einigen Aufgaben schwierig sein kann, insbesondere wenn nur ein kleiner Datensatz zur Verfügung steht.
- Sicherheitsrisiken: Generative Methoden können Inhalte erzeugen, die möglicherweise falsch, unvollständig oder irreführend sind. Dies kann zu Sicherheitsrisiken führen, wenn der generierte Inhalt für wichtige Entscheidungen verwendet wird.
- Unterstützungsbedarf: Generative Methoden können erhebliche Unterstützung benötigen, um effektiv zu sein. Dies kann die Modellabstimmung, die Auswahl optimaler Parameter und die Optimierung der Modellleistung auf einer bestimmten Hardwarekonfiguration umfassen.
- Modellbeschränkungen: Generative Methoden können Beschränkungen hinsichtlich der Arten von Inhalten haben, die sie erzeugen können, insbesondere wenn sie nur auf bestimmte Datentypen trainiert wurden.

Eine der beliebtesten Methoden zur Texterstellung ist die sequence-to-sequence-Methode (seq2seq). Die seq2seq-Methode basiert auf Recurrent Neural Networks (RNN), die die Simulation von Datenfolgen ermöglichen. Sie besteht aus zwei Hauptteilen: einem Encoder und einem Decoder. Ein

Encoder empfängt eine Wortfolge und baut daraus einen Kontextvektor auf, der Informationen über die Eingabedaten enthält. Der Decoder erhält diesen Vektor als Eingabe und beginnt mit der Generierung einer Folge von Antwortnachrichten, wobei er schrittweise den Kontext und die zuvor generierten Wörter berücksichtigt. [Alammar o. D.]

Einer der Hauptvorteile der seq2seq-Methode ist ihre Fähigkeit, qualitative und grammatikalisch korrekte Textantworten zu generieren, einschließlich Antworten, die nicht in den Trainingsdaten enthalten waren. Sie kann auch mit langen Sequenzen umgehen, was sie ideal für die Generierung von Antworten in Dialogsystemen macht. Darüber hinaus kann die seq2seq-Methode in einer Vielzahl von Anwendungen eingesetzt werden, z. B. in der maschinellen Übersetzung, der Spracherkennung und anderen.

Die seq2seq-Methode hat jedoch ihre Nachteile. Sie erfordert große Datenmengen zum Trainieren und Verarbeiten sowie erhebliche Rechenressourcen. Dies kann die Anwendung der Methode bei einigen Anwendungen einschränken. Wenn der Trainingsdatensatz nicht eine ausreichend große Bandbreite möglicher Antworten repräsentiert, kann das Modell außerdem dazu neigen, vorhersehbare oder falsche Antworten zu erzeugen.

Die Implementierung der seq2seq-Methode in Python kann mit der TensorFlow-Bibliothek erfolgen, die eine Reihe von Werkzeugen für den Aufbau und das Training neuronaler Netze bietet. In TensorFlow kann man die vortrainierten seq2seq-Modelle verwenden oder ein eigenes Modell erstellen, indem die Architektur und die Trainingsparameter des Netzwerks konfiguriert wird. [TensorFlow o. D.[b]]

Kapitel 4

Konzept

In diesem Kapitel wird zunächst erläutert, wie die Steuerung der Getränkemischmaschine durch Sprachbefehle im Allgemeinen ablaufen wird. Anschließend werden mehrere Konzepte vorgestellt, die das allgemeine Konzept konkretisieren. Diese werden anhand der, in Kapitel 4.2 erläuterten Kriterien, bewertet. Zuletzt wird die Wahl des finalen Konzepts begründet.

4.1 Allgemein

Der Benutzer soll über Spracheingaben mit der Mischmaschine interagieren können. Dafür muss das Gesprochene zunächst durch ein Mikrofon aufgenommen werden. Anschließend können die Audiosignale weiterverarbeitet werden. Der Benutzer soll hierbei nicht auf fest vorgegebene Sprachbefehle beschränkt sein, sondern für nahezu jede Eingabe eine sinnvolle Antwort zurückerhalten. Um dies zu gewährleisten wird die Spracheingabe durch ein Sprachmodell, welches mittels maschinellen Lernverfahren trainiert wurde, verarbeitet. Ergebnisse dieser Verarbeitung sind die Antwort, die an den Benutzer zurückgegeben wird, und ein konkreter Befehl für die Mischmaschine. Ein Beispiel für einen solchen Befehl könnte etwa die Zubereitung eines bestimmten Getränks sein. Für die Ausgabe einer Antwort ist ein Lautsprecher notwendig. Denkbar wäre auch eine textbasierte Ausgabe, allerdings ginge damit der Eindruck des Benutzers verloren eine echte Konversation mit der Mischmaschine zu führen. Das Sprachmodell mit der Getränkemischmaschine zu verknüpfen stellt eine Herausforderung dieser Arbeit dar.

4.2 Bewertungskriterien

Im Folgenden sind die Bewertungskriterien für die einzelnen Konzepte aufgelistet:

- Freiheitsgrade in der Spracheingabe des Benutzers: Erhält der Benutzer passende Antworten zurück egal was er sagt oder ist er auf einige wenige Befehle beschränkt?

- Hardwarekosten: Wie kostspielig ist das Konzept bezüglich der zusätzlich benötigten Hardware?
- Verfügbare Rechenleistung: Wie hoch ist die Verfügbare Rechenleistung im Vergleich zu den anderen Konzepten? Reicht diese aus um das Sprachmodell auszuführen?
- Performanz: Als wie performant wird die Lösung eingeschätzt? Ist mit Latenzen zwischen der Einagbe des Benutzers, der Ausgabe einer Antwort und Ausführung der Aktion zu rechnen?
- Overhead: Wie hoch ist im Allgemeinen der Mehraufwand einzuschätzen?

4.3 Konzept A: Spracherkennung und -verarbeitung mittels Arduino

Ein erstes Konzept sieht vor, dass das Audiosignal direkt von einem der Arduinos in der Getränkemischmaschine aufgenommen wird. Das Audiosignal wird vom Arduino interpretiert und eine passende Antwort wird ausgegeben. Außerdem sendet der Arduino die entsprechenden Signale, um die vom Benutzer gewünschte Aktion von der Getränkemischmaschine ausführen zu lassen. Ein Problem ist hierbei die Interpretation des Audiosignals durch den Arduino, da dessen Leistung nicht für das Ausführen eines Sprachmodells ausreicht. Folglich muss dieser Prozess ausgelagert werden. Das Konzept wird deshalb um ein cloudbasiertes Sprachverarbeitungssystem ergänzt, welches den Sprachbefehl des Benutzers vom Arduino entgegennimmt und einen passenden Befehl und eine passende Antwort zurückgibt (s. Abb. 4.1). Die Kommunikation zwischen Arduino und Cloudsystem kann über das Hypertext Transfer Protocol (HTTP) erfolgen.

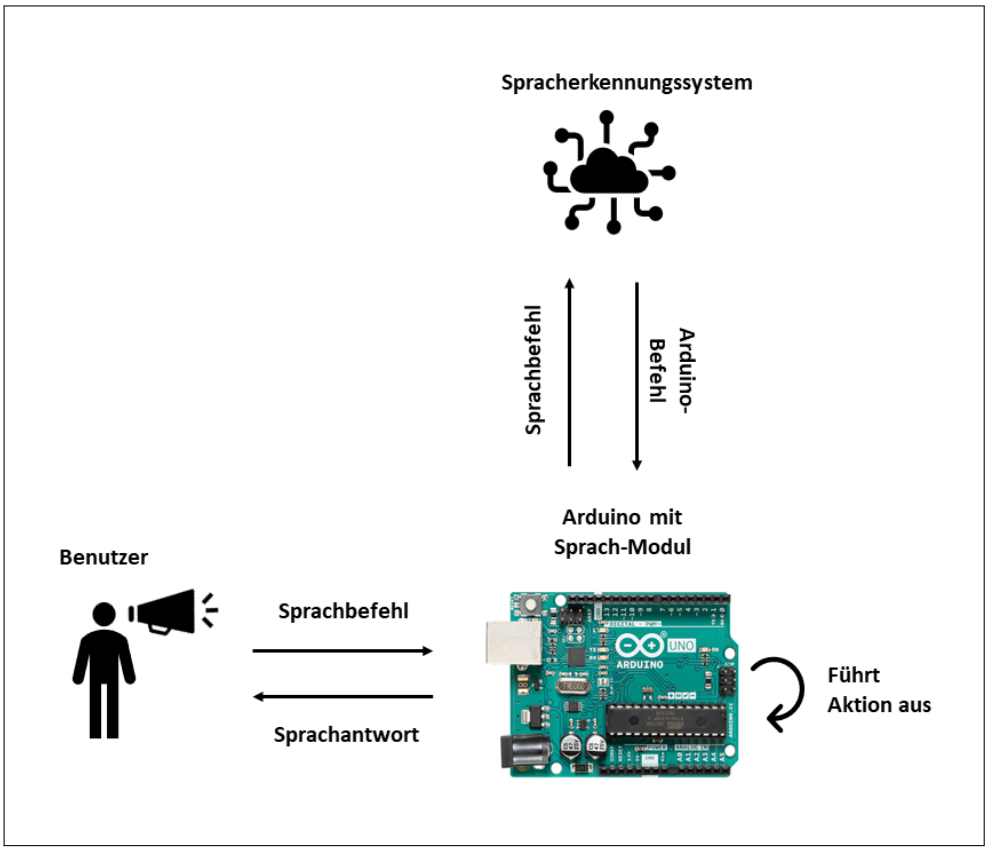

Abbildung 4.1: Spracherkennung und -verarbeitung mittels Arduino

Es muss ein geeignetes Format zum Versenden des Sprachbefehls über HTTP gefunden werden. Eine Möglichkeit besteht darin, das eingehende Audiosignal im Arduino in textform umzuwandeln und diesen String zu versenden. Die auf dem Markt verfügbaren Arduino-Sprach-Module sind jedoch nicht in der Lage beliebige Spracheingaben in Text umzuwandeln, sondern bieten diese Funktionalität nur für vordefinierte Werte an. Dies würde das Ziel dieser Arbeit verfehlen, dem Benutzer eine Konversation mit der Mischmaschine zu ermöglichen. Ein weiteres Problem dieser Lösung besteht darin, dass beispielsweise bei wechselnder Getränkeauswahl die zur Verfügung stehenden Sprachbefehle wie „Ich hätte gerne Getränk xy“ jedes Mal aufs neue manuell angepasst werden müssten. Dies hat zur Folge, dass auch die reine Spracherkennung aus der Mischmaschine ausgelagert werden muss. Ein denkbares Format sind die rohen Audiosignale, die vom Arduino aufgenommen werden.

4.4 Konzept B: Spracherkennung und -verarbeitung mittels mobiler Anwendung

Die Audiosignale über ein Mikrofon in der Mischmaschine aufzunehmen und eine Antwort über einen Lautsprecher auszugeben, so wie es in Konzept A der Fall ist, kann ein Problem darstellen. Zum Einen wird dadurch zusätzliche Hardware benötigt und zum Anderen muss diese korrekt verbaut werden. Das Tonsignal muss vom Mikrofon in einer guten Qualität aufgenommen werden können und die Antwort aus dem Lautsprecher für den Benutzer verständlich sein. Konzept B umgeht dieses Problem durch den Einsatz einer mobilen Anwendung, die durch den Benutzer installiert wird. Über diese Anwendung können anschließend die Aufnahme der Audiosignale, die Spracherkennung und die Kommunikation mit dem Sprachverarbeitungsservice und der Mischmaschine abgewickelt werden, wie in Abbildung 4.2 zu sehen ist.

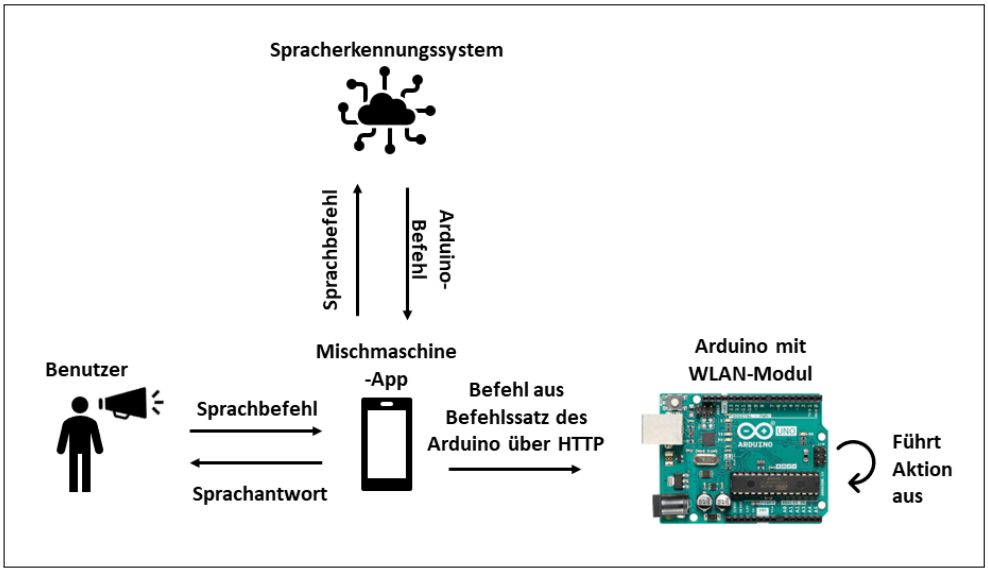

Abbildung 4.2: Spracherkennung und -verarbeitung mittels mobiler Anwendung

Ein Problem dieser Lösung ist der offensichtliche Mehraufwand durch die Entwicklung einer eigenen Anwendung für Mobiltelefone. Auch der Anwender hat zusätzlichen Aufwand durch die Installation. Außerdem ist die Spracheingabe und -ausgabe über das Mobiltelefon nicht intuitiv, da der Anwender eigentlich mit der Maschine kommunizieren sollte. Dieser Effekt kann dadurch abgeschwächt werden, dass wenigstens die Antwort durch einen Lautsprecher in der Mischmaschine an den Benutzer zurückgegeben wird.

4.5 Konzept C: Spracherkennung und -verarbeitung auf Computer-Hardware

Ein weiteres Konzept stützt sich auf die Verwendung eines Computers in der Mischmaschine anstelle eines Mikrocontrollers wie dem Arduino. Motivation ist hierbei der Leistungsgewinn gegenüber eines Mikrocontrollers, um die Spracherkennung und -verarbeitung mittels Sprachmodell zu gewährleisten. Ein Beispiel für einen solchen Miniaturcomputer ist der Raspberry-Pi. Dieser bietet genügend Schnittstellen, wie etwa USB-Hubs, zum verbinden von Mikrofon als auch Lautsprecher. Nimmt der Computer das Audiosignal auf verarbeitet er dieses und generiert daraus die Antwort, die durch den Lautsprecher ausgegeben wird, zusammen mit der Aktion für die Getränkemischmaschine. Diese muss an den Arduino, welcher die Mischmaschine steuert, übermittelt werden. Um dies zu ermöglichen können der Computer und der Arduino über eine serielle Schnittstelle, wie etwa einem USB-Kabel, miteinander verbunden werden. Abbildung 4.3 stellt den konzeptionellen Aufbau graphisch dar.

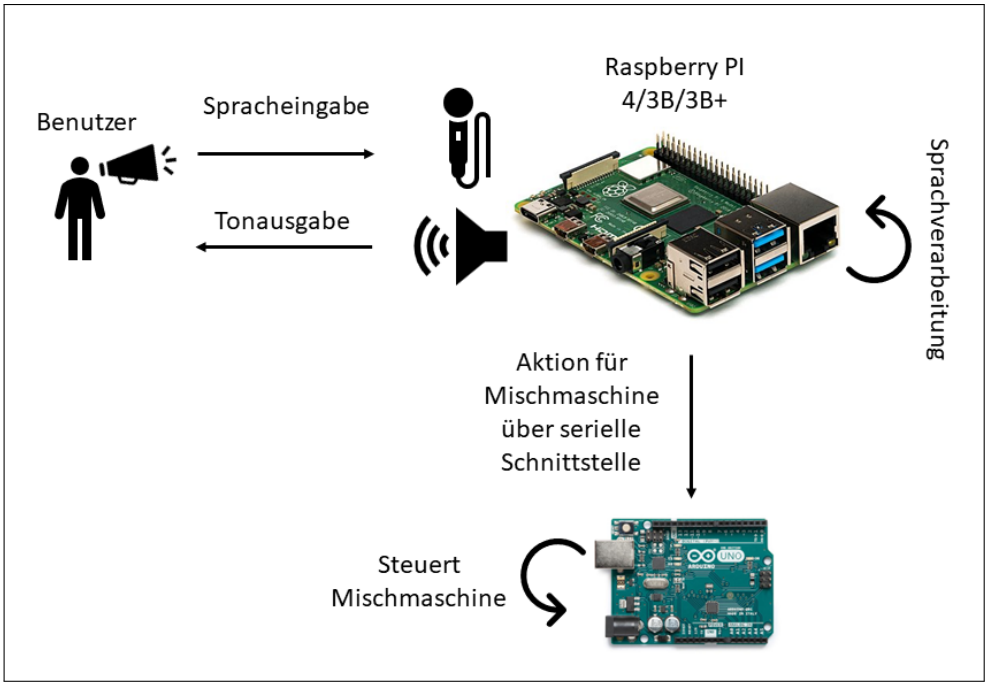

Abbildung 4.3: Spracherkennung und -verarbeitung auf Computer-Hardware

Obwohl ein Computer wie der Raspberry-Pi im Allgemeinen eine höhere Leistung als ein Mikrocontroller hat ist damit nicht sichergestellt, dass diese zur Ausführung des Sprachmodells ausreicht. Beispielsweise ist das vierte Modell der Raspberry-Pi-Serie mit nur maximal acht Gigabyte Arbeitsspeicher erhältlich. Das Sprachmodell könnte allerdings noch weitaus mehr Daten im Arbeitsspeicher benötigen. Des Weiteren ist zu beachten, dass die Miniaturcomputer von Raspberry-Pi im Speziellen zum Zeitpunkt dieser Arbeit kaum zu vertretbaren Preisen

verfügbar sind.

4.6 Finales Hardware-Konzept

Die folgende Tabelle zeigt eine Übersicht bezüglich der Bewertung der einzelnen Konzepte.

Bewertungsmatrix	Konzept A (nur Arduino)	Konzept A	Konzept B	Konzept C
Freiheitsgrade in der Spracheingabe des Benutzers	niedrig	sehr hoch	sehr hoch	hoch
Hardwarekosten	niedrig	hoch	niedrig	sehr hoch
Verfügbare Rechenleistung	niedrig	sehr hoch	sehr hoch	hoch
Performanz	sehr hoch	mittel	mittel	sehr hoch
Overhead	niedrig	hoch	sehr hoch	niedrig

Tabelle 4.1: Bewertung der Konzepte

Ein erster Ansatz wurde im Kapitel 4.3 zu Konzept A erläutert. Hierbei war die Idee, die Spracherkennung und -verarbeitung nur auf dem Arduino auszuführen. Aufgrund der mangelnden Leistung eines Arduinos können mit Hilfe von Sprachmodulen jedoch nur vordefinierte Sätze oder Wörter erkannt werden. Damit ist der Freiheitsgrad in der Spracheingabe des Benutzers äußerst eingeschränkt. Dafür sind die aufzuwendenden Hardwarekosten minimal. Lediglich das Sprachmodul sowie ein Lautsprecher müssten besorgt werden. Die Performanz wird als sehr hoch eingeschätzt, da die Spracherkennung direkt auf dem Ardunino erfolgen kann, der auch die Mischmaschine steuert. Niedrig ist hingegen der benötigte Mehraufwand, da kaum zusätzliche Awendungen, Services oder Hardwarekomponenten benötigt werden.

Das Konzept wurde schließlich durch einen Sprachverarbeitungsservice in der Cloud ergänzt (Konzept A). Der Benutzer hat bei diesem Ansatz große Freiheiten in seinen Formulierungen, da es kein Problem darstellt ein großes Sprachmodell in der Cloud auszuführen. Die Hardwarekosten könnten allerdings hoch sein, je nach dem, ob der Server selbst bereitgestellt oder von einem externen Anbieter bezogen wird. Auch die letzten Endes tatsächlich benötigte Rechenleistung spielt dabei eine Rolle. Die verfügbare Rechenleistung ist theoretisch unbegrenzt, wobei die Performanz des Gesamtsystems nur als mittelmäßig eingestuft werden kann. Nach der Aufnahme und eventuell einer Vorverarbeitung des Aduiosignals durch den Arduino müssen HTTP-Nachrichten gesendet und Empfangen werden. Je nach Last auf dem Netzwerk kann es dadurch zu Latenzen oder sogar Verbindungsabbrüchen kommen. Außerdem ist der Overhead durch den Einsatz einer Cloud recht hoch.

Konzept B unterscheidet sich in Sachen Freiheitsgrad, Rechenleistung und Performanz nicht von Konzept A, da auch hier eine Cloud zum Einsatz kommt. Die Hardwarekosten sind jedoch niedriger, da immerhin kein Sprachmodul, Mikrofon und Lautsprecher benötigt werden. Die Aufgaben dieser Komponenten kann das Mobiltelefon des Anwenders übernehmen. Der Overhead ist deutlich größer, da das Konzept die Entwicklung einer eigenen Mobilanwendung voraussetzt.

Der Freiheitsgrad wird bei Konzept C als hoch, jedoch nicht als sehr hoch, bewertet. Grund hierfür ist die, im Vergleich zur Cloud, etwas beschränkte Leistung, welche die Leistung/Größe des Sprachmodells beeinträchtigen könnte. Hardwarekosten können jedoch sehr hoch werden. Bei dem Einsatz einer Cloud kann ein günstiger Anbieter gefunden werden, sodass die Anschaffung eigener Hardware entfällt. Dies ist hier nicht der Fall. Die Performanz des Gesamtsystems kann, wie bei Konzept A (nur mittels Arduino), sehr hoch eingeschätzt werden, da Spracherkennung und -verarbeitung direkt in der Mischmaschine von der Hardware übernommen wird. Der Mehraufwand ist gering, da weder ein Cloudservice noch eine externe Anwendung entwickelt werden müssen.

Letzten Endes wurde das Konzept C als finales Hardware-Konzept gewählt. Wie ein Blick in die Bewertungsmatrix zeigt lässt sich dies mit den sehr guten Werten, die sich aus der Betrachtung der beschriebenen Bewertungskriterien ergaben, begründen.

4.7 Konzept für die Sprachsteuerung

4.7.1 Ansatz für das Dialogsystem

Aus dem Vergleich der wichtigsten Ansätze zur Erstellung von Chatbots, die in Kapitel 3.3.5 besprochen wurden, lassen sich die folgenden Vor- und Nachteile der einzelnen Methoden ableiten:

Ansatz	Vorteile	Nachteile
Musterabgleich	• Einfacher Einstieg • Leicht wiederverwendbar • Modularität • Leicht zu kontrollieren/ einzuschränken	• Themenbereich begrenzt • Die Möglichkeiten sind durch die Arbeitsbelastung des Entwicklers begrenzt • Komplexität der Fehlersuche • Strenge und „spröde“ Regeln
Grounding	• Gut im Beantworten logischer Fragen • Leicht zu kontrollieren/ einzuschränken	• Künstlicher, mechanischer Ton • Probleme mit Zweideutigkeiten • Probleme mit dem Allgemeinwissen • Begrenzt auf strukturierte Daten • Erfordert die Extraktion von Informationen in großem Umfang • Erfordert menschliche Aufsicht
Suche	• Einfachheit • Leicht zu lehren • Simulation von menschlicher Konversation	• Unzureichende Skalierung • Die simulierte Persönlichkeit des Bots ist inkonsistent • Kennt den Kontext nicht • Keine sachlichen Fragen
Generierungs-methoden	• Neue, kreative Dialoge • Weniger Arbeit für den Entwickler • Kontextsensitiv	• Schwierig zu lehren • Erfordert mehr Daten (Dialoge) • Schwierig, in die richtige Richtung zu lenken • Erfordert mehr Rechenleistung

Tabelle 4.2: Bewertung der Ansätze für die Erstellung eines Dialogsystems

Bei der Analyse des Problems, ein Dialogsystem für eine Getränkmischmaschine zu entwickeln, kann man zu dem Schluss kommen, dass die beste Option eine Mischung aus dem Ansatz der Informationssuchemethode und dem Musterabgleich ist.

Das erste Argument, das für diesen Ansatz spricht, ist die Möglichkeit, die Sprachsteuerung in deutscher Sprache zu verwenden, was die Anwendung der Generierungsmethoden erschwert, da der Zugang zu einer geeigneten Datenbank in dieser Sprache, die den Anforderungen des Projekts, nämlich eine ausreichende Anzahl von Beleidigungen zu enthalten, nicht möglich ist. Gleichzeitig können bei der Verwendung von Musterabgleich Antwortvorlagen und Muster für

entsprechende Anfragen in deutscher Sprache im Voraus erstellt werden, was die Erstellung und das Training des Dialogsystems erleichtert.

Das zweite Argument ist, dass in diesem Projekt ein Raspberry Pi verwendet wird, was die Verwendung generativer Methoden aufgrund der begrenzten Hardware-Ressourcen einschränken kann. Andererseits kann das Modell für die Informationssuche und den Musterabgleich auf Geräten mit geringem Stromverbrauch implementiert werden, was diesen Ansatz für dieses Projekt vorteilhaft macht.

Generierungsmethoden könnten auch für dieses Problem zu mächtig sein. Beim Mixen von Cocktails können die Antworten einfach und formelhaft sein, wie z. B. „Das hört sich eklig an, bist du sicher, dass du es willst?“ oder „Ich hoffe, ich sehe dich nie wieder“. In diesem Fall kann der Einsatz von Generierungsmethoden wie seq2seq-Modellen überflüssig und ineffizient sein. Für diese Aufgabe ist im Gegensatz zu komplexen natürlichsprachlichen Abfragen keine detaillierte semantische Verarbeitung erforderlich, so dass der Musterabgleich einen einfacheren und effizienteren Ansatz darstellt.

Bei der Verwendung des Musterabgleiches kann man den Ton und den Humor des Geräts leicht steuern und so die gewünschte Atmosphäre erzeugen. Die Antworten des seq2seq-Modells sind wiederum sehr schwer zu steuern. Generierungsmethoden können zu unerwünschtem Maschinenverhalten führen, wenn das Modell auf ungeeigneten Daten trainiert wird oder Fehler in der Betriebslogik enthält. Die Verwendung einer Datenbank, die genügend Beleidigungen enthält (z. B. die 4chan-Datenbank), kann dazu führen, dass die Antworten der Maschine über das Ziel hinausschießen und statt lustig zu sein, den Benutzer beleidigen.

Mit diesem Ansatz wird auch die Effizienz der Maschinensteuerung verbessert. Beim Musterabgleich kann eine Kategorie „Getränkebestellung“ zugewiesen werden, die, wenn sie erkannt wird, die Maschine automatisch zur Bearbeitung der Befehle veranlasst. Im Falle von seq2seq-Modell ist keine Kategorie vorgesehen, so dass eine zusätzliche Prüfung jeder Eingabeanweisung eingeführt werden müsste, um festzustellen, wann der Benutzer die Bestellung aufgegeben hat.

Ein letztes Argument, das für den Ansatz spricht, ist die Möglichkeit, die Maschine bei Bedarf schnell an neue Anfragen und Anforderungen anzupassen, indem neue Muster und Regeln in das System eingeführt werden. Daher ist eine Mischung aus Informationssuchemethode und Musterabgleich für das vorliegende Problem am besten geeignet.

4.7.2 Befehle

Kapitel 5

Implementierung

5.1 Implementierung des Sprachverarbeitungssystems

5.1.1 Word2Vec-Modell

5.1.2 Sequence-to-Sequence-Modell

5.2 Implementierung der Sprachsteuerung

Im Folgenden wird erläutert, wie die Sprachsteuerung für die Getränkemischmaschine implementiert wurde und welche Technologien dafür zum Einsatz kamen. Dabei wird zunächst auf die Spracherkennung d.h., die Umwandlung der Audiosignale (Sprachbefehl des Benutzers) in eine Form, die innerhalb des Quelltextes weiterverarbeitet werden kann, eingegangen (s. Abschnitt 5.2.1). Danach wird die Anbindung des Sprachverarbeitungssystems beschrieben, dessen Implementierung in Abschnitt 5.1 erklärt wird. Abschließend wird die Kommunikation mit der Mischmaschine über den Arduino illustriert (s. Abschnitt 5.2.3).

5.2.1 Spracherkennung

Die Spracherkennung ist der erste Schritt bei der Implementierung einer Sprachsteuerung für die Getränkemischmaschine. Mit Spracherkennung ist die Aufnahme eines Tonsignals über ein Audio-Eingabegerät (Mikrofon) und die Umwandlung der Audiodaten in Text gemeint. Der Quelltext zur Implementierung der Sprachsteuerung erfolgt mit der Programmiersprache *Python*, da hier sehr viele, leicht zu bedienende Bibliotheken zur Spracherkennung, -verarbeitung und KI zur Verfügung stehen.

Für dieses Projekt viel die Wahl auf das Paket *SpeechRecognition*, das die Verwendung verschiedener Spracherkennungsdienste über eine einheitliche Schnittstelle ermöglicht und zu diesem Zweck auch zur Aufnahme und Verarbeitung der Audiosignale verwendet werden kann [Zhang (Uberi)

o. D.] Ein großer Vorteil davon ist, dass dadurch ein schneller Wechsel der eingesetzten API erfolgen kann, sollte dies erforderlich sein. Die Verwendung des *SpeechRecognition*-Pakets findet fast ausschließlich über die *Recognizer*-Klasse statt. Um Audiosignale über eine physische Audioquelle (bspw. ein Mikrofon am Computer) aufzunehmen kann die *Microphone*-Klasse verwendet werden, die ebenfalls im Paket enthalten ist. Mit Hilfe eines Objekts vom Typ *Microphone* und der Methode *listen* der *Recognizer*-Klasse können anschließend Audiodaten aufgenommen werden, die in einem Objekt vom Typ *AudioData* gespeichert sind. Die Verwendung von *Recognizer* und *Microphone* sind in Listing 5.1 zu sehen.

```
1 import speech_recognition as sr

3 r = sr.Recognizer()
4 m = sr.Microphone()

6 with m as source:
7     audio = r.listen(source)
```

Listing 5.1: Audioaufnahme mit *SpeechRecognition*

Das *AudioData*-Objekt kann nun verwendet werden um die darin gespeicherten Audiodaten zu erkennen und in Text umzuwandeln. Das *SpeechRecognition*-Paket stellt dafür verschiedene Möglichkeiten zur Verfügung, wie eingangs erwähnt wurde. Diese sollen im Folgenden kurz beschrieben werden:

- Whisper: Whisper ist ein neuronales Netz das von der Firma *OpenAI* trainiert und als Open-Source-Projekt zur Verfügung gestellt wird [OpenAI o. D.[a],[b]]. Neben der Fähigkeit Sprache in Text zu konvertieren kann es auch eingesetzt werden um Transkripte zu generieren, die gesprochene Sprache automatisch zu erkennen oder in die englische Sprache zu übersetzen. Das Paket *SpeechRecognition* lässt sowohl die Verwendung der von *OpenAI* zur Verfügung gestellten Online-API zu, als auch die lokale Ausführung des Sprachmodells. Aufgrund der Anforderung nach Offline-Funktionalität entfällt die erste Möglichkeit (s. Kapitel 2).

- Sphinx: Das *CMUSphinx* Projekt wird von der Carnegie Mellon University (CMU) unterhalten und stellt eine Reihe von Werkzeugen und Bibliotheken zur Spracherkennung zur Verfügung [Shmyrev o. D.] Darunter fallen bspw. *PocketSphinx*, *SphinxTrain* und *sphinx4*. Bei *PocketSphinx* handelt es sich um eine C-Bibliothek zur Spracherkennung, die auch innerhalb von Python verwendet werden kann. *SphinxTrain* hingegen stellt Ressourcen zum Trainieren eigener Modelle bereit. *sphinx4* ist das Java-Equivalent zu *PocketSphinx*. Somit ist für dieses Projekt nur *PocketSphinx* interessant. Leider unterstützt enthält das Paket standardmäßig nur ein Modell für die englische Sprache. Anderssprachige Modelle und weitere dazugehörige Ressourcen müssen mühsam aus externen Quellen bezogen und eingebunden werden.

- Snowboy:

- Vosk:
- Google: *SpeechRecognition* bietet zwei Möglichkeiten Google-Dienste zur Konvertierung von Sprache zu Text zu verwenden. Die erste Möglichkeit besteht in der Verwendung der *Google Speech API* in der *Google Cloud* [Google o. D.] Die zweite Möglichkeit besteht in der Verwendung der *Google Speech API* ohne die *Google Cloud*. Die Verwendung der Google-Schnittstelle entfällt jedoch für dieses Projekt, da eine aktive Internetverbindung Voraussetzung dafür ist. Außerdem sind die Google-Dienste weder kostenfrei noch open-source.
- Microsoft: Auch hier gibt es zwei verschiedene Möglichkeiten Microsoft-Dienste zur Umwandlung von Sprache zu Text über das *SpeechRecognition*-Paket zu nutzen. Eine davon ist die Verwendung der *Speech to text* Funktion in der *Microsoft Azure Cloud* [Microsoft o. D.] Hierfür sind ein aktiver *Azure*-Account sowie ein gültiger API-Schlüssel notwendig. Die zweite Möglichkeit besteht in der *Microsoft Bing Voice Recognition*. Diese ist jedoch veraltet und wird nicht mehr unterstützt. Auch diese Schnittstelle enfällt jedoch wegen der notwendigen Internetverbindung und Bezahlung.
- IBM: Auch die Firma IBM bietet ihre eigene Schnittstelle zur Spracherkennung mit Hilfe der von IBM entwickelten KI *Watson* an [IBM 2023]. Auch hier gilt: zur Verwendung der API ist sowohl eine aktive Internetverbindung als auch Bezahlung vorgesehen, weshalb diese Schnittstelle für das Projekt nicht in Frage kommt.
- Weitere Schnittstellen sind *Wit.ai*, *Houndify* und *Tensorflow* [SoundHound o. D.; TensorFlow o. D.[a]; Wit.ai o. D.] Da diese Schnittstellen allderdings ebenso eine aktive Internetverbindung voraussetzen und damit ausscheiden soll nicht weiter auf sie eingegangen werden.

Die einzelnen APIs lassen sich jeweils über einen Methodenaufruf der Form *Recognizer.recognize_x* verwenden, wobei das „x“ für den Namen der jeweiligen API steht. Für dieses Projekt viel die Wahl auf die Verwendung der *Whisper*-Bibliothek von *OpenAI*. Dies ist damit zu begründen, dass sie von den offline verwendbaren APIs die mit Abstand am einfachsten zu verwendende ist und gleichzeitig sehr gute Ergebnisse beim Testen damit erzielt wurden. Beispielsweise werden die Modelle für viele verschiedene Sprachen bereits von OpenAI zur Verfügung gestellt und für jedes Modell stehen weitere Ausführungen zur Verfügung die nach den eigenen Ansprüchen und vorhandenen Ressourcen ausgewählt werden können. Die verschiedenen Ausführungen sind nach der „Größe“ des Modells unterteilt in „tiny“, „base“, „small“, „medium“ und „large“ [OpenAI 2023].

```
1 import speech_recognition as sr

3 r = sr.Recognizer()
4 m = sr.Microphone()
```

```
6  with m as source:
7      print("Start listening...")
8      audio = r.listen(source)
9      text = ""
10     try:
11         recognized_text = r.recognize_whisper(audio, language="german", model="tiny")
12         text = recognized_text
13         print("Recognized text: " + text)
14     except sr.UnknownValueError:
15         print("Whisper could not understand audio.")
16     except sr.RequestError as e:
17         print("Could not request results from Whisper; {0}".format(e))
```

Listing 5.2: Sprache zu Text mit *OpenAI Whisper*

Listing 5.2 zeigt, wie mit Hilfe der *Whisper*-API die aufgenommenen Audiodaten zu Text verarbeitet und ausgegeben werden können. Die Sprache und die Modellgröße werden bei dem Methodenaufruf *Recognizer.recognize_whisper* angegeben. Sollte das entsprechende Modell noch nicht lokal vorliegen wird dieses bei der ersten Ausführung automatisch installiert, was einen hohen Grad an Benutzerfreundlichkeit seitens der API bedeutet.

Herausforderungen bei der Implementierung der Spracherkennung

Installation von Whisper bzw. PyTorch

5.2.2 Anbindung des Sprachmodells an die Mischmaschine

5.2.3 Befehlsverarbeitung in der Mischmaschine

Kapitel 6

Fazit und Ausblick

TODO

Literatur

AIML Foundation [o. D.] URL: http://www.aiml.foundation/doc.html [besucht am 12. 03. 2023] [siehe S. 15].

Alammar, Jay [o. D.] *Sequence-to-Sequence Models for Chatbots* [siehe S. 20].

amazon [o. D.] *Amazon Lex: How It Works - Amazon Lex*. URL: https://docs.aws.amazon.com/lex/latest/dg/how-it-works.html [besucht am 12. 03. 2023] [siehe S. 18].

Chawla, Sumit [o. D.] *Building Chatbots with Google Dialogflow: Create chatbots with Dialogflow's natural language processing and machine learning capabilities* [siehe S. 18].

Diana, Perez-Marin und Pascual-Nieto Ismael [Juni 2011]. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices*. en. Google-Books-ID: 2nUcqtbCOBcC. IGI Global. isbn: 978-1-60960-618-3 [siehe S. 15, 17].

Google [o. D.] *Speech-to-Text: Automatic Speech Recognition*. en. URL: https://cloud.google.com/speech-to-text [besucht am 26. 03. 2023] [siehe S. 4, 32].

Götz, Felix, Moritz Höckele und Florian Lobert [2022]. *Mischmaschine*. Projektarbeit Software des Studiengangs Mechatronik an der DHBW Karlsruhe [siehe S. 1, 6].

Herczeg, Michael [März 2018]. »9. Zeitverhalten interaktiver Systeme«. de. In: *9. Zeitverhalten interaktiver Systeme*. De Gruyter Oldenbourg, S. 173–182. isbn: 978-3-11-044686-9. doi: 10.1515/9783110446869-187. URL: https://www.degruyter.com/document/doi/10.1515/9783110446869-187/html [besucht am 26. 03. 2023] [siehe S. 3].

IBM [März 2023]. *IBM Watson Speech to Text - Overview*. en-us. URL: https://www.ibm.com/cloud/watson-speech-to-text [besucht am 02. 04. 2023] [siehe S. 32].

Jansen, Jonas [Feb. 2017]. »Digitalisierung: 8,4 Milliarden vernetzte Geräte im Internet der Dinge«. de. In: *FAZ.NET*. issn: 0174-4909. URL: https://www.faz.net/aktuell/wirtschaft/netzwirtschaft/digitalisierung-8-4-milliarden-vernetzte-geraete-im-internet-der-dinge-14865654.html [besucht am 15. 03. 2023] [siehe S. 1].

Jurafsky, Dan u. a. [2009]. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition / Daniel Jurafsky (Stanford*

University), James H. Martin (University of Colorado at Boulder). eng. ISBN: 978-0-13-504196-3 [siehe S. 8].

Lane, Hobson, Cole Howard und Hannes Max Hapke [2019]. *Natural language processing in action: understanding, analyzing, and generating text with Python / Hobson Lane, Cole Howard, Hannes Max Hapke*. eng. ISBN: 978-1-61729-463-1 [siehe S. 9, 14].

Li, Jiwei u. a. [Sep. 2017]. *Adversarial Learning for Neural Dialogue Generation*. arXiv:1701.06547 [cs]. DOI: 10.48550/arXiv.1701.06547. URL: http://arxiv.org/abs/1701.06547 [besucht am 12. 03. 2023] [siehe S. 19].

Ltd, Raspberry Pi [o. D.] *Raspberry Pi 4 Model B specifications*. en-GB. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ [besucht am 24. 03. 2023] [siehe S. 5].

Marwedel, Peter [2021]. *Eingebettete Systeme: Grundlagen Eingebetteter Systeme in Cyber-Physikalischen Systemen*. de. Wiesbaden: Springer Fachmedien. ISBN: 978-3-658-33436-9 978-3-658-33437-6. DOI: 10.1007/978-3-658-33437-6. URL: https://link.springer.com/10.1007/978-3-658-33437-6 [besucht am 15. 03. 2023] [siehe S. 1].

Microsoft [o. D.] *Speech to Text – Audio to Text Translation | Microsoft Azure*. en-US. URL: https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text [besucht am 02. 04. 2023] [siehe S. 32].

Nupur Sharma, Anupriya [o. D.] *Chatbot Development using Machine Learning Techniques* [siehe S. 19].

OpenAI [o. D.[a]]. *About*. en-US. URL: https://openai.com/about [besucht am 01. 04. 2023] [siehe S. 31].

— [o. D.[b]]. *Introducing Whisper*. en-US. URL: https://openai.com/research/whisper [besucht am 01. 04. 2023] [siehe S. 31].

— [Apr. 2023]. *Whisper*. original-date: 2022-09-16T20:02:54Z. URL: https://github.com/openai/whisper [besucht am 02. 04. 2023] [siehe S. 32].

scikit-learn [o. D.] *scikit-learn documentation — DevDocs*. en. URL: https://devdocs.io/scikit_learn/ [besucht am 12. 03. 2023] [siehe S. 17].

Shmyrev, Nickolay [o. D.] *About CMUSphinx*. URL: http://cmusphinx.github.io/wiki/about/ [besucht am 02. 04. 2023] [siehe S. 31].

SoundHound [o. D.] *Home*. en-US. URL: https://www.soundhound.com/ [besucht am 02. 04. 2023] [siehe S. 32].

TensorFlow [o. D.[a]]. *TensorFlow*. en. URL: https://www.tensorflow.org/ [besucht am 02. 04. 2023] [siehe S. 32].

TensorFlow [o. D.[b]]. *TensorFlow - Sequence-to-Sequence Models*. URL: `https://chromium.googlesource.com/external/github.com/tensorflow/tensorflow/+/r0.7/tensorflow/g3doc/tutorials/seq2seq/index.md` [besucht am 12. 03. 2023] [siehe S. 20].

Wit.ai [o. D.] *Wit.ai*. URL: `https://wit.ai/` [besucht am 02. 04. 2023] [siehe S. 32].

Woudenberg, Aswin van [2014]. »A Chatbot Dialogue Manager-Chatbots and Dialogue Systems: A Hybrid Approach«. Magisterarb. Open Universiteit Nederland [siehe S. 14].

Zhang (Uberi), Anthony [o. D.] *SpeechRecognition: Library for performing speech recognition, with support for several engines and APIs, online and offline.* URL: `https://github.com/Uberi/speech_recognition#readme` [besucht am 30. 03. 2023] [siehe S. 30].

Liste der ToDo's